der Bundeswehr
Universität München

The Complexity of the Product Logics $\mathrm{K4} \times \mathrm{S5}$ and $\mathrm{S4} \times \mathrm{S5}$ and of the Logic SSL of Subset Spaces

Gisela Krommes

Vollständiger Abdruck der bei der Fakultät für Informatik der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Gutachter/Gutachterin:

1. Prof. Dr. Peter Hertling
2. Prof. Dr. Ulrike Sattler

Die Dissertation wurde am 11.09.2019 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Informatik am 03.01.2020 angenommen.
Die mündliche Prüfung fand am 08.01.2020 statt.

To my mother Ruth and
my cousin Reinhard

Acknowledgements

I would like to express my sincere gratitude to my advisor Prof. Peter Hertling for the continuous support of my Ph.D study, for his patience and motivation. His guidance helped me during all the time of my research and writing of this thesis.
I would also like to thank the whole team at our institute for the friendly and always encouraging atmosphere. It was a great pleasure to work and do research here.

Zusammenfassung

Wir zeigen, dass die Erfüllbarkeitsprobleme der bimodalen Produktlogiken $K4 \times S5$ und $S4 \times S5$ und der bimodalen Logik von Teilmengenräumen SSL jeweils EXPSPACE-vollständig sind. Tatsächlich geben wir einerseits für diese drei Probleme Entscheidungsalgorithmen an, die sogar in der Komplexitätsklasse ESPACE liegen. Der Kern des Beweises, dass diese Probleme andererseits EXPSPACE-hart sind, ist eine in logarithmischem Platz durchführbare Reduktion des Wortproblems für sogenannte Alternierende Turingmaschinen, die in Exponentialzeit arbeiten, auf das Erfüllbarkeitsproblem der Logik SSL. Bekanntlich erkennen derartige Maschinen gerade die Sprachen, die von gewöhnlichen Turingmaschinen in exponentiellem Platz erkannt werden. Die EXPSPACE-Härte von $S4 \times S5$ und $K4 \times S5$ zeigen wir durch ebenfalls in logarithmischem Platz durchführbare Übersetzungen, einerseits von SSL-Formeln in erfüllbarkeitsäquivalente $S4 \times S5$-Formeln, und schließlich von $S4 \times S5$-Formeln in erfüllbarkeitsäquivalente $K4 \times S5$-Formeln.

Abstract

We show that the satisfiability problems of the bimodal product logics $K4 \times S5$ and $S4 \times S5$ and of the bimodal logic of subset spaces SSL are EXPSPACE-complete. In fact, on the one hand we construct for these three problems decision algorithms working even in ESPACE. The heart of the proof, that on the other hand these problems are EXPSPACE-hard, is a reduction, computable in logarithmic space, of the word problem for so called Alternating Turing machines working in exponential time to the satisfiability problem for the logic SSL. It is known, that these machines accept exactly the languages accepted by usual Turing machines working in exponential space. We show EXPSPACE-hardness of $S4 \times S5$ and $K4 \times S5$ by translating on the one hand SSL-formulas to equisatisfiable $S4 \times S5$-formulas and finally by translating $S4 \times S5$-formulas to equisatisfiable $K4 \times S5$-formulas, where both translations are computable in logarithmic space as well.

Contents

Chapter 1

Introduction

One of the fundamental complexity-theoretic results about logic is Cook's theorem which says that the satisfiability problem for Boolean formulas is NP-complete [21]. Since then the complexity of many other logics has been analysed. In this thesis we are concerned with the bimodal product logics K4 × S5 and S4 × S5, and with the subset space logic SSL, a bimodal logic as well. To the best of our knowledge, the complexity of K4 × S5, of S4 × S5, and of SSL are open problems, and the goal of this thesis is to fill this gap. The main results of the thesis can be summarized in the following theorem:

Theorem 1.1. *The logics* K4 × S5*,* S4 × S5*, and* SSL *are* EXPSPACE-*complete.*

In [67, Question 5.3(i)] Marx posed the question what the complexity of the bimodal logic S4 × S5 is. This question is restated and extended to the logic K4 × S5 in [61, Problem 6.67, Page 334]. There it is also stated that "M. Marx conjectures that these logics are also EXPSPACE-complete". Our result shows that this conjecture is correct. Actually, we are considering the satisfiability problems of these three logics, and we are going to show that the satisfiability problems of these logics are EXPSPACE-complete. Of course, this assertion is equivalent to the theorem above because EXPSPACE is closed under complements.
For the complexity of the satisfiability problems of the logics K4 × S5 and S4 × S5 the best upper bound known was N2EXPTIME [61, Theorem 5.28], and the best lower bound known was NEXPTIME-hardness [61, Theorem 5.42]; compare also [61, Table 6.3, Page 340]. That it is desirable to know the complexity of SSL and similar logics is mentioned by Parikh, Moss, and Steinsvold in [75, page 30] and by Heinemann in [48, Page 153] and in [49, Page 513]. It is known that for any SSL-satisfiable formula there exists a model of at most doubly exponential size [22, Section 2.3]. This shows that

the complexity of the satisfiability problem of SSL is in N2EXPTIME as well. The best lower bound known for SSL is PSPACE-hardness [58, 59].
The thesis consists of nine chapters. The present, short chapter consists of an introduction, an overview of the thesis, and the definition of some fundamental notions from complexity theory. The second chapter is a short, general introduction to modal logic. In Chapter 3 we introduce the bimodal logics K4 $\times$ S5, S4 $\times$ S5, and SSL. First the syntax of bimodal formulas is defined, then various kinds of models are presented, and some additional notions are introduced that will be needed later on. In the usual Kripke semantics for modal logics the modal operators correspond to binary (accessibility)-relations on the domain of a model. Different logics can describe different properties of these accessibility relations. The logic K4 describes models with a transitive accessibility relation, while the accessibility relation in models of the logic S4 is reflexive and transitive. The accessibility relation in models of the logic S5 is an equivalence relation. The bimodal logics considered here combine in a certain way a logic with an at least transitive accessibility relation (K4, S4) with the logic S5. The logics K4 $\times$ S5 and S4 $\times$ S5 are products of their unimodal components. Such products are explained in detail in Chapter 3. The models of product logics have a grid structure, leading usually to high complexity or even undecidability. The logic SSL combines the components S4 and S5 not as a full product. But, as we will see, the missing parts of the product properties do not lead to lower complexity. SSL was originally designed to talk about subset spaces and to support elementary topological reasoning. But describing steps along smaller and smaller subsets turns out to be an interesting tool for describing knowledge acquisition as well. In Chapter 4 we prove upper bounds for the complexity of the satisfiability problems of the three bimodal logics K4 $\times$ S5, S4 $\times$ S5, and SSL. We show that all three problems are in ESPACE. In fact, we show that the satisfiability problem of K4 $\times$ S5 can be solved in space $O(n \cdot 2^{3n})$ and that the satisfiability problems of the other two logics, S4 $\times$ S5 and SSL, can be solved in space $O(n \cdot 2^{2n})$. We present recursive decision algorithms for these problems that are based on certain kinds of tableaux. This method was introduced by Beth [12]. Such a tableau is a special model where the domain of the model consists of formula sets satisfying certain consistency conditions. We will construct tableaux not as usual brick by brick. Instead we shall use prefabricated parts that we call “tableau-clouds” and that are somewhat similar to mosaics [72]. Tableau-clouds mirror the S5-component of the considered logics, while the rules for “gluing them together” mirror the K4- or S4-component. Our construction will yield tree-like structures where we allow backwards loops along branches to achieve termination of the search algorithm. Our recursive algorithms are similar to the recursive

algorithm of Ladner [63] for the modal logic S4. We would like to point out that Section 4.4 starts with some general combinatorial observations on certain binary relations that may be of interest elsewhere as well. The next five chapters are devoted to proving that all three problems are EXPSPACE-hard under logarithmic space reduction. Chapter 5 contains some preparations for this. For the two logics $S4 \times S5$ and SSL we introduce certain formulas that we call "shared variables" and that play the role of a certain kind of variables. Then, we show that binary counters can be simulated in both $S4 \times S5$ and in SSL with the aid of these shared variables. Finally, we give a short overview of Alternating Turing Machines which we use in order to establish the lower bounds of the satisfiability problem of these two logics. Note that Lange and Lutz [64] used Alternating Turing Machines in order to establish a sharp lower bound for the complexity of a certain dynamic logic. In Chapter 6 we show that any language recognized by an Alternating Turing Machine working in exponential time can be reduced in logarithmic space to the satisfiability problem of the logic SSL. As Alternating Turing Machines working in exponential time accept exactly the languages in EXPSPACE [19, Corollary 3.6] this shows that the satisfiability problem of SSL is EXPSPACE-hard. Together with the ESPACE-algorithm for this problem presented in Chapter 4 this shows that the satisfiability problem of SSL is EXPSPACE-complete. For the EXPSPACE-hardness of the satisfiability problem of $S4 \times S5$ two proofs are given. On the one hand, similarly to the reduction in Chapter 6, in Chapter 7 we show that any language recognized by an Alternating Turing Machine working in exponential time can be reduced in logarithmic space to the satisfiability problem of the logic $S4 \times S5$. This shows that the satisfiability problem of $S4 \times S5$ is EXPSPACE-hard. On the other hand, in Chapter 8 we show that the satisfiability problem of SSL can be reduced in logarithmic space to the satisfiability problem of $S4 \times S5$. As in Chapter 6 we have shown that the satisfiability problem of SSL is EXPSPACE-hard, this implies that the satisfiability problem of $S4 \times S5$ is EXPSPACE-hard as well. Again, in combination with the ESPACE-algorithm for this problem presented in Chapter 4 we can conclude that the satisfiability problem of $S4 \times S5$ is EXPSPACE-complete as well. In Chapter 9 we show that the satisfiability problem of $S4 \times S5$ can be reduced in logarithmic space to the satisfiability problem of $K4 \times S5$. This together with the EXPSPACE-hardness of the satisfiability problem of $S4 \times S5$ shows that the satisfiability problem of $K4 \times S5$ is EXPSPACE-hard as well. Once again, in combination with the ESPACE-algorithm for this problem presented in Chapter 4 we can conclude that the satisfiability problem of $K4 \times S5$ is EXPSPACE-complete as well. Finally, Chapter 10 contains some concluding remarks.

Let us end this introduction by mentioning some complexity-theoretic notions that will be used. The required notions from logic will be introduced in Chapter 3. First, as usual $\mathbb{N} = \{0, 1, 2, \ldots\}$ is the set of natural numbers, that is, of non-negative integers. An *alphabet* is a finite, nonempty set. For an alphabet Σ let Σ^* be the set of all finite strings over Σ. A *language* is any subset $L \subseteq \Sigma^*$, where Σ is any alphabet. For a function $s : \mathbb{N} \to \mathbb{N}$ we say that a language L *can be decided in space* $O(s)$ if there exists a deterministic Turing machine that decides L in space $O(s)$; for the precise definition of what this means the reader is referred to [74] or to any other textbook on complexity theory. The following two complexity classes have already been mentioned.

> EXPSPACE is the set of languages that can be decided by a deterministic Turing machine in space $2^{p(n)}$ for some polynomial p.

> ESPACE is the set of languages that can be decided by a deterministic Turing machine in space $2^{c \cdot n + c}$, for some constant $c \in \mathbb{N}$, that is, the exponent is linear.

Note that in order to speak about the complexity of a decision problem one should encode the instances of the decision problem by strings. In this way one gets a language. Finally, for reducing one language to another one we use the logarithmic space bounded reduction as in [74].

Chapter 2

A Short Introduction to Modal Logic

Before we describe some syntactic and semantic aspects of basic modal logic we start with the question: "What is modal logic?"

2.1 What is Modal Logic?

2.1.1 Introduction

Modal logic is a discipline of many facets.

- The class of modal logics was originally developed by philosophers to study different 'modes' of truth, and for a long time it was known as the logic of *necessity* and *possibility*.

- But the big breakthrough as an applied science happened when it turned out that modal logic could provide languages for talking about various relational structures, such as state transition systems for computer programs or semantic networks for knowledge representation. The invention of graph-based relational semantics (by Jaakko Hintikka, Stig Kanger, and Saul Kripke) in the late 1950s and early 1960s showed that standard modal logics could be regarded as fragments of first or second-order predicate logics.

- The main reason for the so successful spreading of modal propositional logics is their good balance between reasonable expressive power and good algorithmic behavior, especially their unusual robust *decidability* and in many cases low computational complexity.

- Many systems with various kinds of modal operators have been constructed in the course of time in order to provide effective formalisms for many different applications, e.g. talking about time, space, knowledge, beliefs, actions, obligations, etc.: *temporal, spatial, epistemic, dynamic, deontic logic* and so forth.
  A fact worth to be mentioned here is the close relationship between modal logics and *description logics*, a branch of knowledge representation and reasoning in AI. Nowadays it supplies many of the formalisms used to fix terminologies in medical and bio-informatics and has been proposed as the language for annotating web pages to develop a semantic web. It became apparent that many of the description logics were in fact modal logics in a different notational guise.

- Modern applications often require rather complex formal models and corresponding languages that are capable of reflecting different features of the application domain. For instance, to analyze the behavior of a multi-agent distributed system, we may need a formalism containing both, epistemic operators for capturing the knowledge of agents and temporal operators for taking care of the evolution of this knowledge in time. This motivates the development of *combined modal logics*, also called *many-dimensional modal logics.*

Some of the mentioned aspects in modal logic are explained in more detail in the following. Syntax and semantics are treated in the next sections, techniques for combining modal logics, important for the here investigated *logic of subset spaces SSL* and S4$\times$S5, are introduced in the next chapter and computational aspects will be discussed in the then following chapters.
We will consider only *propositional modal logics*, although there is a lot of research concerned with modal first-order logic, also called *quantified modal logic.*

2.1.2 Modal logics as formalization of modalities

Different 'modes' of truth were first discussed in a systematic way by Aristotle in *De Interpretatione.* He distinguished between "necessary" and "possible" truth of statements.
Consider for example the statements A: "Berlin is the capital of Germany" and B: "all humans are mortal". Both sentences are true but with different 'strength' of truth. Since we cannot imagine that some people live forever, we consider B as "*necessarily true*" or, in other words, as "true in all possible worlds". On the other hand, Germany could as well have a different capital, thus the first statement is true in our actual world but other worlds

making this sentence false are possible. For this reason, the statement A is only "*possibly true*", that means in at least one possible world.
The notion of "*possible worlds*" was coined by Gottfried Leibniz who suggested that there are other possible worlds besides the actual one. This opens the door to the intuitive correspondence with the standard quantifiers "there exists" and "for all".
In this way the intentional operators for "necessity" and "possibility" expand the descriptive scope of 'standard' logic and technically modal logics are obtained from standard logical systems by adding the *non-truth-functional operators* $\Box$ (read "necessarily") and $\Diamond$ (read "possibly") which form propositions from propositions. Then φ's being a necessary truth is expressed by $\Box\varphi$ and φ's being a possible truth is expressed by $\Diamond\varphi$.
If a statement is false in at least one possible world and true in at least one possible world, so that $(\Diamond\varphi \wedge \Diamond\neg\varphi)$ holds, it is called *contingent*.
But necessity and possibility are not the only assertions about the truth of a statement one may be interested in, and in the course of time a number of different modalities have been considered. Roughly speaking, a modality is any phrase that can be applied to a given statement φ to create a new statement that makes an assertion about the mode of φ's truth. Some examples of modalities are given below (see [51]).

$\Box\varphi$	$\Diamond\varphi$	
φ is *necessarily* true	φ is *possibly* true	modal logic
φ *ought to be*	φ is *permitted* to be	deontic logic
φ will *always* be true	φ will *sometimes* be true	temporal logic
agent A *knows* φ	for all agent A knows, φ *may be* true	epistemic logic
agent A *believes* φ	φ is *consistent with A's beliefs*	doxastic logic
after any execution of program P φ holds	*there is an execution* of P such that afterwards φ holds	dynamic logic

In particular, temporal, dynamic and epistemic logics found their way into computer science, AI, and economic game theory. Temporal logics are now used in industry for automated verification of hardware and software and modal logics of active agents with knowledge, beliefs and desires form a theoretical backbone of modern accounts to intelligent distributed computing. Pioneers of modal methods in computer science include Edmund Clarke, Joe Halpern, Zohar Manna, Robin Milner, Rohit Parikh, Amir Pnueli and Vaughan Pratt.
A modern approach to modal logic, that is deduction of theorems from a given set of axioms by means of explicitly stated rules of inference, started with

C. I. Lewis. In the attempt to avoid the ‘paradoxes’ of material implication (a false proposition implies any proposition and a true proposition is implied by any proposition) used in the *Principia Mathematica* by Whitehead and Russell, he defined in his pioneering book *Symbolic Logic*, written with C. H. Langford, five logical systems S1 – S5 to axiomatize ‘strict’ implication. In modern form, “φ strictly implies ψ” is expressed $\neg\Diamond(\varphi \wedge \neg\psi)$. Two of these systems, S4 and S5, are still in use today and well known by every modal logician.

2.1.3 Modal logics as fragments of standard logics

Modal operators are themselves a kind of quantifiers, but special “local” ones, referring only to objects “*accessible*” from the current one. Viewed in this way, modal logics are not extensions, but rather *fragments* of classical logic, with restricted forms of quantification.
For ordinary quantifiers $\forall$ and $\exists$, the domain of quantification depends not on an assignment to individual variables, or on the variables which are free in the quantified formula. Thus classical quantifiers have an ”external” view of the domain, i.e. they always can ”see” all objects which live there. On the contrary, the “local” modal quantification depends on the view which one can have from a certain point inside the model: it quantifies over the worlds which are seen from a given world.
Besides the important fact that relational structures can be unwound into trees satisfying exactly the same formulas, which is called the *tree-model property* of the modal logic, the restricted form of quantification is the main reason for the “robust decidability” of modal logics: the basic propositional modal logic is decidable and remains so even if one adds features like number restrictions, path quantification, least and greatest fixed points and so on, which are of crucial importance for applications (cf. [36, 91]).
After the disappointing discovery of the undecidability of first-order predicate logic FO, a vivid research was concerned with the search for decidable fragments of FO. For a detailed exposition see [16]. Many decidable fragments are based on restriction of the quantifier prefixes. There are two decidable fragments of FO that can be linked to modal logics in a natural way via the *standard translation* of modal formulas into classical logic:

the *two-variable fragment* FO^2 and the

guarded fragment GF, restricting quantification along an “*accessibility relation*” R between the objects of a domain.

Before we have a closer look at these fragments we start with the standard translation ST, developed by van Benthem [87, 88], that translates every

modal language into a corresponding fragment of a standard logical languages, mostly first-order, sometimes higher-order or infinitary, formalizing modal truth conditions.
For the basic unimodal logic the efficient standard translation ST is defined recursively using a first-order language with one binary predicate letter R and unary predicate letters $P_A, P_B, \cdots$ matching propositional variables $A, B, \cdots$.

$$\begin{array}{lll}
ST(A) & = & P_A x \\
ST(\neg\varphi) & = & \neg ST(\varphi) \\
ST(\varphi \wedge \psi) & = & ST(\varphi) \wedge ST(\psi) \\
ST(\varphi \vee \psi) & = & ST(\varphi) \vee ST(\psi) \\
ST(\Diamond\varphi) & = & \exists y(Rxy \wedge ST(\varphi)[y/x]) \quad \text{where } y \text{ is a new variable} \\
ST(\Box\varphi) & = & \forall y(Rxy \rightarrow ST(\varphi)[y/x]) \quad \text{where } y \text{ is a new variable}
\end{array}$$

Here $\varphi[y/x]$ is the result of substituting y for all free occurrences of x in φ.

Some examples:
$ST(\Diamond A) = \exists y(Rxy \wedge P_A y)$: there exists a successor world of x where A holds,
$ST(\Box A) = \forall y(Rxy \rightarrow P_A y)$: A holds at all successor worlds of x,
$ST(\Box\Diamond(A \vee B)) = \forall y(Rxy \rightarrow \exists z(Ryz \wedge (P_A z \vee P_B z)))$.

Compare the syntactical simplicity of the (variable-free) modal formulas with their translations.
Note that $ST(\varphi)$ always contains exactly one free variable. This free variable is what allows the internal perspective, typical of modal logic.
The translation ST provides embedding of modal logic into the following fragments of first-order logic:

- The *finite-variable fragment* restricting the use of variables to a certain number.
  Basic modal logic can be embedded into FO^2. By suitable re-use of variables, the above translation can make do with just two world variables. (Semantically, this means that evaluation never stores more than two worlds at a time.)
  Three variables become essential with dyadic modalities as in temporal logic with the until operator U. ($U\varphi\psi$ says that ψ will be true until a time when φ is true.). Since FO^k is undecidable for $k \geqslant 3$, the decidability of all natural temporal logics is not due to finite-variable restrictions.

- The *guarded fragment*. The guarded fragment consists of all formulas generated by the following syntax rule:

$$\begin{aligned} \varphi :: \quad & \text{atoms } P_A\overline{x} \text{ with } \overline{x} \text{ a tuple of variables} \\ & \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \exists\overline{y}(G(\overline{x},\overline{y}) \wedge \varphi(\overline{x},\overline{y})) \end{aligned}$$

where the "guard" $G(\overline{x},\overline{y})$ is an atom with variables from the sequences $\overline{x},\overline{y}$, occurring in any order and multiplicity.
Guarded logics generalize the characteristic feature of modal quantification, that one can only access nodes along basic edge relations, to a more general setting. In the guarded scenario, this means accessibility of all tuples that are covered (guarded) by some ground atom G.
The concept of guarded quantification was introduced by Andréka, van Benthem, and Németi [1].

The set of formulas achieved as translation of modal formulas is called the *modal fragment* of first-order logic. But there are also first-order formulas outside the modal fragment that are logically equivalent to formulas inside. This raises the question of which first-order formulas have an equivalent modal counterpart. This is answered by *van Benthem's characterization theorem*, using the notion of *bisimulation.* Bisimulation is a binary relation between state transition systems, associating systems with each other that behave in the same way in the sense that one system simulates the other and vice-versa. Van Benthem's Theorem says that *modal logic is the fragment of first-order logic that is closed under bisimulation.*
The standard translation allows to transfer immediately some meta-theoretic results for first-order logic to modal logic:

- Basic modal logic has the compactness property. That is, if every finite subset of a set of basic modal formulas is satisfiable, then the set itself is satisfiable.
- Basic modal logic has the Löwenheim-Skolem property. That is, if a set of basic modal formulas is satisfiable in at least one infinite model, then it is satisfiable in models of every infinite cardinality.
- The set of valid formulas in basic modal language is recursively enumerable.

2.2 Syntax and Semantics

2.2.1 Syntax of Modal Logics

Before we deal with bimodal formulas in the next chapter we consider unimodal languages, since the concepts for unimodal logics can in most cases

be extended to the multimodal case in a natural way.
Many modal logics are constructed from a weak basic logic called K (in honor of Saul Kripke). The formal language $\mathcal{L}_K$ of K can be based on the following *alphabet*:

> a countable infinite set AT of proposition letters whose elements are often denoted $A, B, C, \ldots$,
>
> the boolean connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$,
>
> the unary modal connectives $\Box$, $\Diamond$,
>
> punctuation symbols (,).

Then the language $\mathcal{L}_K$ of K, that is the class of all (well-formed) K-formulas (*wffs*), is recursively generated by the following Backus-Naur grammar:

$$\varphi \ ::= \ A \mid \neg\varphi \mid \Box\varphi \mid \Diamond\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\varphi \leftrightarrow \varphi)$$

for $A \in AT$. Brackets may be omitted where convenient, the convention for reading formulas being that $\neg$, $\Box$ and $\Diamond$ bind more strongly than $\wedge$ and $\vee$, the latter binding more strongly than $\rightarrow$ and $\leftrightarrow$ having the least binding strength.
The elements of AT are called *atomic formulas.*

Remark 2.1. First, note that by this definition every propositional formula can be viewed as a modal formula.
Second, there is redundancy in the way we have defined the basic modal language, since we don't need all these boolean connectives as primitives, and in most modal logics the necessity and possibility operators satisfy the following analogs of DeMorgan's laws from boolean algebra: $\Box\varphi$ is equivalent to $\neg\Diamond\neg\varphi$ and $\Diamond\varphi$ is equivalent to $\neg\Box\neg\varphi$, so $\Box$ and $\Diamond$ are what is known as *dual connectives*, just as $\exists$ and $\forall$ are in first-order logic.
Finally, note that, using De Morgan's laws and by suppressing double negations, every modal formula can be transformed into an equivalent formula in *negation normal form* (NNF), where only atomic formulas are allowed to be negated. But equivalent formulas in *conjunctive* or *disjunctive normal form* obeying the usual definition do not exist for every modal formula.

This is the basic modal language $\mathcal{L}_K$. But there are modal languages with several different modal operators and even operators that take more than one formula as argument.

2.2.2 Axiom Systems

Usually a modal logic Λ based on a language $\mathcal{L}$ is defined

> 'semantically' as the set of formulas in the language $\mathcal{L}$ valid in all models of some class of models, or
>
> 'syntactically' as the set of formulas in the language $\mathcal{L}$ derivable in a given derivation system,

whereas the syntactical way has the longer tradition. Usually, the semantical and syntactical definitions complement each other: the former explains the (intended) meaning of the logical constants and connectives, while the latter provides us with a reasoning machinery.

Definition 2.2 (Theorems and syntactically defined Logics)**.** 1. A derivation system Λ consists of a selected set of formulas, called *axioms*, and a finite set of *inference rules*. Derivation systems are also called *axiom systems* or *logical systems*.

2. A formula φ is called a *theorem* of Λ, written $\vdash_\Lambda \varphi$, if there exists in Λ a *proof*, also called a *derivation*, of φ, i.e. a finite sequence of formulas whose last member is φ, and such that each member of the sequence is either an axiom or derivable from earlier members by one of the rules of inference in Λ.

3. We call the set of theorems of Λ the *logic* Λ.

Where no ambiguity is likely to arise, we often omit the subscript 'Λ'.

Remark 2.3. Each axiom system has a corresponding logic, the converse however is not true. Not every (semantically defined) logic is axiomatisable – if we don't take the whole logic as an axiomatization of itself.

