Modal Logics of Ordered Trees

Ulrich Endriss

Modal Logics of Ordered Trees

Ulrich Endriss

2003

A thesis submitted to the University of London
for the degree of Doctor of Philosophy

King's College London
Department of Computer Science

Revised Edition (March 2003)

Abstract

This thesis makes a contribution to the area of modal and temporal logics, an important family of non-classical logics that have found numerous applications in computer science and several other disciplines. Temporal logics, in particular, have been applied very successfully in the area of systems specification and verification. However, these logics often do not support the notion of refinement in a natural manner. There is no simple way to extend a given system specification, expressed in a standard temporal logic such as, for instance, propositional linear temporal logic, by the specification of a new subsystem. It is therefore important to extend available formalisms in ways that allow for the representation of complex systems evolving over time in a modular fashion.

To this end, we propose to equip propositional linear temporal logics with a 'zoom feature' in the following manner. If we extend the class of modal frames of this logic by adding a vertical accessibility relation to connect every time point with a new time line, we obtain a tree-like structure where the children of each node form a linear order. More precisely, we obtain a modal logic with frames that are ordered trees. The horizontal accessibility relation is still given a temporal interpretation, while the vertical allows us to move between different levels of abstraction, that is, between a system and its subsystems. Our logic provides modal operators working both along the branches of a tree and along the order declared over the children of a node.

The body of this work is devoted to the theoretical study of this modal logic of ordered trees. In particular, we are concerned with an axiomatic characterisation of our logic complementing its semantics-driven definition and with proving its decidability.

Acknowledgements

Many people have, in one way or another, played their part in the genesis of this thesis. Individual contributions range from very concrete support with my scientific endeavours to just being nice and making my years as a PhD student a very enjoyable time. In what follows, I shall mostly concentrate on the former type of contribution, but hope everyone else will get the message as well.

First and foremost, I should like to thank Dov Gabbay, my thesis supervisor, for his encouragement and support throughout my studies, for being a great inspiration from Day One, for letting me do things my way, and for always sharing his insights on logic and people. The Group of Logic and Computation at King's College (now the Group of Logic, Language and Computation) has provided the perfect environment for my studies. Everyone in the group, our numerous visitors, and many others in the Department of Computer Science have contributed to a very pleasant and stimulating working atmosphere. This includes, in particular, my first generation of office mates in 542: Stefan Schlobach, Odinaldo Rodrigues, and Wilfried Meyer-Viol. I'm also indebted to Hans Jürgen Ohlbach for his help and support whilst being my second thesis supervisor at the beginning of my PhD studies, before he moved on to Munich.

Stefan Schlobach, Wilfried Meyer-Viol, and Raquel Fernández have read large parts of various earlier incarnations of this thesis and their feedback has substantially improved the final product. Thanks are also due to Misha Zakharyaschev, Agi Kurucz, and Valentin Shehtman for answering a plethora of questions on modal logic; and to the regulars at our logic reading group: David Gabelaia, Roman Kontchakov, Corinna Elsenbroich, and George Metcalfe. I have also greatly benefited from discussing my work with the many people I met along the way, in particular Martina Kullmann, Carsten Lutz, Enrico Franconi, Alessandro Artale, Anuj Dawar, and Ian Hodkinson. Naza Aguirre deserves a special mention for showing me that the software engineering people still need a lot more temporal logic. Finally, I would like to thank Howard Barringer and Robin Hirsch for having agreed to examine this work.

On a more personal note, this seems like a good place to acknowledge the part my parents, Micha and Moni Endriß, have had in all of this.

London, January 2003 U.E.

The research reported here has been generously funded by the UK Engineering and Physical Sciences Research Council (EPSRC) through a number of projects: GR/L91818/01 (grant holders: D. Gabbay and H. J. Ohlbach), GR/R45369/01 (M. Zakharyaschev, D. Gabbay, and I. Hodkinson), and GR/N23028/01 (D. Gabbay), all at King's College London. The writing-up of the thesis was completed while the author was employed at Imperial College London on a research project funded by the European Union under grant reference number IST-2001-32530 (grant holders at Imperial: F. Toni and F. Sadri).

Contents

1 Zooming in **1**
- 1.1 Adding a Zoom to Linear Temporal Logic . . . 1
- 1.2 Thesis Overview . . . 10

2 Modal and Temporal Logics **13**
- 2.1 Introducing Modalities . . . 13
- 2.2 Possible World Semantics . . . 18
- 2.3 Axioms and Correspondences . . . 24
- 2.4 Temporal Logic . . . 31
- 2.5 Bibliographic Notes . . . 37

3 Semantic Characterisation **39**
- 3.1 Syntax and Semantics . . . 39
- 3.2 Some Defined Operators . . . 48
- 3.3 Correspondences . . . 50
- 3.4 Ontological Considerations . . . 55
- 3.5 General Models and P-morphisms . . . 60
- 3.6 Loosely Ordered Trees . . . 67
- 3.7 Summary . . . 94

4 Axiomatic Characterisation **97**
- 4.1 Axioms and Rules . . . 97
- 4.2 Some Derived Theorems . . . 108
- 4.3 Completeness without Descendant Modalities . . . 113
- 4.4 Alternatives . . . 141
- 4.5 Summary . . . 145

5 Decidability **147**
- 5.1 Monadic Second-order Logic . . . 147
- 5.2 Finite Binary Trees . . . 159
- 5.3 Discretely Ordered Trees . . . 163
- 5.4 General Ordered Trees . . . 169
- 5.5 Summary . . . 182

6 Zooming out **185**
6.1 Extensions 185
6.2 Summary of Main Results 192
6.3 Open Problems 193

Appendices **197**

A Relations and Orders **199**
A.1 Relations 199
A.2 Linear Orders 200

B Derivations **203**
B.1 Proof of Lemma 4.8 203

Bibliography **205**

List of Figures **211**

List of Tables **213**

Index **215**

Chapter 1

Zooming in

In this thesis we develop a new modal logic to represent complex systems evolving over time in a modular fashion. As will be explained in this chapter, this new logic may be regarded as the result of extending well-known linear temporal logic by a second dimension that allows us to *zoom* into states and thereby model events associated with them in more detail. In this sense, our logic may be described as an *extended temporal logic*. From a more abstract point of view, our logic is best characterised as a *modal logic of ordered trees*.

In this first chapter we motivate the enterprise of developing such a logic in the first place and give an informal outline of some of the main results reported in the sequel. The final section also provides a compact overview of the thesis as a whole, organised by chapters.

1.1 Adding a Zoom to Linear Temporal Logic

Why? What do we mean by adding a zoom to temporal logic and why would anyone want to do something like this? We begin our explanation with a story, the story of Mary. Mary was an aspiring young software engineer who was working for an up-and-coming information technology company offering state-of-the-art software solutions to its clients and profitable revenue to its shareholders. One day, not that long ago, the company's young and dynamic marketing department managed to secure a major contract with a new heavy-weight client. It was their biggest software development contract yet and a lot (if not everything) depended on its successful completion. Mary was appointed Chief Engineer on the project.

Most people would have been intimidated by that much responsibility. Not Mary though. She was not only a true professional, but had also benefited from an excellent educational system which had provided her with a very respectable academic background. Mary had completed a degree in Computer Science at one of the country's finest universities. She was fluent in all the major programming languages, knew her computer inside out, and commanded an intimate knowledge of all phases of the software development process, be it design, implementation, or evaluation of the final product. She was even good at maths. But was all of this going to be enough for this new and highly critical project? The central question seemed to be: How do you make sure the system will really do what it is supposed to?

Mary was lucky. In her final year at university, besides all the programming languages, software engineering, databases, and computer architecture, she had also attended a class on non-classical logics, which was at the time taught by a very famous logic professor. In fact, she probably did not even realise just how lucky she had been, because most of the time the famous professor was tied up in research, which regularly involved writing books, proving ground-breaking new results, and even developing whole new schools of thought. But this particular year he *did* teach and Mary enjoyed the course very much. One of the subjects that made a lasting impression on her was the field of temporal logics. She could remember it very well. Temporal logics were astonishingly simple, yet extremely powerful; full of mathematical depth and beauty, but also — as she had realised by now — of immediate industrial relevance. Several of the big companies leading the market were already using this exciting new technology to help them develop provably correct software systems. Mary knew she had to find out more about temporal logic, and she had to do so quickly. This she owed to her company's shareholders.

Temporal logic. So what is temporal logic? In short, a temporal logic is a formal system to reason about propositions that change their truth value over time. In the context of designing reliable software artefacts, examples for propositions would be sentences such as 'the variable x currently has value 542' or 'the file *critical.xml* is currently being written on by process 4711'. A proposition may be either *true* or *false* at any given point in time.

Naturally, the notion of *time* itself will be central to our understanding of temporal logics. A simple yet useful model of time would be the following:

$$\begin{array}{ccccccccccc} \circ & \longrightarrow & \bullet & \longrightarrow & \bullet & \longrightarrow & \bullet & \longrightarrow & \bullet & \longrightarrow & \bullet \cdots\cdots \\ t_0 & & t_1 & & t_2 & & t_3 & & t_4 & & t_5 \end{array}$$

That is, we can think of time as a sequence of time points or *states*. The first one, t_0, denotes the beginning of time and after that we have one state for each and every natural number. Of course, this is a very simplified if not artificial model of time. It does, for instance, not take into account special relativity theory or the latest findings in quantum physics. But for the basis of a logic intended to describe a sequence of events, such as those taking place while a complex software system is being executed, this seems quite appropriate.[1]

[1] We are painting a somewhat simplified picture of the field of temporal logics here. The particular logic we are in the process of describing is known as *propositional linear temporal logic*. In fact, at this point we restrict our attention to the future-fragment of propositional linear temporal logic over discrete flows of time and we shall not consider the so-called *until*-operator either. A fuller account of the area of (modal and) temporal logics is given in Chapter 2.

Let φ and ψ be propositions that are either *true* or *false* at different states in our model. We can use operations familiar from propositional logic such as negation or conjunction to build more complex propositions (like, for instance, $\neg\varphi$ or $\varphi \wedge \psi$). Such compound propositions are intended to have the following meaning:

$\neg\varphi$: The formula φ is *not* true at this state (negation).
$\varphi \wedge \psi$: Both φ *and* ψ are true at this state (conjunction).
$\varphi \vee \psi$: Either φ *or* ψ (or both) are true at this state (disjunction).
$\varphi \rightarrow \psi$: *If* φ is true at this state *then* so is ψ (implication).

So far, we are only doing propositional logic. None of these operators allows us to refer to states in our model other than the one we have been at in the first place. To this end, temporal logic provides a number of *modal operators*. Three of the most important ones are the so-called *diamond-operator* $\Diamond$, the *box-operator* $\Box$, and the *next-operator* $\bigcirc$. They have the following meaning:

$\Diamond\varphi$: The formula φ will be true *sometime* in the future.
$\Box\varphi$: The formula φ will be true *always* in the future.
$\bigcirc\varphi$: The formula φ will be true at the *next* state.

That is, $\Diamond\varphi$ is defined to be true at, say, state t_2 iff φ is true at *some* (at least one) of the states succeeding t_2 (i.e. t_3, t_4, t_5, ...). The formula $\Box\varphi$ is true at t_2 iff φ holds at *all* succeeding states. Finally, $\bigcirc\varphi$ holds at t_2 iff φ is true at its immediate successor t_3.

Examples. This rather simple language already allows us to express a number of interesting properties of a system evolving over time. Let us look at two examples. Suppose we want to specify a set of rules governing a system's access to a particular file. Let *OpenFile* be a basic proposition that is true whenever the system is currently opening the file in question for writing access and let *CloseFile* be a proposition that is true whenever the system is closing that file again. Then the following formula expresses a very simple property of the system:

$$\Box(\mathit{OpenFile} \rightarrow \Diamond\, \mathit{CloseFile})$$

This formula is true at the first state of the model corresponding to the system iff, whenever the file is being opened at some state t (after the first state), then there must be a state t' lying in the future of t at which the file is being closed again.

By its very (young, dynamic, and innovative) nature, Mary's company offers its clients advanced e-commerce solutions as an optional extra. Such a system may, for instance, be capable of automatically negotiating prices for goods it intends to purchase on-line. The process of agreeing on a price may be modelled as a sequence of proposals and counter proposals sent back and forth between two trading partners. Suppose now, our system receives such a proposal and suppose we want to specify the set of admissible reactions by means of a temporal logic formula. A particular negotiation protocol may allow for three possible follow-ups. Firstly, the system may simply accept the proposal it received; secondly, it may reply with a suitable counter proposal; and thirdly, it may choose to terminate the negotiation thread (for instance, if it seems impossible to come to an agreement). Here is the corresponding formula:

$$\Box(\mathit{ReceiveProposal} \rightarrow \bigcirc(\mathit{AcceptProposal} \vee \mathit{SendCounterProposal} \vee \mathit{EndNegotiation}))$$

To emphasise that these are the system's options for its *next* move (rather than *some* future move) we now use the next-operator $\bigcirc$ in place of the diamond-operator $\Diamond$ from the (otherwise very similar) previous example.

Of course, this is a very abstract specification of what needs to be done when the system receives a particular proposal. For instance, the process of computing a suitable counter proposal (where applicable) will usually be rather complex a task. In fact, the subtask of computing such a counter proposal should probably itself be covered by a suitable temporal logic specification.

The problem of modularity. At this point, Mary noticed a problem. The sample formula for our e-commerce application above suggests that we should choose a model of time where each time unit corresponds to one turn in the negotiation dialogue between the system and its trading partner. This is fine as far as specifying the negotiation protocol in operation is concerned. However, to specify the system's internal calculations, pertaining to, for instance, the evaluation of an incoming proposal, a finer time granularity is required. But how do you put the two together in a single model? In other words, how do you combine the specifications of different modules to form the overall specification of a complex system?

John, one of the programmers working in the basement, was quick to suggest a solution: "We could simply refine the temporal model of our system by adding the required number of states in between the state representing the point in time where the system receives an incoming proposal and the state where it reacts to that proposal." Mary immediately realised that this was certainly not a workable solution: "But then any formula starting with a next-operator that was previously true at the state where we receive a proposal will not necessarily remain to hold, because the state satisfying its witness formula will not be the immediate successor anymore." John replied that this may well be an indication that one should not use the next-operator in the first place. Maybe the concept of a 'next' point in time is not that appropriate after all and one should use the diamond-operator instead? Mary strongly disagreed with this view. She insisted on using the next-operator: "The next-operator very accurately refers to the next turn in the dialogue between the two communicating systems. We should definitely keep it." But John liked his pragmatic idea of inserting additional states into a model and did not want to give up that quickly: "We could add a few next-operators at the beginning of each problematic formula, one for every additional state ..." Mary could not help interrupting: "Sorry, John, but if we start using dirty tricks like that, then we might as well not even bother using formal methods in the first place." — "Oh, forget it then!" — "Hacker!" — "B**ch!"

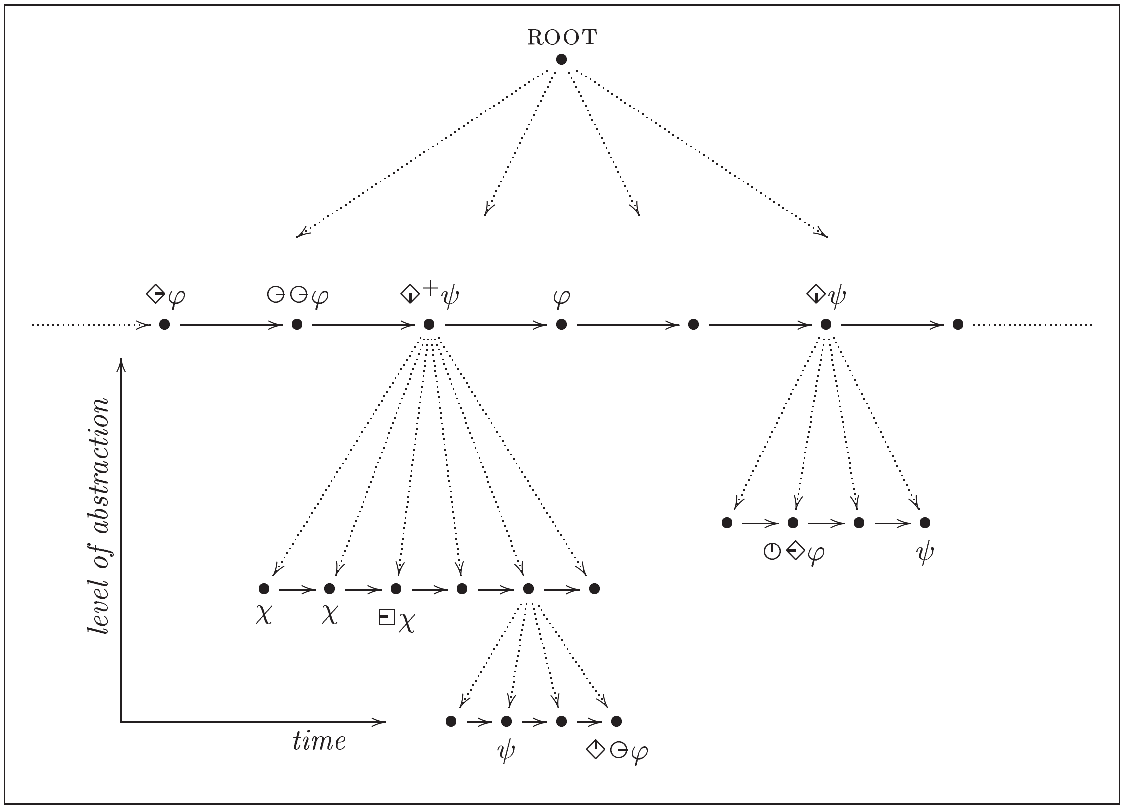

Figure 1.1: Adding a zoom to linear temporal logic

This is where we leave Mary and her problem. I do not know whether, eventually, she found the kind of simple, elegant, and workable solution she was looking for. In what follows, I propose my own ideas.

Zooming in. Apart from the mere technical difficulties associated with the naïve method of refining a given temporal logic model by simply inserting a few extra states (particularly where next-operators are involved), such an approach also suffers from a rather more conceptual deficiency. Presumably, the process of refinement will often follow some modular structure of the system to be specified. We may start out with a very abstract description of the system as a whole and then, step by step, refine its modules and submodules. It would be nice to see this modular structure of a system also reflected in the final model used for the verification of a specification.

This is where our central idea comes into the picture: we propose to *add a zoom to linear temporal logic.* Instead of inserting additional states into the single time line of a linear temporal logic model, we propose to explicitly relate the state to be refined to another time line which represents the course of events taking place 'during' that 'state' (or rather the time interval associated with that state) at the next lower level of abstraction. This relation may be interpreted as the process of 'zooming' into an abstract state of the main time line. The new states may themselves be refined, that is, we may also zoom into the states on the second level of abstraction, and so forth. This idea is illustrated by the model shown in Figure 1.1.

In the model of time underlying this new logic, 'states' may be regarded as representing time *periods* (rather than points). Our 'zoom relation' connects subsequent levels of abstraction and indicates how these periods are to be decomposed into smaller units at the next level. In general, we make no specific assumptions on the nature of either the main time line or any of the time lines associated with a lower levels of abstraction. They may be discrete (as was the case for the basic linear temporal logic) and possibly even finite (if we only want to decompose a time period into a fixed number of smaller ones), but they may also be infinite and dense.

Modal operators. Figure 1.1 also shows examples for some of the modal operators we propose to include into our new temporal language. The first set of operators is taken from the basic temporal logic we have seen earlier. It includes an existential diamond-operator and a universal box-operator to refer to states in the future at the same level of abstraction, as well as a next-operator to refer to the state immediately following the current one (in case there is such a state), also without changing to another level of abstraction. Here is an informal definition of the semantics of these operators:

$\overrightarrow{\Diamond}\varphi$: φ is true at *some future* state (at the same level of abstraction).
$\overrightarrow{\Box}\varphi$: φ is true at *all future* states (at the same level of abstraction).
$\overrightarrow{\bigcirc}\varphi$: φ is true at the *next* state (at the same level of abstraction).

The next set of operators serves a similar purpose; this time to refer to states lying in the past of the current one:

$\overleftarrow{\Diamond}\varphi$: φ is true at *some past* state (at the same level of abstraction).
$\overleftarrow{\Box}\varphi$: φ is true at *all past* states (at the same level of abstraction).
$\overleftarrow{\bigcirc}\varphi$: φ is true at the *previous* state (at the same level of abstraction).

We now turn to the new *vertical* operators. There are two operators to refer to the states on the next lower level of abstraction, i.e. to the states representing the decomposition of the current state. Observe that it would not make sense to include a next-operator into this group of connectives, because there will (usually) not be a unique immediate successor with respect to the zoom relation underlying these operators. The diamond- and the box-operator to change to the next lower level of abstraction are defined as follows:

$\Diamond_{\downarrow}\varphi$: φ is true at *some* state of the *next lower* level of abstraction.
$\Box_{\downarrow}\varphi$: φ is true at *all* states of the *next lower* level of abstraction.

If we do not just want to refer to states on the *next* lower level, but to states on any of the levels below the current one, i.e. to any of the states that can be reached by following the (transitive closure of the) zoom relation from the current state, we may use the following two modal operators:

$\Diamond_{\downarrow}^{+}\varphi$: φ is true at *some* state of *some lower* level of abstraction.
$\Box_{\downarrow}^{+}\varphi$: φ is true at *all* states of *all lower* levels of abstraction.

Finally, to be able to move back up again within the hierarchy of states, we define the following set of operators:

$\Diamond_{\uparrow}\varphi$: φ is true at *some higher* level of abstraction.
$\Box_{\uparrow}\varphi$: φ is true at *all higher* levels of abstraction.
$\bigcirc_{\uparrow}\varphi$: φ is true at the *next higher* level of abstraction.

When moving up, the concept of a next-operator does make sense again, because for any state there is a unique state that it represents an immediate refinement of (unless we are already at the highest level of abstraction).

A formal definition of the semantics of these operators will be given in Chapter 3.

Modal logics of ordered trees. Linear temporal logics are based on linear orders. What is the structure underlying our extended temporal logic? If we add a 'first level of abstraction' consisting of only a single state (which may be thought of as representing the specified system as a whole) to the kind of model we have described so far, we end up with a tree-like structure. The single state at the top would then be the root of that tree (as indicated also in Figure 1.1). The children of any node in such a tree are ordered, that is, the structure underlying our logic is in fact an *ordered tree.* Hence, from an abstract point of view, our extended temporal logic may be characterised as a *modal logic of ordered trees.*

Our logic provides modal operators working both along the branches of a tree, that is, with respect to the level-of-abstraction dimension, and along the order declared over the children of a node, that is, with respect to the temporal dimension. Given the terminology of ordered trees, we can now give a more succinct characterisation of these operators. The horizontal operators $\Box_{\rightarrow}$ and $\Diamond_{\rightarrow}$ refer to the *righthand siblings* of a node, while $\bigcirc_{\rightarrow}$ points to the immediate *righthand neighbour* (if any). Similarly, $\Box_{\leftarrow}$ and $\Diamond_{\leftarrow}$ may be used to describe nodes that are *lefthand siblings* and $\bigcirc_{\leftarrow}$ refers to a node's *lefthand neighbour* in the particular order declared over a node and its siblings. The *parent* modality $\bigcirc_{\uparrow}$ points to the node immediately above a given node, while $\Box_{\uparrow}$ and $\Diamond_{\uparrow}$ range over all its *ancestors.* The *children* of a node may be addressed using either $\Box_{\downarrow}$ or $\Diamond_{\downarrow}$, while $\Box_{\downarrow}^{+}$ and $\Diamond_{\downarrow}^{+}$ also reach *descendants* that are not necessarily immediate vertical successors of the reference node. On the whole, we end up with a new modal logic with a simple and intuitive semantics based on ordered trees that may also be interpreted as an extended temporal logic with a significant potential for important applications.

This thesis is devoted to the theoretical study of this new logic. The three main chapters are concerned with (1) a precise *semantic characterisation* of the logic, (2) a complementary *axiomatic characterisation*, and (3) a proof of its *decidability.* A more detailed overview is given in the next section.

Related modal and temporal logics. To a certain degree, our logic has the flavour of a temporal interval logic, because it allows us to represent the decomposition of primitive 'units of time' (by adding children to a given node in an ordered tree). Therefore, logics such as the modal interval logic of Halpern and Shoham [37] or Moszkowski's **ITL** [58] (Interval Temporal Logic) may be considered related formalisms. However, *technically* we are still working with (single) points. The time periods are just a useful interpretation. For the definition of the semantics of 'genuine' interval logics, on the contrary, we require reference to *two* points (namely the endpoints of an interval). Under this perspective, our logic may be classified as being simpler than the logics of either Halpern and Shoham or Moszkowski, which (provided it is still expressive enough for a particular application) is of course an advantage. As we shall see in Chapter 5, our logic is decidable, which is not the case for (the full versions) of the aforementioned interval logics. On the downside, a feature of temporal intervals that our logic cannot express is the concept of *overlapping* intervals. For further discussion on this and related points, we refer to Chapter 6, where we describe an embedding of a restricted interval logic similar to that of Halpern and Shoham into the modal logic of ordered trees.

Another logic that allows for the decomposition of time periods is the metric and layered temporal logic for reasoning about time granularities proposed by Montanari [56]. This logic is particularly suited for domains where reasoning about fixed time units and their relation to each other is required. A particular model may, for instance, include temporal domains representing days, hours, and minutes, respectively. The language of this temporal logic provides so-called contextualised modalities to refer to any one of these time granularities.

The logic that seems to come closest to our logic (that we are aware of) is probably the logic of finite binary trees of Blackburn and Meyer-Viol [9] (later generalised to a logic of finite trees [10]). Like ours, these logics are modal logics over frames that are trees and they provide a similar set of modal operators. Important differences include that the horizontal dimension in our logic need not be discrete and that branches may be of infinite length. As we shall see in later chapters, these differences are significant, both with respect to completeness and decidability results. Apart from such technical differences, our interpretation of this kind of logic as an extended temporal logic (with time running orthogonally to the tree structure, i.e. from left to right) is also new.

Related work on temporal logic specification. The application of temporal logics to the specification of (reactive) systems started with the publication of Pnueli's influential paper in 1977 [60].[2] Since then a vast amount of literature has been published on this subject and, more generally, in the area of applied temporal logics in software engineering. We shall make no attempt of providing a comprehensive review here, and are only

[2]Reactive systems (as opposed to transformational systems that compute an output for a given set of inputs) are typically non-terminating and characterised by a high degree of interaction with their environment. Examples include operating systems or communication protocols.

going to briefly mention a selection of works that are directly relevant to the particular problem we have addressed here, namely the *refinement* of temporal logic specifications.

In our example in the first section, the problem of not being able to refine a given specification was triggered by the use of the next-operator (rather than the less problematic diamond). Lamport [47] argues against the inclusion of a next-operator into the temporal language altogether, on the grounds that it would add unnecessary and unwanted expressive power. He argues that, while the next-operator refers to the next point in time with respect to some (irrelevant) global clock, the truly interesting 'next' state would be the first state in the future where a change in the truth values of the (relevant) atomic propositions can be observed. As we have seen earlier, some of the technical problems of refining a given specification by means of adding additional states can be avoided if we do not allow for any next-operators in our formulas. Barringer, Kuiper, and Pnueli [6] go one step further by proposing a temporal logic based on the reals rather than the natural numbers. This seems the more appropriate model of time for a language without next-operators. However, other authors, such as Fiadeiro and Maibaum [23] for instance, argue that the next-operator is a very intuitive construct for describing the transition from one state to the next, which can greatly simplify the development of a specification. It seems therefore desirable not to completely abandon the next-operator.

In a different paper, Barringer *et al.* [5] have put forward a proposal for a temporal logic that *does* include a next-operator and where some of the difficulties of combining specifications are overcome in a different manner. This logic, which is based on a discrete model of time, not only includes propositions that may or may not be true at any given state in a model, but it also evaluates propositions over the set of edges between successive states. These so-called edge propositions allow for an explicit distinction between transitions performed by that module and transitions performed by the environment. It is then possible to specify conditions under which two given modules can be combined (because they do not interfere with each other).

While the frameworks proposed both by Lamport and by Barringer *et al.* allow for the combination of program modules, the resulting model does not reflect anymore *how* it has been put together, i.e. it does not have a modular structure. Fiadeiro and Maibaum [23] have addressed this problem by connecting different temporal logic models, each of which corresponds to a particular module and is assumed to have a time granularity appropriate for that model. This idea is not dissimilar to our zoom feature. However, Fiadeiro and Maibaum do not introduce a new logic, but instead concentrate on methods for transferring properties proved for one level of granularity onto the next level.

The works cited here all promote specific (temporal logic based) formalisms for the specification of reactive systems. At this point, we should emphasise that this is not at the centre of our own agenda. We have discussed problems with the refinement of specifications in this chapter, because they provide a good illustration for the kind of systems that can be modelled using our new modal logic. However, the main body of this thesis will be concerned with the theoretical study of our logic rather than its application to the specification of reactive systems.

1.2 Thesis Overview

The remainder of this thesis is structured as follows.

Chapter 2. This chapter provides a general introduction to the area of *modal and temporal logics*. It may be skipped by readers who are familiar with the basic concepts of the field.

Chapter 3. This chapter is mostly concerned with the *semantic characterisation* of our modal logic of ordered trees. The first section provides a formal definition of the class of ordered trees as well as a definition of the *syntax* and *semantics* of our logic, which we have only introduced informally so far.

The subsequent sections are aimed at demonstrating the *expressive power* of the logic. We first show how a number of powerful operators, such as the difference operator, may be defined within the logic. Another section discusses a number of *correspondence results*. For instance, it is possible to give axioms that characterise the class of all ordered trees for which the children of each node form a discrete order, or the class of all ordered trees with branches of finite length. In yet another sequence of examples we show how we can incorporate certain *ontological concepts*, such as the homogeneity of a proposition, into our system when it is interpreted as an extended temporal logic.

In the second half of Chapter 3 we first introduce *p-morphisms* as a means of studying the relation between models based on ordered trees and more general models. This leads us to the definition of a class of modal frames we call *loosely ordered trees*, which provide a sharper semantic characterisation of our logic than ordered trees. This result is also relevant to our completeness proof in Chapter 4.

Chapter 4. This chapter is concerned with an *axiomatic characterisation* of the modal logic of ordered trees. In the first section we put forward a set of *axioms* and *inference rules* for the logic and in the second section we prove the validity of a number of useful theorems that can be derived in this system.

Most of the chapter is devoted to proving *completeness* of the proposed axiomatic system with respect to the model-theoretic semantics given in Chapter 3. Our completeness result is restricted to the fragment of our logic excluding the transitive descendant operators. Towards the end of the chapter we briefly discuss various options how one may be able to extend this result to the full language.

Chapter 5. In this chapter we show that the modal logic of ordered trees is *decidable*. This result is achieved via an embedding of our logic into the decidable *monadic second-order theory of two successor functions*.

The first section of the chapter serves as an introduction to monadic second-order logic. In subsequent sections we give different embeddings to show decidability of our logic, first for two special cases and then for the logic in general. The two special cases are interesting logics in their own right. The first one is the modal logic of *finite binary trees* and the second one is the modal logic of *discretely ordered trees*.

Chapter 6. In the concluding chapter we propose an *extension* to the logic of ordered trees by adding a number of *until*-style operators to the language. The subsequent discussion includes the outline of an embedding of a (restricted) *interval logic* into this extended modal logic of ordered trees.

Finally, we give a brief summary of the main results obtained and make suggestions for future work in the area of modal logics of ordered trees.

Appendices. Chapter 6 is followed by two appendices, a bibliography of the literature referred to in the text, and a detailed subject index. Appendix A discusses some of the basic definitions on *relations* and *linear orders*, which play a central role in this work. A number of *sample derivations* in the axiomatic proof system developed in Chapter 4 are given in Appendix B.

Chapter 2

Modal and Temporal Logics

In this chapter we give a brief introduction to the area of (propositional) modal and temporal logics. We start by introducing the basic modal language and explain how it may be given different interpretations to allow for reasoning about such different domains as knowledge, necessity, or time. We then discuss some of the fundamental concepts surrounding the study of basic modal logics, including possible world semantics, axiomatisations, correspondence theory, and the standard translation into first-order logic. Most of this material would typically be covered by an introductory textbook on modal logic. It is included here for the sake of completeness of our presentation. Following this general exposition, we briefly discuss propositional linear temporal logic and its application to the specification of reactive systems.

We assume familiarity with classical propositional logic, but not much else. In some isolated places, we also refer to classical first-order logic, but a deeper understanding of first-order logic is certainly not required to follow the main ideas of the text.

2.1 Introducing Modalities

Choosing the right logic. There is not just one logic; there are many. When choosing one to, say, help solve a problem in a computing application, it is crucial to pick the right one. The most widely known exponents of the (very broad) landscape of logics are classical propositional and first-order predicate logic. First-order logic, in particular, is very popular with computer scientists. Everybody knows it, everybody loves it, and you can say pretty much everything you like in the usual first-order language. But expressive power is not the only quality we should look for in a logic. Favourable computational behaviour, for instance, would be another one. And here, first-order logic fails us. As shown by Church in 1936 [16] (and independently by Turing [68]), the validity problem for formulas of first-order logic is *undecidable*; that is, there can be no algorithm that will tell us whether a given first-order formula is true under all possible interpretations of the symbols appearing in that formula. This is bad news, certainly if our interest in logic goes beyond using it merely as a representation language, but if we actually want to *reason* about the *meaning* of formulas. This is not to say that using first-order logic is a bad thing in general. On the contrary, there are many examples where it turns out to be precisely the right tool for the job. In other cases, however, full first-order logic is simply *too* powerful and we can benefit from choosing a less expressive but more manageable language.

The second most prominent exponent in the aforementioned landscape of logics is classical propositional logic, the logic of truth tables. This logic *is* decidable and has also a number of other advantages. Its semantics, for instance, is very simple. On the other hand, for many applications the language of *'not'*, *'and'*, and *'or'* is simply not expressive enough.

Much of the work in *non-classical logics* is concerned with the study of systems that (in some sense) lie between the two 'extremes' given by classical propositional and first-order logic.[1] The goal is to strike the right balance between computational *complexity* (the lower the better) on the one hand and *expressiveness* (just about high enough to be able to model the application domain in question) on the other. In Einstein's famous words: *"Everything should be made as simple as possible, but not simpler."* As for complexity, we take the view that *decidability* is a particularly desirable minimal requirement. Another desideratum for a 'good' logic would be that it enjoys a *simple and intuitive semantics*. That is, it should be easily understood what it means for a formula to be 'true'. Propositional modal logics are widely regarded as performing particularly well with respect to these criteria. They extend classical propositional logic with operators that permit us not only to distinguish between propositions that are either *true* or *false*, but to speak about different *modes of truth*. To see what we mean by this, let us first take a look at propositional logic and some of its limitations.

Propositional logic. In the language of classical propositional logic we can represent simple statements such as the following:

If it rains, then the grass gets wet.

This is a complex proposition, which is made up of two simpler ones and a logical connective. The first of these simpler propositions is *'it rains'*, the second is *'the grass gets (or simply: is) wet'*. The above statement connects these two basic propositions by expressing that the former (*'it rains'*) implies the latter (*'the grass is wet'*).

For practical reasons, when working with a formal logical system, we do not usually manipulate statements in natural language directly, but rather use symbolic representations of them. For example, let the propositional letter P stand for *'it rains'* and let Q stand for *'the grass is wet'*. Then our statement is represented by the implication $P \rightarrow Q$. Other connectives of propositional logic include negation ($\neg$), conjunction ($\wedge$),

[1] We should point out, that this is just *one* aspect of research into alternatives to classical logic. Other directions of research include the study of languages that are even more expressive than first order logic, others aim at devising less expressive systems that have particularly good computational properties. Others, again, are just *different*.

and disjunction ($\vee$). The formula $\neg P \wedge Q$, for instance, stands for the sentence *'it does not rain and the grass is wet'*. The semantics of propositional logic is very simple. Formulas may either be *true* or *false* and the truth table for the principal connective of a given formula determines the truth value of that formula given the truth values of its subformulas. The propositional letter P being true, for instance, implies that $P \vee Q$ must be true as well (whatever the value of Q).

Necessity. A formula like $\neg P \vee P$ must be true under any interpretation of the symbol P, that is, it is a logical tautology. In other words, the formula $\neg P \vee P$ is *necessarily* true. We may also wish to express that our original formula $P \rightarrow Q$ is necessarily true. It is certainly *not* for purely logical reasons (unlike $\neg P \vee P$), but provided P and Q are being interpreted as *'it rains'* and *'the grass is wet'*, respectively, then (at least in the context of the *weather forecast for planet Earth* domain) it makes sense to postulate necessary truth of $P \rightarrow Q$. (Maybe, just maybe, in a world very different from ours the grass does not get wet when it rains, but it seems reasonable not to include such a possibility into our formal model.)

Can we express this idea of necessary truth in propositional logic? What if we simply pick truth values for P and Q in such a way that $P \rightarrow Q$ evaluates to *true*? Inspection of the truth table for the implication operator $\rightarrow$ reveals that this would amount to considering a model where it is not the case that P has the truth value *true* and Q has the truth value *false*. That is, we would describe a world where it happens not to be the case that it rains and the grass stays dry at the same time. But this is conceptually different from what we wanted to express, namely that it is *necessarily* the case that rain implies wet grass. The latter means that there is no possible scenario where it rains without the grass getting wet in the process. The difference becomes even clearer when we compare this situation with the case of the simpler formula P. A model in which P has the truth value *true* describes a world in which it rains. This does *not* entail that it rains necessarily, that is, in *every* possible world.

So while *we* can distinguish *actual truth* from *necessary truth*, classical propositional logic is not rich enough a language to model this distinction.

Modes of truth. Actual truth and necessary truth are just two examples for the different *modes* of truth a proposition may take. To exemplify a few of them, let us move away from the world of rain and wet grass and consider a somewhat brighter subject: the Easter bunny. It seems safe to assume that, in our world, there is no such thing:

The Easter bunny does not exist.

This is a (negative) statement on the *actual* truth value of the basic proposition *'the Easter bunny exists'*. Let P stand for *'the Easter bunny exists'*. Then the above statement may be written (in propositional logic) as $\neg P$. However, $\neg P$ is not *necessarily* true, in the sense that we may well imagine a world where there *is* an Easter bunny:

It is possible that the Easter bunny exists.

Mixing symbolic notation and English for the moment, this statement may be abbreviated as *'possibly P'*. We can observe some kind of *duality* between the notions of necessity and possibility. What is possibly (or conceivably) true (like the existence of the Easter bunny) is not something that is necessarily false. Or the other way round: what is necessarily true cannot possibly be false. So we can distinguish two *alethic* modes of truth that are interrelated: necessary truth and possible truth.

Now, while not everyone would agree with the previous statement regarding the possibility of the existence of the Easter bunny, some people (most of them small children) undoubtedly *do* believe in it. So does John, for instance:

John believes that the Easter bunny exists.

In our semi-formal notation this may be represented as *'john-believes P'*. This does not contradict the fact that in our actual world the Easter bunny does not exist. Beliefs do not always match reality. The above statement is an example for the *doxastic* mode of truth a proposition may take.

There is an important difference between *beliefs* and *knowledge*. Something that is known (in the true sense of the word) needs to be actually the case. So to claim that John, Mary, or whoever, *knows* that the Easter bunny exists would be a lie. Still, Mary, for instance, may neither know about the existence nor the non-existence of the Easter bunny. That is, the following would be an admissible sentence regarding Mary's state of knowledge:

Mary considers it possible that the Easter bunny exists.

That is, assuming that the Easter bunny exists is not at odds with Mary's knowledge. So again, in analogy to the case of necessity and possibility, we can observe a duality between what Mary knows (what is necessarily true according to Mary) and what she considers plausible (what is possibly true according to Mary). These are *epistemic* modes of truth.

So far we have seen that the Easter bunny does not exist, that this is not necessarily so, that John believes otherwise, and that Mary's state of knowledge does not allow her to rule out the possibility of its existence. And if gene technology is worth even a fraction of the amounts of money invested into it every year, then surely:

At some point in the future, the Easter bunny will exist.

This is an example for a sentence where we are interested in the *temporal* mode of truth of our basic proposition P. We may denote this as *'sometime P'*. Dual to this, we may also make statements regarding propositions that are true *always* in the future.

The various modes of truth we have seen examples for are summarised in Table 2.1. We can distinguish four pairs of *modalities*, each of which gives rise to a different *modal logic*. First of all, there is alethic modal logic, the logic of necessity and possibility. Then

Logic	Universal Modality	Existential Modality
Alethic (necessity)	*necessarily*	*possibly*
Doxastic (belief)	*John believes*	*John considers it possible that*
Epistemic (knowledge)	*Mary knows*	*Mary considers it possible that*
Temporal (time)	*always in the future*	*at some point in the future*

Table 2.1: Modes of truth

there are doxastic logic and epistemic logic which allow us to reason about the beliefs and the knowledge, respectively, of a particular person (or agent). Finally, there is temporal logic to speak about propositions that change their truth values over time.[2]

Modalities have either *universal* or *existential* character. In the case of the two alethic modalities, *'possibly'* is an existential operator, because it postulates the *existence* of some scenario (or world) where the proposition in question holds. Necessary truth, on the other hand, is a notion with universal character, because it requires the respective proposition to be true in *every* possible scenario under consideration. The situation for epistemic and doxastic operators is similar. Something, say P, is possible according to, say, Mary iff there *exists* a state of affairs that she considers possible where P is satisfied. Mary *knows* P iff the proposition P is true in *every* situation she considers possible.[3] Finally, in temporal logic the universal modality *'always'* refers to *all* time points in the future, whereas *'sometime'* postulates the existence of at least one time point where a particular proposition holds.

Syntax of the basic modal language. We have now seen a number of examples demonstrating the diversity of domains modal logics are being applied to. We have also seen that these very different domains have more in common than one would probably have expected at first. In each of the four domains we considered, we observed the need to introduce 'operators' of both universal and existential character. And indeed, a single (basic) formal language is sufficient to work in all four of these domains (and many more). We are now going to define this language, which is an extension of the language of classical propositional logic, before we are going to move on to discuss its semantics in the next section. In short, we obtain the basic modal language by adding two new operators to propositional logic: $\Box$ (pronounced *box*) and $\Diamond$ (pronounced *diamond*). As we shall see later on, box-operators are used to represent universal modalities, while

[2] This is not an exhaustive list. There are many other modal logics, for different application domains, and there are also several different variants of those that we have mentioned here.

[3] At this point it is interesting to note that the English language does not really allow for a simple and clear distinction between the existential modalities referring to knowledge on the one hand and belief on the other. In both cases the best transliteration seems to be *'the agent considers it possible that'*. Under the doxastic reading, the existential modality is intended to refer to the lack of a contradiction with the agent's beliefs, while in the context of epistemic logic, considering something to be possible amounts to that something not contradicting one's knowledge.

diamond-operators stand for existential modalities.

Now for the formal definition. Given a fixed set of propositional letters, the set of well-formed *formulas* of the basic modal language can be defined inductively as follows:

(1) Propositional letters and the special symbols $\top$ and $\bot$ are formulas.

(2) If φ and ψ are formulas, so are $(\neg\varphi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, $(\varphi \leftrightarrow \psi)$, $(\Diamond\varphi)$, and $(\Box\varphi)$.

(3) There are no other formulas in this language.

The formulas from item (1) are the atomic propositions of our language. Recall that $\top$ (*verum*) stands for a proposition that is true under any interpretation, while $\bot$ (*falsum*) represents a proposition that can never be true. The second part of the definition specifies how we can build compound formulas from simpler ones using the propositional connectives and the two modal operators $\Diamond$ and $\Box$. The third condition simply states that this is the only way of constructing well-formed formulas. Here are some examples for formulas belonging to the basic modal language:[4]

$$\Box(P \rightarrow Q), \quad \Box P \leftrightarrow \neg\Diamond\neg P, \quad \Box P \rightarrow \Box\Box P, \quad \neg P \vee P$$

For the alethic reading, the first of these stands for 'it is necessarily the case that P implies Q'. The second one says that P is necessarily true iff it is not the case that it is possibly false. This is the duality principle which we have already touched upon before. It also applies in the case of temporal logic: P will always be true iff it is not the case that at some point in the future P will be false. And so on. To make this discussion of the *meaning* of modal formulas precise we have to define a *formal semantics* first. This is our objective for the next section.

2.2 Possible World Semantics

Possible worlds. In the previous section we have already sneaked in, through the backdoor as it were, the idea that something is *necessarily true iff it is true in all possible worlds*. This idea goes back to Leibniz, who — besides introducing this very useful metaphor — famously argued that, not only, our actual world is the best of all possible worlds, but that this is in fact an expression of the benevolence of some omnipotent deity. We will be concerned with rather more worldly business here.

The possible worlds metaphor is also applicable to the other domains we have discussed in the previous section. For instance, John believes φ iff φ is true in all worlds that are possible according to John's beliefs. And if we think of time as a 'fourth dimension', that is, if we model time as a sequence of worlds (snapshots of our world taken at consecutive points in time) then we could say that something will always be true iff it will be true in all 'future worlds'. A difference between the case of alethic logic and the other

[4] We are going to drop unnecessary brackets whenever possible.

domains is that in the former we really mean *all* possible worlds, while in most other cases we mean all those possible worlds that meet certain requirements, such as lying in the future (rather than the past) or being possible according to John (but maybe not according to Mary). Which worlds we intend to refer to in any given situation depends on the state of the actual world we are in at that moment. Which worlds are *relevant* from the standpoint of a particular world (the actual world in a given situation) can be modelled by means of a binary relation over possible worlds. Those other worlds that are accessible from the current world via this relation are those that we intend our modal operator to refer to.

Combining Leibniz' idea of possible worlds with this notion of an *accessibility relation* may stretch the imagination a little, or as Hughes and Cresswell put it in the first edition of their *Introduction to Modal Logic* [41]:

> *"This notion of one possible world being accessible to another has at first sight a certain air of fantasy or science fiction about it."*

Still, these kinds of relational structures are probably what is most characteristic about (the semantics of) modal logics and modern accounts of modal logic tend to emphasise this aspect rather than the philosophical roots of the field. This view is epitomised by the "first slogan" in the preface of the recent textbook on modal logic by Blackburn, de Rijke, and Venema [8]:

> *"Modal languages are simple yet expressive languages for talking about relational structures."*

We have seen what these languages (or rather its most basic representative) look like at the end of the previous section. Now we turn to the formal definition of the relational structures themselves.

Frames and models. A *frame* is a pair $\mathcal{F} = (W, R)$, where W is a non-empty set and R is a binary relation over W. The elements of W are called *worlds* and R is called the *accessibility relation* of $\mathcal{F}$. Depending on the application domain, R may be required to have certain properties, such as being transitive or reflexive. For the time being, however, we do not make any assumptions of this kind.

Frames are the relational structures mentioned before. We now have to define how formulas of our basic modal language relate to these structures. To this end we introduce the notion of a *model*.[5] Models specify, in the first instance, which propositional letters are true at which worlds. Formally, a model is a pair $\mathcal{M} = (\mathcal{F}, V)$, where $\mathcal{F} = (W, R)$ is a frame and V is a function from propositional letters to sets of worlds, that is, to subsets of W. The function V is called the *valuation* function of $\mathcal{M}$. Intuitively, we think of $V(P)$ as the set of worlds at which the atomic proposition P is true. We are soon going to make this idea more precise.

[5] Models in modal logic are also known as *Kripke models*, a reference to the work of Kripke who established possible world semantics as a formal tool for the analysis of modal logics in his seminal paper from 1963 [45].

Definable operators. In propositional logic, it is not necessary to explicitly provide semantics (that is, to give a truth table) for all the connectives of the language. Instead, we may treat some of them as definable operators. Negation and conjunction alone, for instance, form a *functionally complete* set of operators, that is, all of the other propositional connectives can be defined in terms of them. This approach has the advantage that we only need to consider these two core operators during formal proofs and the like, because we can always translate a general formula into a formula using only negation and conjunction. We may rewrite formulas involving propositional connectives other than negation and conjunction as follows:

$$\begin{array}{rcl}\varphi \vee \psi & = & \neg(\neg\varphi \wedge \neg\psi) \\ \varphi \rightarrow \psi & = & \neg\varphi \vee \psi \\ \varphi \leftrightarrow \psi & = & (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)\end{array}$$

Additionally, we may treat the symbol $\bot$ as an abbreviation for $\neg\top$. Hence, we only have to give semantics for the operators $\neg$, $\wedge$, $\Diamond$, and $\Box$ and the special proposition $\top$ here. (In fact, as we shall see later on, the modal operator $\Box$ may also be considered a defined operator.)

Truth in a model. Given the notion of a model introduced before, we can now give a formal definition of the semantics of our basic modal language. Let $\mathcal{F} = (W, R)$ be a frame and let $\mathcal{M} = (\mathcal{F}, V)$ be a model based on that frame. We write $\mathcal{M}, w \models \varphi$ to express that the formula φ is *true* at world $w \in W$ in that model. (Alternatively, we may say that φ holds or is satisfied at w.) The truth relation $\models$ is defined inductively as follows:

(1) $\mathcal{M}, w \models P$ iff $w \in V(P)$ for propositional letters P;

(2) $\mathcal{M}, w \models \top$ (i.e. $\top$ is true at every world in any model);

(3) $\mathcal{M}, w \models \neg\varphi$ iff $\mathcal{M}, w \not\models \varphi$ (i.e. iff $\mathcal{M}, w \models \varphi$ is not the case);

(4) $\mathcal{M}, w \models \varphi \wedge \psi$ iff both $\mathcal{M}, w \models \varphi$ and $\mathcal{M}, w \models \psi$;

(5) $\mathcal{M}, w \models \Diamond\varphi$ iff there exists a $w' \in W$ such that $(w, w') \in R$ and $\mathcal{M}, w' \models \varphi$;

(6) $\mathcal{M}, w \models \Box\varphi$ iff for all $w' \in W$ with $(w, w') \in R$ we have $\mathcal{M}, w' \models \varphi$.

We can now infer the semantics of the definable operators such as the connective for disjunction. For instance, we have $\mathcal{M}, w \models \varphi \vee \psi$ iff $\mathcal{M}, w \models \neg(\neg\varphi \wedge \neg\psi)$ (by rewriting of the disjunction) iff it is not the case that neither $\mathcal{M}, w \models \varphi$ nor $\mathcal{M}, w \models \psi$ hold (by definition of the semantics of the negation and the conjunction operators) iff $\mathcal{M}, w \models \varphi$ or $\mathcal{M}, w \models \psi$ (by application of the respective de Morgan law on the meta-level).

Observe that we have $\mathcal{M}, w \models \Box\varphi$ iff $\mathcal{M}, w \models \neg\Diamond\neg\varphi$, i.e. φ is true at all accessible worlds iff it is not the case that φ is not true at at least one accessible world. This equivalence provides a formal basis for the duality of the box- and the diamond-operator

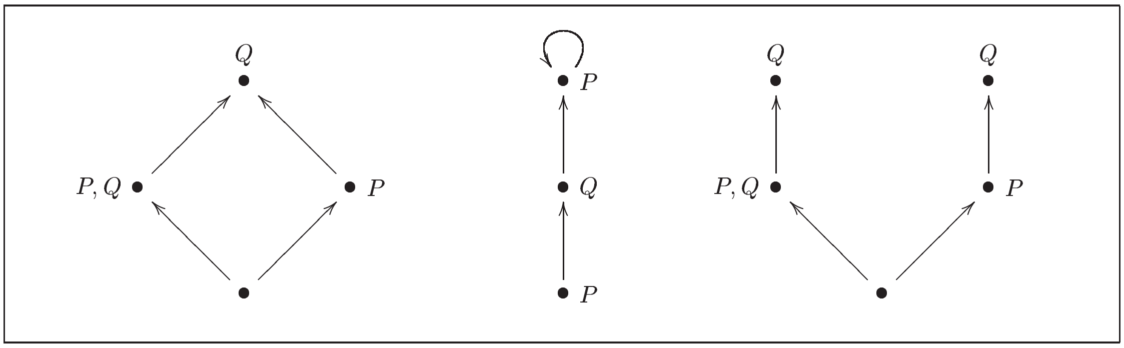

Figure 2.1: Three models

we have already pointed out earlier. It also shows that we could treat the box-operator as a defined operator, just as disjunction and implication:

$$\Box\varphi \quad = \quad \neg\Diamond\neg\varphi$$

Examples. Figure 2.1 shows three sample models. The one on the lefthand side, for instance, is based on a frame with four worlds. The accessibility relation is shown as arrows between worlds. The valuation function is indicated by the propositional letters written next to the worlds. In these examples we consider a language with only two propositional letters: P and Q. Neither of them happens to hold at the first world at the bottom in the model on the lefthand side. This world has two successors. Both P and Q are true at the left one, while only P holds at the one to the right. The fourth world (at the top of the frame) again only satisfies Q. Two examples for formulas that are true at the first world of this model would be $\Box P$ and $\Diamond Q$, because P holds at every accessible world, while there is at least one successor world at which Q is true. The formula $\Diamond Q$ is also true at the first world in the second model, but $\Box P$ is not. And so on.

It is interesting to observe that our basic modal language is not powerful enough to 'distinguish' between the first and the third sample model. Precisely the same formulas hold at the world at the bottom of either one of the two models. To see this, consider that, if we look along the accessibility relation from the standpoint of that first world and if we are only interested in either the existence of an accessible world at which certain propositions are true or in universal properties of all accessible worlds, then the two models indeed 'look' the same. For instance, for both first worlds it is the case that Q is satisfied at all worlds that are accessible in two steps. The appropriate technical term to describe this phenomenon is to say that the first model is a *p-morphic image* of the third one. P-morphisms are discussed in some detail in the later parts of Chapter 3.

We should point out that the examples from Figure 2.1 are rather simple models that have been selected purely for the purpose of illustrating the definition of truth given before. None of them appears to be of much use with respect to any of the four application domains (i.e. reasoning about necessity, belief, knowledge, or time, respectively) discussed earlier. Only models (or rather frames) with certain properties are admissible for these specific modal logics. We are going to see what these properties are in the next section.

Satisfiability and validity. The formula $\Box P \wedge \Diamond \neg P$ does not hold at any of the worlds in any of our sample models from Figure 2.1. In fact, this formula can never be true at any world, whatever the model. To see this, recall the semantics of $\Box P \wedge \Diamond \neg P$. On the one hand, the first conjunct requires P to be true at every accessible world. On the other hand, the second conjunct postulates the existence of at least one accessible world at which P is false. That is, this formula would require the existence of a world that is propositionally inconsistent (as it would have to satisfy both P and $\neg P$). This is not possible (see definition of the truth relation $\models$ for the negation operator). We call formulas such as $\Box P \wedge \Diamond \neg P$ *unsatisfiable* formulas.

The general definition of satisfiability goes as follows. A formula φ is called *satisfiable* iff there are a model $\mathcal{M}$ and a world w in that model such that $\mathcal{M}, w \models \varphi$ holds. If φ is true at all worlds in a given model $\mathcal{M}$ then we say that φ is *globally true* in $\mathcal{M}$. This is written as $\mathcal{M} \models \varphi$. If φ is true at every world in every model then φ is called a *valid* formula. This is written as $\models \varphi$. If φ is at least true at every world in every model based on a frame with certain properties (such as having a transitive accessibility relation, for instance) then we say that φ is valid in that class of frames. If φ is true at every world in every model based on a particular frame $\mathcal{F}$, then we say that φ is valid in that frame and write $\mathcal{F} \models \varphi$.

Recall that a logic is said to be *decidable* iff there exists a general algorithm that will tell us for any given well-formed formula φ of the respective language whether φ is valid, that is, whether φ belongs to the set of formulas that 'make' this logic, or not (in finite time). Alternatively, we may also adopt an algorithm that decides whether φ is satisfiable or not, because the problems of deciding validity and of deciding satisfiability are closely related. A formula φ is valid iff its negation $\neg\varphi$ is not satisfiable. Much of Chapter 5 is devoted to a proof of decidability for our logic of ordered trees.

Multi-modal logics. For certain applications we may require a logic with more than one pair of modal operators. In the case of epistemic logics, we may, for instance, wish to model the knowledge of two agents, say, Mary and John (rather than just a single agent as before). On the semantic level, we introduce one accessibility relation for each mode of truth we want to include in the model. On the syntactic level, we introduce a box- and a diamond-operator for each one of these accessibility relations. The various definitions generalise in the obvious way.

The modal logic of ordered trees studied in this thesis is an example for such a multi-modal logic.

Translation into first-order logic. An alternative way of giving semantics to a new logic would be to define a translation from formulas of that logic into another well-understood system. Here we are going to give the so-called *standard translation* from general multi-modal logics into classical first-order logic. Such an embedding from one logic into the other also shows that the basic (multi-)modal language cannot be more expressive than first-order logic. (This is a good thing, because otherwise we could not hope for decidability.)

Below we are going to give an inductive definition of a translation function ST from modal formulas to first-order formulas. Every propositional letter P of the modal language will be translated into a unary predicate P'.[6] For every pair of modal operators $\Box_R$ and $\Diamond_R$ pertaining to an accessibility relation R we introduce a binary predicate R'. Every formula (with the exception of $\top$) will be translated into a first-order formula with a single free variable. The name of that free variable appears as an index to the translation function. For instance, we use ST_x to indicate that the resulting first-order formula will have the free variable x. Intuitively, this variable stands for the worlds at which the modal logic counterpart of the first-order formula is true. The standard translation function is defined as follows:

$$\begin{array}{rcl} \mathrm{ST}_x(P) & = & P'(x) \quad \text{for propositional letters } P \\ \mathrm{ST}_x(\top) & = & \top \\ \mathrm{ST}_x(\neg\varphi) & = & \neg\, \mathrm{ST}_x(\varphi) \\ \mathrm{ST}_x(\varphi \wedge \psi) & = & \mathrm{ST}_x(\varphi) \wedge \mathrm{ST}_x(\psi) \\ \mathrm{ST}_x(\Diamond_R\, \varphi) & = & (\exists y)[R'(x,y) \;\wedge\; \mathrm{ST}_y(\varphi)] \\ \mathrm{ST}_x(\Box_R\, \varphi) & = & (\forall y)[R'(x,y) \rightarrow \mathrm{ST}_y(\varphi)] \end{array}$$

Here, each time we translate another diamond- or box-operator we have to choose a *new* bound variable y. The standard translation of modal formulas involving any of the other propositional connectives in the language may be inferred from the definitions of these operators in terms of negation and conjunction.

As an example, we translate the modal formula $\Box(P \wedge \Diamond Q)$ into first-order logic. If we use the unary predicates P' and Q' as translations of the propositional letters P and Q, respectively, and if we use the binary predicate R' to represent the accessibility relation underlying the pair of modal operators $\Box$ and $\Diamond$, using the standard translation will yield the following result:

$$(\forall y)[R'(x,y) \rightarrow P'(y) \wedge (\exists z)[R'(y,z) \wedge Q'(z)]]$$

Here, x is the free variable. Note that we have used a third variable name, namely z, for the translation of the second modal operator appearing in the input formula. Alternatively, we could have reused the variable name x at this point. The resulting formula would then have been a first-order formula with only two distinct variable names (x and y). Inspection of the translation function ST shows that, in fact, this is always possible: any formula of the basic modal logic can be translated using at most two

[6] In order to clearly distinguish the language used for modal formulas from that of first-order logic, we use letters with accents (such as P') to denote first-order predicate symbols. Of course, in practice this would not be necessary.

variable names. In other words, our basic modal logic can be embedded into the *two-variable-fragment* of first-order logic. The significance of this observation is that the two-variable-fragment is known to be *decidable*.[7] Hence, the basic modal logic must be decidable, too.

Our decidability proof for our modal logic of ordered trees given in Chapter 5 is also based on the embedding into another system that is known to be decidable (in this case the second-order theory of two successor functions). A translation of our logic into first-order logic (over the theory of ordered trees) is given in Chapter 6.

2.3 Axioms and Correspondences

Axiomatic characterisation. In the previous section, we have seen how we can characterise the basic modal logic in terms of its model-theoretic semantics. Now we are going to complement this view by giving an axiomatic (or proof-theoretical) characterisation of the same logic. An *axiomatisation* of a logic is a collection of *axioms* and *inference rules*. The axioms are the basic formulas that are taken to be 'true' and the rules can be used to infer further such formulas. Formulas that can be derived in an axiomatic system are called *theorems* of the system. A formula is called *consistent* iff it is not possible to derive its complement.

Intuitively, the concept of consistency corresponds to the semantic concept of satisfiability, while the concept of theoremhood corresponds to semantic validity. Showing that these concepts really coincide, that is, showing that the semantic and the axiomatic characterisation of a logic match can provide deep insights into either one of them. We call an axiomatisation *sound* (with respect to a given semantic characterisation) iff every satisfiable formula is consistent. Dual to the concept of soundness is that of completeness. We call an axiomatisation (weakly) *complete* iff every consistent formula is satisfiable.[8]

Axioms and rules. Modal logic extends classical propositional logic. In particular, any formula that is a theorem of classical propositional logic (i.e. a tautology) will also be a theorem of the basic modal logic. Furthermore, any formula that we can construct by uniformly replacing propositional letters in a classical tautology by a well-formed formula will be a theorem. For instance, $\neg P \vee P$ is a tautology and $\Diamond(\varphi \wedge \psi)$ is a well-formed modal formula. Hence, the formula $\neg\Diamond(\varphi \wedge \psi) \vee \Diamond(\varphi \wedge \psi)$ will be a theorem.

We are only going to give those axioms of the basic modal logic here that go beyond classical logic. In fact, there are only two of them. The first one simply encodes the duality of the box- and the diamond-operator which we have already discussed earlier:

(Dual) $\quad \Box A \leftrightarrow \neg\Diamond\neg A$

[7] This was first shown by Scott in 1962 [63], for first-order logic without equality. Decidability of the two-variable-fragment with equality is due to Mortimer [57] (see also [11]).

[8] In Chapter 4 we also discuss the related concept of *strong* completeness (see page 106).

That is, $\Box A \leftrightarrow \neg\Diamond\neg A$ is a theorem of the basic modal logic and so is any formula that we can construct by substituting a well-formed formula for the letter A in this axiom. This matches our earlier definition of the semantics of the two modal operators. A formula such as A will be true in *all* accessible worlds iff it is not the case that it is not true in *some* accessible world.

While the duality axiom is essentially a definition of one modal operator in terms of the other, the second modal axiom really tells us something about the properties of our modal operators:

$$\text{(K)} \quad \Box(A \to B) \to (\Box A \to \Box B)$$

Axiom (K) is also known as the *distribution axiom*, because it expresses that we may distribute the box-operator with respect to an implication.[9] Again, we can easily verify that this is a sound axiom: whenever $A \to B$ is true in all accessible worlds and we then find that also A is true in all accessible worlds, we may conclude that the consequent B must be true in all accessible worlds as well.

There are three inference rules in the system. The first one is the rule of *uniform substitution*, which is familiar form classical propositional logic. It states that whenever we have derived a particular theorem we may uniformly substitute any one propositional letter in that formula by an arbitrary well-formed formula of our modal language and we obtain another theorem of the system. The second rule of inference is the well-known *modus ponens*, which says that whenever an implicational formula as well as its antecedent are theorems, so is its consequent:

$$\frac{A,\ A \to B}{B}$$

The only strictly modal rule of the system is the so-called *generalisation rule*, which is also known as the rule of *necessitation*, a reference to the alethic interpretation of the modal box-operator. The rule states that whenever a formula A is known to be a theorem then $\Box A$ will be a theorem as well:

$$\frac{A}{\Box A}$$

To see that this rule is semantically valid we make the following observation. If A is a theorem then A must be true at every world in any model. Hence, A will certainly be true at all worlds accessible from some other world. But the latter is the truth condition we have used to semantically characterise the box-operator, i.e. $\Box A$ will also be true at any world in any model.

This axiomatic system is known to be *complete* with respect to the semantics defined in the previous section, that is, any formula for which we cannot derive its complement must be a satisfiable formula. In other words, any formula that is valid *can* be derived.

[9] The name of the axiom (K) is generally understood to refer to Kripke, because it is characteristic for the most basic modal logic that can be interpreted using Kripke models [45].

We are not going to prove this result here. (A proof may, for instance, be found in [8].) Chapter 4 is devoted to these issues as far as our modal logic of ordered trees is concerned. It also outlines the general approach of obtaining completeness results by constructing a so-called *canonical model*.

Example. As an example, we demonstrate how the formula $\Diamond(A \vee B) \rightarrow \Diamond A \vee \Diamond B$ can be derived from the axioms of the calculus just presented, that is, how it can be shown to be a theorem. We first observe that it is easily seen to be *semantically* valid: whenever there is an accessible world in which either A or B is true then it will also be the case that there is either an accessible world satisfying A or there is an accessible world satisfying B. Now let us see how we can get to the same result on the syntactic level. The proof steps in the following derivation should be mostly self-explanatory. Some steps that do only involve simple propositional reasoning are not carried out in full detail, but we could easily fill in the missing bits. To begin with, we have to make a good guess what instantiation of an axiom or a propositional tautology may lead to the desired result. Here is our derivation:

(1)	$\Box(\neg A \rightarrow (\neg B \rightarrow \neg A \wedge \neg B)) \rightarrow (\Box\neg A \rightarrow \Box(\neg B \rightarrow \neg A \wedge \neg B))$	from (K)
(2)	$\neg A \rightarrow (\neg B \rightarrow \neg A \wedge \neg B)$	propositional tautology
(3)	$\Box(\neg A \rightarrow (\neg B \rightarrow \neg A \wedge \neg B))$	from (2) by generalisation
(4)	$\Box\neg A \rightarrow \Box(\neg B \rightarrow \neg A \wedge \neg B)$	from (1) and (3)
(5)	$\Box(\neg B \rightarrow \neg A \wedge \neg B) \rightarrow (\Box\neg B \rightarrow \Box(\neg A \wedge \neg B))$	from (K)
(6)	$\Box\neg A \wedge \Box\neg B \rightarrow \Box(\neg A \wedge \neg B)$	from (4) and (5)
(7)	$\neg\Box(\neg A \wedge \neg B) \rightarrow \neg\Box\neg A \vee \neg\Box\neg B$	from (6)
(8)	$\Diamond\neg(\neg A \wedge \neg B) \rightarrow \Diamond A \vee \Diamond B$	from (7) and (Dual)
(9)	$\Diamond(A \vee B) \rightarrow \Diamond A \vee \Diamond B$	from (8)

In practice, axiomatic derivations can be difficult to find. One usually has to work backwards, starting from the theorem to be derived.

Additional axioms for specific logics. As we have already observed in the previous section, not all modal frames are suitable for reasoning about, say, knowledge. Similarly, not all formulas that are consistent with respect to the basic axiomatic system presented in this section seem appropriate once we assign specific meaning to our modal operators. As an example, consider the formula $\Box P \wedge \neg P$. This formula is consistent, because its negation is not derivable in our proof system.[10] But do we *want* it to be consistent, for instance, when working within an epistemic logic?

If we interpret the universal modality $\Box$ as *'Mary knows'*, then our formula $\Box P \wedge \neg P$ reads: *'Mary knows P but P is not true'*. This is certainly *not* compatible with our

[10] This should be intuitively clear by inspection of the axioms and proof rules available to us (but a truly formal argument why this is not possible is certainly more difficult). Going back to semantics, it is very easy to see that $\Box P \wedge \neg P$ is also a *satisfiable* formula. All that is required is a model with a world where P is false but true at every accessible world.

understanding of the notion of *knowledge*. Anything that is known (by Mary or whomever else) in the true sense of the word must in fact *be* the case. We can integrate this idea into our proof system by adding the following axiom:[11]

(T) $\Box A \rightarrow A$

Postulating this axiom (for epistemic modal logics) means postulating *veridicality* of Mary's knowledge: her knowledge accurately reflects the actual state of her environment. Given this additional axiom, $\Box P \wedge \neg P$ is easily seen to be inconsistent. Applying modus ponens to $\Box P$ and a suitable instantiation of (T) yields P, which contradicts the second conjunct $\neg P$. Axiom (T) is also desirable when reasoning about necessity. Surely, everything that is necessarily true will also be actually true. For doxastic logic, on the other hand, the situation is different. Not every proposition believed to be true by John will in fact *be* true.

Next, let us consider the following axiom that may also arguably be added to our proof system:[12]

(4) $\Box A \rightarrow \Box\Box A$

In the context of reasoning about an agent's knowledge, this axiom embodies the notion of *positive introspection*. It expresses that everything that is known must also be known to be known. Axiom (4) also makes sense for the three other areas of application we have discussed earlier. The alethic reading would be that whenever A is necessarily true then this is necessarily the case. Under a doxastic interpretation, it states that what is believed is believed to be believed. Finally, in temporal logic, axiom (4) expresses that whenever A will be true always in the future, then also always in the future it will be the case that always in the future A will be true.

In epistemic logic, related to positive introspection is the concept of *negative introspection*. We say that an agent is capable of performing a negative introspection of its knowledge iff whenever it does not know about a proposition A then it *does* know that it does not have this piece of knowledge. This property can be expressed by means of the formula $\neg\Box A \rightarrow \Box\neg\Box A$. If we uniformly substitute A by $\neg A$ in this formula and then use the duality axiom to rewrite combinations of negation and the box-operator by diamond-operators, we end up with the following (neater version of this) axiom:[13]

(5) $\Diamond A \rightarrow \Box\Diamond A$

The concept of negative introspection is not generally accepted as a property of knowledge. That is, there are epistemic logics that include axiom (5) and there are others that do not include it. Being aware of what one does *not* know (as in the case of negative

[11] The name of axiom (T) goes back to the work of Feys [21, 22] who first described a logical system (*"la logique t"*) centred around this axiom (quoted after [15]).

[12] The name of this axiom is a reference to the work of Lewis [49]. Axiom (4) is the characteristic axiom in **S4**, the fourth of his logical systems of strict implication.

[13] Axiom (5) is characteristic for **S5**, Lewis' fifth system of strict implication; hence the name [49].

introspection) arguably requires a higher state of consciousness than merely being aware of what one does know (as for positive introspection). In the case of belief, i.e. for doxastic logics, the situation is less critical. If John does not believe in the truth of A then he can be expected to also believe that he does not believe in A.

For alethic logic, axiom (5) seems reasonably acceptable: if something is possibly true then this is necessarily so. For temporal logic, however, we clearly have to reject axiom (5). It is not the case that, if A will be true at some point in the future, then this will remain so forever.

We have mentioned on more than one occasion that the notion of belief is weaker than that of knowledge, in the sense that what is believed is not necessarily required to be true. Nevertheless, a *rational* agent will not believe just about anything. In particular, it seems reasonable to postulate that a rational agent would not believe in a proposition that is logically inconsistent. In this case, we should adopt the formula $\neg\Box\bot$ as an axiom. The same is true for both alethic and epistemic logics: what is inconsistent can hardly be necessarily true or 'known' to be true by a rational agent. Exploiting the duality of $\Box$ and $\Diamond$, we can rewrite the formula $\neg\Box\bot$ as follows:[14]

(D) $\Diamond\top$

The temporal interpretation of (D) is clear: it embodies the concept of an endless future; for any point in time there always still exists a later point. Time never stops. This will be a useful assumption for most applications, but whether or not to adopt it in a particular situation is, as ever, entirely up to the logic engineer devising the formal system in question.

Naming conventions for standard modal logics. The axioms we have introduced in this section form the basis of some of the most important systems of basic modal logic. The logic determined by axiom (K), together with the duality axiom and the set of all classical tautologies as well as modus ponens, the generalisation rule, and the rule of uniform substitution is known under the name of **K**. This is the logic we have so far referred to as 'the' basic modal logic. The names of other logical systems can be derived by appending the letters (or numbers) referring to the chosen additional axioms at the end of the name of a logic. For instance, the logic **KT4** is the modal logic we get by adding axioms (T) and (4) to the basic calculus of **K**. Besides this systematic naming scheme, also a number of rather more traditional names are being used for particular basic modal logic systems. The logic **KT4**, for instance, is usually referred to as **S4**, while **KT5** is better known as **S5**. These are the original names of Lewis' systems of strict implications [49].

It follows from our discussion of the various additional axioms presented here, that **S5** is a system that is appropriate for reasoning about alethic modes of truth,[15] while **KD45**

[14] The name of axiom (D) stems from its central role in *deontic logic*, another important branch of modal logics. Deontic logics are used to formalise concepts such as *obligation* and *permission* [3].

[15] Axioms (D) and (4) are derivable in **S5**.

is an example for a doxastic logic. Both **S4** and **S5** may serve as epistemic logics that allow us to model the knowledge of a single agent (the latter only if we are prepared to accept negative introspection). However, no combination of the simple axioms presented here seems to be strong enough to adequately characterise a system of temporal logic.

Indeed, axiomatising temporal logics is typically more difficult than axiomatising various other systems of modal logic. Chapter 4 addresses some of the issues involved in the context of our extended temporal logic based on ordered trees.

Correspondence theory. As we have just seen, the axiomatic approach can be very helpful in making certain characteristics of modalities, such as veridicality or positive introspection in the case of an epistemic interpretation, precise. On the other hand, the semantic view often provides the better intuitions when we think about what is true, can be true, or must be true. For instance, it is usually more convenient to think about satisfiability and the existence of models rather than consistency and the lack of axiomatic derivations. To bring these two perspectives on modal logic together once again, in the remainder of this section, we shall discuss the question of how the additional axioms introduced earlier are reflected in the structure of the corresponding models.

As we shall see, each one of them, in fact, corresponds to a simple (and intuitive) property of the underlying accessibility relation. This object of study in modal logics is known as *correspondence theory* [69].

The case of reflexive frames. Axiom (T), for instance, which we have introduced earlier, characterises the class of *reflexive* frames, i.e. frames with a reflexive accessibility relation. Recall that (T) stands for the formula $\Box A \rightarrow A$. By saying that this formula *characterises* the class of reflexive frames we mean, more concretely, the following:

Claim: Let $\mathcal{F} = (W, R)$ be a frame. Then (T) is valid in $\mathcal{F}$ iff R is reflexive.

How can we prove such a correspondence result? Showing that reflexivity of the accessibility relation associated with a particular frame implies validity of (T) in that frame is the easier part. In short, we have to verify that $\Box A \rightarrow A$ is true at every world in any model based on a reflexive frame. So let $\mathcal{F} = (W, R)$ be a frame with a reflexive accessibility relation R. By definition of the notion of frame validity, showing that $\Box A \rightarrow A$ is valid in $\mathcal{F}$ amounts to showing that $\mathcal{M}, w \models \Box A \rightarrow A$ holds for every model $\mathcal{M} = (\mathcal{F}, V)$ based on the frame $\mathcal{F}$ and every world $w \in W$. So let $\mathcal{M} = (\mathcal{F}, V)$ be an arbitrary model based on $\mathcal{F}$ and let w be an arbitrary world in W. To prove that $\mathcal{M}, w \models \Box A \rightarrow A$ holds, we assume $\mathcal{M}, w \models \Box A$ and try to infer $\mathcal{M}, w \models A$. The former is the case iff $\mathcal{M}, w' \models A$ holds for all $w' \in W$ with $(w, w') \in R$. This is where our premise of the reflexivity of the accessibility relation comes into play. Given that R is reflexive, we have $(w, w) \in R$, i.e. $\mathcal{M}, w \models A$ must hold. That is, we have shown that $\Box A \rightarrow A$ will be true in any model $\mathcal{M}$ based on the frame $\mathcal{F}$ and at any world w in that frame. Hence, (T) is valid in $\mathcal{F}$ as claimed.

	Axiom	Frame Condition	
(T)	$\Box A \rightarrow A$	$(\forall x)R(x,x)$	(*reflexive*)
(B)	$A \rightarrow \Box\Diamond A$	$(\forall x)(\forall y)[R(x,y) \rightarrow R(y,x)]$	(*symmetric*)
(4)	$\Box A \rightarrow \Box\Box A$	$(\forall x)(\forall y)(\forall z)[R(x,y) \wedge R(y,z) \rightarrow R(x,z)]$	(*transitive*)
(5)	$\Diamond A \rightarrow \Box\Diamond A$	$(\forall x)(\forall y)(\forall z)[R(x,y) \wedge R(x,z) \rightarrow R(y,z)]$	(*euclidean*)
(D)	$\Diamond\top$	$(\forall x)(\exists y)R(x,y)$	(*serial*)

Table 2.2: Basic correspondences

Now for the other direction. Suppose (T) is valid in the frame $\mathcal{F} = (W, R)$, i.e. $\Box A \rightarrow A$ is true in every model based on $\mathcal{F}$ at every world in the frame. We need to show that R must be a reflexive relation. For the sake of contradiction, assume this is not the case, i.e. there exists a world $w \in W$ such that $(w, w) \notin R$. We are going to use this world w to define a valuation function V such that $\Box A \rightarrow A$ is *not* satisfied at w in the model $\mathcal{M} = (\mathcal{F}, V)$. In fact, all we need to specify about V is what set of worlds it maps the propositional letter A to. Let V be any valuation function such that $V(A) = W \setminus \{w\}$ and let $\mathcal{M} = (\mathcal{F}, V)$ be the corresponding model. It follows that $\mathcal{M}, w \models \Box A$ holds, because A is satisfied at every world accessible from w (given that w itself, the only world where A is false, is not accessible), and that $\mathcal{M}, w \models A$ does not hold. Hence, $\mathcal{M}, w \models \Box A \rightarrow A$ is not the case either, which contradicts our original assumption of (T) being valid in $\mathcal{F}$. This completes our proof of the above claim.

Other correspondences. Similarly to the reflexivity axiom (T), each one of the additional axioms we have discussed earlier also corresponds to a simple frame condition. These correspondences are summarised in Table 2.2 and can be proved to hold in essentially the same manner as we have verified the correspondence between (T) and the class of reflexive frames. The table also shows how the respective frame conditions can be expressed in first-order logic. The formula $(\forall x)R(x,x)$, for example, defines the relation associated with the binary predicate symbol R as being reflexive.

Axiom (4) characterises *transitive* frames and axiom (5) characterises those that have a *euclidean* accessibility relation, i.e. frames where it is the case that whenever two worlds are accessible from the same world then those two worlds must be accessible from each other as well. Axiom (D) is valid in frames where every world has at least one successor with respect to the accessibility relation; that is, (D) embodies the notion of *seriality*. In addition to the axioms discussed earlier, Table 2.2 also lists another important schema:[16]

(B) $A \rightarrow \Box\Diamond A$

This formula may, for instance, be adopted as an axiom of alethic logic, in which case it expresses that anything that is true in the actual world is necessarily possible. It is valid in a frame iff that frame has got an accessibility relation that is *symmetric*.

[16] It appears that axiom (B) was first proposed by Becker [7], who referred to it as the *"Brouwersche Axiom"* (quoted after [49]; see also [15]).

In the context of our ordered tree logics, we are briefly going to discuss issues in correspondence theory in Chapter 3 (Section 3.3).

First-order definability. Our presentation may suggest that *any* modal axiom we can think of characterises a frame condition that can be defined in terms of a first-order formula over the predicate R representing the accessibility relation. This is in fact not the case. The best known counterexample is the so-called McKinsey axiom:[17]

(M) $\quad \Box\Diamond A \rightarrow \Diamond\Box A$

It can be shown that (M) does not correspond to any first-order frame condition. A proof may, for instance, be found in [69].

Translation again. As a final remark in this section let us point out that in those cases where a modal logic *is* first-order definable, the respective first-order frame conditions must be taken into account if we want to embed that modal logic into first-order logic (by means of translation). The logic **S4**, for example, — which is characterised by the additional axioms (T) and (4) — *cannot* be embedded into the two-variable-fragment of first-order logic anymore, because we require a minimum of three distinct variable names to encode the transitivity condition corresponding to (4). Hence, the argument used at the end of the previous section to show that the basic modal logic without any additional axioms (i.e. the logic **K**) is decidable, would fail in this case.[18]

This concludes our general introduction to basic modal logic. The next section is more specifically devoted to the case of temporal logic.

2.4 Temporal Logic

Temporal modal logics are modal logics over very specific frames that embody those aspects of *time* we wish (and are able) to represent.

What is time? This is a (largely philosophical) question, which we are not going to answer here. Instead, we are going to be rather more pragmatic and discuss some of our options as far as representing aspects of time by means of a modal logic frame are concerned. Recall that a frame consists of a set of worlds together with an accessibility relation (or several accessibility relations in the case of multi-modal logics). If we want to give such a frame a temporal interpretation, we first have to decide what worlds and accessibility relation(s) should stand for. Generally speaking, worlds will be used to represent the primitive objects in the domain of our temporal semantics. In most cases, that is, for most temporal logics discussed in the literature, these primitive objects are *time points*. Alternatively, we may also choose time *intervals* as primitives.

[17] Axiom (M) is called the McKinsey axiom, because it is a theorem in the system **S4.1** proposed by McKinsey in [55] (see also [42]).

[18] **S4** is still a decidable logic, but we require a different method to prove this [8].

In the case of a point-based semantics, the accessibility relation induces an *earlier-later ordering* over time points. An example would be the temporal frame $(\mathbb{N}_0, <)$, where the set of points (worlds) is the set of the natural numbers (including 0) and the accessibility relation $<$ is the usual lower-greater ordering over $\mathbb{N}_0$. By adopting this frame, we are making certain assumption about time: it is *discrete* (in particular, for every point in time there is a unique *next* point), *bounded* to the left, namely by the first time point 0 (the 'big bang', the beginning of time), and *unbounded* to the right (i.e. the future is taken to be without end). None of these choices is obligatory though. In particular, we may consider temporal frames that are *dense* (if the notion of a 'next' moment is inappropriate for an application) or unbounded to the left (if we want to model both future and past). More fundamentally, in the case of the frame $(\mathbb{N}_0, <)$, time is understood to be *linear*. But again, this is not an obligatory choice. For instance, if we wish to explicitly represent different possible futures, we may adopt a temporal frame where time is *branching*. However, for the remainder of this introduction (the remainder of this thesis, in fact) we are going to assume that time is linear.[19]

If we choose intervals as primitives, then a single accessibility relation representing a simple earlier-later ordering (possibly together with the notion of alternative futures, in the case of branching time) will not be enough anymore. Two distinct intervals may not only lie either *before* or *after* each other, but for instance, one of them may also occur *during* the other or they may *overlap*. While in this introduction, we are only going to be concerned with point-based temporal logics, our modal logic of ordered trees discussed in the rest of this thesis may also be interpreted as a (restricted) interval logic.[20]

Temporal logic operators. The language of the most basic temporal logics is the same as the basic modal language introduced earlier. Besides the propositional connectives, we have a box-operator $\Box$ to refer to all future points in time and a diamond-operator $\Diamond$ to refer to some future point. In addition to that, temporal logics often also include a so-called *next-operator* $\bigcirc$ to refer to the time point immediately after the current one. If we also wish to speak about events that happened in the past, we may choose a logic with two sets of operators, one for the future and the other one for the past. We are going to use the symbols $\overrightarrow{\Box}$, $\overrightarrow{\Diamond}$, and $\overrightarrow{\bigcirc}$ to denote future operators and $\overleftarrow{\Box}$, $\overleftarrow{\Diamond}$, and $\overleftarrow{\bigcirc}$ for past operators.

An even more expressive temporal logic may also include the additional operators SINCE and UNTIL. Intuitively, the formula $(\varphi \text{ UNTIL } \psi)$ is true now iff ψ is true at some point in the future and φ is true *until* that point in time. The operator SINCE serves a similar purpose with respect to past events. To be able to give a precise definition of the meaning of these and the other operators, we first have to decide on a particular semantics for our temporal logic. In what follows, we shall discuss *propositional linear temporal logics* over various flows of (linear) time.

[19] An overview of branching time logics (as well as references to further reading) may be found in [33].
[20] On this point, consult in particular our discussion on pages 45 and 189.

Semantics of propositional linear temporal logics. A linear temporal logic *frame* is a strict linear order $\mathcal{F} = (T, <)$.[21] The set T is called the set of *states* (or *time points*) and the relation $<$ represents the temporal order declared over these states. A temporal frame is a special case of a modal frame: T takes the role of the set of worlds, which we have previously denoted by W, and $<$ is a particular kind of accessibility relation, which we have used the letter R for in the general case. A *model* is again a pair $\mathcal{M} = (\mathcal{F}, V)$, where $\mathcal{F} = (T, <)$ is a temporal frame and V is a valuation function mapping propositional letters to sets of states in the frame. The *truth* of a temporal formula at a state $t \in T$ in a model $\mathcal{M}$ is now defined as follows:[22]

(1) $\mathcal{M}, t \models P$ iff $t \in V(P)$ for propositional letters P;

(2) $\mathcal{M}, t \models \top$;

(3) $\mathcal{M}, t \models \neg\varphi$ iff $\mathcal{M}, t \not\models \varphi$;

(4) $\mathcal{M}, t \models \varphi \wedge \psi$ iff $\mathcal{M}, t \models \varphi$ and $\mathcal{M}, t \models \psi$;

(5) $\mathcal{M}, t \models \overrightarrow{\Diamond}\varphi$ iff there exists a $t' \in T$ such that $t < t'$ and $\mathcal{M}, t' \models \varphi$;

(6) $\mathcal{M}, t \models \overrightarrow{\Box}\varphi$ iff for all $t' \in T$ with $t < t'$ we have $\mathcal{M}, t' \models \varphi$;

(7) $\mathcal{M}, t \models \overrightarrow{\bigcirc}\varphi$ iff t has an immediate successor $t' \in T$ and $\mathcal{M}, t' \models \varphi$;

(8) $\mathcal{M}, t \models \varphi$ UNTIL ψ iff there exists a $t' \in T$ with $t < t'$ and $\mathcal{M}, t' \models \psi$, and furthermore $\mathcal{M}, t'' \models \varphi$ holds for all $t'' \in T$ with $t < t''$ and $t'' < t'$.

The semantics of the past operators are defined in direct analogy to the above. For instance, we have $\mathcal{M}, t \models \varphi$ SINCE ψ iff there is a $t' \in T$ with $t' < t$ and $\mathcal{M}, t' \models \psi$, and φ holds at all states t'' in between that t' and t. The notions of *satisfiability* and *validity* of a temporal logic formula are defined in analogy to the general case discussed earlier.

Instead of a general linear order $(T, <)$ we may also choose to work with a specific *flow of time*, such as the natural numbers $(\mathbb{N}_0, <)$ or the reals $(\mathbb{R}, <)$, or we may put at least some additional constraints on $(T, <)$, such as being dense, discrete, or unbounded to either side. These choices influence what formulas are going to be valid in the respective logic. For instance, a formula of the form $\overrightarrow{\bigcirc}\varphi$ will never be true at any state if we choose $(\mathbb{R}, <)$ as the underlying flow of time, because no real number has got an immediate successor.

Definable operators. Rather than defining the semantics of the two box-operators of the temporal language directly (as we have done here), we may also choose to treat them as defined operators. As we have seen earlier, a box-operator can be defined in terms of negation and the corresponding diamond-operator:

$$\overrightarrow{\Box}\varphi \;=\; \neg\overrightarrow{\Diamond}\neg\varphi \qquad\qquad \overleftarrow{\Box}\varphi \;=\; \neg\overleftarrow{\Diamond}\neg\varphi$$

[21] Strict linear orders are defined in Appendix A.

[22] Propositional connectives other than negation and conjunction are again treated as defined operators.

In fact, now that we have enriched our language with the operators SINCE and UNTIL, both diamond- and next-operators may also be treated as defined operators:

$$\begin{array}{rclcrcl} \overrightarrow{\Diamond}\varphi & = & \top \text{ UNTIL } \varphi & \qquad & \overrightarrow{\bigcirc}\varphi & = & \bot \text{ UNTIL } \varphi \\ \overleftarrow{\Diamond}\varphi & = & \top \text{ SINCE } \varphi & \qquad & \overleftarrow{\bigcirc}\varphi & = & \bot \text{ SINCE } \varphi \end{array}$$

These syntactic definitions are easily seen to coincide with the semantic truth conditions given earlier. For instance, the first part of the semantic definition of truth for the formula ($\top$ UNTIL φ) postulates the existence of a future time point at which φ holds — this is precisely how we have defined the truth of $\overrightarrow{\Diamond}\varphi$. To understand the syntactic definition of the next-operators, consider the meaning of the formula ($\bot$ UNTIL φ). We have $\mathcal{M}, t \models \bot$ UNTIL φ iff, to begin with, there exists a state t' such that $t < t'$ and $\mathcal{M}, t' \models \varphi$. The second part of the truth condition says that $\bot$ must be true a every state that lies strictly between t and t'. But $\bot$ can never be true, i.e. there cannot be any states between t and t'. Hence, t' must be the *immediate* successor of t and $\mathcal{M}, t \models \overrightarrow{\bigcirc}\varphi$ will hold.

Expressive completeness. Our discussion shows that we can reduce the set of connectives to four core operators: negation, conjunction, SINCE and UNTIL. They form a *functionally complete* set of operators with respect to our chosen temporal language, that is, all of the other operators we have discussed here are definable in terms of these four. In classical propositional logic, we know that negation and conjunction form a functionally complete set of operators, not only with respect to the usual *language* of propositional logic (which includes, for instance, the disjunction and the implication operators), but also with respect to the logic *itself*. That is, it is not possible to even conceive an additional operator that is not expressible in terms of negation and conjunction alone. This is a direct consequence of the simple observation that negation and conjunction suffice to uniquely identify any row in a truth table.

Is there a similar phenomenon in temporal logic? Defining the notion of functional completeness for classical propositional logic is straightforward. Propositional logic is *truth-functional*, that is, a set of operators is functionally complete iff it can be used to encode every possible function over a finite number of truth values. As for temporal logic, a suitable definition of functional completeness (pertaining to the semantics of the logic rather than a particular language) is not that immediate. What would be a suitable benchmark, that is, how can we measure the expressive power of a set of temporal operators? At this point, let us recall the standard translation of modal logics into classical first-order logic discussed in Section 2.2. If we use $<$ as the binary predicate symbol representing the accessibility relation, then the formula (φ UNTIL ψ) translates into the following first-order formula (with a free variable x):

$$(\exists z)[x < z \wedge \mathrm{ST}_z(\psi) \wedge (\forall y)(x < y \wedge y < z \rightarrow \mathrm{ST}_y(\varphi))]$$

Here, $\mathrm{ST}_y(\varphi)$ and $\mathrm{ST}_z(\psi)$ represent the translations of the subformulas φ and ψ. The translation of formulas involving other operators works accordingly. That is, the translation of any temporal logic formula will always be a first-order formula with one free variable, a single binary predicate symbol $<$ together with a number of unary predicate symbols (representing the propositional letters in the input formula), and no constant or function symbols. Furthermore, $<$ has a fixed intended meaning, namely the earlier-later ordering over the chosen flow of time. In other words, temporal logic formulas are being translated into the monadic first-order theory of a particular linear order. Now the question whether we can always translate any formula of that theory *back* into temporal logic provides a useful yardstick for the expressive power of a temporal logic.

A set of operators of a temporal logic is called *expressively complete* with respect to a certain flow of time $(T, <)$ iff for every formula φ^* of the monadic first-order theory of $(T, <)$ there is a formula φ of the temporal logic such that the standard translation of φ is logically equivalent to φ^*. The concept of expressive completeness for temporal logics has been introduced by Kamp in 1968 [44]. His main result was that SINCE and UNTIL (together with the connectives for negation and conjunction) are expressively complete for linear orders $(T, <)$ that are *complete*, i.e. where every non-empty bounded subset of T has both a greatest lower bound and a least upper bound with respect to $<$. This includes, in particular, the temporal frames $(\mathbb{N}_0, <)$ and $(\mathbb{R}, <)$, but not, for instance, $(\mathbb{Q}, <)$. For arbitrary linear flows of time additional operators — the so-called *Stavi connectives* — are required to obtain expressive completeness [32]. The subject of expressive completeness is comprehensively treated in the monograph on temporal logic by Gabbay, Hodkinson, and Reynolds [30].

We shall briefly comment on issues related to expressive completeness for our modal logic of ordered trees in Chapter 6.

Temporal logic and specification. The application of temporal logics in the area of systems specification and verification is one of the great success stories of non-classical logics in mainstream computer science. In fact, in recent years, writing temporal logic based specifications and developing tools for their verification have become engineering disciplines in their own right. In a theoretical text such as this we can only hope, at best, to convey some of the basic ideas by means of example.

If we think of a *system* as a sequence of states in which certain basic propositions (say, a variable having a certain value or a certain event occurring) are either true or false, then we can identify such a system with a linear temporal logic model. A *specification* may be thought of as a list of *properties* that we would like our system to have. Two important classes of program properties — that arguably cover most properties one could wish to include into a system specification — are the so-called *safety* and *liveness* properties, which have been characterised by Lamport [47] as follows:

- Safety properties *"assert that the program does not do something bad"*.
- Liveness properties *"assert that the program does eventually do something good"*.

The latter immediately reminds us of the temporal diamond-operator: at some point in the future something good is going to happen. The former is reminiscent of the box-operator: always in the future it is not the case that things get pear-shaped. That is, a safety property can typically be expressed by a formula of the form $\boxminus\neg \mathit{Bad}$, where Bad is itself a temporal formula, while a liveness property will often be of the form $\Diamond \mathit{Good}$. Such formulas may or may not be true at (all) the states of the model representing a system, that is, such a system may or may not conform to its specification.

Examples. We conclude with a number of examples for simple specification formulas taken form a variety of sources [30, 51]. For these examples, we do not assume any specific flow of time. Our only requirement is that there is a first point in time and that time is not bounded to the right. The specified system has a particular property iff the corresponding formula is true at the first state in the model representing the system. Many of these formulas start with the box-operator $\boxminus$ to express that a particular formula is supposed to be true at every state of the system.

- *Mutual exclusion.* Two processes A and B are never going to be in their critical sections of execution at the same time:

$$\boxminus\neg(\mathit{CriticalA} \wedge \mathit{CriticalB})$$

- *Partial correctness.* After computation has finished a given postcondition will hold:

$$\boxminus(\mathit{Finished} \rightarrow \mathit{PostCondition})$$

- *Termination and total correctness.* Partial correctness together with termination yields the property of total correctness:

$$\Diamond \mathit{Finished} \wedge \boxminus(\mathit{Finished} \rightarrow \mathit{PostCondition})$$

- *Response to an impulse.* Whenever the system receives a particular impulse then it will eventually respond appropriately:

$$\boxminus(\mathit{Trigger} \rightarrow \Diamond \mathit{Response})$$

- *Absence of unsolicited response.* If at some point during program execution the system performs a certain action, then that action must have been triggered by the appropriate event:

$$\Diamond \mathit{Action} \rightarrow (\neg \mathit{Action}\ \textsc{until}\ \mathit{Trigger})$$

  This action may, for instance, be thought of as the system response from the previous example.

- *Unique events.* A particular event may happen at most once, that is, whenever it does happen it must be the case that it did not happened earlier as well:

$$\boxminus(\mathit{Event} \rightarrow \overleftarrow{\boxminus}\neg \mathit{Event})$$

2.5 Bibliographic Notes

The aim of this chapter has been to introduce some of the fundamental concepts of the study of modal and temporal logics and to indicate how they relate to the original work reported in the remainder of this thesis. Our choice of topics has been somewhat selective. Important omissions include, for instance, the whole area of deduction [4, 25, 59, 73] and particularly model checking [17, 18], first-order modal logics [26, 34, 42], and combinations of modal systems [31, 53].

Literature. The brief survey by van Benthem [70] is a very readable introduction to the field of modal logics and the forthcoming handbook chapter by Goldblatt [36] provides an extensive review of its history. Recommendable textbooks on modal logic in general include, for instance, those by Hughes and Cresswell [42], Chagrov and Zakharyaschev [13], and Blackburn, de Rijke, and Venema [8]. The latter, in particular, has proved to be an extremely useful guide throughout the preparation of this thesis. Another indispensable source of information on modal logic in general and temporal (and dynamic) logics in particular is the excellent book by Goldblatt [35]. An important book devoted specifically to the study of temporal logics is the work by Gabbay, Hodkinson, and Reynolds [30] (see also the second volume [33]).

Throughout this chapter we have borrowed from the area of modal logics of knowledge and belief, because they are particularly well suited to exemplify issues such as the need for different sets of axioms for different applications. We are not going to further continue this theme here and instead refer to the book by Fagin *et al.* [20] (and references therein) for more information on the subject. The book by Huth and Ryan [43] is a recent textbook focussing on the use of logic (and notably modal and temporal logics) in different areas of computer science. It discusses, in particular, epistemic logics for multi-agent systems and temporal logic model checking for specification and verification. The area of temporal logic specification is also treated in great breadth as well as depth in the two books on the subject by Manna and Pnueli [51, 52].

Conclusion. This concludes our general exposition of elementary modal and temporal logics. In the next chapter we are going to return to our main object of investigation, the modal logic of ordered trees.

Chapter 3

Semantic Characterisation

This chapter is mostly concerned with a semantic (as opposed to axiomatic) characterisation of our modal logic of ordered trees. We start with a formal definition of its syntax and model-theoretic semantics. After that, we are going to give a number of examples that demonstrate the expressive power of ordered tree logics. For instance, powerful operators such as the universal modality turn out to be definable within our logic. We are also going to discuss a number of correspondence results, which show that the logic is expressive enough to distinguish different classes of ordered trees (such as finite trees, for example). Thereafter, we are going to show how we can characterise different ontological categories of propositions. This allows us, for instance, to distinguish so-called properties from events, which are notions commonly referred to in the literature on planning in artificial intelligence.

In our logic, frames are required to be ordered trees. Towards the end of this chapter we investigate to what extent we can relax the conditions on *frames* without changing the *logic* (that is, without changing the set of formulas that are satisfiable with respect to these frames). For this investigation we are going to make heavy use of the concept of p-morphic images of frames.

3.1 Syntax and Semantics

In this section we are going to give formal definitions of the syntax and semantics of the modal logic of ordered trees we have already introduced informally in Chapter 1. We shall refer to this logic as **OTL**, for ordered tree logic.

Syntax. We start by defining the syntax of well-formed formulas of **OTL**. The language is built around a countable set of *propositional letters* (such as P or Q) and complex formulas can be formed by combining simpler formulas by means of a number of logical operators.

Definition 3.1 (Formulas) *The set of well-formed formulas of* **OTL** *is the smallest set such that propositional letters and the symbols* $\top$ *and* $\bot$ *are formulas and, if* φ *and*

ψ are formulas, so are $(\neg\varphi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, $(\varphi \leftrightarrow \psi)$, $(\bigcirc_{\uparrow}\varphi)$, $(\Diamond_{\uparrow}\varphi)$, $(\Box_{\uparrow}\varphi)$, $(\bigcirc_{\leftarrow}\varphi)$, $(\Diamond_{\leftarrow}\varphi)$, $(\Box_{\leftarrow}\varphi)$, $(\bigcirc_{\rightarrow}\varphi)$, $(\Diamond_{\rightarrow}\varphi)$, $(\Box_{\rightarrow}\varphi)$, $(\Diamond_{\downarrow}\varphi)$, $(\Box_{\downarrow}\varphi)$, $(\Diamond_{\downarrow}^{+}\varphi)$, *and* $(\Box_{\downarrow}^{+}\varphi)$.

The symbols $\top$ (verum) and $\bot$ (falsum) as well as the connectives $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), $\rightarrow$ (implication), and $\leftrightarrow$ (if-and-only-if) are familiar from propositional logic. As for the modal operators, we distinguish *next-operators* ($\bigcirc_{\uparrow}$, $\bigcirc_{\leftarrow}$, $\bigcirc_{\rightarrow}$), *diamond-operators* ($\Diamond_{\uparrow}$, $\Diamond_{\leftarrow}$, $\Diamond_{\rightarrow}$, $\Diamond_{\downarrow}$, $\Diamond_{\downarrow}^{+}$), and *box-operators* ($\Box_{\uparrow}$, $\Box_{\leftarrow}$, $\Box_{\rightarrow}$, $\Box_{\downarrow}$, $\Box_{\downarrow}^{+}$). Following standard convention, we are usually going to avoid redundant brackets when writing formulas.

Semantics. The semantics of formulas in **OTL** will be given by means of a definition of the notion of truth of a formula in a model. Models are based on ordered trees, i.e. the frames for this modal logic are ordered trees.

Ordered trees. An ordered tree is a tree where the children of each node are ordered. We use standard terminology when speaking about trees. In particular, we shall refer to the *parent* of a node (the node directly above a given node) and the *children* of a node (the nodes directly below a given node). The set of *ancestors* of a node includes that node's parent, the parent's parent, and so forth. Similarly, the *descendants* of a node are all the nodes on the branches below that node. Nodes sharing the same parent are called *siblings*. As we are working with ordered trees, we shall distinguish *siblings to the left* and *siblings to the right* of a given node. Finally, a node can have a *left* and a *right neighbour*, that is, a sibling coming directly before or after, respectively, in the order declared over that particular set of siblings.

Branches may be of finite or infinite length. Also, a node may have a finite or an infinite number of children. A node does not necessarily have to have a first or a last child; there could be infinitely many siblings in either direction. Also, we do *not* require the order declared over the children of a node to be, say, discrete. Instead, we admit any kind of linear order. (Discrete or dense linear orders may, however, be considered as interesting special cases.)

Next we are going to formally define the notion of ordered tree. However, later on we will often find it more convenient to refer to the intuitive notions introduced before. The following definition of a tree is a slight modification of that given by Goldblatt in [35].[1]

Definition 3.2 (Trees) *A tree is a pair $\mathcal{T} = (T, R)$, where T is a set and R is an irreflexive binary relation over T satisfying the following conditions:*

(1) For every $t \in T$ there exists at most one $t' \in T$ with $(t', t) \in R$.

[1] Here and in the sequel we are going to refer to a number of relational algebra operations. The inverse of a relation R is denoted by R^{-1}. The composition of two relations R_1 and R_2 is written as $R_1 \circ R_2$. Given a relation R, let R^* denote the reflexive transitive closure of R and R^+ the non-reflexive transitive closure of R, respectively. (See also Appendix A.)

(2) There exists a unique $r \in T$ such that $\{t \in T \mid (r,t) \in R^*\} = T$.

The elements of T are called nodes. The element r from condition (2) is called the root of $\mathcal{T}$. R is called the child relation and also gives rise to the following: the parent relation R^{-1}, the descendant relation R^+, the ancestor relation $(R^{-1})^+$, and the sibling relation $R^{-1} \circ R$.

So by referring, for instance, to a node t_1 as the parent of t_2, we mean that $(t_1, t_2) \in R$. Node t_2 is then called a child of t_1. Similarly, if $(t_1, t_2) \in R^+$ then t_1 is called an ancestor of t_2, and t_2 is called a descendant of t_1. Two nodes that share the same parent are called siblings. Starting at any node, if we first move up one level using the parent relation (that is, the inverse of the child relation) and then one level down via the child relation, we will end up at a sibling of the first node. In other words, if we have $(t_1, t_2) \in R^{-1} \circ R$ then t_1 is called a sibling of t_2 and vice versa. Nodes that do not have any children at all are called *leaves*. All other nodes of a tree are called *internal nodes*.

The nodes in a tree (the kind of tree characterised through Definition 3.2) may be considered as being 'generated' by the root node via the transitive closure of the child relation R. This excludes certain kinds of structures, such as trees (in a wider sense of the word) where the child relation could, for instance, be dense. In other words, we only consider trees of *order-type* $\mathbb{N}$, that is, any branch could be mapped to an initial segment of the natural numbers. It follows that the length of a branch will be either finite or countably infinite.

The following is a formal definition of the notions of branch and subtree. Here (and throughout) we write (T', R), where R is a binary relation declared over some superset T of T', rather than explicitly restricting R to range only over T'.

Definition 3.3 (Branches and subtrees) *Let $\mathcal{T} = (T, R)$ be a tree and $t \in T$ a node within $\mathcal{T}$. We refer to the set $\{t' \in T \mid (t', t) \in R^*\}$ as (the nodes on) the branch above t and to the tree $(\{t' \in T \mid (t, t') \in R^*\}, R)$ as the subtree generated by t.*

We are now ready to define ordered trees as trees with an additional (linear) order over the nodes. Observe that the sibling relation $R^{-1} \circ R$ is an equivalence relation over the set of nodes excluding the root. For a given node t the 'quasi-equivalence class' $[t]_{R^{-1} \circ R} = \{t' \in T \mid (t, t') \in R^{-1} \circ R\}$ is the set of siblings of t (including t itself); only if t is the root of the tree then $[t]_{R^{-1} \circ R}$ is the empty set.

Definition 3.4 (Ordered trees) *An ordered tree is a triple $\mathcal{T} = (T, R, S)$ where (T, R) is a tree, $S \subseteq R^{-1} \circ R$, and $([t]_{R^{-1} \circ R}, S)$ is a strict linear order for every $t \in T$.*

If $(t_1, t_2) \in S$ then t_1 is called a left sibling of t_2, and t_2 is called a right sibling of t_1. If furthermore $(t_1, t_2) \notin S \circ S$ then t_1 is called the left neighbour of t_2, and t_2 is called the right neighbour of t_1.

When referring to a node t in a tree $\mathcal{T}$ we will occasionally write $t \in \mathcal{T}$ instead of explicitly introducing the set of nodes T (i.e. instead of writing $t \in T$ with $\mathcal{T} = (T, R, S)$).

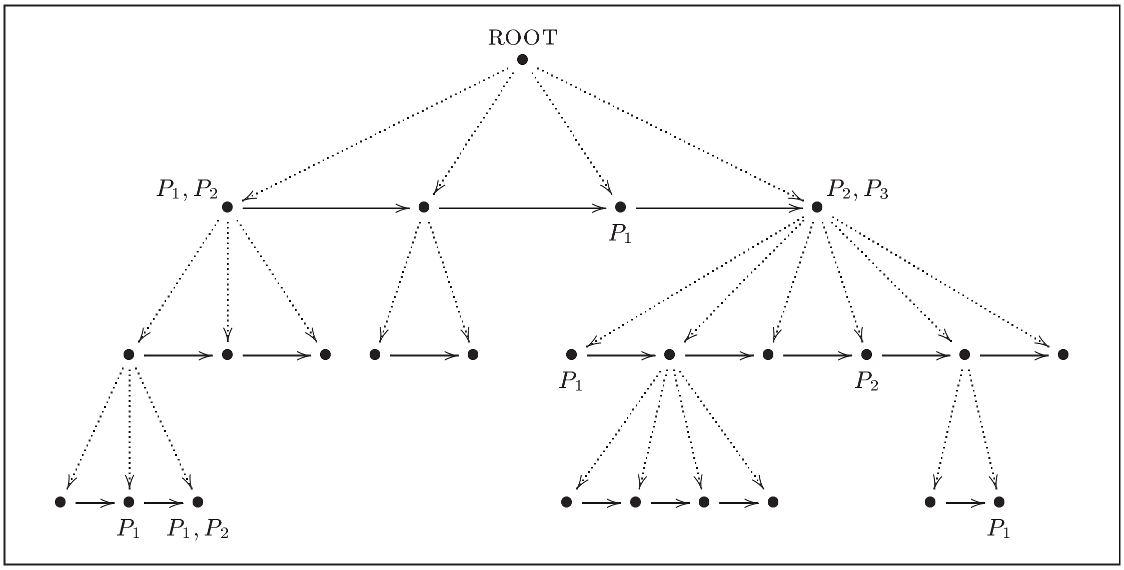

Figure 3.1: A simple ordered tree model

We will sometimes write $\prec$ for the child relation R and $<$ for the sibling order S. As before, $\prec^*$ and $\prec^+$ refer to the reflexive and the non-reflexive transitive closure of $\prec$, respectively. We also write $t_1 \leq t_2$ as an abbreviation for $t_1 < t_2$ or $t_1 = t_2$.

According to our definition of ordered trees, the horizontal relation declared over the children of a given node could be any strict linear order. In particular, it is not required to be discrete. This means, for example, that it is possible that a node has righthand siblings but no righthand neighbour, namely when that node is dense, i.e. when there are infinitely many nodes between itself and any of its righthand siblings. In other words, our logic is going to be an extension of linear temporal logics over *general* flows of time.

Models. We can now move on to the definition of models and the notion of truth of formulas in models for our logic **OTL**. Frames in this modal logic are ordered trees.

Definition 3.5 (Ordered tree models) *An ordered tree model is a pair $\mathcal{M} = (\mathcal{T}, V)$, where $\mathcal{T}$ is an ordered tree and V is a valuation, that is, a mapping from propositional letters to subsets of the set of nodes in $\mathcal{T}$.*

When speaking about models, we will sometimes simply write $\mathcal{M} = (T, R, S, V)$ as a shorthand for $\mathcal{M} = (\mathcal{T}, V)$ with $\mathcal{T} = (T, R, S)$.

Figure 3.1 shows a simple example of an ordered tree model.[2] In the picture, the child relation R is shown as a dotted line, while the right neighbour relation $S \setminus S \circ S$ is drawn as a solid line. The valuation function V is indicated by writing propositional letters (in the example these are P_1, P_2 and P_3) next to the respective nodes. The set

[2]In this particular example every node has got a finite number of children and, consequently, the order declared over sibling nodes must be discrete. However, this need not be the case in general.

$V(P_3)$, for instance, happens to be a singleton. It contains only the rightmost child of the root node.

Truth. Intuitively, we think of $V(P)$ as the set of nodes in $\mathcal{T}$ at which the atomic proposition P is true. This notion is extended to complex formulas in the following definition. (This definition only covers some of the operators of our language as the remaining ones can be considered defined operators; see Definition 3.7.)

Definition 3.6 (Truth in an ordered tree model) *We inductively define the notion of a formula being true in an ordered tree model $\mathcal{M} = (\mathcal{T}, V)$ at a node $t \in \mathcal{T}$ as follows:*

(1) $\mathcal{M}, t \models P$ *iff* $t \in V(P)$ *for propositional letters* P*;*

(2) $\mathcal{M}, t \models \top$*;*

(3) $\mathcal{M}, t \models \neg\varphi$ *iff* $\mathcal{M}, t \not\models \varphi$*;*

(4) $\mathcal{M}, t \models \varphi \wedge \psi$ *iff* $\mathcal{M}, t \models \varphi$ *and* $\mathcal{M}, t \models \psi$*;*

(5) $\mathcal{M}, t \models \bigcirc_{\uparrow}\varphi$ *iff t is not the root of $\mathcal{T}$ and* $\mathcal{M}, t' \models \varphi$ *for the parent t' of t;*

(6) $\mathcal{M}, t \models \Diamond_{\uparrow}\varphi$ *iff t has an ancestor t' with* $\mathcal{M}, t' \models \varphi$*;*

(7) $\mathcal{M}, t \models \bigcirc_{\leftarrow}\varphi$ *iff t has a left neighbour t' with* $\mathcal{M}, t' \models \varphi$*;*

(8) $\mathcal{M}, t \models \Diamond_{\leftarrow}\varphi$ *iff t has a left sibling t' with* $\mathcal{M}, t' \models \varphi$*;*

(9) $\mathcal{M}, t \models \bigcirc_{\rightarrow}\varphi$ *iff t has a right neighbour t' with* $\mathcal{M}, t' \models \varphi$*;*

(10) $\mathcal{M}, t \models \Diamond_{\rightarrow}\varphi$ *iff t has a right sibling t' with* $\mathcal{M}, t' \models \varphi$*;*

(11) $\mathcal{M}, t \models \Diamond_{\downarrow}\varphi$ *iff t has a child t' with* $\mathcal{M}, t' \models \varphi$*;*

(12) $\mathcal{M}, t \models \Diamond_{\downarrow}^{+}\varphi$ *iff t has a descendant t' with* $\mathcal{M}, t' \models \varphi$*.*

Instead of saying that φ is true at t in $\mathcal{M}$ (i.e. $\mathcal{M}, t \models \varphi$) we may also say that φ is satisfied at t or that it holds at t. In case of the truth definitions for formulas involving either a diamond- or a next-operator, we call the satisfying node t' a *witness* for the truth of the formula in question.

Going back to the example of Figure 3.1, we can put this definition of truth into practice. The formula $\Diamond_{\downarrow}P_3$, for instance, is true at the root of the tree, because it has got a child at which P_3 is true. The formula $\Diamond_{\downarrow}(P_1 \wedge P_3)$, on the other hand, does not hold at the root, because none of its children satisfies both P_1 and P_3. An example for a more complex formula that is true at the root would be $\Diamond_{\downarrow}(P_2 \wedge \Diamond_{\rightarrow}(\neg P_1 \wedge \Diamond_{\downarrow}^{+}(P_1 \wedge \bigcirc_{\leftarrow}\top)))$. Indeed, the root has got a child (its leftmost child to be precise) that satisfies P_2 and which has got a righthand sibling where P_1 does not hold (two to be precise, but only the second one also satisfies the other conditions) and which has got a descendent that satisfies P_1 and also has got a lefthand neighbour. This last node satisfying P_1 is the one shown in the lower righthand corner of the picture.

Semantics of the remaining operators. For the remaining propositional connectives and the box-operators of our language we may either directly define a semantics along the lines of Definition 3.6 or define them in terms of the operators already covered by that definition. In order to simplify proofs later on, we choose the latter way.

Definition 3.7 (Definable operators) *The remaining operators of our language are defined as abbreviations on the syntactic level:*

$$
\begin{array}{rcl@{\qquad\qquad}rcl}
\bot & = & \neg\top & \boxup\varphi & = & \neg\diamondup\neg\varphi \\
\varphi \vee \psi & = & \neg(\neg\varphi \wedge \neg\psi) & \boxleft\varphi & = & \neg\diamondleft\neg\varphi \\
\varphi \rightarrow \psi & = & \neg\varphi \vee \psi & \boxright\varphi & = & \neg\diamondright\neg\varphi \\
\varphi \leftrightarrow \psi & = & (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi) & \boxdown\varphi & = & \neg\diamonddown\neg\varphi \\
 & & & \boxdown^{+}\varphi & = & \neg\diamonddown^{+}\neg\varphi
\end{array}
$$

Given these definitions, we can infer the semantics of the remaining operators. Take, for instance, the universal ancestor modality $\boxup$, which has been defined to be a shorthand for $\neg\diamondup\neg$. A formula of the form $\boxup\varphi$ is true at a node t iff the rewritten formula $\neg\diamondup\neg\varphi$ is true at t. By Definition 3.6, the latter is the case iff it is not the case that t has got an ancestor t' such that φ is not true at t'. In other words, $\boxup\varphi$ is true at t iff φ is true at *all* ancestors of t.

For reference only, we spell out the definition of truth in an ordered tree model also for the definable operators of our language. Let $\mathcal{M} = (\mathcal{T}, V)$ be an ordered tree model and let $t \in \mathcal{T}$ be a node within that model. Then it is easily seen that the following hold as a consequence of Definitions 3.6 and 3.7:

(1) $\mathcal{M}, t \not\models \bot$;

(2) $\mathcal{M}, t \models \varphi \vee \psi$ iff $\mathcal{M}, t \models \varphi$ or $\mathcal{M}, t \models \psi$;

(3) $\mathcal{M}, t \models \varphi \rightarrow \psi$ iff $\mathcal{M}, t \models \varphi$ implies $\mathcal{M}, t \models \psi$;

(4) $\mathcal{M}, t \models \varphi \leftrightarrow \psi$ iff either both $\mathcal{M}, t \models \varphi$ and $\mathcal{M}, t \models \psi$ or neither;

(5) $\mathcal{M}, t \models \boxup\varphi$ iff $\mathcal{M}, t' \models \varphi$ for all ancestors t' of t;

(6) $\mathcal{M}, t \models \boxleft\varphi$ iff $\mathcal{M}, t' \models \varphi$ for all left siblings t' of t;

(7) $\mathcal{M}, t \models \boxright\varphi$ iff $\mathcal{M}, t' \models \varphi$ for all right siblings t' of t;

(8) $\mathcal{M}, t \models \boxdown\varphi$ iff $\mathcal{M}, t' \models \varphi$ for all children t' of t;

(9) $\mathcal{M}, t \models \boxdown^{+}\varphi$ iff $\mathcal{M}, t' \models \varphi$ for all descendants t' of t.

Satisfiability and validity. We are now ready to give definitions of the central notions of *satisfiability* and *validity* for our modal logic. A formula φ is called *satisfiable* iff there are a model and a node within that model where φ is true.

Definition 3.8 (Satisfiability) *We say a formula φ has an ordered tree model $\mathcal{M} = (\mathcal{T}, V)$ iff there is a node t in $\mathcal{T}$ such that $\mathcal{M}, t \models \varphi$ holds. A formula is called satisfiable iff it has an ordered tree model.*

Truth in the sense of Definition 3.6 is a relation between a formula, a model, and a single node in that model. A stronger requirement would be to ask for truth of a formula at *every* node within a given model. This is the concept of *global truth*.

Definition 3.9 (Global truth) *A formula φ is called globally true in an ordered tree model $\mathcal{M}$ iff it is true at every node in $\mathcal{M}$. We write $\mathcal{M} \models \varphi$.*

Instead of asking whether a given formula is true at every node in a single given model, we could also ask whether that formula is globally true in *every* model based on a particular ordered tree (or possibly even a whole class of ordered trees with certain common properties). This is the concept of *tree validity*.[3]

Definition 3.10 (Tree validity) *A formula φ is called valid in an ordered tree $\mathcal{T}$ iff it is satisfied at every node in every model based on $\mathcal{T}$. We write $\mathcal{T} \models \varphi$.*

Finally, if we also drop the reference to a particular ordered tree we get to the concept of general *validity*.

Definition 3.11 (Validity) *A formula φ is called valid iff it is satisfied at every node in every ordered tree model. We write $\models \varphi$.*

The problem of determining validity for a given formula φ can be reduced to the problem of checking whether $\neg\varphi$ is satisfiable.

Interpretation as an extended temporal logic. Our initial inspiration for devising a modal logic of ordered trees was to extend propositional linear temporal logic in a way that allows us to describe systems evolving over time in a modular fashion. Under this extended temporal logic view of **OTL**, *nodes* in a tree represent *time periods* (or *intervals*). If node t_1 is an ancestor of node t_2 then we can think of the time period corresponding to t_2 as taking place *during* that corresponding to t_1. If t_1 is a lefthand sibling of t_2 then the time period corresponding to t_1 takes place *before* that corresponding to t_2. If t_1 and t_2 are neighbours then the corresponding time periods *meet*, that is, t_1 takes place immediately before t_2. We can apply the notion of one event taking

[3] In the general modal logic literature, the corresponding concepts are known as *validity in a frame* or *validity in a class of frames*, respectively [8].

place before another also to nodes that are not siblings. If t_1 has an ancestor that is a lefthand sibling of one of the ancestors of t_2 then we also think of the period associated with t_1 as happening before that denoted by t_2. As we shall see in the next section, modal operator relating to this 'global' notions of past and future are definable in terms of our basic operators.

Section 3.4 also discusses a number of issues relevant to using **OTL** as a temporal representation formalism and an embedding of a (restricted) interval logic into an extended version of **OTL** is proposed in Chapter 6.

Interaction of vertical and horizontal operators. Apart from its potential with regard to the specification of reactive systems and their refinement, as well as other application domains that may benefit from the modular representation of processes developing over time, **OTL** is also of interest for rather more abstract reasons. The modal operators of our language fall into two groups. There are the sibling and neighbour operators pertaining to the *horizontal* dimension of an ordered tree on the one hand, and then there are the remaining ones pertaining to its *vertical* dimension on the other. The two interact in a subtle way. Suppose we consider the 'vertical fragment' and the 'horizontal fragment' of **OTL** as two separate logical systems. The former would be a modal logic over frames that are trees (without the additional sibling order) and the latter would be a modal logic over linear frames. **OTL** may then be considered a combination of the two. The study of combining different systems is an important field of activity in modal logic research [31] and provides an interesting additional perspective on our logic.

The simplest way of combining two modal logics (with disjoint sets of modal operators) would be to form an *independent join* (or *fusion*), i.e. the smallest multi-modal logic in the combined language (containing at least the union of the two logics in question). That is, in a fusion there is no interaction at all between the modal operators of the two original logics. In *products* of modal logics, on the other hand, operators from the combined logics interact in very strong manner. A typical property of a product logic is *commutativity*. A vertical and a horizontal accessibility relation are said to commute iff by first following the vertical and then the horizontal relation we can reach the same points in the model that we can reach by first following the horizontal and then the vertical relation.

In **OTL**, vertical and horizontal accessibility relations do not commute in this strong sense, but at least in some cases we can observe similar properties, albeit in a weaker form. For instance, if — starting at some node t in the middle of a tree — we first move *down* the child relation by a number of levels, then *right* along the sibling relation, and then *up* again, we will eventually get back to the node t where we started. If, on the other hand, we first move *up*, then *right*, and back *down* again at the end, we will *not* encounter the starting node t again. So, intuitively, our logic does share *some* properties with typical product logics, but it is not a product itself. In any case, the interaction of the two kinds of operators is considerably stronger than in the case of a simple fusion of two independent logics.[4]

While the fusion of two decidable multi-modal logics is known to be itself decidable, products of logics such as the purely vertical and the purely horizontal components of **OTL** are notoriously undecidable [31]. This makes **OTL** an interesting candidate to examine the question of decidability. In Chapter 5, we are going to give a positive answer to this question: **OTL** *is* in fact decidable. In the light of our discussion here, this may be considered a surprising result.

A note on the choice of symbols. Our language includes thirteen different symbols for the modal operators of **OTL** (and we are going to introduce even more in the next section). Some readers may find this a little confusing at first and may even have their objections regarding certain choices ("Why do some transitive operators have a + in the superscript while others do not?"). Our reasons for this particular choice of symbols were as follows. The first objective was to make the language simple. That means firstly using familiar symbols (symbols familiar from temporal logics) and secondly avoiding additions such as sub- or superscripts as far as possible. In the case of the modalities 'pointing' *up*, to the *left*, or to the *right* in an ordered tree the choice was easy. In all three cases we have a diamond-operator referring to a node *somewhere* in the direction indicated by the little bar, a box-operator referring to *all* nodes in that direction, and a next-operator (i.e. a circle) to refer to the (unique) *next* node in that direction (if any). So far we have a direct correspondence to the symbols used in standard temporal logics.

The dilemma begins when we have to decide on symbols for the modal operators pointing *down* in a tree. Here the concept of a next-operator does not make sense anymore, because typically there will be no unique next node when looking down a tree, but rather several children. Instead, we already need both a universal and an existential modality when referring to the immediate successors (i.e. the children) of a node. If we still wish to distinguish transitive and intransitive modalities to speak about descendents in general and children in particular, then we require two pairs of box- and diamond-operators. In view of our second objective (namely to keep the symbols themselves as simple as possible and to avoid unnecessary sub- and superscripts) the symbols $\Diamond_\downarrow$ and $\Box_\downarrow$ should be used for our language, either for the intransitive child operators or for the transitive descendant operators. Adding a + in the superscript of a given modal operator is a very simple way of indicating that this new operator is meant to relate to the transitive closure of the relation underlying the first operator. On the other hand, there is no clear and simple way to do the opposite, that is, to turn a transitive modal operator into an operator pertaining to the immediate successor relation (not function!) of the former. This is why we have decided to use $\Diamond_\downarrow$ and $\Box_\downarrow$ to refer to the children of a node and $\Diamond_\downarrow^+$ and $\Box_\downarrow^+$ to refer to descendants.

So the fact that, for instance, $\Diamond^\uparrow$ relates to a transitive relation while $\Diamond_\downarrow$ does not is

[4] We are going to provide an axiomatic characterisation of this interaction of vertical and horizontal operators in Chapter 4 (see remark on page 140).

in fact not the result of an 'asymmetric' choice of symbols, but rather a consequence of an asymmetric feature of the logic under investigation itself.

3.2 Some Defined Operators

Reflexive transitive closure. Apart from $\Box_{\downarrow}$ and $\Diamond_{\downarrow}$, all box- and diamond-operators of **OTL** correspond to transitive relations. The descendant relation, which corresponds to the modalities $\Diamond_{\downarrow}^{+}$ and $\Box_{\downarrow}^{+}$, is the (non-reflexive) transitive closure of the simple child relation underlying $\Diamond_{\downarrow}$ and $\Box_{\downarrow}$. As explained earlier, this is indicated by the $+$ in the superscript. To indicate a *reflexive* transitive closure we shall use $*$ rather than $+$. The transitive closure of a transitive relation is just that relation itself. Therefore, we can easily define modalities which correspond to reflexive transitive closures of some of our operators as abbreviations on the syntactic level.

$$\begin{array}{rcl@{\qquad\qquad}rcl}
\Diamond_{\uparrow}^{*}\varphi & = & \varphi \vee \Diamond_{\uparrow}\varphi & \Box_{\uparrow}^{*}\varphi & = & \varphi \wedge \Box_{\uparrow}\varphi \\
\Diamond_{\leftarrow}^{*}\varphi & = & \varphi \vee \Diamond_{\leftarrow}\varphi & \Box_{\leftarrow}^{*}\varphi & = & \varphi \wedge \Box_{\leftarrow}\varphi \\
\Diamond_{\rightarrow}^{*}\varphi & = & \varphi \vee \Diamond_{\rightarrow}\varphi & \Box_{\rightarrow}^{*}\varphi & = & \varphi \wedge \Box_{\rightarrow}\varphi \\
\Diamond_{\downarrow}^{*}\varphi & = & \varphi \vee \Diamond_{\downarrow}^{+}\varphi & \Box_{\downarrow}^{*}\varphi & = & \varphi \wedge \Box_{\downarrow}^{+}\varphi
\end{array}$$

The ancestor relation underlying $\Box_{\uparrow}$ and $\Diamond_{\uparrow}$ is the transitive closure of the parent relation underlying $\bigcirc_{\uparrow}$. If we required our order over siblings to be (strongly) *discrete*, then the relation corresponding to $\Box_{\leftarrow}$ and $\Diamond_{\leftarrow}$ would in fact be the transitive closure of the relation underlying $\bigcirc_{\leftarrow}$ (and accordingly for the other direction). In the general case, however, it is possible that a node has a sibling *somewhere* to the left, but no direct left neighbour.

Universal modality. Observe that, starting anywhere in a tree, the ancestor relation can take us to the tree's root, and from there the descendant relation takes us to any node in the tree. Hence, the two modalities defined next are *universal modalities*,[5] that is, operators which range over all the nodes in a tree.

$$\Diamond\varphi \;=\; \Diamond_{\uparrow}^{*}\Diamond_{\downarrow}^{*}\varphi \qquad\qquad \Box\varphi \;=\; \Box_{\uparrow}^{*}\Box_{\downarrow}^{*}\varphi$$

Suppose, for example, $\Box\varphi$ is true at a certain node t in some model $\mathcal{M}$. Then φ must be true at every node in $\mathcal{M}$, that is, φ must be globally true.

Global past and future modalities. If we further pursue the idea that has led to the definition of the universal modalities above, we can combine existing modalities in such a way as to catch not only the siblings to, say, the right of a given node t, but all the nodes that lie somewhere to the right of t anywhere in the tree. In our extended temporal logic setting, such an operator may be described as a 'global future' modality.

[5] A less frequently used but arguably more appropriate term for the universal modality is *global modality*. This avoids confusion with the use of the word 'universal' to refer to a box-operator (which has universal meaning) as opposed to a diamond-operator (which has existential meaning).

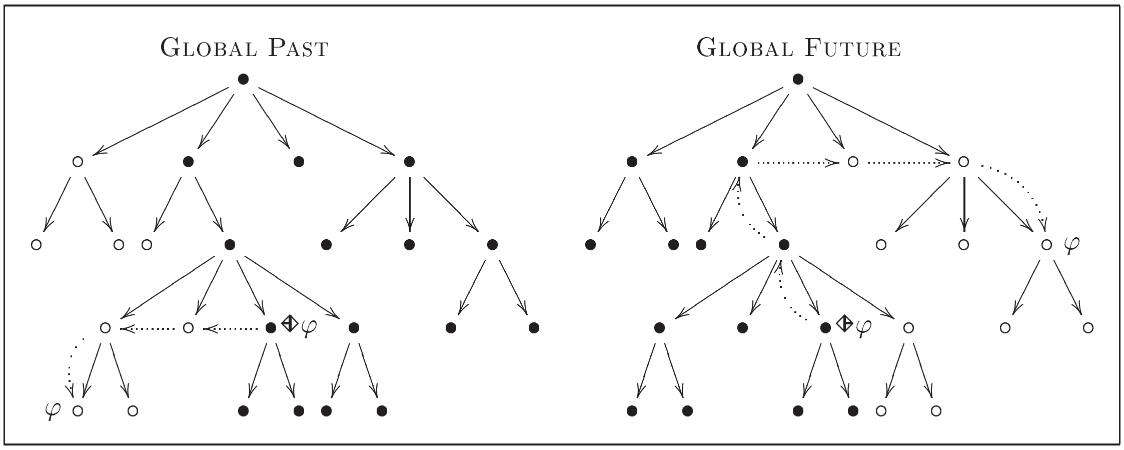

Figure 3.2: Scope of the global past and future modalities

We think of the nodes representing time points in the future of t as those nodes we can reach by first going up an unspecified number of levels in the tree (or staying at t), then making at least one step to the right (that is, moving to a righthand sibling), and finally moving down any number of levels (or possibly staying where we are).

Analogously, we can define a global past modality to refer to nodes anywhere to the left of the current node. Here are the formal definitions of a pair of global past operators $\Diamond_{\Leftarrow}$ and $\Box_{\Leftarrow}$ and corresponding global future operators $\Diamond_{\Rightarrow}$ and $\Box_{\Rightarrow}$:

$$\begin{aligned} \Diamond_{\Leftarrow}\varphi &= \Diamond_{\uparrow}^{*}\Diamond_{\leftarrow}\Diamond_{\downarrow}^{*}\varphi & \qquad \Box_{\Leftarrow}\varphi &= \Box_{\uparrow}^{*}\Box_{\leftarrow}\Box_{\downarrow}^{*}\varphi \\ \Diamond_{\Rightarrow}\varphi &= \Diamond_{\uparrow}^{*}\Diamond_{\rightarrow}\Diamond_{\downarrow}^{*}\varphi & \qquad \Box_{\Rightarrow}\varphi &= \Box_{\uparrow}^{*}\Box_{\rightarrow}\Box_{\downarrow}^{*}\varphi \end{aligned}$$

Figure 3.2 provides a graphical depiction of the scope of our global past and future operators. Take, for instance, the tree on the righthand side. The formula $\Diamond_{\Rightarrow}\varphi$ is supposed to hold at one of the nodes in the tree. To get to the witness formula φ shown in the picture, we first have to move up two levels in the tree, then move two steps to the right, and finally move down again by one level. The nodes in the scope of $\Diamond_{\Rightarrow}$ are shown as little empty circles ($\circ$), while the rest of the nodes are represented as filled circles ($\bullet$). The lefthand side of the picture shows a similar example for the case of the existential global past modality.

Difference operator. From the standpoint of a particular given node t in an ordered tree we can partition that tree into four distinct 'areas'. To start with, there is the branch *above* t, which we can refer to using our ancestor modalities $\Diamond_{\uparrow}$ and $\Box_{\uparrow}$, respectively. Then there is the subtree *below* t corresponding to the descendant modalities $\Diamond_{\downarrow}^{+}$ and $\Box_{\downarrow}^{+}$. With the global past and future operators at hand, we can now also refer to the set of nodes to the *left* and to the *right* of t, respectively.

If we combine these four pairs of operators we can define a modality to refer to any node in the tree but the current one. This kind of modality is known as the *difference operator* in the modal logic literature. Given the following definition, the formula $\Diamond_{\Leftrightarrow}\varphi$, for instance, will be true at a node t iff φ is true at some node different from t.

$$\Diamond_{\uparrow\downarrow\leftarrow\rightarrow}\varphi \;=\; \Diamond_{\uparrow}\varphi \vee \Diamond_{\downarrow}^{+}\varphi \vee \Diamond_{\leftarrow}\varphi \vee \Diamond_{\rightarrow}\varphi \qquad\qquad \Box_{\uparrow\downarrow\leftarrow\rightarrow}\varphi \;=\; \Box_{\uparrow}\varphi \wedge \Box_{\downarrow}^{+}\varphi \wedge \Box_{\leftarrow}\varphi \wedge \Box_{\rightarrow}\varphi$$

The difference operator is a very powerful construct[6] as it allows us to 'name' nodes in a tree. As an example, suppose the formula $Q \wedge \Box_{\uparrow\downarrow\leftarrow\rightarrow}\neg Q$ is known to be true at node t (in some model). Then Q is true at t but false at every other node in the tree underlying our model. That is, Q acts as a *name* (on the syntactic level) for the node t.

Iterated modalities. To express that the same modality is applied n times to a formula (with $n \in \mathbb{N}$), we can use the following notation.[7]

$$\Box_{\uparrow}^{n}\varphi \;=\; \underbrace{\Box_{\uparrow}\cdots\Box_{\uparrow}}_{n\ \text{times}}\varphi$$

If, for the sake of generality, at some point we are going to use $\Box_{\uparrow}^{n}\varphi$ for $n \in \mathbb{N}_0$, that is for natural numbers n including 0, then $\Box_{\uparrow}^{0}\varphi$ will simply stand for φ. The same notation is also used for the other modalities of our language.

Tree features. In our logic, we can easily characterise simple features of nodes in a tree, such as being the root or being a leftmost child. For example, a node is the leftmost node in a group of siblings iff it satisfies the formula $\Box_{\leftarrow}\bot$ (which is equivalent to $\neg\Diamond_{\leftarrow}\top$), i.e. iff there are no siblings to its left. The following propositions define roots, leaves, leftmost and rightmost siblings, respectively. The proposition ROOT, for example, is true at a node t iff t is the root of the tree on which our model is based.

$$\begin{array}{rclcrcl} \text{ROOT} & = & \Box_{\uparrow}\bot & \qquad & \text{LEFTMOST} & = & \Box_{\leftarrow}\bot \\ \text{LEAF} & = & \Box_{\downarrow}\bot & \qquad & \text{RIGHTMOST} & = & \Box_{\rightarrow}\bot \end{array}$$

To be precise, the propositions LEFTMOST and RIGHTMOST are also satisfied at the root of a tree.

3.3 Correspondences

In this section, we are going to give a number of examples of simple formulas that can be used to characterise certain properties of ordered trees (such as being finite or having a discrete branching factor). The study of these kinds of relationships between syntax (formulas) and semantics (structural properties of frames, or trees in our case) is known as *correspondence theory*.

[6] Blackburn *et al.* [8], for instance, refer to the difference operator as "the hammer" at the bottom of their modal logic toolbox.

[7] The set $\mathbb{N}$ of natural numbers is the set $\{1, 2, 3, \ldots\}$, that is, 0 is *not* a natural number.

Bounded branching. In the general case, the orders declared over the children of a node are not *bounded*, i.e. there need not be a leftmost or a rightmost sibling. By postulating validity of the following proposition we can force the set of children of every node in a tree to be bounded to either side:

$$\textsc{bounded} \quad = \quad \Diamond_{\leftarrow}^{*}\textsc{leftmost} \wedge \Diamond_{\rightarrow}^{*}\textsc{rightmost}$$

The formula guarantees the existence of both a leftmost and a rightmost node in every set of children. So BOUNDED is valid in a tree iff that tree is an ordered tree with a bounded branching factor. Alternatively, we could say that a tree has a bounded branching factor whenever $\Box$BOUNDED is true at some node in some model based on that tree.

Observe that bounded branching does not necessarily mean that the number of children of each node has to be finite. A particular node may, for instance, have one child for every rational number between 0 and 1.

Dense branching. Recall that the children of any given node in a tree form a (strict, but otherwise general) linear order. We may wish this linear order to be *dense*. The following formula expresses that the order declared over any set of sibling nodes in the tree underlying a model is required to be dense. No node has either a left or a right neighbour.[8]

$$\textsc{dense} \quad = \quad \neg \ominus_{\leftarrow}\top \wedge \neg \ominus_{\rightarrow}\top$$

Weakly discrete branching. We can also express that every node with a sibling to its left should have a closest left sibling (i.e. a left neighbour) and every node with a sibling to its right should have a closest right sibling. We call ordered trees satisfying this condition *weakly discrete* (see Definition A.5 and our discussion of different notions of discreteness in Appendix A).

$$\textsc{weakly-discrete} \quad = \quad (\Diamond_{\leftarrow}\top \rightarrow \ominus_{\leftarrow}\top) \wedge (\Diamond_{\rightarrow}\top \rightarrow \ominus_{\rightarrow}\top)$$

Discrete branching. We are now going to characterise trees with a *strongly discrete* branching factor (that is, between any two sibling nodes there can be only finitely many other siblings; see also Definition A.6).[9] The following proposition is an adaptation of the axiom schema (Z) used to characterise (strongly) discrete flows of time for linear temporal logics [35]. Let A be an arbitrary propositional letter of our language.[10] Then

[8] In fact, it suffices to postulate either $\neg \ominus_{\leftarrow}\top$ or $\neg \ominus_{\rightarrow}\top$ to guarantee that sibling nodes are ordered densely. If no node can have a lefthand neighbour then no node can have a righthand neighbour either (and vice versa).

[9] In the sequel, if not further specified, by a discrete order (or branching factor) we mean a strongly discrete order (branching factor), as opposed to a weakly discrete one.

[10] As will be made precise in Proposition 3.13, DISCRETE is valid in a tree iff that tree has a discrete branching factor. For formulas, such as BOUNDED, that do *not* include any propositional letters, this is equivalent to saying that the formula is globally true in an ordered tree model iff that model is based on

the following proposition is valid in a tree iff that tree is an ordered tree with a discrete branching factor:

$$\textsc{discrete} \quad = \quad \boxminus(\boxminus A \to A) \to (\diamondminus\boxminus A \to \boxminus A)$$

As it may be less evident than our previous examples, we shall formally prove this correspondence result here. First, let us fix a definition of ordered trees with a discrete branching factor.

Definition 3.12 (Discretely ordered trees) *An ordered tree $\mathcal{T} = (T, R, S)$ is called a discretely ordered tree iff $([t]_{R^{-1}\circ R}, S)$ is a strongly discrete order for every node $t \in T$.*

Now we can formally state our claim that the formula DISCRETE characterises the class of discretely ordered trees.

Proposition 3.13 (Discrete branching) *Let $\mathcal{T} = (T, R, S)$ be an ordered tree. Then $\mathcal{T} \models$ DISCRETE holds iff $\mathcal{T}$ is a discretely ordered tree.*

Proof. To simplify notation we shall write $t < t'$ for $(t, t') \in S$ and also use $t \leq t'$ as an abbreviation for $t < t'$ or $t = t'$.

'$\Leftarrow$': Suppose $<$ restricted to any set of sibling nodes is a strongly discrete strict linear order. We need to show that $\boxminus(\boxminus A \to A) \to (\diamondminus\boxminus A \to \boxminus A)$ is true at any given node t in any model $\mathcal{M}$ based on $\mathcal{T}$. So let us assume $\mathcal{M}, t \models \boxminus(\boxminus A \to A)$ (1) and $\mathcal{M}, t \models \diamondminus\boxminus A$ (2). (We need to show $\mathcal{M}, t \models \boxminus A$.)

From (1) we get $\mathcal{M}, t' \models \boxminus A \to A$ for all t' with $t < t'$ (3). From (2) we can infer that there is a t^* with $t < t^*$ and $\mathcal{M}, t^* \models \boxminus A$ (4). Applying modus ponens to (4) and (3) at node t^* yields $\mathcal{M}, t^* \models A$ (5).

Because $<$ is a strongly discrete order we can define the distance $d(t, t^*)$ of t and t^* (with $t \neq t^*$) as the number of elements between t and t^* plus one. By induction over $d(t, t^*)$, we can now show that $\mathcal{M}, t \models \boxminus A$ will be true in any case. For the base case, suppose $d(t, t^*) = 1$, i.e. t^* is the direct successor of t. From (4) and (5) we immediately obtain $\mathcal{M}, t \models \boxminus A$. For the induction step, we gradually move t forward and keep t^* fixed. So suppose $d(t, t^*) = n + 1$ (and that the claim has already been shown for distances up to n). Let t_1 be the direct successor of t, i.e. we have $d(t_1, t^*) = n$ and hence, by the induction hypothesis, $\mathcal{M}, t_1 \models \boxminus A$. Applying modus ponens to the latter and (3) at t_1 yields $\mathcal{M}, t_1 \models A$. So we get $\mathcal{M}, t \models \boxminus A$, which concludes our proof of this direction of the claim.

'$\Rightarrow$': Now suppose $\boxminus(\boxminus A \to A) \to (\diamondminus\boxminus A \to \boxminus A)$ is true at every node in every model based on $\mathcal{T}$. We need to show that $<$ restricted to any given set of siblings is a strongly discrete order. For the sake of contradiction, assume this is not the case, i.e. we have

a tree with the respective property. In the case of DISCRETE, however, if we wish to talk about models rather than trees, we need to use an instance of DISCRETE for each and every propositional letter in the language (that is, we need to use it as an axiom *schema*).

two sibling nodes t_1 and t_2 with $t_1 < t_2$ and there are infinitely many nodes t such that $t_1 < t$ and $t < t_2$. We partially fix a valuation function V as follows:

$$\begin{array}{rcl} V(A) & = & \{t \in T \mid t < t_2 \text{ and } [t, t_2] \text{ is finite}\} \,\cup\, \{t \in T \mid t_2 \leq t\} \\ & & \text{where } [t, t_2] = \{t' \in T \mid t \leq t' \text{ and } t' \leq t_2\} \end{array}$$

That is, V maps the propositional letter A to the set of those siblings of t_1 and t_2 that either lie to the right of t_2 or that are only separated from t_2 by a finite number of nodes. By definition, the set $V(A)$ will not include t_1.

Now consider the model $\mathcal{M} = (\mathcal{T}, V)$ and check whether $\mathcal{M}, t_1 \models \textsc{discete}$ holds (as it should). We have $\mathcal{M}, t \models A$ for all $t \in V(A)$ and $\mathcal{M}, t \not\models \boxminus A$ for all $t \in T$ with $t < t_2$ and $t \notin V(A)$. Hence, the implication $\boxminus A \rightarrow A$ will be true at any sibling of t_2. In particular, we get $\mathcal{M}, t_1 \models \boxminus(\boxminus A \rightarrow A)$. Because of $\mathcal{M}, t_2 \models \boxminus A$ we also get $\mathcal{M}, t_1 \models \diamondminus \boxminus A$. On the other hand, we have $\mathcal{M}, t_1 \not\models \boxminus A$, i.e. we get $\mathcal{M}, t_1 \not\models \textsc{discrete}$. This is a contradiction to the premise that $\textsc{discrete}$ is valid in $\mathcal{T}$, i.e. our assumption that $<$ restricted to any set of siblings is not discrete must be false. □

Finite branching. For many applications, trees with a finite (or even a fixed) branching factor will be sufficient. We can restrict models to those based on *finitely branching* trees by forcing the branching factor to be both bounded and strongly discrete.

$$\textsc{finite-branching} \;=\; \textsc{bounded} \wedge \textsc{discrete}$$

We can also fix a specific finite branching factor. Let $n \in \mathbb{N}$. Then the following formula characterises a tree with branching factor n:

$$\textsc{branching}(n) \;=\; \neg\textsc{leaf} \rightarrow \diamondvert(\textsc{leftmost} \wedge \ominus^{n-1}\textsc{rightmost})$$

This formula is true at an internal node (i.e. a node that is not a leaf) iff it has got a child that is a leftmost sibling which in turn has got a rightmost sibling $n-1$ steps to its right. Hence, every node that has any children at all will have exactly n children. The formula $\textsc{branching}(2)$, for instance, is valid in an ordered tree iff that tree is a binary tree. An alternative way of characterising the class of *binary trees* is given by the following formula:

$$\textsc{binary} \;=\; (\textsc{leftmost} \leftrightarrow \textsc{rightmost}) \rightarrow \textsc{root}$$

For any node that is not the root of a tree, the above formula can only be true if $\textsc{leftmost}$ and $\textsc{rightmost}$ have distinct truth values. Hence, every node but the root has to be either a leftmost or a rightmost sibling, but not both.

Blackburn and Meyer-Viol [9] have developed a modal logic of finite binary trees. As demonstrated by the authors, such trees have important applications in computational linguistics. For binary trees, we can easily define modalities to refer to the left or right child of a given node. The formula $\diamondvert(\textsc{leftmost} \wedge \varphi)$, for example, is true at a node t whenever t has got a left child at which φ is true.

Finite depth. The following is an adaptation of Blackburn and Meyer-Viol's axiom (F3). It characterises trees of *finite depth*. Again, A may be any propositional letter.

$$\textsc{finite-depth} \;=\; A \rightarrow \Diamond_{\downarrow}^{*}(A \wedge \Box_{\downarrow}^{+}\neg A)$$

We can easily check that FINITE-DEPTH is valid in trees of finite depth. Whenever a formula A is true at some node t then on any of the (finite) branches below t there must be a last occurrence of A (this could be at t itself). At that node of last occurrence (which may be a leaf node) the formula $A \wedge \Box_{\downarrow}^{+}\neg A$ is true.

For the other direction, i.e. to show that FINITE-DEPTH being valid in a tree forces that tree to be of finite depth, we shall only sketch a proof here. Let $\mathcal{T}$ be a tree with a least one infinite branch (we will show that it does not validate FINITE-DEPTH). We fix a valuation V for the propositional letter A as follows. Let A be false in the leaves of $\mathcal{T}$ as well as any internal nodes that dominate solely finite branches. For all other nodes t we declare an alternating valuation: let A be true at t iff the depth of t in the tree (that is, the distance from the root) is an even number. This means, in particular, that A is true at the root of $\mathcal{T}$ (because there must be at least one infinite branch). Then, in the model $\mathcal{M} = (\mathcal{T}, V)$, the formula FINITE-DEPTH does not hold at the root of $\mathcal{T}$: A is true at the root, but there is no node t in the tree such that A is true at t and false at every node below t.

Characterising trees that have only *infinite branches*, that is, trees without leaves is considerably easier than axiomatising finite depth. A model is based on an ordered tree without leaves iff the formula $\neg$LEAF is globally true in that model. Observe that $\neg$LEAF is equivalent to $\Diamond_{\downarrow}\top$, the seriality axiom with respect to the existential child modality.

Degenerate trees. In general, a node may have any number of children. However, for certain purposes we may regard a tree where some nodes have a *single* child as a degenerate tree. For instance, if we use **OTL** as an extended temporal logic we may run into problems when we interpret the single child t_1 of a node t_2 as a time period *during* the period associated with t_2. Such an interpretation would entail that period t_1 should be strictly shorter than t_2. On the other hand, one may wish to interpret t_1 and t_2 as the same time periods. In order to avoid problems of this kind in the first place, we may wish to restrict our attention to the class of *non-degenerate* trees, that is, to trees where every internal node has got at least two children.

The following formula is valid in a tree iff every node in that tree besides the root is not both the leftmost and the rightmost member of a set of siblings. This excludes the possibility of single children.

$$\textsc{non-degenerate} \;=\; (\textsc{leftmost} \wedge \textsc{rightmost}) \rightarrow \textsc{root}$$

3.4 Ontological Considerations

We may wish to use the logic of ordered trees to model the behaviour of a complex system changing over time. Here, time runs from left to right (that is, along the relation underlying $\oplus$), and the children of a node allow us to zoom into a particular period of time to specify the system in more detail. If we start thinking about **OTL** as a temporal representation language, we will soon encounter a number of questions and possibly even, initially, entertain doubts regarding the logic's adequacy for such an undertaking. Suppose we have φ being true at some node t. If we think of the children of t as a more fine-grained representation of the time period associated with t, then should this not imply that φ must also hold at any of the children of t?

The best possible answer to such a question would be: *it depends*. In fact, it depends on the *ontology* of φ. What kind of proposition is φ supposed to represent?

Ontological categories. Suppose, for example, that φ stands for *'Mary travelled all the way from Islington to Aldwych'*. In that case, φ may very well be true at some node t, but false at one of t's children (that is, during a time period within the one corresponding to t). In fact, we might even want to enforce $\neg\varphi$ for all descendants of t as Mary cannot complete her entire journey both over the duration of t and also in a shorter period within t.[11] If, on the other hand, φ stands for a proposition like *'Mary has long blonde hair'* then we *do* want φ to hold at descendants of t as well. In the literature on foundations of planning in artificial intelligence propositions of the latter kind have been called *properties* or *facts*, while propositions like *'Mary travelled all the way from Islington to Aldwych'* are known as *events* [2, 54, 65].[12]

Such ontological distinctions often form an integral part of the definition of a planning formalism. In Allen's work [2], for example, the definition of the predicates HOLDS and OCCUR, which are used to distinguish between different types of propositions, is central to the entire system. HOLDS is applied only to properties and OCCUR is reserved for the use with occurrences (an ontological category that includes events). While Allen's approach has undoubtedly been one of the most influential in this area, it has also received a certain amount of criticism. Shoham [65], in particular, argues against the "seductive unclarity" of a logic with this kind of built-in ontology. He proposes that, in the first instance, one should rather not make any specific commitments to ontology, but then allow for the definition of a fine ontological structure on top of the basic formalism.

This is also the approach we have taken in the definition of our logic. Propositions in **OTL** are just that: propositions. Initially, we do not associate them with any specific ontological category (such as properties or events). However, certain ontological

[11] We should stress that we are not concerned with the grammatical tense of the verb *'travelled'* here. The proposition *'Mary travelled all the way from Islington to Aldwych'* being true at node t is interpreted as Mary completing her journey during the time period t, not during some period in the past of t.

[12] In the area of software engineering, the corresponding ontological categories are sometimes referred to as *states* and *actions*, respectively [23, 46].

categories can be defined *within* the logic. In this section we give a number of examples.

Homogeneity. According to the semantics given for **OTL**, a formula φ being satisfied at some node t does not imply that φ also has to hold at all nodes that are below t in the tree. Whether this is appropriate or not depends on the kind of proposition represented by the formula φ.

Propositions like *'Mary has long blonde hair'*, i.e. properties, are *homogeneous*: 'temporally' speaking, a homogeneous proposition φ is true iff it is true at every time interval within the reference interval. Following Shoham [65] we define homogeneous propositions as propositions that are both *downward-hereditary* and *upward-hereditary*. A proposition is called downward-hereditary if whenever it holds at some node it also holds at all descendants of that node. Analogously, a proposition is called upward-hereditary if whenever it holds at all descendants of a given node it also holds at that node itself (assuming that node has any descendants at all).

$$\begin{array}{rcl}
\textsc{down-hered}(\varphi) & = & \varphi \rightarrow \boxdot^{+}\varphi \\
\textsc{up-hered}(\varphi) & = & \Diamond\top \rightarrow (\boxdot^{+}\varphi \rightarrow \varphi) \\
\textsc{hom}(\varphi) & = & \textsc{down-hered}(\varphi) \wedge \textsc{up-hered}(\varphi)
\end{array}$$

So φ is a homogeneous proposition (with respect to a given model) iff $\textsc{hom}(\varphi)$ is globally true in that model.

Strong negation. Negation as we have defined it here is *weak* in the sense that $\neg\varphi$ being satisfied at a particular node simply means that it is not the case that φ is satisfied at that node. It does, however, not preclude φ from being satisfied at some of the descendants of that node. For propositions corresponding to certain kinds of ontological categories the notion of *strong negation* may be more appropriate. We can define a strong negation operator as follows:

$$\sim\varphi \quad = \quad \boxdot^{*}\neg\varphi$$

The formula $\sim\varphi$ is true at a node t iff φ is false at t and all nodes that are descendants of t. What kind of properties does this new operator have? The following proposition, for instance, gives us a restricted version of de Morgan-like laws for strong negation. We may rewrite $\sim\varphi \wedge \sim\psi$ as $\sim(\varphi \vee \psi)$ and vice versa. We may also rewrite $\sim\varphi \vee \sim\psi$ as $\sim(\varphi \wedge \psi)$, but not the other way round.

Proposition 3.14 (Restricted de Morgan laws for strong negation) *The following hold for all formulas φ and ψ:*

(1) $\models \sim\varphi \wedge \sim\psi \leftrightarrow \sim(\varphi \vee \psi)$

(2) $\models \sim\varphi \vee \sim\psi \rightarrow \sim(\varphi \wedge \psi)$

Proof. (1) We show that the formulas $\sim\varphi \wedge \sim\psi$ and $\sim(\varphi \vee \psi)$ are equivalent: $\sim\varphi \wedge \sim\psi \;\equiv\; \boxdot^*\neg\varphi \wedge \boxdot^*\neg\psi \;\equiv\; \boxdot^*(\neg\varphi \wedge \neg\psi) \;\equiv\; \boxdot^*\neg(\varphi \vee \psi) \;\equiv\; \sim(\varphi \vee \psi)$.

(2) Suppose $\sim\varphi \vee \sim\psi$, which is equivalent to $\boxdot^*\neg\varphi \vee \boxdot^*\neg\psi$, is true. Then also $\boxdot^*(\neg\varphi \vee \neg\psi)$ has to be true. The latter is equivalent to $\boxdot^*\neg(\varphi \wedge \psi)$, which in turn is equivalent to $\sim(\varphi \wedge \psi)$. □

To see that indeed $\sim\varphi \vee \sim\psi$ and $\sim(\varphi \wedge \psi)$ are not equivalent, i.e. that the opposite direction of (2) would not be valid, consider the following counterexample. Suppose we have a tree consisting only of a root node with a single child. Further suppose that φ holds at the root, but not the child, and that ψ holds at the child node, but not at the root. Then $\sim(\varphi \wedge \psi)$ is clearly true at the root, because at no node both φ and ψ are true. However, $\sim\varphi \vee \sim\psi$ does *not* hold at the root, because for neither φ nor for ψ it is the case that they do not hold at any of the two nodes.

The following result suggests that $\sim$ may be the appropriate negation operator to be applied to homogeneous propositions.

Proposition 3.15 (Strong negation of homogeneous propositions) *The strong negation of a homogeneous proposition φ is itself homogeneous, that is, global truth of* HOM(φ) *implies global truth of* HOM$(\sim\varphi)$.

Proof. We start by showing that the strong negation of *any* proposition (that is, not just that of homogeneous ones) is downward-hereditary. It is easy to see that $\boxdot^*\neg\varphi \rightarrow \boxdot^+\boxdot^*\neg\varphi$ is a valid formula. This is equivalent to DOWN-HERED$(\sim\varphi)$, which means that the strong negation of any formula φ will be downward-hereditary.

Next we prove that $\sim\varphi$ is upward-hereditary under the assumption that φ is homogeneous. We need to show the global truth of UP-HERED$(\sim\varphi)$, i.e. that given $\Diamond\top$ and $\boxdot^+\boxdot^*\neg\varphi$ at some node t we can infer that $\boxdot^*\neg\varphi$ is also true at t. Observe that $\boxdot^+\boxdot^*\neg\varphi$ is equivalent to $\boxdot^+\neg\varphi$, so we only need to show that we can infer $\neg\varphi$. For the sake of contradiction, suppose that φ is true at t. Because t has descendants (enforced by $\Diamond\top$) and because φ is homogeneous, t will have descendants satisfying φ. This contradicts $\boxdot^+\neg\varphi$. Hence, both $\neg\varphi$ and $\boxdot^+\neg\varphi$, and therefore also $\boxdot^*\neg\varphi$, need to be true at t. This completes our proof of Proposition 3.15. □

Strong double negation. We have $\sim\sim\varphi \;\equiv\; \boxdot^*\neg\boxdot^*\neg\varphi \;\equiv\; \boxdot^*\Diamond^*\varphi$. As $\boxdot^*\varphi$ implies $\boxdot^*\Diamond^*\varphi$ it is easy to see that the formula $\varphi \rightarrow \sim\sim\varphi$ must be true for homogeneous propositions φ.

The converse, however, does not hold, that is, we cannot eliminate a strong double negation, not even for homogeneous propositions. To see this, consider the following counterexample. Let $\mathcal{T}$ be an infinite binary tree where every right child is a leaf (i.e. there is one long branch to the left with many short ones to the right). Let φ be the only propositional letter in our language and define the ordered tree model $\mathcal{M} = (\mathcal{T}, V)$, with $V(\varphi)$ being the set of all right children. Then $\boxdot^*\Diamond^*\varphi$ (and thereby $\sim\sim\varphi$) is true in the root of $\mathcal{T}$, but φ is not.

Finally, it is worth noting that over *trees of finite depth* the formula $\sim\sim\varphi \rightarrow \varphi$ actually *is* globally true for homogeneous propositions φ.[13] This can be shown by induction over the height of a node in a tree. For the base case, assume we are in a leaf node. Then $\sim\sim\varphi$, which is equivalent to $\boxdot^{*}\Diamonddot^{*}\varphi$, directly implies φ. For the induction step, assume we are in a node t of height $n + 1$, that is, all branches below t have at most length n. Let $\sim\sim\varphi$ be true at t. By Proposition 3.15, $\sim\sim\varphi$ is homogeneous, i.e. $\sim\sim\varphi$ also holds in every node below t. Now, by the induction hypothesis, we can conclude that φ must be true at every node below t. As φ is homogeneous, this means that φ must also hold at t itself.

Comparison with Allen's system. It is interesting to compare our discussion of homogeneity and strong negation with Allen's HOLDS predicate used to characterise *properties* in [2]. Allen's definition of strong negation is essentially the same as ours and his properties roughly correspond to our homogeneous propositions. What is different is that the strong negation operator of [2] is involutive, that is, elimination of strong double negation *is* a valid operation for those 'homogeneous' propositions. In fact, Allen characterises his properties as precisely those propositions φ that validate $\varphi \leftrightarrow \boxdot^{+}\Diamonddot^{+}\varphi$ (translated into our formalism). That is, a proposition φ is called a property (a 'homogeneous' proposition) iff φ being true at t is equivalent to φ being true infinitely many times on every branch below t. In Allen's setting, where every interval has subintervals (every node has children, in our formalism), this is a *stronger* requirement[14] than just postulating the equivalence of φ and $\boxdot^{+}\varphi$ (see [2] for a proof). As Allen defines the strong negation of a property φ as $\boxdot^{+}\neg\varphi$ (i.e. $\sim\sim\varphi$ is equivalent to $\boxdot^{+}\Diamonddot^{+}\varphi$), we immediately get $\sim\sim\varphi \equiv \varphi$.

Strict implication. Our implication operator $\rightarrow$ has classical semantics: $\varphi \rightarrow \psi$ is true at a node t whenever φ is false at t or ψ is true at t. We may also wish to employ the concept of *strict* implication: a proposition φ strictly implies another proposition ψ (at a node t) iff whenever φ is true at a descendent t' of t (including t itself) then also ψ is true at t'. Such a strict implication operator $\Rightarrow$ can be defined as follows:

$$\varphi \Rightarrow \psi \quad = \quad \boxdot^{*}(\varphi \rightarrow \psi)$$

The connection between strong negation and strict implication is summarised in the equivalence $\sim\varphi \equiv \varphi \Rightarrow \bot$, the validity of which is readily verified.

Connections to intuitionistic logic. In the proof of Proposition 3.15 it has been noted that $\sim\varphi$ is a downward-hereditary formula for any formula φ. By the same argument, we can show that also $\varphi \Rightarrow \psi$ (strict implication) is downward-hereditary for any φ and ψ. Furthermore, it is easily observed that both $\varphi \wedge \psi$ and $\varphi \vee \psi$ are

[13] This observation is due to Naza Aguirre (personal communication).