Usually, the axioms are defined via *axiom schemes*, that is a set of sentences all having the same form. For example, we take the schema $\Box\varphi \to \varphi$ to be $\{\Box\psi \to \psi \mid \psi \in \mathcal{L}\}$, and the *instances* of this schema are just the members of this set. Sometimes axioms are instead defined using propositional variables and a rule of substitution is given that permits to substitute any formula for them. For convenience we will in most cases not distinguish between axiom schemes and axioms. Axioms are often used in their *dual form*, for example, the dual form of $\Box\varphi \to \varphi$ is $\varphi \to \Diamond\varphi$ and $\Diamond\Diamond\varphi \to \Diamond\varphi$ is dual to $\Box\varphi \to \Box\Box\varphi$. If the set of axioms or axiom schemes can be chosen as a recursive (or finite) set, then we say that (the set of theorems of) this logic is *recursively* (*finitely*) *axiomatizable.* Reasonable rules of inference are *validity-preserving*, i.e. applied to valid formulas the theorems they yield are always valid too. We introduce some important rules of inference:

> *Modus ponens*: given φ and $\varphi \to \psi$, prove ψ.
>
> *Necessitation* (introduced by Gödel), also called *Generalization*: given φ, prove $\Box\varphi$. This rule says: if φ has a proof, then φ is necessary.
>
> *Uniform Substitution*: given φ, prove ψ, where ψ is obtained from φ by uniformly replacing proposition letters in φ by arbitrary wffs.
>
> *Regularity*: given $\varphi \to \psi$ prove $\Box\varphi \to \Box\psi$.

The first formalizations of modal logic were axiomatic. The start of this tradition is usually attributed to Lewis and Langford [65], but it seems that the historically correct attribution is MacColl [66]. The *axiom of modal distribution K* (see the table below) is the only axiom for the basic modal logic K. Numerous variations of K, based on additional axioms, with very different properties have been proposed. Table 2.1 shows a list of some of the best known axioms and combination of axioms together with their historical names, first-order translations and corresponding properties of the accessibility relation R in their relational models.
The axiom K is important because it lets us transform $\Box(\varphi \to \psi)$ (a boxed formula) into $\Box\varphi \to \Box\psi$ (an implication). This box-over-arrow distribution enables further purely propositional reasoning to take place. On the other hand, necessitation 'modalizes' provable formulas by stacking boxes in front. Roughly speaking, while the K axiom lets us apply classical reasoning inside modal contexts, necessitation creates new modal contexts for us to work with; modal proofs arise from the interplay of these two mechanisms.
In "An Interpretation of the Intuitionistic Propositional Calculus" [33] Gödel proposes the here used alternative axiomatization of the Lewis system S4. He also claims that a formula $\Box\varphi \vee \Box\psi$ is not provable in S4, unless either $\Box\varphi$ or $\Box\psi$ is provable. This claim has been proved algebraically by McKinsey and Tarski [70].
Gödel's short note is important for starting the fruitful practice of axiomatizing modal systems by separating the propositional calculus from the strictly modal part, but also for connecting intuitionistic and modal logic.

Remark 2.4. Note that the axioms defining the systems S4 and S5 allow *reduction of the sequence of modalities*:
Every S4-formula is equivalent to one of the form

$$\varphi,\ \Box\varphi,\ \Diamond\varphi,\ \Box\Diamond\varphi,\ \Diamond\Box\varphi,\ \Box\Diamond\Box\varphi,\ \Diamond\Box\Diamond\varphi,$$

where φ is not prefixed by a modal operator. Every S5-formula is equivalent to one of the form

$$\varphi,\ \Box\varphi,\ \Diamond\varphi,$$

where φ is not prefixed by a modal operator.

Table 2.1: Some well known axiom systems. Note that also different names for these combinations of axioms are in use.

K	$\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$	modal distribution
T	K + $(\Box\varphi \to \varphi)$ $\forall w(wRw)$	reflexive
B	K + $(\varphi \to \Box\Diamond\varphi)$ $\forall w, v\,(wRv \to vRw)$	symmetric
D	K + $(\Box\varphi \to \Diamond\varphi)$ $\forall w \exists v\,(wRv)$	serial
2	K + $(\Diamond\Box\varphi \to \Box\Diamond\varphi)$ $\forall w, w', v(wRv \wedge wRw' \to \exists v'(vRv' \wedge w'Rv'))$	confluent or weakly directed
4	K + $(\Box\varphi \to \Box\Box\varphi)$ $\forall w, v, u\,(wRv \wedge vRu \to wRu)$	transitive
5	K + $(\Diamond\varphi \to \Box\Diamond\varphi)$ $\forall w, v, u\,(wRv \wedge wRu \to vRu)$	euclidean
D1	K + $(\Box(\Box\varphi \to \psi) \vee \Box(\Box\psi \to \varphi))$ $(wRv \wedge wRu \to (vRu \vee uRv \vee v = u))$	linear or weakly connected
S4	T + 4	preorder
S4.2	S4 + 2	
S4.3	S4 + D1	
S5	S4 + 5	

2.2.3 Semantics

The *relational semantics* is nowadays the dominating kind of model theoretic semantics, interpreting formulas over graph-like structures. It was developed by Saul Kripke, Stag Kanger, Jaakko Hintikka and others in the late 1950s and early 1960s. Since Kripke's contributions [56,57] are the best known and regarded as landmarks in the development of modal semantics, relational

semantics is often called *Kripke semantics.* What made this approach so successful is the rich diversity of form supplied by relational structures and the excellent help in guiding logical intuitions.
The relational structures are called *frames* (or Kripke frames in honor to Saul Kripke), and consist of a pair $F = (W, R)$ such that W, the domain, is a non-empty set whose elements are usually called *points*, *states* or *worlds* and $R \subseteq (W \times W)$ is a relation, called *accessibility relation.*
A *model* M based on F is a triple (W, R, σ) with a *valuation* function

$$\sigma : AT \to \mathcal{P}(W)$$

(where AT is the set of propositional variables and $\mathcal{P}(W)$ is the power set of W) that assigns to each propositional variable $A \in AT$ a subset of W.
Informally we think of $\sigma(A)$ as the set of points in our model where A is true.
The above definition can be extended to the multimodal case in a natural way. In this case we have different accessibility relations $R_a, R_b, \ldots$, that means the edges of the graph are labeled $a, b, \ldots$, and we have corresponding labeled modalities $\langle a \rangle, [a], \langle b \rangle, [b]$, etc.

Remark 2.5. Note the *non-truth-functional* nature of the modal operators, in opposite to the Boolean connectives: the truth of $\Box\varphi$ and $\Diamond\varphi$ is not determined only by the truth value of φ.

With the work of this section at hand we can now ask about the *expressive power* of modal logic compared with classical logic at the level of frames.

2.2.4 Definability of classes of frames

We give ourselves a class of *frames* and ask whether it is possible to give a modal formula, or a collection thereof, which exactly characterizes this class. More precise:

Definition 2.6 (Definability)**.** Let $\mathcal{F}$ be a class of frames. We say that φ *defines* (or *characterizes*) $\mathcal{F}$ if

$$F \in \mathcal{F} \quad \Leftrightarrow \quad F \models \varphi.$$

Similarly, if Γ is a set of modal formulas, we say that Γ defines $\mathcal{F}$ if

$$F \in \mathcal{F} \Leftrightarrow F \models \Gamma.$$

In short, a modal formula φ defines a class of frames $\mathcal{F}$ if φ is valid in all $F \in \mathcal{F}$ and can be falsified in all $F' \notin \mathcal{F}$. In most cases, $\mathcal{F}$ is determined by a certain property of the accessibility relation, and it is therefore common to say that φ defines this property.
Note that truth in models is not appropriate to define properties of the accessibility relation, since special valuations may validate axioms even if the underlying frame behaves not as desired. For example, the axiom for transitivity $\Box A \rightarrow \Box\Box A$ is globally true in a model if the valuation value $\sigma(A) = W$, with W the domain of the model.
We have already seen some well known modal axioms together with the corresponding first-order formulas and the according properties of the accessibility relations, like *reflexivity*, *transitivity*, *symmetry*, etc. This raises two questions:

- If a class of frames (or more informally, a property) can be defined by a first-order formula, is there always a corresponding modal formula?
- And converse: If a class of frames can be defined by a modal formula, is there always a corresponding first-order formula?

These questions are what *correspondence theory* is concerned with, and in both cases the answer is no.
We list some examples of not modally definable properties:

irreflexivity:	$\forall x \neg Rxx$
frames having a reflexive point:	$\exists x Rxx$
asymmetric frames:	$\forall x \forall y (Rxy \rightarrow \neg Ryx)$
every state has a reflexive successor:	$\forall x \exists y (Rxy \wedge Ryy)$

Now the second question: although modal logic is weaker than first-order logic, there are indeed frame properties that are modally but not first-order definable. Here are two examples:

transitive frames which have no infinite ascending chain $x_0Rx_1Rx_2R\ldots$
Gödel-Löb formula: $\Box(\Box A \rightarrow A) \rightarrow \Box A$
(essential for provability logic)

Axiom of cyclic return: $(\Diamond A \wedge \Box(A \rightarrow \Box A)) \rightarrow A$

Now, what is the reason for the fact that modal logic can express such complex properties of frames? The answer is that validity in frames is essentially a *second-order* property, since we quantify over all states of the domain and all possible valuations, which are functions assigning a *subset* of the domain to each proposition letter. Thus we quantify across all subsets of the domain. In this sense, modal logic is a fragment of monadic second-order logic.

2.3 Some known Complexity Results

Ladner showed in [63] that satisfiability for K, T and S4 is PSPACE-complete under logspace reduction. He established the upper bounds with tableau-like procedures. Although these logics have satisfiable formulas requiring models of exponential size, the decision procedure can be computed using only polynomial space. This is due to the fact that satisfiable formulas of these logics are satisfiable in a tree-like model structure, with each branch of only polynomial length. Hence the structure can be constructed one branch at a time.
He also showed that for S5 satisfiability is NP-complete, hence validity is co-NP-complete. This rather low complexity is due to the fact that S5-satisfiable formulas have models of size linear in the number of modal connectives.
We briefly recall what is known about the relationship of the most important complexity classes:

$$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXPTIME} \subseteq \mathrm{NEXPTIME} \subseteq \mathrm{EXPSPACE}.$$

We know that (N)P $\neq$ (N)EXPTIME.

Chapter 3

Combined Modal Logics

First, we introduce general ways to combine unimodal logics. Then we describe in more detail different combinations of the logics K4 and S4 with S5, namely the logics K4 $\times$ S5 and S4 $\times$ S5 and the logic of subset spaces SSL. In later chapters we will investigate the complexity of the satisfiability problems of these three logics.

3.1 Combining modal logics

Combined logics have been attracting much interest. Modern applications are often concerned with composite domains. One example is the formalization of the development of an agent's knowledge in the course of time, leading in a natural way to the combination of epistemic and temporal logics.
The following is mainly based on the chapter "Combining Modal Logics" by A. Kurucz in the Handbook of Modal Logic [60] and the book "Many-dimensional Modal Logics: Theory and Applications" by D. Gabbay, A. Kurucz, F. Wolter and M. Zakharyaschev [61].
There are many ways to combine modal logics. The first methods were *products* of logics (introduced by K. Segerberg (1973 [82]) and independently by V. Sehtman (1978) [83]) and *fusion* (introduced by R. Thomason (1984) [86]).

The language of combined modal logics is independent from the method of combination:
For n unimodal languages $\mathcal{L}_1, \ldots, \mathcal{L}_n$ sharing the set of proposition letters and Boolean connectives and with distinct modal operators $\Box_1, \ldots, \Box_n$ the n-modal language

$$\mathcal{L}_1 \otimes \ldots \otimes \mathcal{L}_n$$

is built in the usual way using the given sets of proposition letters and Boolean connectives together with the set of modal operators $\{\Box_1, \ldots, \Box_n\}$. For bi-modal logics with modal operators $\Box$ and K we give a precise definition later on.

3.1.1 Fusion of Modal Logics

The method of fusion of normal modal logics, also called *independent join*, is the simplest and perhaps most frequently used way to combine logics.
For n axiomatisable logics $\Lambda_1, \ldots \Lambda_n$ models of their fusion

$$\Lambda_1 \otimes \ldots \otimes \Lambda_n$$

can be seen as graph-likes structures where the edges have n different labels and it can be axiomatised by the union of the axiom sets of the components. Especially, no axiom containing modal operators from more than one component is used. Thus, fusions are useful if the modal operators of the components are not supposed to interact. Many well-known multimodal logics like K_n, $\mathrm{S4}_n$, $\mathrm{S5}_n$ are fusions of their unimodal components.
The absence of interaction axioms often enables the transfer of good algorithmic properties from the components to their fusion and the reduction of reasoning in the fusion to reasoning in the components. But upper complexity bounds do *not* always transfer to their fusion. In [85] Spaan provides a criterion for the transfer of coNP-completeness.
The following complexity results are known:
Let $n > 1$ and $\Lambda_i \in \{\mathrm{K}, \mathrm{T}, \mathrm{S4}, \mathrm{S5}\}$, for all $1 \leqslant i \leqslant n$. Then $\Lambda_1 \otimes \ldots \otimes \Lambda_n$ is PSPACE-complete. A proof of Halpern and Moses in [38] can easily be modified to obtain this result.

3.1.2 Products of Modal Logics

In contrast to the situation with fusions, products of modal logics have real many-dimensional frames in the geometric sense, leading to interaction of the modal operators. Products of Kripke frames allow us to reflect interactions between modal operators representing time, space, knowledge, actions, etc.
The product construction for modal logics was introduced in [27, 82, 83] and is well established for applications in computer science and artificial intelligence.
Most of the material here is taken from [27,60–62]. We only consider products of two logics, but the concepts can easily be extended to products of more logics.

Definition 3.1 (Product Frames)**.** Given two Kripke frames

$$F_1 = (W_1, R_1), \quad F_2 = (W_2, R_2),$$

their product $F_1 \times F_2$ is the frame

$$(W_1 \times W_2, \overline{R}_1, \overline{R}_2)$$

where each $\overline{R}_i$ is the binary relation on $W_1 \times W_2$ defined by

$$(w_1, w_2)\,\overline{R}_1\,(w_1', w_n') \quad \text{iff} \quad w_1\,R_1\,w_1' \text{ and } w_2 = w_2',$$

$$(w_1, w_2)\,\overline{R}_2\,(w_1', w_n') \quad \text{iff} \quad w_2\,R_2\,w_2' \text{ and } w_1 = w_1'.$$

That is, the accessibility relations are defined coordinate-wise. Such a frame is called a *product frame.*

The *product* of two unimodal logics is *semantically* defined via product frames.

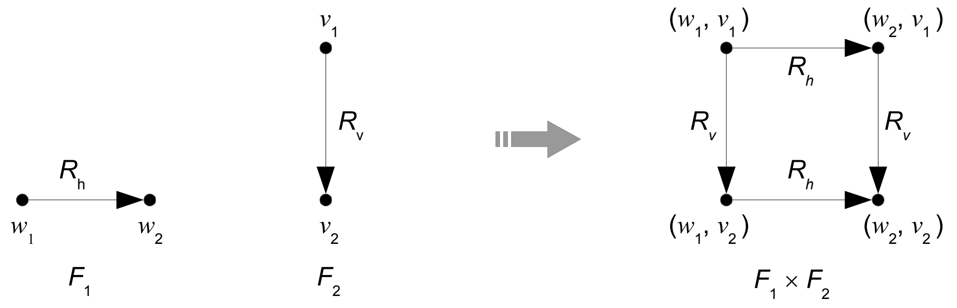
Definition 3.2 (Products of Modal Logics)**.** Let Λ_1, Λ_2 be two unimodal logics formulated in the languages $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively. Let

$$\mathcal{F} := \{F_1 \times F_2 \mid F_1 \models \Lambda_1 \text{ and } F_2 \models \Lambda_2\}.$$

Then the product Logic $\Lambda_1 \times \Lambda_2$ is defined by

$$\Lambda_1 \times \Lambda_2 := \{\varphi \in \mathcal{L}_1 \otimes \mathcal{L}_2 \mid \hat{F} \models \varphi \text{ for all } \hat{F} \in \mathcal{F}\}.$$

If the product is built from two unimodal logics, then the geometrical intuition suggests to take in the product frame one accessibility relation as the 'horizontal' relation R_h and the other as the 'vertical' relation R_v. The idea for the following picture illustrating such a product is taken from [60].



Axiomatizing product logics:

The definition of product frames implies that every binary product frame of the form (W, R_h, R_v) satisfies the following three properties:

- *left commutativity*:
  $\forall w \forall u \forall u' \, (wR_v u \wedge uR_h u') \to \exists w'(wR_h w' \wedge w'R_v u'))$,
- *right commutativity*:
  $\forall w \forall w' \forall u' \, (wR_h w' \wedge w'R_v u' \to \exists u(wR_v u \wedge uR_h u'))$,
- *Church-Rosser property*, also called *confluence*:
  $\forall w \forall w' \forall u \, ((wR_h w' \wedge wR_v u) \to \exists u'(w'R_v u' \wedge uR_h u'))$.

These properties are illustrated in the following picture shown in [61].

$$\begin{tikzcd} w \arrow[r, dashed, "R_h"] \arrow[d, "R_v"'] & w' \arrow[d, dashed, "R_v"] \\ u \arrow[r, "R_h"'] & u' \end{tikzcd}$$

left commutativity

$$\begin{tikzcd} w \arrow[r, "R_h"] \arrow[d, dashed, "R_v"'] & w' \arrow[d, "R_v"] \\ u \arrow[r, dashed, "R_h"'] & u' \end{tikzcd}$$

right commutativity

$$\begin{tikzcd} w \arrow[r, "R_h"] \arrow[d, "R_v"'] & w' \arrow[d, dashed, "R_v"] \\ u \arrow[r, dashed, "R_h"'] & u' \end{tikzcd}$$

Church-Rosser property

An important consequence is that the corresponding modal interaction formulas

$$com = \Diamond_1 \Diamond_2 \varphi \leftrightarrow \Diamond_2 \Diamond_1 \varphi$$

$$chr = \Diamond_1 \Box_2 \varphi \to \Box_2 \Diamond_1 \varphi$$

belong to every 2-dimensional product logic. The question arises whether we can axiomatize product logics with these formulas.

Definition 3.3 (Commutator)**.** Let Λ_1, Λ_2 be two unimodal logics axiomatized by axiom sets Ax_1 and Ax_2, defining frame-classes $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively (see Definition 2.6).

1. The *commutator*
   $$[\Lambda_1, \Lambda_2]$$
   is defined as the logic axiomatized by the set of axioms
   $$Ax_1 \cup Ax_2 \cup \{com, chr\}.$$

2. The pair Λ_1, Λ_2 is called *product-matching* if

$$[\Lambda_1, \Lambda_2] = \{\varphi \mid F_1 \times F_2 \models \varphi \text{ for all } F_1 \in \mathcal{F}_1, F_2 \in \mathcal{F}_2\}.$$

In this case we have

$$[\Lambda_1, \Lambda_2] = \Lambda_1 \times \Lambda_2.$$

It turns out that many pairs of standard modal logics are indeed product-matching. We are mainly interested in the following results which were among others obtained by Gabbay and Shehtman [27, Theorem 7.12]:

[K, K] = K×K

[S4, S4] = S4×S4

[S5, S5] = S5×S5

[K4, S5] = K4×S5

[S4, S5] = S4×S5

Transfer results, decidability and complexity:

The transfer of good properties from the components to their combination is less common with products than with fusions. We briefly list some of the known results.

- The logics K×K and S5×S5 have the finite product model property [60].
- The logics K4 × S5 and S4 × S5 both lack the finite product model property [61, Theorem 5.32], but are decidable. In fact, they are in coN2EXPTIME [61, Theorem 5.28]. We will come back to this later.
- [S4,S4] and S4×S4 are both undecidable [28].
- The satisfiability problem for S5×S5 is NEXPTIME-complete, and so the validity problem for S5×S5 is coNEXPTIME-complete [61].
- Let Λ be a bimodal logic between K×K and S5×S5. Then the satisfiability problem for Λ is NEXPTIME-hard, and so the validity problem for Λ is coNEXPTIME-hard [67].

In the next sections we investigate some properties of the combination of K4 and S4 with S5 and then turn to the introduction of the full product logics K4 × S5 and S4 × S5 and the logic of subset spaces SSL. Note that SSL only satisfies the left commutativity property, but there is an additional restriction on the allowed valuations.

3.2 Combining K4 and S4 with S5

3.2.1 Syntax of bimodal formulas

In this section we define the syntax of the bimodal formulas considered in combinations of K4 and S4 with S5 using the two modal operators $\Box$ (for the K4 or S4 part) and K (for the S5 part). We then introduce some additional notions and investigate some properties of combinations of the logics K4 and S4 with S5 that satisfy at least left commutativity. The logics K4 $\times$ S5, S4 $\times$ S5, and SSL, that are the subject of our complexity investigations, have the left commutativity property.
For the aimed complexity proofs it is convenient to define the syntax in such a way that propositional variables are represented as $x\,binary$ where *binary* is some binary number without leading zeros. The set of well-formed bimodal formulas $\mathcal{L}$ is generated by a context-free grammar.

Definition 3.4 (Syntax of Bimodal Formulas)**.** The set $\mathcal{L}$ of well-formed *bimodal formulas* is recursively generated using the following Backus-Naur grammar:

$$\begin{array}{lcl} \varphi & ::= & \neg\varphi \mid (\varphi \wedge \varphi) \mid K\varphi \mid \Box\varphi \mid \langle var\rangle \\ \langle var\rangle & ::= & x0 \mid x1 \mid x1\langle binstring\rangle \\ \langle binstring\rangle & ::= & 0 \mid 1 \mid 0\langle binstring\rangle \mid 1\langle binstring\rangle \end{array}$$

The set of propositional variables in $\mathcal{L}$ is defined by

$$AT := \{w \in \mathcal{L} \mid x \text{ is prefix of } w\}.$$

We also need some formulas of special type. For a modal operator $\circ \in \{K, \Box\}$ we define the set

$$\mathcal{L}_\circ := \{\psi \in \mathcal{L} \mid (\exists \chi \in \mathcal{L})\ \psi = \circ\chi\}$$

We adopt standard abbreviations for additional propositional connectives and the dual modal operators: $(\varphi \vee \psi) := \neg(\neg\varphi \wedge \neg\psi)$, $(\varphi \to \psi) := (\neg\varphi \vee \psi)$, $(\varphi \leftrightarrow \psi) := ((\neg\varphi \vee \psi) \wedge (\neg\psi \vee \varphi))$, $L\varphi := \neg K \neg\varphi$ and $\Diamond\varphi := \neg\Box\neg\varphi$. We will omit brackets whenever there is no danger that this might lead to confusion. We introduce some further syntactical concepts and notions.

Definition 3.5 (Subformula)**.** The set $\mathrm{sf}(\varphi)$ of *subformulas* of a bimodal formula φ is defined as usual by structural induction:

$$\begin{array}{rcl} \mathrm{sf}(A) & := & \{A\} \text{ for } A \in AT, \\ \mathrm{sf}(\neg\varphi) & := & \{\neg\varphi\} \cup \mathrm{sf}(\varphi), \\ \mathrm{sf}((\varphi \wedge \psi)) & := & \{(\varphi \wedge \psi)\} \cup \mathrm{sf}(\varphi) \cup \mathrm{sf}(\psi), \\ \mathrm{sf}(\Box\varphi) & := & \{\Box\varphi\} \cup \mathrm{sf}(\varphi), \\ \mathrm{sf}(K\varphi) & := & \{K\varphi\} \cup \mathrm{sf}(\varphi). \end{array}$$

3.2.2 Transitive Relations and Equivalence Relations

We now shed some light on the S5-equivalence classes and on an induced relation on them in models combining K4 or S4 with S5 in such a way that at least the left commutativity property holds. We start with some preliminaries. In the following let W be a nonempty set, and let $\equiv$ be an equivalence relation on W. As usual, for any $w \in W$, by

$$[w]_\equiv := \{v \in W \mid w \equiv v\}$$

we denote the $\equiv$-equivalence class of w, and, for any subset $A \subseteq W$, by

$$A_\equiv := \{[a]_\equiv \mid a \in A\}$$

we denote the set of $\equiv$-equivalence classes of elements of A.

Definition 3.6 (Induced Relation)**.** For any binary relation $R \subseteq W \times W$ we define the relation $R^\equiv \subseteq W_\equiv \times W_\equiv$ *induced on* $W_\equiv$ *by* R by

$$C\,R^\equiv D \;:\Leftrightarrow\; (\exists w \in C)(\exists v \in D)\; wRv,$$

for $C, D \in W_\equiv$.

Lemma 3.7. *If R is reflexive then $R^\equiv$ is reflexive as well.*

Proof. Consider some $C \in W_\equiv$. Then C is nonempty, that is, there is some $w \in C$. Then, as R is reflexive, we have wRw. This implies $C\,R^\equiv C$. Hence, $R^\equiv$ is reflexive. □

Lemma 3.8. *If R is transitive and the relations R and $\equiv$ have the left commutativity property then $R^\equiv$ is transitive as well.*

Proof. Consider $C, D, E \in W_\equiv$ with $C\,R^\equiv D$ and $D\,R^\equiv E$. We wish to show $C\,R^\equiv E$. There exist $w \in C$, $v, v' \in D$ and $u \in E$ with wRv and $v'Ru$. Due to the left commutativity property there exists some $w' \in C$ with $w'Rv'$. As R is transitive, we obtain $w'Ru$. Hence $C\,R^\equiv E$. □

In the following sections we will introduce *frames* for the logics K4 × S5, S4 × S5, and SSL. They will consist of a set W and two relations, a transitive relation or preorder $\xrightarrow{\Diamond}$ for the modal operator $\Box$ and an equivalence relation $\xrightarrow{L}$ for the modal operator K. In the case of the logics K4 × S5 the relation $\xrightarrow{\Diamond}$ is transitive, and in the case of the logics S4 × S5 and SSL the relation $\xrightarrow{\Diamond}$ is a preorder.

Corollary 3.9. *Let $M = (W, \xrightarrow{\Diamond}, \xrightarrow{L})$ be a triple consisting of a set W, a transitive relation $\xrightarrow{\Diamond}$ on W and an equivalence relation $\xrightarrow{L}$ on W such that $\xrightarrow{\Diamond}$ and $\xrightarrow{L}$ have the left commutativity property.*

1. *Then $\xrightarrow{\Diamond}^{\xrightarrow{L}}$ is a transitive relation on $W_{\xrightarrow{L}}$.*

2. *If the relation $\xrightarrow{\Diamond}$ is even a preorder then $\xrightarrow{\Diamond}^{\xrightarrow{L}}$ is a preorder as well.*

Proof. This follows from Lemmas 3.7 and 3.8. □

Often, in a model as described above, we will call the $\xrightarrow{L}$-equivalence class of a point w, denoted $[w]_{\xrightarrow{L}}$ or shorter $[w]_L$, the *cloud* of w.
It gives a good intuition to think of commutativity as follows. Let C, D be two clouds in a model M such that $C \xrightarrow{\Diamond}^{\xrightarrow{L}} D$. Then

1. M has the left commutativity property iff for all $v \in D$ there is some $w \in C$ with $w \xrightarrow{\Diamond} v$ (all points in D have a father in C).

2. M has the right commutativity property iff for all $w \in C$ there is some $v \in D$ with $w \xrightarrow{\Diamond} v$ (all points in C have a son in D).

3.3 The logics K4 × S5 and S4 × S5

For the syntax definition of bimodal formulas we refer the reader to Definition 3.4.
K4 × S5 is defined as the logic of K4 × S5-product frames, and S4 × S5 is defined as the logic of S4 × S5-product frames, defined as follows.

Definition 3.10 (The Logics K4 × S5 and S4 × S5). 1. A K4-*frame* is a pair $(W, R_\Diamond)$ such that W is a non-empty set and $R_\Diamond \subseteq W \times W$ is a transitive relation on W.
An S4-*frame* is a pair $(W, R_\Diamond)$ such that W is a non-empty set and $R_\Diamond \subseteq W \times W$ is a preorder on W, that is a reflexive and transitive relation.
An S5-*frame* is a pair (W, R_L) such that W is a non-empty set and $R_L \subseteq W \times W$ is an equivalence relation on W, that is a reflexive, transitive and symmetric relation.

2. Let $X \in \{\mathrm{K4}, \mathrm{S4}\}$. Let $F_1 := (W_1, R_\Diamond)$ be some X-frame, $F_2 := (W_2, R_L)$ be some S5-frame. Then the *product* $F_1 \times F_2$ is the triple

$$F := (W_1 \times W_2, \xrightarrow{\Diamond}, \xrightarrow{L})$$

where $\xrightarrow{\Diamond}$ and $\xrightarrow{L}$ are the binary relations on $W_1 \times W_2$ defined by

$$\begin{aligned}(v_1, v_2) \xrightarrow{\Diamond} (w_1, w_2) \quad &\Longleftrightarrow \quad v_1 R_\Diamond w_1 \text{ and } v_2 = w_2,\\ (v_1, v_2) \xrightarrow{L} (w_1, w_2) \quad &\Longleftrightarrow \quad v_2 R_L w_2 \text{ and } v_1 = w_1,\end{aligned}$$

for all $(v_1, v_2), (w_1, w_2) \in W_1 \times W_2$. Any such product is called an $X \times$ S5-*product frame.*

3. Let $X \in \{\mathrm{K4}, \mathrm{S4}\}$. Then an $X \times$ S5-*product model* is a quadruple $(W, \xrightarrow{\Diamond}, \xrightarrow{L}, \sigma)$ such that the triple $(W, \xrightarrow{\Diamond}, \xrightarrow{L})$ is an $X \times$S5-product frame and

$$\sigma : AT \to \mathcal{P}(W)$$

is a function mapping proposition letters to subsets of W.

4. Let $X \in \{\mathrm{K4}, \mathrm{S4}\}$. The logic $X \times$ S5 is defined as the set of formulas valid in all $X \times$ S5-product frames.

Let $X \in \{\mathrm{K4}, \mathrm{S4}\}$. Note that the relation $\xrightarrow{\Diamond}$ in an $X \times$S5-product frame is automatically transitive and in the case of $X =$ S4 even a preorder and that the relation $\xrightarrow{L}$ in such product frames is automatically an equivalence relation. Furthermore, recall that product frames satisfy left and right commutativity as well as the Church-Rosser property. Actually, in $X \times$ S5-product frames the Church-Rosser property is implied by the commutativity rules. To see this, let w, w', v be points such that $w \xrightarrow{L} w' \wedge w \xrightarrow{\Diamond} v$. By the symmetry of $\xrightarrow{L}$ we get $w' \xrightarrow{L} w$ and with $w \xrightarrow{\Diamond} v$ and right commutativity it follows that there is some point v' such that $w' \xrightarrow{\Diamond} v'$ and $v' \xrightarrow{L} v$. Again, by the symmetry of $\xrightarrow{L}$ we get $v \xrightarrow{L} v'$. That is, there exists some point v' such that $w' \xrightarrow{\Diamond} v' \wedge v \xrightarrow{L} v'$ as demanded by the Church-Rosser property.
But since $X \times \mathrm{S5} = [X, \mathrm{S5}]$ we can equivalently characterize $X \times$S5 as the set of formulas derivable from the X-axioms for the $\Box$-operator, the S5-axioms for the K-operator and the axioms

$$com_{right} = L\Diamond\,\varphi \to \Diamond L\,\varphi$$

$$com_{left} = \Diamond L\,\varphi \to L\Diamond\,\varphi$$

for right and left commutativity. We call Kripke-frames satisfying these axioms $X \times$ S5-commutator frames.

Definition 3.11. (K4 $\times$ S5- and S4 $\times$ S5-Commutator Frames and Models).

1. A K4 $\times$ S5-*commutator frame* is a a triple $(W, \xrightarrow{\Diamond}, \xrightarrow{L})$ such that $\xrightarrow{\Diamond}$ is a transitive relation on W, such that $\xrightarrow{L}$ is an equivalence relation on W, and such that right commutativity and left commutativity hold.
2. An S4 $\times$ S5-*commutator frame* is a a triple $(W, \xrightarrow{\Diamond}, \xrightarrow{L})$ such that $\xrightarrow{\Diamond}$ is a preorder on W, such that $\xrightarrow{L}$ is an equivalence relation on W, and such that right commutativity and left commutativity hold.
3. Let $X \in \{\mathrm{K4}, \mathrm{S4}\}$. An $X \times$ S5-*commutator model* or short $X \times$ S5-*model* is a quadruple $(W, \xrightarrow{\Diamond}, \xrightarrow{L}, \sigma)$ such that the triple $(W, \xrightarrow{\Diamond}, \xrightarrow{L})$ is an $X \times$ S5-commutator frame and
$$\sigma : AT \to \mathcal{P}(W)$$
is a function mapping proposition letters to subsets of W.

For $X \in \{\mathrm{K4}, \mathrm{S4}\}$, we define semantics of $X \times$S5, based on $X \times$S5-commutator frames, via the satisfaction relation $\models$.

Definition 3.12 (Semantics)**.** Let $X \in \{\mathrm{K4}, \mathrm{S4}\}$. Let $M = (W, \xrightarrow{\Diamond}, \xrightarrow{L} \sigma)$ be some $X\times$ S5-commutator model. The satisfaction relation $\models \subseteq W \times \mathcal{L}$ is defined as follows. Let $w \in W$, let A be an arbitrary propositional variable, and let φ be a bimodal formula. Then

$$\begin{array}{lll}
M, w \models A & :\Longleftrightarrow & w \in \sigma(A), \\
M, w \models \neg\varphi & :\Longleftrightarrow & \text{not } M, w \models \varphi, \\
M, w \models (\varphi \wedge \psi) & :\Longleftrightarrow & M, w \models \varphi \text{ and } M, w \models \psi, \\
M, w \models \Box\varphi & :\Longleftrightarrow & \text{for all } v \in M \text{ with } w \xrightarrow{\Diamond} v \text{ we have } M, v \models \varphi, \\
M, w \models K\varphi & :\Longleftrightarrow & \text{for all } v \in M \text{ with } w \xrightarrow{L} v \text{ we have } M, v \models \varphi.
\end{array}$$

When the model M is clear, then we often write $w \models \varphi$ instead of $M, w \models \varphi$. The following lemma prepares the definition of satisfiable formulas.

Lemma 3.13. *Let $X \in \{\mathrm{K4}, \mathrm{S4}\}$. For a formula φ in the language $\mathcal{L}$ the following two conditions are equivalent.*

1. *There exist an $X \times$ S5-product model M and some point w in M such that $M, w \models \varphi$.*
2. *There exist an $X \times$ S5-commutator model M and some w in M such that $M, w \models \varphi$.*

Proof. The direction "1 ⇒ 2" is clear. For the direction "2 ⇒ 1" see Theorem 7.12 in [27]. □

Definition 3.14 (Satisfiable K4 × S5- and S4 × S5-Formulas)**.** Let $X \in \{\mathrm{K4}, \mathrm{S4}\}$. Let φ be a formula in the language $\mathcal{L}$. The formula φ is $X \times \mathrm{S5}$-*satisfiable* iff one and then both of the two equivalent conditions in Lemma 3.13 are satisfied.
In this case a pair (M, w) satisfying the first condition is called an $X \times \mathrm{S5}$-*product model of* φ, while a pair (M, w) satisfying the second condition in the previous lemma is called an $X \times \mathrm{S5}$-*commutator model of* φ. If the type of model is clear from the context we simply call (M, w) a *model* of φ.

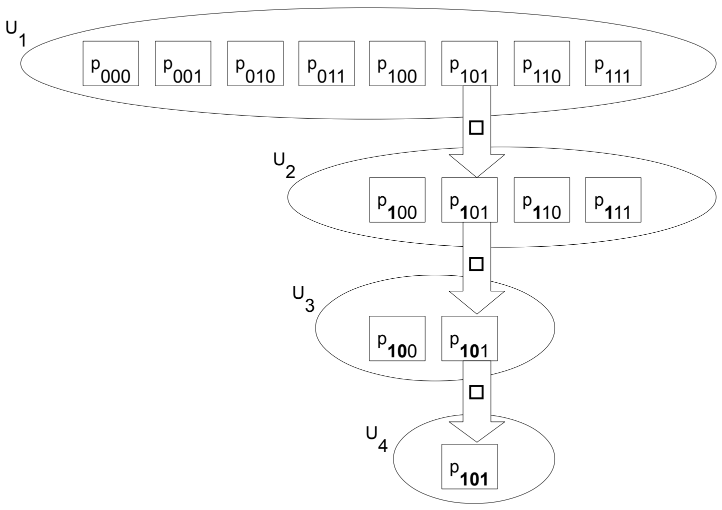
Let $X \in \{\mathrm{K4}, \mathrm{S4}\}$. Actually, it is known that whenever a formula φ is $X \times \mathrm{S5}$-satisfiable then there exists even an $X \times \mathrm{S5}$-commutator model of size doubly exponential in the length of φ [61, Theorem 5.27]. This is not true for product models: there exists an $X \times \mathrm{S5}$-satisfiable formula φ such that any $X \times \mathrm{S5}$-product model (M, w) of φ is infinite [61, Theorem 5.32].

3.4 The logic of subset spaces SSL

3.4.1 Introduction

The *subset space logic* SSL was introduced by Moss and Parikh in the early 90s [71] and in more detail presented in [22]. It is a bimodal logic with the modal operators $\Box$ and K, basically designed for elementary reasoning about points and sets. It combines the systems S4 (for $\Box$) and S5 (for K), thus embedding topological aspects into epistemic logic. Tarski and McKinsey proved that the topologically valid sentences are exactly those provable in the logical system S4 [68, 69], and the system S5 is usually taken to reason about an agent's knowledge.
As the name already indicates, the most natural interpretation of SSL is not in usual Kripke structures but in subset spaces $(X, \mathcal{O}, \sigma)$, consisting of a nonempty set X, a family $\mathcal{O}$ of subsets of X and a valuation function σ. The S5-operator K then quantifies 'horizontally' over points in a set $U \in \mathcal{O}$, and the S4-operator $\Box$ quantifies 'downwards' over subsets of U contained in $\mathcal{O}$. In the context of reasoning about knowledge, X can be seen as a set of possible worlds and $\mathcal{O}$ as the set of all *observations* about these worlds available to an agent.
In SSL the knowledge is defined with respect to both a point $x \in X$ and a *neighborhood* U of x. Such a pair x, U with $x \in U \in \mathcal{O}$ is called a *neighborhood situation*. Then in the neighborhood situation x, U the agent knows φ if φ

holds at all points in U. On the other hand, if in the actual situation the agent does not know whether φ holds or not he considers both as possibilities. But the more the agent knows, the smaller the number of alternatives he takes into account, thus *acquisition of knowledge* corresponds to shrinking the set of possible alternatives or, in the notion of topology, to shrinking the neighborhood U of x. Since gaining knowledge usually requires some effort, the $\Box$-operator is also called the *effort operator.*
We want to demonstrate this with an example that is small enough to be visualized. Imagine we have three dice boxes numbered 1-3, each containing one coin marked '0' on one side and '1' on the other. Playing with these dice boxes, there are 8 possible outcomes: 000, 001, ... 111. Let '101' be the actual result. Before removing one of these dice boxes, the observer or agent knows nothing about the result and thus considers all alternatives as possible. After spending some effort to remove the first box, the set of alternatives shrinks to the set containing the four outcomes beginning with '1', and so on. The picture shows the development of the agent's knowledge if the dice boxes are removed one by one, in the order of their number. To



formalize this example, we choose the propositional variables C_1, C_2, C_3 to represent the three coins and a subset space $(X, \mathcal{O}, \sigma)$, where

$X := \{p_{000}, p_{001}, p_{010}, p_{011}, p_{100}, p_{101}, p_{110}, p_{111}\}$
is the set of possible outcomes,

$\mathcal{O} := \{U_1, U_2, U_3, U_4\} \subseteq X$, with $U_1 := X$, $U_2 := \{p_{100}, p_{101}, p_{110}, p_{111}\}$, $U_3 := \{p_{100}, p_{101}\}$ and $U_4 := \{p_{101}\}$,
is a set of equivalence classes for p_{101}, and finally

σ is a valuation function such that $p_j \in \sigma(C_i)$ iff the binary string j contains '1' at the i-th position.

As we have seen, reasoning about knowledge and knowledge acquisition leads in a natural way to the subject of topology where the notion of *knowledge* corresponds to *closeness* in topology and *knowledge acquisition* corresponds to *approximation.*

Remark 3.15. Although $\mathcal{O}$ is not necessarily closed under union and finite intersection, it gives a good intuition to think about $(X, \mathcal{O})$ as some kind of topological space. For this reason, adopting topological terminology, the elements of $\mathcal{O}$ are called *opens.*

3.4.2 Subset Spaces

For the syntax definition of bimodal formulas we refer the reader again to Definition 3.4.
The intended semantical domains of SSL are *subset spaces* and defined as follows.

Definition 3.16 (Subset Space)**.** A *subset frame* is a pair

$$\mathcal{X} = (X, \mathcal{O})$$

where X is a non-empty set and $\mathcal{O}$ is a collection of subsets of X (not necessarily a topology) satisfying $X \in \mathcal{O}$. The elements of $\mathcal{O}$ are called *opens.*
A *subset space model* is a triple

$$S = (X, \mathcal{O}, \sigma)$$

where $(X, \mathcal{O})$ is a subset frame, and $\sigma : AT \to \mathcal{P}(X)$ is a function mapping propositional letters to subsets of the domain X.

Although formulas will be evaluated at *neighborhood situations* U, w, the valuation σ is defined only with respect to points and this fact gives reason to the persistence axiom introduced later on.
Semantics for SSL is defined via the satisfaction relation $\models$ between neighborhood situations in a subset space model and formulas in $\mathcal{L}$.

Definition 3.17 (Semantics)**.** Let $S = (X, \mathcal{O}, \sigma)$ be a subset space model. The satisfaction relation $\models$ is defined by recursion on the form of the bimodal formula considered. Consider $p \in U \in \mathcal{O}$, $\varphi \in \mathcal{L}$, and $A \in AT$.

$$\begin{array}{lll}
p, U \models A & :\Longleftrightarrow & p \in \sigma(A), \\
p, U \models \neg\varphi & :\Longleftrightarrow & \text{not } p, U \models \varphi, \\
p, U \models \varphi \wedge \psi & :\Longleftrightarrow & p, U \models \varphi \;\text{ and }\; p, U \models \psi, \\
p, U \models K\varphi & :\Longleftrightarrow & q, U \models \varphi \;\text{ for all }\; q \in U, \\
p, U \models \Box\varphi & :\Longleftrightarrow & p, V \models \varphi \;\text{ for all }\; V \in \mathcal{O} \;\text{ such that }\; p \in V \subseteq U
\end{array}$$

This is the semantical way to define SSL, but there is also a syntactical way based on a set of axiom schemes.

3.4.3 Axiomatizing

Together with the Hilbert-style inference rules

$$\frac{\varphi \to \psi,\ \varphi}{\psi}\text{ modus ponens} \qquad \frac{\varphi}{K\varphi}\,K\text{-necessitation} \qquad \frac{\varphi}{\Box\varphi}\,\Box\text{-necessitation}$$

the following axiom schemes establish the set of SSL-provable formulas:

(1) all substitution instances of tautologies of classical propositional logic

the S5-axioms for K:

(2) $K(\varphi \to \psi) \to (K\varphi \to K\psi)$

(3) $K\varphi \to \varphi$

(4) $K\varphi \to KK\varphi$

(5) $L\varphi \to KL\varphi$

the S4-axioms for $\Box$:

(6) $\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$

(7) $\Box\varphi \to \varphi$

(8) $\Box\varphi \to \Box\Box\varphi$

the *persistence axioms*:

(9) $(A \to \Box A) \wedge (\neg A \to \Box\neg A)$ for all propositional variables A

the *cross axioms*:

(10) $K\Box\varphi \to \Box K\varphi$

The cross axiom is the typical one for the logic of subset spaces and a structure satisfying this axiom has the left commutativity property, for SSL traditionally called *cross property.*
The axioms are often used in their dual form, e.g.

$$K\Box\varphi \to \Box K\varphi \;\equiv\; \Diamond L\varphi \to L\Diamond\varphi.$$

Note that because of the persistence axiom, SSL is not closed under substitution.

Theorem 3.18. *Let $\vdash_{\mathrm{SSL}}$ denote derivability in the above described deductive system and let $\mathfrak{S}$ denote the class of subset models. Then*

$$\vdash_{\mathrm{SSL}} \varphi \;\Leftrightarrow\; \mathcal{S} \models \varphi \textit{ for all } \mathcal{S} \in \mathfrak{S}.$$

The proof of this theorem can be found in [22].
In the next subsection we introduce a second type of models for SSL that turns out to be very useful.

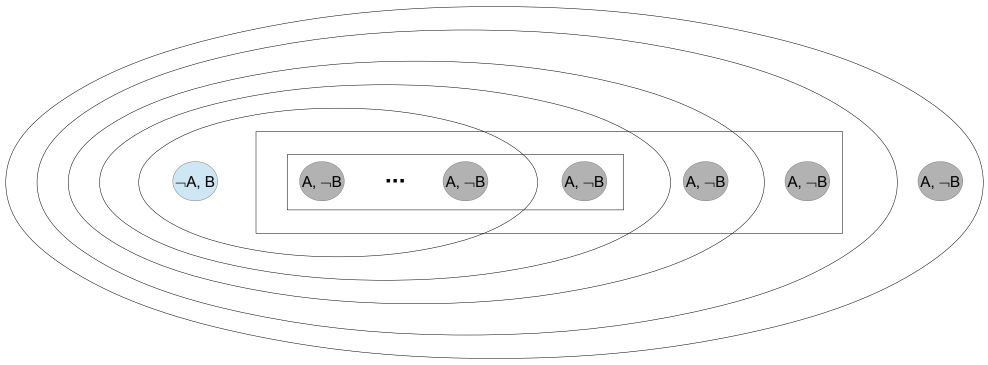
3.4.4 Cross Axiom Models

Unfortunately, SSL lacks the finite model property w.r.t. the class of subset spaces and hence establishing decidability results based on this class of models is a hard or even unsolvable task.
The following example, which is a simplification of example B in [22], gives a kind of infinity axiom for the class of subset spaces.

Example 3.19. Let A and B be two different propositional variables. For $\alpha := K(\neg A \vee \Diamond K\neg B)$ consider the formula

$$\varphi := \Box\Diamond\alpha \wedge \Box\Diamond\neg\alpha.$$

We claim that φ has no finite subset space model. For suppose that S were a finite subset space model containing p and U such that $p, U \models \varphi$. We may assume that U is a $\subseteq$-minimal open about p with this property. But the minimality implies that $p, U \models \alpha \wedge \neg\alpha$, and this is absurd. Nevertheless this formula is satisfiable in an infinite model as shown below.



This example motivates that we move to a bigger class of models, and this class will turn out to be behaved better.

Definition 3.20 (Cross Axiom Model)**.** A *cross axiom frame* is a tuple

$$M := (W, \xrightarrow{\Diamond}, \xrightarrow{L})$$

such that W is a non-empty set, $\xrightarrow{\Diamond}$ is a preorder on W, $\xrightarrow{L}$ is an equivalence relation on W, and the cross property holds: If

$$w \xrightarrow{\Diamond} v \xrightarrow{L} v',$$

then there is some $w' \in W$ such that

$$w \xrightarrow{L} w' \xrightarrow{\Diamond} v'.$$

A *cross axiom model* or short SSL-*model* is a cross axiom frame together with a function

$$\sigma : AT \to \mathcal{P}(W)$$

mapping proposition letters to subsets of W and satisfying the following condition for all $v, w \in W$ and for all propositional variables A:

$$w \xrightarrow{\Diamond} v \ \to (w \in \sigma(A) \ \leftrightarrow \ v \in \sigma(A))\,.$$

Definition 3.21 (Semantics)**.** Let $M = (W, \xrightarrow{\Diamond}, \xrightarrow{L} \sigma)$ be some cross axiom model. The satisfaction relation $\models \subseteq W \times \mathcal{L}$ is defined as follows. Consider some $w \in W$, $\varphi \in \mathcal{L}$, and $A \in AT$.

$$\begin{array}{lll}
M, w \models A & :\Longleftrightarrow & w \in \sigma(A), \\
M, w \models \neg\varphi & :\Longleftrightarrow & \text{not } M, w \models \varphi, \\
M, w \models (\varphi \wedge \psi) & :\Longleftrightarrow & M, w \models \varphi \text{ and } M, w \models \psi, \\
M, w \models \Box\varphi & :\Longleftrightarrow & \text{for all } v \in M \text{ with } w \xrightarrow{\Diamond} v \text{ we have } M, v \models \varphi, \\
M, w \models K\varphi & :\Longleftrightarrow & \text{for all } v \in M \text{ with } w \xrightarrow{L} v \text{ we have } M, v \models \varphi.
\end{array}$$

When the model M is clear, then we often write $w \models \varphi$ instead of $M, w \models \varphi$. The following summarizes argumentations in [22].

Definition 3.22 (Cross Axiom Models based on Subset Space Models)**.** Let $S = (X, \mathcal{O}, \sigma_S)$ be a subset space model. Then we define a quadruple $M := (W, \xrightarrow{\Diamond}, \xrightarrow{L}, \sigma_M)$ based on S as follows.

$W := \{(p, U) \mid p \in U \in \mathcal{O}\}$,
that is we take the neighborhood situations of S as the points in W,

$\xrightarrow{\Diamond} := \{((p, U), (q, V)) \in W^2 \mid p = q \text{ and } V \subseteq U\}$,

$\xrightarrow{L} := \{((p, U), (q, V)) \in W^2 \mid U = V\}$,

$\sigma_M(A) := \{(p, U) \in W \mid p \in \sigma_S(A)\}$, for all propositional variables A.

Lemma 3.23. *Let $S = (X, \mathcal{O}, \sigma_S)$ be a subset space model and let $M := (W, \xrightarrow{\Diamond}, \xrightarrow{L}, \sigma_M)$ be the quadruple based on S as defined in Definition 3.22.*

1. *M is a cross axiom model.*
2. *For all $\varphi \in \mathcal{L}$ and for all neighborhood situation $p, U \in S$*

$$S, p, U \models \varphi \quad \Leftrightarrow \quad M, (p, U) \models \varphi.$$

Proof. 1. Obviously, $\xrightarrow{\Diamond}$ is a preorder, $\xrightarrow{L}$ is an equivalence relation, persistence of propositional variables is satisfied and left commutativity holds, that is, M is indeed a cross axiom model.

2. This is an easy induction on the structure of φ. □

SSL is also sound and complete w.r.t. the class of cross axiom models (not cross axiom *frames* because of the restrictions on the interpretation of propositional letters).

Theorem 3.24. *Let φ be a formula in $\mathcal{L}$. Then φ is* SSL*-derivable iff φ is valid in every cross axiom model.*

Proof. Soundness can be proved by induction on the structure of proofs. Completeness follows from the completeness of SSL for the class of subset space models together with Lemma 3.23. □

Thus we get the following equivalent definitions of satisfiable SSL-formulas.

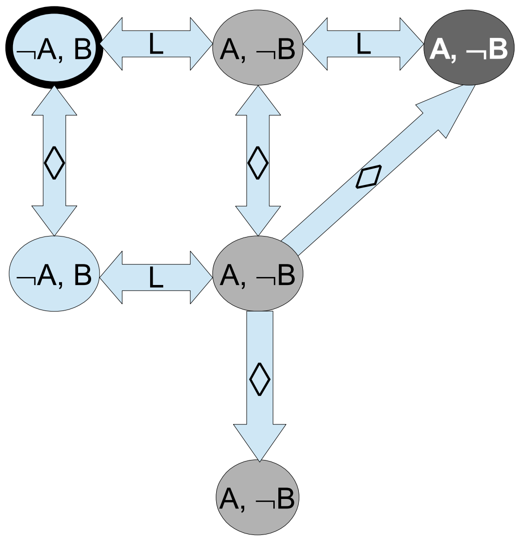
Definition 3.25 (Satisfiable SSL-Formulas)**.** Let φ be a formula in the language $\mathcal{L}$. The formula φ is SSL-*satisfiable* iff one and then both of the following two equivalent conditions are satisfied.

1. There exist a subset space model S and some neighborhood situation p, U in S such that $p, U \models \varphi$.
2. There exist a cross axiom model M and some point w in M such that $M, w \models \varphi$.

Example 3.26. We continue Example 3.19. Let A and B be two different propositional variables. We have seen that the formula

$$\varphi \equiv \Box\Diamond K(\neg A \vee \Diamond K \neg B) \wedge \Box\Diamond L(A \wedge \Box LB)$$

has an infinite subset space model but no finite subset space model. By Lemma 3.23 it has a cross axiom model as well. But we can show more: φ even has a *finite* cross axiom model:



Using cross axiom models and a filtration argument, decidability of the satisfiability problem for SSL is proved in [22] and in a simplified version in [58]. The following lemma follows from considerations in [22, Section 2.3.]

Lemma 3.27. *Every satisfiable formula $\varphi \in \mathcal{L}$ has a finite cross axiom model $M := (W, \xrightarrow{\Diamond}, \xrightarrow{L}, \sigma)$ of size doubly exponential in the length $|\varphi|$ of φ.*

In Chapter 4 we develop a tableau algorithm for SSL, providing alternative proofs of decidability and completeness for the class of cross axiom models.

3.4.5 Work linked to SSL

In the course of time several modifications of SSL have been investigated, offering additional properties for reasoning about space, knowledge or time. We can only mention some of these systems and some relationship to other logics.
While SSL was defined for arbitrary subset spaces, several extensions of SSL with additional axioms consider restricted families of sets, representing different spatial properties. We give some examples.

- Georgatos [29] gave a sound and complete axiomatization for subset spaces that are complete lattices.
- In [40] Heinemann studies spaces which satisfy finite chain conditions of various sorts.
- An important extension of SSL is the logic called topologic with the following additional axioms for 'real' topological spaces:

  WD: $\Diamond\Box\varphi \to \Box\Diamond\varphi$
  sound for weakly directed spaces and

  Un:$\Diamond\varphi \wedge L\Diamond\psi \to \Diamond\left(\Diamond\varphi \wedge L\Diamond\psi \wedge K\Diamond L(\varphi \vee \psi)\right)$
  sound for spaces closed under binary unions.