[14] See also Shoham's critique of Allen's definition of the HOLDS predicate in [65].

downward-hereditary whenever φ and ψ are. In other words, the set of downward-hereditary propositions is closed under application of $\sim$, $\wedge$, $\vee$, and $\Rightarrow$.

At this point it is interesting to observe that, for downward-hereditary propositions, the $\{\sim, \wedge, \vee, \Rightarrow\}$-fragment of our logic coincides with propositional intuitionistic logic. This can be seen by comparison with the definition of the possible worlds semantics for intuitionistic logic as given, for instance, in [72].[15] In particular, observe that the law of the excluded middle does not hold for the strong negation operator $\sim$. Formulas of the form $\sim \varphi \vee \varphi$ are not necessarily globally true in all models, even when φ is downward-hereditary. A simple counterexample would be any model where $\neg\varphi$ holds at one node but φ is true at one of its children.

Other ontological categories. The notion of homogeneity has been discussed in some detail here as it is probably the most important ontological concept in the context of temporal reasoning. We are going to give two more examples of ontological categories.

The first one concerns the characterisation of events. An event, that is, a proposition like our *'Mary travelled all the way from Islington to Aldwych'* should not hold over two time periods one of which properly contains the other. Shoham [65] calls propositions of this type *gestalt*.[16] In the context of a tree, we can interpret this condition as follows. A proposition φ is gestalt iff whenever φ is true at a node t then it must be false at all ancestors and descendants of t.

$$\textsc{gestalt}(\varphi) \quad = \quad \varphi \;\rightarrow\; (\Box_{\uparrow}\neg\varphi \,\wedge\, \Box_{\downarrow}^{+}\neg\varphi)$$

Another ontological concept that can be expressed is locality. A proposition φ is called *local* if φ is true during a particular time period iff it is true at the beginning of that period [37]. In our system, the beginning of a time period may be identified with the leftmost child of a node. The following definition takes into account that a node does not necessarily have to have a leftmost child (although we may wish to postulate that property independently, as outlined in the previous section).

$$\textsc{local}(\varphi) \quad = \quad (\Diamond_{\downarrow}\textsc{leftmost} \wedge \varphi) \;\leftrightarrow\; \Diamond_{\downarrow}(\textsc{leftmost} \wedge \varphi)$$

What we have seen here are just examples. Hopefully, they give some impression of the variety of options available to us when using **OTL** as a temporal representation formalism.

[15] The semantics of intuitionistic logic can be defined via possible world models with frames that are partial orders and persistent valuation functions (that is, a propositional letter that is true at some world cannot be false at a later world). The aspect of persistence is captured by our restriction to downward-hereditary propositions. Also observe that the accessibility relation R^* underlying the modality $\Box_{\downarrow}^{*}$ is a partial order. The fact that R^* is generated by a tree is not a restriction, because in the definition of the semantics of intuitionistic logic the accessibility relation is only used 'in one direction'.

[16] Related to this, Shoham also defines the class of *solid* propositions, that is, propositions that cannot be true at overlapping time intervals. This would provide an alternative way to characterise events, but the notion of overlapping intervals cannot be represented in **OTL**.

3.5 General Models and P-morphisms

In this and the following section we are going to take a closer look at the semantics of **OTL**. At the beginning of this chapter we have defined our logic as a modal logic over frames that are ordered trees and we have given the various modal operators of our language a semantics by referring to intuitive notions such as 'left sibling' or 'parent'. Taking a somewhat broader view, we could also think of **OTL** as a multi-modal logic over frames with two basic accessibility relations R (the child relation) and S (the order declared over siblings). The meaning of our modal operators can then be defined in terms of these basic relations and a number of compound accessibility relations constructed from R and S.

In our intended semantics, the relation R is required to characterise a tree and S is required to be a linear order for each and every set of sibling nodes in that tree. Loosening these constraints on R and S *slightly* may not always change the logic, in the sense that the same formulas as before may be those that have models. In what follows, we are going to make precise this idea of widening the class of admissible frames.[17] Our main technical tool will be certain types of mappings between frames that are known as *p-morphisms*.

General models. Our first aim will be to give an alternative definition of the semantics of our logic, namely as a multi-modal logic with a number of compound accessibility relations based on the basic relations R and S. To this end, we are going to introduce the notion of a *general model* where we drop the constraints on the basic accessibility relations R and S completely. This will provide us with a basis upon which we are later going to define manipulations of the accessibility relations that are satisfiability preserving.

Next-operators with general models? Our modal language includes a number of next-operators. Defining the semantics of formulas with next-operators over general models is problematic, because it is not clear how to generalise the notion of a next state when moving from orders with unique immediate successors to general relations.

Rather than attempting to generalise this notion, we will simply think of next-operators as ordinary diamond-operators, that is, we do not generally require the underlying accessibility relation to be functional. For example, in our intended models (models based on ordered trees) the parent relation R^{-1} is functional, that is, any node can have at most one parent. For the general models, we will drop this condition and treat the

[17]At this point it maybe worthwhile to point out that the material of this and the following section is not only intended to provide us with a deeper understanding of the model-theoretic semantics of **OTL**, but will also be required to prove completeness for the axiomatisation discussed in Chapter 4. In fact, the aforementioned 'wider class of admissible frames' (which we are going to give a precise definition for later on, cf. *loosely ordered trees*) is precisely the class of frames we encountered during our initial efforts to prove completeness for a preliminary set of axioms.

parent modality ⦶ as an existential modality (i.e. like a diamond-operator) over R^{-1}.

In our intended models the accessibility relation underlying ⊖ (the right sibling relation) is definable in terms of S, namely as $S \setminus S \circ S$. This definition makes sense, because we know that S is not branching to the right. Hence, it captures the notion of an immediate successor. In the general setting, where S could be any binary relation, such a definition would seem rather arbitrary and we shall not adopt it. Instead, we will introduce a third accessibility relation N, which only in intended models will be required to coincide with the immediate successor relation induced by S.

Definition 3.16 (General frames) *A general frame is a quadruple $\mathcal{F} = (T, R, S, N)$ where T is a non-empty set, and R, S and N are binary relations over T. The elements of T are called worlds.*

As for the case of ordered trees before, we shall occasionally simply refer to worlds t in a frame $\mathcal{F}$ rather than explicitly introducing a set of worlds T for that frame first.

When speaking about an *intended frame* we shall mean a frame that corresponds to an ordered tree. In other words, a general frame $\mathcal{F} = (T, R, S, N)$ is called an intended frame if (T, R, S) is an ordered tree and $N = S \setminus S \circ S$, that is, for every node in the tree N points to its immediate righthand successor with respect to the linear order S, whenever there is such a successor.

Definition 3.17 (General models) *A general model is a pair $\mathcal{M} = (\mathcal{F}, V)$ where $\mathcal{F} = (T, R, S, N)$ is a general frame and V is a valuation, that is, a mapping from propositional letters to subsets of the set of worlds T.*

Again, we may either write $\mathcal{M} = (\mathcal{F}, V)$ with $\mathcal{F} = (T, R, S, N)$ or directly $\mathcal{M} = (T, R, S, N, V)$. In Definition 3.18 we are going to fix the notion of truth for general models. The semantics of our various modalities will be defined with respect to (possibly complex) accessibility relations depending on R, S and N. The relation S^{-1}, for example, will be used to refer to those worlds that, within an ordered tree model, we would call left siblings.

The following definition should be read in conjunction with Definition 3.6, where the notion of truth for our *intended* models has been defined.

Definition 3.18 (Truth in a general model) *We inductively define the notion of a formula being true in a general model $\mathcal{M} = (\mathcal{F}, V)$ with $\mathcal{F} = (T, R, S, N)$ at a world $t \in T$ as follows:*

(1) $\mathcal{M}, t \models P$ *iff* $t \in V(P)$ *for propositional letters* P;

(2) $\mathcal{M}, t \models \top$;

(3) $\mathcal{M}, t \models \neg\varphi$ *iff* $\mathcal{M}, t \not\models \varphi$;

(4) $\mathcal{M}, t \models \varphi \wedge \psi$ *iff* $\mathcal{M}, t \models \varphi$ *and* $\mathcal{M}, t \models \psi$;

(5) $\mathcal{M}, t \models \bigcirc\!\!\!\!\!\uparrow\,\varphi$ *iff there is a* $t' \in T$ *with* $(t, t') \in R^{-1}$ *and* $\mathcal{M}, t' \models \varphi$*;*

(6) $\mathcal{M}, t \models \Diamond\!\!\!\!\uparrow\,\varphi$ *iff there is a* $t' \in T$ *with* $(t, t') \in (R^{-1})^{+}$ *and* $\mathcal{M}, t' \models \varphi$*;*

(7) $\mathcal{M}, t \models \bigcirc\!\!\!\!\!\leftarrow\,\varphi$ *iff there is a* $t' \in T$ *with* $(t, t') \in N^{-1}$ *and* $\mathcal{M}, t' \models \varphi$*;*

(8) $\mathcal{M}, t \models \Diamond\!\!\!\!\leftarrow\,\varphi$ *iff there is a* $t' \in T$ *with* $(t, t') \in S^{-1}$ *and* $\mathcal{M}, t' \models \varphi$*;*

(9) $\mathcal{M}, t \models \bigcirc\!\!\!\!\!\rightarrow\,\varphi$ *iff there is a* $t' \in T$ *with* $(t, t') \in N$ *and* $\mathcal{M}, t' \models \varphi$*;*

(10) $\mathcal{M}, t \models \Diamond\!\!\!\!\rightarrow\,\varphi$ *iff there is a* $t' \in T$ *with* $(t, t') \in S$ *and* $\mathcal{M}, t' \models \varphi$*;*

(11) $\mathcal{M}, t \models \Diamond\!\!\!\!\downarrow\,\varphi$ *iff there is a* $t' \in T$ *with* $(t, t') \in R$ *and* $\mathcal{M}, t' \models \varphi$*;*

(12) $\mathcal{M}, t \models \Diamond\!\!\!\!\downarrow^{+}\varphi$ *iff there is a* $t' \in T$ *with* $(t, t') \in R^{+}$ *and* $\mathcal{M}, t' \models \varphi$*.*

Observe that in case the underlying frame $\mathcal{F} = (T, R, S, N)$ is such that (T, R, S) is an ordered tree and $N = S \setminus S \circ S$, the above notion of truth coincides with the one originally formulated in Definition 3.6. The remaining operators of our language can again be considered defined operators; see Definition 3.7.

While we are going to reserve the term 'satisfiable' for formulas that have an ordered tree model, we are going to speak about formulas that are satisfiable in a general model based on some frame belonging to a certain class of frames.

Definition 3.19 (Satisfiability in general models) *We say a formula φ is satisfiable in a model based on a frame belonging to some class of general frames iff there is a general model $\mathcal{M} = (\mathcal{F}, V)$ based on a frame $\mathcal{F}$ from that class with a world t in $\mathcal{F}$ such that $\mathcal{M}, t \models \varphi$ holds.*

The concepts of global truth, validity with respect to a particular frame, and validity in general can be defined in analogy to Definitions 3.9, 3.10 and 3.11, respectively.

P-morphisms. The notion of *p-morphism* (or bounded morphism) plays a central role in the study of modal logics [8, 13, 35]. It provides us with a powerful tool to describe satisfiability preserving transformations between models. Further below we will state the definition of p-morphisms for our particular modal logic with three accessibility relations, all of which are used 'in both directions'. We start by introducing the concept of p-morphism for pairs of a domain and a binary relation declared over that domain.

Definition 3.20 (P-morphisms for relational structures) *Let T and T' be sets and let R be a binary relation over T and R' a binary relation over T'. A function $f : T \rightarrow T'$ is called a (simple) p-morphism from (T, R) to (T', R') iff it satisfies the following conditions:*

(1) if $(t_1, t_2) \in R$ *then* $(f(t_1), f(t_2)) \in R'$*;*

(2) if $(f(t_1), t'_2) \in R'$ *then there is a* $t_2 \in T$ *with* $f(t_2) = t'_2$ *and* $(t_1, t_2) \in R$*.*

We call f a bidirectional p-morphism, if in addition to the above it also satisfies the following condition:

(3) if $(t'_1, f(t_2)) \in R'$ then there is a $t_1 \in T$ with $f(t_1) = t'_1$ and $(t_1, t_2) \in R$.

The first of these conditions is sometimes called the *homomorphic condition* as it guarantees f to be a homomorphism. The second condition has been called the *back condition* in [8] and the third one may be called the *inverse back condition.* If in the above definition the p-morphism f is a surjective function then (T', R') is called the *p-morphic image* of (T, R).

Next we are going to introduce the notion of p-morphism for our general frames. In short, we call a function a p-morphism from one frame to another if that function is a bidirectional p-morphism with respect to the three accessibility relations of a general frame.

Definition 3.21 (P-morphisms for frames) *Let $\mathcal{F} = (T, R, S, N)$ and $\mathcal{F}' = (T', R', S', N')$ be general frames. A function $f : T \to T'$ is called a p-morphism from $\mathcal{F}$ to $\mathcal{F}'$ iff it satisfies the following conditions:*

(1) f is a bidirectional p-morphism from (T, R) to (T', R');

(2) f is a bidirectional p-morphism from (T, S) to (T', S');

(3) f is a bidirectional p-morphism from (T, N) to (T', N').

We are not going to distinguish simple and bidirectional p-morphisms for frames. When speaking about a p-morphism in the context of a general frame we shall always mean a bidirectional p-morphism. Again, if the p-morphism f is a surjective function from T onto T', then the frame $\mathcal{F}'$ is called the *p-morphic image* of $\mathcal{F}$.

Finally, we extend the notion of p-morphisms to general models.

Definition 3.22 (P-morphisms for models) *Let $\mathcal{M} = (\mathcal{F}, V)$ and $\mathcal{M}' = (\mathcal{F}', V')$ be general models with $\mathcal{F} = (T, R, S, N)$ and $\mathcal{F}' = (T', R', S', N')$. A function $f : T \to T'$ is called a p-morphism from $\mathcal{M}$ to $\mathcal{M}'$ iff it satisfies the following conditions:*

(1) f is a p-morphism from $(\mathcal{F}, R)$ to $(\mathcal{F}', R')$;

(2) $t \in V(P)$ iff $f(t) \in V'(P)$ for all $t \in T$ and all propositional letters P.

Also in the case of models, if there exists a p-morphism f from $\mathcal{M}$ to $\mathcal{M}'$ that is a surjective function then we say that $\mathcal{M}'$ is a *p-morphic image* of $\mathcal{M}$.

As will be stated as Theorem 3.24 below, the truth of a formula φ in a general model $\mathcal{M}$ is invariant under p-morphisms. The proof of that theorem will make use of the following lemma.

Lemma 3.23 (P-morphisms for compound relations) *Let f be a bidirectional p-morphism from (T, R_1) to (T', R'_1) and from (T, R_2) to (T', R'_2). Then also the following are true:*

(1) f is a bidirectional p-morphism from (T, R_1^{-1}) to $(T', R_1'^{-1})$;

(2) f is a bidirectional p-morphism from $(T, R_1 \cup R_2)$ to $(T', R'_1 \cup R'_2)$;

(3) f is a bidirectional p-morphism from $(T, R_1 \circ R_2)$ to $(T', R'_1 \circ R'_2)$;

(4) f is a bidirectional p-morphism from (T, R_1^n) to $(T', R_1'^n)$ for all $n \in \mathbb{N}$;

(5) f is a bidirectional p-morphism from (T, R_1^+) to $(T', R_1'^+)$;

(6) f is a bidirectional p-morphism from (T, R_1^*) to $(T', R_1'^*)$.

Proof. For each of the six statements we have to show that the three conditions for p-morphisms formulated in Definition 3.20 also hold for the complex relation in question.

(1) *Inverse.* (1) If $(t_1, t_2) \in R_1^{-1}$ then $(t_2, t_1) \in R_1$, which implies $(f(t_2), f(t_1)) \in R'_1$ (because f is a p-morphism with respect to R_1), which in turn implies $(f(t_1), f(t_2)) \in R_1'^{-1}$.

(2) If $(f(t_1), t'_2) \in R_1'^{-1}$ then $(t'_2, f(t_1)) \in R'_1$, i.e. there exists a $t_2 \in T$ such that $f(t_2) = t'_2$ and $(t_2, t_1) \in R_1$, which implies $(t_1, t_2) \in R_1^{-1}$.

Condition (3) is dual to condition (2) and can be checked analogously.

(2) *Union.* (1) $(t_1, t_2) \in R_1 \cup R_2$ implies either $(t_1, t_2) \in R_1$ or $(t_1, t_2) \in R_2$. That is, we get $(f(t_1), f(t_2)) \in R_1$ or $(f(t_1), f(t_2)) \in R_2$, hence the desired result $(f(t_1), f(t_2)) \in R_1 \cup R_2$.

(2) Let $(f(t_1), t'_2) \in R'_1 \cup R'_2$, i.e. either $(f(t_1), t'_2) \in R'_1$ or $(f(t_1), t'_2) \in R'_2$. Hence, there must be a $t_2 \in T$ with $f(t_2) = t'_2$ satisfying either $(t_1, t_2) \in R_1$ or $(t_1, t_2) \in R_2$, and thereby also $(t_1, t_2) \in R_1 \cup R_2$.

Condition (3) is checked analogously.

(3) *Composition.* (1) Let $(t_1, t_3) \in R_1 \circ R_2$, i.e. there is a $t_2 \in T$ such that $(t_1, t_2) \in R_1$ and $(t_2, t_3) \in R_2$ hold. Because f is a p-morphism for both R_1 and R_2 we get $(f(t_1), f(t_2)) \in R'_1$ and $(f(t_2), f(t_3)) \in R'_2$, hence $(f(t_1), f(t_3)) \in R'_1 \circ R'_2$.

(2) Let $(f(t_1), t'_3) \in R'_1 \circ R'_2$ i.e. there is a $t'_2 \in T'$ such that $(f(t_1), t'_2) \in R'_1$ and $(t'_2, t'_3) \in R'_2$ hold. Because f is a p-morphism for R_1, there is a $t_2 \in T$ with $f(t_2) = t'_2$ and $(t_1, t_2) \in R_1$. As we also obtain $(f(t_2), t'_3) \in R'_2$ and f is a p-morphism with respect to R_2, there has to be a $t_3 \in T$ with $f(t_3) = t'_3$ and $(t_2, t_3) \in R_2$. Finally, we therefore get $(t_1, t_3) \in R_1 \circ R_2$.

Condition (3) is checked analogously.

(4) *Iteration.* Given the result for the composition of relations, it can easily be shown by complete induction that f is a bidirectional p-morphism from (T, R_1^n) to $(T, R_1'^n)$ for any $n \in \mathbb{N}$.

(5) *Transitive closure.* Checking the three p-morphism conditions for the transitive closure is now straightforward:

(1) Let $(t_1, t_2) \in R_1^+$, that is, there exists an $n \in \mathbb{N}$ such that $(t_1, t_2) \in R_1^n$. But then also $(f(t_1), f(t_2)) \in R_1'^n$ for the same n, i.e. $(f(t_1), f(t_2)) \in R_1'^+$.

(2) Let $(f(t_1), t_2') \in R_1'^+$, i.e. $(f(t_1), t_2') \in R_1'^n$ for some $n \in \mathbb{N}$. So there exists a $t_2 \in T$ with $f(t_2) = t_2'$ and $(t_1, t_2) \in R_1^n$, hence also $(t_1, t_2) \in R_1^+$.

Condition (3) is checked analogously.

(6) *Reflexive transitive closure.* First observe that any function from T to T' must be a bidirectional p-morphism from $(T, =)$ to $(T', =)$. Given that the reflexive transitive closure of a relation is the union of the non-reflexive transitive closure of that relation and $=$, the desired result follows immediately from cases (2) and (5).

This concludes our proof of Lemma 3.23. $\square$

We should point out that p-morphisms are *not* generally preserved under taking complements or forming either intersections or differences of relations.[18]

The following *P-morphism Theorem* is, essentially, a standard result in modal logics. What is a little different in our case is that the results proved in Lemma 3.23 allow us to limit the checking of the p-morphism conditions to the three basic accessibility relations of a model (namely R, S and N) rather than checking them for all of the eight compound accessibility relations referred to in the definition of the semantics of our modal operators.[19]

Theorem 3.24 (P-morphisms) *Let $\mathcal{M} = (\mathcal{F}, V)$ and $\mathcal{M}' = (\mathcal{F}', V')$ be general models with $\mathcal{F} = (T, R, S, N)$ and $\mathcal{F}' = (T', R', S', N')$, and let f be a (bidirectional) p-morphism from $\mathcal{M}$ to $\mathcal{M}'$. Then for all worlds $t \in T$ and all formulas φ we have:*

$$\mathcal{M}, t \models \varphi \quad \textit{iff} \quad \mathcal{M}', f(t) \models \varphi$$

Proof. By induction on the structure of formulas φ. For the base case, assume φ is a propositional letter. The statement then is a direct consequence of the second condition for p-morphisms formulated in Definition 3.22. It is also trivially true for the formula $\top$.

[18] Here is a simple counterexample for the case of the difference operation. Let $T = \{t_1, t_2\}$ and $T' = \{t'\}$ and define the following relations over these two domains: $R_1 = \{(t_1, t_1), (t_1, t_2), (t_2, t_2)\}$, $R_2 = \{(t_1, t_1), (t_2, t_2)\}$, $R_1' = \{(t', t')\}$, and $R_2' = \{(t', t')\}$. Then the function f with $f(t_1) = t'$ and $f(t_2) = t'$ is a bidirectional p-morphism both from (T, R_1) to (T', R_1') and from (T, R_2) to (T', R_2'), because all three conditions of Definition 3.20 are fulfilled. However, f is *not* a p-morphism from $(T, R_1 \backslash R_2)$ to $(T', R_1' \backslash R_2')$, because the homomorphic condition fails for one pair of elements: although we have $(t_1, t_2) \in R_1 \backslash R_2$, the required $(f(t_1), f(t_2)) \in R_1' \backslash R_2'$ does not hold.

[19] See items (5)–(12) in Definition 3.18.

For the induction step, we have to consider all the core operators of our language. Those are the operators referred to in Definition 3.18. The definable operators from Definition 3.7 need not be considered here. We start with the propositional connectives:

(1) *Negation.* We have $\mathcal{M}, t \models \neg\varphi$ iff $\mathcal{M}, t \not\models \varphi$, which by the induction hypothesis is equivalent to $\mathcal{M}', f(t) \not\models \varphi$, which in turn is equivalent to $\mathcal{M}', f(t) \models \neg\varphi$.

(2) *Conjunction.* We have $\mathcal{M}, t \models \varphi \wedge \psi$ iff both $\mathcal{M}, t \models \varphi$ and $\mathcal{M}, t \models \psi$. By the induction hypothesis this is equivalent to $\mathcal{M}', f(t) \models \varphi$ and $\mathcal{M}', f(t) \models \psi$, i.e. $\mathcal{M}', f(t) \models \varphi \wedge \psi$.

Next we turn to the modal operators. The formulations of the truth conditions for the various modal operators in Definition 3.18 have a common structure: truth of a formula requires the existence of a successor world with respect to a (possibly complex) accessibility relation and the truth of the main subformula of the formula in question at that world. These accessibility relations are all constructed from R, S, and N using only the operations of inverse and transitive closure, both of which are covered by Lemma 3.23. That is, f is a p-morphism with respect to all these relations (and the corresponding relations of $\mathcal{M}'$).

These observations allow us to treat all modal operators in the same way. For the induction step for the case of a modal operator applied to a principal subformula φ, we shall prove the following general claim:

> *Claim:* If α and α' are binary relations such that f is a p-morphism from (T, α) to (T', α') then, for all $t \in T$ and all formulas φ (covered by the induction hypothesis), there exists a $t_\alpha \in T$ with $(t, t_\alpha) \in \alpha$ and $\mathcal{M}, t_\alpha \models \varphi$ *iff* there exists a $t'_\alpha \in T'$ with $(f(t), t'_\alpha) \in \alpha'$ and $\mathcal{M}', t'_\alpha \models \varphi$.

'$\Rightarrow$': Assume there exists a $t_\alpha \in T$ with $(t, t_\alpha) \in \alpha$ and $\mathcal{M}, t_\alpha \models \varphi$. From the induction hypothesis we get $\mathcal{M}', f(t_\alpha) \models \varphi$ and from the homomorphic condition we get $(f(t), f(t_\alpha)) \in \alpha'$. Hence, there indeed exists a $t'_\alpha \in T'$, namely $f(t_\alpha)$, satisfying the required conditions.

'$\Leftarrow$': Now assume there exists a $t'_\alpha \in T'$ with $(f(t), t'_\alpha) \in \alpha'$ and $\mathcal{M}', t'_\alpha \models \varphi$. By the back condition, there exists a $t_\alpha \in T$ with $f(t_\alpha) = t'_\alpha$ (that is, in particular $\mathcal{M}', f(t_\alpha) \models \varphi$) and $(t, t_\alpha) \in \alpha$. Now, using the induction hypothesis, we obtain $\mathcal{M}, t_\alpha \models \varphi$.

This concludes our proof for the case of diamond- and next-operators and thereby our proof of Theorem 3.24. $\Box$

Composing p-morphisms. Before we move on to discuss how we can apply the P-morphism Theorem for our investigation of ordered tree logics, we shall prove two very simple lemmas that show that we can compose p-morphisms as expected.

Lemma 3.25 (Composition of p-morphisms for relational structures) *If f is a bidirectional p-morphism from (T, R) onto (T', R') and g is a bidirectional p-morphism* *from* (T', R') *onto* (T'', R'') *then* $f \circ g$ *must be a bidirectional p-morphism from* (T, R) *onto* (T'', R'').

Proof. We need to check the three conditions from Definition 3.20:

(1) *Homomorphic condition.* Let $(t_1, t_2) \in R$. We can infer $(f(t_1), f(t_2)) \in R'$ and then $(g(f(t_1)), g(f(t_2))) \in R''$, using the homomorphic condition for f and g, respectively. Thus we obtain $(f \circ g(t_1), f \circ g(t_2)) \in R''$ as required.

(2) *Back condition.* Let $(f \circ g(t_1), t_2'') \in R''$, which is the same as $(g(f(t_1)), t_2'') \in R''$. By the back condition for g, there must be a $t_2' \in T'$ with $g(t_2') = t_2''$ and $(f(t_1), t_2') \in R'$. Now the back condition for f provides us with a $t_2 \in T$ with $f(t_2) = t_2'$ (i.e. also $f \circ g(t_2) = t_2''$) and $(t_1, t_2) \in R$, as required.

(3) *Inverse back condition.* This can be shown in analogy to the back condition.

Finally, the surjectivity of $f \circ g$ follows immediately from the surjectivity of the single functions f and g. $\square$

Lemma 3.26 (Composition of p-morphisms for models) *Let* $\mathcal{M}$, $\mathcal{M}'$ *and* $\mathcal{M}''$ *be general models. If* f *is a p-morphism from* $\mathcal{M}$ *onto* $\mathcal{M}'$ *and* g *is a p-morphism from* $\mathcal{M}'$ *onto* $\mathcal{M}''$ *then* $f \circ g$ *must be a p-morphism from* $\mathcal{M}$ *onto* $\mathcal{M}''$.

Proof. We need to check the conditions from Definition 3.22. Let $\mathcal{M} = (T, R, S, N, V)$, $\mathcal{M}' = (T', R', S', N', V')$ and $\mathcal{M}'' = (T'', R'', S'', N'', V'')$. The fact that $f \circ g$ is a bidirectional p-morphism from (T, R) onto (T'', R''), from (T, S) onto (T'', S''), and from (T, N) onto (T'', N'') follows from Lemma 3.25. We also have $t \in V(P)$ iff $f(t) \in V'(P)$ iff $g(f(t)) \in V''(P)$ for all $t \in T$ and all propositional letters P, so we are done. $\square$

3.6 Loosely Ordered Trees

Applications of p-morphisms. How are we going to apply the P-morphism Theorem? The general idea is to exploit it in situations where we want to give a constructive proof of the satisfiability of a given formula, but can only build a model that *almost* corresponds to an ordered tree. If we can show that a general model $\mathcal{M}'$ is the p-morphic image of some model $\mathcal{M}$ that is based on an ordered tree then satisfiability of a formula φ in $\mathcal{M}'$ entails, by the P-morphism Theorem, satisfiability of φ in $\mathcal{M}$, and thereby satisfiability of φ in **OTL**.

Of course, only some very specific models can be shown to be p-morphic images of intended models. Our aim for this section will be to identify a suitable class of frames that (1) can be shown to be p-morphic images of ordered trees and that (2) relax the definition of an ordered tree sufficiently to really simplify constructive satisfiability proofs.[20] We are first going to establish a number of necessary conditions that any such frame will have to satisfy (in order to qualify as the p-morphic image of an ordered tree).

[20] This class of frames will later be defined as the class of *loosely ordered trees*.

Some properties preserved under taking p-morphic images. For a model based on an ordered tree, the accessibility relations underlying the three next-operators are all *functional*: any given node has *at most* one successor with respect to any of these three relations. The following lemma shows that this property is preserved under taking p-morphic images.

An immediate consequence of this is that for a general frame $\mathcal{F} = (T, R, S, N)$ to be the p-morphic image of an intended frame (i.e. a general frame corresponding to an ordered tree) it is a necessary condition that R^{-1}, N and N^{-1} are all functional relations.

Lemma 3.27 (P-morphic images of functional relations) *Let f be a surjective p-morphism from (T, R) to (T', R') and let R be a functional relation. Then R' must be functional, too.*

Proof. R is functional, i.e. $(t, t_1) \in R$ and $(t, t_2) \in R$ implies $t_1 = t_2$ for all t, t_1, $t_2 \in T$. We need to prove the same property for R'. So let t', t'_1, $t'_2 \in T'$ with $(t', t'_1) \in R'$ and $(t', t'_2) \in R'$. Given that f is required to be surjective there must be a $t \in T$ (at least one) with $f(t) = t'$, so we have $(f(t), t'_1) \in R'$ and $(f(t), t'_2) \in R'$. Applying the back condition to the former we can establish that there exists a $t_1 \in T$ with $f(t_1) = t'_1$ and $(t, t_1) \in R$. By the same argument we get $f(t_2) = t'_2$ and $(t, t_2) \in R$ for some $t_2 \in T$. As R is functional, we can infer $t_1 = t_2$, hence also $f(t_1) = f(t_2)$. This in turn gives us $t'_1 = t'_2$, the required result. $\square$

Another simple property that is preserved under taking p-morphic images is that of one relation being a subrelation to another relation.

Lemma 3.28 (P-morphic images of subrelations) *Let f be a surjective p-morphism from (T, R_1) to (T', R'_1) and from (T, R_2) to (T', R'_2) and let $R_1 \subseteq R_2$. Then $R'_1 \subseteq R'_2$ must hold, too.*

Proof. Let $(t'_1, t'_2) \in R'_1$. We need to show $(t'_1, t'_2) \in R'_2$. Given that f is a surjective function there must be a $t_1 \in T$ with $f(t_1) = t'_1$, i.e. we have $(f(t_1), t'_2) \in R'_1$. Using the back condition we see that now there must also be a $t_2 \in T$ with $f(t_2) = t'_2$ and $(t_1, t_2) \in R_1$. The latter implies $(t_1, t_2) \in R_2$, because we have $R_1 \subseteq R_2$. The homomorphic condition then yields the desired result, namely $(f(t_1), f(t_2)) \in R'_2$ and thereby $(t'_1, t'_2) \in R'_2$. $\square$

This last result entails, for instance, that for any general frame $\mathcal{F} = (T, R, S, N)$ to be the p-morphic image of an ordered tree we require $N \subseteq S$ (because the right neighbour relation is a subrelation to the right sibling relation) and $S \subseteq R^{-1} \circ R$ (because the right sibling relation is a subrelation to the sibling relation).

The next lemma establishes that transitivity is preserved as well. Hence, as the order declared over siblings in an ordered tree is transitive, the relation S in a general frame $\mathcal{F} = (T, R, S, N)$ needs to be transitive as well for $\mathcal{F}$ to qualify as the p-morphic image of an intended frame.

Lemma 3.29 (P-morphic images of transitive relations) *Let f be a surjective p-morphism from (T, R) to (T', R') and let R be a transitive relation. Then R' must be transitive, too.*

Proof. Suppose we have $(t'_1, t'_2) \in R'$ and $(t'_2, t'_3) \in R'$. By the same kind of argument as before (using surjectivity and the back condition) we show that there exist t_1, t_2, $t_3 \in T$ with $f(t_1) = t'_1$, $f(t_2) = t'_2$, and $f(t_3) = t'_3$ as well as $(t_1, t_2) \in R$ and $(t_2, t_3) \in R$. Transitivity of R yields $(t_1, t_3) \in R$. Hence, by the homomorphic condition, $(f(t_1), f(t_3)) \in R'$, that is, $(t'_1, t'_3) \in R'$ as required. $\square$

Another property that can easily be seen to be preserved under taking p-morphic images is *connectedness*. In fact, we only require the homomorphic condition to show this. A textbook example for a property that is *not* preserved is *irreflexivity*. A counterexample would be the function $f : (\mathbb{Z}, <) \to (\{0\}, \leq)$, where $<$ and $\leq$ are the usual order relations over integers, with $f(n) = 0$ for all $n \in \mathbb{Z}$. All three conditions from Definition 3.20 are fulfilled, i.e. f is a (surjective) bidirectional p-morphism, but the irreflexivity of $<$ is not preserved in the p-morphic image. Hence, the p-morphic image of any strict linear order (irreflexive, transitive, and connected), such as the sibling order S of an ordered tree, is bound to be a linear order (transitive and connected, but not necessarily either reflexive or irreflexive).

While irreflexivity is not preserved in the general case, there are certain classes of irreflexive frames for which any p-morphic image will also be irreflexive. We are going to see an example for this very soon.

P-morphic images of trees. So far we have studied what happens to relatively simple properties of relations (such as transitivity or irreflexivity) when we apply a p-morphism. Our next aim will be to prove that the p-morphic image of a tree is always a tree itself. This result will be stated in Lemma 3.32. The next two lemmas will be required in the proof of the former. They show how two particular properties of trees are preserved under taking p-morphic images.

Trees are *rooted*, that is, within every tree there must be a unique node, the root, from which every other node can be reached via the transitive closure of the child relation R. As the following lemma shows, this property is preserved under the application of p-morphisms.

Lemma 3.30 (P-morphic images of rooted frames) *Let f be a surjective bidirectional p-morphism from (T, R) to (T', R') and let there be a unique element $r \in T$ such that $\{t \in T \mid (r, t) \in R^*\} = T$. Then there exists a unique element $r' \in T'$ such that $\{t' \in T' \mid (r', t') \in R'^*\} = T'$.*

Proof. First recall that, by Lemma 3.23, f is a p-morphism from (T, R^*) to (T', R'^*). Define $r' = f(r)$. We need to show that $(f(r), t') \in R'^*$ for all $t' \in T'$. So assume there exists a $t' \in T'$ with $(f(r), t') \notin R'^*$. T' is the p-morphic image of T, so there must be a $t \in T$ with $f(t) = t'$. But (T, R) is rooted with root r, that is, we have $(r, t) \in R^*$ and hence, by the homomorphic condition, $(f(r), f(t)) \in R'^*$, which contradicts our assumption.

It remains to be shown that the root element in T' will be unique. So for the sake of contradiction, let us assume that besides r' there also exists an alternative root element $a' \in T'$ with $a' \neq r'$ and $\{t' \in T' \mid (a', t') \in R'^*\} = T'$. Given that r' is already known to be a root of T', the latter condition will hold iff $(a', r') \in R'^*$. (To see this, observe that whenever we can reach r' we can also reach any other node in the set.) This is equivalent to $(a', f(r)) \in R'^*$. Using the inverse back condition we can now establish the existence of an element $a \in T$ with $f(a) = a'$ and $(a, r) \in R^*$. But then, by the same argument as before, a must be a root element in T. Because T is supposed to have a unique root, we must infer $a = r$. Thus we obtain $f(a) = f(r)$ and then $a' = r'$, which contradicts our earlier assumption of a' being a second root element. □

As pointed out before, irreflexivity, whilst not preserved in the general case, may still be a property of the p-morphic images of certain classes of irreflexive frames. One such class is the class of trees, as the next lemma will show. Recall that in a tree (T, R), R is not only required to be irreflexive, but in addition we know that (T, R) must be rooted and R is what we may call *conversely functional:* every element has at most one predecessor with respect to the relation R. These additional properties will allow us to prove that irreflexivity is preserved in the p-morphic image.

Lemma 3.31 (P-morphic images of trees: irreflexivity) *Let f be a surjective bidirectional p-morphism from (T, R) to (T', R') and let (T, R) be a tree. Then R' must be irreflexive.*

Proof. Let r be the root of (T, R). For the sake of contradiction, assume there is a reflexive point $t' \in T'$, i.e. $(t', t') \in R'$. Let t be any element in T with $f(t) = t'$. Given that t is an element in a tree it must have a distance $n \in \mathbb{N}_0$ from the tree's root r, that is, we have $(r, t) \in R^n$. Then there can be no element $t^* \in T$ for which we have, for instance, $(t^*, t) \in R^{n+1}$, because there are no further elements 'above' the root.

By Lemma 3.23, f must also be a p-morphism both from (T, R^n) to (T, R'^n) and from (T, R^{n+1}) to (T, R'^{n+1}). Thus, by the homomorphic condition, we get $(f(r), f(t)) \in R'^n$. But t', which is the same as $f(t)$, is assumed to be a reflexive point, so we also get $(f(r), f(t)) \in R'^{n+1}$. If we rewrite $f(r)$ as r' we obtain $(r', f(t)) \in R'^{n+1}$. Hence, by the inverse back condition, there must be a $t^* \in T$ with $f(t^*) = r'$ and $(t^*, t) \in R^{n+1}$, which contradicts what has been said before. Hence, R' will indeed be an irreflexive relation. □

We are now ready to put these results together and show that p-morphic images of trees are trees themselves.

Lemma 3.32 (P-morphic images of trees) *Let f be a surjective bidirectional p-morphism from (T, R) to (T', R') and let (T, R) be a tree. Then (T', R') must be a tree, too.*

Proof. We need to show that (T', R') is a tree according to Definition 3.2, that is, we need to show that R' is an irreflexive and conversely functional relation and that (T', R') is rooted.

(1) *Irreflexivity.* By Lemma 3.31, the relation R' must be irreflexive.

(2) *Converse functionality.* By Lemma 3.23, f is also a p-morphism from (T, R^{-1}) onto (T', R'^{-1}). Hence, by Lemma 3.27, R'^{-1} must be functional, as required.

(3) *Rootedness.* Lemma 3.30 guarantees that (T', R') will be rooted and will have a unique root element.

This conclues our proof of Lemma 3.32. □

Necessary conditions for p-morphic images of ordered trees. We have now seen several examples for properties that are necessarily preserved under taking p-morphic images. Most of them are rather general concepts, such as transitivity or functionality. Only the last result on p-morphic images of trees is a little more specific. They allow us to formulate a number of *necessary conditions* any p-morphic image of any ordered tree will comply to. From our discussion so far, we can already draw up the following list of properties. Most of these have already been mentioned above; the following merely serves as a summary.

Suppose $\mathcal{F} = (T, R, S, N)$ is the p-morphic image of an ordered tree (i.e. an intended frame). Then $\mathcal{F}$ will have (at least) the following properties:[21]

(1) (T, R) must be a tree.

(2) S must be a transitive relation.

(3) S must be a subrelation to $R^{-1} \circ R$.

(4) S must be connected over quasi-equivalence classes with respect to $R^{-1} \circ R$.

(5) N and N^{-1} must be functional relations.

(6) N must be a subrelation to S.

As we shall see, this list is not exhaustive yet. We can be even more specific about the properties of p-morphic images of ordered trees.

[21] These results will be fomally proved as part of Proposition 3.34.

Sufficient conditions for p-morphic images of ordered trees. More importantly than the necessary conditions discussed so far, we want to formulate a set of *sufficient conditions*, that, if met by a general frame $\mathcal{F}$, guarantee that $\mathcal{F}$ is the p-morphic image of an ordered tree. The above list of necessary conditions already leaves only a small degree of freedom for such a definition. Essentially, our choices are restricted to some aspects of the relation N.

Frames that fulfil our sufficient conditions will be called *loosely ordered trees*. That, indeed, any loosely ordered tree is the p-morphic image of an ordered tree will be proved in Lemma 3.40 towards the end of this section. Our second aspiration, namely that this definition of a class of frames is *useful* will be put to the test in the next chapter.

Definition 3.33 (Loosely ordered trees) *A general frame $\mathcal{F} = (T, R, S, N)$ is called a loosely ordered tree iff it satisfies the following conditions:*

(1) (T, R) *is a tree.*

(2) S *is a transitive relation.*

(3) $S \subseteq R^{-1} \circ R$

(4) $R^{-1} \circ R \subseteq S^* \cup S^{-1}$

(5) N *and* N^{-1} *are functional relations.*

(6) $N \subseteq S$

(7) $S \setminus S \circ S \subseteq N$

(8) $N^{-1} \circ S \subseteq S^*$

(9) $S \circ N^{-1} \subseteq S^*$

Here, S^* denotes the reflexive closure of S, that is, the relation $S \cup \{(t,t) \mid t \in T\}$. Given that S is required to be transitive, there is no need to distinguish the reflexive transitive closure (for which we had originally reserved this notation) from the simple reflexive closure.

The bigger picture. From a technical point of view, the central result of this section will be formulated in Lemma 3.40 (on page 93), which essentially expresses that the set of conditions laid down in the definition above are *sufficient* for a frame to provably be the p-morphic image of an ordered tree. The following proposition complements this result by showing that all of these conditions are in fact also *necessary*. This means that our choice of definition of loosely ordered trees (our definition of general frames characterising p-morphic images of ordered trees) is ideal in the sense that no class of frames strictly including the class of loosely ordered trees could still characterise our logic **OTL**.[22]

[22]At least there is no such class of frames that we could identify using only our methods based on p-morphisms.

Proposition 3.34 (P-morphic images of ordered trees) *Let $\mathcal{F}$ and $\mathcal{F}'$ be general frames and let f be a surjective p-morphism from $\mathcal{F}$ to $\mathcal{F}'$. Then $\mathcal{F}'$ must be a loosely ordered tree whenever $\mathcal{F}$ is an ordered tree.*

Proof. Let $\mathcal{F} = (T, R, S, N)$ be an ordered tree, that is a general frame such that (T, R, S) is an ordered tree according to Definition 3.4 and N corresponds to the immediate successor function underlying S (i.e. $N = S \setminus S \circ S$). Furthermore, let $\mathcal{F}' = (T', R', S', N')$ be a general frame and let f be a surjective p-morphism from $\mathcal{F}$ to $\mathcal{F}'$. To show that $\mathcal{F}'$ is a loosely ordered tree as claimed, we need to check that it fulfils the conditions laid down in Definition 3.33. In fact, all but one of these follow almost immediately from the lemmas on properties preserved under taking p-morphic images already proved in this section.

Most of the conditions in Definition 3.33 express that a particular (compound) relation is required to be a subrelation to another (compound) relation. With the exception of the difference operation used in condition (7), all the relational algebra operations used to construct the relations involved are covered by Lemma 3.23. This means, for instance, that f must be a bidirectional p-morphism from $(T, S \circ N^{-1})$ onto $(T', S' \circ N'^{-1})$, because f is known to be a bidirectional p-morphism from (T, S) onto (T', S') and from (T, N) onto (T', N'). Therefore, Lemma 3.28 (on p-morphic images of subrelations) applies to all of these conditions. Hence, all of the following must be true: $S' \subseteq R'^{-1} \circ R'$, $R'^{-1} \circ R' \subseteq S'^{*} \cup S'^{-1}$ (which means that S' is connected over quasi-equivalence classes with respect to $R'^{-1} \circ R'$), $N' \subseteq S'$, $N'^{-1} \circ S' \subseteq S'^{*}$, and $S' \circ N'^{-1} \subseteq S'^{*}$.

Furthermore, (T', R') must be a tree (Lemma 3.32), S' must be a transitive relation (Lemma 3.29), and both N' and N'^{-1} must be functional (Lemma 3.27).

The only condition that remains to be verified is condition (7): $S' \setminus S' \circ S' \subseteq N'$. So let t_1' and t_2' be two elements of T' such that $(t_1', t_2') \in S' \setminus S' \circ S'$, that is, we have $(t_1', t_2') \in S'$ but $(t_1', t_2') \notin S' \circ S'$. We are going to show that this implies $(t_1', t_2') \in N'$. The function f is required to be surjective, which means there exists a world $t_1 \in T$ with $f(t_1) = t_1'$. That is, we have $(f(t_1), t_2') \in S'$. By the back condition, there exists a $t_2 \in T$ such that $f(t_2) = t_2'$ and $(t_1, t_2) \in S$. We also know that $(t_1, t_2) \notin S \circ S$, because otherwise the homomorphic condition would yield $(f(t_1), f(t_2)) \in S' \circ S'$, which would contradict our assumption $(t_1', t_2') \notin S' \circ S'$. Together we obtain $(t_1, t_2) \in S \setminus S \circ S$. But $\mathcal{F}$ is an ordered tree, that is, we have $N = S \setminus S \circ S$. Hence, $(t_1, t_2) \in N$ must also be the case. Now the homomorphic condition yields $(f(t_1), f(t_2)) \in N'$. Thus $(t_1', t_2') \in N'$ indeed holds as required. □

What does a loosely ordered tree look like? At first sight, some of the conditions of Definition 3.33 may seem somewhat technical, if not arbitrary (at least before having studied the proof of Proposition 3.34). So let us motivate the name *loosely ordered tree* a little. Our main structure is a tree: T is the set of nodes and R is the child relation. S is a transitive relation operating over sibling nodes. That no other nodes can be related via S is ensured by condition $S \subseteq R^{-1} \circ R$. In fact, any two nodes that are siblings (that are related via $R^{-1} \circ R$) are either related via S or its inverse (or they are identical). This is guaranteed by condition $R^{-1} \circ R \subseteq S^* \cup S^{-1}$. This means that S is a transitive and connected relation over any given quasi-equivalence class $[t]_{R^{-1} \circ R}$ (for $t \in T$, except the root of the tree). For an ordered tree, S would be required to be a strict linear order over the elements of the same equivalence classes. Recall that a strict linear order is a relation that is irreflexive, transitive, and connected. So the only property that is missing here is irreflexivity. The triple (T, R, S) is almost like an ordered tree, the only difference is that S is not required to be irreflexive.

Clusters and irreflexive points. We are now going to take a closer look at the structure imposed on a set of sibling nodes by the relation S. Given two distinct sibling nodes t_1 and t_2, we have either $(t_1, t_2) \in S$ or $(t_2, t_1) \in S$ (or both), because S is connected over siblings. Without loss of generality, let us assume that t_1 is the node that comes 'first', i.e. let $(t_1, t_2) \in S$. On top of that, $(t_2, t_1) \in S$ may or may not hold as well. Suppose it does. Then, because S is also transitive, any node t 'between' t_1 and t_2, that is, every node t with $(t_1, t) \in S$ and $(t, t_2) \in S$, will in fact be related to all the other nodes between t_1 and t_2 via both S and S^{-1} as well.

We are going to refer to such a (maximal) set of nodes all of which are related via both S and S^{-1} as a *cluster*. This includes the case of a single reflexive point, that is, a node related to itself via S. Because S is a transitive relation, any node that is part of a cluster must be a reflexive point. Nodes that do not belong to a cluster will be called *irreflexive points*.

The children of any given node form a strictly ordered collection of clusters and irreflexive points. Next we turn our attention to the internal structure of clusters imposed on them by the relation N.

Sequences and cycles. In an ordered tree, N is meant to represent the immediate successor relation underlying S. In a loosely ordered tree, the conditions imposed on N are a little weaker. However, N is still required to be a subrelation to S and both N and N^{-1} are required to be functional. The former means that N can only operate on sibling nodes (because S is subject to the same restriction). Functionality means that any node can have at most one successor and one predecessor with respect to N. Hence, N will give rise to a *sequence* of nodes (a set of nodes where the first one is connected to the second, the second to the third, and so forth). Such a sequence may or may not be *cyclic* (that is, the last node in a sequence could again be connected to the first one). A single point without either a successor or a predecessor may be regarded as a special case of a non-cyclic sequence. A single reflexive point is a special case of a cycle.

How do these structures imposed on the children of a node by the relation N relate to the structure of clusters imposed by S? Let us first consider the case of cycles. Two nodes t_1 and t_2 are part of the same cycle iff we have both $(t_1, t_2) \in N^+$ and $(t_2, t_1) \in N^+$. Hence, t_1 and t_2 must also be part of the same S-cluster, because $N \subseteq S$ and S is transitive. This means that N-cycles can only occur within S-clusters, no cycle can involve nodes from more than one cluster. Also, no node that is part of a cycle can be an S-irreflexive point.

Next we turn to the case of non-cyclic N-sequences. The condition $S \setminus S \circ S \subseteq N$ says that nodes that *do* have an immediate successor with respect to S will be connected via N. This is the case for successive S-irreflexive points, which means that any sequence of S-irreflexive points will also be a non-cyclic N-sequence.

Apart from that, non-cyclic N-sequences can also occur within S-clusters. As we shall see, the last two conditions of Definition 3.33 guarantee that any sequence that forms part of a cluster will occur entirely within that cluster. To see this, suppose we have two nodes t_1 and t_2 with $(t_1, t_2) \in N$ and we know that t_1 is part of some S-cluster. We will show that t_2 must belong to the same cluster. Any node that is part of a cluster must be a reflexive point, i.e. we have $(t_1, t_1) \in S$. Hence, we get $(t_2, t_1) \in N^{-1} \circ S$. By condition $N^{-1} \circ S \subseteq S^*$ we therefore have $(t_2, t_1) \in S^*$. But we also have $(t_1, t_2) \in S$, so t_2 must indeed be part of the same cluster as t_1. We can use the final condition $S \circ N^{-1} \subseteq S^*$ in a similar manner in order to guarantee that also in the case where t_2 is known to belong to a cluster, its predecessor t_1 will have to be part of that very same cluster.[23]

In summary, a loosely ordered tree is like an ordered tree where the order declared over siblings is only required to be a strict linear order over clusters of sibling nodes, rather than over the nodes themselves. Furthermore, the notion of a successor function over siblings has been replaced with the (a little less restrained) functional relation N. In the context of a loosely ordered tree, we are sometimes going to refer to S as the *pseudo sibling relation* (or *order*) and to N as the *pseudo neighbour relation*.

Ordered trees are loosely ordered trees. We should also point out that the notion of loosely ordered trees constitutes a generalisation of ordered trees. The conditions on ordered trees only involving the relations R and S are in fact directly stated in Definition 3.33.[24] For a general frame corresponding to an ordered tree we have $N = S \setminus S \circ S$. Both $S \setminus S \circ S$ and its inverse are functional relations whenever S is a strict linear order. Furthermore, replacing N by $S \setminus S \circ S$ in conditions (6)–(9) of Definition 3.33 results in simple truths (again, provided S is a strict linear order). This shows that every ordered tree is also a loosely ordered tree.

As mentioned before, our aim is to show that every loosely ordered tree is the p-

[23] Another, possibly more intuitive, way of motivating conditions (8) and (9) would be the following. The relation N is intended to model small 'steps' to the right, while S corresponds to bigger steps, which may be made up of one or more of the small steps. Under this perspective, the compound relation $N^{-1} \circ S$ expresses the concept of first making a small step to the left and then a big step to the right. If N and S model these steps the way we intend them to, then after these two steps we should come out in a position to the right of where we started (or at most at the starting position itself). This can be assured by postulating condition (8): $N^{-1} \circ S \subseteq S^*$. Condition (9) has a corresponding effect for the case where we first make a big step to the right and then a small one to the left.

[24] In particular, the condition $R^{-1} \circ R \subseteq S^* \cup S^{-1}$ is simply another way of saying that S is a connected relation over quasi-equivalence classes with respect to the relation $R^{-1} \circ R$.

morphic image of an ordered tree. The proof proceeds over several steps; we shall introduce two p-morphisms to eliminate the different types of possible 'defects' in a loosely ordered tree.

Defects. By *defects* we mean features of a loosely ordered tree that violate conditions of an ordered tree. Our informal discussion of loosely ordered trees suggests that there are two types of defects. The first type of defect concerns the pseudo neighbour relation N. In an ordered tree all N-sequences have to be non-cyclic. Loosely ordered trees, on the other hand, also allow for N-cycles. Therefore, we have to consider any such cycle a defect. The following definition gives a formal characterisation of a loosely ordered trees lacking this type of defect.

Definition 3.35 (Cycles) *A loosely ordered tree $\mathcal{F} = (T, R, S, N)$ is called a loosely ordered tree without cycles iff there is no $t \in T$ such that $(t, t) \in N^+$.*

The second type of defect we will have to look at concerns the lack of irreflexivity of the pseudo sibling relation S in loosely ordered trees. In our informal discussion of loosely ordered trees above we have seen that this allows for clusters of nodes. Clusters are groups of nodes all of which are related to each other via S (and thereby also via S^{-1}). This means, in particular, that every node in a cluster will be reflexive with respect to S. In fact, the latter is sufficient a condition to characterise clusters. The following definition makes this notion formal.

Definition 3.36 (Clusters) *A loosely ordered tree $\mathcal{F} = (T, R, S, N)$ is called a loosely ordered tree without clusters iff there is no $t \in T$ such that $(t, t) \in S$.*

The next proposition shows that, of the two kinds of defects we have identified above, clusters are the main 'culprits'. Whenever a loosely ordered tree can be shown not to have any clusters at all, that tree will in fact be an ordered tree (and will therefore not contain any cycles either).

Proposition 3.37 (Clusters) *A loosely ordered tree without clusters is an ordered tree.*

Proof. Let $\mathcal{F} = (T, R, S, N)$ be a loosely ordered tree without clusters. Recall that $\mathcal{F}$ corresponds to an ordered tree iff (T, R, S) is an ordered tree according to Definition 3.4 and N is the immediate successor function underlying S, that is, $N = S \setminus S \circ S$.

By Definition 3.33, (T, R) is a tree and S is a transitive relation that is connected over quasi-equivalence classes $[t]_{R^{-1} \circ R}$ for $t \in T$ (except the root). As $\mathcal{F}$ is also required to have no clusters, S must be irreflexive. Hence, $([t]_{R^{-1} \circ R}, S)$ is a strict linear order for every $t \in T$. Thus (T, R, S) is an ordered tree.

We still need to show $N = S \setminus S \circ S$. As we have $S \setminus S \circ S \subseteq N$ for any loosely ordered tree, this amounts to proving $N \subseteq S \setminus S \circ S$. For the sake of contradiction, assume there are t_1, $t_2 \in T$ with $(t_1, t_2) \in N$ but $(t_1, t_2) \notin S \setminus S \circ S$. Using condition

$N \subseteq S$, we can infer $(t_1, t_2) \in S \circ S$, i.e. there exists a $t \in T$ such that $(t_1, t) \in S$ and $(t, t_2) \in S$. Putting $(t_2, t_1) \in N^{-1}$ and $(t_1, t) \in S$ together, we obtain $(t_2, t) \in N^{-1} \circ S$. Condition $N^{-1} \circ S \subseteq S^*$ yields $(t_2, t) \in S^*$. Together with $(t, t_2) \in S$ we then have $(t_2, t_2) \in S^* \circ S$. But S is transitive, so we also get $(t_2, t_2) \in S$, which contradicts our assumption of $\mathcal{F}$ not having any clusters. Hence, $N = S \setminus S \circ S$ must hold and we are done. $\Box$

Despite this result (namely that eliminating all clusters is enough to obtain ordered trees) our actual construction will fall into two parts. In the first step we will show how to eliminate all cycles in a given loosely ordered tree. Only in the second step we will then eliminate clusters, taking advantage of the fact that our trees may be assumed to be cycle-free.

Bulldozing and unravelling. The technique central to our construction is known as *bulldozing* in the modal logic literature. It has first been used by Segerberg to prove a number of completeness results for basic temporal logics over different irreflexive flows of time [64]. The basic idea is to replace a part of a frame that (1) has some undesired property (typically lack of irreflexivity) and (2) locally encodes an infinite nesting of accessibility (like, for instance, a single reflexive point) by an infinite number of copies of the respective part of the frame in a way that makes this infinite aspect explicit and does away with the undesired property at the same time. Showing that such a frame transformation is an admissible operation involves showing that the original frame can be considered the p-morphic image of the new frame. In the context of loosely ordered trees, an example for such a part of a frame would be a cycle: (1) it constitutes an undesired property (because ordered trees do not allow for cycles) and (2) it finitely encodes an infinite number of N-successors (because we can go round the cycle an infinite number of times). In the proof of Lemma 3.38 below we are going to apply (a variant of) the bulldozing technique to the problem of eliminating cycles from a loosely ordered tree.

Another important technique that we are going to make use of is *unravelling*. Given a frame and some designated world in that frame we can construct a new frame by considering every finite path we can follow in the original frame, starting from that designated world, as a world of the new frame. The original frame can then be shown to be the p-morphic image of the unravelled frame. What makes the unravelling technique particularly interesting for our case is that the resulting frame must always be a tree (and the designated world used for the unravelling process will be the root of that tree).

In the proofs of the following two lemmas we are going to use a combination of bulldozing and unravelling to help eliminate cycles and clusters from a given loosely ordered tree. (Applications of these techniques for logics other than **OTL** may, for instance, be found in [8].)

Lemma 3.38 (Cycle elimination) *Every model based on a loosely ordered tree is the p-morphic image of a model based on a loosely ordered tree without cycles.*

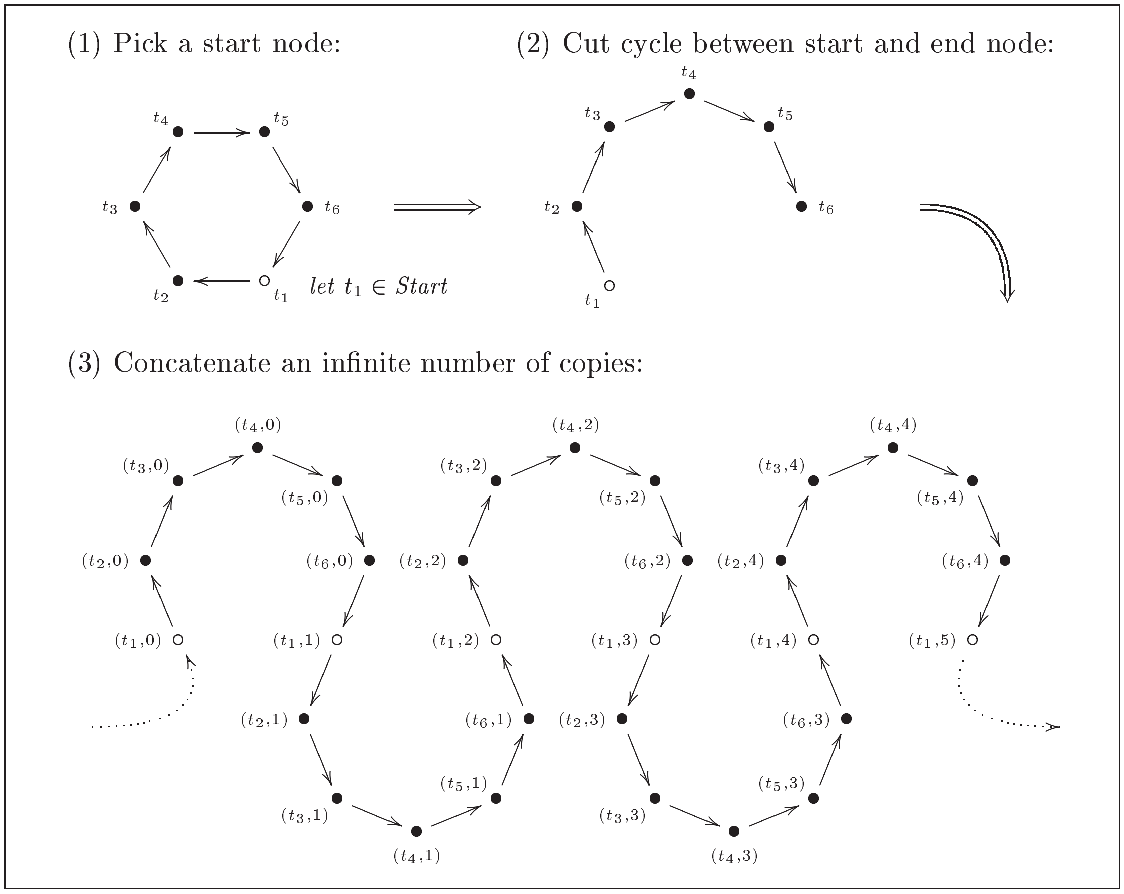

Figure 3.3: Unwinding (bulldozing) cycles

Proof. Let $\mathcal{F} = (T, R, S, N)$ be a loosely ordered tree that may contain a number of cycles and let $\mathcal{M} = (\mathcal{F}, V)$ be a model based on that tree. Our aim will be to construct a new frame and a corresponding model in such a way that we can show that there exists a p-morphism from that new model onto $\mathcal{M}$ and that the new frame is a loosely ordered tree without cycles.

For our construction, the general idea is to 'unwind' (or bulldoze) cycles, that is, we will 'cut open' each cycle to turn it into a non-cyclic sequence and then replace the original cycle with an infinite number of copies of that sequence. To be able to cut open a cycle we will pick an arbitrary start node in each cycle. That is, the result of cutting open a cycle will be a non-cyclic sequence commencing with that start node and ending with the node originally preceding the start node in the cycle. The infinite number of copies will be generated by considering pairs of nodes (from the original frame) and integers. Figure 3.3 exemplifies the process of unwinding for the simple case of a cycle with six nodes.

While, on the one hand, this approach allows us to eliminate any cycles in a frame, on the other hand, the resulting structure will not be a tree anymore. When we replaced a node by an infinite number of copies of itself, we should have applied the same process to the subtree below that node. However, the structure described before may be considered a compact representation of a tree. The second frame depicted in Figure 3.4 on page 80 is an example of such a compact structure. We can get a proper tree by unravelling this structure along the child relation R.

We are going to present our construction in two stages. In the first stage we describe the process of unwinding cycles. This involves the formal definition of our set of pairs of nodes and integers as well as the definition of a new auxiliary neighbour relation over these pairs. In the second stage we then unravel the structure over pairs constructed before to again obtain a tree.

Unwinding cycles. We are now going to formally define the set of pairs mentioned before. The first element in each pair will be a node from T and the second element will be an integer. For nodes that are part of a cycle the full set of integers will be used. Any other node $t \in T$ will only induce a single pair $(t, 0)$. Recall that a node t is part of a cycle iff $(t, t) \in N^+$ holds. We define our set of pairs as follows:

$$Pairs \quad = \quad \{(t, n) \in T \times \mathbb{Z} \mid (t, t) \in N^+\} \ \cup \ \{(t, 0) \in T \times \mathbb{Z} \mid (t, t) \notin N^+\}$$

Next, in order to be able to describe the process of cutting open cycles, we have to pick a start node for each cycle. These will be represented in form of a set of nodes called *Start*. Two nodes t_1 and t_2 belong to the same cycle iff we have both $(t_1, t_2) \in N^+$ and $(t_2, t_1) \in N^+$. If we already know that the two nodes in question are nodes of a cycle then checking only one of these conditions suffices to establish that they belong to the same cycle. Hence, *Start* needs to be a set of nodes satisfying the following conditions:

(1) $Start \subseteq \{t \in T \mid (t, t) \in N^+\}$

(2) If $(t_1, t_1) \in N^+$ then there exists a $t_2 \in Start$ with $(t_1, t_2) \in N^+$.

(3) If both $t_1 \in Start$ and $t_2 \in Start$ with $t_1 \neq t_2$ then $(t_1, t_2) \notin N^+$.

There are no other constraints on the choice of the set *Start*.

The crucial step in our construction is the definition of the new pseudo neighbour relation N_{Pairs}. We first give the formal definition and then we are going to explain the details.

$$
\begin{array}{lll}
((t_1, n_1), (t_2, n_2)) \in N_{Pairs} & \textit{iff} & (t_1, t_2) \in N \text{ and one of the following holds:} \\
 & & - \ (t_2, t_1) \in N^+ \text{ and } n_1 = n_2 \text{ and } t_2 \notin Start \\
 & & - \ (t_2, t_1) \in N^+ \text{ and } n_2 = n_1 + 1 \text{ and } t_2 \in Start \\
 & & - \ (t_2, t_1) \notin N^+
\end{array}
$$

First of all, observe that $((t_1, n_1), (t_2, n_2)) \in N_{Pairs}$ can only hold if we also have $(t_1, t_2) \in N$ in the original frame. So nodes that are not neighbours in the original frame can never 'become' neighbours in the new structure.

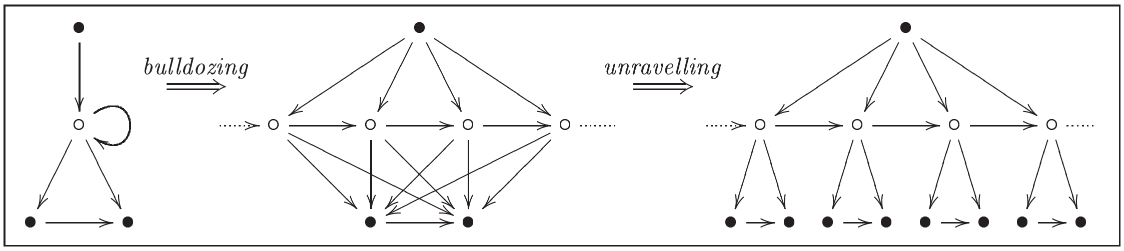

Figure 3.4: Unravelling a bulldozed tree

For the first and second case of our definition of N_{Pairs}, the condition $(t_2, t_1) \in N^+$ means that t_1 and t_2 must belong to the same cycle. The additional condition $n_1 = n_2$ (in the first case) says that they also belong to the same copy of the same cycle. Then the copies of the original nodes will be neighbours in the new structure if they are also neighbours in the cut cycle. This can be checked by ensuring that t_2 is not the start node of the cycle in question. So for the first case, our formal definition coincides with the informal outline of the construction given before.

For the second case of our definition, t_1 and t_2 are also members of the same cycle. If t_2 is the start node of our cycle then t_1, which is the lefthand neighbour of t_2 in the original frame, must be the end node of the cut version of that cycle. Hence, we want the two to become neighbours in the new frame only when they are part of consecutive copies, that is, whenever $n_2 = n_1 + 1$.

For the third and final case we have $(t_1, t_2) \in N$ but $(t_2, t_1) \not\in N^+$. This can only be the case if t_1 and t_2 are neighbours in a non-cyclic sequence. (Hence, both n_1 and n_2 will be 0 in this case.) They will continue to be neighbours in the new structure, i.e. we have given the right definition also for this case.

Unravelling. We can now turn to the second part of our construction, namely the unravelling of the compact structures obtained as a result of bulldozing all cycles in the given loosely ordered tree. The general idea is exemplified in Figure 3.4. Here we have a very simple loosely ordered tree with just one cycle consisting only of a single node (the only child of the root node). Unwinding this cycle results in the structure depicted in the middle part of the figure. We obtain infinitely many copies of the node originally contained in the cycle and each one of them is related to the same two 'grandchildren'. To turn this structure into a proper tree we have to make the appropriate number of copies of the grandchildren. One way of achieving this is to unravel the compact structure, that is, to create a grandchild node for every distinct path from the root to one of the grandchildren in the compact structure.[25] The resulting tree is depicted on the righthand side of Figure 3.4.

We are now going to formalise this idea. Let $t_0 \in T$ be the root of our original tree

[25] In fact, this is a rather simple variant of the unravelling technique, because we only apply it to our 'compact structures' (as opposed to general frames), which are already relatively close to actual trees.

(T, R). We define $\vec{T}$ as the set of all finite sequences[26] of pairs $[(t_0, n_0), \ldots, (t_m, n_m)]$ with $(t_i, n_i) \in \mathit{Pairs}$ for all $i \in \{0, \ldots, m\}$ and $(t_i, t_{i+1}) \in R$ for all $i \in \{0, \ldots, m-1\}$. The elements of $\vec{T}$ shall be referred to as *paths*. We write $\vec{t} + [p]$ for the path we obtain by appending the pair p to the end of path $\vec{t}$. At times, we will also require the notion of an *empty path*, which will be denoted as $[\,]$. The empty path itself is not a member of $\vec{T}$, but $[\,] + [(t_0, 0)]$, for instance, is.

We have to define three accessibility relations and a valuation function over paths in $\vec{T}$. The unravelling process will proceed along the original child relation R, or rather what is still left of it after the move from nodes in a tree to pairs. The new relation $\vec{R}$ will hold for pairs of paths, where the second path is the result of adding an additional R-step to the first path:

$$(\vec{t_1}, \vec{t_2}) \in \vec{R} \quad \textit{iff} \quad \vec{t_2} = \vec{t_1} + [p] \text{ for some } p \in \mathit{Pairs}$$

Next we define a new pseudo sibling relation $\vec{S}$:

$$\begin{array}{lll} (\vec{t_1}, \vec{t_2}) \in \vec{S} & \textit{iff} & \vec{t_1} = \vec{t} + [(t_1, n_1)] \text{ and } \vec{t_2} = \vec{t} + [(t_2, n_2)] \text{ and } (t_1, t_2) \in S \\ & & \text{for some } \vec{t} \in \vec{T} \text{ and some } (t_1, n_1),\ (t_2, n_2) \in \mathit{Pairs} \end{array}$$

Our definition of the new pseudo neighbour relation $\vec{N}$ refers to the relation N_{Pairs} over pairs defined before:

$$\begin{array}{lll} (\vec{t_1}, \vec{t_2}) \in \vec{N} & \textit{iff} & \vec{t_1} = \vec{t} + [p_1] \text{ and } \vec{t_2} = \vec{t} + [p_2] \text{ and } (p_1, p_2) \in N_{\mathit{Pairs}} \\ & & \text{for some } \vec{t} \in \vec{T} \text{ and some } p_1,\ p_2 \in \mathit{Pairs} \end{array}$$

Finally, we declare a new valuation function $\vec{V}$ mapping propositional letters P to sets of paths in $\vec{T}$. Given a propositional letter P, we want a path $\vec{t}$ to belong to $\vec{V}(P)$ iff $t \in V(P)$ for the last pair (t, n) in $\vec{t}$. (The remainder of $\vec{t}$, i.e. the path excluding the last pair (t, n), is either itself an element of $\vec{T}$ or the empty path $[\,]$.)

$$\begin{array}{lll} \vec{t} \in \vec{V}(P) & \textit{iff} & \vec{t} = \vec{t_1} + [(t, n)] \text{ and } t \in V(P) \\ & & \text{for some } \vec{t_1} \in \vec{T} \cup \{[\,]\} \text{ and some } (t, n) \in \mathit{Pairs} \end{array}$$

Let $\vec{\mathcal{F}} = (\vec{T}, \vec{R}, \vec{S}, \vec{N})$ be our new frame and let $\vec{\mathcal{M}} = (\vec{\mathcal{F}}, \vec{V})$ be our new model based on that frame. This completes our construction as outlined before. We now need to show that (1) $\vec{\mathcal{F}}$ is a loosely ordered tree without cycles and that (2) $\mathcal{M}$ is a p-morphic image of $\vec{\mathcal{M}}$.

Properties of the unravelled frame. To verify that $\vec{\mathcal{F}}$ is indeed a loosely ordered tree as claimed we need to check the nine conditions from Definition 3.33 in turn. Then we will have to check that there are no cycles left.

[26] Here we use the term *sequence* in the usual mathematical sense, i.e. this has nothing to do with our N-sequences.

(1) $(\vec{T}, \vec{R})$ *is a tree.* We have to check the three conditions of Definition 3.2. The relation $\vec{R}$ is certainly irreflexive, because we can never append an additional pair to a path and obtain the same path again. Furthermore, $\vec{R}$ must be conversely functional, because removing the last element from a path (which is the operation underlying $\vec{R}^{-1}$) always leaves us with exactly one path. Finally, $(\vec{T}, \vec{R})$ must be rooted with root element $[(t_0, 0)]$, because $\vec{T}$ has been defined precisely as the set of paths we can construct by successively appending new pairs to $[(t_0, 0)]$ and this operation coincides with the definition of $\vec{R}$.

(2) $\vec{S}$ *is transitive.* This is an immediate consequence of the definition of $\vec{S}$ and the fact that S is known to be a transitive relation.

(3) *Condition* $\vec{S} \subseteq \vec{R}^{-1} \circ \vec{R}$. Let $(\vec{t_1}, \vec{t_2}) \in \vec{S}$. By definition of $\vec{S}$, there must be a $\vec{t} \in \vec{T}$ and (t_1, n_1), $(t_2, n_2) \in \mathit{Pairs}$ such that $\vec{t_1} = \vec{t} + [(t_1, n_1)]$ and $\vec{t_2} = \vec{t} + [(t_2, n_2)]$. This implies $(\vec{t_1}, \vec{t}) \in \vec{R}^{-1}$ and $(\vec{t}, \vec{t_2}) \in \vec{R}$, i.e. we get $(\vec{t_1}, \vec{t_2}) \in \vec{R}^{-1} \circ \vec{R}$.

(4) *Condition* $\vec{R}^{-1} \circ \vec{R} \subseteq \vec{S}^* \cup \vec{S}^{-1}$. Two paths can only stand in the relation $\vec{R}^{-1} \circ \vec{R}$ if they are either identical or identical up to their penultimate elements, because moving from the first to the second means first removing the final element from the first and then appending a (possibly different) element to that shorter path. What elements could possibly be removed or added in that process depends on R. That is, we have $(\vec{t_1}, \vec{t_2}) \in \vec{R}^{-1} \circ \vec{R}$ iff there exist some $\vec{t} \in \vec{T}$ and some (t_1, n_1), $(t_2, n_2) \in \mathit{Pairs}$ such that $\vec{t_1} = \vec{t} + [(t_1, n_1)]$, $\vec{t_2} = \vec{t} + [(t_2, n_2)]$, and $(t_1, t_2) \in R^{-1} \circ R$. Given that the original frame is known to be a loosely ordered tree, the latter condition implies $(t_1, t_2) \in S^* \cup S^{-1}$. Inspection of the definition of $\vec{S}$ shows that this implies $(\vec{t_1}, \vec{t_2}) \in \vec{S}^* \cup \vec{S}^{-1}$ as required.

(5) $\vec{N}$ *and* $\vec{N}^{-1}$ *are functional.* We first show that N_{Pairs} is a functional relation. Suppose we have $((t, n), (t_1, n_1)) \in N_{\mathit{Pairs}}$ and $((t, n), (t_2, n_2)) \in N_{\mathit{Pairs}}$. (We need to show $(t_1, n_1) = (t_2, n_2)$.) Given that N is known to be functional, we immediately get $t_1 = t_2$. We now have to consider the three subconditions in the definition of N_{Pairs}. Which one of them could possibly be fulfilled is entirely determined by t and t_1 (or the equivalent t_2, respectively). If we have $(t_1, t) \in N^+$ then we either have $t_1 \in \mathit{Start}$ or $t_1 \notin \mathit{Start}$. In both cases the value of the second coordinate is fixed and only depends on n, that is, we also get $n_1 = n_2$. If we have $(t_1, t) \notin N^+$, then we immediately obtain $n = n_1 = n_2 = 0$. The fact that N_{Pairs}^{-1} must be functional as well can be established by an analogous argument.

Functionality of both $\vec{N}$ and $\vec{N}^{-1}$ now follows immediately by inspection of the definition of $\vec{N}$.

(6) *Condition* $\vec{N} \subseteq \vec{S}$. By definition of N_{Pairs}, $((t_1, n_1), (t_2, n_2)) \in N_{\mathit{Pairs}}$ implies $(t_1, t_2) \in N$. The latter in turn implies $(t_1, t_2) \in S$, because the original frame is a loosely ordered tree. Now $\vec{N} \subseteq \vec{S}$ follows immediately from the definitions of $\vec{N}$ and $\vec{S}$.

(7) *Condition* $\vec{S} \setminus \vec{S} \circ \vec{S} \subseteq \vec{N}$. Let $\vec{t_1}$ and $\vec{t_2}$ be paths such that $(\vec{t_1}, \vec{t_2}) \in \vec{S} \setminus \vec{S} \circ \vec{S}$, i.e. there are $\vec{t} \in \vec{T}$ and (t_1, n_1), $(t_2, n_2) \in Pairs$ such that $\vec{t_1} = \vec{t} + [(t_1, n_1)]$ and $\vec{t_2} = \vec{t} + [(t_2, n_2)]$ as well as $(t_1, t_2) \in S$ but $(t_1, t_2) \notin S \circ S$. This implies $(t_1, t_2) \in N$, because the original frame is a loosely ordered tree. It also implies $(t_2, t_1) \notin N^+$, because otherwise we could infer $(t_1, t_2) \in S \circ S$ (using $N^+ \subseteq S$). Hence, $((t_1, n_1), (t_2, n_2)) \in N_{Pairs}$, which has been the missing condition for $(\vec{t_1}, \vec{t_2}) \in \vec{N}$ to hold. This proves $\vec{S} \setminus \vec{S} \circ \vec{S} \subseteq \vec{N}$.

(8) *Condition* $\vec{N}^{-1} \circ \vec{S} \subseteq \vec{S}^*$. Let $\vec{t_1}$ and $\vec{t_2}$ be paths such that $(\vec{t_1}, \vec{t_2}) \in \vec{N}^{-1} \circ \vec{S}$. Then there must be a $\vec{t} \in \vec{T}$ as well as (t_1, n_1), (t_2, n_2), $(t, n) \in Pairs$ such that $\vec{t_1} = \vec{t} + [(t_1, n_1)]$, $\vec{t_2} = \vec{t} + [(t_2, n_2)]$, and $(\vec{t} + [(t, n)]) \in \vec{T}$. The latter is meant to represent the element in $\vec{T}$ we reach when moving from $\vec{t_1}$ 'backwards' via $\vec{N}^{-1}$ before moving 'forward' to get to $\vec{t_2}$. That is, we have $((t, n), (t_1, n_1)) \in N_{Pairs}$ and $(t, t_2) \in S$. The former implies $(t, t_1) \in N$. Thus we get $(t_1, t_2) \in N^{-1} \circ S$, which in turn entails $(t_1, t_2) \in S^*$. Putting everything together, we finally obtain $(\vec{t_1}, \vec{t_2}) \in \vec{S}^*$, the desired result.

(9) *Condition* $\vec{S} \circ \vec{N}^{-1} \subseteq \vec{S}^*$. This condition can easily be verified using an argument similar to that employed for the previous case.

(10) *No cycles.* We have to show that there is no $\vec{t} \in \vec{T}$ such that $(\vec{t}, \vec{t}) \in \vec{N}^+$. If we can prove that $((t, n), (t, n)) \notin N^+_{Pairs}$ for all $(t, n) \in Pairs$, then the desired result will follow immediately by inspection of the definition of $\vec{N}$.

For the sake of contradiction, assume there exists a pair $(t, n) \in Pairs$ with $((t, n), (t, n)) \in N^+_{Pairs}$. That is, there exists an $m \in \mathbb{N}$ such that $((t, n), (t, n)) \in N^m_{Pairs}$. So there must be a set of pairs $\{(t_i, n_i) \in Pairs \mid 1 \leq i \leq m\}$ with $(t, n) = (t_1, n_1) = (t_m, n_m)$ and $((t_i, n_i), (t_{i+1}, n_{i+1})) \in N_{Pairs}$ for all $i < m$. The latter implies $(t_i, t_{i+1}) \in N$ as well as $(t_{i+1}, t_i) \in N^+$ for all $i < m$. Hence, the third case in the definition of N_{Pairs} never applies for any of the $((t_i, n_i), (t_{i+1}, n_{i+1}))$. For the first two cases, the integer coordinate in a pair can only stay constant or increase. Given that n_1 and n_m are both required to be equal to n, the integer coordinate can in fact never increase, that is, we have $n_i = n$ for all $i \leq m$. Therefore, $((t_i, n_i), (t_{i+1}, n_{i+1})) \in N_{Pairs}$ must hold for all $i < m$ by the first case in the definition of N_{Pairs}. But then none of the t_i can be a member of *Start*. This contradicts the second constraint on the definition of the set *Start*, because the t_i are exactly those nodes for which we have $(t, t_i) \in N^+$.