  These axioms are among others presented by Moss and Parikh in [71]. They were first determined by Georgatos in his thesis [30] and proved to be complete for topological spaces. Georgatos also showed that topologic has the finite model property. The decidability proof presented in [22] uses argumentation different from that of Georgatos and links topologic to early work of Tarski and McKinsey. Heinemann showed in [45] that topologic is also complete with respect to the Cantor space, i.e., the set of all infinite $0-1$-sequences endowed with the initial segment topology.
  Note that axiom (WD) is not complete for the class of directed spaces. In [95] it is shown that axiomatizing this class in the language of topologic requires an infinite set of axiom schemes.
  In [42] Heinemann adds an *overlap operator* $\bigcirc$ to the language of topologic, extending the semantics to

  $$x, U \models \bigcirc\varphi \quad \text{iff} \quad x, V \models \varphi \text{ for all } V \in \mathcal{O} \text{ such that } x \in V.$$

  With this new operator, Heinemann extends topologic to a system called ET which provides a *finite* axiomatization of *directed spaces.*

Adding *nominals* for opens then increases the expressive power to capture *weak connectedness* of the space.
Nominals are simply propositional variables whose valuation is always a singleton set and Heinemann considered a variety of adding nominals to the language of topologic, see e.g. [43]. Hybrid extensions of SSL have also been investigated by Wáng [92].
Among the modal logics for spatial reasoning the separate quantification over points (K) and set of points ($\Box$) is a special feature of topologic and SSL.

Temporal aspects of the effort operator are investigated by Heinemann and Georgatos. Georgatos studies tree-like spaces (given two open sets in a tree-like topological space, they are either disjoint, or one is a subset of the other) [31]. This system can be seen as generalization of modal branching time logics. In [41] Heinemann adds a *nextstep* operator and together with the $\Box$-operator this operator allows to express statements like *next time* and *always in the future*. This is a step towards generalizing linear time.

Subset space semantics has also attracted considerable attention in the epistemic context. Again, we list some examples:

- Georgatos presents in [32] a variant of SSL that handles arbitrary changes of the agent's epistemic state, that is, changes that do not necessarily result in knowledge increase. He proves decidability and completeness of this system for subset spaces.

- Extensions to multi-agent settings have been presented by the following authors:

  - A first approach to *multi-agent* settings was proposed by Pacuit and Parikh in [73]. They consider a set of agents connected in a *communication graph*, and such that agent i may receive information from agent j only if there is an edge from i to j. The arising logic uses a language very similar to topologic, and it is shown that for each graph, the logic is decidable, and completely characterizes the graph. But this approach is based on relational semantics rather than on subset spaces.

  - Other attempts at multi-agent versions of SSL have been made by Başkent [10] and Wen et al. [96], but both present no meta-logical results and have problems with the semantics of nested epistemic formulas, as was pointed out in [93].

- The first sound and complete axiomatization for multi-agent epistemic subset space logic as a conservative extension of the single agent case in the spirit of [23] was presented by Wáng and Ågotnes in [93]. Their language, already used in e.g. [10], replaces the K and $\Box$ modalities of the single-agent SSL by K_i and $\Box_i$ for each agent i. As an important feature, their agents have S5 knowledge as in the single-agent case.
- Heinemann investigates various scenarios to reason about the knowledge of more than one agent, see e.g. [44,46–48]. We cite his reflections in [49] on new and previous work and his comparison with the work of Wáng and Ågotnes:

  > In the paper [44], an attempt was made to obtain an appropriate multi-agent version of LSS [we use the abbreviation SSL]. The key idea behind that approach is to incorporate the agents in terms of additional modalities. This leads to an essential modification of the logic (namely to a *hybridization* in the spirit of [15], Sect. 7.3), while the original semantics basically remains unchanged. The 'right' generalization of LSS to multiple agents has then been found by Wáng and Ågotnes; see [93]. Their partition semantics, however, assigns sets of sets of subsets to the agents, complicating the handling of the system considerably. Thus, one might still ask how far-reaching the 'naive' subset space semantics for multiple agents is. In this paper, we argue that the effect of agents being exclusively geared to *group knowledge* can satisfactorily be embodied on that basis. In doing so, we follow a two-faced principle. The neighborhood components of any subset space are as of now (and here for the first time) viewed as knowledge states of the given group G of agents on the one hand, on the other hand, the agents are represented by the actions they are able to perform at any state; moreover, these actions are of such a nature that no decrease of the knowledge of G can result.[1] In other words, only (the improvement in) group knowledge is of interest to us and the individual knowledge of the agents is neglected here. In this way, the semantic problems with the latter can be avoided and a multi-agent

[1]This is in accordance with the general setting in the context of subset spaces.

logic of subset spaces can nevertheless still be brought about for an interesting special case.

- It is worth to mention that Wáng and Ågotnes show in [93] that (multi-agent) SSL has a relational aspect in the sense that every subset model has a one-to-one correspondence to a certain multi-dimensional relational model.

- Because of the dynamic nature of the effort modality there is also a natural connection between SSL and dynamic logics. *Dynamic logic* (DL) was developed by Vaughan Pratt [78] as an approach to assigning meaning to *Hoare logic* by expressing the Hoare formula $p\{a\}q$ as $p \to [a]q$ with the meaning that after an execution of program a q holds under the condition that p holds before. For details see [39]. Pratt's original dynamic logic was a first-order modal logic, since p and q may contain first-order components such as functions. A propositional variant *PDL* (Propositional Dynamic Logic) was derived by Fischer and Ladner [24].
  Dynamic epistemic logic (DEL) (pronounced "dell") combines dynamic and epistemic aspects and is a very active area with applications in e.g. Belief Revision, multi-agent and distributed systems, Artificial Intelligence, Non-monotonic Reasoning, and Epistemic Game Theory. It is a framework dealing with the change of the knowledge of agents, where the change depends on an event or *announcement* of some formula ψ. Such an announcement can be *public* available to all agents or *private* available only to selected agents. Variants of DEL are e.g. *PAL* (Public Announcement Logic), *APAL* (Arbitrary Public Announcement Logic) or *sPAL* (semi-public Announcement Logic), named according to the type of announcement. More about DEL and announcements can be found in [5, 8, 9, 77, 89, 90].
  Although the connection between dynamic logics and SSL is quite intuitive, it is by no means easy to establish. Details about this connection can be found among others in [3, 6, 11, 13, 14, 94]. In [7] Baltag, Özgün, and Sandoval present the system *APALM* which augments APAL with memory. They point out that there is a deep connection between APALM and SSL. Baltag, Fiutek, and Smets [4] show that full DDL (here agents revise their beliefs about the world *and* about their own beliefs) is a generalization of SSL.

Chapter 4

ESPACE Algorithms for K4 $\times$ S5, S4 $\times$ S5, and SSL

In this chapter we prove that the satisfiability problems of the three bimodal logics K4 $\times$ S5, S4 $\times$ S5, and SSL are in ESPACE. Actually, we prove the following theorem.

Theorem 4.1. *1. The satisfiability problem of the bimodal logic* K4 $\times$ S5 *can be decided in space* $O(n \cdot 2^{3n})$.

2. The satisfiability problems of the two bimodal logics S4 $\times$ S5 *and* SSL *can be decided in space* $O(n \cdot 2^{2n})$.

For K4 $\times$ S5 and S4 $\times$ S5 it was already known that their satisfiability problems are in N2EXPTIME [61, Theorem 5.28], that is, they can be solved by a nondeterministic Turing machine working in doubly exponential time. It is known as well that for any SSL-satisfiable formula there exists a cross axiom model of at most doubly exponential size [22, Section 2.3]. This shows that the complexity of the satisfiability problem of SSL is in N2EXPTIME as well.

We present decision algorithms for these problems that are based on certain kinds of tableaux. The origins of the tableau method for classical logic can be found in Beth [12], Hintikka [50] and, further developed, in Smullyan [84], Kanger [53], Rautenberg [79] and Schütte [81].

In the propositional case the tableau method for a given formula φ looks for a consistent set of subformulas of φ that arises by decomposing φ according to the meaning of the Boolean connectives and that cannot be further decomposed. For modal formulas things are more complicated, since the tableau for a satisfiable formula has to mirror the structure of a model for the formula (if one exists). Decomposing formulas with modal operator like $\Box\psi$ or

$\Diamond\psi$ leads to formulas that make assertions about *successor worlds.* Thus the tableau construction must in some way refer to the accessibility relations. Details about tableau methods for modal logics can be found in the following sources: In the book [25] and the chapter [26] by Melvin Fitting, in [34] by Rajeev Goré, in [35] by Guido Governatori, and in [2] by Baader and Sattler. The chapter is organized as follows.

- We start with the definition of what we call *partial tableaux*, for each of the three logics.
- We then show that the existence of a partial tableau for a bimodal formula φ is equivalent to its satisfiability in the respective class of models.
- We present recursive tableau algorithms that decide if there exists a partial tableau for a given bimodal formula φ or not, and we prove the correctness of these algorithms. They are somewhat similar to the recursive algorithm of Ladner [63] for the modal logic S4.
- We show that, for $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$, given a bimodal formula φ of length n the space used by the algorithm for the logic X is of the order $O(n \cdot |\mathcal{T}_\varphi^X|^3)$ where $\mathcal{T}_\varphi^X$ is the set of all so-called X-tableau-sets with respect to φ (to be defined in Section 4.1). Note that it is obvious that $|\mathcal{T}_\varphi^X| \leqslant 2^n$. Thus, we establish $O(n \cdot 2^{3n})$ as an upper bound for the space complexity of the satisfiability problems of all three logics.
- Then we consider the cases $X \in \{\mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$. By an additional counting argument, we show that $|\mathcal{T}_\varphi^X| \leqslant 2^{2n/3}$, for all $n \geqslant 3$, where φ is any bimodal formula and n its length. Thus, the algorithms for $X \in \{\mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$ actually work in space $O(n \cdot 2^{2n})$. This can certainly be improved even further. A similar counting argument could be applied in the case $X = \mathrm{K4} \times \mathrm{S5}$ as well, but in order to do that one should slightly change the definition of tableau sets, and even then the gain is smaller. Therefore, this is not worked out here.

Although it might seem somewhat paradox that we establish ESPACE as an upper bound and EXPSPACE-hardness as a lower bound this is not a contradiction. This phenomenon is due to the fact that the complexity class ESPACE is not closed under reduction, neither reductions running in polynomial time nor those running in logarithmic space.
We would like to point out that Section 4.4 starts with some general combinatorial observations on certain binary relations that may be of interest elsewhere as well.

Due to the similarity of the three logics we can do much work in parallel for all three logics.

4.1 The Definition of Tableaux for K4 × S5, S4 × S5, and SSL

At the beginning let us have a few thoughts about the construction of a tableau for φ. We took only $\Box$ and K as primitive modal operators because it often makes proofs shorter. But it is perhaps more understandable to talk about how to handle $\Diamond$- and L-formulas. These formulas are introduced as abbreviations of negated $\Box$- and K-formulas, respectively, and they are the ones that require appropriate successor points. So in informal descriptions we will talk about $\Diamond$- and L-formulas while in formal parts we only use the original operators.
We will construct tableaux not as usual brick by brick, we will instead use prefabricated parts.

- Instead of expanding a set of formulas step-by-step to a propositional tableau we work with complete *tableau-sets* as defined in Definition 4.2.1.

- The next step is to combine tableau-sets to sets of tableau-sets, called *tableau-clouds*, under the conditions given in Definition 4.2.3.

Tableau-clouds are somewhat similar to mosaics [72]. They mirror the $\xrightarrow{L}$-equivalence classes in corresponding models. The benefit of working with tableau-clouds is twofold: On the one hand, we only have to take care of $\Diamond$-formulas because in tableau-clouds all L-formulas are satisfied within the tableau-cloud. On the other hand, demanded commutativity properties are automatically satisfied if we meet the conditions for sequences of tableau-clouds defined in Definition 4.3. Commutativity is hard to guarantee if one builds tableaux from single formula sets.
The tableaux we construct are sets of tableau-clouds. We construct them recursively and pathwise. A $\Diamond$-formula may demand that there exists a suitable successor to an element in a tableau-cloud. In order to arrive at a finite tableau we will not immediately try to construct a suitable new successor tableau-cloud containing a suitable successor element but first check whether in the already constructed sequence of tableau-clouds there is a suitable one that would lead to the satisfaction of the currently considered $\Diamond$-formula. Thus, one might say that the algorithm tries to construct backwards loops whenever possible.

The backwards loops and the recursive design of the intended algorithms result in the need for *partial tableaux for a sequence of tableau-clouds.* Assume that we have to satisfy a formula $\Diamond\chi$ occurring in some tableau-cloud $\mathcal{C}$ at some component p, that cannot be satisfied by a backwards loop to one of the predecessors $\mathcal{C}_0, \ldots, \mathcal{C}_{m-1}$ of $\mathcal{C}$. Then we try all tableau-clouds $\mathcal{C}'$ that contain χ in some component q such that $\mathcal{C}'$ can be a successor of $\mathcal{C}$ and p can be linked to q, until one recursive tableau search for $\mathcal{C}'$ gives a positive feedback. Because of backwards loops that might be possible, we hand over to the new instance of the algorithm not only $\mathcal{C}'$ but also the sequence $\mathcal{C}_0, \ldots \mathcal{C}_{m-1}, \mathcal{C}$. Additionally we hand over the formula φ that determines the set $\mathrm{sf}(\varphi)$ and the set of tableau-clouds defined below.
We speak of a *partial* tableau for the sequence $(\varphi, \mathcal{C}_0, \ldots \mathcal{C}_{m-1}, \mathcal{C})$ because in the present instance of the algorithm we do not care whether the elements of the sequence $(\mathcal{C}_0, \ldots \mathcal{C}_{m-1})$ can be provided with all successors needed to satisfy their $\Diamond$-formulas. This is checked by other instances of the algorithm. We start with the definition of tableau-sets and tableau-clouds as the building blocks of the aimed tableaux.

Definition 4.2 (Tableau-sets and Tableau-clouds)**.** Let φ be a bimodal formula and let $X \in \{\mathrm{K4} \times \mathrm{S5},\ \mathrm{S4} \times \mathrm{S5},\ \mathrm{SSL}\}$.

1. A K4 × S5-*tableau-set with respect to* φ is a subset $F \subseteq \mathrm{sf}(\varphi)$ such that the following conditions are satisfied for all $\psi \in \mathrm{sf}(\varphi)$:

   (a) If $\psi = \neg\chi$ then $(\psi \in F \iff \chi \notin F)$.

   (b) If $\psi = (\chi_1 \wedge \chi_2)$ then $(\psi \in F \iff (\chi_1 \in F \text{ and } \chi_2 \in F))$.

   (c) If $\psi = K\chi$ then $(\psi \in F \Rightarrow \chi \in F)$.

2. For $X \in \{\mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$ an X-*tableau-set with respect to* φ is a subset $F \subseteq \mathrm{sf}(\varphi)$ such that for all $\psi \in \mathrm{sf}(\varphi)$ the conditions (a), (b), and (c) of a K4 × S5-tableau-set with respect to φ and additionally the following condition are satisfied:

   (d) If $\psi = \Box\chi$ then $(\psi \in F \Rightarrow \chi \in F)$.

3. The set $\mathcal{T}_\varphi^X$ of all X-tableau-sets with respect to φ is defined by

$$\mathcal{T}_\varphi^X := \{F \subseteq \mathrm{sf}(\varphi) \mid F \text{ is an } X\text{-tableau-set with respect to } \varphi\}.$$

4. An X-*tableau-cloud with respect to* φ is a subset $\mathcal{F} \subseteq \mathcal{T}_\varphi^X$ such that the following conditions are satisfied:

   (a) For all $F, G \in \mathcal{F}$,
   $F \cap \mathcal{L}_K = G \cap \mathcal{L}_K$
   (b) For all χ with $K\chi \in \mathrm{sf}(\varphi)$,
   if $\chi \in \bigcap_{F \in \mathcal{F}} F$ then $K\chi \in \bigcap_{F \in \mathcal{F}} F$.

5. The set $\mathfrak{C}^X_\varphi$ of all X-tableau-clouds with respect to φ is defined by

$$\mathfrak{C}^X_\varphi := \{\mathcal{F} \subseteq \mathcal{T}^X_\varphi \mid \mathcal{F} \text{ is an } X\text{-tableau-cloud with respect to } \varphi\}.$$

Before we come to the definition of tableaux we specify the conditions under which tableau-sets resp. tableau-clouds can be composed into a *sequence.*

Definition 4.3 (Sequences of Tableau-sets and of Tableau-clouds)**.** Let $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$. Let φ be a bimodal formula, let $F, G \in \mathcal{T}^X_\varphi$, and let $\mathcal{F}, \mathcal{G} \in \mathcal{P}(\mathcal{T}^X_\varphi)$.

1. We say that G *can be an X-successor of* F and write shortly $F \preccurlyeq_X G$ if the following conditions are satisfied:
   (a) in the case $X = \mathrm{K4} \times \mathrm{S5}$ the conditions
   $$F \cap \mathcal{L}_\Box \subseteq G \text{ and}$$
   $$\{\psi \in \mathcal{L} \mid \Box\psi \in F\} \subseteq G.$$
   (b) in the case $X = \mathrm{S4} \times \mathrm{S5}$ the condition
   $$F \cap \mathcal{L}_\Box \subseteq G.$$
   (c) in the case $X = \mathrm{SSL}$ the conditions
   $$F \cap \mathcal{L}_\Box \subseteq G \text{ and}$$
   $$F \cap AT = G \cap AT.$$
2. We say that $\mathcal{G}$ *can be an X-successor of* $\mathcal{F}$ and write shortly $\mathcal{F} \preccurlyeq_X \mathcal{G}$ if the following conditions are satisfied:
   (a) in the case of $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}\}$ the two conditions
      i. For all $G \in \mathcal{G}$ there exists some $F \in \mathcal{F}$ such that $F \preccurlyeq_X G$.
      ii. For all $F \in \mathcal{F}$ there exists some $G \in \mathcal{G}$ such that $F \preccurlyeq_X G$.
   (b) in the case of $X = \mathrm{SSL}$ the condition
      i. For all $G \in \mathcal{G}$ there exists some $F \in \mathcal{F}$ such that $F \preccurlyeq_{\mathrm{SSL}} G$.

3. We define a binary relation $\equiv_X$ on $\mathcal{P}(\mathcal{T}_\varphi^X)$ by
$$\mathcal{F} \equiv_X \mathcal{G} :\iff (\mathcal{F} \leqslant_X \mathcal{G} \text{ and } \mathcal{G} \leqslant_X \mathcal{F}).$$

4. Finally, we define a binary relation $<_X$ on $\mathcal{P}(\mathcal{T}_\varphi^X)$ by
$$\mathcal{F} <_X \mathcal{G} :\iff (\mathcal{F} \leqslant_X \mathcal{G} \text{ and not } \mathcal{G} \leqslant_X \mathcal{F}).$$

Lemma 4.4. *Let φ be a bimodal formula.*

1. *The relation $\preccurlyeq_{\mathrm{K4}\times\mathrm{S5}}$ on $\mathcal{T}_\varphi^{\mathrm{K4}\times\mathrm{S5}}$ is transitive.*
2. *The relation $\leqslant_{\mathrm{K4}\times\mathrm{S5}}$ on $\mathcal{P}(\mathcal{T}_\varphi^{\mathrm{K4}\times\mathrm{S5}})$ is transitive.*
3. *The relation $\equiv_{\mathrm{K4}\times\mathrm{S5}}$ on $\mathcal{P}(\mathcal{T}_\varphi^{\mathrm{K4}\times\mathrm{S5}})$ is transitive and symmetric.*

Proof. All assertions can be checked straightforwardly. □

Lemma 4.5. *Let $X \in \{\mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$, and let φ be a bimodal formula.*

1. *The relation $\preccurlyeq_X$ on $\mathcal{T}_\varphi^X$ is a preorder.*
2. *The relation $\leqslant_X$ on $\mathcal{P}(\mathcal{T}_\varphi^X)$ is a preorder.*
3. *The relation $\equiv_X$ on $\mathcal{P}(\mathcal{T}_\varphi^X)$ is an equivalence relation.*

Proof. All assertions can be checked straightforwardly. □

Definition 4.6 (Partial Tableaux for a Sequence of Tableau-clouds)**.** Let $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$, and let φ be a bimodal formula. Let $(\mathcal{F}_0, \ldots, \mathcal{F}_m)$ for some $m \geqslant 0$ be a finite sequence of pairwise different X-tableau-clouds (that is, $\mathcal{F}_i \in \mathfrak{C}_\varphi^X$, for $i = 0, \ldots, m$) with respect to φ such that
$$\mathcal{F}_i \leqslant_X \mathcal{F}_{i+1}, \qquad \text{for all } i < m.$$

A *partial X-tableau for* $(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m)$ is a subset $\mathfrak{T} \subseteq \mathfrak{C}_\varphi^X$ satisfying the following two conditions:

1. $\mathcal{F}_i \in \mathfrak{T}$, for $i = 0, \ldots, m$.
2. For all $\mathcal{F} \in \mathfrak{T} \backslash \{\mathcal{F}_0, \ldots, \mathcal{F}_{m-1}\}$, for all $F \in \mathcal{F}$, and for all χ with $\Box\chi \in \mathrm{sf}(\varphi)$, if $\Box\chi \notin F$, then there exists some $\mathcal{G} \in \mathfrak{T}$ such that $\mathcal{F} \leqslant_X \mathcal{G}$ and such that there exists some $G \in \mathcal{G}$ with $F \preccurlyeq_X G$ and $\chi \notin G$.

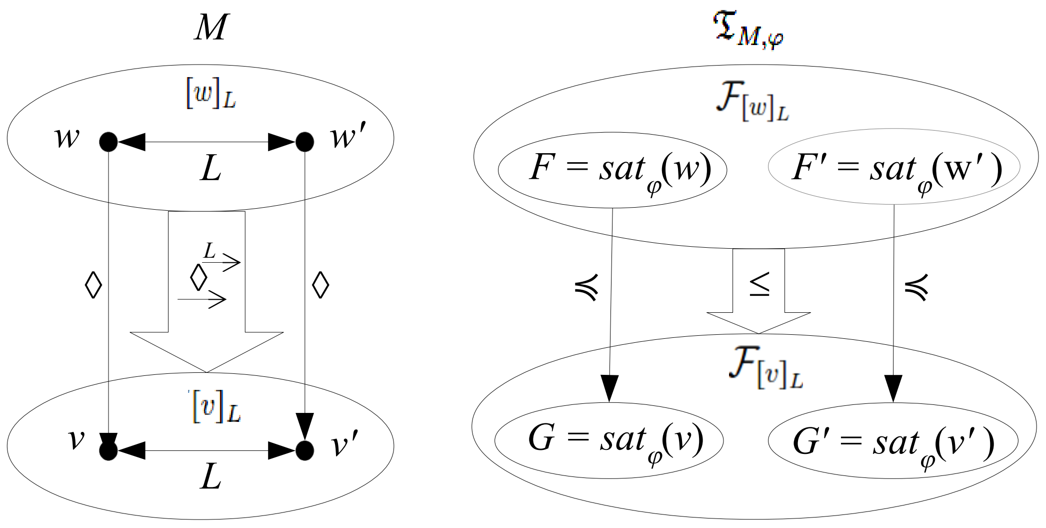
4.2 Tableaux and Models

In this section we show that the satisfiability of a bimodal formula φ is equivalent to the existence of a partial tableau for φ. This is true for all three considered bimodal logics, K4 × S5, S4 × S5, and SSL. We proceed as follows.

- Given a model M we define for any point w in M the tableau-cloud "of the point w". Then we show that the set of tableau-clouds of M is a partial tableau for the one-point sequence of tableau-clouds that consists of the tableau-cloud of some point w.
- Given a partial tableau for a one-point sequence of tableau-clouds, we construct a model that satisfies the same bimodal formulas, in a certain sense.

Definition 4.7 (Tableaux based on Models)**.** Let φ be a bimodal formula, and let $X \in \{\text{K4} \times \text{S5}, \text{S4} \times \text{S5}, \text{SSL}\}$. Let $M = (W, \overset{\Diamond}{\to}, \overset{L}{\to}, \sigma)$ be an X-model.

1. For all $w \in W$ we define
$$sat_\varphi(w) := \{\psi \in \text{sf}(\varphi) \mid M, w \models \psi\}.$$
2. For $q \in W_{\overset{L}{\to}}$ we define
$$\mathcal{F}_q := \{sat_\varphi(w) \mid w \in q\}.$$
3. Let
$$\mathfrak{T}_{M,\varphi} := \{\mathcal{F}_q \mid q \in W_{\overset{L}{\to}}\}.$$



Lemma 4.8. *Let $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$, and let φ be a bimodal formula. Let $M = (W, \overset{\Diamond}{\to}, \overset{L}{\to}, \sigma)$ be an X-model.*

1. *For all $w \in W$, the set $sat_\varphi(w)$ is an X-tableau-set with respect to φ.*
2. *For all $u, v \in W$, if $u \overset{\Diamond}{\to} v$ then $sat_\varphi(u) \preccurlyeq_X sat_\varphi(v)$.*
3. *For all $q \in W_{\overset{L}{\to}}$ the set $\mathcal{F}_q$ is an X-tableau-cloud with respect to φ.*
4. *For all $p, q \in W_{\overset{L}{\to}}$, if $p \overset{\Diamond}{\to}^{\overset{L}{\to}} q$ then $\mathcal{F}_p \preccurlyeq_X \mathcal{F}_q$.*
5. *For all $w \in W$, the set $\mathfrak{T}_{M,\varphi}$ is a partial X-tableau for $(\varphi, \mathcal{F}_{[w]_L})$.*

Proof. 1. This is straightforward to see. Note that in the cases $X \in \{\mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$ the sets $sat_\varphi(w)$ for $w \in W$ satisfy Condition (d) in Definition 4.2.2 because the relation $\overset{\Diamond}{\to}$ in an X-model is reflexive.

2.-4. All of these assertions are straightforward to check as well in each case for X.

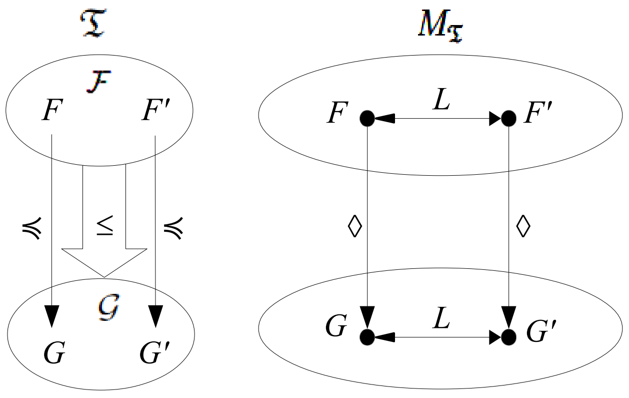
5. Let us fix some $w \in W$. It is clear that $\mathcal{F}_{[w]_L} \in \mathfrak{T}_{M,\varphi}$. Let us fix some $\mathcal{F} \in \mathfrak{T}_{M,\varphi}$ and some $F \in \mathcal{F}$. Let us assume that χ is a bimodal formula with $\Box\chi \in \mathrm{sf}(\varphi) \backslash F$. We have to show that there exists some $\mathcal{G} \in \mathfrak{T}_{M,\varphi}$ such that $\mathcal{F} \preccurlyeq_X \mathcal{G}$ and such that there exists some $G \in \mathcal{G}$ with $F \preccurlyeq_X G$ and $\chi \notin G$. Indeed, let us fix some point $u \in W$ with $F = sat_\varphi(u)$ and $\mathcal{F} = \mathcal{F}_{[u]_L}$. From $\Box\chi \notin F = sat_\varphi(u)$ we conclude $M, u \models \neg\Box\chi$, hence, $M, u \models \Diamond\neg\chi$. As M is an X-model there exists some point $v \in W$ with $u \overset{\Diamond}{\to} v$ and $M, v \models \neg\chi$. Let $G := sat_\varphi(v)$ and $\mathcal{G} := \mathcal{F}_{[v]_L}$. Then $\neg\chi \in G$, hence, $\chi \notin G$. Furthermore $G \in \mathcal{G}$ and $\mathcal{G} \in \mathfrak{T}_{M,\varphi}$. Finally, by the second assertion of this lemma, $u \overset{\Diamond}{\to} v$ implies $F = sat_\varphi(u) \preccurlyeq sat_\varphi(v) = G$. And it implies $[u]_L \overset{\Diamond}{\to}^{\overset{L}{\to}} [v]_L$, which, by the fourth assertion of this lemma, implies $\mathcal{F} = \mathcal{F}_{[u]_L} \preccurlyeq_X \mathcal{G}_{[v]_L} = \mathcal{G}$. □

Definition 4.9 (Models based on Tableaux)**.** Let φ be a bimodal formula, and let $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$. Let $\mathcal{F}_0$ be an X-tableau-cloud with respect to φ. Let $\mathfrak{T} \subseteq \mathfrak{C}_\varphi^X$ be a partial X-tableau for $(\varphi, \mathcal{F}_0)$. We define a quadruple

$$M_{\mathfrak{T}} = (W, \overset{\Diamond}{\to}, \overset{L}{\to}, \sigma)$$

consisting of a nonempty set W, of two binary relations $\xrightarrow{\Diamond}$ and $\xrightarrow{L}$ on W, and of a function $\sigma : AT \to \mathcal{P}(W)$ as follows:

$$\begin{array}{rcl} W & := & \{(\mathcal{F},F) \in \mathfrak{T} \times \mathcal{P}(\mathrm{sf}(\varphi)) \mid F \in \mathcal{F}\}, \\ (\mathcal{F},F) \xrightarrow{\Diamond} (\mathcal{G},G) & :\Longleftrightarrow & (\mathcal{F} \preccurlyeq_X \mathcal{G} \text{ and } F \preccurlyeq_X G), \\ & & \quad \text{for } (\mathcal{F},F),(\mathcal{G},G) \in W, \\ (\mathcal{F},F) \xrightarrow{L} (\mathcal{G},G) & :\Longleftrightarrow & \mathcal{F} = \mathcal{G}, \\ & & \quad \text{for } (\mathcal{F},F),(\mathcal{G},G) \in W, \\ \sigma(A) & := & \{(\mathcal{F},F) \in W \mid A \in F\}, \\ & & \quad \text{for } A \in AT. \end{array}$$



Lemma 4.10. *Let X, φ, $\mathcal{F}_0$ and $\mathfrak{T}$ be as in the previous definition.*

1. *The quadruple $M_{\mathfrak{T}}$ is an X-model.*

2. *(Truth Lemma)*

$$(\forall \psi \in \mathrm{sf}(\varphi))\ (\forall (\mathcal{F},F) \in W)\ \ (M_{\mathfrak{T}}, (\mathcal{F},F) \models \psi \iff \psi \in F)\,.$$

Proof. 1. The relations $\preccurlyeq$ on $\mathcal{T}_\varphi^X$ and $\preccurlyeq_X$ on $\mathfrak{C}_\varphi^X$ are transitive. Hence, the relation $\xrightarrow{\Diamond}$ is transitive as well. Furthermore, in the cases $X \in \{\mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$ the relations $\preccurlyeq$ on $\mathcal{T}_\varphi^X$ and $\preccurlyeq_X$ on $\mathfrak{C}_\varphi^X$ are reflexive. Hence, in these cases the relation $\xrightarrow{\Diamond}$ is reflexive as well. It is clear that the relation $\xrightarrow{L}$ is an equivalence relation.

Next, we show that left commutativity holds. Let us consider pairs $(\mathcal{F},F),(\mathcal{G},G),(\mathcal{G}',G') \in W$ with

$$(\mathcal{F},F) \xrightarrow{\Diamond} (\mathcal{G},G) \ \text{ and } \ (\mathcal{G},G) \xrightarrow{L} (\mathcal{G}',G').$$

Then $\mathcal{G}' = \mathcal{G}$. Furthermore, $\mathcal{F} \preccurlyeq_X \mathcal{G}$ and $F \preccurlyeq_X G$. Due to $G' \in \mathcal{G}' = \mathcal{G}$ and $\mathcal{F} \preccurlyeq_X \mathcal{G}$ there exists some $F' \in \mathcal{F}$ with $F' \preccurlyeq_X G'$. We conclude $(\mathcal{F}, F') \xrightarrow{\Diamond} (\mathcal{G}, G')$. As $(\mathcal{F}, F) \xrightarrow{L} (\mathcal{F}, F')$ is clear, we have shown left commutativity.

In the cases $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}\}$ right commutativity is shown in the same way.

Finally, let us consider the case $X = \mathrm{SSL}$. We still need to show that in this case the persistence property holds true. For $(\mathcal{F}, F), (\mathcal{G}, G) \in W$, the condition $(\mathcal{F}, F) \xrightarrow{\Diamond} (\mathcal{G}, G)$ implies $F \preccurlyeq_{\mathrm{SSL}} G$ which, in turn, implies $F \cap AT = G \cap AT$. Hence, for any propositional variable A and any $(\mathcal{F}, F), (\mathcal{G}, G) \in W$ with $(\mathcal{F}, F) \xrightarrow{\Diamond} (\mathcal{G}, G)$ we have $A \in F \iff A \in G$, hence, $(\mathcal{F}, F) \in \sigma(A) \iff (\mathcal{G}, G) \in \sigma(A)$. Thus, the persistence property is satisfied. We have shown that $M_{\mathfrak{T}}$ is a cross axiom model.

2. Let us consider some $\psi \in \mathrm{sf}(\varphi)$. We wish to show

$$M_{\mathfrak{T}}, (\mathcal{F}, F) \models \psi \iff \psi \in F,$$

for all $(\mathcal{F}, F) \in W$. This is shown by structural induction. We distinguish the following cases:

- $\psi = A \in AT$.
  For $(\mathcal{F}, F) \in W$, the condition $M_{\mathfrak{T}}, (\mathcal{F}, F) \models A$ is equivalent to $(\mathcal{F}, F) \in \sigma(A)$, and by definition of σ, this is equivalent to $A \in F$.
- $\psi = \neg\chi$.
  In this case, the following four conditions are equivalent (the second and the third condition by induction hypothesis) for $(\mathcal{F}, F) \in W$:
  (a) $M_{\mathfrak{T}}, (\mathcal{F}, F) \models \psi$,
  (b) $M_{\mathfrak{T}}, (\mathcal{F}, F) \not\models \chi$,
  (c) $\chi \notin F$,
  (d) $\psi \in F$.
- $\psi = \psi_1 \wedge \psi_2$.
  This case is treated similarly.
- $\psi = K\chi$.
  Let us first assume $M_{\mathfrak{T}}, (\mathcal{F}, F) \models K\chi$. We wish to show $K\chi \in F$. By the semantics definition $M_{\mathfrak{T}}, (\mathcal{F}, G) \models \chi$, for all $G \in \mathcal{F}$. By induction hypothesis, $\chi \in G$ for all such G. Thus, we have $\chi \in$

$\bigcap_{G \in \mathcal{F}} G$. As $\mathcal{F}$ is an X-tableau-cloud, we obtain $K\chi \in \bigcap_{G \in \mathcal{F}} G$. As $F \in \mathcal{F}$ as well we finally obtain $K\chi \in F$.

For the other direction let us consider some $(\mathcal{F}, F) \in W$, and let us assume $K\chi \in F$. We wish to show $M_{\mathfrak{T}}, (\mathcal{F}, F) \models K\chi$. As $F \in \mathcal{F}$ and $\mathcal{F}$ is a tableau-cloud, we have $F \cap \mathcal{L}_K = G \cap \mathcal{L}_K$, for all $G \in \mathcal{F}$. This implies $K\chi \in G$, for all $G \in \mathcal{F}$. As all such G are X-tableau-sets, we obtain $\chi \in G$, for all $G \in \mathcal{F}$. By induction hypothesis $M_{\mathfrak{T}}, (\mathcal{F}, G) \models \chi$, for all $G \in \mathcal{F}$. But this implies $M_{\mathfrak{T}}, (\mathcal{F}, F) \models K\chi$.

- $\psi = \Box\chi$.
  Let us first assume $M_{\mathfrak{T}}, (\mathcal{F}, F) \models \Box\chi$. We wish to show $\Box\chi \in F$. The assumption implies that $M_{\mathfrak{T}}, (\mathcal{G}, G) \models \chi$, for all $(\mathcal{G}, G) \in W$ with $(\mathcal{F}, F) \xrightarrow{\Diamond} (\mathcal{G}, G)$. By induction hypothesis we obtain $\chi \in G$, for all such $(\mathcal{G}, G) \in W$. Hence, $\chi \in G$ for all $(\mathcal{G}, G) \in W$ satisfying $\mathcal{F} \leqslant_X \mathcal{G}$ and $F \preccurlyeq_X G$. The second condition in Definition 4.6 implies $\Box\chi \in F$.

  For the other direction, let us consider some $(\mathcal{F}, F) \in W$ and let us assume $\Box\chi \in F$. We wish to show $M_{\mathfrak{T}}, (\mathcal{F}, F) \models \Box\chi$. It is sufficient to show that $M_{\mathfrak{T}}, (\mathcal{G}, G) \models \chi$ for all $(\mathcal{G}, G) \in W$ with $(\mathcal{F}, F) \xrightarrow{\Diamond} (\mathcal{G}, G)$. By induction hypothesis it is sufficient to show that $\chi \in G$ for all $(\mathcal{G}, G) \in W$ with $(\mathcal{F}, F) \xrightarrow{\Diamond} (\mathcal{G}, G)$. But $(\mathcal{F}, F) \xrightarrow{\Diamond} (\mathcal{G}, G)$ implies $F \preccurlyeq_X G$. In the case $X = \mathrm{K4} \times \mathrm{S5}$ this condition and the assumption $\Box\chi \in F$ immediately imply $\chi \in G$. In the cases $X \in \{\mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$ the condition $F \preccurlyeq_X G$ and the assumption $\Box\chi \in F$ imply $\Box\chi \in G$. Using additionally the fact that G is an X-tableau-set, we obtain $\chi \in G$. □

We are now ready to state and prove the main result of this section.

Proposition 4.11. *Let $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$, and let φ be a bimodal formula. The following two conditions are equivalent.*

1. *φ is X-satisfiable.*
2. *There exists an X-tableau-cloud $\mathcal{F}_0$ such that there exist a set $F \in \mathcal{F}_0$ with $\varphi \in F$ and a partial X-tableau for $(\varphi, \mathcal{F}_0)$.*

Proof. Let us first assume that φ is X-satisfiable. Then there are some X-model $M = (W, \xrightarrow{\Diamond}, \xrightarrow{L}, \sigma)$ and some point $w \in W$ such that $M, w \models \varphi$. According to Lemma 4.8.5 the set $\mathfrak{T}_{M,\varphi}$ defined in Definition 4.7 is a partial

X-tableau for $(\varphi, \mathcal{F}_{[w]_L})$. Due to $M, w \models \varphi$ the formula φ is an element of the set $F := sat_\varphi(w)$ and this in turn is an element of $\mathcal{F}_{[w]_L}$.
For the other direction let us assume that there exist an X-tableau-cloud $\mathcal{F}_0$, an X-tableau-set $F \in \mathcal{F}_0$ with $\varphi \in F$ and a partial X-tableau $\mathfrak{T}$ for $(\varphi, \mathcal{F}_0)$. According to Lemma 4.10.1 the quadruple $M_\mathfrak{T} = (W, \xrightarrow{\Diamond}, \xrightarrow{L}, \sigma)$ defined in Definition 4.9 is an X-model. Furthermore, we have $F \in \mathcal{F}_0$, hence, the pair $(\mathcal{F}_0, F)$ is an element of W. Finally, due to $\varphi \in F$ and due to Lemma 4.10.2 we obtain $M_\mathfrak{T}, (\mathcal{F}_0, F) \models \varphi$. Hence, φ is X-satisfiable. □

This shows that we can replace the search for a model of φ by the search for a partial tableau for φ. We will organize this search by recursive algorithms that will be described in the following section.

4.3 The Tableau Algorithms

The algorithms use the following recursive procedures $alg_{\mathrm{K4\times S5}}$, $alg_{\mathrm{S4\times S5}}$, and alg_{SSL}.

Definition 4.12 (Procedures $alg_{\mathrm{K4\times S5}}$, $alg_{\mathrm{S4\times S5}}$, and alg_{SSL})**.** Assume that $X \in \{\mathrm{K4 \times S5}, \mathrm{S4 \times S5}, \mathrm{SSL}\}$. Given a bimodal formula φ and for some $m \geqslant 0$ a sequence $(\mathcal{F}_0, \ldots, \mathcal{F}_m)$ of pairwise different tableau-clouds $\mathcal{F}_i \in \mathfrak{C}_\varphi$ with $\mathcal{F}_i \leqslant_X \mathcal{F}_{i+1}$, for all $i < m$, the algorithm

$$alg_X(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m)$$

checks for every pair $(\Box\chi, F) \in \mathrm{sf}(\varphi) \times \mathcal{F}_m$ with $\Box\chi \notin F$ first

(I) whether there exists some $i \in \{0, \ldots, m\}$ with $\mathcal{F}_m \leqslant_X \mathcal{F}_i$ and such that there exists some $G \in \mathcal{F}_i$ with $F \leqslant_X G$ and $\chi \notin G$,

and, if this is not the case,

(II) whether there exists some tableau-cloud $\mathcal{F}_{m+1} \in \mathfrak{C}^X_\varphi \backslash \{\mathcal{F}_0, \ldots, \mathcal{F}_m\}$ with $\mathcal{F}_m \leqslant_X \mathcal{F}_{m+1}$ such that
- there exists some $G \in \mathcal{F}_{m+1}$ with $F \leqslant_X G$ and $\chi \notin G$ and
- $alg_X(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m, \mathcal{F}_{m+1})$ returns "yes".

If for every pair $(\Box\chi, F) \in \mathrm{sf}(\varphi) \times \mathcal{F}_m$ with $\Box\chi \notin F$ Condition (I) or Condition (II) is satisfied then $alg_X(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m)$ returns "yes", otherwise it returns "no". This ends the description of the algorithm $alg_X(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m)$.

We show that its works correctly, for each $X \in \{\mathrm{K4 \times S5}, \mathrm{S4 \times S5}, \mathrm{SSL}\}$.

Proposition 4.13. *Let $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$. Let φ be a bimodal formula. Let $(\mathcal{F}_0, \ldots, \mathcal{F}_m)$ for some $m \geqslant 0$ be a sequence of pairwise different tableau-clouds with respect to φ satisfying $\mathcal{F}_i \leqslant_X \mathcal{F}_{i+1}$, for $i < m$. Then $alg_X(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m)$ returns "yes" if, and only if, there exists a partial X-tableau for $(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m)$.*

Proof. We show each direction of this equivalence by induction over the cardinality of the following set

$$S(\mathcal{F}_0, \ldots, \mathcal{F}_m) := \{\mathcal{G} \in \mathfrak{C}_\varphi^X \backslash \{\mathcal{F}_0, \ldots, \mathcal{F}_m\} \mid \mathcal{F}_m \leqslant_X \mathcal{G}\}.$$

Note that this set is finite because $\mathfrak{C}_\varphi^X$ is a finite set.
Let us first assume that there exists a partial X-tableau for $(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m)$. We claim that $alg_X(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m)$ will return "yes". This is clear if there are no pairs $(\Box\chi, F) \in \mathrm{sf}(\varphi) \times \mathcal{F}_m$ with $\Box\chi \notin F$, or if for all such pairs Condition (I) is true. So, let us consider the case when there are such pairs for which Condition (I) is not true. Let us fix a pair $(\Box\chi, F) \in \mathrm{sf}(\varphi) \times \mathcal{F}_m$ with $\Box\chi \notin F$ such that (I) is not true for this pair. We claim that (II) is true for this pair. Consider a partial X-tableau $\mathfrak{T}$ for $(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m)$. Due to $\Box\chi \in \mathrm{sf}(\varphi)\backslash F$ and $F \in \mathcal{F}_m$ and due to the second condition in Definition 4.6 there exists an element $\mathcal{G} \in \mathfrak{T}$ with $\mathcal{F}_m \leqslant_X \mathcal{G}$ such that there exists some $G \in \mathcal{G}$ with $F \preccurlyeq_X G$ and $\chi \notin G$. The set $\mathcal{F}_{m+1} := \mathcal{G}$ is an X-tableau-cloud with $\mathcal{F}_m \leqslant_X \mathcal{F}_{m+1}$, with $G \in \mathcal{F}_{m+1}$, with $F \preccurlyeq_X G$, and with $\chi \notin G$. Furthermore, as (I) is not true for the pair $(\Box\chi, F)$, we have $\mathcal{F}_{m+1} \notin \{\mathcal{F}_0, \ldots, \mathcal{F}_m)$. This shows that $\mathcal{F}_0, \ldots, \mathcal{F}_m, \mathcal{F}_{m+1}$ are pairwise different. Thus, $\mathfrak{T}$ is a partial X-tableau for $(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_{m+1})$. Due to $\mathcal{F}_{m+1} \notin \{\mathcal{F}_0, \ldots, \mathcal{F}_m\}$, the set $S(\mathcal{F}_0, \ldots, \mathcal{F}_m, \mathcal{F}_{m+1})$ contains strictly less elements than the set $S(\mathcal{F}_0, \ldots, \mathcal{F}_m)$. Hence, the algorithm $alg_X(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m, \mathcal{F}_{m+1})$ returns "yes" by induction hypothesis and hence, (II) is true. This ends our proof by induction of the claim that if a partial X-tableau for $(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m)$ exists then $alg_X(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m)$ will return "yes".
For the other direction, let us assume that $alg_X(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m)$ returns "yes". In the following we will construct a partial X-tableau $\mathfrak{T}$ for $(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m)$. Let Pairs be the set of all pairs $(\Box\chi, F) \in \mathrm{sf}(\varphi) \times \mathcal{F}_m$ with $\Box\chi \notin F$. As by assumption the algorithm $alg_X(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m)$ returns "yes" the set Pairs is the disjoint union of the sets $\mathrm{Pairs}_{I,0}, \ldots, \mathrm{Pairs}_{I,m}$, Pairs_{II}, where

- $\mathrm{Pairs}_{I,i}$, for $i \in \{0, \ldots, m\}$, is the set of all pairs $(\Box\chi, F) \in$ Pairs such that (I) is satisfied and i is the smallest number in $\{0, \ldots, m\}$ such that $\mathcal{F}_m \leqslant_X \mathcal{F}_i$ and such that there exists some $G \in \mathcal{F}_i$ with $F \preccurlyeq_X G$ and $\chi \notin G$,

- Pairs_{II} is the set of all pairs in Pairs such that (I) is not satisfied but (II) is.

Let k be the number of pairs in Pairs_{II}, and let $(\Box\chi_j, F_j)$ for $j = 0, \dots, k-1$ be the elements of Pairs_{II}. For each $j \in \{0, \dots, k-1\}$ there exists a tableau-cloud $\mathcal{F}_{m+1}^{(j)} \in \mathfrak{C}_\varphi \backslash \{\mathcal{F}_0, \dots, \mathcal{F}_m\}$ with $\mathcal{F}_m \preccurlyeq_X \mathcal{F}_{m+1}^{(j)}$ such that there exists some $G \in \mathcal{F}_{m+1}^{(j)}$ with $F_j \preccurlyeq_X G$ and $\chi_j \notin G$ and such that $alg_X(\varphi, \mathcal{F}_0, \dots, \mathcal{F}_m, \mathcal{F}_{m+1}^{(j)})$ returns "yes". Furthermore, the set $S(\mathcal{F}_0, \dots, \mathcal{F}_m, \mathcal{F}_{m+1}^{(j)})$ contains less elements than the set $S(\mathcal{F}_0, \dots, \mathcal{F}_m)$, due to $\mathcal{F}_{m+1}^{(j)} \notin \{\mathcal{F}_0, \dots, \mathcal{F}_m\}$. Hence, by induction hypothesis, there exists a partial X-tableau $\mathfrak{T}^{(j)}$ for the sequence $(\varphi, \mathcal{F}_0, \dots, \mathcal{F}_m, \mathcal{F}_{m+1}^{(j)})$. We define

$$\mathfrak{T} := \bigcup_{j=0}^{k-1} \mathfrak{T}^{(j)}.$$

We claim that $\mathfrak{T}$ is a partial X-tableau for $(\varphi, \mathcal{F}_0, \dots, \mathcal{F}_m)$.
Indeed, it is clear that $\{\mathcal{F}_0, \dots, \mathcal{F}_m\} \subseteq \mathfrak{T}$ because $\{\mathcal{F}_0, \dots, \mathcal{F}_m\} \subseteq \mathfrak{T}^{(j)}$ even for every $j < k$. Let us consider some $\mathcal{F} \in \mathfrak{T} \backslash \{\mathcal{F}_0, \dots, \mathcal{F}_{m-1}\}$, some $F \in \mathcal{F}$, and some formula $\Box\chi \in \mathrm{sf}(\varphi) \backslash F$. We wish to show that there exists some $\mathcal{G} \in \mathfrak{T}$ such that $\mathcal{F} \preccurlyeq_X \mathcal{G}$ and such that there exists some $G \in \mathcal{G}$ with $F \preccurlyeq_X G$ and $\chi \notin G$. We distinguish the following two cases.

1. $\mathcal{F} \neq \mathcal{F}_m$.

   Then there exists a $j \in \{0, \dots, k-1\}$ with $\mathcal{F} \in \mathfrak{T}^{(j)} \backslash \{\mathcal{F}_0, \dots, \mathcal{F}_m\}$. As $\mathfrak{T}^{(j)}$ is a partial X-tableau for $(\varphi, \mathcal{F}_0, \dots, \mathcal{F}_m, \mathcal{F}_{m+1}^{(j)})$ there exists an X-tableau-cloud $\mathcal{G} \in \mathfrak{T}^{(j)}$ such that $\mathcal{F} \preccurlyeq_X \mathcal{G}$ and such that there exists some $G \in \mathcal{G}$ with $F \preccurlyeq_X G$ and $\chi \notin G$. As $\mathfrak{T}^{(j)}$ is a subset of $\mathfrak{T}$ we are done.

2. $\mathcal{F} = \mathcal{F}_m$.

   Then $(\Box\chi, F) \in$ Pairs. Either there exists a unique $i \in \{0, \dots, m\}$ with $(\Box\chi, F) \in \mathrm{Pairs}_{I,i}$ or $(\Box\chi, F) \in \mathrm{Pairs}_{II}$.

   In the first case $\mathcal{F}_m \preccurlyeq_X \mathcal{F}_i$ and there exists some $G \in \mathcal{F}_i$ with $F \preccurlyeq_X G$ and $\chi \notin G$. In this case we set $\mathcal{G} := \mathcal{F}_i$.

   In the second case there exists a number $j \in \{0, \dots, k-1\}$ with $(\Box\chi, F) = (\Box\chi_j, F_j)$. Then $\mathcal{F} = \mathcal{F}_m \preccurlyeq_X \mathcal{F}_{m+1}^{(j)}$, and there exists an X-tableau-set $G \in \mathcal{F}_{m+1}^{(j)}$ with $F \preccurlyeq_X G$ and with $\chi \notin G$. In this case we set $\mathcal{G} := \mathcal{F}_{m+1}^{(j)}$.

This shows that the procedure alg_X is correct. $\Box$

Now, with the procedures $alg_{\mathrm{K4}\times\mathrm{S5}}$, $alg_{\mathrm{S4}\times\mathrm{S5}}$, and alg_{SSL} at hand we can present tableau algorithms $ALG_{\mathrm{K4}\times\mathrm{S5}}$, $ALG_{\mathrm{S4}\times\mathrm{S5}}$, and ALG_{SSL} for the logics under consideration.

Definition 4.14 (Tableau Algorithms $ALG_{\mathrm{K4}\times\mathrm{S5}}$, $ALG_{\mathrm{S4}\times\mathrm{S5}}$, and ALG_{SSL})**.** Let $X \in \{\mathrm{K4}\times\mathrm{S5}, \mathrm{S4}\times\mathrm{S5}, \mathrm{SSL}\}$. Given a bimodal formula φ the algorithm $ALG_X(\varphi)$ lets $\mathcal{F}_0$ run through all X-tableau-clouds $\mathcal{F}_0 \in \mathfrak{C}_\varphi^X$ such that there exists some $F \in \mathcal{F}_0$ with $\varphi \in F$ and applies alg_X to $(\varphi, \mathcal{F}_0)$. It accepts φ iff $alg_X(\varphi, \mathcal{F}_0)$ returns "yes" for at least one such pair $(\varphi, \mathcal{F}_0)$.

Proposition 4.15. *Let $X \in \{\mathrm{K4}\times\mathrm{S5}, \mathrm{S4}\times\mathrm{S5}, \mathrm{SSL}\}$. The algorithm ALG_X accepts a bimodal formula φ if and only if φ is X-satisfiable.*

Proof. Let $X \in \{\mathrm{K4}\times\mathrm{S5}, \mathrm{S4}\times\mathrm{S5}, \mathrm{SSL}\}$. Let φ be a bimodal formula. The algorithm ALG_X accepts φ by definition if, and only if, there exists an X-tableau-cloud $\mathcal{F}_0 \in \mathfrak{C}_\varphi^X$ such that $\varphi \in F$ for some $F \in \mathcal{F}_0$ and such that $alg_X(\varphi, \mathcal{F}_0)$ returns "yes". According to Proposition 4.13 $alg_X(\varphi, \mathcal{F}_0)$ returns "yes" if, and only if, there exists a partial tableau for $(\varphi, \mathcal{F}_0)$. According to Proposition 4.11 there exists a tableau-cloud $\mathcal{F}_0 \in \mathfrak{C}_\varphi^X$ such that there exist a set $F \in \mathcal{F}_0$ with $\varphi \in F$ and a partial X-tableau for $(\varphi, \mathcal{F}_0)$ if, and only if, φ is X-satisfiable. $\square$

Finally let us point out that, whenever the algorithm $ALG_X(\varphi)$ makes a call $alg_X(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m)$ for some bimodal formula φ and some finite sequence $\mathcal{F}_0, \ldots, \mathcal{F}_m$ of X-tableau-sets, then all of these X-tableau-sets are pairwise different.