Hence, $\vec{\mathcal{F}}$ is indeed a loosely ordered tree without cycles.

P-morphic image of the unravelled model. Our next aim is to show that the original model $\mathcal{M}$ is a p-morphic image of the unravelled model $\vec{\mathcal{M}}$. Let f be the function from $\vec{T}$ to T that maps every path to the first coordinate (a node in the original tree) of its final element:

$$f \; : \; [(t_0, n_0), \ldots, (t_m, n_m)] \; \mapsto \; t_m$$

We need to prove that f is a p-morphism from $\vec{\mathcal{M}}$ onto $\mathcal{M}$ (according to Definition 3.22). This amounts to showing f is a bidirectional p-morphism form $(\vec{T}, \vec{R})$ to (T, R), from $(\vec{T}, \vec{S})$ to (T, S), and from $(\vec{T}, \vec{N})$ to (T, N), that $\vec{t} \in \vec{V}(P)$ iff $f(\vec{t}) \in V(P)$ for all paths $\vec{t}$ and all propositional letters P, and that f is surjective. We start by showing that f is a bidirectional p-morphism from $(\vec{T}, \vec{R})$ to (T, R).

(1) *Homomorphic condition.* We need to show that $(\vec{t_1}, \vec{t_2}) \in \vec{R}$ implies $(f(\vec{t_1}), f(\vec{t_2})) \in R$. So let $(\vec{t_1}, \vec{t_2}) \in \vec{R}$ with, say, $\vec{t_1} = \vec{t} + [(t_1, n_1)]$, that is, the final pair of the path $\vec{t_1}$ is (t_1, n_1).[27] Then $\vec{t_2}$ must be of the form $\vec{t} + [(t_1, n_1), (t_2, n_2)]$. As $\vec{t_2} \in \vec{T}$ we must have $(t_1, t_2) \in R$. But $f(\vec{t_1}) = t_1$ and $f(\vec{t_2}) = t_2$, i.e. we have $(f(\vec{t_1}), f(\vec{t_2})) \in R$.

(2) *Back condition.* We need to show that whenever $(f(\vec{t_1}), t_2) \in R$ holds then there exists a $\vec{t_2} \in \vec{T}$ such that $f(\vec{t_2}) = t_2$ and $(\vec{t_1}, \vec{t_2}) \in \vec{R}$. So let $(f(\vec{t_1}), t_2) \in R$ and let $\vec{t_1}$ be of the form $\vec{t} + [(t_1, n_1)]$. We certainly have $(t_2, 0) \in \mathit{Pairs}$, because $t_2 \in T$. Thus $\vec{t} + [(t_1, n_1), (t_2, 0)]$ must be a path in $\vec{T}$ and we can define $\vec{t_2} = \vec{t} + [(t_1, n_1), (t_2, 0)]$. For this choice of $\vec{t_2}$ we have $f(\vec{t_2}) = t_2$ and $(\vec{t_1}, \vec{t_2}) \in \vec{R}$ as required.

(3) *Inverse back condition.* We need to show that whenever $(t_1, f(\vec{t_2})) \in R$ holds then there exists a $\vec{t_1} \in \vec{T}$ such that $f(\vec{t_1}) = t_1$ and $(\vec{t_1}, \vec{t_2}) \in \vec{R}$. Let $(t_1, f(\vec{t_2})) \in R$ and let (t_2, n_2) be the last pair in $\vec{t_2}$, i.e. $f(\vec{t_2}) = t_2$. Then we have $(t_1, t_2) \in R$, which means that $\vec{t_2}$ must be of the form $\vec{t} + [(t_1, n_1), (t_2, n_2)]$. Now, if we define $\vec{t_1} = \vec{t} + [(t_1, n_1)]$ we get $f(\vec{t_1}) = t_1$ and $(\vec{t_1}, \vec{t_2}) \in \vec{R}$ as required.

Next we prove that f is also a bidirectional p-morphism from $(\vec{T}, \vec{S})$ to (T, S).

(1) *Homomorphic condition.* $(\vec{t_1}, \vec{t_2}) \in \vec{S}$ implies $(f(\vec{t_1}), f(\vec{t_2})) \in S$ as required. This is an immediate consequence of the definition of $\vec{S}$.

(2) *Back condition.* We need to show that whenever $(f(\vec{t_1}), t_2) \in S$ holds then there exists a $\vec{t_2} \in \vec{T}$ such that $f(\vec{t_2}) = t_2$ and $(\vec{t_1}, \vec{t_2}) \in \vec{S}$. So let $(f(\vec{t_1}), t_2) \in S$ and suppose $\vec{t_1}$ is of the form $\vec{t} + [(t_1, n_1)]$, i.e. $f(\vec{t_1}) = t_1$ and $(t_1, t_2) \in S$. The latter implies $(t_1, t_2) \in R^{-1} \circ R$, because $\mathcal{F}$ is a loosely ordered tree. Hence, $\vec{t} + [(t_2, 0)]$ must be an element of $\vec{T}$. (Observe that $(t_2, 0) \in \mathit{Pairs}$, because $t_2 \in T$.) We can now define $\vec{t_2} = \vec{t} + [(t_2, 0)]$, which fulfils all requirements: $\vec{t_2} \in \vec{T}$, $f(\vec{t_2}) = t_2$, and $(\vec{t_1}, \vec{t_2}) \in \vec{S}$.

(3) *Inverse back condition.* We have to show that whenever $(t_1, f(\vec{t_2})) \in S$ holds then there exists a $\vec{t_1} \in \vec{T}$ such that $f(\vec{t_1}) = t_1$ and $(\vec{t_1}, \vec{t_2}) \in \vec{S}$. So let $(t_1, f(\vec{t_2})) \in S$ and suppose $\vec{t_2}$ is of the form $\vec{t} + [(t_2, n_2)]$. Our construction is analogous to the previous case: $\vec{t_1} = \vec{t} + [(t_1, 0)]$ must be an element of $\vec{T}$ and fulfils the required conditions.

[27] In situations like this, where we say that a path $\vec{t_1} \in \vec{T}$ is of the form $\vec{t} + [(t_1, n_1)]$, but do not claim that $\vec{t}$ is itself an element of $\vec{T}$, we include the possibility that $\vec{t}$ stands for the empty path $[\,]$, that is, $\vec{t_1}$ may just be $[(t_1, n_1)]$. (In that case t_1 would have to be the root node t_0 and n_1 would have to be 0, because $[(t_0, 0)]$ is the only path of length one in $\vec{T}$.)

The next step is to show that f is a bidirectional p-morphism from $(\vec{T}, \vec{N})$ to (T, N).

(1) *Homomorphic condition.* $(\vec{t_1}, \vec{t_2}) \in \vec{N}$ implies $(f(\vec{t_1}), f(\vec{t_2})) \in N_{Pairs}$ by definition of $\vec{N}$, which in turn implies $(f(\vec{t_1}), f(\vec{t_2})) \in N$ by definition of N_{Pairs}. So we are done.

(2) *Back condition.* We have to prove that $(f(\vec{t_1}), t_2) \in N$ implies the existence of a path $\vec{t_2} \in \vec{T}$ such that $f(\vec{t_2}) = t_2$ and $(\vec{t_1}, \vec{t_2}) \in \vec{N}$. So suppose we have $(f(\vec{t_1}), t_2) \in N$ and that $\vec{t_1}$ has the form $\vec{t} + [(t_1, n_1)]$, i.e. $f(\vec{t_1}) = t_1$ and $(t_1, t_2) \in N$.

First we show that there must be an n_2 such that $((t_1, n_1), (t_2, n_2)) \in N_{Pairs}$. We have either $(t_2, t_1) \in N^+$ or $(t_2, t_1) \notin N^+$. In the former case, t_1 and t_2 are part of the same cycle (that is, one of the first two cases in the definition of N_{Pairs} will apply). We define $n_2 = n_1 + 1$ if $t_2 \in \mathit{Start}$ and $n_2 = n_1$ otherwise. In either case we have $((t_1, n_1), (t_2, n_2)) \in N_{Pairs}$. Now suppose $(t_2, t_1) \notin N^+$, i.e. t_1 and t_2 are neighbours in a non-cyclic sequence. In this case n_1 must be 0 and if we set $n_2 = 0$ as well, we also get $((t_1, n_1), (t_2, n_2)) \in N_{Pairs}$. In summary, there always exists an integer n_2 such that $((t_1, n_1), (t_2, n_2)) \in N_{Pairs}$.

Now define $\vec{t_2} = \vec{t} + [(t_2, n_2)]$. This path fulfils our requirements: we have $f(\vec{t_2}) = t_2$ by definition of f and $(\vec{t_1}, \vec{t_2}) \in \vec{N}$ by definition of $\vec{N}$. That $\vec{t_2}$ really is an element of $\vec{T}$ can easily be shown by exploiting the fact $(t_1, t_2) \in N$ entails $(t_1, t_2) \in R^{-1} \circ R$ (because $\mathcal{F}$ is a loosely ordered tree).

(3) *Inverse back condition.* We need to show that whenever $(t_1, f(\vec{t_2})) \in N$ is the case then there exists a $\vec{t_1} \in \vec{T}$ such that $f(\vec{t_1}) = t_1$ and $(\vec{t_1}, \vec{t_2}) \in \vec{N}$ hold. So let $(t_1, f(\vec{t_2})) \in N$ where $\vec{t_2}$ is of the form $\vec{t} + [(t_2, n_2)]$. As with the back condition, the crucial step is to show that there exists an integer n_1 such that $((t_1, n_1), (t_2, n_2)) \in N_{Pairs}$. Our construction is essentially the same as before. We define $n_1 = n_2 - 1$ iff both $(t_2, t_1) \in N^+$ and $t_2 \in \mathit{Start}$ and $n_1 = n_2$ in all other cases. $((t_1, n_1), (t_2, n_2)) \in N_{Pairs}$ will hold in any case. Now we can define a new path $\vec{t_1} = \vec{t} + [(t_1, n_1)]$, which meets all of our requirements.

The final condition to be verified concerns the valuation functions of our models. We need to check that $\vec{t} \in \vec{V}(P)$ iff $f(\vec{t}) \in V(P)$ for all paths $\vec{t} \in \vec{T}$ and all propositional letters P. But this is easily seen to be the case by a simple inspection of the definitions of $\vec{V}$ and f.

So f is indeed a p-morphism from $\vec{\mathcal{M}}$ to $\mathcal{M}$. Furthermore, f must be a surjective function, because for every $t \in T$ there is at least one pair $(t, n) \in \mathit{Pairs}$, and for every element in Pairs there is at least one path in $\vec{T}$ leading to that pair. Hence, $\mathcal{M}$ is a p-morphic image of $\vec{\mathcal{M}}$. This completes our proof of Lemma 3.38, as we have shown that for any model $\mathcal{M}$ based on a loosely ordered tree we can construct a model $\vec{\mathcal{M}}$ based on a loosely ordered tree without cycles for which we can define a p-morphism f mapping onto $\mathcal{M}$. □

The next lemma shows how we can eliminate clusters in a loosley ordered tree that is known not to contain any cycles. Recall that we have seen in Proposition 3.37 that loosely ordered trees without clusters are in fact ordered trees.

Lemma 3.39 (Cluster elimination) *Every model based on a loosely ordered tree without cycles is the p-morphic image of a model based on an ordered tree.*

Proof. Let $\mathcal{F} = (T, R, S, N)$ be a loosely ordered tree without cycles and let $\mathcal{M} = (\mathcal{F}, V)$ be a model based on that tree. We are going to use $\mathcal{M}$ to construct a new frame and a new model based on that frame in such a way that we can show that the new frame is in fact an ordered tree and $\mathcal{M}$ is a p-morphic image of the new model.

In our construction clusters will be eliminated through bulldozing. In a loosely ordered tree without cycles, a cluster can contain any number of non-cyclic N-sequences as well as single points (that is, points not related to any other node via the relation N). As mentioned before, we may regard those single points as special case of N-sequences. So, in fact, a cluster is a collection of non-cyclic N-sequences. In the first step of our construction we will define a strict linear order S' over the children of any one child that is both a subrelation to S and a superrelation to the transitive closure of N. In other words, S' is an arbitrary ordering of the nodes in a cluster that 'respects' the order already imposed by N.

It can easily be seen that simply replacing S by S' does not suffice for the construction of a suitable new model. In the original model, all nodes in a cluster are related to each other via the relation S. That means, that in the constructed model we will have to generate enough copies of each node to ensure that still every kind of node comes up both before and after every kind of node within any one cluster. This will be achieved by generating pairs of nodes of the original model and integers. That is, every cluster will be replaced with the full set of integers and every integer will correspond to one copy of the 'straightened' version (using S') of the cluster.

To see that this is still not enough, consider that a node in a cluster may not have direct neighbours to both sides, namely if it is an endnode of an N-sequence. We have to ensure that this characteristic will also be maintained after our bulldozing operation. To this end, we are going to introduce triples of nodes of the original tree, integers, and rational numbers. In the model construction described so far, we then replace every N-sequence with infinitely many copies of that sequence, one for each rational number. As we do not break up the sequences themselves, any node that is supposed to have a neighbour will still have that neighbour, but in any other node the new order will be dense.

Figure 3.5 provides a simple example for the bulldozing part of the construction we have just outlined. The cluster under considertaion here consists of the four nodes t_1, t_2, t_3 and t_4. The first three form an N-sequence, while t_4 is a single node (or an N-sequence consisting of a single node). When choosing a strict linear order S' to straighten this cluster, we have two options. The node t_4 could either come before t_1

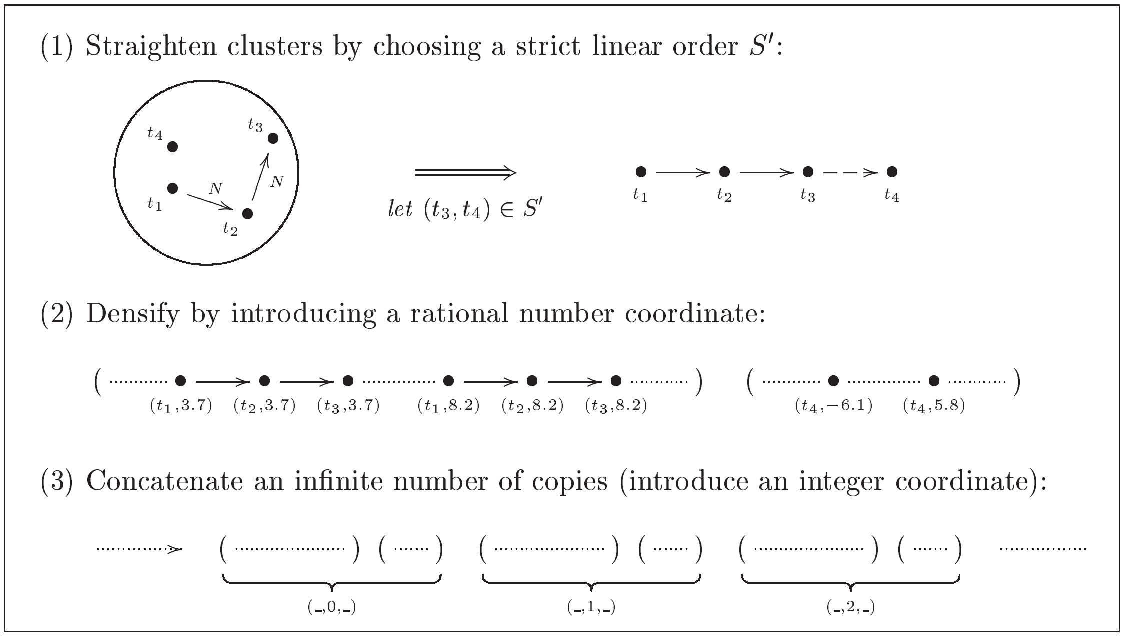

Figure 3.5: Bulldozing clusters

or after t_3. In Figure 3.5 we have opted for the latter. In the next step, we introduce the rational number coordinate to 'densify' the order constructed so far. Observe that S' 'dominates' the order over rational numbers whenever two nodes come from distinct N-sequences, otherwise the rational numbers dominate the nodes. For instance, $(t_3, 3.7)$ precedes $(t_1, 8.2)$, because t_1 and t_3 come from the same N-sequence and 3.7 is lower than 8.2. On the other hand, $(t_1, 8.2)$ precedes $(t_4, -6.1)$, even though -6.1 is the lower number of the two. This is so, because t_1 and t_4 do not belong to the same N-sequence and we have defined S' such that $(t_1, t_4) \in S'$ holds. In the third step we introduce the integer coordinate to generate an infinite number of copies of the densified straightened clusters. We should stress that this example only refers to a single cluster. The ordering of nodes from distinct clusters is not affected by our bulldozing operation. So whenever the node coordinates in two triples come from different clusters then they dominate the order over triples.

At the end of the construction, we will have to unravel again (as in the proof of Lemma 3.38) to turn the bulldozed structure described before into a tree. Figure 3.4 on page 80 may also serve as an example for the unravelling part of this construction.

Bulldozing clusters. We are now going to give a formal definition of the set of triples of nodes, integers, and rational numbers required for the bulldozing part of our construction. In the case of nodes that belong to a cluster the full sets of integers and rational numbers, respectively, will be used. In the case of irreflexive nodes t only a single triple $(t, 0, 0)$ will be introduced. Recall that, according to Definition 3.36, a node t is part of a cluster iff it is reflexive with respect to the pseudo sibling relation S, that is, iff $(t, t) \in S$ holds.

Hence, the following is an appropriate definition of the required set of triples:

$$\mathit{Triples} \quad = \quad \{(t,n,x) \in T\times\mathbb{Z}\times\mathbb{Q} \mid (t,t) \in S\} \cup \{(t,0,0) \in T\times\mathbb{Z}\times\mathbb{Q} \mid (t,t) \notin S\}$$

For the next step of our construction, we have to choose a relation S' with $N^+ \subseteq S' \subseteq S$ such that $([t]_{R^{-1}\circ R}, S')$ is a strict linear order for each $t \in T$. This is always possible, because (1) we have $N^+ \subseteq S$ (because $N \subseteq S$ is required to hold in any loosely ordered tree and S is required to be transitive), (2) the relation S is connected and transitive over the same equivalence classes, and (3) the relation N^+ is known to be irreflexive (because we have explicitly excluded loosely ordered trees with N-cycles). Now we can give a definition of a new pseudo sibling order over triples. Note that this is only an auxiliary construction; the final sibling relation will be given later on as part of our definition of the unravelled frame. We first give the formal definition of the new relation S_{Triples} and then explain the different parts of the definition in detail.

$$
\begin{array}{lll}
((t_1,n_1,x_1),(t_2,n_2,x_2)) \in S_{\mathit{Triples}} & \textit{iff} & (t_1,t_2) \in S \text{ and } (t_2,t_1) \notin S \text{ or} \\
& & (t_1,t_2) \in S \text{ and } (t_2,t_1) \in S \text{ and } n_1 < n_2 \text{ or} \\
& & (t_1,t_2) \in S \text{ and } (t_2,t_1) \in S \text{ and } n_1 = n_2 \text{ and} \\
& & -\ (t_1,t_2) \in S' \setminus N^* \text{ or} \\
& & -\ (t_1,t_2) \in (N \cup N^{-1})^* \text{ and } x_1 < x_2 \text{ or} \\
& & -\ (t_1,t_2) \in N^+ \text{ and } x_1 = x_2
\end{array}
$$

The righthand part of this definition is a disjunction of three conjunctions, where the fourth conjunct inside the third disjunct is again a disjunction. The first conjunct in each of the three main disjuncts is $(t_1,t_2) \in S$, which means that S 'dominates' the order S_{Triples}. In other words, triples with nodes that have not been pseudo siblings in our loosely ordered tree cannot become siblings in the bulldozed structure.

In the first case of the above definition, that is, in case $(t_1,t_2) \in S$ but also $(t_2,t_1) \notin S$ apply, the nodes t_1 and t_2 must belong to distinct clusters or irreflexive points in the original tree. In this case the other two coordinates have no influence on the ordering of the two triples concerned.

If, on the other hand, t_1 and t_2 belong to the same cluster, that is, if we have both $(t_1,t_2) \in S$ and $(t_2,t_1) \in S$, we can distinguish the following three cases. First of all, the two triples concerned, (t_1,n_1,x_1) and (t_2,n_2,x_2), may be part of different copies in the bulldozed structure. If this is the case, then the former precedes the latter iff $n_1 < n_2$, that is, iff the 'index' of the former copy is lower than that of the latter. If $n_1 > n_2$ then (t_1,n_1,x_1) should certainly not precede (t_2,n_2,x_2), so $n_1 = n_2$ must be true in all remaining cases (of which we are going to distinguish three).

It may, for instance, be the case that t_1 and t_2 (although part of the same cluster), are not part of the same N-sequence. If, however, t_1 still precedes t_2 with respect to our straightening order S', then (t_1,n,x_1) should precede (t_2,n,x_2) irrespectively of the values of x_1 and x_2. This is the case if $(t_1,t_2) \in S' \setminus N^*$ holds.

For the two remaining cases, we are concerned with triples that belong to the same copy (integer coordinate) and that have been generated from nodes belonging to the same

N-sequence in the original tree. Observe that t_1 and t_2 belong to the same N-sequence iff $(t_1, t_2) \in (N \cup N^{-1})^*$ is true. Then, if we also have $x_1 < x_2$, the first triple must precede the second. If we only have $x_1 = x_2$, then the two triples belong to the same copy also with respect to the rational number coordinate. In this case t_1 is required to precede t_2 in the original N-sequence, that is, $(t_1, t_2) \in N^+$ is required to hold. There are no other cases where we intend $((t_1, n_1, x_1), (t_2, n_2, x_2)) \in S_{\mathit{Triples}}$ to hold.

Let us now establish some properties of the relation S_{Triples}. First of all, S_{Triples} must be an *irreflexive* relation, because by the above definition either t_1 and t_2 or n_1 and n_2 or x_1 and x_2 are required to be distinct. This must be so, because for each case of the definition we refer to an irreflexive relation over one of the coordinates. (In particular, $S' \setminus N^*$ and N^+ are known to be irreflexive.)

Furthermore, S_{Triples} must be a *transitive* relation. This follows essentially from the transitivity of S and we only sketch a proof here. Let $((t_1, n_1, x_1), (t_2, n_2, x_2)) \in S_{\mathit{Triples}}$ and $((t_2, n_2, x_2), (t_3, n_3, x_3)) \in S_{\mathit{Triples}}$. Node t_2 is either part of a cluster or it is not. If it is not, then both $((t_1, n_1, x_1), (t_2, n_2, x_2)) \in S_{\mathit{Triples}}$ and $((t_2, n_2, x_2), (t_3, n_3, x_3)) \in S_{\mathit{Triples}}$ must hold by the first condition in the definition of S_{Triples} and $((t_1, n_1, x_1), (t_3, n_3, x_3)) \in S_{\mathit{Triples}}$ follows immediately by transitivity of S. Otherwise they must both hold by one of the other conditions and we get $(t_1, t_3) \in S$ as well as $(t_3, t_1) \in S$. A detailed inspection of the possible combinations of conditions shows that we also get $((t_1, n_1, x_1), (t_3, n_3, x_3)) \in S_{\mathit{Triples}}$.

We also note that whenever two nodes t_1 and t_2 are siblings then any two triples to which they give rise will be related either via S_{Triples} or its inverse. In other words, S_{Triples} is *connected* over triples the first coordinates of which are sibling nodes in the original loosely ordered tree.

Before we move on to the unravelling part of our construction, we also define an auxiliary neighbour relation over triples:

$$((t_1, n_1, x_1), (t_2, n_2, x_2)) \in N_{\mathit{Triples}} \quad \textit{iff} \quad (t_1, t_2) \in N \text{ and } n_1 = n_2 \text{ and } x_1 = x_2$$

That is to say, two triples are neighbours iff they belong to the same copy, with respect to both the integer and the rational number coordinate, and the two nodes concerned have been pseudo neighbours in the original loosely ordered tree.

Observe that $((t_1, n_1, x_1), (t_2, n_2, x_2)) \in N_{\mathit{Triples}}$ entails $((t_1, n_1, x_1), (t_2, n_2, x_2)) \in S_{\mathit{Triples}}$, namely by the last condition in the definition of S_{Triples}. Also, no other triple would 'fit' between (t_1, n_1, x_1) and (t_2, n_2, x_2) in this situation, i.e. we certainly have $N_{\mathit{Triples}} \subseteq S_{\mathit{Triples}} \setminus S_{\mathit{Triples}} \circ S_{\mathit{Triples}}$. Further inspection of the definition of S_{Triples} shows that the converse must hold as well, that is, we have $N_{\mathit{Triples}} = S_{\mathit{Triples}} \setminus S_{\mathit{Triples}} \circ S_{\mathit{Triples}}$.

Unravelling. The result of the bulldozing operation described before is not a tree, but rather a compact representation of a tree. In the second part of our construction, we need to turn this structure into a proper tree (an ordered tree, in fact, as we shall see in the sequel). This will be achieved through unravelling, in analogy to the process of unravelling described in the proof of Lemma 3.38. For the new unravelled model $\vec{\mathcal{M}}$, every finite path starting in the root of the bulldozed frame gives rise to a node in the new tree. The intuition underlying this construction is illustrated in Figure 3.4 on page 80.

Let t_0 be the root node in the original tree (T, R). We define the set of worlds $\vec{T}$ in the unravelled frame as the set of all finite sequences of triples $[(t_0, n_0, x_0), \ldots, (t_m, n_m, x_m)]$ with $(t_i, n_i, x_i) \in \mathit{Triples}$ for $i \in \{0, \ldots, m\}$ and $(t_i, t_{i+1}) \in R$ for $i \in \{0, \ldots, m-1\}$. Again, the elements of $\vec{T}$ will be referred to as *paths* and we write $\vec{t} + [u]$ to denote the path we obtain by appending the triple u to the end of path $\vec{t}$.

The child relation $\vec{R}$ of the unravelled frame is defined as follows:

$$(\vec{t}_1, \vec{t}_2) \in \vec{R} \quad \mathit{iff} \quad \vec{t}_2 = \vec{t}_1 + [u] \text{ for some } u \in \mathit{Triples}$$

The sibling relation $\vec{S}$ is supposed to hold between two paths iff the auxiliary sibling relation S_{Triples} holds over the respective last triples in those paths:

$$\begin{array}{lll} (\vec{t}_1, \vec{t}_2) \in \vec{S} & \mathit{iff} & \vec{t}_1 = \vec{t} + [u_1] \text{ and } \vec{t}_2 = \vec{t} + [u_2] \text{ and } (u_1, u_2) \in \ S_{\mathit{Triples}} \\ & & \text{for some } \vec{t} \in \vec{T} \text{ and some } u_1,\, u_2 \in \mathit{Triples} \end{array}$$

Similarly, $\vec{N}$ is defined in terms of N_{Triples}:

$$\begin{array}{lll} (\vec{t}_1, \vec{t}_2) \in \vec{N} & \mathit{iff} & \vec{t}_1 = \vec{t} + [u_1] \text{ and } \vec{t}_2 = \vec{t} + [u_2] \text{ and } (u_1, u_2) \in \ N_{\mathit{Triples}} \\ & & \text{for some } \vec{t} \in \vec{T} \text{ and some } u_1,\, u_2 \in \mathit{Triples} \end{array}$$

Finally, our new valuation function $\vec{V}$ is defined as a mapping from propositional letters to sets of paths in $\vec{T}$. (Observe that the following definition also covers the case of the 'root path' $[(t_0, 0, 0)]$.)

$$\begin{array}{lll} \vec{t} \in \vec{V}(P) & \mathit{iff} & \vec{t} = \vec{t}_1 + [(t, n, x)] \text{ and } t \in V(P) \\ & & \text{for some } \vec{t}_1 \in \vec{T} \cup \{[\,]\} \text{ and some } (t, n, x) \in \mathit{Triples} \end{array}$$

Let $\vec{\mathcal{M}} = (\vec{\mathcal{F}}, \vec{V})$ be the general model based on the general frame $\vec{\mathcal{F}} = (\vec{T}, \vec{R}, \vec{S}, \vec{N})$. This model $\vec{\mathcal{M}}$ is the unravelled bulldozed model the construction of which we have sketched at the beginning of this proof. We now need to show that $\vec{\mathcal{F}}$ is indeed an intended frame (i.e. a general frame that corresponds to an ordered tree) as claimed and then that we can define a p-morphism f mapping $\vec{\mathcal{M}}$ onto the original model $\mathcal{M}$.

Properties of the unravelled frame. We have to show that the unravelled frame $\vec{\mathcal{F}}$ is an ordered tree, or rather that $(\vec{T}, \vec{R}, \vec{S})$ is an ordered tree according to Definition 3.4 and that we have $\vec{N} = \vec{S} \setminus \vec{S} \circ \vec{S}$.

(1) $(\vec{T}, \vec{R})$ *is a tree.* Showing that $(\vec{T}, \vec{R})$ is a tree involves checking that $\vec{R}$ is irreflexive and conversely functional and that $(\vec{T}, \vec{R})$ has a root element (see Definition 3.2). Irreflexivity and converse functionality are immediate consequences of the definition of $\vec{R}$. It is also easy to see that the path $[(t_0, 0, 0)]$ must be the root of $(\vec{T}, \vec{R})$.

(2) *Condition* $\vec{S} \subseteq \vec{R}^{-1} \circ \vec{R}$. This follows directly from the definitions of $\vec{S}$ and $\vec{R}$.

(3) *Strict linear order over siblings.* Irreflexivity and transitivity of $\vec{S}$ immediately follow from the respective properties of the relation S_{Triples}. Furthermore, $\vec{S}$ must be connected over sibling nodes in the tree $(\vec{T}, \vec{R})$, because S_{Triples} is connected over triples that can be appended to the same path. Hence, $\vec{S}$ must be a strict linear order over every set of siblings in the unravelled tree.

(4) *Condition* $\vec{N} = \vec{S} \backslash \vec{S} \circ \vec{S}$. This follows immediately from the corresponding property of the auxiliary sibling and neighbour relations N_{Triples} and S_{Triples}, which we have established earlier: $N_{\mathit{Triples}} = S_{\mathit{Triples}} \setminus S_{\mathit{Triples}} \circ S_{\mathit{Triples}}$.

P-morphic image of the unravelled frame. It remains to be shown that the original model $\mathcal{M}$ is a p-morphic image of the unravelled bulldozed model $\vec{\mathcal{M}}$. To prove this we first define a suitable function f from the set of paths $\vec{T}$ to the original set of nodes T:

$$f \ : \ [(t_0, n_0, x_0), \ldots, (t_m, n_m, x_m)] \ \mapsto \ t_m$$

That is, f maps a path $\vec{t}$ to the first coordinate (a node in the original tree) of its final element (a triple). We claim that f is a surjective p-morphism from $\vec{\mathcal{M}}$ to $\mathcal{M}$. In order to validate this claim we need to show that f is a bidirectional p-morphism from $(\vec{T}, \vec{R})$ to (T, R), from $(\vec{T}, \vec{S})$ to (T, S), and from $(\vec{T}, \vec{N})$ to (T, N), that we have $\vec{t} \in \vec{V}(P)$ iff $f(\vec{t}) \in V(P)$ for all propositional letters P and all $\vec{t} \in \vec{T}$, and that f is indeed a surjective function onto T.

We start by showing that f is a bidirectional p-morphism from $(\vec{T}, \vec{R})$ to (T, R), which involves checking the three conditions from Definition 3.20.

(1) *Homomorphic condition.* Let $(\vec{t}_1, \vec{t}_2) \in \vec{R}$. We need to show that this implies $(f(\vec{t}_1), f(\vec{t}_2)) \in R$. Suppose $\vec{t}_1$ is of the form $\vec{t} + [(t_1, n_1, x_1)]$.[28] Then $\vec{t}_2$ must be of the form $\vec{t} + [(t_1, n_1, x_1), (t_2, n_2, x_2)]$. As $\vec{t}_2$ is an element of $\vec{T}$ we must have $(t_1, t_2) \in R$. This is the desired result, because we have $f(\vec{t}_1) = t_1$ and $f(\vec{t}_2) = t_2$ by definition of f.

(2) *Back condition.* Let $(f(\vec{t}_1), t_2) \in R$. We need to show that there exists a $\vec{t}_2 \in \vec{T}$ such that $f(\vec{t}_2) = t_2$ and $(\vec{t}_1, \vec{t}_2) \in \vec{R}$. Suppose $\vec{t}_1$ is of the form $\vec{t} + [(t_1, n_1, x_1)]$, that is, we have in particular $f(\vec{t}_1) = t_1$ and hence $(t_1, t_2) \in R$. Now define $\vec{t}_2 = \vec{t} + [(t_1, n_1, x_1), (t_2, 0, 0)]$. This must be an element of $\vec{T}$, because $\vec{t}_1$ is, because $(t_2, 0, 0) \in \mathit{Triples}$, and because $(t_1, t_2) \in R$. The path $\vec{t}_2$ fulfils our requirements: we have $f(\vec{t}_2) = t_2$ and $(\vec{t}_1, \vec{t}_2) \in \vec{R}$.

(3) *Inverse back condition.* Let $(t_1, f(\vec{t}_2)) \in R$. We need to prove the existence of a path $\vec{t}_1 \in \vec{T}$ with $f(\vec{t}_1) = t_1$ and $(\vec{t}_1, \vec{t}_2) \in \vec{R}$. Suppose the last triple in $\vec{t}_2$ is (t_2, n_2, x_2), i.e. $f(\vec{t}_2) = t_2$. That is, we have $(t_1, t_2) \in R$. Hence, the first coordinate

[28] That is, $\vec{t}$ is either an element of $\vec{T}$ or the empty path [].

of the penultimate triple in $\vec{t_2}$ must be t_1. In other words, $\vec{t_2}$ must be of the form $\vec{t} + [(t_1, n_2, x_2), (t_2, n_2, x_2)]$. Thus, for $\vec{t_1} = \vec{t} + [(t_1, n_1, x_1)]$ we get $f(\vec{t_1}) = t_1$ and $(\vec{t_1}, \vec{t_2}) \in \vec{R}$ as required.

Our next goal is to show that f is also a bidirectional p-morphism from $(\vec{T}, \vec{S})$ to (T, S).

(1) *Homomorphic condition.* $(\vec{t_1}, \vec{t_2}) \in \vec{S}$ implies $(f(\vec{t_1}), f(\vec{t_2})) \in S_{\mathit{Triples}}$, which in turn implies $(f(\vec{t_1}), f(\vec{t_2})) \in S$. Done.

(2) *Back condition.* Let $(f(\vec{t_1}), t_2) \in S$. We need to show that there is a $\vec{t_2} \in \vec{T}$ with $f(\vec{t_2}) = t_2$ and $(\vec{t_1}, \vec{t_2}) \in \vec{S}$. Suppose $\vec{t_1}$ is of the form $\vec{t} + [(t_1, n_1, x_1)]$, that is, $f(\vec{t_1}) = t_1$ and $(t_1, t_2) \in S$. As $\mathcal{F}$ is a loosely ordered tree we also have $(t_1, t_2) \in R^{-1} \circ R$. Now, given that $t_2 \in T$ and therefore $(t_2, 0, 0) \in \mathit{Triples}$, the path $\vec{t} + [(t_2, 0, 0)]$ must also be an element of $\vec{T}$ (because the first coordinate of the last riple in $\vec{t}$ must be related to t_2 via the relation R). Set $\vec{t_2} = \vec{t} + [(t_2, 0, 0)]$. This path has all the required properties and we are done.

(3) *Inverse back condition.* Let $(t_1, f(\vec{t_2})) \in S$ and let $\vec{t_2}$ be of the form $\vec{t} + [(t_2, n_2, x_2)]$. We have to prove the existence of a path $\vec{t_1} \in \vec{T}$ such that $f(\vec{t_1}) = t_1$ and $(\vec{t_1}, \vec{t_2}) \in \vec{S}$. This can be done in direct analogy to the previous case: the path $\vec{t_1} = \vec{t} + [(t_1, 0, 0)]$ fulfils our requirements.

Next we prove that f is a bidirectional p-morphism from $(\vec{T}, \vec{N})$ to (T, N).

(1) *Homomorphic condition.* $(\vec{t_1}, \vec{t_2}) \in \vec{N}$ implies $(f(\vec{t_1}), f(\vec{t_2})) \in N_{\mathit{Triples}}$, while the latter again implies $(f(\vec{t_1}), f(\vec{t_2})) \in N$. So we are done.

(2) *Back condition.* Let $(f(\vec{t_1}), t_2) \in N$. We need to show that there is a path $\vec{t_2} \in \vec{T}$ such that $f(\vec{t_2}) = t_2$ and $(\vec{t_1}, \vec{t_2}) \in \vec{N}$. Suppose $\vec{t_1}$ is of the form $\vec{t} + [(t_1, n, x)]$, i.e. $f(\vec{t_1}) = t_1$ and $(t_1, t_2) \in N$. The latter means that t_1 and t_2 must belong to the same N-sequence in the original loosely ordered tree. Hence, in case t_1 is part of a cluster (which would imply that n and x could have any values, not just 0) then t_2 would be part of the same cluster. In any case, (t_2, n, x) will be a member of *Triples*. Furthermore, we get $((t_1, n, x), (t_2, n, x)) \in N_{\mathit{Triples}}$. Now define $\vec{t_2} = \vec{t} + [(t_2, n, x)]$. This path must be an element of $\vec{T}$, because we get $(t_1, t_2) \in R^{-1} \circ R$ from $(t_1, t_2) \in N$ and the fact that the original frame $\mathcal{F}$ is a loosely ordered tree. The other two conditions are also fulfilled: $f(\vec{t_2}) = t_2$ and $(\vec{t_1}, \vec{t_2}) \in \vec{N}$.

(3) *Inverse back condition.* This condition can be verified in the same manner as the back condition before.

The next step is to verify that the two valuation functions $\vec{V}$ and V are related to each other as postulated in Definition 3.22. We have indeed $\vec{t} \in \vec{V}(P)$ iff $f(\vec{t}) \in V(P)$ for all $\vec{t} \in \vec{T}$ and all propositional letters P. This is an immediate consequence of the definitions of the new valuation function $\vec{V}$ and the function f.

Finally, it is easy to see that f must be a surjective function. For every node $t \in T$ we have $(t, 0, 0) \in \mathit{Triples}$ and, by definition of the unravelling process, for every triple in *Triples* there is at least one path leading to that triple. In summary, f must be a p-morphism from the unravelled model $\vec{\mathcal{M}}$ onto the original model $\mathcal{M}$ as claimed before. Before we have shown that the frame $\mathcal{F}$ the model $\mathcal{M}$ is based on must be an ordered tree. Hence, $\mathcal{M}$ is the p-morphic image of a model based on an ordered tree. Given that $\mathcal{M}$ could be any model based on any loosely ordered tree without cycles, every such model is the p-morphic image of a model based on an ordered tree. This completes our proof of Lemma 3.39. □

Putting it all together. The hard work is done and, finally, we can reap the rewards of our efforts. At the beginning of this section we set out to characterise a set of sufficient conditions for a general frame to provably be the p-morphic image of an ordered tree. It is now clear that the frame conditions defining a loosely ordered tree (Definition 3.33) meet this requirement. We record this result in the following lemma:

Lemma 3.40 (Loosely ordered trees) *Every model based on a loosely ordered tree is the p-morphic image of a model based on an ordered tree.*

Proof. Let $\mathcal{M}''$ be a general model that is based on a loosely ordered tree. By Lemma 3.38, there exists a model $\mathcal{M}'$ based on a loosely ordered tree without cycles such that $\mathcal{M}''$ is the p-morphic image of $\mathcal{M}'$. Now by Lemma 3.39, there exists a model $\mathcal{M}$ based on an ordered tree such that $\mathcal{M}'$ is the p-morphic image of $\mathcal{M}$. Let f be the p-morphism mapping $\mathcal{M}$ onto $\mathcal{M}'$ and let g be the p-morphism mapping $\mathcal{M}'$ onto $\mathcal{M}''$. Then by Lemma 3.26, the composed function $f \circ g$ must be a surjective p-morphism from $\mathcal{M}$ to $\mathcal{M}''$, that is, $\mathcal{M}''$ is indeed the p-morphic image of a general model based on an ordered tree as claimed. □

Recall that, shortly after we introduced the class of loosely ordered trees, we have shown that the opposite direction holds as well: any p-morphic image of a model based on an ordered tree will be a model based on a loosely ordered tree (Proposition 3.34).

The following theorem states the immediate consequence of Lemma 3.40 as far as the satisfiability of formulas in models based on either ordered trees or loosely ordered trees is concerned. In this theorem, p-morphisms are not mentioned anymore, which has the advantage that we will be able to make use of the results of this section later on without having to delve back into the technical details on how these results have been obtained in the first place.

Theorem 3.41 (Satisfiability in loosely ordered trees) *A formula has got a model based on an ordered tree iff it has got a model based on a loosely ordered tree.*

Proof. Every ordered tree is also a loosely ordered tree. Therefore, every formula that holds in a model based on an ordered tree also has a model based on a loosely ordered tree, namely that very same model. The other direction is an immediate consequence of Lemma 3.40 and Theorem 3.24. □

We are going to apply this theorem in the next chapter (Theorem 4.47 on page 139) where we prove completeness of an axiomatisation with respect to the class of loosely ordered trees and thereby, by Theorem 3.41, also with respect to ordered trees.

Modal logics of loosely ordered trees. Theorem 3.41 could be reformulated to express that a formula is *valid* in every ordered tree iff it is valid in every loosely ordered tree. If we think of a *logic* as the set of formulas that are valid over a particular class of frames, then this means that our modal logic of ordered trees coincides with what we might call the *modal logic of loosely ordered trees*.

Given that ordered trees are the simpler and more intuitive structure of the two, we still prefer to think of our logic as a logic of ordered trees. From a more technical point of view, however, the characterisation as a logic of loosely ordered trees is the 'sharper' one, as it omits a number of unnecessary frame conditions. Proposition 3.34 (on page 73) showed that, at least as long as we restrict our set of tools to p-morphisms, it is in fact the sharpest semantic characterisation of our logic there is. That is, there is no larger class of frames (that we could identify using only our approach based on p-morphisms) that still gives rise to the same logic (the same set of valid formulas).

3.7 Summary

We briefly recall some of the main ideas developed in this chapter.

Semantics of ordered tree logics. We have provided a formal *semantic characterisation* of the modal logic of ordered trees introduced informally in Chapter 1. In the first instance, we have defined the semantics of **OTL** by directly appealing to features of ordered trees such as children, ancestors, and left or right siblings. Each one of these has been associated with a modal operator of our language. We have also given definitions of important semantic concepts such as *satisfiability* and *validity*.

Later we have seen how we can describe the same logic as a special case of a more general multi-modal logic. Under this view, suitable conditions on the accessibility relations are required to ensure that models will still be based on frames that are ordered trees.

Model transformations. The introduction of this general notion of a model allowed us to study transformations between models that are satisfiability preserving. These transformations have been described by means of *p-morphisms*. We have also introduced the notion of a *loosely ordered tree* and showed that satisfiability with respect to models based on loosely ordered trees is an equivalent concept to that of satisfiability with respect to models based on ordered trees introduced before.

Expressiveness. We have demonstrated the expressive power and applicability of **OTL** in a number of ways. For instance, powerful concepts such as the *universal modality* or the *difference operator* are definable directly within our logic. We have also seen how we can incorporate notions from planning in artificial intelligence, such as the distinction of properties and events, into our framework.

Under the heading of *correspondences* we have shown how specific classes of ordered trees, such as *discretely ordered trees* or *finite trees* can be characterised syntactically, that is, by means of simple **OTL** formulas. This directly links in with the next chapter where we shall attempt not only to describe certain aspects of models by means of such axioms, but to provide an *axiomatic characterisation* of our logic itself.

Chapter 4

Axiomatic Characterisation

The aim of this chapter is to give an axiomatic presentation of the modal logic of ordered trees complementing the semantic view taken in the previous chapter, and to show that (or to what extent) semantic and axiomatic characterisation coincide.

We start by presenting the axioms and rules of our proof system. The main part of this chapter is devoted to a completeness proof for a fragment of our modal logic of ordered trees which excludes the two descendant modalities. The difficulty in axiomatising these two operators stems from the fact that their semantics is defined with respect to the transitive closure of the basic child relation R.

4.1 Axioms and Rules

Finding an axiomatisation of a semantically defined logic is one of the great big tasks in modal logic research. We can draw upon an abundance of results in this area and several of our axioms below are in fact variations of well-known formulas characterising certain frame properties that can be found all over the literature. Goldblatt [35] lists many of them. The work of Blackburn and Meyer-Viol [9] on the axiomatisation of a modal logic of finite binary trees is particularly relevant to ours. Naturally, there are a number of similarities between the two axiom systems, but they also differ in crucial points pertaining to, for instance, the order declared over sibling nodes.

We are now going to present our axiomatisation, starting with the *axioms* and later moving on to the *rules* of inference. A *derivation* (or a *proof*) is a finite sequence of formulas where every formula is either an axiom or the result of applying an inference rule to a finite number (either one or two, in fact) of formulas derived earlier. Our aim is to give an axiomatisation that corresponds to our logic **OTL**, which has been defined in terms of its model-theoretic semantics before. This means, we want only those formulas to be derivable in our axiomatic system that are valid with respect to the given semantics. In particular, this means that any axiom should in fact be a valid formula. Furthermore, inference rules should preserve validity, that is, it should not be possible to derive a non-valid formula (using a rule of the system) provided the premises are valid. Where appropriate, we are going to give intuitive justifications for the various axioms and rules along the way. A formal proof of the *soundness* of our system is sketched towards the end of this section.

Axiomatising propositional logic. Our modal logic extends classical propositional logic. Consequently, this axiomatisation will build on an axiomatisation of propositional logic. This could be any axiomatisation covering the propositional connectives we have used (like that given in [40], for instance). To emphasise that our interests lie in those aspects of **OTL** that go beyond propositional logic we are not going to adopt any specific axiomatisation of propositional logic here, but rather assume that our system contains all substitution instances of the classical tautologies. Also, in our axiomatic derivations later on, we are not going to explicitly justify proof steps that amount to purely propositional reasoning.

Duality axioms. We sometimes refer to a box-operator, such as $\Box_{\uparrow}$, as the *dual* of the corresponding diamond-operator, in this case $\Diamond_{\uparrow}$. Our first set of truly modal axioms is aimed at reflecting this relationship between box- and diamond-operators. The following five axioms mirror the syntactic definitions of the box-operators of our language given in Definition 3.7:

(D1) $\Box_{\leftarrow} A \leftrightarrow \neg \Diamond_{\leftarrow} \neg A$
(D2) $\Box_{\rightarrow} A \leftrightarrow \neg \Diamond_{\rightarrow} \neg A$
(D3) $\Box_{\uparrow} A \leftrightarrow \neg \Diamond_{\uparrow} \neg A$
(D4) $\Box_{\downarrow} A \leftrightarrow \neg \Diamond_{\downarrow} \neg A$
(D5) $\Box_{\downarrow}^{+} A \leftrightarrow \neg \Diamond_{\downarrow}^{+} \neg A$

Distribution axioms. The following axioms have the same structure as the (K) axiom familiar from the basic modal language. We call them *distribution axioms*, because they define box-operators as being distributive with respect to implication.

(K1) $\Box_{\leftarrow}(A \rightarrow B) \rightarrow (\Box_{\leftarrow} A \rightarrow \Box_{\leftarrow} B)$
(K2) $\Box_{\rightarrow}(A \rightarrow B) \rightarrow (\Box_{\rightarrow} A \rightarrow \Box_{\rightarrow} B)$
(K3) $\Box_{\uparrow}(A \rightarrow B) \rightarrow (\Box_{\uparrow} A \rightarrow \Box_{\uparrow} B)$
(K4) $\Box_{\downarrow}(A \rightarrow B) \rightarrow (\Box_{\downarrow} A \rightarrow \Box_{\downarrow} B)$
(K5) $\Box_{\downarrow}^{+}(A \rightarrow B) \rightarrow (\Box_{\downarrow}^{+} A \rightarrow \Box_{\downarrow}^{+} B)$

In fact, box-operators are not the only modalities that are distributive with respect to implication. Our next-operators exhibit the same behaviour. Suppose, for instance, $\bigcirc_{\uparrow}(A \rightarrow B)$ is true at a given node t, that is, $A \rightarrow B$ holds at the parent of t. If we also have $\bigcirc_{\uparrow} A$ being true at t, we get A at the parent node. As $A \rightarrow B$ and A imply B, we can conclude that B must also be true at the parent. That is, we get $\bigcirc_{\uparrow} B$ at t. Similar arguments apply to the case of the left and right neighbour modalities. Hence, all of the following are valid formulas:

(K6) $\bigcirc_{\leftarrow}(A \to B) \to (\bigcirc_{\leftarrow} A \to \bigcirc_{\leftarrow} B)$
(K7) $\bigcirc_{\rightarrow}(A \to B) \to (\bigcirc_{\rightarrow} A \to \bigcirc_{\rightarrow} B)$
(K8) $\bigcirc_{\uparrow}(A \to B) \to (\bigcirc_{\uparrow} A \to \bigcirc_{\uparrow} B)$

Axioms pertaining to inverse operators. The next set of axioms are intended to capture the relationship between pairs of modal operators the semantics of which have been defined with respect to two relations that are inverse to each other. An example would be the operators $\Diamond_{\rightarrow}$ and $\Diamond_{\leftarrow}$, which correspond to the order S declared over sibling nodes and its inverse S^{-1}, respectively. The first four of these inverse axioms look very similar to the standard axiom schema (B) characterising symmetric accessibility relations (hence the names).

(B1) $A \to \Box_{\leftarrow}\Diamond_{\rightarrow} A$
(B2) $A \to \Box_{\rightarrow}\Diamond_{\leftarrow} A$
(B3) $A \to \Box_{\uparrow}\Diamond_{\downarrow}^{+} A$
(B4) $A \to \Box_{\downarrow}^{+}\Diamond_{\uparrow} A$

Axiom (B1), for instance, expresses that whenever a formula A is true at a given node, then for every node to the left there exists a node to the right (namely the original node itself) where A is true. Axioms (B3) and (B4) have similar effects with respect to the ancestor and descendant modalities.

The next four axioms characterise the inverses of next-operators. Here the situation is a little more complex than in the case of the transitive operators above, because (unlike box-operators) our next-operators have existential (rather than universal) character. This is why in the antecedent of, say, axiom (B5) below we have to explicitly check that a lefthand neighbour actually exists. Axioms (B7) and (B8) relate the next-operator referring to the parent of a node and the (non-transitive) child modality. The former, for instance, expresses that whenever a formula A is true at a node and that node has got a parent then that parent will have a child (namely the original node) where A is true.

(B5) $(A \wedge \bigcirc_{\leftarrow}\top) \to \bigcirc_{\leftarrow}\bigcirc_{\rightarrow} A$
(B6) $(A \wedge \bigcirc_{\rightarrow}\top) \to \bigcirc_{\rightarrow}\bigcirc_{\leftarrow} A$
(B7) $(A \wedge \bigcirc_{\uparrow}\top) \to \bigcirc_{\uparrow}\Diamond_{\downarrow} A$
(B8) $A \to \Box_{\downarrow}\bigcirc_{\uparrow} A$

Functionality axioms. Next-operators have a functional interpretation. For example, if a node has got a parent node then it must have exactly one parent node, not more. Functionality of the three next-operators in our language can be characterised by means of the following axioms:

(F1) $\bigcirc_{\leftarrow} A \leftrightarrow (\neg \bigcirc_{\leftarrow}\neg A \wedge \bigcirc_{\leftarrow}\top)$
(F2) $\bigcirc_{\rightarrow} A \leftrightarrow (\neg \bigcirc_{\rightarrow}\neg A \wedge \bigcirc_{\rightarrow}\top)$
(F3) $\bigcirc_{\uparrow} A \leftrightarrow (\neg \bigcirc_{\uparrow}\neg A \wedge \bigcirc_{\uparrow}\top)$

Axiom (F1), for instance, expresses that a node has got a left neighbour at which A is true iff that node has got a left neighbour but it is not the case that it has got a left neighbour at which A is false.

Mixing next-operators and diamonds. The following set of axioms expresses certain relationships between modal operators referring to basic accessibility relations (such as the parent relation) and operators pertaining to the transitive closure (or a relation subsuming the transitive closure) of these basic relations. Consider, for instance, axiom (M3) below. The parent modality $\circleup$, which is a next-operator, refers to the parent relation R^{-1} and the corresponding diamond-operator $\diamondup$ refers to the transitive closure of R^{-1}, that is, it refers to the ancestor relation. Axiom (M3) shows how we can *'mix'* these two operators. It expresses that A is true at an ancestor of a given node t iff A holds either at the immediate parent of t or at an ancestor of the parent of t. In the case of the child and descendant modalities, which is covered by axiom (M4), the two operators concerned are in fact not a next-operator and a diamond, but a non-transitive and a transitive diamond-operator.

For the horizontal modalities the situation is a little more complicated, because the left (right) sibling relation is in fact not (just) the transitive closure of the left (right) neighbour relation, but is a relation including that transitive closure. This is reflected by the additional condition on the lefthand side of axioms (M1) and (M2), respectively.

(M1) $(\diamondleft A \wedge \circleleft \top) \leftrightarrow (\circleleft A \vee \circleleft \diamondleft A)$
(M2) $(\diamondright A \wedge \circleright \top) \leftrightarrow (\circleright A \vee \circleright \diamondright A)$
(M3) $\diamondup A \leftrightarrow (\circleup A \vee \circleup \diamondup A)$
(M4) $\diamonddown^{+} A \leftrightarrow (\diamonddown A \vee \diamonddown \diamonddown^{+} A)$

Induction axioms. At this point, readers familiar with axiomatisations of propositional dynamic logic (**PDL**) or linear temporal logic (over discrete flows of time) may miss the so-called *induction axioms* [35]. In the case of the child and descendant modalities, such an axiom would have the following form:

$$\boxdown^{+}(A \to \boxdown A) \to (\boxdown A \to \boxdown^{+} A)$$

Together with the mixing axioms from above, such axioms can help to axiomatise modal operators pertaining to the transitive closure of an accessibility relation. The reasons why we do in fact *not* include any axioms of this kind into our proof system are as follows. In the case of the horizontal modalities, they would simply not be valid, because (in the general case) the sibling relation is not the transitive closure of the neighbour relation. Axiomatising the relationship between the parent modality $\circleup$ and the universal ancestor modality $\boxup$ in this manner turns out to be redundant. (It would also be a little more complex than in our example for an induction axiom above, because $\circleup$ is not a universal modality.) Finally, as will become clear in the course of this chapter, including the induction axiom for the child and descendant modalities as stated above seems not to suffice to allow us to prove completeness for the full modal logic of ordered trees.

Axiomatising the (horizontal) flow of time. The next few axioms characterise different properties of the order declared over the children of a particular node in a tree. In our extended temporal logic interpretation of **OTL** this horizontal order corresponds to the temporal dimension of a system; so the following axioms may be regarded as describing properties of the flow of time we assume. The first such property is that time, or more precisely the sibling order, is *transitive*. The following is the standard axiom schema for transitivity, here with respect to the universal right sibling modality.

(H1) $\boxminus A \rightarrow \boxminus\boxminus A$

The following axiom (H2) expresses that whenever a proposition A is true at every righthand sibling we can reach in *two* steps, then A must hold at *all* righthand siblings, *unless* the current node has a direct righthand neighbour.

(H2) $\boxminus\boxminus A \rightarrow (\boxminus A \vee \ominus\top)$

This axiom bears some similarity to the formula usually given in the temporal logic literature to axiomatise a dense flow of time. For the modality $\boxminus$, that formula would be $\boxminus\boxminus A \rightarrow \boxminus A$ (see [35] and others). In our logic, the accessibility relation underlying $\boxminus$ may be dense in some points and discrete in others. In fact, what axiom (H2) says is that the order is dense whenever there is no direct successor.

Recall that in Section 3.3 of the previous chapter we have identified ordered trees where the order declared over children is dense as those trees that validate the formula $\neg \ominus\top$. Under the assumption that $\neg \ominus\top$ is valid, i.e. that $\ominus\top$ can never be true, the disjunction $\boxminus A \vee \ominus\top$ reduces to $\boxminus A$. In other words, in that case axiom (H2) can be simplified to the standard axiom for density $\boxminus\boxminus A \rightarrow \boxminus A$.

The next axiom (H3) expresses another relationship between the concepts of being siblings as opposed to being direct neighbours. It expresses that whenever A is true at all righthand siblings of a particular node then for every righthand sibling of a righthand sibling of that first node it cannot be the case that there is a lefthand neighbour at which A does not hold. This must be so, because any lefthand neighbour of a righthand sibling of a righthand sibling must itself be a righthand sibling.

(H3) $\boxminus A \rightarrow \boxminus\boxminus\neg \ominus\neg A$

So, in addition to the mixing axioms (M1)–(M4) from before, this last axiom also tells us something about we may *mix* next-operators with box- and diamond-operators. More concretely, it specifies how we can mix the lefthand next-operator with the righthand box-operator. Axiom (H3) also expresses an aspect of *linearity* of the horizontal flow of time. If we step to the right (to a righthand sibling) and then again to the left (a lefthand neighbour), we are still in the scope of the right sibling relation (or possibly back at the node where we started). In other words, time is not branching (to the left). We are going to refer to (H3) as the axiom of *mixed linearity*.

Interaction of horizontal and vertical operators. The following two axioms are probably those that are most characteristic for **OTL** and that clearly distinguish it from other multi-modal logics. They describe the *interaction* between horizontal and vertical modalities.

$$\text{(X1)} \quad \Diamond_{\rightarrow} A \rightarrow \bigcirc_{\uparrow} \Diamond_{\downarrow} A$$
$$\text{(X2)} \quad \bigcirc_{\uparrow} \Diamond_{\downarrow} A \rightarrow (A \vee \Diamond_{\leftarrow} A \vee \Diamond_{\rightarrow} A)$$

The first of these two axioms expresses that whenever a node has got a righthand sibling at which A is true, then its parent must have a child at which A is true. The second axiom expresses that whenever a node's parent has got a child at which A is true, then A must either be true at that node itself or at one of its left or right siblings. Axioms (X1) and (X2) are the only axioms in our system featuring both horizontal and vertical modalities.

Rootedness. Our final axiom characterises another distinctive property of a tree, namely that every node in a tree is accessible from the root node. It expresses that whenever A is true at a node that has ancestors (i.e. that is not the root) then that node has got an ancestor which does not have any ancestors itself (i.e. which *is* the root) and which has got a descendant (namely the original node) where A is true.

$$\text{(R1)} \quad (A \wedge \Diamond_{\uparrow} \top) \rightarrow \Diamond_{\uparrow}(\Diamond_{\downarrow}^{+} A \wedge \Box_{\uparrow} \bot)$$

This concludes the presentation of the axioms of our calculus.

Inference rules. The *rules* of our system are *modus ponens*, *uniform substitution*, and eight different incarnations of the *generalisation rule.* We start with modus ponens:

$$\frac{A, \; A \rightarrow B}{B}$$

This rule states that given two formulas A and $A \rightarrow B$ we can derive B. Here A and B may stand for any well-formed formulas in our language. The second inference rule is the rule of uniform substitution. It states that given a formula A we may derive any formula that we can obtain by uniformly substituting the propositional letters appearing in A by arbitrary well-formed formulas. For instance, $A \rightarrow (B \rightarrow A)$ is derivable in our calculus, because it is a classical tautology. If we replace every occurrence of A by, say, $\bigcirc_{\uparrow}\varphi$ and every occurrence of B by, say, $\neg \Box_{\rightarrow}\varphi$ we obtain the formula $\bigcirc_{\uparrow}\varphi \rightarrow (\neg \Box_{\rightarrow}\varphi \rightarrow \bigcirc_{\uparrow}\varphi)$, which is thus derivable by application of the uniform substitution rule. Both modus ponens and uniform substitution are familiar from axiomatic presentations of standard normal modal logics (and indeed propositional logic).

Another standard inference rule is the rule of generalisation (or *necessitation*). Our system includes a generalisation rule for each one of the five box-operators in the language. Such a rule states that given a formula A we may derive $\Box_{\leftarrow} A$ as well (and analogously for the other box-operators). The intuition behind this rule is that whenever A

is known to be valid, i.e. whenever A is true at every node (in every model), then surely A also has to be true at every node that is a left sibling of some node. Hence, $\boxleft A$ will be valid as well.

This kind of rule is valid for any universal operator, i.e. for any operator that does not postulate the existence of any other nodes. Box-operators are duals of (existential) diamond-operators. Some authors also define modalities that are dual to next-operators.[1] We have not introduced extra symbols for the duals of next-operators here, but shall instead refer to a 'combined' modality such as $\neg \circup \neg$ as the *dual* of $\circup$, and analogously for the other two next-operators. Our axiomatic system also includes generalisation rules with respect to the duals of each of the three next-operators of our modal language. Again, it is not difficult to see that validity of A entails validity of $\neg \circleft \neg A$ and similarly for the other two next-operators. Here are our eight generalisation rules:

$$\frac{A}{\boxleft A} \quad \frac{A}{\boxright A} \quad \frac{A}{\boxup A} \quad \frac{A}{\boxdown A} \quad \frac{A}{\boxdown^+ A} \quad \frac{A}{\neg \circleft \neg A} \quad \frac{A}{\neg \circright \neg A} \quad \frac{A}{\neg \circup \neg A}$$

This concludes our presentation of the axiomatic proof system for **OTL**. The axioms and rules of the system are summarised in Table 4.1. We should stress that we do not make any claims as to the independence of axioms. That is, we cannot exclude the possibility that some of the given axioms could be derived from others.

The fragment excluding the descendant modalities. For reasons that will be discussed in Section 4.3 we are mostly going to work with the fragment of our logic excluding the two descendant modalities $\diamonddown^+$ and $\boxdown^+$. In this context we think of our axiomatic proof system as the system excluding any of the axioms involving either $\diamonddown^+$ or $\boxdown^+$ as well as the generalisation rule with respect to the latter.

Theorems and consistent formulas. The *logic* induced by our axiomatic system is the smallest set of well-formed formulas that contains all propositional tautologies and the axioms given in Table 4.1 and that is closed under modus ponens, uniform substitution, and the eight generalisation rules that are also listed in Table 4.1. In other words, this logic is the set of formulas that can be *derived* in our axiomatic proof system. Derivable formulas are also called *theorems*.

Definition 4.1 (Derived theorems) *A formula φ is called a (derived) theorem iff φ belongs to the smallest set of well-formed formulas that contains all classical tautologies and the axioms of our calculus, and that is closed under application of modus ponens, uniform substitution, and the eight generalisation rules. We write $\vdash \varphi$.*

Intuitively, this syntactic concept of theoremhood corresponds to the semantic notion of validity. Similarly, the semantic notion of satisfiability has its syntactic counterpart in the notion of *consistency* defined next.

[1] Clarke and Schlingloff [18], for instance, distinguish a next-operator ('φ is true at the next state') and a weak next-operator ('it is not the case that φ is false at the next state').

Axioms

Distribution

(K1) $\Box_{l}(A \to B) \to (\Box_{l}A \to \Box_{l}B)$

(K2) $\Box_{r}(A \to B) \to (\Box_{r}A \to \Box_{r}B)$

(K3) $\Box_{u}(A \to B) \to (\Box_{u}A \to \Box_{u}B)$

(K4) $\Box_{d}(A \to B) \to (\Box_{d}A \to \Box_{d}B)$

(K5) $\Box_{d}^{+}(A \to B) \to (\Box_{d}^{+}A \to \Box_{d}^{+}B)$

(K6) $\bigcirc_{l}(A \to B) \to (\bigcirc_{l}A \to \bigcirc_{l}B)$

(K7) $\bigcirc_{r}(A \to B) \to (\bigcirc_{r}A \to \bigcirc_{r}B)$

(K8) $\bigcirc_{u}(A \to B) \to (\bigcirc_{u}A \to \bigcirc_{u}B)$

Inverse

(B1) $A \to \Box_{l}\Diamond_{r}A$

(B2) $A \to \Box_{r}\Diamond_{l}A$

(B3) $A \to \Box_{u}\Diamond_{d}^{+}A$

(B4) $A \to \Box_{d}^{+}\Diamond_{u}A$

(B5) $(A \wedge \bigcirc_{l}\top) \to \bigcirc_{l}\bigcirc_{r}A$

(B6) $(A \wedge \bigcirc_{r}\top) \to \bigcirc_{r}\bigcirc_{l}A$

(B7) $(A \wedge \bigcirc_{u}\top) \to \bigcirc_{u}\Diamond_{d}A$

(B8) $A \to \Box_{d}\bigcirc_{u}A$

Interaction

(X1) $\Diamond_{r}A \to \bigcirc_{u}\Diamond_{d}A$

(X2) $\bigcirc_{u}\Diamond_{d}A \to (A \vee \Diamond_{l}A \vee \Diamond_{r}A)$

Rootedness

(R1) $(A \wedge \Diamond_{u}\top) \to \Diamond_{u}(\Diamond_{d}^{+}A \wedge \Box_{u}\bot)$

Duality

(D1) $\Box_{l}A \leftrightarrow \neg\Diamond_{l}\neg A$

(D2) $\Box_{r}A \leftrightarrow \neg\Diamond_{r}\neg A$

(D3) $\Box_{u}A \leftrightarrow \neg\Diamond_{u}\neg A$

(D4) $\Box_{d}A \leftrightarrow \neg\Diamond_{d}\neg A$

(D5) $\Box_{d}^{+}A \leftrightarrow \neg\Diamond_{d}^{+}\neg A$

Functionality

(F1) $\bigcirc_{l}A \leftrightarrow (\neg\bigcirc_{l}\neg A \wedge \bigcirc_{l}\top)$

(F2) $\bigcirc_{r}A \leftrightarrow (\neg\bigcirc_{r}\neg A \wedge \bigcirc_{r}\top)$

(F3) $\bigcirc_{u}A \leftrightarrow (\neg\bigcirc_{u}\neg A \wedge \bigcirc_{u}\top)$

Mixing

(M1) $(\Diamond_{l}A \wedge \bigcirc_{l}\top) \leftrightarrow (\bigcirc_{l}A \vee \bigcirc_{l}\Diamond_{l}A)$

(M2) $(\Diamond_{r}A \wedge \bigcirc_{r}\top) \leftrightarrow (\bigcirc_{r}A \vee \bigcirc_{r}\Diamond_{r}A)$

(M3) $\Diamond_{u}A \leftrightarrow (\bigcirc_{u}A \vee \bigcirc_{u}\Diamond_{u}A)$

(M4) $\Diamond_{d}^{+}A \leftrightarrow (\Diamond_{d}A \vee \Diamond_{d}\Diamond_{d}^{+}A)$

Transitivity

(H1) $\Box_{r}A \to \Box_{r}\Box_{r}A$

Weak Density

(H2) $\Box_{r}\Box_{r}A \to (\Box_{r}A \vee \bigcirc_{r}\top)$

Mixed Linearity

(H3) $\Box_{r}A \to \Box_{r}\Box_{r}\neg\bigcirc_{l}\neg A$

Rules

Modus Ponens

$$\frac{A,\ A \to B}{B}$$

+ propositional tautologies

+ all substitution instances

Generalisation

$$\frac{A}{\Box_{l}A} \qquad \frac{A}{\Box_{u}A} \qquad \frac{A}{\Box_{d}^{+}A} \qquad \frac{A}{\neg\bigcirc_{r}\neg A}$$

$$\frac{A}{\Box_{r}A} \qquad \frac{A}{\Box_{d}A} \qquad \frac{A}{\neg\bigcirc_{l}\neg A} \qquad \frac{A}{\neg\bigcirc_{u}\neg A}$$

Table 4.1: Axioms and rules

Definition 4.2 (Consistency) *A set of formulas Δ is called inconsistent iff there are formulas $\varphi_1, \ldots, \varphi_n \in \Delta$ such that $\neg(\varphi_1 \wedge \cdots \wedge \varphi_n)$ is a theorem. Otherwise Δ is called a consistent set of formulas.*

The notion of consistency also applies to single formulas. We call a formula φ *consistent* iff the set $\{\varphi\}$ is consistent and we call it *inconsistent* if this is not the case.

An example. As an example for how to work with our axiomatic system, we give a derivation for the formula $\Diamond_{\downarrow}\Diamond_{\rightarrow}A \rightarrow \Diamond_{\downarrow}A$. This formula is interesting, because it illustrates very well the interaction of horizontal and vertical modalities. It essentially expresses that the righthand siblings of children of a given node are children of that node themselves. We first present the derivation itself. This is followed by a number of comments. (Note that the derived theorem $\Diamond_{\downarrow}\Diamond_{\rightarrow}A \rightarrow \Diamond_{\downarrow}A$ is stated in the final line.)

(1)	$\Diamond_{\rightarrow}A \rightarrow \bigcirc_{\uparrow}\Diamond_{\downarrow}A$	from (X1)
(2)	$\bigcirc_{\uparrow}\Diamond_{\downarrow}A \rightarrow \neg\bigcirc_{\uparrow}\neg\Diamond_{\downarrow}A$	from (F3)
(3)	$\Diamond_{\rightarrow}A \rightarrow \neg\bigcirc_{\uparrow}\neg\Diamond_{\downarrow}A$	from (1) and (2)
(4)	$\bigcirc_{\uparrow}\neg\Diamond_{\downarrow}A \rightarrow \neg\Diamond_{\rightarrow}A$	from (3)
(5)	$\Box_{\downarrow}(\bigcirc_{\uparrow}\neg\Diamond_{\downarrow}A \rightarrow \neg\Diamond_{\rightarrow}A)$	from (4) by generalisation
(6)	$\Box_{\downarrow}\bigcirc_{\uparrow}\neg\Diamond_{\downarrow}A \rightarrow \Box_{\downarrow}\neg\Diamond_{\rightarrow}A$	from (5) and (K4)
(7)	$\Diamond_{\downarrow}\Diamond_{\rightarrow}A \rightarrow \Diamond_{\downarrow}\neg\bigcirc_{\uparrow}\neg\Diamond_{\downarrow}A$	from (6) and (D4)
(8)	$\neg\Diamond_{\downarrow}A \rightarrow \Box_{\downarrow}\bigcirc_{\uparrow}\neg\Diamond_{\downarrow}A$	from (B8)
(9)	$\Diamond_{\downarrow}\neg\bigcirc_{\uparrow}\neg\Diamond_{\downarrow}A \rightarrow \Diamond_{\downarrow}A$	from (8) and (D4)
(10)	$\Diamond_{\downarrow}\Diamond_{\rightarrow}A \rightarrow \Diamond_{\downarrow}A$	from (7) and (9)

In line (1) of this proof we simply reproduce axiom (X1). Line (2) is the first example of a case where we represent more than just one atomic proof step within a single derivation line. It can be justified as follows. If we substitute the letter A in axiom (F3) with the formula $\Diamond_{\downarrow}A$ we obtain $\bigcirc_{\uparrow}\Diamond_{\downarrow}A \leftrightarrow (\neg\bigcirc_{\uparrow}\neg\Diamond_{\downarrow}A \wedge \bigcirc_{\uparrow}\top)$. The formula given in line (2) is a propositional consequence of the former. We generally do not spell out meta proof steps of this kind in detail, as it would only distract from the essence of what we intend to demonstrate. This does not pose a problem in view of our formal results, as it would always be possible to translate such a meta derivation into a detailed formal axiomatic proof. Line (3) combines the formulas in lines (1) and (2) by exploiting the transitive character of the implication operator. Again, this is not as such an inference rule of our system, but its validity follows from properties of the propositional calculus.

The next few steps, culminating in line (7), are aimed at putting a diamond-operator $\Diamond_{\downarrow}$ in front of the two principal subformulas of the implication in line (3). This is achieved by first applying the generalisation rule (with respect to $\Box_{\downarrow}$) to the contrapositive of line (3). The resulting formula is shown in line (5). This formula has the structure of the antecedent of the distribution axiom (K4). We could have written down that particular instantiation of (K4) and then applied modus ponens explicitly to obtain the formula in line (6). Again, these two reasoning steps have been merged into a single meta step.

In fact, we are hardly ever going to explicitly mention the application of modus ponens. Instead, abusing terminology to a certain degree, we will sometimes refer to a proof step like that from line (5) to (6) as an 'application' of axiom (K4). Line (7) is essentially the contrapositive of line (6), but we have also rewritten the two box-operators as diamonds using the duality axiom (D4).

We now have derived an implication the antecedent of which is already the antecedent of the implication we set out to prove. We still need to show that we can eliminate the 'prefix' $\Diamond\neg\bigcirc\neg$ from the consequent. This has been achieved in lines (8) and (9) by transforming a suitable instance of the inverse axiom (B8). Line (10), our desired conclusion, is again a combination of the two implicational formulas in lines (7) and (9). (More sample derivations may be found in Appendix B.)

Soundness and completeness. Broadly speaking, the objective of axiomatising a semantically defined logic (or analogously, of providing a semantics for a given axiomatic system) is to show that the notions of satisfiability and consistency coincide (or equivalently, that the notions of validity and theoremhood coincide).

A calculus such as our axiomatic proof system is called *sound* with respect to a semantically defined logic iff satisfiability (of a formula) entails consistency (or equivalently, iff theoremhood entails validity). The notion dual to soundness is (weak) completeness. We call a calculus *(weakly) complete* iff every consistent *formula* is satisfiable (or equivalently, iff every valid formula is a theorem). If it can also be shown that any consistent *set of formulas* is satisfiable, then we call that logic *strongly complete*.[2] Proving soundness is generally much easier than proving completeness. It essentially amounts to demonstrating that all axioms of the system are valid with respect to the chosen semantics. Formal proofs of soundness are often omitted in the literature, and we will also restrict ourselves to a brief proof sketch here.

Proposition 4.3 (Soundness) *Every satisfiable formula is consistent.*

Proof. We have to show that every theorem is a valid formula.[3] This involves firstly showing that every axiom of our system is a valid formula and secondly that all our inference rules preserve validity. That is, a rule applied to a valid formula (or two valid formulas, in the case of modus ponens) will always yield another valid formula. Essentially, this amounts to formalising the intuitive justifications given for the various axioms and rules when we first introduced them. We are only going to sketch a proof here for one of the axioms and one of the rules.

---

[2] This includes in particular infinite sets of formulas. Observe that any finite set can be identified with the conjunction of its elements, so the case of finite sets can be reduced to the case of simple formulas.

[3] Observe that 'every satisfiable formula is consistent' and 'every theorem is a valid formula' are equivalent claims. To see this, consider firstly that whenever every satisfiable formula is consistent, then satisfiability of $\neg\varphi$ implies consistency of $\neg\varphi$, i.e. φ not being valid implies φ not being a theorem, which is the contrapositive of the second claim. For the other direction consider that whenever theoremhood implies validity, then $\neg\varphi$ being a theorem implies validity of $\neg\varphi$, i.e. inconsistency of φ implies that φ cannot be satisfiable. The latter again means that satisfiability of φ implies consistency of φ.

Here is how we can prove validity of the interaction axiom (X1): $\Diamond_{\rightarrow} A \rightarrow \bigcirc_{\uparrow} \Diamond_{\downarrow} A$. Let $\mathcal{M} = (T, R, S, V)$ be an arbitrary ordered tree model and let $t \in T$ be an arbitrary node in the underlying tree. We need to show that $\mathcal{M}, t \models \Diamond_{\rightarrow} \varphi$ implies $\mathcal{M}, t \models \bigcirc_{\uparrow} \Diamond_{\downarrow} \varphi$ for any given formula φ. By Definition 3.6 (of the semantics of our operators), $\mathcal{M}, t \models \Diamond_{\rightarrow} \varphi$ holds iff t has got a right sibling t' with $\mathcal{M}, t' \models \varphi$. This means, in particular, that we have $(t, t') \in S$. By Definition 3.4 (of ordered trees), we have $S \subseteq R^{-1} \circ R$, i.e. there exists a node $t'' \in T$ with $(t, t'') \in R^{-1}$ and $(t'', t') \in R$. The latter means that t' is a child of the node t''. As we also have $\mathcal{M}, t' \models \varphi$, by definition of the semantics of the operator $\Diamond_{\downarrow}$, we can now infer that $\mathcal{M}, t'' \models \Diamond_{\downarrow} \varphi$ must be the case. But t'' is the parent of t, so finally, by definition of the semantics of $\bigcirc_{\uparrow}$, we get $\mathcal{M}, t \models \bigcirc_{\uparrow} \Diamond_{\downarrow} \varphi$ as required.

As an example for proving the soundness of our proof rules, we shall show that applying the generalisation rule with respect to the dual of the next-operator $\bigcirc_{\rightarrow}$ to a valid formula will always yield another valid formula. So let φ be a valid (but otherwise arbitrary) formula, i.e. φ is true at every node in every model. We need to show that this entails that $\neg \bigcirc_{\rightarrow} \neg \varphi$ must be valid as well. So, for the sake of contradiction, assume this is not the case, i.e. there exist a model and a node t in that model such that $\neg \bigcirc_{\rightarrow} \neg \varphi$ does not hold at t. Then $\bigcirc_{\rightarrow} \neg \varphi$ must be true at t, i.e. t must have a righthand neighbour at which $\neg \varphi$ is true. But this contradicts our original assumption of φ being true at every node in every model. Hence, generalisation with respect to $\neg \bigcirc_{\rightarrow} \neg$ is indeed a validity preserving inference rule.

Soundness of the remaining axioms and rules can be established in a similar manner as for the two examples given here. □

Compactness and completeness. We say that a logic has the *compactness property* iff a set of formulas is satisfiable whenever all of its finite subsets are. This is directly relevant to the kind of completeness result we can expect. Given that the derivation rules in our proof system are finitary (and that a proof is a finite sequence of proof steps), strong completeness implies compactness.

Proposition 4.4 (Compactness failure) *The modal logic of ordered trees lacks the compactness property.*

Proof. It suffices to find an example of an infinite set of formulas that, as a whole, is not satisfiable, but for which every finite subset does have a model. Consider the following infinite set of formulas:

$$\Delta \;=\; \{\bigcirc_{\uparrow}\top,\ \bigcirc_{\uparrow}\bigcirc_{\uparrow}\top,\ \bigcirc_{\uparrow}\bigcirc_{\uparrow}\bigcirc_{\uparrow}\top,\ \bigcirc_{\uparrow}\bigcirc_{\uparrow}\bigcirc_{\uparrow}\bigcirc_{\uparrow}\top,\ \ldots\}$$

The nth formula in Δ is true at a node t iff there are at least n nodes above t in the tree. We can easily construct a model for any given finite subset of Δ. If n is the length of the longest formula in the subset then any model based on a tree with a branch of (at least) length n will be a model for that subset.

For Δ itself, on the other hand, there can be no model. Assume otherwise, that is, assume there is a node t where all formulas in Δ are satisfied. If the distance from the root of the tree down to t is n, then any formula (in Δ) longer than n will not be satisfied at t, which contradicts our assumption. Hence, our logic does not have the compactness property. $\square$

In fact, our proof shows that any fragment of **OTL** that contains at least the parent modality $\bigcirc$ lacks the compactness property. As discussed above, this compactness failure rules out a *general* strong completeness result for our logic.

We shall return to the issue of completeness in Section 4.3. Before that we are going to derive a number of useful theorems from our axioms.

4.2 Some Derived Theorems

In this section we are going to establish a number of theorems that can be derived in our axiomatic proof system.