4.4 Upper Bounds for the Space Used by the Algorithms

It is the purpose of this section to prove the following proposition.

Proposition 4.16. *Let $X \in \{\mathrm{K4}\times\mathrm{S5}, \mathrm{S4}\times\mathrm{S5}, \mathrm{SSL}\}$. The algorithm ALG_X can be implemented on a multi-tape Turing machine so that it, given a bimodal formula φ of length n, does not use more than $O(n \cdot (n + |\mathcal{T}_\varphi^X|)^3)$ space.*

Before we prove this, let us deduce one of the assertions of Theorem 4.1

Proof of Theorem 4.1 in the case $X = \mathrm{K4}\times\mathrm{S5}$. We have presented an algorithm $ALG_{\mathrm{K4}\times\mathrm{S5}}$ that, according to Proposition 4.15, accepts a bimodal formula φ if, and only if, φ is $\mathrm{K4}\times\mathrm{S5}$-satisfiable. Let n be the length of φ.

There are at most n subformulas of φ. Hence, $|\mathcal{T}_\varphi^X| \leqslant 2^n$. By Proposition 4.16 the algorithm $ALG_{\mathrm{K4}\times\mathrm{S5}}$ can be implemented in such a way that it works in space $O(n \cdot 2^{3 \cdot n})$. □

In Section 4.5, for $X \in \{\mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$ we shall give a better upper bound for $|\mathcal{T}_\varphi^X|$ than 2^n.
Let $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$. The algorithm ALG_X calls the recursive procedure alg_X. It is clear that the space used by these algorithms is heavily influenced by the recursion depth of calls $alg_X(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m)$ that occur during the execution of $ALG_X(\varphi)$. Therefore, first we plan to give upper bounds for the recursion depth of these algorithms. As a first step for this we will give upper bounds for the maximum chain length of the transitive relation $\leqslant_X$ on $\mathcal{P}(\mathcal{T}_\varphi^X)$, for any bimodal formula φ.

4.4.1 Observations about the Maximum Chain Length

What is the maximum chain length of a transitive relation on a nonempty finite set? Let us define this. For any relation $\leqslant$ on a set S let the relation $<$ on S be defined by

$$s < t :\iff (s \leqslant t \text{ and not } t \leqslant s),$$

for any $s, t \in S$,

Lemma 4.17. *Let $\leqslant$ be a relation on a set S.*

1. *For $s, t \in S$, if $s < t$ then $s \neq t$.*
2. *If $\leqslant$ is a transitive relation on a set S then the relation $<$ on S is transitive as well.*

Proof. Let us consider some elements $s, t \in S$ with $s < t$. Then $s \leqslant t$. If $s = t$ then we would have $t \leqslant s$ as well, contradicting $s < t$.
Let us consider some elements $r, s, t \in S$ with $r < s$ and $s < t$. Then $r \leqslant s$ and $s \leqslant t$. The transitivity of $\leqslant$ implies $r \leqslant t$. We claim that $t \leqslant r$ is not true. For the sake of a contradiction, let us assume $t \leqslant r$. Then the transitivity of $\leqslant$ implies $s \leqslant r$ in contradiction to $r < s$. □

For any transitive relation $\leqslant$ on a finite, nonempty set S we define its *maximum chain length* $\mathrm{mcl}(\leqslant)$ to be the largest natural number l such that there exists a sequence $s_0, \ldots, s_l \in S$ with $s_i < s_{i+1}$, for all $i < l$, that is, such that

$$s_0 < s_1 < \ldots < s_l.$$

We call such a sequence a *$<$-chain.*

Corollary 4.18. *Let S be a finite nonempty set. If $\leqslant$ is a transitive relation on S then* $\operatorname{mcl}(\leqslant)$ *is well-defined and satisfies* $\operatorname{mcl}(\leqslant) \leqslant |S| - 1$.

Proof. This follows from the previous lemma. □

The maximum chain length of an order (a reflexive, transitive and antisymmetric relation) on a finite nonempty set is often called its *length* or its *height*; see, e.g., [37, Page 4] or [80, Section 2.1]. In other contexts the maximum chain length plus one of a preorder on a finite nonempty set S is called the *rank* of the finite preordered set $(S, \leqslant)$ (if S is empty then the rank is 0); see, e.g., [54]. We start with two simple observations.

Lemma 4.19. *If $\leqslant$ is a transitive relation on a finite, nonempty set S then its inverse, the relation $(\leqslant)^{-1}$ on S defined by*

$$s(\leqslant)^{-1}t :\iff t \leqslant s,$$

for $s, t \in S$, is a transitive relation on S as well, and $\operatorname{mcl}((\leqslant)^{-1}) = \operatorname{mcl}(\leqslant)$.

We omit the straightforward proof. Often, instead of $(\leqslant)^{-1}$ we write $\geqslant$.

Lemma 4.20. *If $\leqslant_1$ and $\leqslant_2$ are transitive relations on a finite, nonempty set S, then their intersection $\leqslant_3 := \leqslant_1 \cap \leqslant_2$, that is, the relation $\leqslant_3$ on S given by*

$$s \leqslant_3 t :\iff (s \leqslant_1 t \text{ and } s \leqslant_2 t),$$

for $s, t \in S$, is a transitive relation on S as well, and

$$\operatorname{mcl}(\leqslant_3) \leqslant \operatorname{mcl}(\leqslant_1) + \operatorname{mcl}(\leqslant_2).$$

Proof. It is clear that $\leqslant_3$ is a transitive relation on S. For the other assertion, we observe that

$$s <_3 t \iff ((s <_1 t \text{ and } s \leqslant_2 t) \text{ or } (s \leqslant_1 t \text{ and } s <_2 t)),$$

for all $s, t \in S$. Hence, if $s_0, \ldots, s_l$ is a $<_3$-chain then with

$$I_j := \{k \in \{0, \ldots, l-1\} \; : \; s_k <_j s_{k+1}\},$$

for $j = 1, 2$, we have $\{0, \ldots, l-1\} = I_1 \cup I_2$. The elements s_k for $k \in I_1 \cup \{\max(I_1) + 1\}$ form a $<_1$-chain and the elements s_k for $k \in I_2 \cup \{\max(I_2) + 1\}$ form a $<_2$-chain. We obtain $\operatorname{mcl}(\leqslant_3) \leqslant \operatorname{mcl}(\leqslant_1) + \operatorname{mcl}(\leqslant_2)$. □

If $\leqslant$ is a transitive relation on a set S then by

$$s \equiv t :\Longleftrightarrow (s = t \text{ or } (s \leqslant t \text{ and } t \leqslant s)),$$

for $s, t \in S$, an equivalence relation $\equiv$ on S is defined. If $\leqslant$ is reflexive a well, that is, if $\leqslant$ is a preorder then, for all $s, t \in S$,

$$s \equiv t \iff (s \leqslant t \text{ and } t \leqslant s).$$

Let us assume that $\leqslant$ is transitive. Remember that, for any $s \in S$, by

$$[s]_\equiv := \{t \in S \mid s \equiv t\}$$

we denote the $\equiv$-equivalence class of s, and, for any subset $A \subseteq S$, by

$$A_\equiv := \{[a]_\equiv \mid a \in A\}$$

we denote the set of $\equiv$-equivalence classes of elements of A.

Lemma 4.21. *Let $\leqslant$ be a transitive relation on a nonempty set S. If an equivalence class $q \in S_\equiv$ contains at least two different elements then $s \leqslant t$ is true for all $s, t \in q$.*

Proof. Let $q \in S_\equiv$ be an equivalence class containing at least two different elements. Let us consider some $s, t \in q$. If $s \neq t$ then $s, t \in q$ implies $s \leqslant t$. If $s = t$ then, due to the fact that there is at least one element $r \in q$ with $r \neq s$, we obtain $s \leqslant r$ and $r \leqslant s$ and, by transitivity of $\leqslant$, $s \leqslant s$ as well. $\square$

Note that in particular $s \leqslant s$ if s is an element of an equivalence class containing at least two elements. So, the restriction of a transitive relation to the union of all equivalence classes containing at least two elements is reflexive.
The following proposition is the key for our upper estimates for $\mathrm{mcl}(\leqslant_X)$ on $\mathcal{P}(\mathcal{T}_\varphi^X)$, for any $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$ and any bimodal formula φ.

Proposition 4.22. *Let $\leqslant$ be a transitive relation on a finite, nonempty set S. Then the relation $\leqslant'$ on $\mathcal{P}(S)$ defined by*

$$A \leqslant' B :\Longleftrightarrow (\forall b \in B)\,(\exists a \in A)\ a \leqslant b,$$

for $A, B \subseteq S$, is transitive as well, and $\mathrm{mcl}(\leqslant') \leqslant 2 \cdot |S_\equiv| \leqslant 2 \cdot |S|$.

Proof. It is straightforward to see that $\leqslant'$ is transitive. And it is clear that $|S_\equiv| \leqslant |S|$. Let us prove $\mathrm{mcl}(\leqslant') \leqslant 2 \cdot |S_\equiv|$. Let A be a subset of S. Let us call an element $a \in A$ a *minimal element of A* if there does not exist any $b \in A$ with $b < a$. Let $A_{\min}$ be the set of minimal elements of A. Let $A_{\min,\equiv} := (A_{\min})_{\equiv}$ be the set of $\equiv$-equivalence classes of elements of $A_{\min}$. Note that

$$(\forall a \in A)\,(\exists a' \in A_{\min})\ a' \leqslant a. \tag{4.1}$$

Indeed, let us consider some element $a \in A$. If $a_0 := a$ is not an element of $A_{\min}$ then there exists some $a_1 \in A$ with $a_1 < a_0$. If $a_1 \notin A_{\min}$ then there exists some $a_2 \in A$ with $a_2 < a_1$. And so on. As S is finite, by Corollary 4.18 this can be repeated only finitely often, and finally we arrive at some $a' \in A_{\min}$ with $a' \leqslant a$.
Now let also B be a subset of S. We claim:

$$\text{if } A_{\min,\equiv} = B_{\min,\equiv} \text{ then } (A \leqslant' B \text{ and } B \leqslant' A). \tag{4.2}$$

Indeed, let us assume $A_{\min,\equiv} = B_{\min,\equiv}$. Due to (4.1) applied to B instead of A, for any $b \in B$ there exists some $b' \in B_{\min}$ with $b' \leqslant b$. Due to $A_{\min,\equiv} = B_{\min,\equiv}$ there exists some $a \in A_{\min}$ with $a \equiv b'$. We obtain $a \equiv b' \leqslant b$, hence, $a \leqslant b$. This shows $A \leqslant' B$. By symmetry one obtains $B \leqslant' A$ as well.
Next, let also C be a subset of S and let us assume $A \leqslant' B$ and $B \leqslant' C$. We claim that in this case:

$$\text{if } q \in A_{\min,\equiv} \backslash B_{\min,\equiv} \text{ then } q \notin C_{\min,\equiv}. \tag{4.3}$$

Let us consider some $q \in A_{\min,\equiv} \backslash B_{\min,\equiv}$. For the sake of a contradiction, let us assume $q \in C_{\min,\equiv}$. Fix some $a \in q \cap A_{\min}$ and some $c \in q \cap C$. Due to $B \leqslant' C$, there exists some $b \in B$ with $b \leqslant c$. Due to (4.1) applied to B instead of A, there exists some $b' \in B_{\min}$ with $b' \leqslant b$. Due to $A \leqslant' B$, there exists some $a' \in A$ with $a' \leqslant b'$. We obtain

$$a' \leqslant b' \leqslant b \leqslant c \equiv a,$$

hence, $a' \leqslant a$. Due to $a \in A_{\min}$ we conclude $a \leqslant a'$, and this implies $a' \equiv b' \equiv b \equiv c \equiv a$. Hence $q = [b']_{\equiv} \in B_{\min,\equiv}$ in contradiction to the assumption. We have proved (4.3).
Finally, let us consider a sequence $(A^{(0)}, \ldots, A^{(l)})$ of subsets of S with $A^{(i)} <' A^{(i+1)}$ for all $i < l$. The claim (4.2) shows that for every $i < l$ the set $A^{(i+1)}_{\min,\equiv}$ is different from the set $A^{(i)}_{\min,\equiv}$. Hence, in each step from i to $i+1$ some class $q \in S_{\equiv}$ has to enter or to leave the set $A^{(\ldots)}_{\min,\equiv}$. The claim (4.3) shows that once a class $q \in S_{\equiv}$ has left the set $A^{(\ldots)}_{\min,\equiv}$ it can never re-enter it. Hence, any element $q \in S_{\equiv}$ can enter this set at most once and can leave it at most once. This shows that this set can change at most $2 \cdot |S_{\equiv}|$ times. This proves $l \leqslant 2 \cdot |S_{\equiv}|$. $\square$

Corollary 4.23. *Let $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$, and let φ be a bimodal formula.Then for the relation $\preccurlyeq_X$ on $\mathcal{P}(\mathcal{T}_\varphi^X)$ the following estimate is true.*

1. $\mathrm{mcl}(\preccurlyeq_X) \leqslant 4 \cdot |\mathcal{T}_\varphi^X|$, *if* $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}\}$.

2. $\mathrm{mcl}(\preccurlyeq_{\mathrm{SSL}}) \leqslant 2 \cdot |\mathcal{T}_\varphi^{\mathrm{SSL}}|$.

Proof. For $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}\}$ the relation $\preccurlyeq_X$ is equal to the intersection of the relations $\preccurlyeq'_X$ and $(\succcurlyeq'_X)^{-1}$ (where with $\succcurlyeq_X$ we mean the relation $(\preccurlyeq_X)^{-1}$). We obtain

$$\begin{aligned}
\mathrm{mcl}(\preccurlyeq_X) &\leqslant \mathrm{mcl}(\preccurlyeq'_X) + \mathrm{mcl}((\succcurlyeq'_X)^{-1}) && \text{(by Lemma 4.20)}\\
&= \mathrm{mcl}(\preccurlyeq'_X) + \mathrm{mcl}(\succcurlyeq'_X) && \text{(by Lemma 4.19)}\\
&\leqslant 2 \cdot |\mathcal{T}_\varphi^X| + 2 \cdot |\mathcal{T}_\varphi^X| && \text{(by Prop. 4.22)}\\
&= 4 \cdot |\mathcal{T}_\varphi^X|.
\end{aligned}$$

The relation $\preccurlyeq_{\mathrm{SSL}}$ is equal to the relation $\preccurlyeq'_{\mathrm{SSL}}$. Similarly as above we obtain $\mathrm{mcl}(\preccurlyeq_{\mathrm{SSL}}) \leqslant 2 \cdot |\mathcal{T}_\varphi^{\mathrm{SSL}}|$. □

4.4.2 The Recursion Depth of the Algorithm

Let $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$. The following proposition contains our estimate for the recursion depth that can occur when $ALG_X(\varphi)$ calls the recursive procedure alg_X.

Proposition 4.24. *Let $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$. Let φ be a bimodal formula. Let n be its length. If $(\mathcal{F}_0, \ldots, \mathcal{F}_l)$ for some $l \geqslant 0$ is a sequence of X-tableau-clouds with respect to φ such that during the execution of $ALG_X(\varphi)$ a call $alg_X(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_l)$ occurs then $l < 5 \cdot n \cdot |\mathcal{T}_\varphi^X|^2$.*

Proof. Let $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$. Let us assume that during the execution of $ALG_X(\varphi)$ a call $alg_X(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_l)$ occurs. Then, during the execution of $ALG_X(\varphi)$, for all $m \leqslant l$ a call $alg_X(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m)$ must occur. For all $m < l$ there must exist a pair $(\Box\chi_m, F_m) \in \mathrm{sf}(\varphi) \times \mathcal{F}_m$ with $\Box\chi_m \notin F_m$ which during the execution of $alg_X(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_m)$ leads to a call of $alg_X(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_{m+1})$, hence, such that, on the one hand,

- (I) is not satisfied, that is, there does not exist an $i \in \{0, \ldots, m\}$ with $\mathcal{F}_m \preccurlyeq_X \mathcal{F}_i$ and such that there exists some $G \in \mathcal{F}_i$ with $F_m \preccurlyeq_X G$ and $\chi_m \notin G$,

and on the other hand,

- at least the first part of (II) is satisfied, that is, $\mathcal{F}_{m+1} \in \mathfrak{C}_{\varphi}^{X} \setminus \{\mathcal{F}_0, \dots, \mathcal{F}_m\}$ and $\mathcal{F}_m \leqslant_X \mathcal{F}_{m+1}$ and there exists some $G \in \mathcal{F}_{m+1}$ with $F_m \preccurlyeq_X G$ and $\chi_m \notin G$.

It is clear that for all $m < l$ we have $\mathcal{F}_m \leqslant_X \mathcal{F}_{m+1}$. Let $m_1, \dots, m_{k-1}$ be in increasing order the elements of the set

$$\{j \in \{0, \dots, l-1\} \mid \mathcal{F}_j <_X \mathcal{F}_{j+1}\},$$

(this set can be empty), and set $m_0 := -1$ and $m_k := l$. Then, for each $i \in \{0, \dots, k-1\}$, all tableau-clouds $\mathcal{F}_m$ for $m \in \{m_i + 1, \dots, m_{i+1}\}$ are pairwise $\equiv_X$-equivalent:

$$\dots \equiv_X \mathcal{F}_{m_i} <_X \mathcal{F}_{m_i+1} \equiv_X \mathcal{F}_{m_i+2} \equiv_X \dots \equiv_X \mathcal{F}_{m_{i+1}} <_X \mathcal{F}_{m_{i+1}+1} \equiv_X \dots$$

Furthermore,

$$\mathcal{F}_{m_1} <_X < \mathcal{F}_{m_2} <_X \dots <_X \mathcal{F}_{m_{k-1}} < \mathcal{F}_{m_k}.$$

Hence, $k-1 \leqslant \mathrm{mcl}(\leqslant_X)$. For a moment, let us fix some $i \in \{0, \dots, k-1\}$. Can there be two different numbers $m, \widetilde{m} \in \{m_i + 1, \dots, m_{i+1}\}$, say with $m < \widetilde{m}$, such that $(\Box\chi_m, F_m) = (\Box\chi_{\widetilde{m}}, F_{\widetilde{m}})$? We claim that this cannot be the case. Otherwise, as at least the first part of (II) is satisfied for m, there is some $G \in \mathcal{F}_{m+1}$ with $F_m \preccurlyeq_X G$ and $\chi_m \in G$, hence, with $F_{\widetilde{m}} \preccurlyeq_X G$ and $\chi_{\widetilde{m}} \in G$. Furthermore, as all of the X-tableau-sets $\mathcal{F}_0, \dots, \mathcal{F}_l$ are pairwise different (this is due to the assumption that during the execution of $ALG_X(\varphi)$ a call $alg_X(\varphi, \mathcal{F}_0, \dots, \mathcal{F}_l)$ occurs) the set $\{\mathcal{F}_{m_i+1}, \dots, \mathcal{F}_{m_{i+1}}\}$ contains at least two different elements (because the assumption $m, \widetilde{m} \in \{m_i + 1, \dots, m_{i+1}\}$ with $m < \widetilde{m}$, implies that the set $\{m_i+1, \dots, m_{i+1}\}$ contains at least two numbers), and by Lemma 4.21 this implies $\mathcal{F}_{m+1} \leqslant_X \mathcal{F}_{\widetilde{m}}$ (note that $\mathcal{F}_{m+1} \leqslant_X \mathcal{F}_{\widetilde{m}}$ is clear if $m + 1 < \widetilde{m}$ and also if $m + 1 = \widetilde{m}$ and $X \in \{\mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$; Lemma 4.21 is needed only for the case $m + 1 = \widetilde{m}$ and $X = \mathrm{K4} \times \mathrm{S5}$). But these facts together would contradict the fact that (I) is not satisfied for $\widetilde{m}$. We conclude that for pairwise different numbers $m, \widetilde{m} \in \{m_i + 1, \dots, m_{i+1}\}$ we have $(\Box\chi_m, F_m) \neq (\Box\chi_{\widetilde{m}}, F_{\widetilde{m}})$. This implies

$$m_{i+1} - m_i \leqslant |\mathrm{sf}_{\Box}(\varphi) \times \mathcal{T}_{\varphi}^{X}| \leqslant (n-1) \cdot |\mathcal{T}_{\varphi}^{X}|.$$

As this is true for all $i \in \{0, \dots, k-1\}$, we obtain, using Corollary 4.23, in all three cases for $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$,

$$\begin{array}{rcl} l & = & m_k \\ & = & -1 + \sum_{i=0}^{k-1}(m_{i+1} - m_i) \\ & \leqslant & -1 + k \cdot (n-1) \cdot |\mathcal{T}_\varphi^X| \\ & \leqslant & -1 + (\mathrm{mcl}(\leqslant_X) + 1) \cdot (n-1) \cdot |\mathcal{T}_\varphi^X| \\ & \leqslant & -1 + (4 \cdot |\mathcal{T}_\varphi^X| + 1) \cdot (n-1) \cdot |\mathcal{T}_\varphi^X| \\ & < & 5 \cdot n \cdot |\mathcal{T}_\varphi^X|^2. \end{array}$$

□

4.4.3 Proof of the Upper Bound

We are now prepared for the proof of the statement formulated at the beginning.

Proof of Proposition 4.16. Let $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$. Before we can analyze the space used by the algorithms $ALG_X(\varphi)$ and $alg_X(\varphi, \mathcal{F}_0, \ldots, \mathcal{F}_l)$, we have to explain how the formulas, the tableau-sets and the tableau-clouds with which these algorithms deal are stored in a Turing machine.
So, let φ be a bimodal formula. Let n be its length (as a string over the alphabet $\{(,), \neg, \wedge, \Box, K, X, 0, 1\}$; compare Definition 3.4, but see also Remark 4.25).
Let $a := |\mathrm{sf}(\varphi)|$ be the number of subformulas of φ. Then $a \leqslant n$. Let $\psi_1, \ldots, \psi_a$ be the subformulas of φ in some order. We can identify any subset $T \subseteq \mathrm{sf}(\varphi) = \{\psi_1, \ldots, \psi_a\}$, in particular any X-tableau-set, with a binary string $s_1 \ldots s_a \in \{0,1\}^a$ by defining

$$s_i = 1 :\Longleftrightarrow \psi_i \in T.$$

Let $A := |\mathcal{T}_\varphi^X|$ be the number of all X-tableau-sets with respect to φ. Then $A \leqslant 2^a \leqslant 2^n$. In Section 4.5 we shall give a better upper estimate of A in the cases $X \in \{\mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$. As a preliminary step at the beginning of $ALG_X(\varphi)$ we can check for all binary strings $s_1 \ldots s_a \in \{0,1\}^a$ in alphabetical order whether they describe subsets of $\mathrm{sf}(\varphi)$ that are X-tableau-sets and write down only those. Then we obtain a list of A binary strings of length a. This can be considered as an alphabetical list of all X-tableau-sets with respect to φ. We will keep this list stored on a working tape of the Turing machine during the whole computation. Note that all this can be done in space $O(a \cdot A)$.

Now any set $\mathcal{F}$ whose elements are X-tableau-sets with respect to φ (so, in particular any X-tableau-cloud with respect to φ) can be described in a similar manner by a binary string $b_1 \dots b_A$ of length A where

$$b_i = 1 :\iff \text{the } i\text{-th } X\text{-tableau-set with respect to } \varphi \text{ is an element of } \mathcal{F}.$$

In the algorithm we will assume that any X-tableau-cloud is described by such a binary string of length A.
Note that, given a binary string of length A, it is straightforward to check whether the set of X-tableau-sets with respect to φ described by this string is an X-tableau-cloud with respect to φ or not, and this can also be done within space $O(a \cdot A)$.
Let us consider the for-loop in the algorithm $ALG_X(\varphi)$ as defined in Definition 4.14:

> the algorithm $ALG_X(\varphi)$ lets $\mathcal{F}_0$ run through all X-tableau-clouds $\mathcal{F}_0 \in \mathfrak{C}_\varphi^X$ such that there exists some $F \in \mathcal{F}_0$ with $\varphi \in F$ and applies alg_X to $(\varphi, \mathcal{F}_0)$.

In a detailed implementation of this for-loop ("through all X-tableau-clouds $\mathcal{F}_0 \in \mathfrak{C}_\varphi^X$ such that there exists some $F \in \mathcal{F}_0$ with $\varphi \in F$") one can run through all binary strings of length A and discard all those that do not describe an X-tableau-cloud with respect to φ and all those that do not contain an X-tableau-set F with $\varphi \in F$. It is clear that the conditions that need to be checked here can be checked in space $O(a \cdot A)$.
We come to the recursive calls $alg_X(\varphi, \mathcal{F}_0, \dots, \mathcal{F}_m)$ of the algorithm alg_X that may occur during the execution of $ALG_X(\varphi)$. First, remember that according to Proposition 4.24 we have $m < 5 \cdot n \cdot A^2$. We claim that with each new recursive call of $alg_X(\varphi, \mathcal{F}_0, \dots, \mathcal{F}_m)$ at most an additional number of $O(n + A)$ bits need to be stored.
Indeed, one has to go through all pairs $(\Box\chi, F) \in \mathrm{sf}(\varphi) \times \mathcal{F}_m$ with $\Box\chi \notin F$. These pairs can be stored using $O(\log a + a) \subseteq O(n)$ bits. Then one checks condition (I). The number $i \in \{0, \dots, m\}$ considered in (I) can be stored in $O(\log(m)) = O(n)$ bits. And the set G considered in (I) can be stored in $a \leqslant n$ bits as well. When checking whether (II) is true or not one has to look for a certain tableau-cloud $\mathcal{F}_{m+1}$. Again, this can be stored using not more than A bits. And the set G considered there can be stored in $O(n)$ space again. Thus, one does indeed not need to use more than $O(n+A)$ space with each new recursive call of alg_X.
We have seen that some preliminary steps and the initial for-loop in the algorithm $ALG_X(\varphi)$ can be done in space $O(a \cdot A)$. According to Proposition 4.24 the recursion depth m in the recursive calls of $alg_X(\varphi, \mathcal{F}_0, \dots, \mathcal{F}_m)$

occuring during the computation of $ALG_X(\varphi)$ is at most $5 \cdot n \cdot A^2$. Finally, each recursive call requires at most an additional space of $O(n + A)$. We conclude that $ALG_X(\varphi)$ can be implemented in such a way that the space used is of the order $O(n \cdot (n + A)^3)$. $\square$

Remark 4.25. All arguments in Section 4.4 and Section 4.5 go through as well if with n one does not denote the length of the bimodal formula φ as a string over the alphabet $\{(,), \neg, \wedge, \square, K, X, 0, 1\}$ but instead the "simplified" length of φ as a string over the infinite alphabet $\{(,), \neg, \wedge, \square, K\} \cup AT$. This can also be defined as the number of symbols different from $0, 1$ in φ (again as a string over the alphabet $\{(,), \neg, \wedge, \square, K, X, 0, 1\}$).