Distribution. Axioms (K1)–(K8), together with modus ponens, allow us to *distribute* the modality in question with respect to implication, at least in one direction. For example, given $\Box(A \to B)$ we may infer $\Box A \to \Box B$ using axiom (K3) and modus ponens. (But the opposite direction would *not* constitute a valid inference.) When deriving interesting theorems, we need to draw inferences of this kind all the time. Therefore, the first three lemmas in this section are devoted to extending the range of such distribution laws available to us, both in terms of the modal operator and the propositional connective involved. The first one of these lemmas states that theorems of the same structure as the (K) axioms hold for duals of next-operators.

Lemma 4.5 (K for duals of next-operators) *Let Ω be a next-operator and let A and B be formulas. Then $\neg\Omega\neg(A \to B) \to (\neg\Omega\neg A \to \neg\Omega\neg B)$ is a theorem.*

Proof. For better readability, we are first going to provide a derivation of the supposed theorem for the particular next-operator $\bigcirc$, rather than directly giving a general proof.

(1)	$\bigcirc\top \vee \neg\bigcirc\top$	propositional tautology
(2)	$\neg\bigcirc\neg(A \to B) \wedge \bigcirc\top \to \bigcirc(A \to B)$	from (F3)
(3)	$\neg\bigcirc\neg A \wedge \bigcirc\top \to \bigcirc A$	from (F3)
(4)	$\neg\bigcirc\neg(A \to B) \wedge \neg\bigcirc\neg A \wedge \bigcirc\top \to \bigcirc B$	from (2), (3) and (K8)
(5)	$\bigcirc B \to \neg\bigcirc\neg B$	from (F3)
(6)	$\bigcirc\top \to (\neg\bigcirc\neg(A \to B) \to (\neg\bigcirc\neg A \to \neg\bigcirc\neg B))$	from (4) and (5)
(7)	$\bigcirc\neg B \to \bigcirc\top$	from (F3)
(8)	$\neg\bigcirc\top \to \neg\bigcirc\neg B$	from (7)
(9)	$\neg\bigcirc\neg(A \to B) \to (\neg\bigcirc\neg A \to \neg\bigcirc\neg B)$	from (1), (6) and (8)

As we have corresponding axioms and rules available for $\circleddash$ and $\ominus$ as well, the derivation can easily be adapted for the other two cases. □

The next lemma shows that for all those modal operators for which we have either an axiom of the (K) variety or a derived theorem of the same structure, the same types of formulas with iterated modalities are also theorems.

Lemma 4.6 (K for iterated modalities) *Let Ω be a box-operator, a next-operator, or the dual of a next-operator, let A and B be formulas, and let $n \in \mathbb{N}$. Then the formula $\Omega^n(A \to B) \to (\Omega^n A \to \Omega^n B)$ is a theorem.*

Proof. Again, for better readability, we are first going to give proofs of the different parts of the claim for particular operators rather than using a general notation. The results will immediately generalise to all box- and next-operators, respectively, as we have the same types of axioms and rules available to us for all of them.

As an example for an iterated box-operator, we are going to show that the formula $\Box^n(A \to B) \to (\Box^n A \to \Box^n B)$ is a theorem for all formulas A and B and all natural numbers $n \in \mathbb{N}$. The proof proceeds by induction over n. For the base case of $n = 1$ the claim simply reduces to axiom (K3): $\Box(A \to B) \to (\Box A \to \Box B)$.

In the induction step, we need to show how we may derive $\Box^{n+1}(A \to B) \to (\Box^{n+1} A \to \Box^{n+1} B)$ under the assumption that $\Box^n(A \to B) \to (\Box^n A \to \Box^n B)$ has already be shown to be a theorem. Here is such a derivation:

(1)	$\Box^n(A \to B) \to (\Box^n A \to \Box^n B)$	induction hypothesis
(2)	$\Box(\Box^n(A \to B) \to (\Box^n A \to \Box^n B))$	from (1) by generalisation
(3)	$\Box\Box^n(A \to B) \to \Box(\Box^n A \to \Box^n B)$	from (2) and (K3)
(4)	$\Box(\Box^n A \to \Box^n B) \to (\Box\Box^n A \to \Box\Box^n B)$	from (K3)
(5)	$\Box\Box^n(A \to B) \to (\Box\Box^n A \to \Box\Box^n B)$	from (3) and (4)
(6)	$\Box^{n+1}(A \to B) \to (\Box^{n+1} A \to \Box^{n+1} B)$	from (5)

This concludes our proof for the case of the box-operator $\Box$. The same argument also applies to the other box-operators of our language, as the required axiom as well as the generalisation rule are equally available for all of them. In fact, the proof given also covers the case of the duals of the three next-operators, because Lemma 4.5 provides the theorems for the base cases and we have generalisation rules with respect to duals of next-operators in our system.

It remains to show that the claim also holds for next-operators themselves. As an example for a next-operator we are going to use $\ominus$. Our inductive proof is similar to the previous one. The base case immediately reduces to axiom (K7): $\ominus(A \to B) \to (\ominus A \to \ominus B)$. The derivation for the induction step differs a little, because the generalisation rule for next-operators applies to the dual of the operator rather than the operator itself. This can be compensated for by the corresponding functionality axiom, which in the case of $\ominus$ is (F2): $\ominus A \leftrightarrow (\neg \ominus \neg A \land \ominus \top)$. Here is our derivation for the induction step:

(1)	$\ominus^n(A \to B) \to (\ominus^n A \to \ominus^n B)$	induction hypothesis
(2)	$\neg \ominus \neg(\ominus^n(A \to B) \to (\ominus^n A \to \ominus^n B))$	from (1) by generalisation
(3)	$\neg \ominus \neg \ominus^n(A \to B) \to \neg \ominus \neg(\ominus^n A \to \ominus^n B)$	from (2) and Lemma 4.5
(4)	$\ominus \ominus^n(A \to B) \to \neg \ominus \neg \ominus^n(A \to B) \wedge \ominus \top$	from (F2)
(5)	$\neg \ominus \neg(\ominus^n A \to \ominus^n B) \wedge \ominus \top \to \ominus(\ominus^n A \to \ominus^n B)$	from (F2)
(6)	$\ominus \ominus^n(A \to B) \to \ominus(\ominus^n A \to \ominus^n B)$	from (3), (4) and (5)
(7)	$\ominus(\ominus^n A \to \ominus^n B) \to (\ominus \ominus^n A \to \ominus \ominus^n B)$	from (K7)
(8)	$\ominus^{n+1}(A \to B) \to (\ominus^{n+1} A \to \ominus^{n+1} B)$	from (6) and (7)

This concludes our inductive proof for $\ominus^n(A \to B) \to (\ominus^n A \to \ominus^n B)$. Again, the same type of argument applies to all next-operators, so we are done. $\square$

Next we establish what kind of (possibly iterated) modalities may be distributed with respect to conjunctions and disjunctions, respectively.

Lemma 4.7 (Distribution) *Let A and B be formulas and let $n \in \mathbb{N}$.*

(1) If Ω is a box-operator or the dual of a next-operator, then $\Omega^n(A \wedge B) \leftrightarrow (\Omega^n A \wedge \Omega^n B)$ is a theorem.

(2) If Ω is a diamond-operator or a next-operator, then $\Omega^n(A \vee B) \leftrightarrow (\Omega^n A \vee \Omega^n B)$ is a theorem.

Proof. In this proof, we are again going to exploit the fact that, that we have the same kinds of rules, axioms, and derived theorems available to us for the different modal operators of each type. For instance, if Ω is a box-operator or the dual of a next-operator then, by Lemma 4.6, the formula $\Omega^n(A \to B) \to (\Omega^n A \to \Omega^n B)$, which has the structure of a (K) axiom, will be a theorem for any natural number n. Furthermore, as we may repeat the application of a generalisation rule as often as we wish, generalisation with respect to an iterated box-operator Ω^n is a valid rule in our proof system. The same applies to iterated applications of the generalisation rule with respect to the dual of a next-operator. Therefore, as long as we only make use of (K) axioms (or their derived counterparts) and generalisation rules, we can deal with all box-operators and the duals of all next-operator of our language in one go. The following derivation proves the first claim for the particular case of the box-operator $\boxminus$.

(1)	$(A \wedge B) \to A$	propositional tautology
(2)	$\boxminus((A \wedge B) \to A)$	from (1) by generalisation
(3)	$\boxminus(A \wedge B) \to \boxminus A$	from (2) and (K1)
(4)	$\boxminus(A \wedge B) \to \boxminus B$	in analogy to (1)–(3)
(5)	$A \to (B \to (A \wedge B))$	propositional tautology
(6)	$\boxminus(A \to (B \to (A \wedge B)))$	from (5) by generalisation
(7)	$\boxminus A \to \boxminus(B \to (A \wedge B))$	from (6) and (K1)
(8)	$\boxminus(B \to (A \wedge B)) \to (\boxminus B \to \boxminus(A \wedge B))$	from (K1)
(9)	$(\boxminus A \wedge \boxminus B) \to \boxminus(A \wedge B)$	from (7) and (8)
(10)	$\boxminus(A \wedge B) \leftrightarrow (\boxminus A \wedge \boxminus B)$	from (3), (4) and (9)

As argued before, this shows that $\Omega^n(A \wedge B) \leftrightarrow (\Omega^n A \wedge \Omega^n B)$ must be a theorem for all $n \in \mathbb{N}$ whenever Ω is either a box-operator or the dual of a next-operator.

Part (2) concerns diamond-operators and next-operators. The two groups *could* also be treated under a single heading, but the proof may be clearer if we separate the two. So let us begin with the case of diamond-operators. We can easily derive iterated versions of the duality axioms (D1)–(D5), like for example $\boxminus^n A \leftrightarrow \neg \Diamond^n \neg A$. The following derivation only makes use of such a duality axiom and the result proved in part (1):

(1)	$\boxminus(\neg A \wedge \neg B) \leftrightarrow (\boxminus \neg A \vee \boxminus \neg B)$	from Part (1) of this lemma
(2)	$\neg \Diamond \neg(\neg A \wedge \neg B) \leftrightarrow (\neg \Diamond A \vee \neg \Diamond B)$	from (1) and (D1)
(3)	$\Diamond(A \vee B) \leftrightarrow (\Diamond A \vee \Diamond B)$	from (2)

This shows that the formula $\Omega^n(A \vee B) \leftrightarrow (\Omega^n A \vee \Omega^n B)$ will be a theorem for any diamond-operator Ω.

Finally, we turn to the case of next-operators. Here all that is required is a simple propositional transformation of the result proved in part (1). As an example, we give the derivation for the case of the operator $\ominus$:

(1)	$\neg \ominus \neg(\neg A \wedge \neg B) \leftrightarrow (\neg \ominus \neg\neg A \wedge \neg \ominus \neg\neg B)$	form Part (1) of this lemma
(2)	$\ominus(A \vee B) \leftrightarrow (\ominus A \vee \ominus B)$	from (1)

Hence, $\Omega^n(A \vee B) \leftrightarrow (\Omega^n A \vee \Omega^n B)$ will also be a theorem whenever Ω is one of the next-operators of our modal language. $\square$

From now on, when referring to Lemma 4.7,we will usually simply say that the respective conclusion follows 'by distribution'. While in the above lemma, the distribution laws have been formulated for the case of only two propositions A and B, it can easily be generalised to any number of propositions. For example, if $A_1, \ldots, A_n$ are any finite number of formulas then $\boxminus(A_1 \wedge \cdots \wedge A_n) \leftrightarrow (\boxminus A_1 \wedge \cdots \wedge \boxminus A_n)$ will be a theorem. To obtain this result we require $n - 1$ instances of the derived theorem $\boxminus(A \wedge B) \leftrightarrow (\boxminus A \wedge \boxminus B)$.

More theorems. We are going to require the derived theorems listed in the following lemma at various stages throughout our completeness proof later on.

Lemma 4.8 (Derived theorems) *The following formulas are theorems:*

(1) $\vdash \neg(\boxdot \bot \wedge \odot A)$

(2) $\vdash \Diamond \odot A \rightarrow A$

(3) $\vdash \Diamond \boxdot A \rightarrow A$

(4) $\vdash \boxdot A \rightarrow \boxdot \boxminus A$

(5) $\vdash (\boxminus A \wedge \ominus \top) \rightarrow \ominus A$

(6) $\vdash \ominus \boxminus A \rightarrow \boxminus \boxminus A$

(7) $\vdash \ominus(A \wedge \boxminus A) \to \boxminus A$

Proof. The derivations may be found in Appendix B. □

In the following lemma we prove generalisations of the mixing axioms (M3) and (M4). These will allow us, in certain cases, to rewrite a formula with a transitive diamond-operator as a disjunction of formulas with iterated next-operators (or intransitive diamond-operators, respectively).

For the case of $n = 4$, for instance, the first part of Lemma 4.9 asserts that the following formula is a theorem:

$$\Diamond_{\uparrow} A \leftrightarrow (\bigcirc_{\uparrow} A \vee \bigcirc_{\uparrow}^2 A \vee \bigcirc_{\uparrow}^3 A \vee \bigcirc_{\uparrow}^4 A \vee \bigcirc_{\uparrow}^4 \Diamond_{\uparrow} A)$$

So if we can somehow eliminate the disjunct $\bigcirc_{\uparrow}^4 \Diamond_{\uparrow} A$, then we can rewrite the formula $\Diamond_{\uparrow} A$ as the disjunction $\bigcirc_{\uparrow} A \vee \bigcirc_{\uparrow}^2 A \vee \bigcirc_{\uparrow}^3 A \vee \bigcirc_{\uparrow}^4 A$. In other words, we can eliminate the diamond-operator $\Diamond_{\uparrow}$, which refers to the transitive closure of the parent relation, and use instead the simpler next-operator $\bigcirc_{\uparrow}$, which only refers to the parent relation itself. As we shall see in the sequel, axiomatising the semantics of modalities defined in terms of a transitive closure is particularly difficult, so this kind of construction can be very useful.

Lemma 4.9 (Mixing iterated modalities) *The following formulas are theorems for all natural numbers $n \in \mathbb{N}$:*

(1) $\vdash \Diamond_{\uparrow} A \leftrightarrow (\bigcirc_{\uparrow} A \vee \cdots \vee \bigcirc_{\uparrow}^n A \vee \bigcirc_{\uparrow}^n \Diamond_{\uparrow} A)$

(2) $\vdash \Diamond_{\downarrow}^+ A \leftrightarrow (\Diamond_{\downarrow} A \vee \cdots \vee \Diamond_{\downarrow}^n A \vee \Diamond_{\downarrow}^n \Diamond_{\downarrow}^+ A)$

Proof. Part (1) of the claim can be proved by induction over the maximal index n. For the base case, that is for $n = 1$, the theorem to be shown reduces to our axiom (M3): $\Diamond_{\uparrow} A \leftrightarrow (\bigcirc_{\uparrow} A \vee \bigcirc_{\uparrow} \Diamond_{\uparrow} A)$. For the induction step, we need to show that, under the assumption that the formula $\Diamond_{\uparrow} A \leftrightarrow (\bigcirc_{\uparrow} A \vee \cdots \vee \bigcirc_{\uparrow}^n A \vee \bigcirc_{\uparrow}^n \Diamond_{\uparrow} A)$ is known to be a theorem for a particular n, we can derive the corresponding formula for the case of $n + 1$. The following derivation does the job:

(1)	$\Diamond_{\uparrow} A \leftrightarrow (\bigcirc_{\uparrow} A \vee \cdots \vee \bigcirc_{\uparrow}^n A \vee \bigcirc_{\uparrow}^n \Diamond_{\uparrow} A)$	induction hypothesis
(2)	$\bigcirc_{\uparrow}^n \Diamond_{\uparrow} A \leftrightarrow \bigcirc_{\uparrow}^n (\bigcirc_{\uparrow} A \vee \bigcirc_{\uparrow} \Diamond_{\uparrow} A)$	from (M3)
(3)	$\bigcirc_{\uparrow}^n \Diamond_{\uparrow} A \leftrightarrow (\bigcirc_{\uparrow}^{n+1} A \vee \bigcirc_{\uparrow}^{n+1} \Diamond_{\uparrow} A)$	from (2) by distribution
(4)	$\Diamond_{\uparrow} A \leftrightarrow (\bigcirc_{\uparrow} A \vee \cdots \vee \bigcirc_{\uparrow}^{n+1} A \vee \bigcirc_{\uparrow}^{n+1} \Diamond_{\uparrow} A)$	from (1) and (3)

Hence, our generic formula must be a theorem for all $n \in \mathbb{N}$ as claimed.

Part (2) of Lemma 4.9 can be proved analogously, this time using axiom (M4) in place of (M3). □

4.3 Completeness without Descendant Modalities

The aim of this section is to prove completeness of our axiom system with respect to the model-theoretic semantics of ordered tree logics. In fact, we are only going to show completeness for the fragment of our logic excluding formulas involving the transitive descendant modality $\Diamond^{+}$ and its dual $\Box^{+}$.[4] We shall also try to give an explanation *why* taming this operator is considerably more difficult than axiomatising the rest of the language. There is also a second restriction to our completeness result (that is in fact related to the problem of axiomatising the descendant operators), which we are going to explain once the necessary terminology has been introduced.[5]

Completeness via canonical models. To prove our Completeness Theorem we are going to use a variant of a powerful method that involves building a so-called *canonical model*.[6] The earliest accounts of completeness results for modal logics via a canonical model argument are probably the works of Lemmon and Scott [48] and Makinson [50].

The basic idea underlying this method is relatively simple and may be informally described as follows. Recall that the objective of proving completeness is to show that any given formula φ that is known to be consistent (with respect to the axiom system in question) must also be satisfiable, that is, there must be a model in which φ is true. One way of proving satisfiability is to actually construct a model for φ. The first step in building a model is to decide on a suitable set of worlds (or nodes in the case of our logic). In the canonical model approach, we construct a world for every combination of formulas that could possibly be true 'at the same time'. More precisely, each world is a consistent set of formulas that is maximal in the sense that adding any more formulas would render it inconsistent. Clearly, if φ is a consistent formula then it must be 'true' at some of these worlds, that is, φ must be an element of some of these maximally consistent sets of formulas. The first part of our proof will be devoted to a formal definition of maximally consistent sets of formulas and to proving a number of useful properties they enjoy (see Definition 4.10 on page 115 and the following two lemmas).

Next we have to define suitable accessibility relations over our set of worlds. These need to be arranged in such a way that modal operators 'work' as intended. For instance, if $\Box\varphi$ is an element of the set Δ associated with a particular world, then we only want to make those worlds accessible from Δ where φ is an element of the associated set of formulas. In fact, this is precisely how we are going to define our relations (see, for instance, Definition 4.13 on page 118). This ensures that the semantic truth conditions for formulas starting with a box-operator coincide with the set-membership relation

[4] Consequently, we are not going to use any of the axioms (or the generalisation rule) proposed earlier that involve either $\Diamond^{+}$ or $\Box^{+}$. These axioms have been included into our presentation in Section 4.1 only in order to provide a tentative outline of what a complete axiomatisation for the full logic *might* look like. For further discussion of this point we refer to Section 4.4.

[5] See, in particular, our discussion on page 138, just before the main theorem of this section.

[6] This method is sometimes also referred to as a *Henkin-construction*, because of its similarity to Henkin's completeness proof for first-order logic [39].

between formulas and maximally consistent sets *by definition*, as it were. One of the central tasks in a canonical model based completeness proof is to show that this is also the case for the other modalities (as well as the propositional connectives). The difficult part is usually to prove that in the case of an existential modality (that is, a diamond- or a next-operator) there actually exists a maximally consistent set of formulas, which is related to the first set and which contains a suitable witness formula. We have to prove a so-called *Existence Lemma* for each one of our diamond- and next-operators (see Lemma 4.19 on page 121 for an example). At the end of the day, we need to show that a formula φ belongs to a given maximally consistent set of formulas iff φ is satisfied at the corresponding world in our canonical model. This important intermediate result is often stated in the form of a so-called *Truth Lemma*. (Our Truth Lemma is Lemma 4.46 on page 136.)

Besides showing that modal operators work as intended in a canonical model, we also need to show that the frame underlying the model has in fact the right structure. As the structure imposed on the worlds of our canonical model depends on the kinds of formulas found in the maximally consistent sets associated with these worlds, in the first instance, there is no guarantee that this structure has the properties postulated for frames in the model-theoretic semantics given for the logic under investigation. In our case, we would like this frame to be an ordered tree. In some cases, it is possible to prove that the canonical frame has all the required properties directly; in the case of our logic the situation is a little more difficult. In fact, given the result at the end of the previous chapter, namely that the logic of ordered trees would actually be described more accurately as a logic of *loosely* ordered trees,[7] we cannot expect to be able to uniquely axiomatise ordered trees. That is, we cannot expect our canonical frame (the structure of which is a direct consequence of the characteristics of our axiomatic system) to be an ordered tree. However, as we shall see, we *can* prove that it (or rather part of it, to be precise) is a loosely ordered tree. Given that any formula that is satisfiable in a model based on a loosely ordered tree must also be satisfiable in a model based on an ordered tree, this is just as good a result. (Lemma 4.42 on page 134 states that our canonical frame is a loosely ordered tree and much of the work before that is devoted to proving specific properties of the canonical accessibility relations.)

A final twist to our completeness proof is that we do in fact not use the full canonical model (over *all* maximally consistent sets of formulas) described before. The reason is that we cannot expect the full canonical frame to be irreflexive with respect to the vertical accessibility relation (the canonical child relation). As is well known, giving an axiomatic characterisation of a modal operator pertaining to an irreflexive accessibility relation is not possible in the general case. In our case, the fact that the intended frame (a tree) has got a first node with respect to the canonical child relation, namely the root node, allows us to overcome this problem by means of the following 'trick'. In a tree,

[7] Loosely ordered trees have been defined in Definition 3.33 and the quoted result has been discussed after Theorem 3.41 on page 93.

every node has got a particular distance from the root which, for this purpose, we are going to refer to as the *level* of that node. If certain maximally consistent sets and the vertical accessibility relation declared over them really do give rise to a tree as claimed, then each one of these sets will contain a formula reflecting the level of the corresponding node. For instance, a node on level 2 (i.e. a 'grandchild' of the root node) must satisfy the formula $\odot\odot\boxdot\bot$ (or $\odot^2\boxdot\bot$ in short). In general, any node on level n will satisfy $\odot^n\boxdot\bot$, but not $\odot^m\boxdot\bot$ for any number $m \neq n$. Furthermore, as will be shown formally in Lemma 4.17, whenever a maximally consistent set Δ has got level n (satisfies $\odot^n\boxdot\bot$) then any set accessible from Δ via the canonical child relation will have level $n+1$, that is, it will satisfy $\odot^{n+1}\boxdot\bot$. Hence, if we discard any maximally consistent sets of formulas not containing a formula of this form, we automatically guarantee the irreflexivity of the remaining frame.

Finally, the actual completeness result itself will be stated as Theorem 4.47 on page 139. While most of this section is rather technical in nature, in particular in those parts where we establish the various structural properties of our canonical frame through a sequence of lemmas, we are also going to provide further explanations on the details of our construction as we go along.

Maximally consistent sets. Let us first make the idea of maximally consistent sets precise. By an extension of a set of formulas Δ we mean a superset of Δ that is itself a set of formulas.

Definition 4.10 (Maximally consistent sets) *A consistent set of formulas Δ is called maximally consistent iff every proper extension of Δ is inconsistent.*

It should be pointed out that this is not the only way of defining maximally consistent sets of formulas. Hughes and Cresswell [42], for instance, define maximally consistent sets[8] as consistent sets containing either φ of $\neg\varphi$ for every formula φ in the language. This characterisation will in fact emerge as a consequence of the one chosen here (see Lemma 4.11).

The following two lemmas, both of which concern maximally consistent sets, are standard results in modal logic.

Lemma 4.11 (Basic properties of maximally consistent sets) *Let Δ be a maximally consistent set and let φ, φ_1 and φ_2 be formulas. Then the following statements are true:*

(1) If $\vdash \varphi$ then $\varphi \in \Delta$.

(2) $\neg\varphi \in \Delta$ iff $\varphi \notin \Delta$.

(3) If $\varphi_1 \in \Delta$ and $\varphi_1 \rightarrow \varphi_2 \in \Delta$ then also $\varphi_2 \in \Delta$.

[8] When speaking about 'maximally consistent sets', we always mean maximally consistent sets *of formulas*, even when this is not stated explicitly.

(4) $\varphi_1 \wedge \varphi_2 \in \Delta$ *iff* $\varphi_1 \in \Delta$ *and* $\varphi_2 \in \Delta$.

(5) $\varphi_1 \vee \varphi_2 \in \Delta$ *iff* $\varphi_1 \in \Delta$ *or* $\varphi_2 \in \Delta$.

Proof. Throughout this proof, let Δ be a maximally consistent set of formulas and let φ, φ_1 and φ_2 be formulas.

(1) *Theorems.* Suppose we have $\vdash \varphi$, that is, φ is a theorem. We are going to show that the assumption $\varphi \notin \Delta$ will lead to a contradiction. Given that Δ is maximally consistent, Δ itself is a consistent set, but every proper extension of Δ, in particular the set $\Delta \cup \{\varphi\}$, will be inconsistent. So Δ has no finite subset that is inconsistent, but $\Delta \cup \{\varphi\}$ does. Hence, any such inconsistent subset of the latter must contain φ. In other words, there exist formulas $\psi_1, \ldots, \psi_n \in \Delta$ such that $\vdash \neg(\psi_1 \wedge \cdots \wedge \psi_n \wedge \varphi)$ holds, but at the same time we do *not* have $\vdash \neg(\psi_1 \wedge \cdots \wedge \psi_n)$. Using propositional reasoning, we can see that the former implies that either $\vdash \neg(\psi_1 \wedge \cdots \wedge \psi_n)$ or $\vdash \neg\varphi$ must be true. In either case we obtain a direct contradiction.

(2) *Negation.* The formula $\neg(\varphi \wedge \neg\varphi)$ is a tautology, whatever the truth value of φ. So, by Definition 4.2, any set of formulas containing both φ and $\neg\varphi$ is bound to be inconsistent. Hence, $\neg\varphi \in \Delta$ certainly implies $\varphi \notin \Delta$.

For the opposite direction, assume $\varphi \notin \Delta$. We have to prove that this implies $\neg\varphi \in \Delta$. In eager anticipation of a contradiction, let us assume Δ contains neither φ nor $\neg\varphi$. Then both $\Delta \cup \{\varphi\}$ and $\Delta \cup \{\neg\varphi\}$ would have to be inconsistent (but Δ on its own would still be consistent). So there exist formulas $\chi_1, \ldots, \chi_m \in \Delta$ as well as formulas $\psi_1, \ldots, \psi_n \in \Delta$ such that $\vdash \neg(\chi_1 \wedge \cdots \wedge \chi_m \wedge \varphi)$ and $\vdash \neg(\psi_1 \wedge \cdots \wedge \psi_n \wedge \neg\varphi)$. But these propositionally entail $\vdash \neg(\chi_1 \wedge \cdots \wedge \chi_m \wedge \psi_1 \wedge \cdots \wedge \psi_n)$, which contradicts our assumption of Δ being a consistent set.

(3) *Modus ponens.* Claiming that $\varphi_1 \in \Delta$ and $\varphi_1 \to \varphi_2 \in \Delta$ imply $\varphi_2 \in \Delta$ means claiming that Δ is closed under modus ponens. Suppose we have $\varphi_1 \in \Delta$ and $\varphi_1 \to \varphi_2 \in \Delta$. By the previous case, assuming $\varphi_2 \notin \Delta$ would imply $\neg\varphi_2 \in \Delta$. But $\vdash \neg(\varphi_1 \wedge (\varphi_1 \to \varphi_2) \wedge \neg\varphi_2)$ by propositional reasoning, i.e. such an assumption would render Δ inconsistent.

(4) *Conjunction.* The formula $\varphi_1 \wedge \varphi_2 \to \varphi_1$ is a propositional tautology and thereby also theorem. Hence, by case (1), $\varphi_1 \wedge \varphi_2 \to \varphi_1$ is an element of Δ and then, by case (3), $\varphi_1 \wedge \varphi_2 \in \Delta$ implies $\varphi_1 \in \Delta$. That $\varphi_2 \in \Delta$ also follows from $\varphi_1 \wedge \varphi_2 \in \Delta$ can be shown by the same type of argument.

For the other direction, assume we have both $\varphi_1 \in \Delta_2$ and $\varphi_2 \in \Delta$. As $\varphi_1 \to (\varphi_2 \to (\varphi_1 \wedge \varphi_2))$ is a theorem, the former entails $\varphi_2 \to (\varphi_1 \wedge \varphi_2) \in \Delta$ and then the latter allows us to infer $\varphi_1 \wedge \varphi_2 \in \Delta$.

(5) *Disjunction.* $\varphi_1 \vee \varphi_2 \in \Delta$ iff $\varphi_1 \in \Delta$ or $\varphi_2 \in \Delta$ can be shown to hold by an argument similar to the previous case.

This completes our proof of Lemma 4.11. □

We are going to exploit these properties of maximally consistent sets throughout, usually without specific reference to Lemma 4.11.

The following result is known as *Lindenbaum's Lemma* (see, for instance, [8]) It establishes that for every consistent set of formulas we have a maximally consistent set containing that set. As this includes the case of singleton consistent sets, it also shows that every consistent formula must belong to a (at least one) maximally consistent set. Last but not least, this lemma shows that maximally consistent sets actually exist. This must be so, because we know that consistent formulas do exist ($\top$ for instance).

Lemma 4.12 (Existence of maximally consistent supersets) *Every consistent set of formulas can be extended to a maximally consistent set.*

Proof. Let Δ be a consistent set of formulas. We are going to show how we can construct a maximally consistent set Δ^* with $\Delta \subseteq \Delta^*$. The basic idea is to decide for every well-formed formula whether to add either that formula or its negation to the set. This is possible, because the set of all well-formed formulas in our language is countable. That is, there exists an enumeration $\varphi_1, \varphi_2, \varphi_3, \ldots$ of these formulas. We define an infinite sequence (Δ_n) of supersets of Δ as follows:

$$\begin{array}{rcl} \Delta_0 & = & \Delta \\ \Delta_{n+1} & = & \begin{cases} \Delta_n \cup \{\varphi_{n+1}\} & \text{if that set is consistent} \\ \Delta_n \cup \{\neg\varphi_{n+1}\} & \text{otherwise} \end{cases} \end{array}$$

By induction over n we can show that every element in this sequence must be a consistent set. To start with, Δ_0 is consistent by our original assumption of Δ being a consistent set. For the induction step, we need to show that consistency of Δ_n entails consistency of Δ_{n+1}. By construction of our sequence, Δ_{n+1} will only be inconsistent if there are formulas $\chi_1, \ldots, \chi_k \in \Delta_n$ as well as $\psi_1, \ldots, \psi_m \in \Delta_n$ such that both $\vdash \neg(\chi_1 \wedge \cdots \wedge \chi_k \wedge \varphi_{n+1})$ and $\vdash \neg(\psi_1 \wedge \cdots \wedge \psi_m \wedge \neg\varphi_{n+1})$ hold. It then follows (by propositional reasoning) that also $\vdash \neg(\chi_1 \wedge \cdots \wedge \chi_k \wedge \psi_1 \wedge \cdots \wedge \psi_m)$ must be the case. But then already Δ_n would have to be inconsistent. Hence, consistency of Δ_n indeed implies consistency of Δ_{n+1}, which means that all elements of (Δ_n) must be consistent sets of formulas.

We can now define Δ^* as the union of all sets in the sequence (Δ_n). Then Δ^* must be consistent, because any inconsistent finite subset would also be a subset of some Δ_n, which would contradict the consistency of Δ_n. Furthermore, Δ^* is maximal, as every formula φ_n in our enumeration will either be an element of Δ_n, and thereby also of Δ^*, or adding φ_n to Δ^* would render the set inconsistent. And finally, we have $\Delta \subseteq \Delta^*$ by construction. □

The proofs of Lemma 4.11 and Lemma 4.12 do not make reference to any of our modal axioms. In the case of the former, we have only used basic propositional reasoning (that is, we have implicitly referred to axioms of propositional logic). For the latter, besides propositional reasoning, we have also made use of the fact that the set of well-formed formulas is countable. Therefore, these results hold not only for our full modal language, but also for fragments, where we drop one or more of our modal operators.

Building the canonical model. As far as possible, we are going to separate the presentation of intermediate results into those pertaining to *vertical* modalities and those pertaining to *horizontal* modalities. We start with the vertical dimension, and are first going to define a vertical accessibility relation over maximally consistent sets, the worlds of our canonical model *in spe*. Following this definition (and a brief discussion of some of its properties), we are going to prove 'vertical' Existence Lemmas for the child operator $\Diamond_{\downarrow}$, the parent operator $\bigcirc_{\uparrow}$, and the ancestor operator $\Diamond_{\uparrow}$. To be able to prove the latter, we are going to require the notion of a level of a maximally consistent set, which has already been discussed informally in the introduction to this section. We are also going to show that our vertical accessibility relation imposes a tree structure on the set of those maximally consistent sets that have a level.

We shall refer to our vertical accessibility relation over maximally consistent sets of formulas as the *canonical child relation*. Our aim later on will be to show that it corresponds to the child relation R in a loosely ordered tree.

Definition 4.13 (Canonical child relation) *Let Δ_1 and Δ_2 be maximally consistent sets. We define the canonical child relation $\prec$ over maximally consistent sets as follows:*

$$\Delta_1 \prec \Delta_2 \quad \textit{iff} \quad \Box_{\downarrow}\varphi \in \Delta_1 \textit{ implies } \varphi \in \Delta_2 \textit{ for all formulas } \varphi$$

The non-reflexive transitive closure of the relation $\prec$ will be denoted as $\prec^+$ and the reflexive transitive closure as $\prec^*$ (see also Definition A.3 in Appendix A).

In the next lemma we state some simple consequences of this definition of the relation $\prec$. They concern what, for lack of a better name, we shall call *awareness*. For instance, if we have $\Delta_1 \prec \Delta_2$ and some formula φ is an element of Δ_2, then Δ_1 will be 'aware' of this characteristic of one of its $\prec$-successors and contain the formula $\Diamond_{\downarrow}\varphi$.

Lemma 4.14 (Vertical awareness) *Let Δ_1 and Δ_2 be maximally consistent sets and let φ be a formula. Then the following statements are true:*

(1) $\varphi \in \Delta_1$ *and* $\Delta_1 \prec \Delta_2$ *imply* $\bigcirc_{\uparrow}\varphi \in \Delta_2$.

(2) $\varphi \in \Delta_2$ *and* $\Delta_1 \prec \Delta_2$ *imply* $\Diamond_{\downarrow}\varphi \in \Delta_1$.

(3) $\varphi \in \Delta_1$ *and* $\Delta_1 \prec^+ \Delta_2$ *imply* $\Diamond_{\uparrow}\varphi \in \Delta_2$.

(4) $\varphi \in \Delta_2$ *and* $\Delta_1 \prec^+ \Delta_2$ *imply* $\Diamond_{\downarrow}^{+}\varphi \in \Delta_1$.[9]

[9] We are not actually going to use Part (4) of this lemma later on. It is included here only to show that awareness is *not* the problem when we are dealing with the transitive descendant operator $\Diamond_{\downarrow}^{+}$.

Proof. These claims all follow almost directly from the definition of the canonical child relation $\prec$ (see Definition 4.13).

(1) *Parent awareness.* Let $\varphi \in \Delta_1$ and $\Delta_1 \prec \Delta_2$. Using axiom (B8) together with the former, we can infer $\Box_{\downarrow}\bigcirc^{\uparrow}\varphi \in \Delta_1$. By definition of $\prec$ we then obtain $\bigcirc^{\uparrow}\varphi \in \Delta_2$ as required.

(2) *Child awareness.* Let $\varphi \in \Delta_2$ and $\Delta_1 \prec \Delta_2$. As Δ_2 is maximally consistent, the former is equivalent to $\neg\varphi \notin \Delta_2$. Now we can apply the contrapositive of the formulation used to define $\prec$ and infer that $\Box_{\downarrow}\neg\varphi \notin \Delta_1$ must be the case. Because Δ_1 is a maximally consistent set as well, we get $\neg \Box_{\downarrow}\neg\varphi \in \Delta_1$. By axiom (D4), this is equivalent to $\Diamond_{\downarrow}\varphi \in \Delta_1$, the desired result.

(3) *Ancestor awareness.* Let $\varphi \in \Delta_1$ and $\Delta_1 \prec^+ \Delta_2$, i.e there exists an $n \in \mathbb{N}$ such that $\Delta_1 \prec^n \Delta_2$. Hence, by case (1) of this lemma, we have $(\bigcirc^{\uparrow})^n\varphi \in \Delta_2$. We can now use the derived theorem $\Diamond^{\uparrow}A \leftrightarrow (\bigcirc^{\uparrow}A \vee \cdots \vee (\bigcirc^{\uparrow})^n A \vee (\bigcirc^{\uparrow})^n\Diamond^{\uparrow}A)$ from Lemma 4.9 to infer that we must also have $\Diamond^{\uparrow}\varphi \in \Delta_2$.

(4) *Descendent awareness.* Let $\varphi \in \Delta_2$ and $\Delta_1 \prec^+ \Delta_2$. Hence, there exists an $n \in \mathbb{N}$ such that $\Delta_1 \prec^n \Delta_2$ and we can use case (2) n times to derive $\Diamond_{\downarrow}^n\varphi \in \Delta_1$. The latter together with theorem $\Diamond_{\downarrow}^+A \leftrightarrow (\Diamond_{\downarrow}A \vee \cdots \vee \Diamond_{\downarrow}^n A \vee \Diamond_{\downarrow}^n\Diamond_{\downarrow}^+A)$ from Lemma 4.9 yields the desired $\Diamond_{\downarrow}^+\varphi \in \Delta_1$.

This completes our proof of Lemma 4.14. $\square$

Levels. Next we are going to formalise the notion of a set of formulas having a certain *level*. Recall that our goal is to show that (some) maximally consistent sets of formulas correspond to nodes in a (loosely ordered) tree. Every node in an ordered tree model satisfies a particular type of formula which reflects the distance of that node from the root. The formula $\Box^{\uparrow}\bot$ for instance, is only true at the root itself. The formula $\bigcirc^{\uparrow}\Box^{\uparrow}\bot$ again will only be satisfied at nodes on the first level beneath the root. Generally speaking, the formula $(\bigcirc^{\uparrow})^n\Box^{\uparrow}\bot$ for $n \in \mathbb{N}_0$ is true at t iff t is a node of level n in the tree, that is, iff the distance from t to the root is n.

The following definition 'blindly' applies this intuition to the case of sets of formulas. Showing that, in fact, this definition makes sense and that maximally consistent sets 'with level n' will end up being nodes of level n is the objective of the sequence of lemmas following the definition.

Definition 4.15 (Levels) *Let Δ be a maximally consistent set and let $n \in \mathbb{N}_0$. We say that Δ has level n iff $(\bigcirc^{\uparrow})^n\Box^{\uparrow}\bot \in \Delta$.*

If we simply wish to express that there exists some number $n \in \mathbb{N}_0$ such that $(\bigcirc^{\uparrow})^n\Box^{\uparrow}\bot \in \Delta$ holds, we shall say that the set Δ has *a* level. We say that a *formula* has got a level iff it is a conjunction of the form $\varphi \wedge (\bigcirc^{\uparrow})^n\Box^{\uparrow}\bot$ (with $n \in \mathbb{N}_0$). The following lemma shows that levels are *unique*: a consistent set cannot have more than one level.

Lemma 4.16 (Uniqueness of levels) *Let Δ be a maximally consistent set and let $n_1, n_2 \in \mathbb{N}_0$. Then $\obar^{n_1}\boxdot\bot \in \Delta$ and $\obar^{n_2}\boxdot\bot \in \Delta$ imply $n_1 = n_2$.*

Proof. Let $\obar^{n_1}\boxdot\bot \in \Delta$ and $\obar^{n_2}\boxdot\bot \in \Delta$. For any natural numbers n_1 and n_2 we must have either $n_1 < n_2$ or $n_2 < n_1$ or $n_1 = n_2$. We are first going to exclude the case of $n_1 < n_2$. If $n_1 < n_2$ then there is an $m \in \mathbb{N}$ such that $n_2 = n_1 + m$, i.e. we have $\obar^{n_1}\obar^{m}\boxdot\bot \in \Delta$. Using the distribution law for conjunction with respect to the iterated modality $\obar^{n_1}$ (see Lemma 4.7) we obtain $\obar^{n_1}(\boxdot\bot \wedge \obar^{m}\boxdot\bot) \in \Delta$. But $\obar^{n_1}(\boxdot\bot \wedge \obar^{m}\boxdot\bot)$ can be shown to be an inconsistent formula as follows.

In Lemma 4.8 we have provided a derivation for formulas of the form $\neg(\boxdot\bot \wedge \obar A)$. Hence, $\neg(\boxdot\bot \wedge \obar^{m}\boxdot\bot)$ must be a theorem, as we have $m \geq 1$. If we apply the generalisation rule with respect to the dual of the next-operator $\obar$ for n_1 times to this theorem we obtain $(\neg\obar\neg)^{n_1}\neg(\boxdot\bot \wedge \obar^{m}\boxdot\bot)$. So $\neg\obar^{n_1}(\boxdot\bot \wedge \obar^{m}\boxdot\bot)$ must be a theorem as well.

Hence, $\obar^{n_1}(\boxdot\bot \wedge \obar^{m}\boxdot\bot)$ is indeed inconsistent and $\obar^{n_1}(\boxdot\bot \wedge \obar^{m}\boxdot\bot) \in \Delta$ contradicts our premise of Δ being a maximally consistent set. Therefore, it is not possible that $n_1 < n_2$ holds. By an analogous argument, we can also exclude the case of $n_2 < n_1$. Hence, $n_1 = n_2$ must indeed be true as claimed. $\square$

Levels in the canonical model. So far our discussion of levels has applied to maximally consistent sets of formulas in general. Now we are going to relate the notion of level of a maximally consistent set with the canonical child relation $\prec$ declared over these sets. The following lemma establishes how the level of one maximally consistent set influences the levels of those it is related to via $\prec$ and vice versa.

Lemma 4.17 (Levels of accessible sets) *Let Δ_1 and Δ_2 be maximally consistent sets with $\Delta_1 \prec \Delta_2$. Then the following statements are true:*

(1) $\obar^{n}\boxdot\bot \in \Delta_1$ *implies* $\obar^{n+1}\boxdot\bot \in \Delta_2$ *for all* $n \in \mathbb{N}_0$.

(2) $\obar^{n}\boxdot\bot \in \Delta_2$ *implies* $\obar^{n-1}\boxdot\bot \in \Delta_1$ *for all* $n \in \mathbb{N}$.

Proof. Let Δ_1 and Δ_2 be maximally consistent sets with $\Delta_1 \prec \Delta_2$. We prove the two statements in turn.

(1) We are going to use axiom (B8): $A \rightarrow \square_{\bullet}\obar A$. Suppose we have $\obar^{n}\boxdot\bot \in \Delta_1$ for some $n \in \mathbb{N}_0$. Using modus ponens and (B8) we can infer $\square_{\bullet}\obar^{n+1}\boxdot\bot \in \Delta_1$. Now by definition of $\prec$, we obtain $\obar^{n+1}\boxdot\bot \in \Delta_2$ as claimed.

(2) Suppose $\obar^{n}\boxdot\bot \in \Delta_2$ for some $n \in \mathbb{N}$, that is, we have $\neg\obar^{n}\boxdot\bot \notin \Delta_2$. Then by definition of $\prec$, we must also have $\square_{\bullet}\neg\obar^{n}\boxdot\bot \notin \Delta_1$. Using (D4) this can be seen to be equivalent to $\Diamond_{\bullet}\obar\obar^{n-1}\boxdot\bot \in \Delta_1$. Application of modus ponens with the derived theorem $\Diamond_{\bullet}\obar A \rightarrow A$ (Lemma 4.8) now yields the desired $\obar^{n-1}\boxdot\bot \in \Delta_1$.

This completes our proof of Lemma 4.17. $\square$

Just as a node in a tree with level 0 (i.e. the root of that tree) cannot have any nodes above it, we can show that a maximally consistent set containing the minimal level formula $\Box^{\uparrow}\bot$ must be the first set with respect to the relation $\prec$, that is, there cannot be any other sets before it.

Lemma 4.18 (First level) *Let Δ_r be a maximally consistent set with $\Box^{\uparrow}\bot \in \Delta_r$. Then there exists no maximally consistent set Δ with $\Delta \prec \Delta_r$.*

Proof. Assume $\Box^{\uparrow}\bot \in \Delta_r$ and assume there exists a maximally consistent set Δ with $\Delta \prec \Delta_r$. We are going to derive a contradiction. Our assumption $\Box^{\uparrow}\bot \in \Delta_r$ is equivalent to $\neg \Box^{\uparrow}\bot \notin \Delta_r$. Then $\Delta \prec \Delta_r$ yields $\Box_{\downarrow}\neg \Box^{\uparrow}\bot \notin \Delta$ by definition of $\prec$. Using axiom (D4), this can be seen to imply $\Diamond_{\downarrow}\Box^{\uparrow}\bot \in \Delta$. Now we can use the derived theorem $\Diamond_{\downarrow}\Box^{\uparrow}A \rightarrow A$ (Lemma 4.8) to infer $\bot \in \Delta$, which contradicts our assumption of Δ being a consistent set. $\Box$

Vertical existence lemmas. Let Δ_1 and Δ_2 be two maximally consistent sets with $\Delta_1 \prec \Delta_2$. In Lemma 4.14 we have seen that for every formula $\varphi \in \Delta_2$ there must be the corresponding $\Diamond_{\downarrow}\varphi \in \Delta_1$ and analogously for the other three existential operators pertaining to the child relation in a tree. These results followed almost immediately from the definition of $\prec$. Our next objective is to show that the 'converse' holds as well: whenever we have $\Diamond_{\downarrow}\varphi \in \Delta_1$ then there *exists* a maximally consistent set of formulas Δ_2 containing the witness formula φ. These kinds of results are often called *Existence Lemmas*. In the sequel we are going to prove a *Child Existence Lemma*, a *Parent Existence Lemma*, and an *Ancestor Existence Lemma* for our vertical modalities $\Diamond_{\downarrow}$, $\bigcirc^{\uparrow}$ and $\Diamond^{\uparrow}$, respectively.

Following this, we are briefly going to discuss why we cannot prove a *Descendant Existence Lemma* for the transitive modality $\Diamond_{\downarrow}^{+}$ in a similar manner. (Recall that we have mentioned earlier that the completeness result to be formulated at the end this section only applies to the fragment of **OTL** excluding $\Diamond_{\downarrow}^{+}$ and its dual $\Box_{\downarrow}^{+}$.)

Lemma 4.19 (Child existence) *For any maximally consistent set Δ_1, if $\Diamond_{\downarrow}\varphi \in \Delta_1$ then there exists a maximally consistent set Δ_2 such that $\Delta_1 \prec \Delta_2$ and $\varphi \in \Delta_2$.*

Proof. Let $\Diamond_{\downarrow}\varphi \in \Delta_1$. We first define a set Δ_2' that contains all those formulas that are bound to be part of any maximally consistent set meeting the requirements of the lemma:

$$\Delta_2' \quad = \quad \{\varphi\} \cup \{\psi \mid \Box_{\downarrow}\psi \in \Delta_1\}$$

The formula φ must certainly be contained in any such set, because our aim is to find a set Δ_2 with $\varphi \in \Delta_2$. Furthermore, whenever we have $\Box_{\downarrow}\psi \in \Delta_1$, our set Δ_2 will have to contain ψ if we wish to get $\Delta_1 \prec \Delta_2$.

The set Δ_2' can be shown to be consistent as follows. For the sake of contradiction, let us assume that Δ_2' is not consistent, i.e. there exist formulas $\psi_1, \ldots, \psi_n$ such that $\Box_{\downarrow}\psi_i \in \Delta_1$ for all $i \in \{1, \ldots, n\}$ and $\vdash \neg(\psi_1 \wedge \cdots \wedge \psi_n \wedge \varphi)$. The latter is equivalent to

$\vdash (\psi_1 \wedge \cdots \wedge \psi_n) \rightarrow \neg\varphi$. Generalisation yields $\vdash [\downarrow]((\psi_1 \wedge \cdots \wedge \psi_n) \rightarrow \neg\varphi)$ and using (K4) we obtain $\vdash [\downarrow](\psi_1 \wedge \cdots \wedge \psi_n) \rightarrow [\downarrow]\neg\varphi$. By Lemma 4.7, we also have $\vdash ([\downarrow]\psi_1 \wedge \cdots \wedge [\downarrow]\psi_n) \rightarrow [\downarrow](\psi_1 \wedge \cdots \wedge \psi_n)$, that is, $\vdash ([\downarrow]\psi_1 \wedge \cdots \wedge [\downarrow]\psi_n) \rightarrow [\downarrow]\neg\varphi$ must hold. Hence, $[\downarrow]\neg\varphi \in \Delta_1$, which using (D4) can be seen to be equivalent to $\langle\downarrow\rangle\varphi \notin \Delta_1$. But this contradicts our original assumption $\langle\downarrow\rangle\varphi \in \Delta_1$, i.e. Δ_2' must indeed be a consistent set of formulas.

Given the consistency of Δ_2', we can extend Δ_2' to a maximally consistent set Δ_2, that is, $\Delta_2' \subseteq \Delta_2$. (This is an application of Lemma 4.12.) This set fulfils our requirements by construction: we have both $\varphi \in \Delta_1$ and $\Delta_1 \prec \Delta_2$. □

Lemma 4.20 (Parent existence) *For any maximally consistent set Δ_2, if $(\uparrow)\varphi \in \Delta_2$ then there exists a maximally consistent set Δ_1 such that $\Delta_1 \prec \Delta_2$ and $\varphi \in \Delta_1$.*

Proof. Let $(\uparrow)\varphi \in \Delta_2$. We are going to follow a similar strategy as for the proof of the Child Existence Lemma and start with the definition of a suitable base set:

$$\Delta_1' \quad = \quad \{\psi \mid (\uparrow)\psi \in \Delta_2\}$$

First observe that this definition implies, in particular, that φ will be an element of Δ_1'. We are going to show that Δ_1' is consistent. Assume this is not the case, that is, assume there are formulas $\psi_1, \ldots, \psi_n$ such that $(\uparrow)\psi_1, \ldots, (\uparrow)\psi_n \in \Delta_2$ and $\vdash \neg(\psi_1 \wedge \cdots \wedge \psi_n)$. The former entails $(\uparrow)(\psi_1 \wedge \cdots \wedge \psi_n) \in \Delta_2$ (by Lemma 4.7). On the other hand, generalisation of $\vdash \neg(\psi_1 \wedge \cdots \wedge \psi_n)$ with respect to the dual of the parent operator $(\uparrow)$ yields $\vdash \neg (\uparrow)\neg\neg(\psi_1 \wedge \cdots \wedge \psi_n)$, which is equivalent to $\vdash \neg (\uparrow)(\psi_1 \wedge \cdots \wedge \psi_n)$, i.e. we obtain a contradiction, because the consistent set Δ_2 cannot contain the complement of a theorem.

Now let Δ_1 be a maximally consistent set with $\Delta_1' \subseteq \Delta_1$. We have already seen that we have $\varphi \in \Delta_1$ by construction. It remains to be shown that $\Delta_1 \prec \Delta_2$ holds as well. For the sake of contradiction, assume $\Delta_1 \not\prec \Delta_2$, that is, there exists a formula χ such that $[\downarrow]\chi \in \Delta_1$ but $\chi \notin \Delta_2$. As Δ_2 is maximally consistent, the latter implies $\neg\chi \in \Delta_2$. Furthermore, observe that we certainly have $(\uparrow)\top \in \Delta_2$ (by Lemma 4.14). Axiom (B7) now allows us to deduce $(\uparrow)\langle\downarrow\rangle\neg\chi \in \Delta_2$. By construction of Δ_1', we get $\langle\downarrow\rangle\neg\chi \in \Delta_1'$ and therefore also $\langle\downarrow\rangle\neg\chi \in \Delta_1$. But using (D4), this can be seen to imply $[\downarrow]\chi \notin \Delta_1$, which contradicts $[\downarrow]\chi \in \Delta_1$. Hence, the set Δ_1 really has all the required properties. □

There is an important difference between the following lemma on the one hand and both the Child Existence Lemma and the Parent Existence Lemma on the other hand. To be able to prove our following Ancestor Existence Lemma, we will have to make the assumption that the set containing the existential formula has got a level. (We have made no such assumption for either one of the previous two lemmas.) The level of the maximally consistent set in question gives us an upper limit for how far away the witness formula could possibly be (it needs to be somewhere between the current node and the root). This allows us to explicitly unfold the ancestor operator (pertaining to the transitive closure of the inverse of the canonical child relation) and replace it with a disjunction of iterated parent modalities.

Lemma 4.21 (Ancestor existence) *For any maximally consistent set Δ_2 that has a level, if $\overset{\uparrow}{\Diamond}\varphi \in \Delta_2$ then there exists a maximally consistent set Δ_1 such that $\Delta_1 \prec^+ \Delta_2$ and $\varphi \in \Delta_1$.*

Proof. Let Δ_2 be a maximally consistent set of level n, i.e. we have $\overset{\uparrow}{\bigcirc}^n\overset{\downarrow}{\Box}\bot \in \Delta_2$ with $n \in \mathbb{N}_0$, and let $\overset{\uparrow}{\Diamond}\varphi \in \Delta_2$. In fact, we may assume that $n \geq 1$, because we cannot have both $\overset{\downarrow}{\Box}\bot \in \Delta_2$ and $\overset{\uparrow}{\Diamond}\varphi \in \Delta_2$. So we can make use of the derived theorem $\overset{\uparrow}{\Diamond}A \leftrightarrow (\overset{\uparrow}{\bigcirc}A \vee \cdots \vee \overset{\uparrow}{\bigcirc}^n A \vee \overset{\uparrow}{\bigcirc}^n\overset{\uparrow}{\Diamond}A)$ proved in Lemma 4.9. By that theorem, we must have either $\overset{\uparrow}{\bigcirc}\varphi \vee \cdots \vee \overset{\uparrow}{\bigcirc}^n\varphi \in \Delta_2$ or $\overset{\uparrow}{\bigcirc}^n\overset{\uparrow}{\Diamond}\varphi \in \Delta_2$. We are going to show that the latter is not possible. Assume we have $\overset{\uparrow}{\bigcirc}^n\overset{\uparrow}{\Diamond}\varphi \in \Delta_2$. As we also have $\overset{\uparrow}{\bigcirc}^n\overset{\downarrow}{\Box}\bot \in \Delta_2$, we get $\overset{\uparrow}{\bigcirc}^n(\overset{\uparrow}{\Diamond}\varphi \wedge \overset{\downarrow}{\Box}\bot) \in \Delta_2$, but the negation of that formula can easily be seen to be a theorem. Clearly, $\overset{\uparrow}{\Diamond}\varphi \wedge \overset{\downarrow}{\Box}\bot$ is inconsistent, i.e. $\neg(\overset{\uparrow}{\Diamond}\varphi \wedge \overset{\downarrow}{\Box}\bot)$ must be a theorem. Now n consecutive applications of the generalisation rule with respect to the dual of $\overset{\uparrow}{\bigcirc}$ yield the theorem $\neg\,\overset{\uparrow}{\bigcirc}^n(\overset{\uparrow}{\Diamond}\varphi \wedge \overset{\downarrow}{\Box}\bot)$.

Hence, we must have $\overset{\uparrow}{\bigcirc}\varphi \vee \cdots \vee \overset{\uparrow}{\bigcirc}^n\varphi \in \Delta_2$, that is, there exists an $m \in \{1, \ldots, n\}$ such that $\overset{\uparrow}{\bigcirc}^m\varphi \in \Delta_2$. Now by the Parent Existence Lemma, there exist maximally consistent sets $\Delta'_1, \ldots, \Delta'_m$ with $\Delta'_m \prec \cdots \prec \Delta'_1 \prec \Delta_2$ and $\overset{\uparrow}{\bigcirc}^{m-i}\varphi \in \Delta'_i$ for all $i \in \{1, \ldots, m\}$. The latter means, in particular, that we have $\varphi \in \Delta'_m$. If we define $\Delta_1 = \Delta'_m$ then this set meets our requirements: we have $\Delta_1 \prec^m \Delta_2$, i.e. also $\Delta_1 \prec^+ \Delta_2$, as well as $\varphi \in \Delta_1$. □

To be able to prove completeness for the full language of **OTL**, at this stage, we would also have to prove a *Descendant Existence Lemma* along the lines of the previous three lemmas. However, this seems not to be possible, at least not with the approach we have taken here. The heart of the problem lies in the fact that the descendant operator $\overset{\downarrow}{\Diamond}^+$ is supposed to end up as an existential modality over the transitive closure of the canonical child relation $\prec$. In the case of the ancestor operator $\overset{\uparrow}{\Diamond}$ we managed to overcome this problem and showed that $\overset{\uparrow}{\Diamond}$ relates to the transitive closure of the inverse of $\prec$ as intended. This has been possible, because for a formula of the form $\overset{\uparrow}{\Diamond}\varphi$ there is always a last possible node where the witness φ could possibly hold, namely the root. As long as maximally consistent sets are equipped with a level this insight allows us to unfold the transitive ancestor modality into a disjunction of iterated parent operators using the iterated version of one of the mixing axioms established in Lemma 4.9. In the case of a formula of the form $\overset{\downarrow}{\Diamond}^+\varphi$, on the other hand, we have no means of 'cutting off' the corresponding disjunction of iterated non-transitive operators. In other words, we cannot assume any maximal depth in the tree where the witness formula has to be satisfied at the latest. We are going to return to the discussion of this problem in Section 4.4.

Generated canonical trees. As indicated already in the introduction to this section, in our completeness proof we are not going to refer to the full canonical model over the whole set of maximally consistent sets of formulas, but we will restrict the set of worlds to maximally consistent sets with a level. In addition to that, we are also going to discard any sets that are not connected to some first set via an iteration of the canonical child relation and its inverse. This first set is said to *generate* the model. The following definition makes precise the notion of a set of maximally consistent sets generated by a given set Δ^*. In our Completeness Theorem at the end of this section, this set Δ^* will be the set containing the consistent formula in question and the corresponding world in the (generated) canonical will be the world at which it will be shown to be satisfied.

Definition 4.22 (Generated set of maximally consistent sets) *Let Δ^* be a maximally consistent set. We define T_{Δ^*} as the set of maximally consistent sets Δ for which there exist maximally consistent sets $\Delta_1, \ldots, \Delta_n$ with $\Delta_1 = \Delta^*$ and $\Delta_n = \Delta$ such that either $\Delta_i \prec \Delta_{i+1}$ or $\Delta_{i+1} \prec \Delta_i$ for all natural numbers $i < n$.*

In other words, T_{Δ^*} is the set of worlds in the *subframe generated by* Δ^* with respect to the accessibility relation that is the union of $\prec$ and its inverse [13].

In Lemma 4.24 further below we are going to establish the first important result about the structure of the frame underlying our canonical model. The claim is that T_{Δ^*}, the set of maximally consistent sets generated by some maximally consistent set Δ^* (which will be required to have a level) and the canonical accessibility relation $\prec$ from Definition 4.13 form a tree according to Definition 3.2. Amongst other things, this involves proving that the canonical child relation $\prec$ is conversely functional, that is, any given maximally consistent set can have at most a single predecessor with respect to $\prec$. As this result does not depend on a restriction to maximally consistent sets with a level, we state it here as a lemma in its own right.

Lemma 4.23 (Converse functionality of canonical child relation) *Let Δ, Δ_1 and Δ_2 be maximally consistent sets. Then $\Delta_1 \prec \Delta$ and $\Delta_2 \prec \Delta$ imply $\Delta_1 = \Delta_2$.*

Proof. We are going to use the inverse axiom (B8): $A \rightarrow \boxdot \odot A$ and the functionality axiom (F3): $\odot A \leftrightarrow (\neg \odot \neg A \wedge \odot \top)$. Let Δ, Δ_1 and Δ_2 be maximally consistent sets with $\Delta_1 \prec \Delta$ and $\Delta_2 \prec \Delta$. For the sake of contradiction, assume that $\Delta_1 \neq \Delta_2$, i.e. there is a formula φ such that $\varphi \in \Delta_1$ and $\neg\varphi \in \Delta_2$. Application of modus ponens in combination with (B8) yields $\boxdot \odot \varphi \in \Delta_1$ and $\boxdot \odot \neg\varphi \in \Delta_2$. By definition of the relation $\prec$, we get both $\odot \varphi \in \Delta$ and $\odot \neg\varphi \in \Delta$. From the former and the functionality axiom (F3) we can derive $\neg \odot \neg\varphi \in \Delta$, i.e. we obtain a contradiction. Therefore, Δ_1 and Δ_2 must indeed be identical sets of formulas. $\square$

We are now ready to show that the order imposed by the canonical child relation over a set of maximally consistent sets generated by a given maximally consistent set with a level must be a tree.

Lemma 4.24 (Generated canonical tree) *Let Δ^* be a maximally consistent set with a level. Then $(T_{\Delta^*}, \prec)$ is a tree.*

Proof. We need to show that $\prec$ is irreflexive and conversely functional over T_{Δ^*}, and that $(T_{\Delta^*}, \prec)$ is rooted.

(1) *Irreflexivity.* Δ^* is required to have a level. Hence, by Lemma 4.17 and the definition of T_{Δ^*}, every set in T_{Δ^*} must have a level, too. Irreflexivity of $\prec$ over T_{Δ^*} can now be proved by contradiction. Assume there is a reflexive point, that is, assume there exists a maximally consistent set $\Delta \in T_{\Delta^*}$ with $\Delta \prec \Delta$. Because Δ must have a level, there exists an $n \in \mathbb{N}_0$ such that $\langle\uparrow\rangle^n [\uparrow]\bot \in \Delta$. Furthermore, because we have $\Delta \prec \Delta$, Lemma 4.17 applies and we can infer $\langle\uparrow\rangle^{n+1}[\uparrow]\bot \in \Delta$ as well. Now Lemma 4.16 on the uniqueness of levels yields $n = n+1$, which is an obvious contradiction.

(2) *Converse functionality.* In Lemma 4.23 the relation $\prec$ has been shown to be conversely functional over *all* maximally consistent sets, so it is certainly conversely functional over T_{Δ^*}.

(3) *Rootedness.* The set Δ^* is required to have a level, so there exists an $m \in \mathbb{N}$ such that $\langle\uparrow\rangle^m [\uparrow]\bot \in \Delta^*$. Applying the Parent Existence Lemma m times proves that there must be a maximally consistent set Δ_r with $\Delta_r \prec^m \Delta^*$ and $[\uparrow]\bot \in \Delta_r$. This set must be an element of T_{Δ^*}. We are going to show that, in fact, Δ_r is the (unique) root of $(T_{\Delta^*}, \prec)$.

Recall that, according to Definition 4.22, for every $\Delta \in T_{\Delta^*}$ there exist $\Delta_1, \ldots, \Delta_n$ with $\Delta_1 = \Delta^*$ and $\Delta_n = \Delta$ such that either $\Delta_i \prec \Delta_{i+1}$ or $\Delta_{i+1} \prec \Delta_i$ for all $i < n$. This n provides us with a notion of 'distance' of a set Δ from the generating set Δ^*. We are going to prove, by induction over n, that for every $\Delta \in T_{\Delta^*}$ we have $\Delta_r \prec^* \Delta$. For the base case ($n = 1$), Δ can only be Δ^*. Clearly, $\Delta_r \prec^* \Delta^*$ does hold, because we have $\Delta_r \prec^m \Delta^*$. For the induction step, suppose the claim has been shown to hold for all cases up to some natural number n. We have to show that $\Delta_r \prec^* \Delta$ also holds for sets Δ whose distance from Δ^* is $n+1$. For such a Δ there must be a Δ_n with either $\Delta_n \prec \Delta$ or $\Delta \prec \Delta_n$ for which the induction hypothesis holds, i.e. we have $\Delta_r \prec^* \Delta_n$. In the case where $\Delta_n \prec \Delta$ holds, we immediately get $\Delta_r \prec^* \Delta$ as well and are done. So suppose $\Delta \prec \Delta_n$. As we have $\Delta_r \prec^* \Delta_n$, we have either $\Delta_r \prec^+ \Delta_n$ or $\Delta_r = \Delta_n$. But the latter would entail that $\Delta \prec \Delta_r$, which would contradict Lemma 4.18. Hence, $\Delta_r \prec^+ \Delta_n$ will hold, which means we immediately get $\Delta_r \prec^* \Delta$ and are done. This concludes our inductive proof for $T_{\Delta^*} \subseteq \{\Delta \mid \Delta_r \prec^* \Delta\}$. The other direction, namely $\{\Delta \mid \Delta_r \prec^* \Delta\} \subseteq T_{\Delta^*}$, is an immediate consequence of the definition of T_{Δ^*} and the fact that $\Delta_r \in T_{\Delta^*}$. Hence, we have $T_{\Delta^*} = \{\Delta \mid \Delta_r \prec^* \Delta\}$, which means that Δ_r is a root of $(T_{\Delta^*}, \prec)$.

We still have to show that Δ_r is a *unique* root element. As we have $[\uparrow]\bot \in \Delta_r$, by Lemma 4.18, there can be no maximally consistent set Δ with $\Delta \prec \Delta_r$. So there is certainly no maximally consistent set in T_{Δ^*} different from Δ_r from which Δ_r itself is accessible via the transitive closure of $\prec$. But this would be a necessary requirement for any root node of $(T_{\Delta^*}, \prec)$, so the root Δ_r is indeed unique.

This concludes our proof of Lemma 4.24 as we have verified all conditions set out in Definition 3.2 for an arbitrary relation to be a tree. □

We note that we have not made any reference to the proposed rootedness axiom (R1), which involves the descendant operator $\Diamond^+$, in this proof. In fact, we are only going to use (R1) in the next section where we are going to analyse the problems related to the axiomatisation of the descendant operator in some more detail.

Horizontal accessibility relations. We now turn to the horizontal dimension of our canonical model. In the following definition we introduce two horizontal accessibility relation over maximally consistent sets of formulas in a similar manner as we have defined the canonical child relation $\prec$. Or aim later on will be to show that these two new relations correspond to the pseudo sibling relation S and the pseudo neighbour relation N of a loosely ordered tree, respectively.

Definition 4.25 (Canonical sibling and neighbour relations) *Let Δ_1 and Δ_2 be maximally consistent sets. We define the canonical sibling relation $<$ and the canonical neighbour relation $<_\circ$ over maximally consistent sets as follows:*

$$\begin{array}{lll} \Delta_1 < \Delta_2 & \textit{iff} & \boxminus\varphi \in \Delta_1 \textit{ implies } \varphi \in \Delta_2 \textit{ for all formulas } \varphi \\ \Delta_1 <_\circ \Delta_2 & \textit{iff} & \ominus\top \in \Delta_1 \textit{ and } \ominus\varphi \in \Delta_1 \textit{ implies } \varphi \in \Delta_2 \textit{ for all formulas } \varphi \end{array}$$

Transitive closures of $<$ and $<_\circ$ will be denoted in accordance with Definition A.3 from Appendix A. The relation $<_\circ^+$, for instance, is the non-reflexive transitive closure of $<_\circ$. The following lemma is the horizontal counterpart to Lemma 4.14. It states that maximally consistent sets are 'aware' of their predecessors and successors with respect to the canonical sibling and neighbour relations.

Lemma 4.26 (Horizontal awareness) *Let Δ_1 and Δ_2 be maximally consistent sets and let φ be a formula. Then the following statements are true:*

(1) $\varphi \in \Delta_1$ and $\Delta_1 < \Delta_2$ imply $\Diamond\varphi \in \Delta_2$.

(2) $\varphi \in \Delta_2$ and $\Delta_1 < \Delta_2$ imply $\Diamond\varphi \in \Delta_1$.

(3) $\varphi \in \Delta_1$ and $\Delta_1 <_\circ \Delta_2$ imply $\ominus\varphi \in \Delta_2$.

(4) $\varphi \in \Delta_2$ and $\Delta_1 <_\circ \Delta_2$ imply $\ominus\varphi \in \Delta_1$.

Proof. The four claims follow almost immediately from the definitions of $<$ and $<_\circ$, respectively, together with some of the basic inverse, duality, and functionality axioms.

(1) *Left sibling awareness.* Let $\varphi \in \Delta_1$ and $\Delta_1 < \Delta_2$. Using the inverse axiom (B2), we can infer $\boxminus\Diamond\varphi \in \Delta_1$ and hence, by definition of $<$, we have $\Diamond\varphi \in \Delta_2$.

(2) *Right sibling awareness.* Let $\varphi \in \Delta_2$ and $\Delta_1 < \Delta_2$. For the sake of contradiction, assume $\overrightarrow{\Diamond}\varphi \notin \Delta_1$, which by (D2) is equivalent to $\overrightarrow{\Box}\neg\varphi \in \Delta_1$. By definition of $<$, the latter entails $\neg\varphi \in \Delta_2$, the contradiction we have been looking for. Thus $\overrightarrow{\Diamond}\varphi \in \Delta_1$ as claimed.

(3) *Left neighbour awareness.* Let $\varphi \in \Delta_1$ and $\Delta_1 <_\circ \Delta_2$. The latter implies in particular $\overrightarrow{\ominus}\top \in \Delta_1$. Hence, axiom (B6) is applicable and we can infer $\overrightarrow{\ominus}\overleftarrow{\ominus}\varphi \in \Delta_1$. Then $\overleftarrow{\ominus}\varphi \in \Delta_2$ follows by definition of $<_\circ$.

(4) *Right neighbour awareness.* Let $\varphi \in \Delta_2$ and $\Delta_1 <_\circ \Delta_2$. For the sake of contradiction, assume $\overrightarrow{\ominus}\varphi \in \Delta_1$ is not the case. Then $\neg \overrightarrow{\ominus}\neg\neg\varphi \in \Delta_1$ must be true. Because of $\Delta_1 <_\circ \Delta_2$, we also have $\overrightarrow{\ominus}\top \in \Delta_1$ and thus may use the functionality axiom (F2) in order to obtain $\overrightarrow{\ominus}\neg\varphi \in \Delta_1$. By definition of $<_\circ$, this implies $\neg\varphi \in \Delta_2$, which contradicts our original assumption $\varphi \in \Delta_2$. Hence, $\overrightarrow{\ominus}\varphi \in \Delta_1$ must be the case.

This completes our proof of Lemma 4.26. □

Horizontal existence lemmas. In analogy to our vertical Existence Lemmas proved earlier, we now have to show that also for the existential operators pertaining to the canonical sibling and the canonical neighbour relation, there will always be a related set containing an appropriate witness formula.

Lemma 4.27 (Right sibling existence) *For any maximally consistent set Δ_1, if $\overrightarrow{\Diamond}\varphi \in \Delta_1$ then there exists a maximally consistent set Δ_2 such that $\Delta_1 < \Delta_2$ and $\varphi \in \Delta_2$.*

Proof. Let $\overrightarrow{\Diamond}\varphi \in \Delta_1$. We start by constructing a set Δ_2' containing all those formulas that are 'forced' to belong to any $<$-successor of Δ_1:

$$\Delta_2' \quad = \quad \{\varphi\} \cup \{\psi \mid \overrightarrow{\Box}\psi \in \Delta_1\}$$

The consistency of Δ_2' can be proved by contradiction. If it is not consistent, then there must be formulas $\psi_1, \ldots, \psi_n$ such that $\overrightarrow{\Box}\psi_i \in \Delta_1$ for all $i \in \{1, \ldots, n\}$ and $\vdash \neg(\psi_1 \wedge \cdots \wedge \psi_n \wedge \varphi)$. The former implies $\overrightarrow{\Box}(\psi_1 \wedge \cdots \wedge \psi_n) \in \Delta_1$ (by Lemma 4.7). The latter is equivalent to $\vdash (\psi_1 \wedge \cdots \wedge \psi_n) \rightarrow \neg\varphi$. If we apply the generalisation rule, we get $\vdash \overrightarrow{\Box}((\psi_1 \wedge \cdots \wedge \psi_n) \rightarrow \neg\varphi)$ and then, using axiom (K2), $\vdash \overrightarrow{\Box}(\psi_1 \wedge \cdots \wedge \psi_n) \rightarrow \overrightarrow{\Box}\neg\varphi$. Hence, $\overrightarrow{\Box}\neg\varphi \in \Delta_1$. But an application of axiom (D2) shows that this is equivalent to $\overrightarrow{\Diamond}\varphi \notin \Delta_1$, that is, we have derived a contradiction and Δ_2' must be a consistent set of formulas.

Therefore, by Lemma 4.12, we can extend Δ_2' to a maximally consistent set Δ_2. We have $\varphi \in \Delta_2$ by construction and also $\Delta_1 < \Delta_2$, because for every formula of the form $\overrightarrow{\Box}\psi$ in Δ_1 its principal subformula ψ will be in Δ_2', and thereby also in Δ_2. □

Lemma 4.28 (Left sibling existence) *For any maximally consistent set Δ_2, if $\overleftarrow{\Diamond}\varphi \in \Delta_2$ then there exists a maximally consistent set Δ_1 such that $\Delta_1 < \Delta_2$ and $\varphi \in \Delta_1$.*

Proof. This can be shown in direct analogy to the proof of Lemma 4.27. □

Lemma 4.29 (Right neighbour existence) *For any maximally consistent set Δ_1, if $\ominus\varphi \in \Delta_1$ then there exists a maximally consistent set Δ_2 such that $\Delta_1 <_\circ \Delta_2$ and $\varphi \in \Delta_2$.*

Proof. Let $\ominus\varphi \in \Delta_1$. As before, our first step is to construct a (consistent) set Δ_2' of formulas that are forced to belong to any set Δ_2 that could fulfil the conditions of the claim, i.e. that could act as a righthand neighbour for the set Δ_1.

$$\Delta_2' \quad = \quad \{\psi \mid \ominus\psi \in \Delta_1\}$$

We prove that Δ_2' is consistent. For Δ_2' to be inconsistent there would have to be formulas $\psi_1, \ldots, \psi_n$ with $\ominus\psi_1, \ldots, \ominus\psi \in \Delta_1$ and $\vdash \neg(\psi_1 \wedge \cdots \wedge \psi_n)$. If we apply the generalisation rule with respect to $\neg\ominus\neg$ to the latter, we obtain $\vdash \neg\ominus\neg\neg(\psi_1 \wedge \cdots \wedge \psi_n)$. On the other hand, as we can distribute the right neighbour operator, the formula $\ominus(\psi_1 \wedge \cdots \wedge \psi_n)$ must belong to Δ_1, which is a contradiction. Hence, Δ_2' must be a consistent set of formulas and can be extended to a maximally consistent set Δ_2.

We have $\varphi \in \Delta_2$ by construction. Furthermore, $\Delta_1 <_\circ \Delta_2$ must hold as well, because we have $\psi \in \Delta_2$ for every $\ominus\psi \in \Delta_1$, and $\ominus\top \in \Delta_1$, which follows from $\ominus\varphi \in \Delta_1$ and axiom (F2). □

Lemma 4.30 (Left neighbour existence) *For any maximally consistent set Δ_2, if $\ominus\varphi \in \Delta_2$ then there exists a maximally consistent set Δ_1 such that $\Delta_1 <_\circ \Delta_2$ and $\varphi \in \Delta_1$.*

Proof. This can be proved in the same way as Lemma 4.29. □

Properties of the canonical frame. In the following sequence of lemmas we shall establish a number of properties of the relations $\prec$, $<$ and $<_\circ$ declared over maximally consistent sets of formulas. As will be made precise later on, (most of) these properties correspond to the properties of loosely ordered trees listed in Definition 3.33. In order to emphasise the correspondences between the semantic and the axiomatic level, at the beginning of each of the proofs, we shall quote the central axiom(s) used to prove the respective property.

In the proof of Lemma 4.42 (on page 134) we are going to summarise these results and show that the frame which our canonical accessibility relations give rise to must be a loosely ordered tree.

Lemma 4.31 (Transitivity of canonical sibling relation) *The relation $<$ declared over maximally consistent sets is transitive.*

Proof. We are going to use axiom (H1): $\boxminus A \rightarrow \boxminus\boxminus A$. Suppose Δ_1, Δ_2 and Δ_3 are maximally consistent sets such that $\Delta_1 < \Delta_2$ and $\Delta_2 < \Delta_3$. We need to show that

$\Delta_1 < \Delta_3$ holds as well, that is, we need to show that $\boxminus\varphi \in \Delta_1$ implies $\varphi \in \Delta_3$ for any given formula φ. So let $\boxminus\varphi \in \Delta_1$. By axiom (H1), this implies $\boxminus\boxminus\varphi \in \Delta_1$. Now, using $\Delta_1 < \Delta_2$, we obtain $\boxminus\varphi \in \Delta_2$, and then, using $\Delta_2 < \Delta_3$, we get the required result $\varphi \in \Delta_3$. □

Lemma 4.32 (Common parents of siblings) *Let Δ_1 and Δ_2 be maximally consistent sets with $\Delta_1 < \Delta_2$. Then there exists a maximally consistent set Δ such that $\Delta \prec \Delta_1$ and $\Delta \prec \Delta_2$.*

Proof. We are going to use the interaction axiom (X1): $\diamondrightarrow A \rightarrow \odot\diamondcdot A$. Let Δ_1 and Δ_2 be maximally consistent sets such that $\Delta_1 < \Delta_2$. According to Lemma 4.26, the set Δ_1 will be 'aware' of its successor, that is, we must have $\diamondrightarrow\top \in \Delta_1$. By axiom (X1), we then also have $\odot\diamondcdot\top \in \Delta_1$. Hence, by the Parent Existence Lemma, there exists a maximally consistent set Δ with $\Delta \prec \Delta_1$ (and $\diamondcdot\top \in \Delta$, but this second conclusion is not of interest here).

To show that $\Delta \prec \Delta_2$ holds as well, we are going to use theorem $\boxdot A \rightarrow \boxdot\boxminus A$, which has been proved correct in Lemma 4.8.[10] Let φ be a formula with $\boxdot\varphi \in \Delta$. Using $\boxdot A \rightarrow \boxdot\boxminus A$ yields $\boxdot\boxminus\varphi \in \Delta$. This implies $\boxminus\varphi \in \Delta_1$, because we have $\Delta \prec \Delta_1$. This in turn implies $\varphi \in \Delta_2$, because we have $\Delta_1 < \Delta_2$. Thus $\Delta \prec \Delta_2$, because we have shown that $\boxdot\varphi \in \Delta$ implies $\varphi \in \Delta_2$ for all formulas φ. □

Lemma 4.33 (Connectedness of children) *Let Δ, Δ_1 and Δ_2 be maximally consistent sets with $\Delta \prec \Delta_1$ and $\Delta \prec \Delta_2$. Then we must have either $\Delta_1 = \Delta_2$ or $\Delta_1 < \Delta_2$ or $\Delta_2 < \Delta_1$.*

Proof. The central axiom for this proof will be the second interaction axiom (X2): $\odot\diamondcdot A \rightarrow (A \vee \diamondleftarrow A \vee \diamondrightarrow A)$. Let $\Delta \prec \Delta_1$ and $\Delta \prec \Delta_2$. For the sake of contradiction, assume that neither $\Delta_1 = \Delta_2$ nor $\Delta_1 < \Delta_2$ nor $\Delta_2 < \Delta_1$ holds. Then there must be formulas φ, χ and ψ such that $\varphi \in \Delta_1$ but $\varphi \notin \Delta_2$ (otherwise $\Delta_1 = \Delta_2$), $\boxminus\chi \in \Delta_1$ but $\chi \notin \Delta_2$ (otherwise $\Delta_1 < \Delta_2$), and $\boxminus\psi \in \Delta_2$ but $\psi \notin \Delta_1$ (otherwise $\Delta_2 < \Delta_1$).

Because Δ_2 is maximally consistent, we also have $\neg\varphi \wedge \neg\chi \wedge \boxminus\psi \in \Delta_2$. We can now apply Lemma 4.14, first to deduce $\diamondcdot(\neg\varphi \wedge \neg\chi \wedge \boxminus\psi) \in \Delta$ (using $\Delta \prec \Delta_2$) and then to obtain $\odot\diamondcdot(\neg\varphi \wedge \neg\chi \wedge \boxminus\psi) \in \Delta_1$ (using $\Delta \prec \Delta_1$). Now axiom (X2) can be used to infer that we must have either (1) $\neg\varphi \wedge \neg\chi \wedge \boxminus\psi \in \Delta_1$ or (2) $\diamondleftarrow(\neg\varphi \wedge \neg\chi \wedge \boxminus\psi) \in \Delta_1$ or (3) $\diamondrightarrow(\neg\varphi \wedge \neg\chi \wedge \boxminus\psi) \in \Delta_1$. But as we shall see, each one of these three options will lead to a contradiction.

(1) $\neg\varphi \wedge \neg\chi \wedge \boxminus\psi \in \Delta_1$ directly contradicts our assumption $\varphi \in \Delta_1$.

(2) If $\diamondleftarrow(\neg\varphi \wedge \neg\chi \wedge \boxminus\psi) \in \Delta_1$ is the case, then by the Left Sibling Lemma there must be a maximally consistent set Δ' such that $\Delta' < \Delta_1$ and $\neg\varphi \wedge \neg\chi \wedge \boxminus\psi \in \Delta'$. This implies in particular $\boxminus\psi \in \Delta'$. Hence, by definition of $<$, we get $\psi \in \Delta_1$, which contradicts our earlier assumption $\psi \notin \Delta_1$.

[10] In fact, the proof given in Lemma 4.8 shows that $\boxdot A \rightarrow \boxdot\boxminus A$ essentially follows from (X1). So (X1) really is the central axiom here.

(3) Now the only remaining option is $\diamondminus(\neg\varphi \wedge \neg\chi \wedge \boxminus\psi) \in \Delta_1$. By the Right Sibling Existence Lemma there exists a maximally consistent set Δ' with $\Delta_1 < \Delta'$ and $\neg\varphi \wedge \neg\chi \wedge \boxminus\psi \in \Delta'$, which implies in particular $\neg\chi \in \Delta'$. But we also have $\boxminus\chi \in \Delta_1$ and therefore $\chi \in \Delta'$, that is, we have again encountered a contradiction.

Hence, our assumption that neither $\Delta_1 = \Delta_2$ nor $\Delta_1 < \Delta_2$ nor $\Delta_2 < \Delta_1$ holds is indeed contradictory and we are done. $\Box$

The following two lemmas do not themselves correspond to properties of loosely ordered trees specifically postulated by Definition 3.33, but are simple consequences of those conditions. We explicitly state them here in order to simplify the presentation of the proofs of some of our later lemmas.

Lemma 4.34 (No branching to the right) *Let Δ, Δ_1 and Δ_2 be maximally consistent sets such that $\Delta < \Delta_1$ and $\Delta < \Delta_2$. Then we have either $\Delta_1 = \Delta_2$ or $\Delta_1 < \Delta_2$ or $\Delta_2 < \Delta_1$.*

Proof. Let Δ, Δ_1 and Δ_2 be maximally consistent sets such that $\Delta < \Delta_1$ and $\Delta < \Delta_2$. Because of the former, by Lemma 4.32, there must be a maximally consistent set Δ_p such that $\Delta_p \prec \Delta$ and $\Delta_p \prec \Delta_1$. We can apply Lemma 4.32 a second time and infer that there must also be a maximally consistent set Δ_p' with $\Delta_p' \prec \Delta$ and $\Delta_p' \prec \Delta_2$. But $\prec$ has been shown to be a conversely functional relation in Lemma 4.23, so we must have $\Delta_p = \Delta_p'$, as these sets are both predecessors of the same set Δ. We can now see that there exists a maximally consistent set, namely Δ_p, that is a predecessor (with respect to $\prec$) to both Δ_1 and Δ_2. In other words, we have both $\Delta_p \prec \Delta_1$ and $\Delta_p \prec \Delta_2$. Therefore, Lemma 4.33 is applicable, that is, we must have either $\Delta_1 = \Delta_2$ or $\Delta_1 < \Delta_2$ or $\Delta_2 < \Delta_1$ as claimed. $\Box$

Lemma 4.35 (No branching to the left) *Let Δ, Δ_1 and Δ_2 be maximally consistent sets such that $\Delta_1 < \Delta$ and $\Delta_2 < \Delta$. Then we have either $\Delta_1 = \Delta_2$ or $\Delta_1 < \Delta_2$ or $\Delta_2 < \Delta_1$.*

Proof. Omitted. The proof would be almost identical to that of Lemma 4.34. $\Box$

Lemma 4.36 (Functionality of canonical neighbour relations) *Both $<_\circ$ and its inverse as declared over maximally consistent sets are functional relations.*

Proof. For the two parts of this proof, we are going to make use of axioms (F2): $\ominus A \leftrightarrow (\neg \ominus\neg A \wedge \ominus\top)$ and (F1): $\ominus A \leftrightarrow (\neg \ominus\neg A \wedge \ominus\top)$, respectively.

We first show that $<_\circ$ is a functional relation. Let Δ, Δ_1 and Δ_2 be maximally consistent sets of formulas with $\Delta <_\circ \Delta_1$ and $\Delta <_\circ \Delta_2$. Functionality of $<_\circ$ would then imply $\Delta_1 = \Delta_2$. So for the sake of contradiction, assume this is not the case, i.e. there exists a formula φ with $\varphi \in \Delta_1$ and $\neg\varphi \in \Delta_2$. By Lemma 4.26, the former entails

$\circlearrowright\varphi \in \Delta$, while the latter implies $\circlearrowright\neg\varphi \in \Delta$. But from (F2) and $\circlearrowright\varphi \in \Delta$ we can infer $\neg\circlearrowright\neg\varphi \in \Delta$, that is, we have derived a contradiction.

Let us now turn to the second part of the claim. Suppose Δ, Δ_1 and Δ_2 are maximally consistent sets such that $\Delta_1 <_\circ \Delta$ and $\Delta_2 <_\circ \Delta$. To prove that the inverse of $<_\circ$ is a functional relation, we are going to show that the assumption $\Delta_1 \neq \Delta_2$ leads to a contradiction. By this assumption, there must be a formula φ with $\varphi \in \Delta_1$ and $\neg\varphi \in \Delta_2$. We apply again Lemma 4.26 and get $\circlearrowleft\varphi \in \Delta$ and $\circlearrowleft\neg\varphi \in \Delta$. The former together with (F1) gives us $\neg\circlearrowleft\neg\varphi \in \Delta$, which contradicts $\circlearrowleft\neg\varphi \in \Delta$. $\square$

Lemma 4.37 (Neighbours and siblings) *Let Δ_1 and Δ_2 be maximally consistent sets. Then $\Delta_1 <_\circ \Delta_2$ implies $\Delta_1 < \Delta_2$.*

Proof. We are going to use theorem $(\boxminus A \wedge \circlearrowright\top) \rightarrow \circlearrowright A$ from Lemma 4.8, whose derivation crucially depended on axiom (M2). Let Δ_1 and Δ_2 be maximally consistent sets such that $\Delta_1 <_\circ \Delta_2$ holds. To show that $\Delta_1 < \Delta_2$ must be true as well, we have to show that $\boxminus\varphi \in \Delta_1$ implies $\varphi \in \Delta_2$ for all formulas φ. So let $\boxminus\varphi \in \Delta_1$. Because of $\Delta_1 <_\circ \Delta_2$, we also have $\circlearrowright\top \in \Delta_1$. That means we can apply theorem $(\boxminus A \wedge \circlearrowright\top) \rightarrow \circlearrowright A$ and get $\circlearrowright\varphi \in \Delta_1$. Now $\Delta_1 <_\circ \Delta_2$ entails $\varphi \in \Delta_2$ by definition of $<_\circ$. $\square$

The following is another auxiliary lemma.

Lemma 4.38 (Composed canonical sibling relation) *Let Δ_1 and Δ_2 be maximally consistent sets. Then $\Delta_1 <^2 \Delta_2$ holds iff $\boxminus\boxminus\varphi \in \Delta_1$ implies $\varphi \in \Delta_2$ for all formulas φ.*

Proof. '$\Rightarrow$': This is the easy part. Let Δ_1 and Δ_2 be maximally consistent sets with $\Delta_1 <^2 \Delta_2$, that is, there exists a maximally consistent set Δ such that $\Delta_1 < \Delta$ and $\Delta < \Delta_2$. Now assume $\boxminus\boxminus\varphi \in \Delta_1$. By definition of $<$ we first get $\boxminus\varphi \in \Delta$ and then $\varphi \in \Delta_2$ as claimed.

'$\Leftarrow$': Now let Δ_1 and Δ_2 be such that $\boxminus\boxminus\varphi \in \Delta_1$ implies $\varphi \in \Delta_2$ for all formulas φ. To prove that this implies $\Delta_1 <^2 \Delta_2$ we are going to show that there exists a maximally consistent set Δ such that $\Delta_1 < \Delta$ and $\Delta < \Delta_2$ are true. We first construct a base set Δ' containing all the formulas that can immediately be seen to be required to be part of any such set 'between' Δ_1 and Δ_2:

$$\Delta' \quad = \quad \{\chi \mid \boxminus\chi \in \Delta_1\} \cup \{\Diamond\psi \mid \psi \in \Delta_2\}$$

We claim that Δ' is a consistent set of formulas. Otherwise there would have to be formulas $\chi_1, \ldots, \chi_n$ with $\boxminus\chi_i \in \Delta_1$ for all $i \in \{1, \ldots, n\}$ and formulas $\psi_1, \ldots, \psi_m \in \Delta_2$ such that $\vdash \neg(\chi_1 \wedge \cdots \wedge \chi_n \wedge \Diamond\psi_1 \wedge \cdots \wedge \Diamond\psi_m)$ holds. This can be seen to lead to a contradiction as follows. We can transform the previous statement into $\vdash (\chi_1 \wedge \cdots \wedge \chi_n) \rightarrow (\neg\Diamond\psi_1 \vee \cdots \vee \neg\Diamond\psi_m)$ and then use the generalisation rule and axiom (K2) to obtain $\vdash \boxminus(\chi_1 \wedge \cdots \wedge \chi_n) \rightarrow \boxminus(\neg\Diamond\psi_1 \vee \cdots \vee \neg\Diamond\psi_m)$. Hence, as we have $\boxminus(\chi_1 \wedge \cdots \wedge \chi_n) \in \Delta_1$ (by distribution), we must also have $\boxminus(\neg\Diamond\psi_1 \vee \cdots \vee \neg\Diamond\psi_m) \in \Delta_1$. By axiom (D2) this is equivalent to $\boxminus(\boxminus\neg\psi_1 \vee \cdots \vee \boxminus\neg\psi_m) \in \Delta_1$, which in turn is easily seen to imply $\boxminus\boxminus(\neg\psi_1 \vee \cdots \vee \neg\psi_m) \in \Delta_1$.[11] Together with our original assumption of $\boxminus\boxminus\varphi \in \Delta_1$ implying $\varphi \in \Delta_2$ for all formulas φ, this entails $\neg\psi_1 \vee \cdots \vee \neg\psi_m \in \Delta_2$. But this contradicts $\psi_1, \ldots, \psi_m \in \Delta_2$.

Thus Δ' must be consistent and, by Lemma 4.12, we can extend it to a maximally consistent set Δ. We have $\Delta_1 < \Delta$ by construction and $\Delta < \Delta_2$ must hold as well, because $\neg\varphi \in \Delta_2$ implies $\Diamond\neg\varphi \in \Delta$ (which is equivalent to $\neg\boxminus\varphi \in \Delta$), again, by construction. Hence, $\Delta_1 <^2 \Delta_2$ holds as claimed. $\square$

Lemma 4.39 (Direct siblings and neighbours) *Let Δ_1 and Δ_2 be maximally consistent sets with $\Delta_1 < \Delta_2$ and suppose there exists no maximally consistent set Δ such that both $\Delta_1 < \Delta$ and $\Delta < \Delta_2$ hold. Then we must have $\Delta_1 <_\circ \Delta_2$.*

Proof. The proof will employ axiom (H2): $\boxminus\boxminus A \to (\boxminus A \vee \ominus\top)$. Suppose Δ_1 and Δ_2 are maximally consistent sets as specified in the claim, that is, we have $\Delta_1 < \Delta_2$ and there is no maximally consistent set Δ such that $\Delta_1 < \Delta$ and $\Delta < \Delta_2$. We are first going to show that there exists a maximally consistent set Δ_2' such that $\Delta_1 <_\circ \Delta_2'$ holds and then that, in fact, we must have $\Delta_2' = \Delta_2$ for any such set.

For the sake of contradiction, assume there exists no Δ_2' such that $\Delta_1 <_\circ \Delta_2'$. By the Right Neighbour Existence Lemma, this implies $\ominus\top \not\in \Delta_1$. We are going to derive our contradiction by showing that under these conditions there will be a Δ such that $\Delta_1 < \Delta < \Delta_2$. According to Lemma 4.38, this is the case iff $\boxminus\boxminus\varphi \in \Delta_1$ implies $\varphi \in \Delta_2$ for every formula φ. So let $\boxminus\boxminus\varphi \in \Delta_1$. Now axiom (H2) applies and we can conclude that the disjunction $\boxminus\varphi \vee \ominus\top$ must also be an element of Δ_1. Together with $\ominus\top \not\in \Delta_1$ this entails $\boxminus\varphi \in \Delta_1$. But then $\Delta_1 < \Delta_2$ entails $\varphi \in \Delta_2$, that is, we have indeed found a contradiction.

Hence, there exists a maximally consistent set Δ_2' with $\Delta_1 <_\circ \Delta_2'$. By Lemma 4.37, $<_\circ$ is a subrelation to $<$, that is, we also have $\Delta_1 < \Delta_2'$. Applying Lemma 4.34 to the latter and $\Delta_1 < \Delta_2$ allows us to infer that either $\Delta_2 = \Delta_2'$ or $\Delta_2 < \Delta_2'$ or $\Delta_2' < \Delta_2$ must be true. We have to exclude the latter two cases. In fact, $\Delta_2' < \Delta_2$ is immediately seen to be contradictory, as in that case we would have a maximally consistent set Δ, namely Δ_2', such that $\Delta_1 < \Delta < \Delta_2$ (and this possibility is explicitly excluded in the claim). So it remains to be shown that the assumption $\Delta_2 < \Delta_2'$ also leads to a contradiction. We are again going to make use of the original assumption that there is no maximally consistent set Δ such that $\Delta_1 < \Delta < \Delta_2$. Again, by Lemma 4.38, this is the same as to say that there exists a formula φ such that $\boxminus\boxminus\varphi \in \Delta_1$ but $\varphi \not\in \Delta_2$. The former together with $\Delta_1 < \Delta_2 < \Delta_2'$ yields $\varphi \in \Delta_2'$. Furthermore, $\varphi \not\in \Delta_2$ and $\Delta_1 < \Delta_2$ entail $\Diamond\neg\varphi \in \Delta_1$. We also have $\ominus\top \in \Delta_1$ because of $\Delta_1 <_\circ \Delta_2'$. Now we can apply axiom (M2):

[11] Here is a sketch for a formal derivation: $\neg\psi_1$ implies the disjunction $\neg\psi_1 \vee \cdots \vee \neg\psi_m$. Using the generalisation rule and axiom (K2) we can see that therefore also $\boxminus\neg\psi_1$ implies $\boxminus(\neg\psi_1 \vee \cdots \vee \neg\psi_m)$. By the same argument, $\boxminus\neg\psi_i$ implies $\boxminus(\neg\psi_1 \vee \cdots \vee \neg\psi_m)$ for all $i \in \{1, \ldots, m\}$. Hence, $\boxminus\neg\psi_1 \vee \cdots \vee \boxminus\neg\psi_m$ implies $\boxminus(\neg\psi_1 \vee \cdots \vee \neg\psi_m)$. Applying the generalisation rule and axiom (K2) once more then yields the desired result.

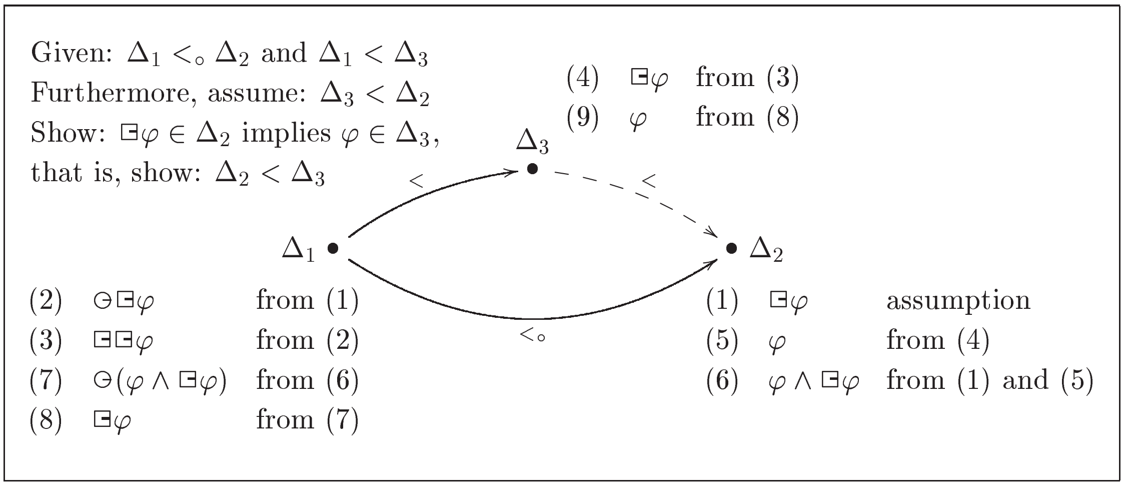

Figure 4.1: Proof detail for Lemma 4.40

$(\diamondminus A \wedge \ominus\top) \leftrightarrow (\ominus A \vee \ominus\diamondminus A)$, that is, we must have either $\ominus\neg\varphi \in \Delta_1$ or $\ominus\diamondminus\neg\varphi \in \Delta_1$. By the Right Neighbour Existence Lemma and the fact that $<_\circ$ is known to be a functional relation (Lemma 4.36), the former contradicts $\varphi \in \Delta_2'$. By the same argument, the latter implies $\diamondminus\neg\varphi \in \Delta_2'$. But then, because of $\Delta_1 < \Delta_2'$, we have $\diamondminus\diamondminus\neg\varphi \in \Delta_1$. This can be seen to contradict $\boxminus\boxminus\varphi \in \Delta_1$ using axiom (D2). In summary, this means we have shown that $\Delta_2 = \Delta_2'$ is the only remaining case, that is, $\Delta_1 <_\circ \Delta_2$ must hold as claimed. □

Lemma 4.40 (Right siblings of left neighbours) *Let Δ_1, Δ_2 and Δ_3 be maximally consistent sets such that $\Delta_1 <_\circ \Delta_2$ and $\Delta_1 < \Delta_3$. Then we must have either $\Delta_2 = \Delta_3$ or $\Delta_2 < \Delta_3$.*

Proof. We are going to use the derived theorems $\ominus\boxminus A \rightarrow \boxminus\boxminus A$ and $\ominus(A \wedge \boxminus A) \rightarrow \boxminus A$ from Lemma 4.8, the derivation of which particularly relied on axioms (M2) and (F2). Let Δ_1, Δ_2 and Δ_3 be maximally consistent sets such that $\Delta_1 <_\circ \Delta_2$ and $\Delta_1 < \Delta_3$. By Lemma 4.37, the former implies $\Delta_1 < \Delta_2$. Hence, we can apply Lemma 4.34 to infer that $\Delta_2 = \Delta_3$ or $\Delta_2 < \Delta_3$ or $\Delta_3 < \Delta_2$. If either $\Delta_2 = \Delta_3$ or $\Delta_2 < \Delta_3$ is the case we are done, so let us assume $\Delta_3 < \Delta_2$. We are going to prove that $\Delta_2 < \Delta_3$ must be true as well, by showing that $\boxminus\varphi \in \Delta_2$ implies $\varphi \in \Delta_3$ for any formula φ. So let $\boxminus\varphi \in \Delta_2$. The reader may find it helpful to refer to Figure 4.1 when following this part of the argument.

By Lemma 4.26, we have $\ominus\boxminus\varphi \in \Delta_1$ (because of $\Delta_1 <_\circ \Delta_2$) and we can use theorem $\ominus\boxminus A \rightarrow \boxminus\boxminus A$ to obtain $\boxminus\boxminus\varphi \in \Delta_1$. Thus we also get $\boxminus\varphi \in \Delta_3$ and then $\varphi \in \Delta_2$ (because of $\Delta_1 < \Delta_3 < \Delta_2$). Now we have $\ominus(\varphi \wedge \boxminus\varphi) \in \Delta_1$ and can apply theorem $\ominus(A \wedge \boxminus A) \rightarrow \boxminus A$ to obtain $\boxminus\varphi \in \Delta_1$. Hence, $\varphi \in \Delta_3$ must hold (because of $\Delta_1 < \Delta_3$), which is what we intended to derive. So at least one of $\Delta_2 = \Delta_3$ and $\Delta_2 < \Delta_3$ will be true in any case. □

Lemma 4.41 (Left siblings of right neighbours) *Let Δ_1, Δ_2 and Δ_3 be maximally consistent sets such that $\Delta_1 < \Delta_3$ and $\Delta_2 <_\circ \Delta_3$. Then we must have either $\Delta_1 = \Delta_2$ or $\Delta_1 < \Delta_2$.*

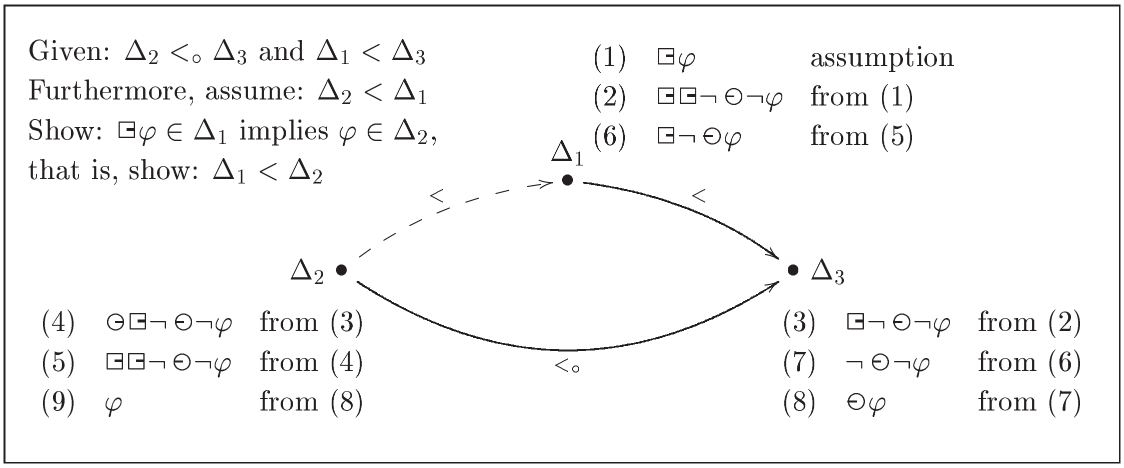

Figure 4.2: Proof detail for Lemma 4.41

Proof. The structure of this proof is similar to that of the previous lemma. This time we are going to use axiom (H3): $\boxminus A \to \boxminus\boxminus\neg\ominus\neg A$ and the derived theorem $\ominus\boxminus A \to \boxminus\boxminus A$ from before. Let Δ_1, Δ_2 and Δ_3 be maximally consistent sets with $\Delta_1 < \Delta_3$ and $\Delta_2 <_\circ \Delta_3$. The latter implies $\Delta_2 < \Delta_3$ and we can apply Lemma 4.35 to see that either $\Delta_1 = \Delta_2$ or $\Delta_1 < \Delta_2$ or $\Delta_2 < \Delta_1$. We are done if we can prove that $\Delta_2 < \Delta_1$ implies $\Delta_1 < \Delta_2$. So assume $\Delta_2 < \Delta_1$ and $\boxminus\varphi \in \Delta_1$. (We need to show $\varphi \in \Delta_2$.) The following argument is also summarised in Figure 4.2.

Axiom (H3) together with $\boxminus\varphi \in \Delta_1$ entails $\boxminus\boxminus\neg\ominus\neg\varphi \in \Delta_1$. Now we obtain first $\boxminus\neg\ominus\neg\varphi \in \Delta_3$ (because of $\Delta_1 < \Delta_3$) and then $\ominus\boxminus\neg\ominus\neg\varphi \in \Delta_2$ (because of $\Delta_2 <_\circ \Delta_3$). At this point theorem $\ominus\boxminus A \to \boxminus\boxminus A$ applies and we get $\boxminus\boxminus\neg\ominus\neg\varphi \in \Delta_2$ as well. Because of $\Delta_2 < \Delta_1 < \Delta_3$ we then get $\neg\ominus\neg\varphi \in \Delta_3$. As we certainly have $\ominus\top \in \Delta_3$ (from $\Delta_2 <_\circ \Delta_3$ and Lemma 4.26), the functionality axiom (F1) allows us to infer $\ominus\varphi \in \Delta_3$. Hence, by the Left Neighbour Existence Lemma and the fact that the inverse of $<_\circ$ is a functional relation (Lemma 4.36) we get $\varphi \in \Delta_2$ as required. $\square$

We are now ready to put together our results on properties of the structure imposed on the set of maximally consistent sets with a level by the three canonical accessibility relations $\prec$, $<$ and $<_\circ$.

Lemma 4.42 (Generated canonical loosely ordered trees) *Let Δ^* be a maximally consistent set with a level. Then $(T_{\Delta^*}, \prec, <, <_\circ)$ is a loosely ordered tree.*

Proof. Let Δ^* be a maximally consistent set with a level, that is, Δ^* contains a formula of the form $\bigcirc^n\boxdot\bot$. Then T_{Δ^*} is the set of maximally consistent sets generated by Δ^*. That is, any maximally consistent set that can be reached from Δ^* via a composition (of arbitrary length) of the relation $\prec$ and its inverse will belong to T_{Δ^*} (see Definition 4.22). Part of the claim of this lemma is that the sets in T_{Δ^*} are in fact nodes in a loosely ordered tree and that $\prec$ is the child relation for that tree. Furthermore, $<$ is claimed to be the corresponding pseudo sibling relation and $<_\circ$ is claimed to be the pseudo neighbour relation.

To prove this, we have to check the nine conditions from Definition 3.33. These have all been shown to hold in earlier lemmas; we merely need to summarise:

(1) By Lemma 4.24, $(T_{\Delta^*}, \prec)$ is a tree.

(2) By Lemma 4.31, $<$ is a transitive relation.

(3) By Lemma 4.32, any two nodes related via $<$ will also be related via the composition of the inverse of $\prec$ and $\prec$ itself.

(4) By Lemma 4.33, any two nodes related via the composition of the inverse of $\prec$ and $\prec$ itself will also be related via either the reflexive closure or the inverse of $<$.

(5) By Lemma 4.36, both $<_\circ$ and its inverse are functional relations.

(6) By Lemma 4.37, any two nodes related via $<_\circ$ will also be related via $<$.

(7) By Lemma 4.39, any two nodes related via $<$ but not via the composition of $<$ with itself will also be related via $<_\circ$.

(8) By Lemma 4.40, any two nodes related via the composition of the inverse of $<_\circ$ and $<$ will also be related via the reflexive closure of $<$.

(9) By Lemma 4.41, any two nodes related via the composition of $<$ and the inverse of $<_\circ$ will also be related via the reflexive closure of $<$.

This concludes our proof of Lemma 4.42 □

Generated canonical models. The canonical relations $\prec$, $<$ and $<_\circ$ from Definitions 4.13 and 4.25 are all binary relations over the full set of maximally consistent sets. So they are also relations over the set of maximally consistent sets T_{Δ^*} generated by some set Δ^* (see Definition 4.22) and we may speak about a *frame* $(T_{\Delta^*}, \prec, <, <_\circ)$ in the sense of Definition 3.16.

Definition 4.43 (Generated canonical frames) *Let Δ^* be a maximally consistent set with a level. Then the general frame $\mathcal{F}_{\Delta^*} = (T_{\Delta^*}, \prec, <, <_\circ)$ is called the canonical frame generated by Δ^*.*

Next we define a canonical valuation to go with our generated canonical frames. This is going to be a function mapping propositional letters to sets of maximally consistent sets of formulas (the 'worlds' in our canonical model to be). A propositional letter P is intended to be true at a world represented by a maximally consistent set Δ iff P is an element of Δ. Our valuation function is defined accordingly.

Definition 4.44 (Generated canonical valuations) *Let Δ^* be a maximally consistent set with a level. The function V_{Δ^*} from the set of propositional letters to the generated set of maximally consistent sets T_{Δ^*}, with $V_{\Delta^*}(P) = \{\Delta \in T_{\Delta^*} \mid P \in \Delta\}$ for propositional letters P, is called the valuation function generated by Δ^*.*

Finally, we can give a formal definition of our canonical models.

Definition 4.45 (Generated canonical models) *Let Δ^* be a maximally consistent set with a level. Then the general model $\mathcal{M}_{\Delta^*} = (\mathcal{F}_{\Delta^*}, V_{\Delta^*})$, is called the canonical model generated by Δ^*.*

Truth in canonical models. After having established that our canonical frames have all the required properties (i.e. they are loosely ordered trees) and after having proved Existence Lemmas for the various existential modalities in our language, the last important step to take before we can formulate our completeness result is to prove an appropriate *Truth Lemma.* We have to show that the notion of truth in a generated canonical model (in the sense of Definition 3.18) coincides with our intended notion of truth, which simply reduces to membership in the respective maximally consistent set. In other words, a formula should be true at a world in a generated canonical model iff that formula is an element of the maximally consistent set of formulas associated with that world.

The following lemma is the first instance where we explicitly have to exclude the transitive descendant modalities $\Diamond^+$ and $\Box^+$ in order to be able to successfully prove the desired result. (Before we could formulate a Truth Lemma covering also the descendant modalities we would first have to prove a Descendant Existence Lemma.) The restriction to this fragment of the logic will be indicated by referring to formulas *in the language without $\Diamond^+$ and $\Box^+$*.

Lemma 4.46 (Truth in a generated canonical model) *Let Δ^* be a maximally consistent set with a level. Then the following holds for all maximally consistent sets $\Delta \in T_{\Delta^*}$ and all formulas φ in the language without $\Diamond^+$ and $\Box^+$:*

$$\mathcal{M}_{\Delta^*}, \Delta \models \varphi \quad \textit{iff} \quad \varphi \in \Delta$$

Proof. By induction over the structure of formulas φ. For the base case we have to consider propositional letter and the special symbols $\top$ and $\bot$. In the case of propositional letters P, the claim follows immediately from the definition of the generated canonical valuation function V_{Δ^*}. We have $\mathcal{M}_{\Delta^*}, \Delta \models P$ iff $\Delta \in V_{\Delta^*}(P)$ (by Definition 3.18) iff $P \in \Delta$ (by Definition 4.44). For the case of $\top$, simply observe that, on the one hand, we will have $\mathcal{M}_{\Delta^*}, \Delta \models \top$ in any model and for any world Δ, and that, on the other hand, we will have $\top \in \Delta$ for any maximally consistent set Δ, because $\top$ is a theorem (see Lemma 4.11). Furthermore, given that any axiomatisation of propositional logic will allow us to rewrite $\bot$ as $\neg\top$, we can treat $\bot$ as a definable proposition.

For the induction step, we have to consider each one of the operators of our language in turn (apart from $\Diamond_{\downarrow}^{+}$ and $\Box_{\downarrow}^{+}$, which have explicitly been excluded from the claim). In fact, we only need to consider the *core operators* of the language. By core operators, we mean the operators for which we have explicitly given a semantics in Definitions 3.6 and 3.18. These are negation and conjunction as well as diamond- and next-operators. Other propositional connectives and box-operators have been treated as definable operators in our semantic characterisation of ordered tree logics in the previous chapter (see also Definition 3.7). We can follow the same route here. Using axiom (D3), for instance, we can rewrite any formula of the form $\Box_{\Uparrow}\varphi$ as $\neg \Diamond_{\Uparrow}\neg\varphi$. Similarly, any axiomatisation of propositional logic will allow us to rewrite a disjunction in terms of negation and conjunction, etc.

The following list shows for each core operator how the claim for a formula where that operator is principal operator can be reduced to a case (or cases) where the induction hypothesis is applicable.

(1) *Negation.* By Definition 3.18 (of truth in a general model), we have $\mathcal{M}_{\Delta^*}, \Delta \models \neg\varphi$ iff $\mathcal{M}_{\Delta^*}, \Delta \not\models \varphi$. The latter is covered by the induction hypothesis, that is, we have $\mathcal{M}_{\Delta^*}, \Delta \models \varphi$ iff $\varphi \in \Delta$, which we may rewrite as $\mathcal{M}_{\Delta^*}, \Delta \not\models \varphi$ iff $\varphi \notin \Delta$. Furthermore, by Lemma 4.11, $\varphi \notin \Delta$ iff $\neg\varphi \in \Delta$, because Δ is a maximally consistent set. Putting them all together, we get $\mathcal{M}_{\Delta^*}, \Delta \models \neg\varphi$ iff $\neg\varphi \in \Delta$ as required.

(2) *Conjunction.* By Definition 3.18, $\mathcal{M}_{\Delta^*}, \Delta \models \varphi_1 \wedge \varphi_2$ iff both $\mathcal{M}_{\Delta^*}, \Delta \models \varphi_1$ and $\mathcal{M}_{\Delta^*}, \Delta \models \varphi_2$. The subformulas fall under the induction hypothesis, i.e. we have $\mathcal{M}_{\Delta^*}, \Delta \models \varphi_1$ iff $\varphi_1 \in \Delta$ as well as $\mathcal{M}_{\Delta^*}, \Delta \models \varphi_2$ iff $\varphi_2 \in \Delta$. By Lemma 4.11, we also have $\varphi_1 \in \Delta$ and $\varphi_2 \in \Delta$ iff $\varphi_1 \wedge \varphi_2 \in \Delta$. In summary, we get $\mathcal{M}_{\Delta^*}, \Delta \models \varphi_1 \wedge \varphi_2$ iff $\varphi_1 \wedge \varphi_2 \in \Delta$.

(3) *Child modality.* By Definition 3.18, we have $\mathcal{M}_{\Delta^*}, \Delta \models \Diamond_{\downarrow}\varphi$ iff there exists a $\Delta' \in T_{\Delta^*}$ such that $\Delta \prec \Delta'$ and $\mathcal{M}_{\Delta^*}, \Delta' \models \varphi$ hold. By the induction hypothesis, the latter is equivalent to $\varphi \in \Delta'$ (because Δ' must be a member of T_{Δ^*} as well). Furthermore, by Lemma 4.14, $\varphi \in \Delta'$ together with $\Delta \prec \Delta'$ implies $\Diamond_{\downarrow}\varphi \in \Delta$ (for any maximally consistent set Δ'). For the opposite direction we use the Child Existence Lemma, by which $\Diamond_{\downarrow}\varphi \in \Delta$ implies the existence of some maximally consistent set Δ' with $\Delta \prec \Delta'$ and $\varphi \in \Delta'$. That this set Δ' will be a member of T_{Δ^*} follows immediately from $\Delta \prec \Delta'$ and the fact that $\Delta \in T_{\Delta^*}$. Altogether, this shows $\mathcal{M}_{\Delta^*}, \Delta \models \Diamond_{\downarrow}\varphi$ iff $\Diamond_{\downarrow}\varphi \in \Delta$ as required.

(4) *Parent modality.* $\mathcal{M}_{\Delta^*}, \Delta \models \bigcirc_{\uparrow}\varphi$ iff $\bigcirc_{\uparrow}\varphi \in \Delta$ can be shown in a manner analogous to the previous case, using again Lemma 4.14 and the Parent Existence Lemma.

(5) *Ancestor modality.* Again by Definition 3.18, $\mathcal{M}_{\Delta^*}, \Delta \models \Diamond_{\Uparrow}\varphi$ implies the existence of some Δ' with $\Delta' \prec^{+} \Delta$ and $\mathcal{M}_{\Delta^*}, \Delta' \models \varphi$. By the induction hypothesis the latter is equivalent to $\varphi \in \Delta'$. Now an application of Lemma 4.14 proves $\Diamond_{\Uparrow}\varphi \in \Delta$.

For the opposite direction we have to use the Ancestor Existence Lemma. Recall that this lemma requires an additional side condition to be fulfilled. We first have to show that the set Δ has got a level. But this follows immediately from the fact that Δ^* has got a level and that, by Lemma 4.17, every maximally consistent set that is accessible from a set with a level by either $\prec$ or its inverse must have a level, too. Given the definition of T_{Δ^*}, every set in T_{Δ^*} (including Δ) must have a level. Now we can apply the Ancestor Existence Lemma and infer that there must be a maximally consistent set Δ' with $\Delta' \prec^+ \Delta$ and $\varphi \in \Delta'$. By definition, Δ' will be a member of T_{Δ^*} and we are done.

(6) *Right sibling modality.* $\mathcal{M}_{\Delta^*}, \Delta \models \Diamond_{\rightarrow}\varphi$ iff $\Diamond_{\rightarrow}\varphi \in \Delta$ essentially follows from Lemma 4.26 and the Right Sibling Existence Lemma. We also need to use the fact that whenever two maximally consistent sets Δ and Δ' are related via the canonical sibling relation and one of them is known to belong to T_{Δ^*}, then the other must also belong to T_{Δ^*}. This follows from Lemma 4.32 and the definition of T_{Δ^*}.

(7) *Left sibling modality.* $\mathcal{M}_{\Delta^*}, \Delta \models \Diamond_{\leftarrow}\varphi$ iff $\Diamond_{\leftarrow}\varphi \in \Delta$ follows from Lemma 4.26 and the Left Sibling Existence Lemma.

(8) *Right neighbour modality.* $\mathcal{M}_{\Delta^*}, \Delta \models \bigcirc_{\rightarrow}\varphi$ iff $\bigcirc_{\rightarrow}\varphi \in \Delta$ follows from Lemma 4.26 and the Right Neighbour Existence Lemma. (Observe that by Lemmas 4.37 and 4.32, $\Delta \in T_{\Delta^*}$ and $\Delta <_\circ \Delta'$ implies $\Delta' \in T_{\Delta^*}$.)

(9) *Left neighbour modality.* $\mathcal{M}_{\Delta^*}, \Delta \models \bigcirc_{\leftarrow}\varphi$ iff $\bigcirc_{\leftarrow}\varphi \in \Delta$ follows from Lemma 4.26 and the Left Neighbour Existence Lemma.

This conclues our inductive proof of Lemma 4.46 □

Completeness. We are now ready to state our Completeness Theorem. The objective of showing completeness for an axiomatic system with respect to a given semantic characterisation of a logic is to show that every formula (or possibly even every set of formulas) that is *consistent* (a notion defined in terms of axioms and rules) is also *satisfiable* (a notion defined in terms of the given model-theoretic semantics). In our case the result is not that general; there are two restrictions.

First of all, our Completeness Theorem only applies to formulas in the language without the transitive descendant modalities $\Diamond_{\downarrow}^{+}$ and $\Box_{\downarrow}^{+}$.[12] While this is certainly an important restriction, the fragment of our logic excluding these modalities is still an interesting and useful system. In particular, the child modality $\Diamond_{\downarrow}$ is still available to explicitly relate a system to its (immediate) subsystems.

The second restriction is that the following Completeness Theorem only applies to formulas that have a level, that is, to formulas of the form $\varphi \wedge \bigcirc_{\uparrow}^{n}\Box_{\uparrow}\bot$ for some number

[12] Consequently, our proof has not taken into account any of the axioms involving either $\Diamond_{\downarrow}^{+}$ or $\Box_{\downarrow}^{+}$.

$n \in \mathbb{N}_0$.[13] Again, this restriction cannot be overlooked as far as our theoretical study is concerned, but it may be of lesser importance with respect to actual applications. It seems natural to describe a particular system from the viewpoint of the root node or possibly from the viewpoint of some 'first' node on the first level of nodes beneath the root. In either case, we can simply add the conjunct $\boxdot\bot$ or $\odot\boxdot\bot$, respectively, to the formula representing the specification and our Completeness Theorem becomes applicable.

Theorem 4.47 (Completeness without descendant modalities) *Every consistent formula in the language without $\Diamond^+$ and $\Box^+$ that has got a level is satisfiable.*

Proof. Let $\varphi \wedge \odot^n\boxdot\bot$ (with $n \in \mathbb{N}_0$) be a consistent formula in the language without the descendant modalities $\Diamond^+$ and $\Box^+$. By Lemma 4.12, there exists a maximally consistent set Δ^* with $\varphi \wedge \odot^n\boxdot\bot \in \Delta^*$. Lemma 4.12 holds independently of a particular modal language. That is, there exists a set Δ^* that is maximally consistent with respect to the axiom system for the fragment excluding the two descendant modalities such that $\varphi \wedge \odot^n\boxdot\bot \in \Delta^*$ and the formulas in Δ^* do not contain the operators $\Diamond^+$ or $\Box^+$. As we have $\odot^n\boxdot\bot \in \Delta^*$, the set Δ^* must have a level as well, namely the very same n. Thus, by Lemma 4.42, the canonical frame $\mathcal{F}_{\Delta^*} = (T_{\Delta^*}, \prec, <, <_\circ)$ generated by Δ^* is a loosely ordered tree. So the canonical model $\mathcal{M}_{\Delta^*} = (\mathcal{F}_{\Delta^*}, V_{\Delta^*})$ is in fact a model based on a loosely ordered tree. By Lemma 4.46 (our Truth Lemma), we have $\mathcal{M}_{\Delta^*}, \Delta^* \models \varphi \wedge \odot^n\boxdot\bot$, that is, our input formula is satisfiable in the generated canonical model. But by Theorem 3.41, any formula satisfiable in a model based on a loosely ordered tree is also satisfiable in a model based on an ordered tree, that is, any such formula is a satisfiable formula in the original sense of the word (that is, in the sense of Definition 3.8). Hence, $\varphi \wedge \odot^n\boxdot\bot$ must be a satisfiable formula as claimed. $\Box$

Validity implies theoremhood. In general, we may phrase a (weak) Completeness Theorem in two ways. By the first formulation, every consistent formula is claimed to be also a satisfiable formula. By the second formulation, every valid formula is claimed to be a derivable theorem.[14] For our Completeness Theorem above, we have chosen the former variant. In the general case, the two formulations would be equivalent, but for our restricted setting where we only deal with certain kinds of formulas, namely those that have a level, we need to be a little more careful. The appropriate formulation for the second variant is given by the following corollary to Theorem 4.47.

Corollary 4.48 (Completeness without descendant modalities) *Every valid formula of the form $\odot^n\boxdot\bot \rightarrow \varphi$ with $n \in \mathbb{N}_0$ in the language without $\Diamond^+$ and $\Box^+$ is a theorem.*

[13] At the beginning of the next section, we are going to take a closer look at the connections between these two restrictions.

[14] The first formulation is sometimes referred to as the *Consistency Lemma*.

Proof. Let $\odot^n \boxdot \bot \rightarrow \varphi$ with some $n \in \mathbb{N}_0$ be a valid formula. For the sake of contradiction, assume $\odot^n \boxdot \bot \rightarrow \varphi$ is not a theorem, that is, assume $\neg(\odot^n \boxdot \bot \rightarrow \varphi)$ is consistent. This formula is equivalent to $\neg\varphi \wedge \odot^n \boxdot \bot$, i.e. Theorem 4.47 is applicable and $\neg(\odot^n \boxdot \bot \rightarrow \varphi)$ must be satisfiable. But then $\odot^n \boxdot \bot \rightarrow \varphi$ cannot be a valid formula, which contradicts our original assumption. Hence, $\odot^n \boxdot \bot \rightarrow \varphi$ must be a theorem as claimed. □

Strong completeness. Theorem 4.47 is a weak completeness result. It applies to formulas rather than to (possibly infinite) sets of formulas. However, we could easily reformulate it to apply to sets of formulas as well.[15]

This seems to contradict what we have said earlier about strong and weak completeness and their connection to the compactness property, which we have disproved in Proposition 4.4. The reason that there is in fact no contradiction is that we have proved completeness only with respect to a certain class of formulas, namely those that have a level. That is, in a corresponding strong Completeness Theorem a consistent set of formulas would only be claimed to be satisfiable if it contained a formula of the form $\odot^n \boxdot \bot$. An inspection of the proof of Proposition 4.4 reveals that the particular infinite set given there as a counterexample for compactness does in fact *not* have a level. It is also easily seen that, if we add a formula of the form $\odot^n \boxdot \bot$ to that set, the proof of Proposition 4.4 will not go through anymore. In fact, the basic idea behind disproving compactness has been precisely the fact that any satisfiable set of formulas that is finite can always be augmented with a level formula, whereas an infinite set allows us to explicitly contradict every possible level formula.

Remark on the interaction of vertical and horizontal operators. In the first section of Chapter 3 we have briefly remarked on the subtle interaction between vertical and horizontal operators in **OTL** and compared it to the interaction of modal operators in product logics (see page 46). Recall that (X1) and (X2) are the *only* axioms that involve both vertical and horizontal operators. Our Completeness Theorem thus shows that the interaction of the vertical and the horizontal component in our modal logic of ordered trees is characterised by these two axioms.

[15]To be precise, this would also require Theorem 3.41 to be reformulated accordingly so as to apply to sets of formulas rather than just plain formulas. To see that such an extension would be straightforward, observe that Theorem 3.24 (which is used in the proof of Theorem 3.41) not only states that a formula which is satisfiable in one frame is also satisfiable in any p-morphic image of that frame (and vice versa), but it also implies that any two formulas true at the same world in the first model will also be true at the same world in the second one. Hence, Theorem 3.24, and therefore also Theorem 3.41, apply to sets of formulas as well as to single formulas.

4.4 Alternatives

Our completeness result is not entirely satisfactory. It only applies to formulas with a level in the fragment of **OTL** that excludes the two descendant modalities. However, we still consider this result a valid contribution to our understanding of ordered trees.

In this section, we are going to discuss the two restriction to our completeness result as well as their connection again in some more detail. We are also going to review a number of alternative approaches to proving completeness that we have considered in our attempts to improve on the present result. Unfortunately, to date, these attempts have not been blessed with success. Whether there is a finite axiomatisation of full **OTL** remains an open question.[16]

What is missing? Let us recapitulate why we had to restrict our Completeness Theorem to the language excluding the descendant modalities $\Diamond_{\downarrow}^{+}$ and $\Box_{\downarrow}^{+}$. The problem can be traced down to the fact that we were not able to prove a Descendant Existence Lemma in analogy to the Ancestor Existence Lemma and the Existence Lemmas for the other compound accessibility relations. A Descendant Existence Lemma would have stated that for any maximally consistent set Δ_1 with $\Diamond_{\downarrow}^{+}\varphi \in \Delta_1$ (for some formula φ) we may infer that there exists a maximally consistent set Δ_2 such that $\Delta_1 \prec^+ \Delta_2$ and $\varphi \in \Delta_2$ hold. The proofs for the other Existence Lemmas were constructive: we first composed a set Δ_2' containing φ and any formula ψ that the set Δ_1 would 'force' into a set meeting the requirements of the lemma (through boxed formulas in Δ_1), then extended Δ_2' to a maximally consistent set Δ_2 (by appealing to Lindenbaum's Lemma), and finally showed that the required relationship between Δ_1 and Δ_2 followed by construction. In the case of the various canonical accessibility relations, this last step in the argument has always been a simple consequence of the definition of the respective relation in terms of the corresponding box-operator. However, such an argument does not apply for modal operators pertaining to the transitive closure of a basic accessibility relation. In the case of the Ancestor Existence Lemma, which also makes reference to $\prec^+$, we were able to circumvent this problem by adding a condition requiring the first maximally consistent set to have a level. This allowed us, essentially, to conclude that for any formula of the form $\Diamond_{\uparrow}\varphi$ in our maximally consistent set there must be a number $n \in \mathbb{N}$ such that $\bigcirc_{\uparrow}^{n}\varphi$ must also be in that set. In fact, this number n could not be greater than the level of the maximally consistent set in question. At this point, the Ancestor Existence Lemma reduced to an n-fold iteration of the uncritical Parent Existence Lemma.

A similar reduction of a Descendant Existence Lemma to an iteration of the Child Existence Lemma has not been possible, because when moving *down* along the branches of a tree we cannot use the root of the tree to provide a bound for the iteration in the same way as we could when moving up the tree. To be able to reduce the Descendant Existence Lemma to the Child Existence Lemma we would have to be able to infer that

[16] In case a finite axiomatisation of the full logic *does* exists, we would expect the induction axiom $\Box_{\downarrow}^{+}(A \to \Box_{\downarrow}A) \to (\Box_{\downarrow}A \to \Box_{\downarrow}^{+}A)$ to be part of that system (see page 100).

there exists an $n \in \mathbb{N}$ such that $\Diamond_{\downarrow}^{n}\varphi \in \Delta$ holds whenever we have $\Diamond_{\downarrow}^{+}\varphi \in \Delta$ (for any formula φ and any maximally consistent set Δ).

Levels and descendant modalities. At the beginning of the previous section we have already hinted at the fact that the two restrictions to our Completeness Theorem are interdependent. We are going to substantiate this claim by showing that (and how), once we have — somehow — mastered the transitive descendant modality $\Diamond_{\downarrow}^{+}$, we can do away with the restriction to formulas that have a level as well.[17]

We are first going to show that, whenever φ is a consistent formula, then either $\varphi \wedge \Box_{\uparrow}\bot$ or $\Diamond_{\downarrow}^{+}\varphi \wedge \Box_{\uparrow}\bot$ will be consistent, too. We are going to use the rootedness axiom (R1): $(A \wedge \Diamond_{\uparrow}\top) \rightarrow \Diamond_{\uparrow}(\Diamond_{\downarrow}^{+}A \wedge \Box_{\uparrow}\bot)$. Let φ be a consistent formula. For the sake of contradiction, assume that both $\varphi \wedge \Box_{\uparrow}\bot$ and $\Diamond_{\downarrow}^{+}\varphi \wedge \Box_{\uparrow}\bot$ are inconsistent formulas. This means that $\neg(\varphi \wedge \Box_{\uparrow}\bot)$ and $\neg(\Diamond_{\downarrow}^{+}\varphi \wedge \Box_{\uparrow}\bot)$ are theorems. As the following derivation shows, this implies that $\neg\varphi$ will be a theorem as well:

(1)	$\neg(\varphi \wedge \Box_{\uparrow}\bot)$	assumption
(2)	$\neg(\Diamond_{\downarrow}^{+}\varphi \wedge \Box_{\uparrow}\bot)$	assumption
(3)	$\Box_{\uparrow}\neg(\Diamond_{\downarrow}^{+}\varphi \wedge \Box_{\uparrow}\bot)$	from (2) by generalisation
(4)	$\neg\,\Diamond_{\uparrow}(\Diamond_{\downarrow}^{+}\varphi \wedge \Box_{\uparrow}\bot)$	form (3) and (D3)
(5)	$\neg(\varphi \wedge \Diamond_{\uparrow}\top)$	from (4) and (R1)
(6)	$\neg\varphi \vee \Box_{\uparrow}\bot$	from (5) and (D3)
(7)	$\neg\varphi \vee \neg\,\Box_{\uparrow}\bot$	from (1)
(8)	$\neg\varphi$	from (6) and (7)

But this contradicts our original assumption of φ being consistent. Hence, at least one of the two formulas $\varphi \wedge \Box_{\uparrow}\bot$ and $\Diamond_{\downarrow}^{+}\varphi \wedge \Box_{\uparrow}\bot$ must be consistent as claimed.

To see how this observation can help us to prove a Completeness Theorem without restrictions, suppose — for the moment — we have managed to prove completeness with respect to the full language of **OTL**, but have yet to eliminate the restriction to formulas with a level. We have to show that every consistent formula φ is satisfiable. Given that φ is consistent, so is either $\varphi \wedge \Box_{\uparrow}\bot$ or $\Diamond_{\downarrow}^{+}\varphi \wedge \Box_{\uparrow}\bot$. Both these formulas have got a level. Hence, by our imaginary Completeness Theorem, at least one of them (namely the consistent one) will also be satisfiable. In case of the former, this immediately implies satisfiability of φ and we are done. Now suppose $\Diamond_{\downarrow}^{+}\varphi \wedge \Box_{\uparrow}\bot$ is the consistent (and thereby also satisfiable) formula. This means that there exists a model such that $\Diamond_{\downarrow}^{+}\varphi$ is true at the root of the underlying tree. But then, by definition of the semantics of the $\Diamond_{\downarrow}^{+}$-operator, the root must have a descendant node at which φ holds, i.e. φ will be satisfiable in any case.

In short, the rootedness axiom (R1), which we have used in the above derivation, provides a 'syntactical connection' between the problems pertaining to the descendant modalities and the problems pertaining to formulas that cannot be associated with a

[17] Of course, this is only a hypothetical argument. Our aim is to make the connection between the two restrictions more apparent.

specific level in a tree. From a semantic point of view, the problems surrounding our completeness result also seem to be connected to our restriction of frames to *well-founded* trees, that is, to trees of order type $\mathbb{N}$, where any descendant of a given node can always be reached by a finite iteration of the child relation. Nodes in trees that are *not* well-founded need not have a level (in the sense in which we have used this term here) and the descendant relation would not be the transitive closure of the child relation anymore either.

Finitary methods. We have mentioned the problem of axiomatising the semantics of a modal operator defined with respect to the transitive closure of a basic accessibility relation, specifically in the case of the transitive descendent modality $\diamondsuit^+$, a few times now. The reader familiar with completeness proofs for propositional dynamic logic (**PDL**) may wonder why we have not followed the standard route and built a canonical model out of sets of formulas that are Fischer-Ladner closed. **PDL** is an example for a logic that includes a transitive closure operator. In standard completeness proofs for this kind of logics, rather than working with (infinite) maximally consistent sets of formulas, we build a canonical model out of finite sets of subformulas of the consistent formula for which we intend to show satisfiability. Additionally, these sets are required to be closed under a number of simple operations aimed at reflecting the semantics of the critical operators. For instance, if such a set contains the formula $\diamondsuit^+\varphi$ then it should also contain either $\diamondsuit\varphi$ or $\diamondsuit\diamondsuit^+\varphi$. In other words, sets have to be closed under 'application' of the mixing axiom (M4). This type of closure operation is known as the *Fischer-Ladner closure*.[18] A basic canonical accessibility relation can be declared over these sets of formulas in a similar fashion as we have seen it in our completeness proof. Provided the calculus includes an induction axiom like the one discussed on page 100, it is possible to prove completeness of the modal operator in question with respect to the transitive closure of the basic accessibility relation. A crucial aspect of this method is that the number of Fischer-Ladner closed sets of formulas is finite.

At first sight, this framework may appear to be applicable also to our logic. However, there is an important difference between **OTL** and logics such as **PDL**. In the class of frames characterising **PDL** the basic accessibility relation is not required to be irreflexive, but the child relation in an ordered tree is.[19] In a standard **PDL**-style canonical model construction, we cannot guarantee the basic accessibility relation to be irreflexive. If we attempt to overcome this problem by combining the approach based on the Fischer-Ladner closure with our idea of identifying sets of formulas with a level, we lose the aspect of finiteness, simply because there are infinitely many level formulas to choose from. Unfortunately, once there are infinitely many different sets of formulas that need to be considered for the construction of a canonical model, the usual argument for the transitive

[18] It would be beyond the scope of this brief reflection on alternative methods to give a detailed account on completeness proofs for **PDL** and related logics. Instead, we refer to [8] for further information.

[19] Another difference is that, typically, the transitive closure operators defined for logics such as **PDL** pertain to reflexive transitive closures, but this in itself would not constitute a serious problem.

closure operator does not go through anymore. This combination of irreflexivity and transitive closure appears to be the central problem in giving a finite axiomatisation of **OTL**. This is not to say that it is definitely impossible to prove completeness of **OTL** using an approach based on Fischer-Ladner closed sets of formulas, or some variant of it, but it is certainly not the case that the standard method would be immediately applicable to our setting.

Blackburn and Meyer-Viol [9] use a variant of this method to prove completeness of an axiomatisation for their modal logic of finite binary trees. In this logic the child relation is, of course, irreflexive and they successfully axiomatise the transitive closure of this child relation. Still, this does not contradict what has just been said about the difficulties of characterising the transitive closure of an irreflexive relation. Blackburn and Meyer-Viol's proof crucially depends on the fact that their logic is defined as a logic of *finite* frames. This allows for an inductive construction of the canonical model which, due to the (axiom encoding the) finiteness condition, is guaranteed to terminate eventually. This path is not open to us as **OTL** explicitly admits infinite trees.

Axiomatisations with an irreflexivity rule. The problem of giving an axiomatic characterisation of the descendant operator concerns the vertical dimension of an ordered tree. As for the horizontal dimension, we have succeeded in giving a complete axiomatisation of the operators concerned, but the technical apparatus required may seem overly powerful. In particular, we required Theorem 3.41 (the proof of which stretches over large parts of the second half of Chapter 3) to show that it was sufficient to prove completeness of our axiomatisation with respect to the class of loosely ordered trees (rather than just ordered trees). Essentially, these difficulties are due to the fact that it is not possible to give axioms that directly characterise the irreflexivity of the horizontal sibling relation in an ordered tree.

A similar problem occurs in standard temporal logics, where we cannot directly axiomatise the irreflexive nature of the underlying flow of time. Proving completeness of such logics can often be simplified by adopting a so-called *irreflexivity rule*, also known as *Gabbay-style rule*, to characterise the irreflexivity of the accessibility relation underlying a particular modal operator [29, 30]. In the case of the horizontal modalities in the language of **OTL**, a suitable irreflexivity rule may have the following form:

$$\frac{\neg Q \wedge \boxleft Q \wedge \boxright Q \rightarrow A}{A}$$

Here, Q represents a propositional atom that does not occur in the formula A. Intuitively, this rule states that whenever A is a theorem under the assumption that the sibling order (i.e. the relation associated with the box-operators $\boxleft$ and $\boxright$) is irreflexive, then A is in fact a theorem. For more information on the axiomatisation of temporal logics with an irreflexivity rule we refer to Gabbay *et al.* [30].

Recall that proving irreflexivity of the canonical child relation $\prec$ (over generated sets of maximally consistent sets) has not been a problematic issue, because we have only considered maximally consistent sets with a level (see Lemma 4.24). We are therefore doubtful whether an irreflexivity rule for the *vertical* dimension could contribute towards a completeness proof for a system including also the problematic (vertical) descendant modalities.

Axiomatisations with a transitive closure rule. Another option that may be worthwhile exploring would be to add an inference rule to our axiomatic system that explicitly captures the fact that the descendant relation is the transitive closure of the child relation. Such a rule may have the following form:

$$\frac{A \rightarrow \Box^n B \text{ for all } n \in \mathbb{N}}{A \rightarrow \Box^+ B}$$

This rule is similar to both the *rule of infinitary convergence* given by Harel *et al.* [38] and to Goldblatt's *omega-iteration rule* [35], which form part of their respective axiomatisations of first-order dynamic logic.

This inference rule is not a 'standard' rule and the resulting system cannot be called a *finite* axiomatisation anymore. Such an approach is somewhat controversial. Some people may argue that only a finite axiomatisation is a good axiomatisation. While this is certainly a valid point of view, a non-standard axiomatisation can still provide valuable insights into the mechanics of the logic under consideration. If anything, at least it makes precise what the 'missing piece' is in view of a complete finite axiomatisation.

4.5 Summary

In this chapter we have presented a sound axiomatisation for our modal logic of ordered trees that is complete for formulas that have a level (i.e. for formulas that are of the form $\varphi \wedge \bigcirc^n \Box\bot$) and that belong to the fragment of the language excluding the transitive descendant modalities.

Axiomatisation. The *axioms* and *rules* of our system are summarised in Table 4.1 on page 104. (The list of axioms also includes a set of tentative axioms that we would expect to be part of any axiomatisation of full **OTL**, including the descendant operators.) Many of the axioms are variations of well-known formulas that have been used to axiomatise a variety of other modal and temporal logics. The two *interaction axioms* (X1) and (X2), which embody the relationship between the modal operators pertaining to the horizontal and the vertical accessibility relations, respectively, may be regarded as being particularly characteristic to our logic.

Completeness proof. Let us briefly summarise the main ideas underlying our completeness proof. We started out with the definition of a *canonical model*, where worlds are maximally consistent sets of formulas and the accessibility relations are defined in terms of the formulas contained in these sets. Every consistent formula with a level can be extended to a maximally consistent set with a level. To show that consistent formulas are satisfiable, we have used these maximally consistent sets to generate a *submodel* of the canonical model and have shown that it satisfies the formula in question. We have also shown that the frame underlying such a generated submodel must be a *loosely ordered tree*. By our results obtained in the previous chapter, satisfiability with respect to loosely ordered trees is equivalent to satisfiability with respect to ordered trees.

The restriction to formulas *with a level* has served two main purposes. Firstly, the vertical accessibility relation restricted to any submodel of the canonical model generated by a maximally consistent set with a level could easily be shown to be irreflexive. Secondly, the concept of a maximally consistent set with a level allowed us to prove the Ancestor Existence Lemma, i.e. it allowed us to show that the transitive ancestor modalities work as intended within the model used to show completeness. A similar construction for the descendant modalities is not possible, which explains the restriction of our result to the fragment of **OTL** excluding the operators $\Diamond\!\!\!\downarrow^+$ and $\Box\!\!\!\downarrow^+$.

Alternatives. Towards the end of the chapter, we have discussed a number of potential alternatives to our axiomatisation as well as to the method chosen to prove completeness. The standard approach to proving completeness for logics that include an operator pertaining to the transitive closure of the accessibility relation of another operator is to work with finite sets of formulas that are Fischer-Ladner closed, rather than building up an infinite canonical model. While we cannot conclusively exclude the possibility of such an approach being successful also in the case of **OTL**, we have discussed some of the difficulties that we have encountered.

An interesting alternative to our axiomatisation, which we have mentioned only briefly, would be a system that includes one or more irreflexivity rules to characterise the irreflexivity of accessibility relations directly. We would expect the completeness proof for such a system to be simpler than ours, but we do not necessarily expect to solve the problems regarding the transitive descendant modalities in this manner.

Chapter 5

Decidability

Our aim for this chapter is to show that the modal logic of ordered trees is a decidable logic. We are going to prove decidability by showing how our logic can be embedded into another logic that is very expressive but still known to be decidable. This logic is the monadic second-order theory of two successor functions.

In the first section of this chapter we are going to introduce this logic and quote Rabin's famous decidability result. Following that, we are going to approach our decidability proof for ordered tree logics through the study of two special cases. To begin with, we are going to show that **OTL** is decidable, if we restrict the class of admissible frames to finite binary trees.[1] Then we are going to prove that **OTL** restricted to the class of all discretely ordered trees is also a decidable logic. Finally, we are going to prove decidability for the full logic.

5.1 Monadic Second-order Logic

The objective of this first section is to introduce monadic second-order logic over the theory of n successor functions. Our exposition is not intended to be complete and is essentially guided by our needs for proving decidability of ordered tree logics in the later sections. For further information on monadic second-order theories we therefore refer to Rabin [61] and Thomas [67].

Second-order theories. First-order logic (as opposed to propositional logic) allows for quantification over *variables*. On top of that, *second-order logic* also allows for quantification over *predicates*. In *monadic* second-order logic, only *unary* predicate symbols are admitted and consequently, besides simple variables, quantification is restricted to those unary predicates. Semantically, unary predicate symbols represent *sets* (of objects with a certain property), so another way of characterising monadic second-order logic would be to say that it allows for quantification over both simple variables and sets.

[1] The resulting logic has previously been studied (and, amongst other things, shown to be decidable) by Blackburn and Meyer-Viol [9].

The following formula, which we may read as *'Napoleon had all the qualities of a great general'*, would be an example for a monadic second-order formula (taken from [71]):

$$(\forall Q)[(\forall x)[GG(x) \to Q(x)] \to Q(n)]$$

Given that classical first-order logic is undecidable, we could not possibly hope for decidability of second-order logic, at least not in the general case. The situation may change if we restrict the domain of interpretation to very specific structures. An example for such a structure would be the set of all words over a given alphabet.

Words, trees and successor functions. An *alphabet* Σ is simply a countable non-empty set and the elements of such an alphabet are called *letters*. A *word* over an alphabet Σ is a finite sequence of letters in Σ. The set of all words over the alphabet Σ will be denoted by Σ^*. The *empty word* Λ (i.e. the sequence of length 0) is understood to be part of Σ^* for any alphabet Σ. If x is a word and i is a letter, then we write $x.i$ for the word we get by appending i to the end of x.

Take, for instance, the alphabet $\{0,1\}$. Then the set of all words over this alphabet would be $\{0,1\}^* = \{\Lambda, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$, and so on. This set of words can also be interpreted as a (or rather *the*) *infinite binary tree:* Λ represents the root and for any node x, the word $x.0$ represents its lefthand child, while $x.1$ stands for its righthand child. Yet another way of thinking of $\{0,1\}^*$ would be as the set *generated* by two *successor functions*, say, $succ_0$ and $succ_1$: the set of all terms we can construct from these two function symbols and a single constant (corresponding to Λ) is isomorphic to $\{0,1\}^*$. The word 011, for instance, corresponds to the term $succ_1(succ_1(succ_0(\Lambda)))$, which in turn corresponds to the righthand child of the righthand child of the lefthand child of the root node of the infinite binary tree. When working with this kind of structure, at times, we are going to refer to all three of these interpretations (i.e. the set of all words over a two-letter alphabet, the infinite binary tree, and the structure generated by two successor functions).

In fact, $\{0,1\}^*$ is the most important structure we are going to work with in this chapter. The monadic second-order logic over this structure is known as **S2S**, the *monadic second-order theory of two successor functions*. Below, we are going to formally introduce **SnS**, the *monadic second-order theory of n successor functions*, for any given natural number $n \in \mathbb{N}$. Formulas of **SnS** will be interpreted with respect to the structure $\{i \in \mathbb{N}_0 \mid i < n\}^*$, i.e. the infinite tree with the fixed branching factor n. A special case is $\mathbf{S\boldsymbol{\omega}S}$, the *monadic second-order theory of ω successor functions*. Here the underlying structure is the infinite tree where each node has exactly one child for every natural number (including 0). Alternatively, we may characterise this structure as the set of all words over $\mathbb{N}_0$ (taken as an alphabet), which will be denoted as ω^*.

Syntax of terms. We are now going to define the syntax of terms and formulas for the generic logic **SnS**, where n may either stand for a particular natural number or for ω.

We start with the definition of *terms*. The language for terms builds on a fixed countable set of variable names. To distinguish these variables from set variables, which we are going to introduce later on, we shall also refer to the former as *simple variables*.

Definition 5.1 (SnS terms) *Let either $n \in \mathbb{N}$ or $n = \omega$. We inductively define the set of terms in the language of the logic* **SnS** *as the smallest set such that:*

(1) every (simple) variable is a term;

(2) if t is a term, so is $succ_i(t)$ for all $i \in \mathbb{N}_0$ with $i < n$.

In the case of the logic **S$\boldsymbol{\omega}$S**, condition (2) is supposed to express that there is a function symbol $succ_i$ for every number $i \in \mathbb{N}_0$. Later on, we are mostly going to write $x.i$ as a shorthand for $succ_i(x)$; that is, we are going to use the same notation on both the semantic and the syntactic level when describing the process of appending a single letter to the end of a given word.

Syntax of formulas. We now turn to the definition of the syntax of *formulas* for the generic logic **SnS**. Besides the set of simple variable names mentioned earlier we now also require a countable set of *unary predicate symbols*. We use capitals such as P and Q to denote predicate symbols and lowercase letters such as x and y for simple variable names.

Definition 5.2 (SnS formulas) *Let either $n \in \mathbb{N}$ or $n = \omega$. We inductively define the set of well-formed formulas of the logic* **SnS** *as the smallest set such that:*

(1) if P is a unary predicate symbol and t is an **SnS** *term then $P(t)$ is a formula;*

(2) $\top$ and $\bot$ are formulas;

(3) if φ and ψ are formulas, so are $(\neg\varphi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, and $(\varphi \leftrightarrow \psi)$;

(4) if x is a variable and φ is a formula, then $(\exists x)\,\varphi$ and $(\forall x)\,\varphi$ are also formulas;

(5) if P is a unary predicate symbol and φ is a formula, then $(\exists P)\,\varphi$ and $(\forall P)\,\varphi$ are also formulas.

We shall also refer to unary predicate symbols as *set variables*, particularly so in contexts where we quantify over a unary predicate symbol P, as in $(\exists P)\,\varphi$. When writing formulas, we are going to omit brackets whenever it is possible to do so without introducing ambiguities.

Assignments. The structure over which we interpret formulas of a particular second-order theory is fixed. This will always be the appropriate structure of n successor functions. The only thing that can vary is the interpretation of particular variable names and predicate symbols. This aspect of the semantics of **SnS** will be modelled by means of so-called variable *assignments*.

Definition 5.3 (Assignments) *An assignment α is a mapping from variable names to words and from set variable names to sets of words.*

In the case of a logic **SnS** for some $n \in \mathbb{N}$, an assignment α is understood to map simple variable names to words over $\{i \in \mathbb{N}_0 \mid i < n\}$ and unary predicate symbols to subsets of $\{i \in \mathbb{N}_0 \mid i < n\}^*$. In the case of **S$\boldsymbol{\omega}$S**, an assignment α will be a mapping to elements and subsets of ω^*, respectively.

We will often say things such as 'the set P contains the word x' when in fact we *should* be a little more formal and say that the word assigned to the variable x under some assignment α belongs to the set of words assigned to the set variable P under assignment α. Occasionally, we are going to write x^α rather than $\alpha(x)$ and P^α rather than $\alpha(P)$ for simple variables x and set variables P, respectively.

Assignments extend from simple variables to terms in the obvious way. For every $i \in \mathbb{N}_0$, the function symbol $succ_i$ is interpreted as the function that appends the letter i at the end of a given word.

Definition 5.4 (Interpretation of terms) *Let α be an* **SnS** *assignment. The definition of α is inductively extended to* **SnS** *terms t as follows:*

$$\alpha(succ_i(t)) \quad = \quad \alpha(t).i \quad \textit{for } i \in \mathbb{N}_0$$

The next definition provides us with a tool to speak about two assignments that only differ in the interpretation of a single symbol (either a simple variable name or a predicate symbol). This will be required later on to define the semantics of quantification.

Definition 5.5 (Assignment variants) *Let α and α' be assignments.*

(1) Let x be a simple variable. Then α' is called an x-variant of α iff $\alpha(y) = \alpha'(y)$ for all variables y with $y \neq x$.

(2) Let P be a set variable. Then α' is called an P-variant of α iff $\alpha(Q) = \alpha'(Q)$ for all set variables Q with $Q \neq P$.

Truth with respect to an assignment. We are now ready to give a definition of the notion of *truth* for formulas of **SnS**. For each n there is just a single model, namely the tree generated by n successor functions; so truth of a formula only depends on the assignment of its free variables to words and sets of words, respectively. We are going to use the same symbol ($\models$) to denote the truth relation as we have used for **OTL**. Indeed, we are going to use the same symbol for all the various second-order logics, such as **S2S** and **SωS**, we are going to work with in this chapter. Which logic we are referring to will always be clear from the context, however.

Definition 5.6 (Truth of SnS formulas) *We inductively define the notion of an* **SnS** *formula being true under an assignment α as follows:*

(1) $\alpha \models P(t)$ iff $\alpha(t) \in \alpha(P)$ for unary predicate symbols P and terms t;

(2) $\alpha \models \top$;

(3) $\alpha \models \neg\varphi$ iff $\alpha \not\models \varphi$;

(4) $\alpha \models \varphi \wedge \psi$ iff $\alpha \models \varphi$ and $\alpha \models \psi$;

(5) $\alpha \models (\exists x)\,\varphi$ iff $\alpha' \models \varphi$ for some x-variant α' of α;

(6) $\alpha \models (\exists P)\,\varphi$ iff $\alpha' \models \varphi$ for some P-variant α' of α.

Observe that for *closed* formulas φ (i.e. for formulas without free variables of either kind) the assignment α does not matter. Such formulas will be either *true* under any assignment or *false* under any assignment.

Definable operators. The operators not covered by Definition 5.6 are all definable in terms of those for which we have provided a semantics.

Definition 5.7 (Definable SnS operators) *The remaining operators of* **SnS** *are defined as abbreviations on the syntactic level:*

$$\begin{array}{rcl@{\qquad}rcl}
\varphi \vee \psi & = & \neg(\neg\varphi \wedge \neg\psi) & \bot & = & \neg\top \\
\varphi \rightarrow \psi & = & \neg\varphi \vee \psi & (\forall x)\,\varphi & = & \neg(\exists x)\,\neg\varphi \\
\varphi \leftrightarrow \psi & = & (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi) & (\forall P)\,\varphi & = & \neg(\exists P)\,\neg\varphi
\end{array}$$

The semantics of these operators can now be inferred from Definitions 5.6 and 5.7. It is easily observed that, for any assignment α, we get:

(1) $\alpha \not\models \bot$;

(2) $\alpha \models \varphi \vee \psi$ iff $\alpha \models \varphi$ or $\alpha \models \psi$;

(3) $\alpha \models \varphi \rightarrow \psi$ iff $\alpha \models \varphi$ implies $\alpha \models \psi$;

(4) $\alpha \models \varphi \leftrightarrow \psi$ iff either both $\alpha \models \varphi$ and $\alpha \models \psi$ or neither;

(5) $\alpha \models (\forall x)\,\varphi$ iff $\alpha' \models \varphi$ for all x-variants α' of α;

(6) $\alpha \models (\forall P)\,\varphi$ iff $\alpha' \models \varphi$ for all P-variants α' of α.

Validity. Let us now define the concept of *validity* in the context of monadic second-order theories of n successor functions.

Definition 5.8 (Validity in SnS) *An* **SnS** *formula φ is called valid iff $\alpha \models \varphi$ holds for every assignment α mapping variables into the structure of n successor functions.*

If there exists at least one **SnS** assignment α such that $\alpha \models \varphi$ holds, then the formula φ is said to be *satisfiable* in the structure generated by n successor functions. Given that the assignment is irrelevant for formulas without free variables (of either kind), the notions of validity and satisfiability coincide in the case of closed formulas.

As we shall see towards the end of this section, the validity problem for the monadic second-order theory of n successor functions may be reduced to that of the theory of *two* successor functions, for any $n \in \mathbb{N}$ as well as for $n = \omega$.

Decidability. The logic **S2S** was proved to be decidable by Rabin in 1969 [61]; that is, there exists a procedure that, given a closed formula φ in the language of **S2S**, will determine, within a finite amount of time, whether or not φ is valid. This important result is known as *Rabin's Theorem:*

Theorem 5.9 (Decidability of S2S) *The monadic second-order theory of two successor functions is decidable.*

We are not going to prove this theorem here. The original proof is widely regarded as being particularly difficult. A more accessible proof may, for instance, be found in [67] (see also references therein).

Our decidability proof for the modal logic of ordered trees (which we are going to give in Section 5.4), essentially, amounts to a reduction to this result.

Some definable predicates. In Rabin's original formulation, the logic **S2S** also includes two binary predicates, $\leq$ and $\preceq$,[2] with fixed semantics. For two variables x and y, the formula $x \leq y$ is intended to be true iff the word assigned to x is a (not necessarily proper) *prefix* of the word assigned to y, while the formula $x \preceq y$ holds iff the word assigned to x precedes the word assigned to y in the *lexicographic ordering* declared over $\{0,1\}^*$, or the two words are the same. Alternatively, as is well known, these predicates can also be defined within the slightly slimmer incarnation of **S2S** we have introduced here. In order to demonstrate some of the features of this logic, in what follows, we are going to show how to define these and a number of other useful predicates as abbreviations on the syntactic level.

[2]Our use of the symbols $\leq$ and $\preceq$ (and later on also $<$ and $\prec$) in *this* chapter should not be confused with the canonical accessibility relations from the previous chapter.

Equality. We start with a definition of a predicate to check whether two given terms are associated with the same word. This is a simple example where we use the second-order feature of **S2S**, i.e. quantification over sets of words (that is, over unary predicates). Observe that two objects are *equal* iff every set that contains the former also contains the latter and vice versa. Hence, we can give the following definition of equality for two given terms t_1 and t_2:

$$(t_1 = t_2) \quad = \quad (\forall P)[P(t_1) \leftrightarrow P(t_2)]$$

That is, we want the formula $t_1 = t_2$ to evaluate to *true* iff for every unary predicate P the formula $P(t_1)$ holds iff $P(t_2)$ does. This is the case precisely when t_1 and t_2 are being interpreted as the same word.

Subsets. Next we define a subset relation over set variables. This is easily expressible in first-order logic alone:

$$(P \subseteq Q) \quad = \quad (\forall x)[P(x) \rightarrow Q(x)]$$

Given the subset relation, we can now also define the equality relation over sets:

$$(P = Q) \quad = \quad (P \subseteq Q) \wedge (Q \subseteq P)$$

Prefix ordering. Our next objective is to define the prefix relation $\leq$. We first give a formula that is true iff the set variable P appearing in that formula is interpreted as a set that is downward closed with respect to the prefix relation. That is, whenever a particular word belongs to that set then every prefix of that word must belong to the set as well. For any given set variable P the formula $\textsc{prefix-closed}(P)$ defined as follows is true iff P is assigned to a prefix-closed set of words:

$$\textsc{prefix-closed}(P) \quad = \quad (\forall x)(\forall y)[P(x) \wedge (y.0 = x \vee y.1 = x) \rightarrow P(y)]$$

Here, y stands for the immediate prefix of x (that is, we get from y to x by appending either a single 0 or a single 1). So the formula $\textsc{prefix-closed}(P)$ specifies that whenever x belongs to P then so does x's immediate prefix. Hence, by induction, every prefix of x will be in P (possibly in addition to a number of other words, i.e. P is not necessarily the smallest prefix-closed set for a particular word).

We can now define the (reflexive) prefix relation $\leq$ as follows:

$$(x \leq y) \quad = \quad (\forall P)[\textsc{prefix-closed}(P) \wedge P(y) \rightarrow P(x)]$$

The formula on the righthand side is true iff it is the case that *every* prefix-closed set P that contains y will also contain x. In particular, this will be the case for the smallest prefix-closed set P containing y, which contains *only* the prefixes of y and no other words. Hence, the word assigned to x will indeed be a prefix of the word associated with the variable y.

Later on we are sometimes going to write $x < y$ as an abbreviation for the proper prefix relation $(x \leq y) \wedge \neg(x = y)$.

Identifying the root of the tree. A simple application of the prefix ordering $\leq$ is the following definition of a predicate that allows us to identify the root of the infinite binary tree, that is, the empty word Λ:

$$\textsc{root}(x) \quad = \quad (\forall y)(x \leq y)$$

The formula $\textsc{root}(x)$ will be true under an assignment α iff α assigns the empty word Λ to the variable x, because Λ is the only word that is a prefix of *every* word.

Lexicographic ordering. The next example shows that we can also treat the lexicographic ordering $\preceq$ as a defined predicate. A word x precedes another word y in the lexicographic ordering declared over $\{0,1\}^*$ either if x is a prefix of y or x and y are identical up to a certain letter and the next letter in x is 0 while there is a 1 at the corresponding position in y. This notion is captured by the following definition:

$$(x \preceq y) \quad = \quad (x \leq y) \,\vee\, (\exists z)[z.0 \leq x \wedge z.1 \leq y]$$

For example, the word 011010 lexicographically precedes 01110, because the two have the common prefix 011 which is followed by a 0 in the former case and by a 1 in the latter. Again, we are going to write $x \prec y$ as an abbreviation for $(x \preceq y) \wedge \neg(x = y)$.