4.5 On the Number of Tableau-sets

In the previous section we have shown that our algorithms for the satisfiability problems of the bimodal logics K4 × S5, S4 × S5, and SSL can be implemented using not more than $O(n \cdot (n + |\mathcal{T}_\varphi^X|)^3)$ space where φ is the given bimodal formula, where n is its length, and where $\mathcal{T}_\varphi^X$ for $X \in \{\mathrm{K4} \times \mathrm{S5}, \mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$ is the set of X-tableau-sets with respect to φ. As there are at most n subformulas of φ we obtain $|\mathcal{T}_\varphi^X| \leqslant 2^n$. Thus, we have shown that the algorithms can be implemented in space $O(n \cdot 2^{3 \cdot n})$. Hence, the satisfiability problems of the bimodal logics K4 × S5, S4 × S5, and SSL are in ESPACE.
In this section we wish to slightly improve this result in the cases $X \in \{\mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$ by giving a slightly better upper bound for $|\mathcal{T}_\varphi^X|$. By making use of the conditions that an X-tableau-set has to satisfy according to Definition 4.2.2 we are going to show that, for all bimodal formulas of length $n \geqslant 3$,

$$|\mathcal{T}_\varphi^X| \leqslant 2^{\frac{2}{3}n}.$$

In fact, we are going to show the following result. Let $X \in \{\mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$. For a bimodal formula φ let $\ell(\varphi)$ be its "simplified length" as considered in Remark 4.25, that is, $\ell(\varphi)$ is the number of symbols different from $0, 1$ in φ (as a string over the alphabet $\{(,), \neg, \wedge, \square, K, X, 0, 1\}$). For $n \geqslant 1$ let

$$T(n) \quad := \quad \max\{|\mathcal{T}_\varphi^X| \ : \ \varphi \text{ is a bimodal formula with } \ell(\varphi) \leqslant n\}.$$

Proposition 4.26.

$$\begin{array}{rll} & T(1) & = 2, \\ & T(2) & = 3, \\ \textit{for } n \geqslant 3, & T(n) & < 2^{(2 \cdot n/3)}. \end{array}$$

Actually, Proposition 4.26 can certainly still be improved by showing an even smaller upper bound for $T(n)$. One can apply similar considerations in the case $X = \mathrm{K4} \times \mathrm{S5}$. But in order to gain something in that case one should use a slightly different definition of $\mathrm{K4} \times \mathrm{S5}$-tableau-sets, and even then the gain in considerably smaller than in the cases $X \in \{\mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$. Therefore, we refrain from treating the case $X = \mathrm{K4} \times \mathrm{S5}$ here.

Proof of Proposition 4.26. In the whole proof let $X \in \{\mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$. As the $\mathrm{S4} \times \mathrm{S5}$-tableau-sets are exactly the SSL-tableau-sets, that is, as $\mathcal{T}_\varphi^{\mathrm{S4}\times\mathrm{S5}} = \mathcal{T}_\varphi^{\mathrm{SSL}}$ for any bimodal formula φ, in the proof we will always suppress X and, for example, simply speak about *tableau-sets* instead of X-tableau-sets and simply write $\mathcal{T}_\varphi$ instead of $\mathcal{T}_\varphi^X$.
In addition to $T(n)$, for $n \geqslant 5$ we define

$$T_\wedge(n) \;:=\; \max\{|\mathcal{T}_\varphi| \;:\; \varphi \text{ is a bimodal formula with } \ell(\varphi) \leqslant n \text{ and there exist bimodal formulas } \chi \text{ and } \psi \text{ with } \varphi = (\chi \wedge \psi)\}.$$

Note that any bimodal formula φ of the form $(\chi \wedge \psi)$ for bimodal formulas χ, ψ satisfies $\ell(\varphi) \geqslant 5$. In addition to the assertions in the proposition we claim

$$\text{for } n \geqslant 5,\; T_\wedge(n) < 2^{(2\cdot n/3)-1}.$$

This is needed for the proof of the assertions in the proposition. We are going to show all of these assertions by induction over n.
If φ is a bimodal formula with $\ell(\varphi) = 1$ then $\varphi = A \in AT$. There are exactly two tableau-sets with respect to φ: the empty set and the set $\{A\}$. This proves the assertion for $n = 1$.
Let φ be a bimodal formula with $\ell(\varphi) = 2$. There are three cases.

1. $\varphi = \neg A$ where $A \in AT$. Then there are exactly two tableau-sets with respect to φ: the set $\{A\}$ and the set $\{\neg A\}$.

2. $\varphi = \Box A$ where $A \in AT$. Then there are exactly three tableau-sets with respect to φ: the empty set, the set $\{A\}$, and the set $\{A, \Box A\}$.

3. $\varphi = KA$ where $A \in AT$. Then there are exactly three tableau-sets with respect to φ: the empty set, the set $\{A\}$, and the set $\{A, KA\}$.

This proves the assertion for $n = 2$. In the second case we made use of the fact that if for some bimodal formula χ the formula $\Box\chi$ is an element of a tableau-set then χ is an element of that tableau-set as well. Similarly, in the third case we made use of the fact that if for some bimodal formula χ the formula $K\chi$ is an element of a tableau-set then χ is an element of that

tableau-set as well. We will make use of these facts in the following cases as well.
Let us consider now a bimodal formula φ with $n = \ell(\varphi) \geqslant 3$. We distinguish several cases.

- $\varphi = \neg\chi$ for some formula χ.

  Then for any tableau-set $T \in \mathcal{T}_\varphi$ with respect to φ the set $T \cap \mathrm{sf}(\chi)$ is a tableau-set with respect to χ. And whether the formula $\neg\chi$ is an element of a given tableau-set $T \in \mathcal{T}_\varphi$ is determined by the answer to the question whether χ is an element of $T \cap \mathrm{sf}(\chi)$. Hence, $|\mathcal{T}_\varphi| = |\mathcal{T}_\chi|$. If $\ell(\chi) = 2$ then we get $|\mathcal{T}_\varphi| = |\mathcal{T}_\chi| \leqslant 3 < 4 = 2^{2\cdot 3/3}$. If $\ell(\chi) \geqslant 3$ then by induction we get $|\mathcal{T}_\varphi| = |\mathcal{T}_\chi| < 2^{2\cdot(n-1)/3} < 2^{2\cdot n/3}$.

- $\varphi = \circ\neg\chi$ for some formula χ and $\circ \in \{\Box, K\}$.

  If $\ell(\varphi) = 3$ then $\chi = A$ for some $A \in AT$. In that case there are exactly three tableau-sets with respect to φ: the set $\{A\}$, the set $\{\neg A\}$, and the set $\{\neg A, \circ\neg A\}$. Note that $3 < 4 = 2^{2\cdot 3/3}$.

  If $\ell(\varphi) \geqslant 4$ then we claim that $|\mathcal{T}_\varphi| \leqslant 2 \cdot |\mathcal{T}_\chi|$. Indeed, if T is a tableau set with respect to φ then $T \cap \mathrm{sf}(\chi)$ is a a tableau set with respect to χ. The only elements in $\mathrm{sf}(\varphi)\backslash\mathrm{sf}(\chi)$ are the two formulas $\neg\chi$ and $\circ\neg\chi$. The question whether $\neg\chi$ is an element of T or not is determined already by $T\cap\mathrm{sf}(\chi)$. We have shown $|\mathcal{T}_\varphi| \leqslant 2\cdot|\mathcal{T}_\chi|$. In the case $\ell(\varphi) = 4$ we obtain $\ell(\chi) = 2$, hence, $|\mathcal{T}_\varphi| \leqslant 2 \cdot |\mathcal{T}_\chi| \leqslant 2 \cdot 3 = 6 < 2^{2\cdot 4/3}$. In the case $\ell(\varphi) \geqslant 5$ we obtain $\ell(\chi) = \ell(\varphi) - 2 \geqslant 3$, hence, by induction hypothesis, $|\mathcal{T}_\varphi| \leqslant 2 \cdot |\mathcal{T}_\chi| < 2 \cdot 2^{2\cdot(n-2)/3} < 2^{2\cdot n/3}$.

- $\varphi = \circ_1 \circ_2 \neg\chi$ for some formula χ and $\circ_1, \circ_2 \in \{\Box, K\}$.

  We claim that $|\mathcal{T}_\varphi| \leqslant 3 \cdot |\mathcal{T}_\chi|$. Indeed, if T is a tableau set with respect to φ then $T \cap \mathrm{sf}(\chi)$ is a a tableau set with respect to χ. The only elements in $\mathrm{sf}(\varphi)\backslash\mathrm{sf}(\chi)$ are the three formulas $\neg\chi$, $\circ_2\neg\chi$, and $\circ_1 \circ_2 \neg\chi$. The question whether $\neg\chi$ is an element of T or not is determined already by $T \cap \mathrm{sf}(\chi)$. And for the two formulas $\circ_2\neg\chi$ and $\circ_1 \circ_2 \neg\chi$ we observe that if $\circ_1 \circ_2 \neg\chi$ is an element of T then so is $\circ_2\neg\chi$. We have shown $|\mathcal{T}_\varphi| \leqslant 3 \cdot |\mathcal{T}_\chi|$.

  It is clear that $\ell(\varphi) = \ell(\circ_1 \circ_2 \neg\chi) \geqslant 4$. In the case $\ell(\varphi) = 4$ we obtain $\ell(\chi) = 1$, hence, $|\mathcal{T}_\varphi| \leqslant 3\cdot|\mathcal{T}_\chi| \leqslant 3\cdot 2 = 6 < 2^{2\cdot 4/3}$. In the case $\ell(\varphi) = 5$ we obtain $\ell(\chi) = 2$, hence, $|\mathcal{T}_\varphi| \leqslant 3 \cdot |\mathcal{T}_\chi| \leqslant 3 \cdot 3 = 9 < 2^{2\cdot 5/3}$. In the case $\ell(\varphi) \geqslant 6$ we obtain $\ell(\chi) = \ell(\varphi) - 3 \geqslant 3$, hence, by induction hypothesis, $|\mathcal{T}_\varphi| \leqslant 3 \cdot |\mathcal{T}_\chi| < 3 \cdot 2^{2\cdot(n-3)/3} < 2^{2\cdot n/3}$.

- $\varphi = \circ_1 \circ_2 \circ_3 \chi$ for some formula χ and $\circ_1, \circ_2, \circ_3 \in \{\Box, K\}$.

  Again, we will use the already mentioned fact for any subformula $\circ_i \chi$ of φ: if $\circ_i \chi$ is an element of a tableau set with respect to φ then χ is an element of the same tableau set.

  First, let us consider the cases $\ell(\varphi) = 4$ and $\ell(\varphi) = 5$. If $\ell(\varphi) = 4$ then $\chi = A$ for some $A \in AT$, and one checks that there are exactly five tableau sets with respect to φ: the sets $\varnothing$, $\{A\}$, $\{A, \circ_3 A\}$, $\{A, \circ_3 A, \circ_2 \circ_3 A\}$, $\{A, \circ_3 A, \circ_2 \circ_3 A, \circ_1 \circ_2 \circ_3 A\}$. Note that $5 < 2^{2 \cdot 4/3}$. Next, let us consider the case $\ell(\varphi) = 5$. Then there exists some $A \in AT$ such that either $\chi = \neg A$ or $\chi = \circ_4 A$ for some $\circ_4 \in \{\Box, K\}$. One checks that in the first case there are again exactly five tableau sets with respect to φ and in the second case there are exactly six tableau sets with respect to φ. Note that $6 < 2^{2 \cdot 5/3}$.

  For the case $\ell(\varphi) \geqslant 6$ we claim that $|\mathcal{T}_\varphi| \leqslant 4 \cdot |\mathcal{T}_\chi|$. Indeed, if T is a tableau set with respect to φ then $T \cap \mathrm{sf}(\chi)$ is a a tableau set with respect to χ. And for the three formulas $\circ_3 \chi$ and $\circ_2 \circ_3 \chi$ and $\circ_1 \circ_2 \circ_3 \chi$ there are only four possibilities: (1) none of them is an element of T, (2) only $\circ_3 \chi$ is an element of T (3) only $\circ_2 \chi$ and $\circ_2 \circ_3 \chi$ are elements of T, (4) all three of them are elements of T. We have shown $|\mathcal{T}_\varphi| \leqslant 4 \cdot |\mathcal{T}_\chi|$. In the case $\ell(\varphi) \geqslant 6$ we obtain $\ell(\chi) = \ell(\varphi) - 3 \geqslant 3$, hence, $|\mathcal{T}_\varphi| \leqslant 4 \cdot |\mathcal{T}_\chi| < 4 \cdot 2^{2 \cdot (n-3)/3} = 2^{2 \cdot n/3}$.

- $\varphi = \circ(\chi \wedge \psi)$ for some formulas χ, ψ and $\circ \in \{\Box, K\}$.

  Then $\ell(\varphi) \geqslant 6$ and $\ell((\chi \wedge \psi)) = \ell(\varphi) - 1 \geqslant 5$. We obtain

$$\begin{aligned} |\mathcal{T}_\varphi| &\leqslant 2 \cdot |\mathcal{T}_{(\chi \wedge \psi)}| \\ &< 2 \cdot 2^{(2 \cdot (n-1)/3) - 1} \qquad \text{(by induction hypothesis for } T_\wedge(n-1)) \\ &< 2^{2 \cdot n/3}. \end{aligned}$$

Finally, let us consider the case $\varphi = (\chi \wedge \psi)$, for some formulas χ, ψ. As before, let $n := \ell(\varphi)$. Note that $n = 3 + \ell(\chi) + \ell(\psi)$. It is sufficient to prove

$|\mathcal{T}_\varphi| < 2^{(2\cdot n/3)-1}$. We observe by induction hypothesis:

$$\begin{aligned}|\mathcal{T}_\varphi| &\leqslant |\mathcal{T}_\chi| \cdot |\mathcal{T}_\psi| \\ &\leqslant \begin{cases} 2 \cdot 2 = 4 < 2^{(2\cdot 5/3)-1} & \text{if } \ell(\chi) = 1 \text{ and } \ell(\psi) = 1, \\ 2 \cdot 3 = 6 < 2^{(2\cdot 6/3)-1} & \text{if } \ell(\chi) = 1 \text{ and } \ell(\psi) = 2, \\ 3 \cdot 2 = 6 < 2^{(2\cdot 6/3)-1} & \text{if } \ell(\chi) = 2 \text{ and } \ell(\psi) = 1, \\ 3 \cdot 3 = 9 < 2^{(2\cdot 7/3)-1} & \text{if } \ell(\chi) = 2 \text{ and } \ell(\psi) = 2, \\ 2 \cdot 2^{2\cdot \ell(\psi)/3} < 2^{(2\cdot n/3)-1} & \text{if } \ell(\chi) = 1 \text{ and } \ell(\psi) \geqslant 3, \\ 2^{2\cdot \ell(\chi)/3} \cdot 2 < 2^{(2\cdot n/3)-1} & \text{if } \ell(\chi) \geqslant 3 \text{ and } \ell(\psi) = 1, \\ 3 \cdot 2^{2\cdot \ell(\psi)/3} < 2^{(2\cdot n/3)-1} & \text{if } \ell(\chi) = 2 \text{ and } \ell(\psi) \geqslant 3, \\ 2^{2\cdot \ell(\chi)/3} \cdot 3 < 2^{(2\cdot n/3)-1} & \text{if } \ell(\chi) \geqslant 3 \text{ and } \ell(\psi) = 2, \\ 2^{2\cdot \ell(\chi)/3} \cdot 2^{2\cdot \ell(\psi)/3} < 2^{(2\cdot n/3)-1} & \text{if } \ell(\chi) \geqslant 3 \text{ and } \ell(\psi) \geqslant 3. \end{cases}\end{aligned}$$

□

Corollary 4.27. *Let $X \in \{\mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$. The algorithm ALG_X can be implemented on a multi-tape Turing machine so that it, given a bimodal formula φ of length n, does not use more than $O(n \cdot 2^{2\cdot n})$ space.*

Proof. This follows immediately from Propositions 4.16 and 4.26. □

Proof of Theorem 4.1 in the cases $X \in \{\mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$. Let us assume that $X \in \{\mathrm{S4} \times \mathrm{S5}, \mathrm{SSL}\}$. We have presented an algorithm ALG_X that, according to Proposition 4.15, accepts a bimodal formula φ if, and only if, φ is X-satisfiable. And according to Corollary 4.27 the algorithm ALG_X can be implemented in such a way that it works in space $O(n \cdot 2^{2\cdot n})$ where n is the length of the input formula φ. □

Chapter 5

Preparations for the Reduction of Alternating Turing Machines to SSL and to S4 × S5

We wish to show that the satisfiability problems of the bimodal logics SSL, K4 × S5, and S4 × S5 are EXPSPACE-hard under logarithmic space reduction. In Chapter 8 we show that the satisfiability problem of SSL can be reduced in logarithmic space to the satisfiability problem of S4 × S5, and in Chapter 9 we show that the satisfiability problem of S4 × S5 can be reduced in logarithmic space to the satisfiability problem of K4 × S5. Thus, EXPSPACE-hardness is shown for the satisfiability problems of all of these three logics once we have shown that the satisfiability problem of SSL is EXPSPACE-hard. In order to show this, we shall use Alternating Turing Machines [18]. In this respect, we follow the example of Lange and Lutz [64] who used Alternating Turing Machines in order to establish a sharp lower bound for the complexity of a certain dynamic logic. As any language in EXPSPACE is accepted by an Alternating Turing Machine (ATM) working in exponential time, it is sufficient to show that any language recognized by an Alternating Turing Machine working in exponential time can be reduced in logarithmic space to the satisfiability problem of SSL. We will present such a reduction in Chapter 6. In Chapter 7 we present a similar reduction of Alternating Turing Machines to the satisfiability problem of S4 × S5. For this purpose we will construct an SSL formula as well as an S4 × S5 formula both of which describe the computation of an exponential time bounded Alternating Turing Machine.
A string w is an element of the language $L(M)$ recognized by an ATM M iff there exists a so-called accepting tree of M on input w. In such a tree each node represents a configuration of M. Our idea is to construct a formula

depending on w such that the models of this formula have the tree structure of an accepting tree of M on input w, where now the nodes of the tree are clouds such that the formulas satisfied in some cloud describe a configuration of M. In this way the induced relation ${\xrightarrow{\Diamond}}^{\xrightarrow{L}}$ between the clouds serves as a simple temporal operator.
In the following section we describe how information is stored and transmitted in a model of such a formula. In particular we introduce certain formulas that we call *shared variables* that can transmit information in the $\xrightarrow{L}$-direction and by which we can overcome the problem in the logic SSL that all propositional variables are persistent. As a first application of this, in Section 5.2 we demonstrate how one can implement a binary counter both in the logic S4 $\times$ S5 and the logic SSL. In Section 5.3 we recall the definition of Alternating Turing Machines. In Chapters 6 and 7 we will come to the reductions that prove EXPSPACE-hardness of SSL and of S4 $\times$ S5.

5.1 Shared Variables

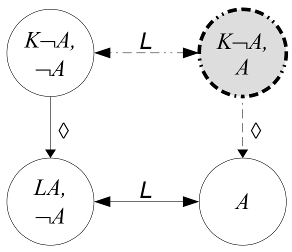
We have to make sure that various kinds of information are stored in a suitable way in any model of the fomula. We also need to copy and transmit various bits of information, both in the $\xrightarrow{\Diamond}$-direction (we always depict this as the vertical direction) as well as in the $\xrightarrow{L}$-direction (we always depict this as the horizontal direction). This will be done by two kinds of formulas.

- On the one hand, we need formulas that have the same truth value in the vertical ($\xrightarrow{\Diamond}$) direction but can change their truth values in the horizontal ($\xrightarrow{L}$) direction. In the case of the logic SSL, for this purpose we can simply use propositional variables as they are persistent anyway. And in the logic S4 $\times$ S5 we can force certain propositional variables to be persistent by a suitable formula.

- On the other hand, we need formulas that have the same truth value in the horizontal ($\xrightarrow{L}$) direction but can change their truth values in the vertical ($\xrightarrow{\Diamond}$) direction. Such formulas will be called *shared variables* and will be defined now.

In the case of the logic S4 $\times$ S5, for this purpose we can simply use a formula α of the form
$$\alpha = LA$$
where A is a propositional variable. Then α has the same truth value at all points of a cloud and we can compare the value of a vector of usual

propositional variables with the value of a corresponding vector of shared variables. In this way we can for instance select a point in a cloud where these values match.
But for the logic SSL this approach does not work. Because of the persistence of propositional variables in SSL, the construction of shared variables in SSL is more complicated than in S4 × S5. Due to the left commutativity, the formula $\neg LA = K\neg A$ is also persistent. The following picture illustrates why an attempt to use $K\neg A$ in a non-persistent way must fail.



A suitable construction for shared variables α and $\neg\alpha$ in SSL must be more complicated and can be realized as defined below.

Definition 5.1 (Shared Variables)**.** For $i \in \mathbb{N}$ let A_i be special propositional variables. Then

1. In S4 × S5 the *shared variables* α_i are defined as follows:

$$\alpha_i := LA_i.$$

2. In SSL we fix another special propositional variable B, different from all A_i, and define the *shared variables* α_i as follows:

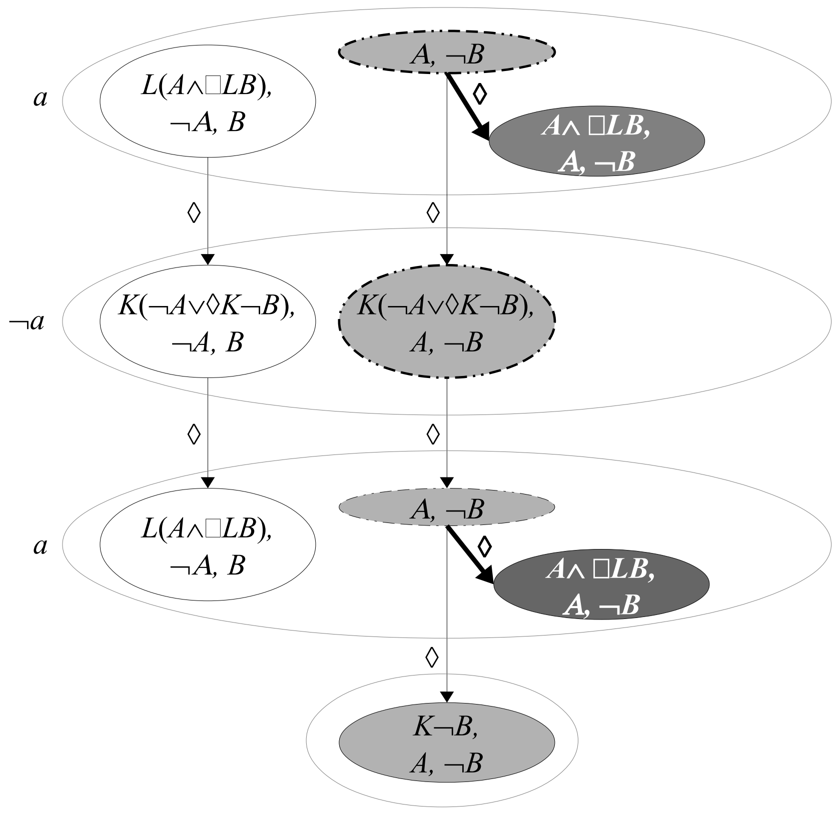
$$\alpha_i := L(A_i \wedge \Box LB).$$

Note that

$$\begin{array}{rl} \text{in S4} \times \text{S5} & \neg\alpha_i \equiv K\neg A_i \\ \text{and in SSL} & \neg\alpha_i \equiv K(\neg A_i \vee \Diamond K\neg B). \end{array}$$

See Figure 5.1 for a model of a single shared variable α_i (in the figure we have omitted the index i) in SSL changing its value from 1 to 0 and back from 0 to 1. In this model the information is stored at the white points which we call *information points.* The gray points are *auxiliary points* that ensure that we obtain a model for the shared variables. Note that the information points differ from the auxiliary points in the value of the propositional variable B,

Figure 5.1: Cross axiom model of $\alpha \wedge \Diamond(\neg\alpha \wedge \Diamond\alpha)$.



which is true at all information points, independent of the value of α stored there, and false at all auxiliary points. Thus, the value of the propositional variable B allows us to distinguish between the information points and the auxiliary points.
Although the shared variables α_i are formulas we are going to use them as if they were variables. The propositional variables A_i (and, in the case of SSL, the propositional variable B) that are used in their definition will not be used in any other way. As a first example of the application of shared variables, in the following section we demonstrate how, using shared variables, one can implement n-bit binary counters, first in $\mathrm{S4} \times \mathrm{S5}$ and then in SSL. Binary counters are going to play a key role in the simulation of Alternating Turing Machines in $\mathrm{S4} \times \mathrm{S5}$ as well as in SSL.

5.2 Binary Counters in S4 × S5 and in SSL

Fix some natural number $n \geqslant 1$. We wish to implement both in $\mathrm{S4} \times \mathrm{S5}$ and in SSL a binary n-bit counter that counts from 0 to $2^n - 1$. That means, for each of the two bimodal logics $\mathrm{S4} \times \mathrm{S5}$ and SSL, we wish to construct a satisfiable formula with the property that any model of it contains a sequence of pairwise distinct points $p_0, \dots, p_{2^n-1}$ such that, for each $i \in \{0, \dots, 2^n - 1\}$, at the point p_i the number i is stored in binary form in a certain way. To describe the implementation of the counter we first introduce some notation.

- For a natural number i, we define the finite set $\mathrm{Ones}(i) \subseteq \mathbb{N}$ by

$$\sum_{k \in \mathrm{Ones}(i)} 2^k = i,$$

  that is, $\mathrm{Ones}(i)$ is the set of the positions of ones in the binary representation of i (where the positions are counted from the right starting with 0).