Finite sets. An interesting application of the lexicographic ordering is that we are now in a position to write a formula that identifies a given set variable as referring to a *finite* set of words. Observe that $\preceq$ induces a total order over the full set of words. We know that a totally ordered set must be finite iff each of its non-empty subsets has both a minimal and a maximal element with respect to the total order in question (see Proposition A.9 in Appendix A). This condition is expressible in second-order logic.

We first define abbreviations for formulas that express that the word assigned to a given variable x is minimal (maximal) within the set assigned to some unary predicate Q with respect to the lexicographic ordering $\preceq$.

$$\begin{array}{rcl} \textsc{min}(x,Q) & = & Q(x) \,\wedge\, (\forall y)[Q(y) \rightarrow x \preceq y] \\ \textsc{max}(x,Q) & = & Q(x) \,\wedge\, (\forall y)[Q(y) \rightarrow y \preceq x] \end{array}$$

Now we are in a position to give a formula that characterises a given unary predicate P as denoting a finite set of words. The set assigned to P is finite iff for every set Q, whenever Q is a subset of P that is not empty then there exists a word x that is minimal in Q and there exists an(other) x that is maximal in Q with respect to the (total) lexicographic ordering $\preceq$ over words. Here is the corresponding formula:

$$\textsc{finite}(P) \quad = \quad (\forall Q)[Q \subseteq P \wedge (\exists x)\, Q(x) \;\rightarrow\; (\exists x)\textsc{min}(x,Q) \wedge (\exists x)\textsc{max}(x,Q)]$$

Having this 'meta-predicate' FINITE available to us, we can augment the language of **S2S** to allow also for the quantification over finite sets of words. For instance, the formula

$(\forall Q)[\text{FINITE}(Q) \to \varphi]$ is true iff φ is true whenever any free appearance of the variable Q within the formula φ is interpreted as a finite set. We introduce the following two abbreviations:

$$\begin{array}{rcl} (\exists_{\text{FIN}} Q)\,\varphi &=& (\exists Q)[\text{FINITE}(Q) \wedge \varphi] \\ (\forall_{\text{FIN}} Q)\,\varphi &=& (\forall Q)[\text{FINITE}(Q) \to \varphi] \end{array}$$

Guarded quantification. We are sometimes going to have to restrict quantification over simple variables to words belonging to a particular set and quantification over set variables to sets that are subsets of particular sets of words. To this end, we introduce the following shorthands for guarded quantification:

$$\begin{array}{rclcrcl} (\exists x \in Q)\,\varphi &=& (\exists x)[Q(x) \wedge \varphi] & \quad & (\exists P \subseteq Q)\,\varphi &=& (\exists P)[P \subseteq Q \wedge \varphi] \\ (\forall x \in Q)\,\varphi &=& (\forall x)[Q(x) \to \varphi] & \quad & (\forall P \subseteq Q)\,\varphi &=& (\forall P)[P \subseteq Q \to \varphi] \end{array}$$

Relations defined by formulas. It is important to keep in mind that our defined 'predicates' such as ROOT or $\leq$ are in fact nothing but syntactic abbreviations of certain **S2S** formulas. That is, even though, say, $\leq$ is (or has the appearance of being) a binary predicate, we are still working in the same monadic logic as before. Each defined predicate that does not involve any free set variables determines a relation over the structure generated by two successor functions. As we have seen earlier, the predicate $\leq$, for example, determines the prefix relation over words in $\{0,1\}^*$. We are now going to introduce notation that will allow us to speak about these relations directly.

Let $\text{PRED}(x_1, \ldots, x_k)$ be an **S2S** formula with k free variables $x_1, \ldots, x_k$ (and no free set variables). On the semantic level, this predicate corresponds to a k-ary relation over words in $\{0,1\}^*$. This relation, which we are going to denote by $[\![\text{PRED}]\!]$, is defined as follows:

$$\begin{array}{rcl} [\![\text{PRED}]\!] &=& \{(w_1, \ldots, w_k) \in (\{0,1\}^*)^k \mid \alpha \models \text{PRED}(x_1, \ldots, x_k) \text{ for all } \alpha \\ && \qquad\qquad\qquad\qquad \text{with } \alpha(x_1) = w_1, \ldots, \alpha(x_k) = w_k\} \end{array}$$

Here, α is understood to range over all **S2S** variable assignments. It follows from our earlier discussion of the definition of the respective predicates that we have, for example, $[\![\text{ROOT}]\!] = \{\Lambda\}$ and $[\![<]\!] = \{(\Lambda,0), (\Lambda,1), (\Lambda,00), (0,00), (\Lambda,01), (0,01), \ldots\}$.

We are going to use the same kind of notation for the semantics of defined predicates also for **SnS** with $n \neq 2$ (simply substitute $(\{0,1\}^*)^k$ by $(\{i \in \mathbb{N}_0 \mid i < n\}^*)^k$ or $(\omega^*)^k$, respectively, and let α range over variable assignments for the logic in question).

Encoding theories with more than two successor functions. We shall conclude our introduction to monadic second-order logic by showing how second-order theories with more than two successor functions, and particularly **S$\boldsymbol{\omega}$S**, can be embedded into the logic **S2S**. This is important, because it shows that, by Rabin's Theorem, all of these logics are decidable and we can pick the one that seems most convenient for our own decidability proofs later on. We start with an encoding of **S$\boldsymbol{\omega}$S** into **S2S**. The result for the other logics will then follow by adding a simple additional restriction.

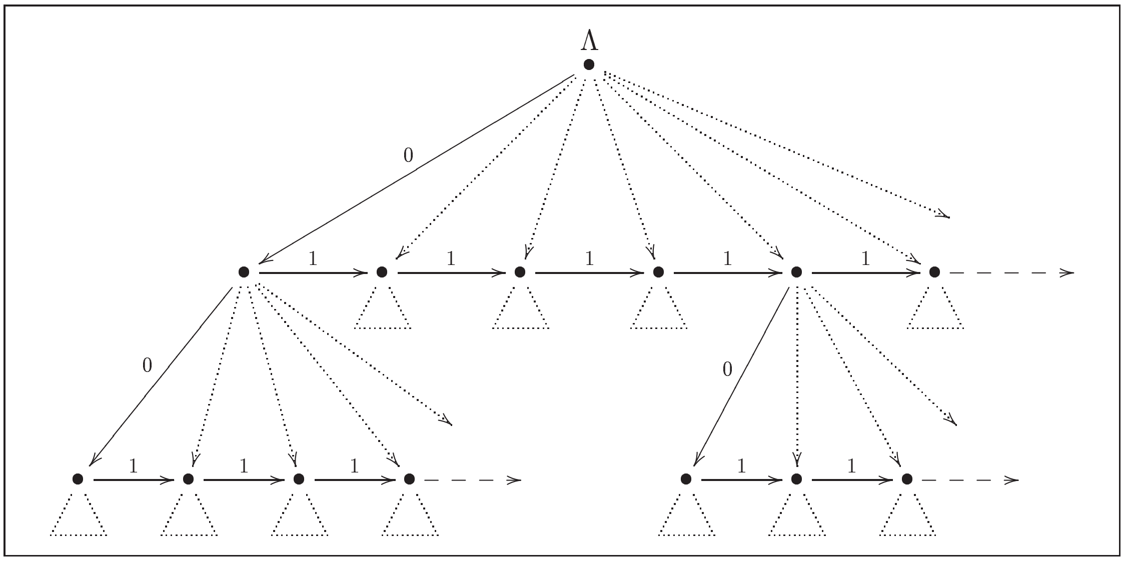

Figure 5.1: Encoding **SωS** in **S2S**

The basic idea underlying this embedding is to use one of the two successor functions of **S2S** to represent arbitrary natural numbers in a unary format and to use the other as a separator between these unary numbers. For instance, the word 01111101111011 (an element of $\{0, 1\}^*$) may be interpreted as encoding the word 542 (an element of ω^*); 0001111111 would stand for 007, and so on. If we think of $\{0, 1\}^*$ as the infinite binary tree and of ω^* as the infinite tree where every node has one child for every number in $\mathbb{N}_0$, then this observation shows that we can identify the latter as a kind of substructure of the former. Figure 5.1 shows how. To reach the nth child of a given node in the tree ω^*, we first use $succ_0$, the first successor function available in **S2S**, to go down to the leftmost child of a given node, and then $succ_1$, the other **S2S** successor function, to move n steps to the right.

Formally, we can define a mapping f from ω^* to $\{0, 1\}^*$ as follows:

$$\begin{array}{rcl} f(\Lambda) & = & \Lambda \\ f(x.i) & = & f(x).0.1^i \quad \text{for } i \in \mathbb{N}_0 \end{array}$$

This mapping is not surjective. In fact, inspection of the definition of f shows that we are mapping the set of words ω^* onto the set of those words in $\{0, 1\}^*$ that do not start with a 1 (equivalently, we are mapping onto the set of words starting with 0 together with the empty word Λ). This set of words can be characterised by an **S2S** formula:

$$\text{FULLTREE}(x) \quad = \quad \neg(\exists y)[\text{ROOT}(y) \wedge y.1 \leq x]$$

The formula FULLTREE(x) will be satisfied under an assignment α iff α maps x to a word that does not start with a 1, i.e. a word that is in the range of the function f.

The language of **SωS** has one successor function for each natural number (including 0). To distinguish the functions symbols of the two logics, we are going to write $succ_i^\omega$

with $i \in \mathbb{N}_0$ for successor functions of $\mathbf{S}\boldsymbol{\omega}\mathbf{S}$ and reserve the original symbols, namely $succ_0$ and $succ_1$, for **S2S**. To be able to translate terms in the language of $\mathbf{S}\boldsymbol{\omega}\mathbf{S}$ into terms in the language of **S2S** we have to show how to represent the function symbols of $\mathbf{S}\boldsymbol{\omega}\mathbf{S}$ within **S2S** in a way that corresponds to the mapping f between the two infinite trees. Here is a simple recursive definition of the translation function:

$$\begin{aligned} succ_0^{\omega}(x) &= succ_0(x) \\ succ_i^{\omega}(x) &= succ_1(succ_{i-1}^{\omega}(x)) \quad \text{for } i \in \mathbb{N} \end{aligned}$$

That is, the $\mathbf{S}\boldsymbol{\omega}\mathbf{S}$ term $succ_3^{\omega}(x)$, for example, would be translated into the **S2S** term $succ_1(succ_1(succ_1(succ_0(x))))$. In our shorthand notation, this means that $x.3$ would be represented as $x.0.1.1.1$.

We are now in a position to describe the process of translating a (closed) $\mathbf{S}\boldsymbol{\omega}\mathbf{S}$ formula φ into an **S2S** formula φ^* in such a way that φ is valid in $\mathbf{S}\boldsymbol{\omega}\mathbf{S}$ iff φ^* is valid in **S2S**. The first step is to rewrite $\mathbf{S}\boldsymbol{\omega}\mathbf{S}$ terms as terms using only the two successor functions of **S2S**. The next step is to ensure that variables in the translated formula only range over those words (or set of words in the case of set variables) of $\{0,1\}^*$ that are in the range of the mapping f from $\{0,1\}^*$ to ω^*. To this end we introduce a special set variable T (which is different from any of the set variables used in φ) that is intended to be assigned to precisely that set of words. This assignment can be enforced by the formula $(\forall x)[\textsc{fulltree}(x) \leftrightarrow T(x)]$. Furthermore, we replace every quantification in φ by a quantification guarded by T. That is, we rewrite $(\exists x)$ as $(\exists x \in T)$ and $\exists P$ as $(\exists P \subseteq T)$, and accordingly for universal quantifiers.

Let φ_{01}^T be the **S2S** formula we get by translating all function symbols in φ by the appropriate combinations of $succ_0$ and $succ_1$ as well as replacing every quantification with a quantification guarded by the set variable T. Then the formula φ^*, defined as follows, will be valid in **S2S** iff our original formula φ is valid in $\mathbf{S}\boldsymbol{\omega}\mathbf{S}$:

$$\varphi^* \quad = \quad (\forall T)[(\forall x)[\textsc{fulltree}(x) \leftrightarrow T(x)] \rightarrow \varphi_{01}^T]$$

Hence, by Rabin's Theorem, $\mathbf{S}\boldsymbol{\omega}\mathbf{S}$ must be a decidable logic.

Example. Consider, for example, the $\mathbf{S}\boldsymbol{\omega}\mathbf{S}$ formula $(\exists P)(\forall x)[P(x) \rightarrow P(x.3)]$. This formula is valid iff there exists a set of words that is closed under appending a 3 at the end of a word. This is clearly the case; the infinite set $\{\Lambda, 3, 33, 333, 3333, \ldots\}$, for instance, would be a candidate. To translate the formula we first have to fix a name, say T, for the set variable corresponding to the set of words in the range of the translation function f. Translating the formula itself then yields the following result: $(\exists P \subseteq T)(\forall x \in T)[P(x) \rightarrow P(x.0.1.1.1)]$. Before we can check validity we have to ensure that T will be interpreted as intended, namely as the set of all words that correspond to words in ω^*. In summary, our original formula $(\exists P)(\forall x)[P(x) \rightarrow P(x.3)]$ is valid in the logic $\mathbf{S}\boldsymbol{\omega}\mathbf{S}$ iff the following formula is valid in **S2S**:

$$(\forall T)[(\forall x)[\textsc{fulltree}(x) \leftrightarrow T(x)] \rightarrow (\exists P \subseteq T)(\forall x \in T)[P(x) \rightarrow P(x.0.1.1.1)]]$$

Prefix and lexicographic ordering for SωS. The definitions of both the prefix and the lexicographic orderings over words given earlier in this sections were specific to **S2S** and analogous definitions for **SωS** are not possible, because we now have an infinite number of successor functions to consider. However, it *is* possible to express these relations within the embedding of **SωS** into **S2S** just described.

Let a and b be two words in ω^* and let $f(a)$ and $f(b)$ be their translations into $\{0,1\}^*$. If a is a proper prefix of b then this will manifest itself in the words $f(a)$ and $f(b)$ as follows. The word $f(b)$ will have $f(a)$ as a prefix and that prefix will be followed by the translation of at least one more number, that is, the string $f(a)$ will at least be followed by a 0 within the word $f(b)$. The converse holds as well, that is, the formula $x.0 \leq y$ characterises the proper prefix ordering for the embedding of **SωS** into **S2S**. We can thus use the formula $(x.0 \leq y) \vee (x = y)$ to rewrite the usual (reflexive) prefix ordering of **SωS** when translating formulas into **S2S**.

Expressing the lexicographic ordering for **SωS** is even simpler. In fact, as is readily verified, the lexicographic ordering $\preceq$ of **S2S** acts also as a lexicographic ordering for **SωS** (if we use the embedding into **S2S** presented above). That is, no specific translation of the defined predicate $\preceq$ is necessary.

Encoding theories with finitely many successor functions. The embedding of **SωS** into **S2S** can easily be adapted in such a way as to serve as an embedding of **SnS** into **S2S** for any finite $n \in \mathbb{N}$. All that is required is an additional formula that further restricts the substructure of $\{0,1\}^*$ used for the actual embedding to ensure that no substrings corresponding to numbers greater or equal to n can appear in the translation of an **SnS** word. To encode, for instance, the logic **S4S** we simply add the formula $\neg(\exists y)(y.1.1.1.1 \leq x)$ to the definition of the predicate $\textsc{fulltree}(x)$.

It follows that **SnS** must be a decidable logic for any finite number of successor functions $n \in \mathbb{N}$.

Defined predicates again. Observe that all the defined predicates discussed earlier are available in all our second-order logics. Whenever a definition does not make any reference to successor functions at all (such as our definition of equality over words, for instance), we can use the same formula for every logic. In cases where the successor functions are required (as in the definition of $\textsc{prefix-closed}$), the definition can easily be adapted for logics **SnS** with a *finite* number n of successor functions.

However, this would not be possible in the case of **SωS**. Still, as we have seen earlier, the critical predicates, namely the prefix ordering $\leq$ and the lexicographic ordering $\preceq$ for **SωS** can both be translated into formulas in **S2S**, the logic we have used to embed **SωS** into. So indeed, *all* of the defined predicates we have seen in this section are available in *all* of the second-order theories we have been discussing.

Applications in modal logic. It turns out that a variety of modal logics can be embedded into the monadic second-order theory of two successor functions. Given Rabin's

Theorem, such an embedding would then show that the modal logic in question must be decidable. A number of early decidability results for modal logics using this method were obtained by Gabbay [27, 28].

We briefly outline some of the basic intuitions underlying this approach. For modal logics with a single fixed frame such as, for instance, the temporal logic over the rational numbers, the first crucial step for an embedding into **S2S** is a mapping from the set of worlds in the modal frame to a suitable substructure of $\{0,1\}^*$ (similar to the mapping from ω^* to $\{0,1\}^*$ discussed earlier). For modal logics defined with respect to a *class* of frames (rather than a single frame) we also have to show that we can capture any member of this class by means of a set variable ranging over all possible sets of words that could make up a frame of this logic. We also have to show that we can define predicates that correspond to the accessibility relations of such frames. Finally, we have to give a translation from modal formulas to second-order formulas. This will be very similar to the standard translation of modal logic formulas into classical first-order logic and the translated formulas will directly refer to the predicates representing the accessibility relations. Propositional letters of the modal language will be translated into unary predicates. We can use quantification over these predicates to simulate the range of all possible valuations.

Broadly speaking, this is the path we are going to follow for our decidability proof for **OTL**. Of course, instead of embedding a modal logic directly into **S2S**, we may also choose to define an embedding into either **S$\boldsymbol{\omega}$S** or **SnS** for some finite $n \in \mathbb{N}$. Decidability then follows from the fact that these second-order logics can themselves be embedded into **S2S** (as shown in this section).

Plan for the rest of this chapter. The following two sections may be considered case studies, where we use first **S2S** and then **S$\boldsymbol{\omega}$S** to prove decidability for the validity problem of **OTL** with respect to finite binary trees and discretely ordered trees, respectively. Our main decidability result, for **OTL** in general, will then be proved in Section 5.4 via a reduction to **S4S**.

5.2 Finite Binary Trees

In this section we are going to show how we can prove that the validity problem for **OTL** formulas with respect to *finite binary trees* is decidable via a reduction to Rabin's Theorem. In fact, such a reduction is almost straightforward. Essentially, what is required are a translation from **OTL** formulas into **S2S** and a means of restricting reference from the infinite binary tree underlying **S2S** to arbitrary (finite) initial subtrees.

Modal logics of finite binary trees. In [9], Blackburn and Meyer-Viol introduce a modal logic of finite binary trees, demonstrate its relevance to applications in the area of computational linguistics, prove completeness of a finite axiomatisation, and show that their logic is decidable. Binary trees may be considered special cases of ordered trees. If every internal node is known to have exactly two children, then we can always identify the left and the right child of a node by reference to the order declared over siblings. Towards the end of the section on *correspondences* in Chapter 3, we have indicated how to express the modal operators available in the logic of Blackburn and Meyer-Viol in terms of **OTL** operators, assuming that the class of admissible frames is restricted to binary trees.

The following definition identifies the class of finite binary trees as a subclass of the class of all ordered trees (as specified in Definition 3.4).

Definition 5.10 (Finite binary trees) *An ordered tree $\mathcal{T} = (T, R, S)$ is called a finite binary tree iff the set of nodes T is finite and the set of siblings $[t]_{R^{-1} \circ R}$ is a set with exactly two elements for every node $t \in T$ except the root.*

Defining the accessibility relations. The structure $\{0,1\}^*$ underlying the logic **S2S** is a binary tree and thereby also an ordered tree. A first requirement for being able to translate modal **OTL** formulas into **S2S** formulas is to show that the accessibility relations relevant to the semantics of **OTL** can be expressed in terms of **S2S**. These accessibility relations are the descendant relation, the child relation, the sibling relation, the neighbour relation, and their inverses. Given that we are going to use the tree $\{0,1\}^*$ *itself*, rather than using it to encode a differently structured tree first, expressing these relations is easy. For instance, the descendant relation directly corresponds to the (proper) prefix relation introduced earlier:

$$\textsc{descendant}(x, y) \quad = \quad (x < y)$$

That is, the word represented by the variable x has as a descendant in the infinite binary tree $\{0,1\}^*$ the word represented by y iff the former is a proper prefix of the latter. The ancestor relation is the inverse of the descendant relation. That is, the formula $\textsc{descendant}(y, x)$ is true iff y represents an ancestor of the word represented by x in the infinite binary tree. We could use the predicate $\textsc{descendant}$ to define a predicate for the child relation, namely by stipulating that y is the child of x iff y is a descendant of x and there is no other node that is both a descendant of x and an ancestor of y. Alternatively, we may also give a direct definition as follows:

$$\textsc{child}(x, y) \quad = \quad (x.0 = y) \vee (x.1 = y)$$

This definition is based on the fact that if appending either a 0 or a 1 to the end of the word represented by x yields the word referred to by y then y represents the child of the word represented by x in the infinite binary tree. Again, we may also use the predicate $\textsc{child}$ to refer to the parent of a given node, because the two accessibility relations are inverse to each other. It follows from our definitions that the predicate $\textsc{descendant}$ must be interpreted as the transitive closure of the relation corresponding to the predicate

CHILD, because the prefix relation *is* the transitive closure of the immediate prefix relation encoded by CHILD.

Next we turn to the definition of an accessibility relation to refer to right- and lefthand siblings. Two words represent sibling nodes iff they only differ in their final letter. The word ending in 0 is the lefthand sibling; the one ending in 1 is the righthand sibling. That is, a sibling relation may be defined as follows:

$$\textsc{sibling}(x, y) \quad = \quad (\exists z)(z.0 = x \wedge z.1 = y)$$

In the case of binary trees, the notions of sibling and neighbour coincide; thus:

$$\textsc{neighbour}(x, y) \quad = \quad \textsc{sibling}(x, y)$$

Characterising initial subtrees. The infinite binary tree generated by two successor functions is a particular ordered tree. The set of nodes is given by the set of words $\{0, 1\}^*$. The child relation R of an ordered tree can be represented in terms of the CHILD predicate and the sibling relation S corresponds to the SIBLING predicate. We would now be in a position to prove decidability of the validity problem of **OTL** for the particular frame given by the infinite binary tree. As we are interested in validity over the class of *all* (finite) binary trees, we first have to put forward suitable conditions that allow us to decide for any given subset of $\{0, 1\}^*$ whether that set represents itself a binary tree or not.

Observe that any given binary tree is isomorphic to an *initial binary subtree* of the infinite binary tree. By an initial subtree we mean any subset of $\{0, 1\}^*$ that includes the root of the infinite tree (i.e. the empty word Λ) such that, whenever a node belongs to this set, so does its parent. We furthermore call an initial subtree *binary* iff it contains either both or none of the two children of any given node. Using the defined predicates encoding the accessibility relations together with the predicate ROOT defined in the previous section, we can express these properties as a formula with a free set variable T corresponding to the set of nodes that make up the respective initial binary subtree. We abbreviate this formula as TREE(T):

$$\begin{array}{rcl} \textsc{tree}(T) & = & (\forall x)[\textsc{root}(x) \rightarrow T(x)] \wedge \\ & & (\forall x)(\forall y)[\textsc{child}(x, y) \wedge T(y) \rightarrow T(x)] \wedge \\ & & (\forall x)(\forall y)[\textsc{sibling}(x, y) \rightarrow (T(x) \leftrightarrow T(y))] \end{array}$$

Now, if α is an **S2S** variable assignment such that $\alpha \models \textsc{tree}(T)$ holds, then the triple $(T^\alpha, [\![\textsc{child}]\!], [\![\textsc{sibling}]\!])$ is a binary ordered tree.[3] Also, for every binary tree there exists an assignment α such that $(T^\alpha, [\![\textsc{child}]\!], [\![\textsc{sibling}]\!])$ is isomorphic to that binary tree (for some set variable T with $\alpha \models \textsc{tree}(T)$). Furthermore, the relations induced on these trees by the predicates DESCENDANT and NEIGHBOUR work as intended: $[\![\textsc{descendant}]\!]$ is the transitive closure of $[\![\textsc{child}]\!]$ and $[\![\textsc{neighbour}]\!]$ is the immediate successor relation underlying $[\![\textsc{sibling}]\!]$.

[3] Of course, the relations $[\![\textsc{child}]\!]$ and $[\![\textsc{sibling}]\!]$ range over the entire infinite binary tree. If we wish restrict them explicitly to T^α, we may write $(T^\alpha, [\![\textsc{child}]\!] \cap T^\alpha \times T^\alpha, [\![\textsc{sibling}]\!] \cap T^\alpha \times T^\alpha)$ instead.

THE TRANSLATION FUNCTION TR_x^T

$$
\begin{array}{rcl}
\mathrm{TR}_x^T(P) & = & P'(x) \quad \text{for propositional letters } P \\
\mathrm{TR}_x^T(\top) & = & \top \\
\mathrm{TR}_x^T(\neg\varphi) & = & \neg\mathrm{TR}_x^T(\varphi) \\
\mathrm{TR}_x^T(\varphi\wedge\psi) & = & \mathrm{TR}_x^T(\varphi)\wedge\mathrm{TR}_x^T(\psi) \\
\mathrm{TR}_x^T(\bigcirc_{\uparrow}\varphi) & = & (\exists y\in T)[\mathrm{CHILD}(y,x)\wedge\mathrm{TR}_y^T(\varphi)] \\
\mathrm{TR}_x^T(\Diamond_{\uparrow}\varphi) & = & (\exists y\in T)[\mathrm{DESCENDANT}(y,x)\wedge\mathrm{TR}_y^T(\varphi)] \\
\mathrm{TR}_x^T(\bigcirc_{\leftarrow}\varphi) & = & (\exists y\in T)[\mathrm{NEIGHBOUR}(y,x)\wedge\mathrm{TR}_y^T(\varphi)] \\
\mathrm{TR}_x^T(\Diamond_{\leftarrow}\varphi) & = & (\exists y\in T)[\mathrm{SIBLING}(y,x)\wedge\mathrm{TR}_y^T(\varphi)] \\
\mathrm{TR}_x^T(\bigcirc_{\rightarrow}\varphi) & = & (\exists y\in T)[\mathrm{NEIGHBOUR}(x,y)\wedge\mathrm{TR}_y^T(\varphi)] \\
\mathrm{TR}_x^T(\Diamond_{\rightarrow}\varphi) & = & (\exists y\in T)[\mathrm{SIBLING}(x,y)\wedge\mathrm{TR}_y^T(\varphi)] \\
\mathrm{TR}_x^T(\Diamond_{\downarrow}\varphi) & = & (\exists y\in T)[\mathrm{CHILD}(x,y)\wedge\mathrm{TR}_y^T(\varphi)] \\
\mathrm{TR}_x^T(\Diamond_{\downarrow}^{+}\varphi) & = & (\exists y\in T)[\mathrm{DESCENDANT}(x,y)\wedge\mathrm{TR}_y^T(\varphi)]
\end{array}
$$

P': set variable(s), encoding valuation for propositional letter(s) P

T: set variable, encoding modal frame

x: simple variable, encoding world in frame

y: simple variable, fresh for each subformula

Table 5.1: Translating **OTL** formulas into monadic second-order logic

Translating modal formulas. We now turn to the actual translation of **OTL** formulas into formulas of monadic second-order logic. This translation is going to be similar to the usual standard translation of modal logics into classical first-order logic.[4] For the standard translation, we require a binary first-order predicate for each of the accessibility relations referred to in the definition of the semantics of the respective modal logic. Here, this role will be filled by our previously defined predicates characterising the descendant, child, sibling, and neighbour relations in an ordered tree. As for the standard translation, every translated formula will have a single free variable x, which may be thought of as ranging over the nodes in a tree. Furthermore, quantification in the translated formulas will be guarded by a set variable T intended to represent the set of words used for the encoding of a particular ordered tree as a substructure of the infinite binary tree.

Our translation function TR_x^T is defined in Table 5.1. The x in the subscript fixes the name of the simple free variable in the resulting second-order formula and the T is the name of the free set variable guarding quantification in the same formula. Any propositional letter P appearing in an **OTL** formula φ to be translated will be represented by a unary predicate P' in the second-order formula $\mathrm{TR}_x^T(\varphi)$.

We are going to use the same translation function in the following two sections, where we are going to describe an encoding of discretely ordered trees into **SωS** and an encoding of general ordered trees into **S4S**, respectively. Only the definition of predicates, such as DESCENDANT, encoding the accessibility relations is going to change, depending on the particular construction.

[4]The standard translation for the basic modal logic has briefly been discussed in the introductory Chapter 2. The standard translation from our logic **OTL** into classical first-order logic will also be defined in Chapter 6.

Decidability for finite binary trees. We are now in a position to prove decidability of the validity problem for the modal logic of finite binary trees. To be able to phrase this result as a specific decidability result for **OTL** we introduce the notion of **OTL** validity in finite binary trees.

Definition 5.11 (Validity in finite binary trees) *A formula φ is called valid in finite binary trees iff $\mathcal{T} \models \varphi$ holds for every finite binary tree $\mathcal{T}$.*

Our first Decidability Theorem now states that for any given **OTL** formula the problem of checking whether that formula is valid over the class of finite binary trees is a decidable problem.

Theorem 5.12 (Decidability for finite binary trees) *The problem of checking validity in finite binary trees is decidable.*

Proof. Let φ be an **OTL** formula and let $P_1, \ldots, P_n$ be the propositional letters appearing in φ. Consider the **S2S** formula φ^* defined as follows:

$$\varphi^* \quad = \quad (\forall_{\text{FIN}} T)[\textsc{tree}(T) \rightarrow (\forall P_1' \subseteq T) \cdots (\forall P_n' \subseteq T)(\forall x \in T)\, \textsc{tr}_x^T(\varphi)]$$

Here, $\textsc{tr}_x^T$ is the translation function given in Table 5.1 and the meta-predicate TREE as well as the binary predicates DESCENDANT, CHILD, SIBLING, and NEIGHBOUR (the latter four of which feature in the definition of $\textsc{tr}_x^T$) are understood to be the respective predicates as previously defined in *this* section.[5] The predicates $P_1', \ldots, P_n'$ are the unary predicates we obtain when we translate the propositional letters $P_1, \ldots, P_n$.

The formula φ^* expresses that, for every finite subset of $\{0,1\}^*$ that is an initial binary subtree of the infinite binary tree, we can choose the assignments to the predicate symbols P_1' to P_n' so as to refer to any subsets of that finite tree, and the translation of φ will be true on that particular structure. This is the case precisely iff φ is valid with respect to all finite binary trees. Hence, checking validity of φ (with respect to finite binary trees) is equivalent to checking validity of φ^* in **S2S**. By Rabin's Theorem, the latter problem is decidable, so checking validity for finite binary trees must be a decidable problem as well. □

Our discussion at the beginning of this section shows that Theorem 5.12 confirms one of the results of Blackburn and Meyer-Viol [9], namely that their modal logic of finite binary trees is decidable.

5.3 Discretely Ordered Trees

Our aim for this section is to improve on Theorem 5.12 and show that our logic remains decidable when we extend the class of frames under consideration from binary trees to

[5] In later sections, we are going to give different definitions for these four predicates.

discretely ordered trees in general. Recall that we call an ordered tree a discretely ordered tree iff the children of each node (together with the sibling relation S) form a strongly discrete order.[6] A formal definition was given in Chapter 3 (Definition 3.12).

We are going to reduce the decidability problem for **OTL** restricted to discretely ordered trees to the decidability problem for **SωS**. The latter is known to be decidable. This was already shown in Rabin's original paper [61]. It also follows from our discussion in Section 5.1, where we have shown that **SωS** can in fact be embedded into **S2S**. All second-order formulas in this section are understood to be **SωS** formulas and any variable assignments α map into the structure of ω successor functions.

Immediate prefixes. In the following construction, we will often have to refer to the immediate prefix of a given word. In **S2S**, such a predecessor relation can be defined in terms of the two successor functions available in the language (the CHILD predicate from the previous section is an example). In **SωS**, on the other hand, this is no longer possible, because we now have an infinite number of successor functions to consider. Instead, we can define a predecessor predicate via the (irreflexive) prefix relation $<$ as follows:

$$\textsc{prec}(x, y) \quad = \quad (x < y) \wedge \neg(\exists z)[x < z \wedge z < y]$$

By this definition, $\textsc{prec}(x, y)$ is a true formula iff the variable x represents the word over ω^* we obtain by removing the final letter from the (non-empty) word assigned to y.

Encoding the infinite unbounded discretely ordered tree. The tree ω^* is an infinite tree where every node has a child for every natural number. That is, branching is infinite and discrete, but bounded to the left. The crucial step in our construction will be to show how we can represent an infinite tree where branching is discrete and *unbounded*. We will then be able to describe the set of all discretely ordered trees as the set of all initial subtrees of that infinite tree.

The central idea underlying our construction is to model infinite branching to both the left and right by using only words ending in either 0 and 1 and interpreting the former as children to the left of the centre and the latter as children to the right of it. More specifically, we only consider those words in ω^* that are of of even length and where every second letter is either a 0 or a 1 (including the empty word Λ). Given a particular word, we obtain its parent in the new tree by removing the final *two* letters. For example, 308151 is the 5th righthand child of the word 3081. The latter, in turn, is the 8th righthand child of the node represented by 30, which again is the 3rd lefthand child of the root Λ. This embedding of the infinite unbounded discretely ordered tree into the tree underlying **SωS** is also sketched in Figure 5.2. Here, every node is labelled with the two additional letters to be appended to the word representing its parent. For instance, the last node shown at the lower righthand corner of the picture corresponds to the word 1121, the 2nd righthand child of the 1st righthand child of the root. (It is also

[6]The term *strongly discrete order* is defined in Appendix A.

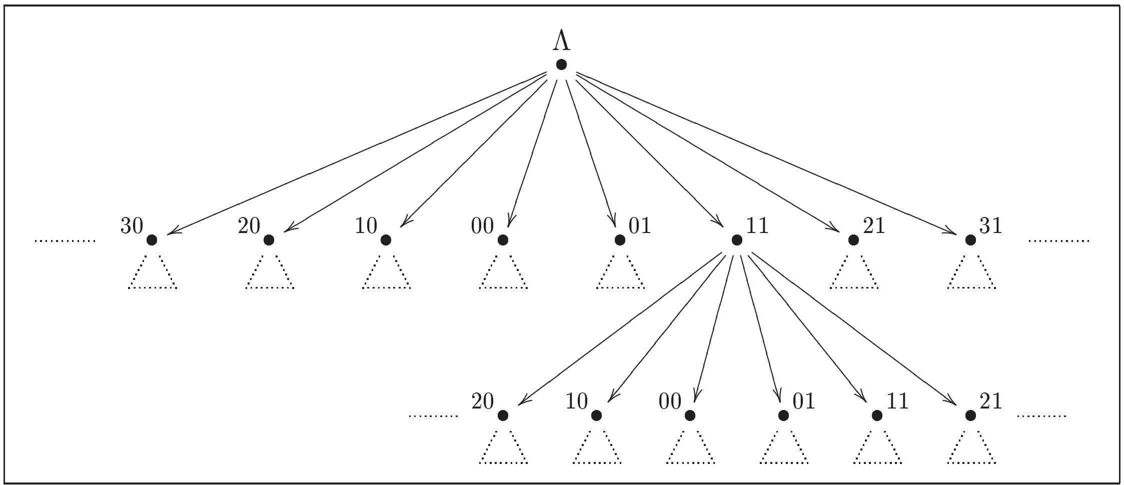

Figure 5.2: Encoding the infinite unbounded discretely ordered tree in $\mathbf{S\omega S}$

the righthand neighbour of the righthand neighbour of 1101, the '0th' righthand child of the word 11.)

We now turn to the definition of the various $\mathbf{S\omega S}$ predicates required to describe the suggested embedding. The following two auxiliary predicates may be used to check whether a given variable x represents a word ending in 0 or 1, respectively:

$$\textsc{left}(x) \;=\; (\exists y)(y.0 = x) \qquad \textsc{right}(x) \;=\; (\exists y)(y.1 = x)$$

So $\textsc{left}(x)$, for instance, will be true iff x represents a node that is one of the lefthand children of its parent (provided it represents one of the nodes used for the embedding in the first place).

Every word in the set of words taken into account for our construction has to end in either 0 or 1, or it has to be the empty word Λ. We can use the following meta-predicate to check whether a given set (assigned to a set variable P) fulfils this condition:

$$\textsc{zero-one}(P) \;=\; (\forall x)[P(x) \rightarrow \textsc{root}(x) \vee \textsc{left}(x) \vee \textsc{right}(x)]$$

Another requirement for the subset of ω^* representing the unbounded infinite tree is that it contains the empty word Λ. We can reuse the predicate $\textsc{root}$ to specify a meta-predicate for set variables P that evaluates to true iff the corresponding set contains the empty word:

$$\textsc{rooted}(P) \;=\; (\exists x)[\textsc{root}(x) \wedge P(x)]$$

Another property of the set of words used to encode the unbounded tree is that, whenever that set contains a given word, it will also contain the predecessor of the predecessor of that word. In other words, the set is closed under removing the last two letters of a word. This property of a set is captured by our next meta-predicate:

$$\textsc{gap-closed}(P) \;=\; (\forall x)(\forall y)(\forall z)[\textsc{prec}(x,y) \wedge \textsc{prec}(y,z) \wedge P(z) \rightarrow P(x)]$$

We are now going to put together the various auxiliary predicates defined earlier and present a predicate that defines the set of words in ω^* that are used for our encoding of the unbounded discretely ordered tree indicated in Figure 5.2. How can we characterise the set of words that make up this substructure of ω^*? Suppose we take a single word x and use it to generate a set of words P as follows: x is in P and for any word in P also the word we get by removing the last two letters is in P (provided that word has at least two letters). This latter operation can be implemented by means of our defined predicate GAP-CLOSED. Then the generated set P shows whether or not x is one of the words we are looking for. If it is, then every word in P has to end either in a 0 or a 1 (this can be implemented using ZERO-ONE) and the empty word Λ has to be part of P (that is, P has to be ROOTED). The latter guarantees that the original word x must have been of even length. In summary, the following predicate FULLTREE determines the set of words required for our construction:

$$\textsc{fulltree}(x) \quad = \quad (\forall P)[P(x) \wedge \textsc{gap-closed}(P) \rightarrow \textsc{rooted}(P) \wedge \textsc{zero-one}(P)]$$

The universal quantification over P guarantees that $\textsc{rooted}(P) \wedge \textsc{zero-one}(P)$ holds, in particular, for the smallest set P that satisfies $P(x) \wedge \textsc{gap-closed}(P)$.

Defining the accessibility relations. If both x and y represent words that belong to the full tree then the latter is a descendant of the former iff it is one of its prefixes. In other words, the prefix relation of ω^* restricted to the set of words used to represent the unbounded discretely ordered tree coincides with the descendant relation declared over the latter. Hence, the following is a suitable definition of the DESCENDANT predicate for this construction:

$$\textsc{descendant}(x, y) \quad = \quad (x < y)$$

We have again more than one option of how to define the child relation. One of them would be to exploit the fact that we get from a given word to the word representing its parent by removing the last two letters, that is, the parent of a node is represented by the immediate prefix of the immediate prefix of the word representing the node in question. Here is the corresponding definition of our new CHILD predicate:

$$\textsc{child}(x, y) \quad = \quad (\exists z)[\textsc{prec}(x, z) \wedge \textsc{prec}(z, y)]$$

Defining a suitable sibling predicate now is a little more involved than in the previous section. Certainly, two nodes are siblings whenever they share a common parent. This observation will enter our definition of the new SIBLING predicate, but it is not enough to determine which of the two nodes lies to the left of the other. Suppose we have already established that two given variables x and y represent sibling nodes in the infinite unbounded tree. We can then distinguish three cases in which x would correspond to a node somewhere to the left of the node represented by the variable y:

(1) First, assume both x and y are lefthand children of their common parent (i.e. both of them end in 0). Let z stand for the parent. Then x is of the form $z.i.0$ and y is

of the form $z.j.0$ (for some $i, j \in \mathbb{N}_0$). The word $z.i.0$ represents a node somewhere to the left of $z.j.0$ iff i is a greater number than j. This is the same as to say that y lexicographically precedes x. Hence, if both x and y are lefthand children of their common parent then x is a lefthand sibling of y iff y strictly precedes x in the lexicographic order $\prec$ declared over ω^*.

(2) By an analogous argument, if both x and y are righthand children of their common parent then x is a lefthand sibling of y iff $x \prec y$.

(3) Finally, x is also a lefthand sibling of y iff x ends in 0 and y ends in 1.

If none of these three conditions applies, then x cannot be a lefthand sibling of y. Altogether, we end up with the following definition of SIBLING:

$$\begin{array}{rcl}\textsc{sibling}(x,y) & = & (\exists z)[\textsc{child}(z,x) \wedge \textsc{child}(z,y)] \wedge \\ & & [(\textsc{left}(x) \wedge \textsc{left}(y) \wedge y \prec x) \vee \\ & & \ (\textsc{right}(x) \wedge \textsc{right}(y) \wedge x \prec y) \vee \\ & & \ (\textsc{left}(x) \wedge \textsc{right}(y))]\end{array}$$

Finally, we define a neighbour relation over the infinite unbounded discretely ordered tree by stipulating that two nodes x and y are neighbours iff they are siblings and there exists no other node z that is both a righthand sibling of x and a lefthand sibling of y:

$$\textsc{neighbour}(x,y) \quad = \quad \textsc{sibling}(x,y) \ \wedge\ \neg(\exists z)[\textsc{sibling}(x,z) \wedge \textsc{sibling}(z,y)]$$

Before we can accept the above definition of the NEIGHBOUR predicate, however, we have to make the following observation. The set of words used to encode a particular discretely ordered tree will be a proper subset of ω^*. The variable z featuring in the definition above is intended to represent nodes in this substructure only. That is, our characterisation of particular ordered trees later on will have to be such that, whenever two nodes are supposed to represent neighbours, then there can be no other word in ω^* that would be a righthand sibling of the first node and a lefthand sibling of the second if it was itself part of the chosen ordered tree. (Another option would have been to explicitly guard the quantification over z.)

Characterising initial subtrees. Just as every strongly discrete order is isomorphic to a suborder of the integers, every discretely ordered tree is isomorphic to an initial subtree of the infinite unbounded discretely ordered tree we have embedded into ω^*. In fact, it even suffices to only consider certain subtrees which we shall call *initial subtrees without gaps.* These are initial subtrees satisfying the following two properties:

(1) Whenever a word x ending in 0 belongs to the subtree, so do any of the righthand siblings of x which themselves end in 0.

(2) Similarly, whenever a word ending in 1 belongs to the subtree, so do any of its lefthand siblings which themselves end in 1.

This class of trees allows us to represent any finite branching factor as well as infinite branching that is either bounded to the left, bounded to the right, or unbounded in either direction. This covers all cases, just as there are only three distinct infinite order types that are strongly discrete: the positive integers (bounded to the left), the negative integers (bounded to the right), and the whole set of the integers (unbounded).

The following meta-predicate encodes the class of initial subtrees without gaps. The conditions are similar to the definition of $\textsc{tree}(T)$ in the previous section. In addition to that, we also have to ensure that T represents a subset of the words satisfying our defined predicate $\textsc{fulltree}$ and that there are no gaps between sibling nodes.

$$\begin{aligned}
\textsc{tree}(T) \quad = \quad & (\forall x)[T(x) \rightarrow \textsc{fulltree}(x)] \wedge \\
& (\forall x)[\textsc{root}(x) \rightarrow T(x)] \wedge \\
& (\forall x)(\forall y)[\textsc{child}(x, y) \wedge T(y) \rightarrow T(x)] \wedge \\
& (\forall x)(\forall y)[\textsc{sibling}(x, y) \wedge \textsc{left}(y) \wedge T(x) \rightarrow T(y)] \wedge \\
& (\forall x)(\forall y)[\textsc{sibling}(x, y) \wedge \textsc{right}(x) \wedge T(y) \rightarrow T(x)]
\end{aligned}$$

Let α be an $\mathbf{S\boldsymbol{\omega}S}$ variable assignment mapping the set variable T to a subset of ω^*. Our discussion shows that $(T^\alpha, [\![\textsc{child}]\!], [\![\textsc{sibling}]\!])$ will be a discretely ordered tree whenever $\alpha \models \textsc{tree}(T)$ holds. Also, vice versa, for any discretely ordered tree there exists an assignment α such that this tree is isomorphic to $(T^\alpha, [\![\textsc{child}]\!], [\![\textsc{sibling}]\!])$ for some set variable T with $\alpha \models \textsc{tree}(T)$. Furthermore, restricted to any T^α with $\alpha \models \textsc{tree}(T)$, the relation $[\![\textsc{descendant}]\!]$ is the transitive closure of $[\![\textsc{child}]\!]$ and $[\![\textsc{neighbour}]\!]$ is the immediate successor relation underlying $[\![\textsc{sibling}]\!]$.

Decidability for discretely ordered trees. Before we state the decidability result for this section, we give a formal definition of the class of formulas that are valid with respect to discretely ordered trees.

Definition 5.13 (Validity in discretely ordered trees) *A formula φ is called valid in discretely ordered trees iff $\mathcal{T} \models \varphi$ holds for every discretely ordered tree $\mathcal{T}$.*

Here is our Decidability Theorem for discretely ordered trees:

Theorem 5.14 (Decidability for discretely ordered trees) *The problem of checking validity in discretely ordered trees is decidable.*

Proof. Let φ be an **OTL** formula and let $P_1, \ldots, P_n$ be the propositional letters that appear in it. Now consider the $\mathbf{S\boldsymbol{\omega}S}$ formula φ^* defined as follows:

$$\varphi^* \quad = \quad (\forall T)[\textsc{tree}(T) \rightarrow (\forall P_1' \subseteq T) \cdots (\forall P_n' \subseteq T)(\forall x \in T)\, \textsc{tr}_x^T(\varphi)]$$

$\textsc{tr}_x^T$ is again the translation function given in Table 5.1, but this time the predicates $\textsc{descendant}$, $\textsc{child}$, $\textsc{sibling}$, and $\textsc{neighbour}$ used in the definition of $\textsc{tr}_x^T$ are the respective **SωS** predicates defined in this section. The meta-predicate TREE is our predicate encoding initial subtrees without gaps. The set variables $P'_1, \ldots, P'_n$ are the unary predicates we obtain when translating the propositional letters $P_1, \ldots, P_n$.

The formula φ^* says that for any initial subtree without gaps (represented by T) of the unbounded discretely ordered tree (which we have encoded as a substructure of ω^*) and for any sets of words (that is, any subsets of the set of words assigned to T) assigned to the set variables P'_1 to P'_n, the translation of the formula φ will be true, whatever node of the subtree we assign the variable x to. This is the case iff φ is valid with respect to any discretely ordered tree. Hence, checking validity of φ in discretely ordered trees is equivalent to checking validity of φ^* in **SωS**. The latter problem is known to be decidable, i.e. checking validity for discretely ordered trees must be a decidable problem as well. □

5.4 General Ordered Trees

In this section, we are going to show that **OTL** is a decidable logic. The proof falls into two parts. We first show how **OTL** with (almost) no restrictions on the admissible class of frames can be embedded into **S4S** (and thereby into **S2S**). The only restriction that we cannot eliminate is that for any such embedding the set of nodes in the respective ordered tree will have to be countable.[7] Therefore, in the second part of our proof we have to show that, in fact, this restriction does not matter: any satisfiable formula will also be satisfiable in a model based on an ordered tree with a countable set of nodes.

As for the first part of our proof, the crucial step up from Theorem 5.14 is that we are now going to drop the restriction to discretely ordered trees. In particular, general ordered trees allow for sibling nodes that are densely ordered. Therefore, before we describe the main construction of an infinite densely ordered tree, we are going to show how we can encode a single dense order into **S2S**.

Encoding dense orders in S2S. Our construction follows that given by Gabbay *et al.* [30]. The basic idea is to 'project' the nodes of the infinite binary tree underlying **S2S** down onto a straight line as shown in Figure 5.3.

Intuitively, such a projection induces a left-to-right ordering on the set of words in $\{0,1\}^*$. Given a word x, the word $x.0$ is supposed to lie slightly to the left of x, while the word $x.1$ lies a little to its right. However, neither of them will lie as far to the left or right of x as any of the prefixes (or ancestors) of x. In other words, $x.1$ for instance, will lie half way between x and the longest prefix of x ending in 1 (if any), i.e. the closest ancestor of x (with respect to the binary tree) lying itself to the righthand side of x. We

[7] In the literature, sets that are either finite or countably infinite are sometimes called *at most countable*. Here, we do not always make this distinction and simply refer to both finite and countably infinite sets as countable ones.

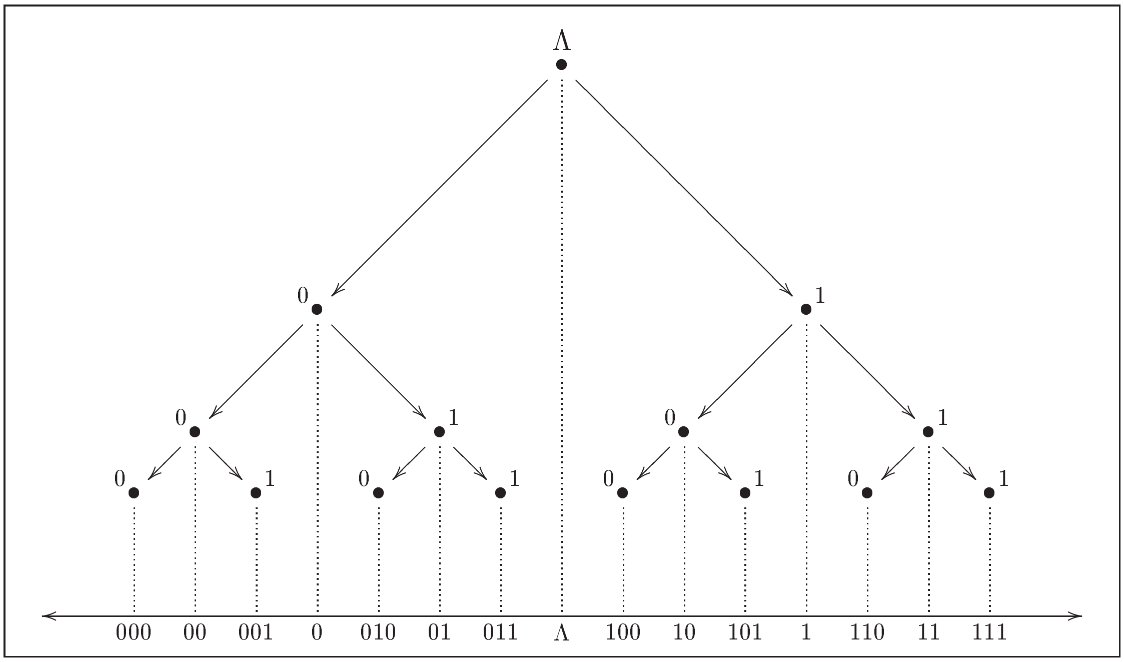

Figure 5.3: Encoding dense orders in **S2S**

now formally define this left-to-right ordering as follows:[8]

$$x \ll y \quad = \quad x.1 \leq y \ \vee \ y.0 \leq x \ \vee \ (\exists z)[z.0 \leq x \wedge z.1 \leq y]$$

That is, we have $x \ll y$ iff $x \neq y$ and there is a node w in the tree such that x is either w itself, its lefthand child, or a descendant of its lefthand child, and y is either w itself, its righthand child, or a descendant of its righthand child. The first disjunct describes the case where x is that dominating node w and y is either its righthand child or a descendant of the latter. The second disjunct describes the case where y is the dominating node w and x is either its lefthand child or a descendant of the latter. The third disjunct, finally, describes the case where both x and y are descendants of a common ancestor w; x is either its lefthand child or a descendant of its lefthand child and y is either its righthand child or a descendant of its righthand child.

We are now going to show that the definition of the predicate $\ll$ really yields the kind of left-to-right ordering we intended: it induces a strict linear order over the set of all words $\{0,1\}^*$ that is dense and without endpoints. We also remark that this order must be countable, the significance of which we are going to comment on at a later point in this section. To establish that $(\{0,1\}^*, \ll)$ is a *strict linear order* we have to check that it is indeed irreflexive, transitive, and connected:

(1) *Irreflexivity.* Follows immediately from the definition of $\ll$.

[8] The binary *predicate* $\ll$ induces a binary *relation* $[\![\ll]\!]$ over the set of words $\{0,1\}^*$. However, to simplify notation, in the sequel we are not always going to make this distinction and shall usually use the symbol $\ll$ also for the (semantic) relation.

(2) *Transitivity.* Again, this follows directly from the definition of $\ll$ (see also the explanation given just after the definition).

(3) *Connectedness.* Any two words in $\{0,1\}^*$ that are not identical must either have a common proper prefix (possibly Λ) or one must be a proper prefix of the other. Hence, we have $x = y$ or $x \ll y$ or $y \ll x$ for any two words $x, y \in \{0,1\}^*$, that is, the relation induced by $\ll$ must be connected.

Next we show that the linear order $(\{0,1\}^*, \ll)$ must be *dense*, i.e. for any $x, y \in \{0,1\}^*$ with $x \ll y$ there exists a word $z \in \{0,1\}^*$ such that $x \ll z$ and $z \ll y$. According to the definition of $\ll$, we can distinguish three cases. Firstly, if $x \ll y$ holds because $x.1 \leq y$ does, then the word $y.0$ fulfils our requirements: we have $x \ll y.0 \ll y$. Secondly, if we have $x \ll y$ by virtue of $y.0 \leq x$ then $x.1$ does the job, i.e. we get $x \ll x.1 \ll y$. And thirdly, in case we have $x \ll y$ because of $(\exists z)[z.0 \leq x \wedge z.1 \leq y]$ then that word z is such that $x \ll z \ll y$ holds. Hence, $(\{0,1\}^*, \ll)$ is indeed dense.

Next we observe that for any word x there exist both words that precede and words that succeed x in the order induced by $\ll$, i.e. $(\{0,1\}^*, \ll)$ is an order *without endpoints.* For instance, we have $x.0 \ll x$ and $x \ll x.1$ for any word $x \in \{0,1\}^*$.

Finally, $\{0,1\}^*$ must be a *countable* set, because there is a direct bijective mapping between $\{0,1\}^*$ and the natural numbers in their binary representation (simply add a single 1 at the beginning of each word).

A quick remark on the side, before we move on and use this kind of construction for our own purposes: By Cantor's Theorem [12], any two countable dense linear orders without endpoints are isomorphic to each other. In particular the theory of the rational numbers $(\mathbb{Q}, <)$ will be isomorphic to $(\{0,1\}^*, \ll)$. Hence, an immediate consequence of the above construction would be that the monadic second-order theory of $(\mathbb{Q}, <)$ — as well as that of any other countable unbounded dense linear order — is decidable [30, 61]. This result can serve as a basis for proving decidability results for linear temporal logic over various flows of time. We refer to Gabbay *et al.* [30] for more information on this subject.

Encoding the infinite densely ordered tree. As we have already mentioned at the beginning of this section, our first step in proving decidability for the full logic **OTL** will be to construct an infinite tree where the children of each node form a (countable) dense linear order without endpoints. We have just seen how to encode a *single* dense order into **S2S**, but this involved collapsing the entire binary tree. How are we going to get back to a tree while maintaining density at the same time?

The basic ideas underlying our construction are illustrated in Figure 5.4. Rather than using **S2S** we are now going to use **S4S**, the monadic second-order theory of *four* successor functions. For the set of children of any given node in the infinite densely ordered tree we are going to use either the first two successor functions $succ_0$ and $succ_1$ *or* the second two successor functions $succ_2$ and $succ_3$ to encode a countable dense linear order without endpoints in the same way as we have just seen. In addition, by alternating

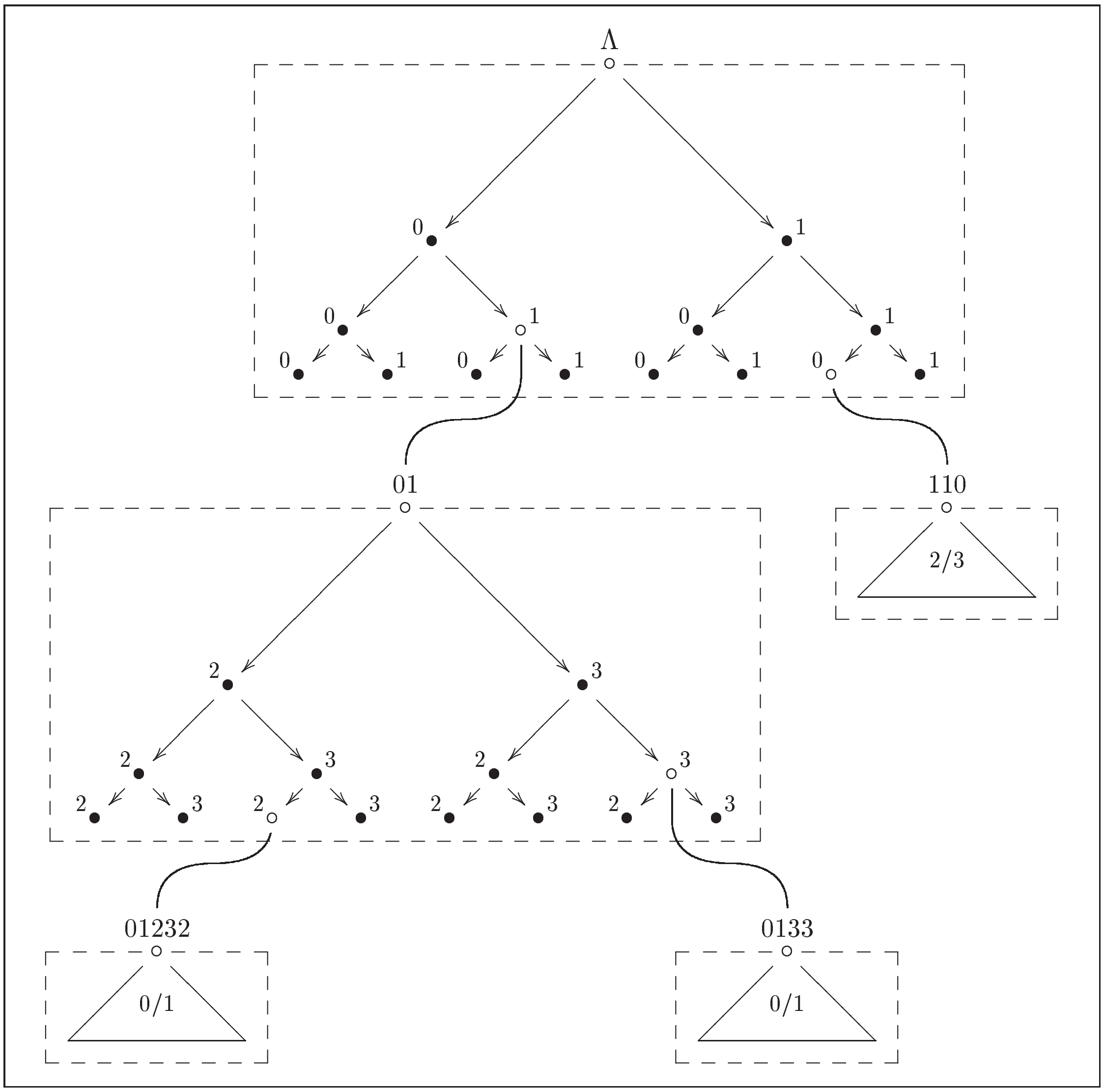

Figure 5.4: Encoding the infinite (densely) ordered tree in **S4S**

between the two pairs of successor functions, we can simulate the levels in a tree. For instance, the word 01 represents an element in the dense linear order we can map the set of words $\{0,1\}^*$ to. Similarly, the word 232 represents an element in the dense linear order we can embed into $\{2,3\}^*$ in precisely the same way. Now the concatenation of the two, i.e. the word 01232, may be considered representing a node on the second level of the infinite ordered tree. Nodes on the third level are again encoded using the first two successor functions, and so on.

The empty word Λ is used to represent the root of the ordered tree, that is, it is not part of the dense linear order of nodes from the first level below the root. Similarly, appending Λ (now considered an element of $\{2,3\}^*$) to the end of, say, 01 yields again 01, i.e. we stay on the first level of nodes. Of course, removing a single element (namely

Λ) from a countable dense linear order without endpoints yields again a countable dense linear order without endpoints, and the two are isomorphic to each other. Hence, our construction really does result in an infinite ordered tree where the set of children of any given node form a countable dense linear order without endpoints which is, for instance, isomorphic to the order of the rationals $(\mathbb{Q}, <)$.

Our suggested embedding of the infinite densely ordered tree into **S4S** makes use of all the words in $\{0, 1, 2, 3\}^*$, except those that begin with either a 2 or a 3. This set of words is characterised by the following predicate:

$$\textsc{fulltree}(x) \quad = \quad \neg(\exists y)[\textsc{root}(y) \wedge (y.2 \leq x \vee y.3 \leq x)]$$

Defining the accessibility relations. We now turn to the definition of the crucial accessibility relations over the infinite densely ordered tree for our embedding. The DESCENDANT predicate to be defined will have a lot in common with the strict prefix relation $<$ declared over words in $\{0, 1, 2, 3\}^*$: in 'most' cases, if x is a proper prefix of y, then the former will in fact represent an ancestor of the latter in the embedded tree. However, this is not the case if a node and its prefix encode elements of the same dense linear order. That is, to find out whether x represents an ancestor of y, we have to check not only that x is a proper prefix of y, but also that, when we 'move' from the encoding of x to that of y, at least one change in the pair of successor functions used will take place. For instance, if x ends either in 0 or in 1, then the prefix identical to x within y has to be immediately followed by either a 2 or a 3. A special case is the empty word Λ, which represents the root of the embedded tree and thereby an ancestor of any other node. Here is the formal definition of the DESCENDANT predicate for this embedding:

$$\begin{array}{rcl} \textsc{descendant}(x, y) & = & [(\exists z)(z.0 = x \vee z.1 = x) \ \wedge \ (x.2 \leq y \vee x.3 \leq y)] \ \vee \\ & & [(\exists z)(z.2 = x \vee z.3 = x) \ \wedge \ (x.0 \leq y \vee x.1 \leq y)] \ \vee \\ & & [\textsc{root}(x) \wedge \neg\textsc{root}(y)] \end{array}$$

A predicate for the child relation in the infinite ordered tree can be expressed in terms of the DESCENDANT predicate as follows:

$$\textsc{child}(x, y) \quad = \quad \textsc{descendant}(x, y) \ \wedge \ \neg(\exists z)[\textsc{descendant}(x, z) \wedge \textsc{descendant}(z, y)]$$

We should point out that whenever x and y are mapped to words that are being used for the embedding of the infinite ordered tree (i.e. if both of them are mapped to words not starting with either 2 or 3), then any word assigned to z validating the formula $\textsc{descendant}(x, z) \wedge \textsc{descendant}(z, y)$ would also have to belong to that set of words used for the embedding. Hence, no explicit restriction of the quantification over z in the above definition of the CHILD predicate is necessary.

Our new definition of the SIBLING predicate is similar to that of the predicate $\ll$, which we have seen earlier as part of the encoding of a dense linear order into **S2S**. We first use the CHILD predicate to check that two given variables x and y do represent sibling nodes at all. If they do, then either both of them will end in 0 or 1 or both of them will end in 2 or 3. In the former case, we simply copy the definition of $\ll$ given earlier; in the latter case we rewrite any 0 as a 2 and any 1 as a 3. We thus end up with the following definition:

$$\begin{array}{rcl} \textsc{sibling}(x,y) & = & (\exists z)[\textsc{child}(z,x) \wedge \textsc{child}(z,y)] \wedge \\ & & [(x.1 \leq y \vee y.0 \leq x \vee (\exists z)(z.0 \leq x \wedge z.1 \leq y)) \vee \\ & & (x.3 \leq y \vee y.2 \leq x \vee (\exists z)(z.2 \leq x \wedge z.3 \leq y))] \end{array}$$

As for the embedding discussed in the previous section, we can again express the NEIGHBOUR predicate in terms of the SIBLING predicate just defined. An important difference is that, this time, we are going to consider *all* initial subtrees of the infinite densely ordered trees, which means that we have to be more careful regarding the quantification over words that may represent sibling nodes in between the two nodes checked for neighbourhood. We solve this problem by explicitly restricting the quantification over the variable z in the following definition of NEIGHBOUR by the set variable T, which is going to represent the set of nodes making up the initial subtree in question later on:

$$\textsc{neighbour}(x,y) \quad = \quad \textsc{sibling}(x,y) \wedge \neg(\exists z \in T)[\textsc{sibling}(x,z) \wedge \textsc{sibling}(z,y)]$$

Strictly speaking, the NEIGHBOUR predicate takes three arguments: the two variables x and y, and the set variable T. However, as the symbol T will stay fixed for all uses of this predicate, we have not explicitly referred to it on the lefthand side of the above definition.

Characterising initial subtrees. The last part of our construction concerns again the characterisation of the set of all initial subtree of the infinite ordered tree, in order to capture the class of all countable ordered trees. This time, we take all possible initial subtrees into account, i.e. the definition of our new TREE is straightforward. We have to ensure that the set variable T is interpreted as a subset of the set of words characterised by the FULLTREE predicate, that the empty word Λ is part of that set, and that for any word belonging to the set also the word representing its parent node is included.

$$\begin{array}{rcl} \textsc{tree}(T) & = & (\forall x)[T(x) \rightarrow \textsc{fulltree}(x)] \wedge \\ & & (\forall x)[\textsc{root}(x) \rightarrow T(x)] \wedge \\ & & (\forall x)(\forall y)[\textsc{child}(x,y) \wedge T(y) \rightarrow T(x)] \end{array}$$

Just as any (at most) countable linear order is isomorphic to a substructure of a countable dense linear order without endpoints, any ordered tree with a set of nodes that is (at most) countable will be isomorphic to an initial subtree of the infinite densely ordered tree characterised by the defined predicate FULLTREE, together with the accessibility relations $[\![\textsc{child}]\!]$ and $[\![\textsc{sibling}]\!]$. That is, if α is an assignment for **S4S** mapping the set variable T to a subset of $\{0,1,2,3\}^*$ such that $\alpha \models \textsc{tree}(T)$ holds, then $(T^\alpha, [\![\textsc{child}]\!], [\![\textsc{sibling}]\!])$ will be a countable ordered tree. As for the other direction, every ordered tree with a countable set of nodes is isomorphic to $(T^\alpha, [\![\textsc{child}]\!], [\![\textsc{sibling}]\!])$ for some variable assignment α.

Furthermore, by construction, $[\![\textsc{descendant}]\!]$ will be the transitive closure of $[\![\textsc{child}]\!]$ and $[\![\textsc{neighbour}]\!]$ will the immediate successor relation induced by $[\![\textsc{sibling}]\!]$.

Decidability for countable ordered trees. We are now in a position to prove that our logic **OTL**, restricted to ordered trees with countable sets of nodes, is decidable. Let us first capture the notion of validity with respect to this class of frames in a formal definition:

Definition 5.15 (Validity in countable ordered trees) *A formula φ is called valid in countable ordered trees iff $\mathcal{T} \models \varphi$ for all ordered trees $\mathcal{T} = (T, R, S)$ such that T is a set that is at most countable.*

Here is our intermediate decidability result:

Lemma 5.16 (Decidability for countable ordered trees) *The problem of checking validity in countable ordered trees is decidable.*

Proof. Let φ be an **OTL** formula and let $P_1, \ldots, P_n$ be the propositional letters appearing in φ. We define the **S4S** formula φ^* as follows:

$$\varphi^* \quad = \quad (\forall T)[\textsc{tree}(T) \rightarrow (\forall P_1' \subseteq T) \cdots (\forall P_n' \subseteq T)(\forall x \in T)\, \textsc{tr}_x^T(\varphi)]$$

Here, $\textsc{tr}_x^T$ is the translation function defined in Table 5.1. The predicates $\textsc{descendant}$, $\textsc{child}$, $\textsc{sibling}$, and $\textsc{neighbour}$ referred to in the definition of $\textsc{tr}_x^T$ are understood to be the respective **S4S** predicates defined earlier in this section. In particular, the unary predicate symbol T appearing as a parameter of the translation function $\textsc{tr}_x^T$ is the same symbol as that appearing in the definition of the $\textsc{neighbour}$ predicate. The set variables $P_1', \ldots, P_n'$ correspond to the propositional letters $P_1, \ldots, P_n$.

The formula φ^* expresses that for any initial subtree (represented by the set variable T) of the countably infinite densely ordered tree and any possible assignments of subsets of the set of words assigned to T to the set variables $P_1', \ldots, P_n'$, as well as any possible assignment of an element of that same set to the variable x, the translation of φ will be true. This is precisely the case iff the formula φ is valid in countable ordered trees. Hence, the problems of checking validity of φ in countable ordered trees and checking validity of φ^* in **S4S** are in fact equivalent. Given that the latter is known to be decidable, we find that checking validity with respect to countable ordered trees must be decidable as well. □

Lemma 5.16 strongly suggests that **OTL** is decidable, also without the restriction to countable frames. In the remainder of this section we are going to verify this intuition and prove that the concepts of validity, on the one hand, and validity with respect to countable ordered trees, on the other, do in fact coincide.

Eliminating the restriction to countable ordered trees. We say that a formula φ has got a *countable ordered tree model* iff there is an ordered tree model $\mathcal{M} = (T, R, S, V)$ such that T is an (at most) countable set and $\mathcal{M}, t \models \varphi$ holds for some node $t \in T$. The following lemma states that every formula that has got an ordered tree model will also have a countable ordered tree model (the converse trivially holds as well).

The corresponding result in classical first-order logic is known as the *downward Löwenheim-Skolem Theorem*: every first-order sentence with a model has got a countable model [14]. We also have similar results in linear temporal logics: the language of temporal logic cannot 'distinguish' countable models from those that are not countable. One way of proving this kind of result would be to translate into first-order logic and then to appeal to the downward Löwenheim-Skolem Theorem [30]. This kind of approach seems likely to also be applicable in the case of **OTL**. Nevertheless, we choose to prove our lemma via a direct argument here.

Lemma 5.17 (Countable models) *Every formula with an ordered tree model has got a countable ordered tree model.*

Proof. Let φ be a satisfiable formula. We are going to show how we can transform a given (possibly uncountable) model for φ into a countable model. Recall that the depth of an ordered tree is always countable. Hence, it is enough to reduce the tree underlying the original model to a tree where every internal node has got a countable set of children; the overall number of nodes will then be countable as well. In fact, we are going to transform the original model of φ into a general model based on a loosely ordered tree (without cycles) where every set of pseudo siblings is countable.[9] It follows from our construction in the proof of Lemma 3.39 in Chapter 3 that the corresponding ordered tree will have at most countably many children for every node (because we have bulldozed using only integers and rationals, either of which are countable).

A formula φ is satisfiable iff there is a model based on an ordered tree where $\varphi \vee \diamondsuit^+ \varphi$ is satisfied at the root. Hence, without loss of generality, we may assume that our input formula φ is true at the root of the ordered tree of the original model.

Types. Before we turn to the description of the transformation of the original model into a model based on a countable loosely ordered tree, we introduce the notion of a type of a node. Let φ be our satisfiable input formula and let $\mathcal{M} = (T, R, S, V)$ be the original ordered tree model satisfying φ at the root. Furthermore, let $sf(\varphi)$ denote the set of subformulas of φ (including φ itself).

We define the *type* of a node $t \in T$ as the set of subformulas of φ satisfied at t in the original model, i.e. as the set $\{\psi \in sf(\varphi) \mid \mathcal{M}, t \models \psi\}$. The maximal number of possible types with respect to a given formula φ is $2^{|sf(\varphi)|}$: each subformula either holds

[9] Recall that, at the end of Chapter 3, we have shown that a formula is satisfiable (in an ordered tree model) iff it is satisfiable in a general model based on a loosely ordered tree (Theorem 3.41). Loosely ordered trees were introduced in Definition 3.33, loosely ordered trees without cycles in Definition 3.35.

at a particular node or it does not hold at that node.[10] The number of subformulas of a formula φ is at most equal to the length of φ: every propositional letter appearing in φ constitutes a subformula and every operator gives rise to another subformula. If a subformula appears more than once, then the number of subformulas will in fact be lower than the length of the formula. In summary, if n is the length of φ then the number of types is at most 2^n. (What matters for our purposes here is that the number of distinct types is finite.)

Constructing the new model. We are now going to describe the process of constructing a new model based on a loosely ordered tree without cycles. The new tree will be put together from (some of the) nodes of the original ordered tree. Typically, 'most' of the nodes in the original tree will not feature in the new one. Whether or not a particular node is used depends on its type. We are going to prune the (possibly uncountable) set of children of a node in an ordered tree to obtain a countable set of children in a loosely ordered tree. The central intuition behind this reduction is that at least one node of every type appearing within that set of children in the original model should survive. Unavoidable infinite repetitions of types are taken care of by means of the introduction of clusters. The same reduction process is to be applied to every set of children in the original model. The new set of nodes T' is a subset of the original set T. A node belongs to T' if it survives the pruning operation *and* if all its ancestors survive as well (otherwise that node would be disconnected from the root). The new child relation R' is the original relation R restricted to nodes in T'. The pseudo sibling and the pseudo neighbour relations S' and N' are going to be defined as part of the pruning operation to be described. The new valuation function V' is the same as the original V, but restricted to the new set of nodes, i.e. $V'(P) = V(P) \cap T'$ for any propositional letter P.

Let us now turn to the details of our inductive pruning operation for a given set of sibling nodes. In a preparatory step we identify for every node t its *maximal sequence of neighbours* $MSN(t)$, which contains all those siblings of t (including t itself) for which there are only a finite number of siblings in between that node and t. Such a sequence may or may not be finite, but observe that $MSN(t)$ will be countable for any node t. For a dense node t without immediate neighbours, for instance, $MSN(t)$ will only contain t itself. For every node t, either all or none of the nodes in $MSN(t)$ will be used for the new model. Those that will be used will become non-cyclic N-sequences in the loosely ordered tree.

A far as possible, we are going to try to put these kinds of sequences together into clusters. We say that a given set of sibling nodes *can be turned into a cluster* iff every node in that set has got both a lefthand and a righthand sibling of each one of the types covered by the nodes in the set. That is, whenever we *cannot* turn a set into a cluster

[10] The number of different types that a node could actually have will often be much smaller than that, because the set of formulas forming a type is certainly required to be propositionally consistent. If, for example, we have a negated formula $\neg\psi$ as a subformula of φ then ψ will also be a subformula, but ψ and $\neg\psi$ can never be satisfied at the same node in a model.

then that set will contain a node with a type that has no further occurrences to the right of that set in the original model (and accordingly for the lefthand side).

We are now going to pick the sequences to be used for the new model. We start by selecting an arbitrary node t, together with the associated sequence $MSN(t)$ (from the set of children we want to reduce). If that sequence can be turned into a cluster, we refer to it as the *current cluster*, otherwise as the *current sequence*. The procedure of selecting sequences to the right of $MSN(t)$ is inductively defined as follows (a similar process applies to sequences to the left). We first consider the case where the node picked in the previous round has *not* become part of a cluster:

(1) If there are no more nodes to the right of the current sequence, then *stop*.

(2) Otherwise, pick a node t to the right such that any node in between the current sequence and t has got a type that also occurs again somewhere to the right of t. (This is always possible.)

(3) Identify the maximal sequence of neighbours $MSN(t)$ and declare it the new cluster if possible; otherwise declare it the new sequence.

Now for the case where the previously picked node *has* become part of a cluster:

(1) If there is no more node to the right of the current cluster that has a type not already represented in the current cluster, then *stop*.

(2) Otherwise, pick a node t (with a type not covered by the current cluster) to the right such that any node in between the current cluster and t has got a type that also occurs somewhere to the right of t. (Again, this is always possible.)

(3) Identify $MSN(t)$. If possible, add it to the current cluster. Failing that, if possible, declare it the new current cluster. If neither is possible, declare it the new current sequence.

Whenever the new current structure is a sequence or a new cluster (which could not just be added to the previous one), the number of types ever to be considered again to the right will decrease by at least one (because the reason for introducing either a sequence or a new cluster is that at least one type has no more occurrences to the right). Otherwise, that is, if we enlarge the current cluster, at least the number of types to be considered in the next step will decrease. Hence, this process will eventually terminate.

If we apply the same technique to nodes to the left of the node picked in the first round, then we end up with an ordering of clusters and irreflexive points, i.e. we end up with a set of pseudo siblings and pseudo neighbours of a loosely ordered tree. We define the pseudo neighbour relation N' by connecting consecutive nodes in any maximal sequence of neighbours. The pseudo sibling relation S' is the smallest extension of the original S (restricted to nodes in T') that also relates any two nodes in the same cluster. This concludes our definition of the new model $\mathcal{M}' = (T', R', S', N', V')$. Clearly, T' will be a countable set, because the children of every node are now made up of a finite number of (countable) sequences of neighbours.

Satisfiability in the new model. It remains to be shown that the new model $\mathcal{M}'$ is indeed a model for our input formula φ. To show that this is the case we prove the following stronger result:

> *Claim:* For every node $t \in T'$ in the new model and every subformula ψ of the input formula φ we have $\mathcal{M}', t \models \psi$ iff ψ belongs to the type of t.

We can prove this claim by induction over the the structure of subformulas ψ of the input formula. Without loss of generality, we only need to consider the core operators of our language (as covered, for instance, by Definition 3.18). For the base case, assume ψ is a propositional letter. The claim then follows immediately from the definition of the new valuation function V'. Also, $\top$ will be true at any node in any model and belong to any type, provided it is a subformula of φ.

For the induction step, we consider each one of the core operators in turn:

(1) *Negation.* Suppose the subformula in question is $\neg\psi$. We have $\mathcal{M}', t \models \neg\psi$ iff $\mathcal{M}', t \not\models \psi$. By the induction hypothesis, the latter is equivalent to ψ not being part of the type of t, i.e. to $\neg\psi$ belonging to the type of t.

(2) *Conjunction.* We have $\mathcal{M}', t \models \psi_1 \wedge \psi_2$ iff both $\mathcal{M}', t \models \psi_1$ and $\mathcal{M}', t \models \psi_2$ iff both ψ_1 and ψ_2 belong to the type of t (induction hypothesis) iff $\psi_1 \wedge \psi_2$ belongs to the type of t.

(3) *Parent operator.* $\mathcal{M}', t \models \ominus\psi$ holds iff there exists a $t' \in T'$ such that $(t', t) \in R'$ and $\mathcal{M}', t' \models \psi$. By the induction hypothesis, the latter is equivalent to ψ being part of the type of t', while the former means that we also have $(t', t) \in R$ in the original model. Hence, $\mathcal{M}', t \models \ominus\psi$ is equivalent to $\mathcal{M}, t \models \ominus\psi$, which in turn is equivalent to $\ominus\psi$ being a member of the type of t.

(4) *Child operator.* If $\mathcal{M}', t \models \diamondsuit\psi$ holds then t will have a child t' at which ψ holds. Hence, by the induction hypothesis, ψ will belong to the type of t' and $\diamondsuit\psi$ to the type of t.

   The other direction is more interesting. Suppose $\diamondsuit\psi$ belongs to the type of t, i.e. we have $\mathcal{M}, t \models \diamondsuit\psi$ in the original model. Then t must have a child t' where ψ is true, i.e. ψ belongs to the type of t'. Our model transformation is such that, for every type present amongst the children of a node in the original model, at least one node of that type will survive. Hence, t' will have a sibling t'' (possibly t' itself) of the same type which is still a child of t in the new tree. Now, by the induction hypothesis, we get $\mathcal{M}', t'' \models \psi$ and thereby also $\mathcal{M}', t \models \diamondsuit\psi$.

(5) *Descendant operator.* A formula of the form $\diamondsuit^+\psi$ belongs to the type of t iff t has got a child with a type containing either ψ or $\diamondsuit^+\psi$. This observation together with an argument along the lines of that used for the previous case covers the induction step for the descendant operator.

(6) *Neighbour operators.* By construction, whenever a node t survives the pruning operation, so do its immediate neighbours (if any), because we have only manipulated maximal sequences of neighbours as a whole. Hence, truth of subformulas of the form $\bigcirc_{\rightarrow}\psi$ as well as $\bigcirc_{\leftarrow}\psi$ is not affected.

(7) *Sibling operators.* These are the only truly non-trivial cases. We start with the proof from left to right for subformulas of the form $\Diamond_{\rightarrow}\psi$. Suppose we have $\mathcal{M}', t \models \Diamond_{\rightarrow}\psi$, i.e. there exists a node $t' \in T'$ such that $(t, t') \in S'$ and $\mathcal{M}', t' \models \psi$. This means that we have either $(t, t') \in S$ in the original model as well or t and t' are part of the same cluster in the new model. In the former case, it follows immediately from the induction hypothesis that ψ will be in the type of t' and, hence, $\Diamond_{\rightarrow}\psi$ will be in the type of t. Now consider the case where $(t, t') \in S$ does not hold, but the two nodes are part of the same cluster. Then, by our condition for the admissibility of turning a set of nodes into a cluster, t must have had a righthand sibling with the same type as t' in the original model. Hence, also in this case, $\Diamond_{\rightarrow}\psi$ will belong to the type of t.

Now suppose $\Diamond_{\rightarrow}\psi$ belongs to the type of node t. Then t must have got a righthand sibling t' in the original model where ψ is true, i.e. ψ belongs to the type of t'. By construction, whenever t has got a righthand sibling of a certain type in the original model, then it will have either a (strictly) righthand sibling of the same type also in the new model, or it will at least share a cluster with a node of that type. In other words, there exists a node $t'' \in T'$ with $(t, t'') \in S'$ that has the same type as t'. Hence, by the induction hypothesis, we have $\mathcal{M}', t'' \models \psi$ and thereby also $\mathcal{M}', t \models \Diamond_{\rightarrow}\psi$.

The case of subformulas of the form $\Diamond_{\leftarrow}\psi$ is accounted for in an analogous manner.

This concludes our inductive proof of the above claim. Recall that a subformula ψ of φ belongs to the type of a node t iff we have $\mathcal{M}, t \models \psi$ in the original model. Hence, given that φ (which is a subformula of itself) is true at the root in the original model $\mathcal{M}$ and given that the root of the tree is part of T', by the above claim, the new model $\mathcal{M}'$ will satisfy φ at the root of the new loosely ordered tree.

Wrapping up. In summary, we have shown that for any ordered tree model $\mathcal{M}$ for a given satisfiable formula φ there exists a general model $\mathcal{M}'$ based on a loosely ordered tree (without cycles) with a countable set of nodes that also satisfies φ. The latter, in turn, implies that φ is satisfiable in a countable ordered tree model, namely the model we get if we bulldoze and unravel this loosely ordered tree model according to the construction outlined in the proof of Lemma 3.39. This concludes our proof of Lemma 5.17. □

Decidability. Putting two and two together (or rather Lemmas 5.16 and 5.17), our main decidability result now follows almost immediately:

Theorem 5.18 (Decidability) *The modal logic of ordered trees is decidable.*

Proof. By Lemma 5.17, any formula that does not have a countable ordered tree model is not satisfiable at all. The converse trivially holds as well. Hence, the notions of validity and validity with respect to countable ordered trees coincide. By Lemma 5.16, checking the latter is a decidable problem, i.e. the problem of checking validity in **OTL** must be decidable as well. □

Was this result to be expected? Although our embedding of **OTL** into the decidable logic **S2S** is of a certain complexity, one may argue that, given Rabin's Theorem, the decidability of a modal logic based on *some* class of trees is not that surprising a result. On the other hand, two-dimensional modal logics with interacting modalities can often be undecidable. In particular, **OTL** may be considered a complex combination of two separate logical systems (the vertical and the horizontal component) that bears certain similarities to product logics that are typically found to be undecidable [31].[11]

Another reason why we consider the above Decidability Theorem an important result is the fact that, under our extended temporal logic interpretation, **OTL** may be considered an (albeit restricted) interval logic (rather than just a point-based temporal logic); and modal interval logics, such as the logic of Halpern and Shoham [37] for instance, are also often found to be undecidable.

Proving decidability directly. A possible objection to decidability proofs such as the ones we have given in this chapter would be that they provide only little intuition on *why* the logic in question is decidable. In that respect, proofs that establish decidability by a direct argument rather than the reduction to a different problem (in our case the validity problem for a decidable monadic second-order theory) have certain advantages. On the other hand, proofs based on reduction, of course, are often easier to produce.

In [19] we have put forward an idea for a direct proof of the decidability of **OTL** and sketched details of a proof for the case of discretely ordered trees. This approach is, in part, inspired by techniques used by Sistla and Clarke [66] to establish a number of upper complexity bounds for propositional linear temporal logics. There are also connections to our proof of Lemma 5.17. The central idea is to show that **OTL** has the (abstract) *bounded finite model property*, that is, to show that any formula φ that is satisfied by *some* model is also satisfied by a model of limited size, where the maximal size of the model is a function of the length of φ. Decidability is a direct consequence of the bounded finite model property, because a naïve decision procedure could simply enumerate all potential models up to the known maximal size and check each one of them.

[11] For example, if we use the modal operators referring to the reflexive closures of our transitive accessibility relations, then **OTL** validates two of the three formulas used to axiomatise the most basic properties of product logics, namely *commutativity* and the *Church-Rosser property* [31]. These formulas are $\Diamond_{\rightarrow}^{*} \Diamond_{\uparrow}^{*} A \rightarrow \Diamond_{\uparrow}^{*} \Diamond_{\rightarrow}^{*} A$, which characterises *one* direction of commutativity (but the converse does not hold), and $\Diamond_{\rightarrow}^{*} \Box_{\uparrow}^{*} A \rightarrow \Box_{\uparrow}^{*} \Diamond_{\rightarrow}^{*} A$, which characterises the Church-Rosser property. See also our discussion of this issue in Chapter 3 (page 46).

To be precise, in the case of **OTL**, we have (or rather expect to have) an *abstract* finite model property, because the models of bounded size are in fact not necessarily based on ordered trees anymore (or loosely ordered trees or any other class of p-morphic pre-images of ordered trees), but on structures that finitely *represent* (possibly) infinite ordered tree models. Our proof idea involves pruning and transforming a given model for an input formula φ in ways not dissimilar from our proof of the 'countable model property' stated in Lemma 5.17. For example, it is possible to show that, whenever two sibling nodes t_1 and t_2 in a given ordered tree model have the same *type* (with respect to the reference formula φ),[12] then we can remove the stretch of nodes (including the subtrees they dominate) between t_1 and t_2 (including either one of them) from the model without affecting satisfiability of φ, provided the node satisfying φ itself does not get pruned away and for any formula of the form $\diamondsuit\psi$ or $\diamondsuit^+\psi$ in the type of the parent of t_1 and t_2 at least one witness remains on the tree. Of course, for instance in the case of an unbounded set of sibling nodes, this kind of model pruning alone will not be sufficient to reduce an infinite model to a finite structure. In addition, we propose to reduce infinite sequences of neighbouring nodes to *periodic* sequences, which can then be represented using only a finite amount of space. Infinite repetitions of types in dense parts of an ordered tree model can be represented by introducing *clusters* (see proof of Lemma 5.17). The vertical dimension of an ordered tree can be reduced in a similar manner. By adding meta-information (on the repetition of types of nodes along branches) we can transform trees of infinite depth into finite structures.

We believe this to be a promising approach, but a number of details of this method of establishing a bounded finite model property — and thereby proving decidability — still remain to be worked out. So while the reduction techniques exploited in this chapter provide excellent tools to establish theoretical results, the model pruning approach may, in the long run, yield a better understanding of ordered tree logics. At this stage, however, we settle for the present proof of Theorem 5.18, but note the continuation of the programme started in [19] as an important direction for future work.

5.5 Summary

In this chapter, we have used embeddings into the monadic second-order logic of two successor functions as a tool to establish decidability results for ordered tree logics. We have proved decidability in the general case as well as for two specific classes of ordered trees. We briefly review the main points.

Decidability over finite binary trees. Decidability of our logic with respect to frames that are finite binary trees is an almost immediate consequence of the fact that the monadic second-order theory of two successor functions remains decidable if we add quantification over set variables that must be interpreted as finite sets of words. We have

[12] The notion of a type of a node has been defined in the proof of Lemma 5.17.

used this case study to introduce the *translation* of **OTL** formulas into second-order logic and to show how we can quantify over the set of all *initial subtrees* of the infinite tree as a means of taking the class of all admissible frames into account.

Decidability over discretely ordered trees. Next we have shown that the modal logic of *discretely ordered trees* is decidable by embedding it into the monadic second-order theory of ω successor functions. Besides establishing decidability of an interesting logic in its own right, this case study is also an example for a rather more complex embedding. The main difference to the previous case is that this time we were not using the infinite tree underlying the second-order theory in question *directly*, but rather a *substructure* of that theory. The interesting aspects of this proof are in particular the characterisation of this substructure and the definition of the accessibility relations of **OTL** with respect to the elements of this structure.

Decidability in the general case. As a first step towards proving decidability of **OTL** we have shown that the modal logic of *countable ordered trees* is decidable by embedding it into the monadic second-order theory of four successor functions. Our construction extends the well-known embedding of countable dense linear orders without endpoints into the infinite binary tree by alternating between two pairs of successor functions to encode the vertical dimension of an ordered tree.

In a second step, we have shown that, in fact, **OTL** is not rich enough a language to distinguish between structures that are countable and those that are not. The proof is constructive in the sense that it shows how, in theory, a model based on a countable ordered tree could be constructed given an arbitrary ordered tree model satisfying a particular formula. Decidability of the full logic then follows from our result for countable ordered trees.

Chapter 6

Zooming out

We conclude by reflecting on possible extensions to our modal logic of ordered trees. This chapter also gives a brief summary of the main results obtained and states a number of open problems and suggested directions for further investigation.

6.1 Extensions

In this section we present an extended version of our modal logic of ordered trees. We suggest a number of additional modal operators to obtain an even more expressive logic. This discussion forms part of our concluding chapter, because it is, essentially, a collection of ideas for future directions of research, rather than a formal investigation into another logic.

Extended modal operators. Given our standard interpretation of **OTL** as a temporal logic, a natural extension would be to include also modal operators such as the *until*-operator familiar from linear temporal logic. In fact, we may even introduce *until*-style operators for the *vertical* dimension of an ordered tree. We start by defining the syntax of well-formed formulas for this extended language.

Definition 6.1 (Extended formulas) *The set of well-formed formulas of extended* **OTL** *is the smallest set such that every simple* **OTL** *formula is an extended formula and, if* φ *and* ψ *are extended formulas, so are* $(\varphi \text{ SINCE } \psi)$, $(\varphi \text{ UNTIL } \psi)$, $(\varphi \text{ UP-TO } \psi)$, *and* $(\varphi \text{ DOWN-TO } \psi)$.

Again, in practice we are going to drop unnecessary brackets whenever the structure of formulas is clear without them.

Semantics for the extended language. The semantics of the extended logic is based on the same notion of ordered tree models as before. Recall that an ordered tree model is a pair $\mathcal{M} = (\mathcal{T}, V)$ where $\mathcal{T}$ is an ordered tree and V is a suitable valuation function (see Definition 3.5). As we shall see, most operators turn out to be definable (as abbreviations on the syntactic level), which is why the following definition only covers the 'core' operators.

Definition 6.2 (Extended truth conditions) *We inductively define the notion of an extended formula being true in an ordered tree model $\mathcal{M} = (\mathcal{T}, V)$ at a node $t \in \mathcal{T}$ as follows:*

(1) $\mathcal{M}, t \models P$ *iff* $t \in V(P)$ *for propositional letters* P*;*

(2) $\mathcal{M}, t \models \top$*;*

(3) $\mathcal{M}, t \models \neg\varphi$ *iff* $\mathcal{M}, t \not\models \varphi$*;*

(4) $\mathcal{M}, t \models \varphi \wedge \psi$ *iff* $\mathcal{M}, t \models \varphi$ *and* $\mathcal{M}, t \models \psi$*;*

(5) $\mathcal{M}, t \models \varphi$ SINCE ψ *iff t has a left sibling t' with $\mathcal{M}, t' \models \psi$, and $\mathcal{M}, t'' \models \varphi$ holds for all nodes t'' that are left siblings of t and right siblings of t';*

(6) $\mathcal{M}, t \models \varphi$ UNTIL ψ *iff t has a right sibling t' with $\mathcal{M}, t' \models \psi$, and $\mathcal{M}, t'' \models \varphi$ holds for all nodes t'' that are right siblings of t and left siblings of t';*

(7) $\mathcal{M}, t \models \varphi$ UP-TO ψ *iff t has an ancestor t' with $\mathcal{M}, t' \models \psi$, and $\mathcal{M}, t'' \models \varphi$ holds for all nodes t'' that are ancestors of t and descendents of t';*

(8) $\mathcal{M}, t \models \varphi$ DOWN-TO ψ *iff t has an descendent t' with $\mathcal{M}, t' \models \psi$, and $\mathcal{M}, t'' \models \varphi$ holds for all nodes t'' that are descendents of t and ancestors of t'.*

Definable operators. The remaining propositional connectives and modalities of our language may be treated as definable operators. To start with, we can define *falsum* in terms of *verum* and negation: $\bot = \neg\top$. Now we can give the following syntactic definitions for our next- and diamond-operators:

$$
\begin{array}{rclrcl}
\bigcirc_{\leftarrow}\varphi & = & \bot \text{ SINCE } \varphi & \qquad \Diamond_{\leftarrow}\varphi & = & \top \text{ SINCE } \varphi \\
\bigcirc_{\rightarrow}\varphi & = & \bot \text{ UNTIL } \varphi & \qquad \Diamond_{\rightarrow}\varphi & = & \top \text{ UNTIL } \varphi \\
\bigcirc_{\uparrow}\varphi & = & \bot \text{ UP-TO } \varphi & \qquad \Diamond_{\uparrow}\varphi & = & \top \text{ UP-TO } \varphi \\
\Diamond_{\downarrow}\varphi & = & \bot \text{ DOWN-TO } \varphi & \qquad \Diamond_{\downarrow}^{+}\varphi & = & \top \text{ DOWN-TO } \varphi
\end{array}
$$

These definitions coincide with the semantic characterisations of the same modalities originally given in Definition 3.6. Take, for instance, the case of the right sibling modality $\Diamond_{\rightarrow}$. A formula of the form $\Diamond_{\rightarrow}\varphi$ is supposed to be true at a node t iff t has got a righthand sibling t' at which φ is true. According to our new syntactic definition the formula $\Diamond_{\rightarrow}\varphi$ is equivalent to ($\top$ UNTIL φ). The latter has been defined to be true at node t iff t has a right sibling t' such that φ holds at t' and $\top$ holds at all nodes t'' that are right siblings of t and left siblings of t'. But the second part of this condition will always be true, because $\top$ has been defined to hold at every node in a tree. Hence, the truth condition reduces to the existence of a righthand sibling t' at which φ is true, i.e. the two definitions coincide as claimed.

As an example for a next-operator consider the case of the parent modality $\bigcirc_{\uparrow}$. According to our original definition, $\bigcirc_{\uparrow}\varphi$ is true at a node t iff t has got a parent t' at which φ holds. By Definition 6.2, the formula $(\bot \text{ UP-TO } \varphi)$ has got the same semantics. It is satisfied at a node t iff t has got an ancestor t' at which φ is true and the formula $\bot$ is true at all nodes t'' which are both ancestors of t and descendants of t'. But $\bot$ cannot be true at any node, that is, there cannot be any nodes in between t and its ancestor t'. Hence, t' must be the parent of t.

Box-operators and the remaining propositional connectives can be defined in the same way as before (see Definition 3.7).

Global since and until. In analogy to the global past and future modalities introduced in Chapter 3 (see page 48), we can now even define global versions of the *until*-style operators. We first define a non-strict version of UP-TO:

$$\varphi \text{ UP-TO}^* \psi \quad = \quad \psi \vee (\varphi \wedge (\varphi \text{ UP-TO } \psi))$$

That is, $(\varphi \text{ UP-TO}^* \psi)$ is true at a node t iff φ is true all the way up (including t itself) to an ancestor of t where ψ holds, or ψ is already true at t itself.

The global versions of *since* and *until* can now be defined like this:

$$\begin{aligned}
\varphi \ \underline{\text{SINCE}}\ \psi \quad &= \quad \square_{\leftarrow}\square_{\downarrow}^{*}\varphi \text{ UP-TO}^* (\square_{\downarrow}^{*}\varphi \text{ SINCE } (\psi \vee (\square_{\rightarrow}\square_{\downarrow}^{*}\varphi \text{ DOWN-TO } (\square_{\rightarrow}\square_{\downarrow}^{*}\varphi \wedge \psi)))) \\
\varphi \ \underline{\text{UNTIL}}\ \psi \quad &= \quad \square_{\rightarrow}\square_{\downarrow}^{*}\varphi \text{ UP-TO}^* (\square_{\downarrow}^{*}\varphi \text{ UNTIL } (\psi \vee (\square_{\leftarrow}\square_{\downarrow}^{*}\varphi \text{ DOWN-TO } (\square_{\leftarrow}\square_{\downarrow}^{*}\varphi \wedge \psi))))
\end{aligned}$$

The formula $(\varphi \ \underline{\text{UNTIL}}\ \psi)$, for instance, is true at a node t iff φ is true at all righthand siblings of t (including their descendants) and the same is true for all ancestors of t up to some ancestor for which φ is true until a particular righthand sibling at which either ψ is true or which at least has a descendant where ψ is true and φ is true at all lefthand siblings (and their descendants) of the nodes we pass when going down to that node satisfying ψ. In other words, there is a node *somewhere* to the right of t at which ψ holds and φ is true at all the nodes in between. This is the usual definition of an *until*-operator, here with respect to the *global* left-to-right ordering over the nodes in an ordered tree. We can easily check that we get $\Diamond_{\Rightarrow}\varphi \equiv \top \ \underline{\text{UNTIL}}\ \varphi$, etc., as expected.

Is there a 'global next-operator'? Observe that it does not make sense to define a (horizontal) global next-operator, as there is no notion of a *unique* global successor node. However, the formula $(\bot \ \underline{\text{UNTIL}}\ \varphi)$ yields something similar: It is true at a node t whenever φ is true either at the righthand neighbour of t, or at a leftmost descendant of the righthand neighbour of t, or — in case t is itself a rightmost child — at the righthand neighbour (or one of its leftmost descendants) of a node of which t is a rightmost descendant.

Analogously, truth of the formula $(\bot \ \underline{\text{SINCE}}\ \varphi)$ at t requires φ to be true at one of the nodes that are closest to t on its lefthand side.

The Standard Translation ST_x		
$\mathrm{ST}_x(P)$	$=$	$P'(x)$ for propositional letters P
$\mathrm{ST}_x(\top)$	$=$	$\top$
$\mathrm{ST}_x(\neg\varphi)$	$=$	$\neg\,\mathrm{ST}_x(\varphi)$
$\mathrm{ST}_x(\varphi \wedge \psi)$	$=$	$\mathrm{ST}_x(\varphi) \wedge \mathrm{ST}_x(\psi)$
$\mathrm{ST}_x(\varphi \text{ SINCE } \psi)$	$=$	$(\exists z)[z < x \wedge \mathrm{ST}_z(\psi) \wedge (\forall y)(z < y \wedge y < x \rightarrow \mathrm{ST}_y(\varphi))]$
$\mathrm{ST}_x(\varphi \text{ UNTIL } \psi)$	$=$	$(\exists z)[x < z \wedge \mathrm{ST}_z(\psi) \wedge (\forall y)(x < y \wedge y < z \rightarrow \mathrm{ST}_y(\varphi))]$
$\mathrm{ST}_x(\varphi \text{ UP-TO } \psi)$	$=$	$(\exists z)[z \prec x \wedge \mathrm{ST}_z(\psi) \wedge (\forall y)(z \prec y \wedge y \prec x \rightarrow \mathrm{ST}_y(\varphi))]$
$\mathrm{ST}_x(\varphi \text{ DOWN-TO } \psi)$	$=$	$(\exists z)[x \prec z \wedge \mathrm{ST}_z(\psi) \wedge (\forall y)(x \prec y \wedge y \prec z \rightarrow \mathrm{ST}_y(\varphi))]$

Table 6.1: The standard translation for extended **OTL** formulas

Translation into first-order logic. Table 6.1 defines the *standard translation* from extended **OTL** into first-order logic. We use the binary predicate symbols $<$ and $\prec$ to refer to *siblings* and *descendants*, respectively.[1] For every propositional letter P appearing in a modal formula we introduce a unary first-order predicate P'. The result of translating an **OTL** formula φ is a first-order formula $\mathrm{ST}_x(\varphi)$ with a single free variable x. If we fix the meaning of the predicates $<$ and $\prec$ in such a way that they are always interpreted as the sibling order and the descendant relation of a proper ordered tree in the sense of Definition 3.4, then ST_x defines a translation from **OTL** into the *monadic first-order theory of ordered trees.*

Expressive completeness. An interesting question would now be whether our language of extended **OTL** is *expressively complete* with respect to certain classes of ordered trees, that is, whether for every monadic first-order formula (of the appropriate theory) there is a suitable modal formula such that the translation of the latter is logically equivalent to the former.[2]

This question is of course related to the question of expressive completeness in linear temporal logics. The language of *since* and *until*, for instance, is known *not* to be expressively complete for the class of general linear orders (but the so-called Stavi connectives are required as well [30, 32]). We would expect to require similar operators also for the case of general ordered trees. As shown by Kamp [44], for strongly discrete orders, *since* and *until* (together with negation and conjunction) *do* form an expressively complete set of operators, so there may be a chance that our extended language is expressively complete with respect to the monadic first-order theory of discretely ordered trees. We should stress, however, that at this stage we do not have a clear picture on how the additional vertical dimension in an ordered tree (as opposed to one-dimensional linear orders) affects expressive completeness results.

[1] That is, now the symbol $\prec$ is used to refer to the *transitive* vertical relation in an ordered tree, whereas in Chapter 4 we have used the same symbol to refer to the intransitive child relation.

[2] This question has been suggested to me by Anuj Dawar (personal communication).

Decidability. The decidability proofs for **OTL** with respect to either finite binary trees or discretely ordered trees given in Chapter 5 (Theorems 5.12 and 5.14, respectively) can easily be amended to also cover the additional *until*-style operators. The same is true for decidability with respect to ordered trees where the set of nodes is required to be countable (Lemma 5.16). All that is required is an extension of the translation function TR_x^T from modal formulas into monadic second-order logic given in Table 5.1 on page 162. As we have predicates encoding the relevant accessibility relations available to us, such an extension of the translation function is straightforward (and very similar to the standard translation function into first-order logic). Here is the definition of TR_x^T for the two additional horizontal operators:

$$\begin{aligned}
\text{TR}_x^T(\varphi \text{ SINCE } \psi) &= (\exists z \in T)[\text{SIBLING}(z,x) \wedge \text{TR}_z^T(\psi) \\
&\quad \wedge (\forall y \in T)[\text{SIBLING}(z,y) \wedge \text{SIBLING}(y,x) \rightarrow \text{TR}_y^T(\varphi)]] \\
\text{TR}_x^T(\varphi \text{ UNTIL } \psi) &= (\exists z \in T)[\text{SIBLING}(x,z) \wedge \text{TR}_z^T(\psi) \\
&\quad \wedge (\forall y \in T)[\text{SIBLING}(x,y) \wedge \text{SIBLING}(y,z) \rightarrow \text{TR}_y^T(\varphi)]]
\end{aligned}$$

And here is the translation for the two vertical *until*-style operators:

$$\begin{aligned}
\text{TR}_x^T(\varphi \text{ UP-TO } \psi) &= (\exists z \in T)[\text{DESCENDANT}(z,x) \wedge \text{TR}_z^T(\psi) \\
&\quad \wedge (\forall y \in T)[\text{DESCENDANT}(z,y) \wedge \text{DESCENDANT}(y,x) \rightarrow \text{TR}_y^T(\varphi)]] \\
\text{TR}_x^T(\varphi \text{ DOWN-TO } \psi) &= (\exists z \in T)[\text{DESCENDANT}(x,z) \wedge \text{TR}_z^T(\psi) \\
&\quad \wedge (\forall y \in T)[\text{DESCENDANT}(x,y) \wedge \text{DESCENDANT}(y,z) \rightarrow \text{TR}_y^T(\varphi)]]
\end{aligned}$$

To be able to prove decidability for extended **OTL** without any restrictions on the class of admissible ordered trees, we would also have to show that the extended language cannot distinguish countable ordered trees from uncountable ones. For the language without *until*-style operators this was shown in Lemma 5.17.

While we strongly conjecture that the extended language cannot distinguish countable and uncountable models either, and hence, that extended **OTL** is decidable, the method of proof chosen for Lemma 5.17 is not immediately amenable to the extended language; so for the time being this issue remains an open question.

Interpretation as a restricted interval logic. We conclude our discussion of the proposed extensions to the modal logic of ordered trees by showing how we can use the extended logic to describe most of the relations that may hold between two time intervals. As we have already stressed earlier (see, for instance, page 45), **OTL** may be interpreted as an extended temporal logic, in which case the nodes in the underlying ordered tree represent time periods. Below we are going to make this reading more explicit by defining a number of additional modal operators inspired by the interval logic of Halpern and Shoham [37].

A time interval x may lie *before* another interval y, it may be *during* y, or the two may be *equal*. In fact, two temporal intervals x and y may be related by any of the basic interval relations shown in Figure 6.1, often referred to as *Allen relations*, due to Allen's

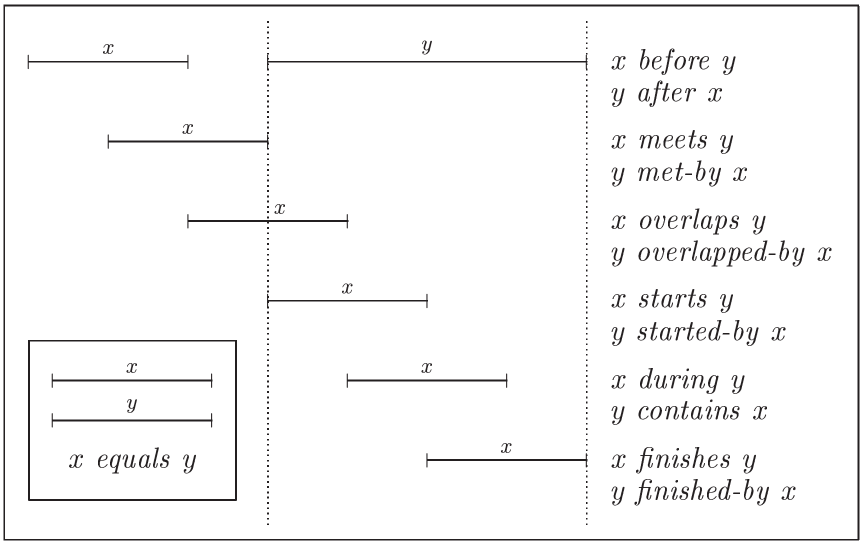

Figure 6.1: Allen's 13 interval relations

influential paper [1]. It is easy to see that they are mutually exclusive and that there are no other interval relations besides the 13 shown.

Halpern and Shoham [37] have proposed a multi-modal logic over intervals where each of the 13 interval relations represents an accessibility relation and thereby gives rise to a pair of modal operators. In this logic, a formula such as $\langle contains \rangle\, \varphi$, for instance, expresses that there exists a time interval lying during the current interval (i.e. the current interval *contains* the other) at which the formula φ is true. This is arguably the most natural definition of a modal interval logic. Unfortunately, the resulting logic is not decidable, in fact, it is not even close to being decidable. Given the importance of time intervals to many applications in artificial intelligence (such as planning, for instance) on the one hand, and the popularity of modal logics as a formal tool in many related areas (such as epistemic logics in the area of multi-agent systems) on the other, the lack of a decidable modal interval logic not only constitutes a serious gap but may also seem rather surprising. We believe that **OTL** provides a useful alternative in many cases and the following encoding of modalities based on eleven of the interval relations (excluding *overlaps* and *overlapped-by*) may further substantiate this claim.

For the purpose of the following operator definitions, let us assume that ordered trees are not degenerate, i.e. let us assume that there are no nodes in a tree with just a single child. Otherwise it would not be clear whether to interpret a single child as the same interval as its parent, or possibly as an interval occurring strictly during the interval represented by its parent. We have seen in Chapter 3 (on page 54) how to characterise non-degenerate trees syntactically.[3]

[3]Other restrictions may be imposed as well, e.g. infinite depth (if every interval should have subintervals), weak discreteness (if every interval *before* another interval should also *meet* some interval), or boundedness (if every interval with subintervals should have intervals *starting* and *finishing* that interval).

To begin with, we can use the global past and future modalities $\Diamond_{\Leftarrow}$ and $\Diamond_{\Rightarrow}$ to encode modal operators relating to the relations *before* and *after*. Two intervals x and y are in the *before*-relation iff x occurs strictly before y and their endpoints do not meet. In the context of an ordered tree this means that the node representing x needs to lie somewhere to the left of y (anywhere in the tree), but the two cannot be neighbours or be closet to each other with respect to the global left-to-right ordering. We can exclude immediate successors simply by using the global future modality twice:

$$\langle \textit{before} \rangle\, \varphi \;=\; \Diamond_{\Rightarrow}\Diamond_{\Rightarrow}\varphi \qquad\qquad \langle \textit{after} \rangle\, \varphi \;=\; \Diamond_{\Leftarrow}\Diamond_{\Leftarrow}\varphi$$

We are only going to give explicit definitions of the interval diamond-operators here. Box-operators can be defined on top of them in the usual fashion. For instance, we may define $[\textit{before}]\, \varphi = \neg\langle \textit{before} \rangle\, \neg\varphi$, in which case $[\textit{before}]\, \varphi$ would be true at a node t iff φ is true at every node in the tree somewhere to the right of t, (possibly) excluding any of the nodes closest to t on any level of the tree.

Two intervals are said to *meet* iff the endpoint of the first happens to be the starting point of the second, that is the latter immediately follows the former. In extended **OTL** this corresponds to the notion of a 'global next-operator' discussed earlier in this chapter:

$$\langle \textit{meets} \rangle\, \varphi \;=\; \bot\ \underline{\textsc{until}}\ \varphi \qquad\qquad \langle \textit{met-by} \rangle\, \varphi \;=\; \bot\ \underline{\textsc{since}}\ \varphi$$

Next we turn to the definition of diamond-operators pertaining to the interval relations *during* and *contains*. Two intervals x and y are in the *during*-relation iff the former occurs strictly during the latter, i.e. the starting point of x lies after that of y, while the endpoint of x strictly precedes that of y. If we wish to encode this relation in **OTL** we cannot simply use the ancestor modality $\Diamond_{\uparrow}$, but we have to ensure that the reference node x is neither one of the leftmost nor one of the rightmost descendants of the node y representing the larger interval. In the following definition of $\langle \textit{during} \rangle$ we make use of the propositions LEFTMOST and RIGHTMOST defined in Chapter 3 (on page 50). The central idea is to use the ancestor operator three times, twice in its reflexive form $\Diamond_{\uparrow}^{*}$ and once in its usual irreflexive form. At the two intermediate nodes we ensure that we are not in a leftmost or a rightmost node, respectively.

$$\begin{aligned} \langle \textit{during} \rangle\, \varphi \;=\;& \Diamond_{\uparrow}^{*}[\neg\textsc{leftmost} \wedge \Diamond_{\uparrow}^{*}(\neg\textsc{rightmost} \wedge \Diamond_{\uparrow}\varphi)] \;\vee \\ & \Diamond_{\uparrow}^{*}[\neg\textsc{rightmost} \wedge \Diamond_{\uparrow}^{*}(\neg\textsc{leftmost} \wedge \Diamond_{\uparrow}\varphi)] \end{aligned}$$

That is, when moving up from the current node to its ancestor representing the larger interval during which the current one occurs we are going to encounter at least one node that is not a leftmost child and at least one node that is not a rightmost child. These may in fact be the same node, and they may also be the current one (but not the one representing the larger interval).

The definition of an operator for *contains*, the inverse of *during*, is analogous:

$$\begin{aligned} \langle \textit{contains} \rangle\, \varphi \;=\;& \Diamond_{\downarrow}^{+}[\neg\textsc{leftmost} \wedge \Diamond_{\downarrow}^{*}(\neg\textsc{rightmost} \wedge \Diamond_{\downarrow}^{*}\varphi)] \;\vee \\ & \Diamond_{\downarrow}^{+}[\neg\textsc{rightmost} \wedge \Diamond_{\downarrow}^{*}(\neg\textsc{leftmost} \wedge \Diamond_{\downarrow}^{*}\varphi)] \end{aligned}$$

An interval x *starts* another interval y iff the starting points of the two coincide and x is strictly shorter than y. In an ordered tree, we would call a node 'starting' one of its ancestors iff it is either its leftmost child or the leftmost child of its leftmost child, and so on. Here are the corresponding definitions:

$$\begin{array}{rcl} \langle \textit{starts} \rangle\, \varphi & = & \text{LEFTMOST} \wedge (\text{LEFTMOST UP-TO } \varphi) \\ \langle \textit{started-by} \rangle\, \varphi & = & \text{LEFTMOST DOWN-TO } (\text{LEFTMOST} \wedge \varphi) \end{array}$$

The definitions of modal operators for *finishes* and *finished-by* are similar:

$$\begin{array}{rcl} \langle \textit{finishes} \rangle\, \varphi & = & \text{RIGHTMOST} \wedge (\text{RIGHTMOST UP-TO } \varphi) \\ \langle \textit{finished-by} \rangle\, \varphi & = & \text{RIGHTMOST DOWN-TO } (\text{RIGHTMOST} \wedge \varphi) \end{array}$$

The thirteenth interval relation is *equals*. If, for the sake of completeness, we want to include a corresponding modality into our system, we may use the following definition:

$$\langle \textit{equals} \rangle\, \varphi \quad = \quad \varphi$$

We cannot express modalities relating to either *overlaps* or *overlapped-by*, because we cannot encode the concept of two intervals overlapping in **OTL**. In that sense, our logic provides only a *restricted* interval logic, which captures *some* aspects of time intervals, but not all of them. The important aspect which it *does* capture is that we can decompose intervals into smaller parts, which is not possible for a point-based temporal logic.

As we have seen, extended **OTL** is expressive enough to distinguish between concepts such as one interval occurring strictly *during* another interval, *starting* it, or *finishing* it. However, often such level of detail will not be necessary, in which case we recommend using the original **OTL** operators.

6.2 Summary of Main Results

Contribution. It is customary to explicitly spell out the contributions made to the chosen field of study, either at the end or the beginning of a thesis. We would like to think that the main contribution of this work lies simply in the inauguration of a 'good' logic. The conception of our modal logic of ordered trees has been inspired by a real problem, namely the lack of modularity in temporal logic specifications. Our proposed solution addresses this problem by formalising the concept of 'zooming in' and this formalisation has led to the definition of a new logic, which is simple and intuitive. Relevance to applications as well as simplicity and intuitiveness are certainly important aspects of a 'good' logical system.

Technical results. That simple does not mean trivial, becomes clear when we consider the extensive technical apparatus required to master this new logic. This is where our concrete results come into play:

- Our investigations described in the later parts of Chapter 3 (culminating in Theorem 3.41) have shown that our modal logic of ordered trees is, in fact, characterised by the class of frames that we call *loosely ordered trees*.
- In Chapter 4 we have given a finite *axiomatisation* of the fragment of our logic excluding the transitive descendant modalities, which is *complete* (as shown in Theorem 4.47) for formulas at the root of a tree (and in fact for any formula associated with a specific level in a tree).
- Finally, in Chapter 5 we have proved that our logic is *decidable* (Theorem 5.18) by showing how it can be embedded into the (decidable) monadic second-order theory of two successor functions.

On top of that, particularly in the earlier parts of Chapter 3, we have demonstrated the potential of **OTL** in view of applications that require a flexible and expressive temporal representation language.

6.3 Open Problems

In this thesis, we have motivated, defined, and begun to analyse a new logic. This is only the beginning; a lot of work, both theoretical and practical, still needs to be done. In what follows, we list a number of open problems and put forward what we feel would be important areas of further investigation.

Axiomatisation. The axiomatisation of Chapter 4 is not entirely satisfactory. Ideally, we would like to be able to give a complete finite axiomatisation for our logic that also covers the transitive descendant modalities. At present, however, we do not know whether such a system exists. Our first open question thus reads as follows:

- *Is full* **OTL***, including the descendant modalities, finitely axiomatisable?*

We conjecture that, *if* there is a finite axiomatisation of full **OTL**, then it may well be the system of Table 4.1 on page 104 together with the induction axiom discussed on page 100, albeit, at this stage, we cannot offer any good advice on how to prove completeness for this system.

The natural next step would be to attempt to axiomatise the extended logic presented in the first section of this chapter:

- *Axiomatise extended* **OTL***.*

For the full extended logic, including, in particular, the DOWN-TO-operator pertaining to the transitive descendant relation, we would expect to encounter the same kind of problems as before. Apart from that, we would also have to address the usual added difficulties faced with when axiomatising *until*-style operators over flows of time that are not necessarily discrete [62].

Decidability and computational complexity. At the end of Chapter 5, we have briefly sketched an alternative method for proving decidability of **OTL** based on techniques that allow us to prune and transform a given model and thereby show that the logic has the (abstract) bounded finite model property. However, various details of the proof based on these ideas still remain to be worked out:

- *Give a direct proof for the decidability of* **OTL**.

Such a direct proof of decidability would not only help us to understand **OTL** better, but may also provide a good starting point to carry out an analysis of the computational complexity of the logic as it will provide a first approximation for upper complexity bounds for the satisfiability problem:

- *Analyse the complexity of the satisfiability problem for* **OTL**.

We certainly expect the satisfiability problem of **OTL** to be at least PSPACE-hard, because that of propositional linear temporal logic (for discrete flows of time and even without *since* and *until*, but with a next-operator) is [66]. Given the connections between **OTL** and **PDL** (both include a transitive closure operator with respect to a non-linear accessibility relation), the satisfiability problem for our logic is likely to be even EXPTIME-hard, because that of **PDL** is [8, 24]. At this stage, we do not have any concrete conjectures regarding upper complexity bounds. Our preliminary construction to establish a bounded finite model property suggests that 'small' models will be of exponential size in *both* dimensions of a tree, even for discretely ordered trees. This means that the encoding of the satisfiability problem into a (non-deterministic) Turing machine requiring only linear space, which Sistla and Clarke [66] use to establish their PSPACE upper bound for propositional linear temporal logic, would not be applicable anymore in the case of our logic.

In the first section of this chapter, we have already conjectured that the extended modal logic of ordered trees is very likely to be a decidable logic as well. Adapting the main part of our decidability proof, namely the embedding into the monadic second-order theory of two successor functions, is straightforward. However, to actually complete a decidability proof along the lines of that given in Chapter 5, we would still have to show that any extended formula with an ordered tree model will also have a *countable* ordered tree model. Of course, we may search for a different kind of proof:

- *Show that extended* **OTL** *is decidable.*

As for the original logic without *until*-style operators, a direct decidability proof (rather than a reduction to a decidable second-order theory) would be of particular interest. A related issue is again the computational complexity of the extended logic:

- *Analyse the complexity of the satisfiability problem for extended* **OTL**.

Expressive completeness. We have already briefly touched upon the issue of expressive completeness for ordered tree logics earlier in this chapter:

- *Is extended* **OTL** *expressively complete for discretely ordered trees? If not, is there an expressively complete set of modalities? How about other types of sibling orders?*

The known negative results for certain branching structures (such as the class of all partially ordered flows of time) seem not to be of immediate relevance here, because in **OTL** the additional order declared over siblings allows us to move between nodes more freely. For instance, it is possible in the monadic first-order theory of ordered trees to express that a tree has at least n distinct nodes using only two variable names, because we can use the descendant relation and the definable global left-to-right ordering to 'comb' through all nodes in a tree without having to keep track of the nodes already encountered, as would be the case in a simple tree. (This is a typical example used to show the failure of expressive completeness for a given theory; see Gabbay *et al.* [30] on the subject of *finite H-dimension.*)

Automated reasoning. Our decidability proof does not lend itself to an efficient decision procedure for **OTL**, because the monadic second-order theory of two successor functions, in which we have embedded our logic, is of such a high computational complexity. We consider the development of an implementable deduction system for **OTL** a difficult but important problem:

- *Devise efficient decision procedures for (extended)* **OTL***.*

As a matter of personal preference, we suggest an investigation into options for a decision procedure based on semantic Tableaux. Devising sound expansion rules for a Tableaux-based deduction system seems unproblematic; the crux of the matter lies in the formulation of appropriate termination conditions. Wolper's Tableaux system for propositional linear temporal logic [73] provides a good starting point and some of the recent work on Tableaux for expressive description logics may also be relevant (see, for instance, the survey [4] and references therein).

While, from a logician's point of view, deduction may be regarded as the central problem of automated reasoning, it is in fact the availability of highly efficient model checking algorithms (particularly for branching time logics) that have contributed significantly to the recent success of temporal logics in the area of systems specification and verification [17]. The following thus seems a promising area of activity:

- *Devise efficient model checking algorithms for (extended)* **OTL***.*

We would expect a combination of techniques developed for linear and branching time temporal logics to be applicable also in the case of **OTL**, but have not investigated this issue to date.

Applications. Although *inspired* by applications, the work reported in this thesis is mostly (some would argue, entirely) theoretical. We do believe that this is the right order of approaching difficult problems, but eventually we should take the fruits of our efforts back into the real world and seek to apply our findings to real problems. For instance:

- *Use* **OTL** *for the modular specification of practical systems.*

More logics. Our final suggestion for further work in the area is again — at least initially — of a theoretical nature. The modal logic of ordered trees has originally been conceived as the result of adding a zoom to propositional linear temporal logics. The same idea may usefully be applied to other temporal logics as well:

- *Combine the idea of 'zooming in' with branching time logics.*

The result would be a logic that combines two modal systems built on trees: one for the (branching) flow of time and one to model changes in the level of abstraction.

So many questions, so little time . . .

Appendices

Appendix A

Relations and Orders

This appendix serves as a references to definitions of some of the basic concepts regarding relations and linear orders referred to throughout this thesis.

A.1 Relations

Definition A.1 (Relational algebra operations) *Let R, R_1, and R_2 be binary relations over a common domain T. We define the standard operations of relational algebra as follows:*

(1) Intersection: $(t_1, t_2) \in R_1 \cap R_2$ *iff* $(t_1, t_2) \in R_1$ *and* $(t_1, t_2) \in R_2$.

(2) Union: $(t_1, t_2) \in R_1 \cup R_2$ *iff* $(t_1, t_2) \in R_1$ *or* $(t_1, t_2) \in R_2$.

(3) Complement: $(t_1, t_2) \in \overline{R}$ *iff* $(t_1, t_2) \notin R$.

(4) Difference: $(t_1, t_2) \in R_1 \backslash R_2$ *iff* $(t_1, t_2) \in R_1$ *but* $(t_1, t_2) \notin R_2$.

(5) Inverse: $(t_1, t_2) \in R^{-1}$ *iff* $(t_2, t_1) \in R$.

(6) Composition: $(t_1, t_3) \in R_1 \circ R_2$ *iff* $(t_1, t_2) \in R_1$ *and* $(t_2, t_3) \in R_2$ *for some* $t_2 \in T$.

(7) Iteration: $(t_1, t_2) \in R^n$ *iff* $(t_1, t_2) \in R \circ R^{n-1}$ *for* $n \in \mathbb{N}$ *and* $(t_1, t_2) \in R^0$ *iff* $t_1 = t_2$.

Definition A.2 (Basic properties of relations) *Let R be a binary relation over some domain T. We define the following basic properties of relations:*

(1) R is called reflexive iff $(t, t) \in R$ for all $t \in T$.

(2) R is called irreflexive iff $(t, t) \notin R$ for all $t \in T$.

(3) R is called symmetric iff $(t_1, t_2) \in R$ implies $(t_2, t_1) \in R$ for all t_1, $t_2 \in T$.

(4) R is called transitive iff $(t_1, t_2) \in R$ and $(t_2, t_3) \in R$ implies $(t_1, t_3) \in R$ for all t_1, t_2, $t_3 \in T$.

(5) R is called euclidean iff $(t_1, t_2) \in R$ and $(t_1, t_3) \in R$ implies $(t_2, t_3) \in R$ for all t_1, t_2, $t_3 \in T$.

(6) R is called serial iff for all $t_1 \in T$ there exists a $t_2 \in T$ such that $(t_1, t_2) \in R$.

(7) R is called connected iff $(t_1, t_2) \in R$ or $t_1 = t_2$ or $(t_2, t_1) \in R$ for all t_1, $t_2 \in T$.

We also sometimes refer to the notion of a relation R being *connected* over a particular subset of $T \times T$ like, for instance, the set of pairs that stand in some other relation R'. This is the case iff $(t_1, t_2) \in R$ or $t_1 = t_2$ or $(t_2, t_1) \in R$ for all $(t_1, t_2) \in R'$.

Definition A.3 (Transitive closure) *Let R be a binary relation over a domain T. The transitive closure R^+ of R is the smallest binary relation over T with $R \subseteq R^+$ that is transitive. The reflexive transitive closure R^* of R is the smallest binary relation over T with $R^+ \subseteq R^*$ that is reflexive.*

We sometimes refer to R^+ as the *non-reflexive* transitive closure of R, as opposed to the reflexive transitive closure R^*.

As is well-known, whenever we have $(t_1, t_2) \in R^+$, then there exists an $n \in \mathbb{N}$ such that $(t_1, t_2) \in R^n$ holds.

A.2 Linear Orders

Definition A.4 (Strict linear orders) *A strict linear order is a pair $(T, <)$ where T is a set and $<$ is a binary relation over T that is irreflexive, transitive, and connected.*

Strict linear orders are also called *strict total orders.* Next we are going to define what it means for a strict linear order to be either *discrete* or *dense.* In the case of discreteness, two different definitions can be found in the literature. Here, we shall use both of them and distinguish between *weak* and *strong discreteness.*

Definition A.5 (Weakly discrete orders) *A strict linear order $(T, <)$ is called weakly discrete iff for any $t \in T$ we have a closest point to either side of t (if any).*

Definition A.6 (Strongly discrete orders) *A strict linear order $(T, <)$ is called strongly discrete iff for any t_1, $t_2 \in T$ there are only finitely many $t \in T$ such that $t_1 < t$ and $t < t_2$ hold.*

Observe that in any strongly discrete order for every element we have a closest element to either side (with the exception of a possible first or last element in the order), that is, strong discreteness is indeed (just as the names suggest) at least as strong a requirement as weak discreteness. The converse, on the other hand, does not hold, that is, having both a closest left and a closest right neighbour is not a sufficient condition for a strict linear order to be strongly discrete. To see this, consider the following set of rational numbers:

$$T \;=\; \{\tfrac{1}{n} \mid n \in \mathbb{Z} \setminus \{0\}\} \;=\; \{-1, -\tfrac{1}{2}, -\tfrac{1}{3}, -\tfrac{1}{4}, -\tfrac{1}{5}, \ldots, \tfrac{1}{5}, \tfrac{1}{4}, \tfrac{1}{3}, \tfrac{1}{2}, 1\}$$

Then $(T, <)$, with the usual ordering relation $<$ over rational numbers, is *not* a discrete order according to Definition A.6, because there are infinitely many elements between -1 and 1. This is so despite the fact that every element in our set T (apart from the endpoints -1 and 1) has got a closest element to both their left and right.

Definition A.5 appears to be the 'standard' definition for discreteness. Most definitely, one would want to call a *point* discrete in a structure iff it has a closest point to both its left and right.[1] In the context of temporal logics, on the other hand, Definition A.6 is often the preferred option (see Goldblatt [35], for instance). It certainly provides a better match for our intuitions about time: one would want to be able to reach any future point in time after a finite number of elementary steps (provided we admit the concept of an elementary step in the first place). However, there are exceptions: Gabbay *et al.* [30] use the term discreteness in the sense of Definition A.5. In this thesis, when referring to a discrete order, without specifically qualifying it as being either weak or strong, we mean a strongly discrete order in the sense of Definition A.6.

Definition A.7 (Dense orders) *A strict linear order $(T, <)$ is called dense iff for any $t_1, t_2 \in T$ with $t_1 < t_2$ there exists a $t \in T$ such that $t_1 < t$ and $t < t_2$.*

Definition A.8 (Total orders) *A total order is a pair $(T, \leq)$ where T is a set and $\leq$ is a binary relation over T that is reflexive, transitive, and connected.*

In view of Definition A.4 we may also call a total order a *non-strict linear order*. The following proposition gives a criterion for finiteness of a total order.[2]

Proposition A.9 (Finite total orders) *Let $(T, \leq)$ be a total order. Every non-empty subset of T has got a minimal and a maximal element with respect to $\leq$ iff T is finite.*

Proof. '$\Rightarrow$': We are going to show the converse, i.e. that T being infinite implies that T must have a non-empty subset that does not have both a minimal and a maximal element. Assume T is infinite. Then it must have either an infinite ascending chain or an infinite descending chain, i.e. T must contain an infinite sequence of elements $t_0, t_1, t_2, \ldots$ such that either $t_0 \leq t_1 \leq t_2 \leq \cdots$ or $\cdots \leq t_2 \leq t_1 \leq t_0$. But then $\{t_0, t_1, t_2, \ldots\}$ is a subset of T that lacks either a maximal or a minimal element, i.e. we are done.

'$\Leftarrow$': If T is finite, then also every non-empty subset of T must be finite and will certainly have both a minimal and a maximal element. $\Box$

[1] In other words, what we call a weakly discrete order here is in fact a strict linear order in which every point is discrete.

[2] A total order $(T, \leq)$ is called a *well-ordering* iff every non-empty subset of T has got a minimal element with respect to $\leq$. That is, an alternative way of phrasing Proposition A.9 would be to say that a total order $(T, \leq)$ is finite iff T is well-ordered by both $\leq$ and its inverse.

Appendix B

Derivations

In this appendix we prove a number of derivable theorems of the axiomatic proof system presented in Chapter 4. Axioms and rules of this system are listed in Table 4.1 on page 104.

B.1 Proof of Lemma 4.8

The following are derivations for the theorems listed in Lemma 4.8 on page 111.

Part 1. $\vdash \neg(\Box\bot \wedge \odot A)$

(1)	$\odot A \to \Diamond A$	from (M3)
(2)	$\odot A \to \neg \Box\neg A$	from (1) and (D3)
(3)	$\Box\neg A \to \neg \odot A$	from (2)
(4)	$\bot \to \neg A$	propositional tautology
(5)	$\Box(\bot \to \neg A)$	from (4) by generalisation
(6)	$\Box\bot \to \Box\neg A$	from (5) and (K4)
(7)	$\Box\bot \to \neg \odot A$	from (3) and (6)
(8)	$\neg(\Box\bot \wedge \odot A)$	from (7)

Part 2. $\vdash \diamondsuit\odot A \to A$

(1)	$\odot\neg A \to \neg \odot A$	from (F3)
(2)	$\boxdot(\odot\neg A \to \neg \odot A)$	from (1) by generalisation
(3)	$\boxdot\odot\neg A \to \boxdot\neg \odot A$	from (2) and (K4)
(4)	$\neg \boxdot\neg \odot A \to \neg \boxdot\odot\neg A$	from (3)
(5)	$\diamondsuit\odot A \to \neg \boxdot\odot\neg A$	from (4) and (D4)
(6)	$\neg A \to \boxdot \odot\neg A$	from (B8)
(7)	$\neg \boxdot \odot\neg A \to A$	from (6)
(8)	$\diamondsuit\odot A \to A$	from (5) and (7)

Part 3. $\vdash \Diamond_{\downarrow}\Box^{\uparrow} A \to A$

(1)	$\neg A \to \Box_{\downarrow}\bigcirc^{\uparrow}\neg A$	from (B8)
(2)	$\bigcirc^{\uparrow}\neg A \to \Diamond^{\uparrow}\neg A$	from (M3)
(3)	$\Box_{\downarrow}(\bigcirc^{\uparrow}\neg A \to \Diamond^{\uparrow}\neg A)$	from (2) by generalisation
(4)	$\Box_{\downarrow}\bigcirc^{\uparrow}\neg A \to \Box_{\downarrow}\Diamond^{\uparrow}\neg A$	from (3) and (K4)
(5)	$\neg A \to \Box_{\downarrow}\Diamond^{\uparrow}\neg A$	from (1) and (4)
(6)	$\neg \Box_{\downarrow}\Diamond^{\uparrow}\neg A \to A$	from (5)
(7)	$\Diamond_{\downarrow}\neg \Diamond^{\uparrow}\neg A \to A$	from (6) and (D4)
(8)	$\Diamond_{\downarrow}\Box^{\uparrow} A \to A$	from (7) and (D3)

Part 4. $\vdash \Box_{\downarrow} A \to \Box_{\downarrow}\Box_{\rightarrow} A$

(1)	$\Diamond_{\rightarrow}\neg A \to \bigcirc^{\uparrow}\Diamond_{\downarrow}\neg A$	from (X1)
(2)	$\neg \bigcirc^{\uparrow}\Diamond_{\downarrow}\neg A \to \neg \Diamond_{\rightarrow}\neg A$	from (1)
(3)	$\neg \bigcirc^{\uparrow}\Diamond_{\downarrow}\neg A \to \Box_{\rightarrow} A$	from (2) and (D2)
(4)	$\Box_{\downarrow}(\neg \bigcirc^{\uparrow}\Diamond_{\downarrow}\neg A \to \Box_{\rightarrow} A)$	from (3) by generalisation
(5)	$\Box_{\downarrow}\neg \bigcirc^{\uparrow}\Diamond_{\downarrow}\neg A \to \Box_{\downarrow}\Box_{\rightarrow} A$	from (4) and (K4)
(6)	$\Diamond_{\downarrow}\bigcirc^{\uparrow}\Diamond_{\downarrow}\neg A \to \Diamond_{\downarrow}\neg A$	from Part (2) of this lemma
(7)	$\Box_{\downarrow} A \to \Box_{\downarrow}\neg \bigcirc^{\uparrow}\Diamond_{\downarrow}\neg A$	from (6) and (D4)
(8)	$\Box_{\downarrow} A \to \Box_{\downarrow}\Box_{\rightarrow} A$	from (5) and (7)

Part 5. $\vdash (\Box_{\rightarrow} A \wedge \bigcirc_{\rightarrow}\top) \to \bigcirc_{\rightarrow} A$

(1)	$\bigcirc_{\rightarrow}\neg A \to \Diamond_{\rightarrow}\neg A$	from (M2)
(2)	$\Box_{\rightarrow} A \to \neg \bigcirc_{\rightarrow}\neg A$	from (1) and (D2)
(3)	$(\Box_{\rightarrow} A \wedge \bigcirc_{\rightarrow}\top) \to \bigcirc_{\rightarrow} A$	from (2) and (F2)

Part 6. $\vdash \bigcirc_{\rightarrow}\Box_{\rightarrow} A \to \Box_{\rightarrow}\Box_{\rightarrow} A$

(1)	$(\Diamond_{\rightarrow}\Diamond_{\rightarrow}\neg A \wedge \bigcirc_{\rightarrow}\top) \to \bigcirc_{\rightarrow}\Diamond_{\rightarrow}\neg A$	from (M2)
(2)	$(\neg \Box_{\rightarrow}\Box_{\rightarrow} A \wedge \bigcirc_{\rightarrow}\top) \to \bigcirc_{\rightarrow}\neg \Box_{\rightarrow} A$	from (1) and (D2)
(3)	$(\neg \bigcirc_{\rightarrow}\neg \Box_{\rightarrow} A \wedge \bigcirc_{\rightarrow}\top) \to \Box_{\rightarrow}\Box_{\rightarrow} A$	from (2)
(4)	$\bigcirc_{\rightarrow}\Box_{\rightarrow} A \to (\neg \bigcirc_{\rightarrow}\neg \Box_{\rightarrow} A \wedge \bigcirc_{\rightarrow}\top)$	from (F2)
(5)	$\bigcirc_{\rightarrow}\Box_{\rightarrow} A \to \Box_{\rightarrow}\Box_{\rightarrow} A$	from (3) and (4)

Part 7. $\vdash \bigcirc_{\rightarrow}(A \wedge \Box_{\rightarrow} A) \to \Box_{\rightarrow} A$

(1)	$(\Diamond_{\rightarrow}\neg A \wedge \bigcirc_{\rightarrow}\top) \to (\bigcirc_{\rightarrow}\neg A \vee \bigcirc_{\rightarrow}\Diamond_{\rightarrow}\neg A)$	from (M2)
(2)	$(\neg \bigcirc_{\rightarrow}\neg A \wedge \neg \bigcirc_{\rightarrow}\Diamond_{\rightarrow}\neg A) \to (\neg \Diamond_{\rightarrow}\neg A \vee \neg \bigcirc_{\rightarrow}\top)$	from (1)
(3)	$(\neg \bigcirc_{\rightarrow}\neg A \wedge \neg \bigcirc_{\rightarrow}\neg \Box_{\rightarrow} A) \to (\Box_{\rightarrow} A \vee \neg \bigcirc_{\rightarrow}\top)$	from (2) and (D2)
(4)	$(\neg \bigcirc_{\rightarrow}\neg A \wedge \neg \bigcirc_{\rightarrow}\neg \Box_{\rightarrow} A \wedge \bigcirc_{\rightarrow}\top) \to \Box_{\rightarrow} A$	from (3)
(5)	$(\bigcirc_{\rightarrow} A \wedge \bigcirc_{\rightarrow}\Box_{\rightarrow} A) \to \Box_{\rightarrow} A$	from (4) and (F2)
(6)	$\bigcirc_{\rightarrow}(A \wedge \Box_{\rightarrow} A) \to \Box_{\rightarrow} A$	from (5) by distribution

Bibliography

The numbers in brackets at the end of each bibliographic entry refer to the pages where that item has been cited in the text.

[1] J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983. (190)

[2] J. F. Allen. Towards a General Theory of Action and Time. *Artificial Intelligence*, 23:123–154, 1984. (55, 58)

[3] L. Åquist. Deontic Logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 8. Kluwer Academic Publishers, 2nd edition, 2002. (28)

[4] F. Baader and U. Sattler. Tableau Algorithms for Description Logics. In R. Dyckhoff, editor, *Automated Reasoning with Tableaux and Related Methods*, number 1847 in LNAI, pages 1–18. Springer-Verlag, 2000. (37, 195)

[5] H. Barringer, R. Kuiper, and A. Pnueli. Now You May Compose Temporal Logic Specifications. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, pages 51–63. ACM, 1984. (9)

[6] H. Barringer, R. Kuiper, and A. Pnueli. A Really Abstract Concurrent Model and its Temporal Logic. In *Proceedings of the 13th Annual ACM Symposium on Principles of Programming Languages*, pages 173–183. ACM, 1986. (9)

[7] O. Becker. Zur Logik der Modalitäten. *Jahrbuch für Philosophie und Phänomenologische Forschung*, 6:497–548, 1930. (30)

[8] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001. (19, 26, 31, 37, 45, 50, 62, 63, 77, 117, 143, 194)

[9] P. Blackburn and W. Meyer-Viol. Linguistics, Logic and Finite Trees. *Logic Journal of the IGPL*, 2:3–29, 1994. (8, 53, 97, 144, 147, 159, 163)

[10] P. Blackburn, W. Meyer-Viol, and M. de Rijke. A Proof System for Finite Trees. In H. Kleine Büning, editor, *Computer Science Logic*, volume 1092 of *LNCS*, pages 86–105. Springer-Verlag, 1996. (8)

[11] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer-Verlag, 1997. (24)

[12] G. Cantor. Beiträge zur Begründung der Transfiniten Mengenlehre. *Mathematische Annalen*, 46:481–512, 1895. (171)

[13] A. Chagrov and M. Zakharyaschev. *Modal Logic*. Oxford University Press, 1997. (37, 62, 124)

[14] C. C. Chang and H. J. Keisler. *Model Theory*. Elsevier Science Publishers, 3rd edition, 1990. (176)

[15] B. F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980. (27, 30)

[16] A. Church. An Unsolvable Problem of Elementary Number Theory. *American Journal of Mathematics*, 58:345–363, 1936. (13)

[17] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999. (37, 195)

[18] E. M. Clarke and B.-H. Schlingloff. Model Checking. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 2. Elsevier Science Publishers, 2001. (37, 103)

[19] U. Endriss. Adding a Zoom to Linear Temporal Logic. In T. Walsh, editor, *Proceedings of the 9th Workshop on Automated Reasoning*. AISB, April 2002. (181, 182)

[20] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995. (37)

[21] R. Feys. Les Logiques Nouvelles des Modalités (Part 1). *Revue Néoscholastique de Philosophie*, 40:517–553, 1937. (27)

[22] R. Feys. Les Logiques Nouvelles des Modalités (Part 2). *Revue Néoscholastique de Philosophie*, 41:217–252, 1938. (27)

[23] J. L. Fiadeiro and T. Maibaum. Sometimes "tomorrow" is "sometime": Action Refinement in a Temporal Logic of Objects. In D. Gabbay and H. J. Ohlbach, editors, *Temporal Logic*, volume 827 of *LNAI*, pages 48–66. Springer-Verlag, 1994. (9, 55)

[24] M. J. Fischer and R. E. Ladner. Propositional Dynamic Logic of Regular Programs. *Journal of Computer and System Sciences*, 18:194–211, 1979. (194)

[25] M. Fisher. A Resolution Method for Temporal Logic. In J. Mylopoulos and R. Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 99–104. Morgan Kaufmann Publishers, 1991. (37)

[26] M. Fitting and R. L. Mendelsohn. *First-Order Modal Logic.* Kluwer Academic Publishers, 1998. (37)

[27] D. Gabbay. Decidability Results in Non-Classical Logics. Part I. *Annals of Mathematical Logic*, 8:237–295, 1975. (159)

[28] D. Gabbay. *Investigations in Modal and Tense Logics with Applications to Problems in Philosophy and Linguistics.* Reidel Publishing Company, 1976. (159)

[29] D. Gabbay. An Irreflexivity Lemma with Applications to Axiomatizations of Conditions on Tense Frames. In U. Mönnich, editor, *Aspects of Philosophical Logic*, pages 67–89. Reidel Publishing Company, 1981. (144)

[30] D. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects*, volume 1. Oxford University Press, 1994. (35, 36, 37, 144, 169, 171, 176, 188, 195, 201)

[31] D. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-dimensional Modal Logics: Theory and Applications.* Forthcoming. (37, 46, 47, 181)

[32] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the Temporal Analysis of Fairness. In *Proceedings of the 7th ACM Symposium on Principles of Programming Languages*, pages 163–173. ACM, 1980. (35, 188)

[33] D. Gabbay, M. Reynolds, and M. Finger. *Temporal Logic: Mathematical Foundations and Computational Aspects*, volume 2. Oxford University Press, 2000. (32, 37)

[34] J. W. Garson. Quantification in Modal Logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 3. Kluwer Academic Publishers, 2nd edition, 2001. (37)

[35] R. Goldblatt. *Logics of Time and Computation.* CSLI, 2nd edition, 1992. (37, 40, 51, 62, 97, 100, 101, 145, 201)

[36] R. Goldblatt. Mathematical Modal Logic: a View of its Evolution. In D. van Dalen, J. Dawson, and A. Kanamori, editors, *A History of Mathematical Logic.* Forthcoming. (37)

[37] J. Y. Halpern and Y. Shoham. A Propositional Modal Logic of Time Intervals. *Journal of the ACM*, 38(4):935–962, 1991. (8, 59, 181, 189, 190)

[38] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic.* MIT Press, 2000. (145)

[39] L. Henkin. The Completeness of the First-Order Functional Calculus. *Journal of Symbolic Logic*, 14(3):159–166, 1949. (113)

[40] D. Hilbert and W. Ackermann. *Principles of Mathematical Logic*. Chelsea Publishing Company, 1950. Translation of the 2nd German edition of 1938. (98)

[41] G. E. Hughes and M. J. Cresswell. *An Introduction to Modal Logic*. Methuen, 1968. (19)

[42] G. E. Hughes and M. J. Cresswell. *A New Introduction to Modal Logic*. Routledge, 1995. (31, 37, 115)

[43] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, 2000. (37)

[44] J. A. W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, Department of Philosophy, University of California at Los Angeles, 1968. (35, 188)

[45] S. A. Kripke. Semantical Analysis of Modal Logic I: Normal Propositional Calculi. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963. (19, 25)

[46] L. Lamport. Specifying Concurrent Program Modules. *ACM Transactions on Programming Languages and Systems*, 5(2):190–222, 1983. (55)

[47] L. Lamport. What Good Is Temporal Logic? In R. E. A. Mason, editor, *Information Processing*, volume 83, pages 657–668. Elsevier Science Publishers, 1983. (9, 35)

[48] E. J. Lemmon (in collaboration with D. Scott). *The "Lemmon Notes". An Introduction to Modal Logic*, volume 11 of *American Philosophical Quarterly Monograph Series*. Blackwell Publishers, 1977. Printed version of an unpublished draft from 1966, edited by K. Segerberg. (113)

[49] C. I. Lewis and C. H. Langford. *Symbolic Logic*. The Century Co., New York, 1932. (27, 28, 30)

[50] D. Makinson. On Some Completeness Theorems in Modal Logic. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 12:379–384, 1966. (113)

[51] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, 1992. (36, 37)

[52] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, 1995. (37)

[53] M. Marx and Y. Venema. *Multi-dimensional Modal Logic*. Kluwer Academic Publishers, 1997. (37)

[54] D. V. McDermott. A Temporal Logic for Reasoning about Processes and Plans. *Cognitive Science*, 6:101–155, 1982. (55)

[55] J. C. C. McKinsey. On the Syntactical Construction of Systems of Modal Logic. *Journal of Symbolic Logic*, 10(3):83–94, 1945. (31)

[56] A. Montanari. *Metric and Layered Temporal Logic for Time Granularity*. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, 1996. (8)

[57] M. Mortimer. On Languages with Two Variables. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 21:135–140, 1975. (24)

[58] B. Moszkowski. A Temporal Logic for Multilevel Reasoning about Hardware. *IEEE Computer*, 18(2):10–19, 1985. (8)

[59] H. J. Ohlbach. A Resolution Calculus for Modal Logics. In E. L. Lusk and R. A. Overbeek, editors, *Automated Deduction (CADE-9)*, volume 310 of *LNCS*, pages 500–516, 1988. (37)

[60] A. Pnueli. The Temporal Logic of Programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE, 1977. (8)

[61] M. O. Rabin. Decidability of Second-Order Theories and Automata on Infinite Trees. *Transactions of the AMS*, 141:1–35, 1969. (147, 152, 164, 171)

[62] M. Reynolds. An Axiomatization for Until and Since over the Reals without the IRR Rule. *Studia Logica*, 51:165–194, 1992. (193)

[63] D. Scott. A Decision Method for Validity of Sentences in Two Variables. *Journal of Symbolic Logic*, 27:477, 1962. (24)

[64] K. Segerberg. Modal Logics with Linear Alternative Relations. *Theoria*, 36:301–322, 1970. (77)

[65] Y. Shoham. Temporal Logics in AI: Semantical and Ontological Considerations. *Artificial Intelligence*, 33:89–104, 1987. (55, 56, 58, 59)

[66] A. P. Sistla and E. M. Clarke. The Complexity of Propositional Linear Temporal Logics. *Journal of the ACM*, 32(3):733–749, 1985. (181, 194)

[67] W. Thomas. Languages, Automata, and Logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3. Springer-Verlag, 1997. (147, 152)

[68] A. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936. (13)

[69] J. van Benthem. Correspondence Theory. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 3. Kluwer Academic Publishers, 2nd edition, 2001. (29, 31)

[70] J. van Benthem. Modal Logic. In D. Jacquette, editor, *A Companion to Philosophical Logic*, pages 391–409. Blackwell Publishers, 2001. (37)

[71] J. van Benthem and K. Doets. Higher-order Logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 1. Kluwer Academic Publishers, 2nd edition, 2001. (148)

[72] D. van Dalen. Intuitionistic Logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 5. Kluwer Academic Publishers, 2nd edition, 2002. (59)

[73] P. Wolper. The Tableau Method for Temporal Logic: An Overview. *Logique et Analyse*, 28(110–111):119–136, 1985. (37, 195)

List of Figures

1.1 Adding a zoom to linear temporal logic 5

2.1 Three models 21

3.1 A simple ordered tree model 42
3.2 Scope of the global past and future modalities 49
3.3 Unwinding (bulldozing) cycles 78
3.4 Unravelling a bulldozed tree 80
3.5 Bulldozing clusters 87

4.1 Proof detail for Lemma 4.40 133
4.2 Proof detail for Lemma 4.41 134

5.1 Encoding $\mathbf{S\omega S}$ in **S2S** 156
5.2 Encoding the infinite unbounded discretely ordered tree in $\mathbf{S\omega S}$ 165
5.3 Encoding dense orders in **S2S** 170
5.4 Encoding the infinite (densely) ordered tree in **S4S** 172

6.1 Allen's 13 interval relations 190

List of Tables

2.1 Modes of truth 17
2.2 Basic correspondences 30

4.1 Axioms and rules 104

5.1 Translating **OTL** formulas into monadic second-order logic 162

6.1 The standard translation for extended **OTL** formulas 188

Index

A

accessibility relation . . . 19
action . . . 55
actual truth . . . 15
alethic logic . . . 16
Allen relations . . . 189
alphabet . . . 148
ancestor . . . 41
Ancestor Existence Lemma . . . 121
applications
    artificial intelligence . . . 190
    computational linguistics . . . 53
    e-commerce . . . 3
    planning . . . 55
    software engineering . . . 8, 35, 55
assignment . . . 150
assignment variant . . . 150
automated reasoning . . . 195
awareness . . . 118, 126
axiom . . . 24, 104

B

back condition . . . 63
belief . . . 16
bidirectional p-morphism . . . 63
BINARY . . . 53
binary tree . . . 53
BOUNDED . . . 51
bounded branching . . . 51
bounded finite model property . . . 181
bounded morphism . . . 62
box-operator . . . 17, 40
brackets . . . 18
branch . . . 41
branching
    bounded . . . 51
    dense . . . 51
    finite . . . 53
    strongly discrete . . . 51
    weakly discrete . . . 51
branching time . . . 32, 196
BRANCHING(n) . . . 53
bulldozing . . . 77
    clusters . . . 86, 87
    cycles (unwinding) . . . 78

C

canonical child relation . . . 118
canonical frame . . . 128, 135
canonical model . . . 113, 118, 135
canonical neighbour relation . . . 126
canonical sibling relation . . . 126
Cantor's Theorem . . . 171
CHILD . . . 160, 166, 173
child . . . 41
    leftmost . . . 50
    rightmost . . . 50
Child Existence Lemma . . . 121
Church-Rosser property . . . 181
closed formula . . . 151
cluster . . . 74, 76, 177
cluster elimination . . . 86
commutativity . . . 46, 181
compactness . . . 107, 140
complement . . . 199
completeness . . . 24, 106, 113, 138
complexity . . . 14, 194
composition of relations . . . 199
connectedness . . . 200
consistent . . . 24, 105

conversely functional 70, 124
correspondence theory 29
correspondences 29, 50
countable ordered tree model 176
countable sets 169
cycle 74, 76
cycle elimination 77

D

decidability 22, 47, 180
  extended logic 189
  for countable ordered trees 175
  for discretely ordered trees 168
  for finite binary trees 163
  second-order theories 152
defect 76
definable operators 20, 33, 44, 151
definable predicates 152
degenerate tree 54
DENSE 51
dense branching 51
density 201
density axiom 101
deontic logic 28
depth
  finite 54
derivation 97, 105
derived theorem 103
DESCENDANT 160, 166, 173
descendant 41
Descendant Existence Lemma 123
diamond-operator 17, 40
difference of two relations 199
difference operator (⟐,⊞) 49
DISCRETE 52
discrete branching 51
discretely ordered tree 52, 164
discreteness
  strong 200
  weak 200
distribution 108, 110
distribution axiom 25, 98
double negation 57
DOWN-HERED 56
DOWN-TO 186
downward-hereditary 56, 58
doxastic logic 16
dual of a next-operator 103
duality 16
duality axiom 24, 98

E

empty path 81
empty word (Λ) 148
epistemic logic 16
equivalence class 41
euclidean axiom 30
euclidean relation 200
event 55, 59
Existence Lemma 114, 121, 127
expressive completeness 35, 188, 195
expressiveness 14, 95
extended formula 185
extended temporal logic 45

F

fact 55
falsum ($\bot$) 18
finite axiomatisation 145
finite binary tree 53, 159
FINITE-BRANCHING 53
finite branching 53
FINITE-DEPTH 54
finite depth 54, 58
finite model property 181
finite tree 53
first-order definability 31
Fischer-Ladner closure 143
flow of time 33
formula 18, 39
  closed 151
  extended 185
  second-order 149
frame

basic modal logic 19
general 61
intended 61
linear temporal 33
frame validity 62
FULLTREE 156, 166, 173
functional 68
conversely 70, 124
functional completeness 20, 34
functionality axiom 99
fusion 46

G

Gabbay-style rule 144
general frame 61
general model 60, 61
generalisation rule 25, 102
generated canonical frame 135
generated canonical model 135
generated canonical tree 124
generated canonical valuation 136
generated set of worlds 124
generated subframe 124
GESTALT 59
gestalt proposition 59
global future modality (⬗,◫) 48
global past modality (⬖,◫) 48
global next-operator 187
global truth 22, 45, 62
global *until*-style operators 187
granularities 8

H

H-dimension 195
HOLDS 55, 58
HOM 56
homogeneity 56
homogeneous proposition 56, 58
homomorphic condition 63
horizontal accessibility relation 118, 126

I

immediate prefix of a word 164
implication
strict (⇒) 58
inconsistent 105
induction axiom 100, 141
inference rule 24, 102, 104
infinite binary tree 148
infinite branch 54
initial subtree 161, 167, 174
intended frame 61
intended model 61
interaction 46, 140
interaction axiom 102, 129, 140
internal node 41
intersection 199
interval 45
interval logic 8, 189
interval relations 189
intuitionistic logic 58
inverse axiom 99
inverse back condition 63
inverse relation 199
irreflexive point 74
irreflexivity 69, 199
irreflexivity rule 144
iterated modality 50, 109
iterated relations 199
ITL 8

K

K (axiom) 25, 98
knowledge 16
Kripke model 19

L

LEAF 50, 54
leaf 41, 50
left sibling 41
LEFTMOST 50, 191
leftmost child 50
letter 148
level 115, 119, 122
level of abstraction 5

lexicographic ordering ($\preceq$) 154
Lindenbaum's Lemma 117
linear order 200
liveness property 35
LOCAL 59
local proposition 59
logic 94, 103
loosely ordered tree 72, 94, 128, 134
Löwenheim-Skolem Theorem 176

M

maximal sequence of neighbours 177
maximally consistent set 115
McKinsey axiom 31
mixed linearity axiom 101, 134
mixing axiom 100, 112
modality 16
model
    basic modal logic 19
    countable ordered tree 176
    general 60, 61
    linear temporal logic 33
    ordered tree 42
model checking 195
model pruning 182
modes of truth 15
modus ponens 25, 102
monadic second-order theory 148
multi-modal logic 22, 60

N

naming 50
natural numbers 50
necessary truth 15
necessitation rule 25, 102
negation
    strong ($\sim$) 56
negative introspection 27
NEIGHBOUR 161, 167, 174
next-operator 9, 40, 60
    dual of a 103
    weak 103
node 41
NON-DEGENERATE 54

O

OCCUR 55
ontological categories 55, 59
ontology 55
operators 47
order type $\mathbb{N}$ 41, 143
ordered tree 41, 62, 75
    discretely 52
    loosely 72
ordered tree model 42
OTL 39
overlapping intervals 8, 59, 192

P

p-morphic image 21, 63, 68
    of a tree 70
    of an ordered tree 73
p-morphism
    bidirectional 63
    composed 66
    for frames 63
    for models 63
    for relational structures 62
P-morphism Theorem 65, 67
parent 41
Parent Existence Lemma 121
path 81, 90
    empty 81
PDL 143, 194
positive introspection 27
possible worlds 18
prefix ordering ($\leq$) 153
product 46, 181
proof 97
property 55, 58
propositional axiom 98
propositional letter 39
propositional logic 14
pseudo neighbour relation 75

pseudo sibling relation 75

Q

quantification 147
  guarded 155
  over finite sets 154
quasi-equivalence class 41

R

Rabin's Theorem 152
reactive systems 8
reflexive closure 72
reflexive transitive closure 200
  operators 48
reflexivity 199
reflexivity axiom 29
related work 8
restrictions
  completeness 138, 142
right sibling 41
RIGHTMOST 50, 191
rightmost child 50
ROOT 50, 154
root 41, 50
rooted 69
rootedness axiom 102, 142
rule 24, 102, 104

S

safety property 35
satisfiability 22, 45
  in a general model 62
  in loosely ordered trees 93
second-order theory 147
semantics 40
  extended 185
sequence 74
seriality 200
seriality axiom 30, 54
set variable 149
SIBLING 161, 167, 174
sibling 41
simple variable 149
SINCE 32, 186
$\underline{\text{SINCE}}$ 187
SnS 148, 158
S2S 148, 152
SωS 148, 155, 157
solid proposition 59
soundness 24, 106
specification 8, 35
standard translation 22, 188
state 55
Stavi connectives 35, 188
strict implication ($\Rightarrow$) 58
strict linear order 200
strong completeness 106, 140
strong discreteness 200
strong negation ($\sim$) 56
strongly discrete branching 51
subrelation 68
substitution 102
subtree 41
  initial 161, 167, 174
successor functions 148
symbols 47
symmetry 199
symmetry axiom 30
syntax 39
  extended 185

T

temporal logic 2, 16, 45
term 148
  interpretation 150
theorem 24, 103
time period 45, 55
total order 201
transitive closure 143, 200
  non-reflexive 200
  reflexive 200
transitive closure rule 145
transitivity 199
transitivity axiom 30, 101, 128
translation

into first-order logic 22, 188
into second-order logic 162, 189
TREE 161, 168, 174
tree 40
binary 53
degenerate 54
discretely ordered 52
finite 53
loosely ordered 72
ordered 62
tree validity 45
truth
extended logic 186
global 45, 62
in a basic modal logic model 20
in a general model 61
in a temporal logic model 33
in an ordered tree model 43
second-order theories 150
Truth Lemma 114, 136
two-variable-fragment 24, 31
type of a node 176

U

undecidability 13, 47
uniform substitution 25, 102
union 199
universal/global modality ($\Diamond$,$\Box$) 48
unravelling 77, 80, 89
UNTIL 32, 186
<u>UNTIL</u> 187
until-style operators 186
basic temporal logic 32
global (<u>SINCE</u>, <u>UNTIL</u>) 187
horizontal (SINCE, UNTIL) 186
vertical (UP-TO, DOWN-TO) 186
unwinding cycles 78
UP-HERED 56
UP-TO 186
upward-hereditary 56

V

validity 22, 45, 62
in a frame 22, 62
in a tree 45
in countable ordered trees 175
in discretely ordered trees 168
in finite binary trees 163
second-order theories 152
valuation 19, 42, 136
variable
set 149
simple 149
veridicality 27
vertical accessibility relation 118
verum ($\top$) 18

W

weak completeness 106
weak density axiom 101, 132
weak discreteness 200
weak next-operator 103
WEAKLY-DISCRETE 51
weakly discrete branching 51
well-ordering 201
witness 43
word 148
empty (Λ) 148
world 19

Z

zoom 5