- We will also need the bits $b_k(i) \in \{0, 1\}$ of the binary representation of i, for $i, k \in \mathbb{N}$. They are defined by

$$b_k(i) := \begin{cases} 1 & \text{if } k \in \mathrm{Ones}(i), \\ 0 & \text{if } k \notin \mathrm{Ones}(i). \end{cases}$$

- For natural numbers i, n with $n > 0$ and $i \leqslant 2^n - 1$ the *binary representation of length n of i* is the string

$$\mathrm{bin}_n(i) := b_{n-1}(i), \dots, b_0(i).$$

For	the following expression	is an abbreviation of the following formula
$l \geqslant 1, k \geqslant -1$	persistent$(\underline{F}, > k)$	$\bigwedge_{h=k+1}^{l-1} K(\Box F_h \vee \Box\neg F_h)$
$l \geqslant 1$	persistent$(\underline{F})$	persistent$(\underline{F}, > -1)$
$l \geqslant 1, k \geqslant -1$	$(\underline{F} = \underline{G}, > k)$	$\bigwedge_{h=k+1}^{l-1}(F_h \leftrightarrow G_h)$
$l \geqslant 1$	$(\underline{F} = \underline{G})$	$(\underline{F} = \underline{G}, > -1)$
$l \geqslant 1,\ 0 \leqslant i < 2^l$	$(\underline{F} = \mathrm{bin}_l(i))$	$\bigwedge_{k\in \mathrm{Ones}(i)} F_k \wedge$ $\bigwedge_{k\in\{0,\dots,l-1\}\setminus \mathrm{Ones}(i)} \neg F_k$
$l \geqslant 1,\ 0 \leqslant k < l$	rightmost_zero$(\underline{F}, k)$	$\neg F_k \wedge \bigwedge_{h=0}^{k-1} F_h$
$l \geqslant 1,\ 0 \leqslant k < l$	rightmost_one$(\underline{F}, k)$	$F_k \wedge \bigwedge_{h=0}^{k-1} \neg F_h$

Table 5.1: Some (partially numerical) abbreviations for logical formulas, where $\underline{F} = (F_{l-1}, \dots, F_0)$ and $\underline{G} = (G_{l-1}, \dots, G_0)$ are vectors of formulas. As usual, an empty conjunction like $\bigwedge_{h=0}^{-1} F_h$ can be replaced by any propositional formula that is true always.

Table 5.1 lists expressions that we use as abbreviations of formulas.
The idea of the construction is as follows.

- We store the counter values in a vector $\underline{\alpha} := \alpha_{n-1}, \dots, \alpha_0$ of shared variables. To this end we embed the sequence $p_0, \dots, p_{2^n-1}$ of points in a sequence of clouds $C_0, \dots, C_{2^{n-1}}$ such that the cloud C_i contains the point p_i and such that the vector $\underline{\alpha}$ of shared variables satisfied at p_i (and hence at all points in C_i) encodes the number i.

- Let $i \leqslant 2^n - 1$ be the number encoded by $\underline{\alpha}$. If $\underline{\alpha}$ contains no 0 then i has reached its highest posible value, the number $2^n - 1$. Otherwise let k be the position of the rightmost 0. We determine the position k with the aid of the formula rightmost_zero(α, k). In order to increment the counter we have to keep all α_j at positions $j > k$ unchanged and to switch all α_j at positions $j \leqslant k$. We do this in two steps:

  1. First me make an $\xrightarrow{L}$-step from the point p_i to a point p_i' where we store the number $i + 1$ in a vector $\underline{X} := X_{n-1}, \dots, X_0$ of usual propositional variables by demanding that

$$p_i' \models (\underline{X} = \underline{\alpha}, > k) \wedge \mathrm{rightmost\_one}(\underline{X}, k).$$

2. Then we make a $\overset{\Diamond}{\rightarrow}$-step from the point p'_i to a point p_{i+1} in the cloud C_{i+1} and demand that

$$p_{i+1} \models (\underline{X} = \underline{\alpha}).$$

Note that in SSL the value of $\underline{X}$ is copied from p'_i to its $\overset{\Diamond}{\rightarrow}$-successor p_{i+1} since in SSL propositional variables are persistent. We achieve the same for S4 × S5 by demanding

$$p_0 \models \text{persistent}(\underline{X}).$$

Altogether we demand that for the number k

$$p_i \models L\big((\underline{X} = \underline{\alpha}, > k) \wedge \text{rightmost\_one}(\underline{X}, k) \wedge \Diamond(\underline{X} = \underline{\alpha})\big).$$

- Additionally we need a formula to ensure that the starting value is 0, that is we demand
$$p_0 \models (\underline{\alpha} = \text{bin}_n(0)).$$

We now define the complete counter formulas, for $n > 0$. Remember that in both formulas $\underline{\alpha}$ is a vector $(\alpha_{n-1}, \ldots, \alpha_0)$ of formulas α_i where in the case of the formula $\text{counter}_{\text{SSL},n}$ the formula α_i is defined by $\alpha_i := L(A_i \wedge \Box LB)$ while in the case of the formula $\text{counter}_{\text{S4}\times\text{S5},n}$ the formula α_i is defined by $\alpha_i := LA_i$; compare Definition 5.1.

$$\begin{aligned}
&\text{counter}_{\text{SSL},n} := \\
&B \wedge (\underline{\alpha} = \text{bin}_n(0)) \\
&\wedge K\Box\Bigg(\bigwedge\nolimits_{k=0}^{n-1}\Bigg((B \wedge \text{rightmost\_zero}(\underline{\alpha}, k)) \rightarrow \\
&L\Big(B \wedge (\underline{X} = \underline{\alpha}, > k) \wedge \text{rightmost\_one}(\underline{X}, k) \wedge \Diamond(\underline{X} = \underline{\alpha})\Big)\Bigg)\Bigg),
\end{aligned}$$

$$\begin{aligned}
&\text{counter}_{\text{S4}\times\text{S5},n} := \\
&\text{persistent}(\underline{X}) \wedge (\underline{\alpha} = \text{bin}_n(0)) \\
&\wedge K\Box\Bigg(\bigwedge\nolimits_{k=0}^{n-1}\Bigg(\text{rightmost\_zero}(\underline{\alpha}, k) \rightarrow \\
&L\Big((\underline{X} = \underline{\alpha}, > k) \wedge \text{rightmost\_one}(\underline{X}, k) \wedge \Diamond(\underline{X} = \underline{\alpha})\Big)\Bigg)\Bigg).
\end{aligned}$$

Proposition 5.2. *1. For all $n \in \mathbb{N}\backslash\{0\}$, the formula* $\text{counter}_{\text{S4}\times\text{S5},n}$ *is* $\text{S4} \times \text{S5}$*-satisfiable.*

2. For all $n \in \mathbb{N}\backslash\{0\}$, the formula $\text{counter}_{\text{SSL},n}$ *is* SSL*-satisfiable.*

3. For all $n \in \mathbb{N}\backslash\{0\}$, for every cross axiom model of $\text{counter}_{\text{SSL},n}$ *and for every point p_0 in this model with $p_0 \models$* $\text{counter}_{\text{SSL},n}$ *there exist a sequence of $2^n - 1$ points $p_1, p_2, \ldots, p_{2^n-1}$ and a sequence of $2^n - 1$ points $p'_0, p'_1, \ldots, p'_{2^n-2}$ such that*

$$\begin{array}{lll} \textit{for } 0 \leqslant i \leqslant 2^n - 1, & p_i \models (\underline{\alpha} = \text{bin}_n(i)), & \\ \textit{for } 0 \leqslant i \leqslant 2^n - 2, & p_i \xrightarrow{L} p'_i \quad \textit{and} \quad p'_i \xrightarrow{\Diamond} p_{i+1} & \textit{and} \\ & p'_i \models (\underline{X} = \text{bin}_n(i+1)). & \end{array}$$

4. The same holds for every $\text{S4} \times \text{S5}$*-commutator model of* $\text{counter}_{\text{S4}\times\text{S5},n}$ *and for every point p_0 in this model with $p_0 \models$* $\text{counter}_{\text{S4}\times\text{S5},n}$*.*

Proof. For the following let us fix some $n > 0$.

1. We construct an $\text{S4} \times \text{S5}$-product model M with a point $(0,0)$ in M such that $M,(0,0) \models \text{counter}_{\text{S4}\times\text{S5},n}$ as follows; see Figure 5.2. We define an S4-frame $(W_1, R_\Diamond)$ by

$$W_1 := \{0, \ldots, 2^n - 1\} \quad \text{and, for } i, i' \in W_1,\ iR_\Diamond i' :\iff i \leqslant i'.$$

We define an S5-frame (W_2, R_L) by

$$W_2 := \{0, \ldots, 2^n - 1\} \quad \text{and} \quad R_L := W_2 \times W_2.$$

Then the product frame $(W, \xrightarrow{\Diamond}, \xrightarrow{L})$ with $W := W_1 \times W_2$ and with $\xrightarrow{\Diamond}$ and $\xrightarrow{L}$ defined as in Definition 3.10.2 is an $\text{S4} \times \text{S5}$-frame. We define the valuation σ by

$$\begin{aligned} \sigma(A_k) &:= \{(i,j) : i,j \in \{0, \ldots, 2^n - 1\} \text{ and } k \in \text{Ones}(i)\}, \\ \sigma(X_k) &:= \{(i,j) : i,j \in \{0, \ldots, 2^n - 1\} \text{ and } k \in \text{Ones}(j)\}, \end{aligned}$$

for $k \in \{0, \ldots, n-1\}$. This implies for all $i, j \in \{0, \ldots, 2^n - 1\}$

$$(i,j) \models (\underline{\alpha} = \text{bin}_n(i)) \quad \text{and} \quad (i,j) \models (\underline{X} = \text{bin}_n(j)).$$

We claim

$$(0,0) \models \text{counter}_{\text{S4}\times\text{S5},n}.$$

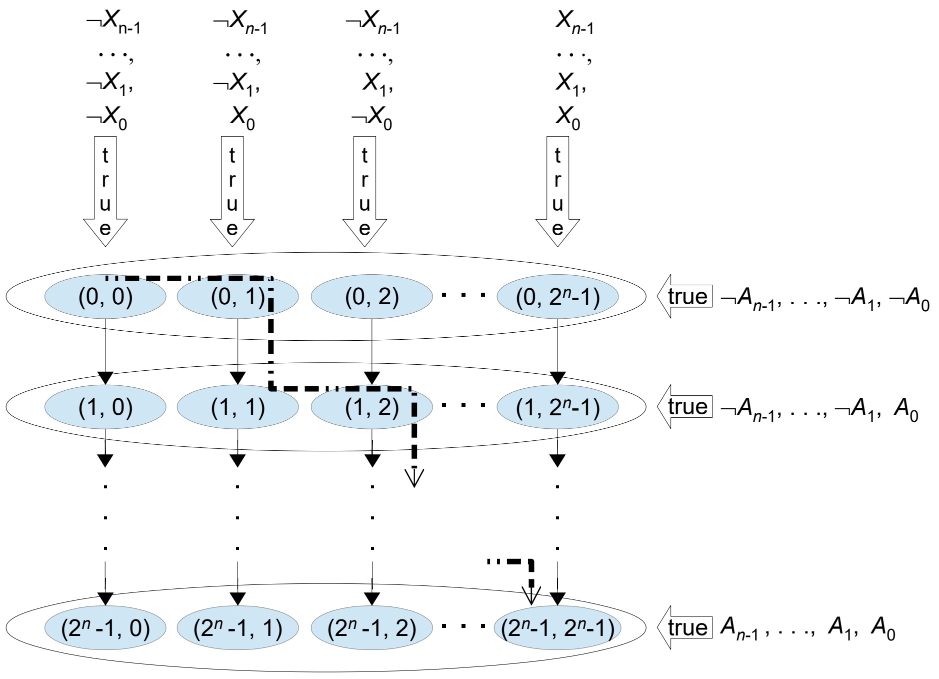


Figure 5.2: An S4 × S5-product model of the formula $\text{counter}_{\text{S4}\times\text{S5},n}$.

Indeed, it is clear that the propositional variables X_k are persistent, hence, we have

$$(0,0) \models \text{persistent}(\underline{X}).$$

It is also clear that

$$(0,0) \models (\underline{\alpha} = \text{bin}_n(0)).$$

Let us assume that for some $(i,j) \in W$ and some $k \in \{0,\dots,n-1\}$ we have

$$(i,j) \models \text{rightmost\_zero}(\underline{\alpha},k).$$

It is sufficient to show that

$$(i,j) \models L\Big((\underline{X} = \underline{\alpha}, > k) \wedge \text{rightmost\_one}(\underline{X},k) \wedge \Diamond(\underline{X} = \underline{\alpha})\Big).$$

Indeed, $(i,j) \models \text{rightmost\_zero}(\underline{\alpha},k)$ implies $\{0,\dots,n-1\}\backslash\text{Ones}(i) \neq \varnothing$ and $k = \min(\{0,\dots,n-1\}\backslash\text{Ones}(i))$. Note that this implies $i < 2^n - 1$, hence, $(i,i+1) \in W$ and $(i+1,i+1) \in W$. In view of $(i,j) \xrightarrow{L} (i,i+1) \xrightarrow{\Diamond} (i+1,i+1)$ it is sufficient to show

$$(i,i+1) \models (\underline{X} = \underline{\alpha}, > k) \wedge \text{rightmost\_one}(\underline{X},k)$$

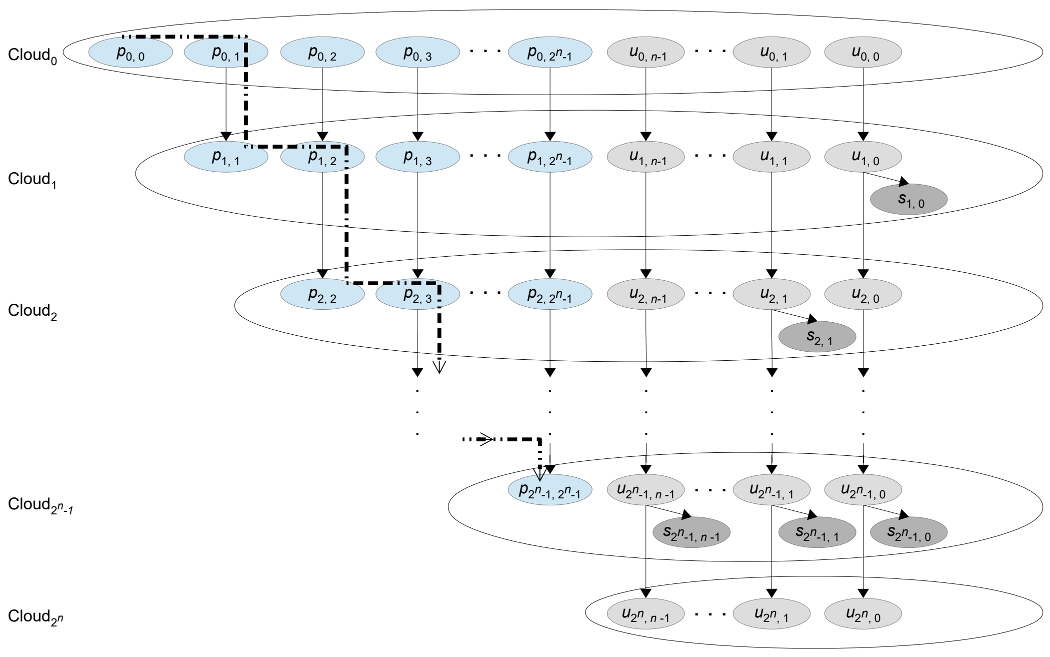


Figure 5.3: A cross axiom model of the formula $\text{counter}_{\text{SSL},n}$.

and

$$(i+1, i+1) \models (\underline{X} = \underline{\alpha}).$$

Both claims follow from the facts that $k = \min(\{0, \dots, n-1\} \setminus \text{Ones}(i))$ and that every point $(x, y) \in W$ satisfies the formulas $(\underline{\alpha} = \text{bin}_n(x))$ and $(\underline{X} = \text{bin}_n(y))$.

2. We construct a cross axiom model $M = (W, \xrightarrow{L}, \xrightarrow{\Diamond}, \sigma)$ with a point $p_{0,0}$ satisfying $M, p_{0,0} \models \text{counter}_{\text{SSL},n}$ as follows; see Figure 5.3. We define

$$W := P \cup U \cup S$$

where

$$\begin{aligned}
P &:= \{p_{i,j} \;:\; i, j \in \{0, \dots, 2^n - 1\} \text{ and } i \leqslant j\}, \\
U &:= \{u_{i,k} \;:\; i \in \{0, \dots, 2^n\} \text{ and } k \in \{0, \dots, n-1\}\}, \\
S &:= \{s_{i,k} \;:\; i \in \{0, \dots, 2^n - 1\} \text{ and } k \in \text{Ones}(i)\}.
\end{aligned}$$

As the relation $\xrightarrow{L}$ is supposed to be an equivalence relation we can define it by defining the $\xrightarrow{L}$-equivalence classes. These are the sets

$$\begin{aligned} C_i &:= \{p_{i,j} : i \leqslant j < 2^n\} \cup \{u_{i,k} : k \in \{0,\dots,n-1\}\} \\ &\qquad \cup \{s_{i,k} : k \in \mathrm{Ones}(i)\}, \end{aligned}$$

for all $i \in \{0,\dots,2^n-1\}$, and

$$C_{2^n} := \{u_{2^n,k} : k \in \{0,\dots,n-1\}\}.$$

We define the relation $\xrightarrow{\Diamond}$ by:

$$\begin{aligned} \xrightarrow{\Diamond} \;:=\; &\{(p_{i,j}, p_{i',j'}) \in P \times P : i \leqslant i' \text{ and } j = j'\} \\ &\cup\{(u_{i,k}, u_{i',k'}) \in U \times U : i \leqslant i' \text{ and } k = k'\} \\ &\cup\{(u_{i,k}, s_{i',k'}) \in U \times S : i \leqslant i' \text{ and } k = k'\} \\ &\cup\{(s_{i,k}, s_{i',k'}) \in S \times S : i = i' \text{ and } k = k'\}. \end{aligned}$$

It is straightforward to check that $\xrightarrow{\Diamond}$ is reflexive and transitive. The cross property is satisfied as well. Thus, $(W, \xrightarrow{L}, \xrightarrow{\Diamond})$ is a cross axiom frame. We define the valuation σ by

$$\begin{aligned} \sigma(A_k) &:= \{u_{i,k} : i \in \{0,\dots,2^n\}\} \cup \\ &\qquad \{s_{i,k} : i \in \{0,\dots,2^n-1\} \text{ and } k \in \mathrm{Ones}(i)\}, \\ \sigma(B) &:= P, \\ \sigma(X_k) &:= \{p_{i,j} : i,j \in \{0,\dots,2^n-1\} \text{ and } k \in \mathrm{Ones}(j)\}, \end{aligned}$$

for $k \in \{0,\dots,n-1\}$.
It is obvious that all of the propositional variables $A_0,\dots,A_{n-1},B$ and $X_0,\dots,X_{n-1}$ are persistent. Thus, $(W, \xrightarrow{L}, \xrightarrow{\Diamond}, \sigma)$ is a cross axiom model. We claim $p_{0,0} \models \mathrm{counter}_{\mathrm{SSL},n}$. Before we show this we show the following claim, for all $i \in \{0,\dots,2^n-1\}$ and for all $p \in C_i$,

$$p \models (\underline{\alpha} = \mathrm{bin}_n(i)). \tag{5.1}$$

In order to show this it is sufficient to show for all $i \in \{0,\dots,2^n-1\}$, for all $p \in C_i$, and for all $k \in \{0,\dots,n-1\}$

$$p \models \alpha_k \iff k \in \mathrm{Ones}(i).$$

Let us fix some $k \in \{0,\dots,n-1\}$. Note that, for all $p' \in P$, we have $p' \models \neg A_k$, hence

$$(\forall p' \in P) \quad p' \models (\neg A_k \vee \Diamond K \neg B).$$

Furthermore, $u_{2^n,h} \models K\neg B$ for all $h \in \{0,\dots,n-1\}$. Since for all $i \in \{0,\dots,2^n\}$ and $h \in \{0,\dots,n-1\}$ we have $u_{i,h} \xrightarrow{\Diamond} u_{2^n,h}$ we obtain $u_{i,h} \models \Diamond K\neg B$ for all $i \in \{0,\dots,2^n\}$ and $h \in \{0,\dots,n-1\}$. Hence,

$$(\forall u' \in U) \quad u' \models (\neg A_k \vee \Diamond K\neg B).$$

This shows that, for any $i \in \{0,\dots,2^n-1\}$, the shared variable $\alpha_k = L(A_k \wedge \Box LB)$ is true in the cloud C_i if, and only if, there exists some $s' \in C_i \cap V$ with $s' \models (A_k \wedge \Box LB)$. As $p' \models B$ for all $p' \in P$ and any $s' \in S$ is $\xrightarrow{L}$-equivalent to some $p' \in P$, we have $s' \models LB$, for all $s' \in S$. Actually, for $s' \in S$ we even have $s' \models \Box LB$ as s' does not have any $\xrightarrow{\Diamond}$-successors besides itself. Thus, the shared variable α_k is true in the cloud C_i if, and only if, there exists some $s' \in C_i \cap S$ with $s' \models A_k$. As the only elements $s' \in C_i \cap S$ are the elements $s_{i,h}$ with $h \in \mathrm{Ones}(i)$ and as $s_{i,h} \models A_k \iff h = k$, we obtain for $i \in \{0,\dots,2^n-1\}$ and for $p \in C_i$,

$$p \models \alpha_k \iff k \in \mathrm{Ones}(i).$$

We have shown the claim (5.1).

We claim that $p_{0,0} \models \mathrm{counter}_{\mathrm{SSL},n}$. Indeed, it is obvious that

$$p_{0,0} \models B.$$

Due to $p_{0,0} \in C_0$ and (5.1) we obtain

$$p_{0,0} \models (\underline{\alpha} = \mathrm{bin}_n(0)).$$

Let us assume that for some $p \in W$ and some $k \in \{0,\dots,n-1\}$ we have $p \models (B \wedge \mathrm{rightmost\_zero}(\underline{\alpha}, k))$. It is sufficient to show that

$$p \models L\big(B \wedge (\underline{X} = \underline{\alpha}, > k) \wedge \mathrm{rightmost\_one}(\underline{X}, k) \wedge \Diamond(\underline{X} = \underline{\alpha})\big).$$

From $p \models B$ we conclude $p \in P$, hence, there exist $i, j \in \{0,\dots,2^n-1\}$ with $i \leqslant j$ and with $p = p_{i,j}$. Thus, $p \in C_i$. Then we have $p \models (\underline{\alpha} = \mathrm{bin}_n(i))$. Now, $p \models \mathrm{rightmost\_zero}(\underline{\alpha}, k)$ implies $\{0,\dots,n-1\}\backslash\mathrm{Ones}(i) \neq \emptyset$ and $k = \min(\{0,\dots,n-1\}\backslash\mathrm{Ones}(i))$. Note that this implies $i < 2^n-1$. We have $p_{i,j} \xrightarrow{L} p_{i,i+1} \xrightarrow{\Diamond} p_{i+1,i+1}$, and since $p_{i,i+1} \in P$ we have

$$p_{i,i+1} \models B.$$

It is sufficient to show

$$p_{i,i+1} \models (\underline{X} = \underline{\alpha}, > k) \wedge \mathrm{rightmost\_one}(\underline{X}, k)$$

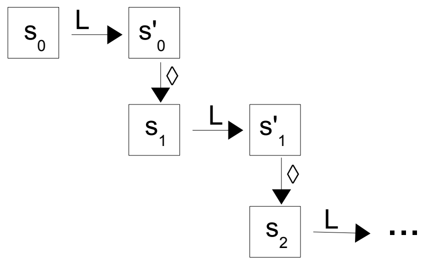


Figure 5.4: staircase of points

and

$$p_{i+1,i+1} \models (\underline{X} = \underline{\alpha}).$$

By definition of σ we have for all $j \in \{0, \ldots, n-1\}$

$$p_{i,i+1} \models X_j \quad \Leftrightarrow \quad j \in \mathrm{Ones}(i+1)$$

and hence on the one hand $p_{i,i+1} \models (\underline{X} = \mathrm{bin}_n(i+1))$. Due to (5.1), we have on the other hand $p_{i,i+1} \models (\underline{\alpha} = \mathrm{bin}_n(i))$. This proves the first claim. For the second claim we observe that by definition of σ also $p_{i+1,i+1} \models (\underline{X} = \mathrm{bin}_n(i+1))$. Since $p_{i+1,i+1} \in C_{i+1}$ we also have by (5.1) that $p_{i+1,i+1} \models (\underline{\alpha} = \mathrm{bin}_n(i+1))$ and hence $p_{i+1,i+1} \models (\underline{X} = \underline{\alpha})$. Thus, we have constructed a cross axiom model for $\mathrm{counter}_{\mathrm{SSL},n}$.

3. Suppose there are a cross axiom model M of the formula $\mathrm{counter}_{\mathrm{SSL},n}$ and some point $p_0 \in M$ with $M, p_0 \models \mathrm{counter}_{\mathrm{SSL},n}$. We show by induction that the claimed sequences of points $p_1, \ldots p_{2^n-1}$ and $p'_0, \ldots p'_{2^n-2}$ with the claimed properties and additionally with $p_i \models B$, for $0 \leqslant i \leqslant 2^n - 1$, and with $p'_i \models B$, for $0 \leqslant i \leqslant 2^n - 2$, exist. In addition, we show that there exist points t_i with $p_0 \xrightarrow{L} t_i$ and $t_i \xrightarrow{\Diamond} p_i$, for $1 \leqslant i \leqslant 2^n - 1$. Note that the sequences $p_0, \ldots p_{2^n-1}$ and $p'_0, \ldots p'_{2^n-2}$ are supposed to form a "staircase" as in Figure 5.4. By definition, $p_0 \models B \wedge (\underline{\alpha} = \mathrm{bin}_n(0))$. By induction hypothesis, let us assume that for some m with $0 \leqslant m < 2^n - 1$ there exist $p_1, \ldots, p_m$ and $p'_0, \ldots, p'_{m-1}$ with

$$p_0 \xrightarrow{L} p'_0 \xrightarrow{\Diamond} p_1 \xrightarrow{L} \ldots \xrightarrow{L} p'_{m-1} \xrightarrow{\Diamond} p_m,$$

with

$$p_i \models (B \wedge (\underline{\alpha} = \mathrm{bin}_n(i))),$$

for $0 \leqslant i \leqslant m$, and with

$$p'_i \models (B \wedge (\underline{X} = \mathrm{bin}_n(i+1))),$$

for $0 \leqslant i < m$, and that there are t_i with $p_0 \xrightarrow{L} t_i$ and $t_i \xrightarrow{\Diamond} p_i$, for $1 \leqslant i \leqslant m$. Since $m < 2^n - 1$, the set $\{0, \ldots, n-1\} \backslash \mathrm{Ones}(m)$ is nonempty. With $k := \min(\{0, \ldots, n-1\} \backslash \mathrm{Ones}(m))$ we have

$$p_m \models \mathrm{rightmost\_zero}(\underline{\alpha}, k).$$

Due to $p_0 \models \mathrm{counter}_{\mathrm{SSL},n}$ as well as $p_0 \xrightarrow{L} t_m \xrightarrow{\Diamond} p_m$ this implies

$$\begin{aligned} p_m \models \quad & L\big(\; B \wedge (\underline{X} = \underline{\alpha}, > k) \wedge \mathrm{rightmost\_one}(\underline{X}, k) \\ & \quad \wedge \Diamond(\underline{X} = \underline{\alpha})\big). \end{aligned}$$

Thus, there must exist points p'_m and p_{m+1} satisfying $p'_m \models B$ as well as $p_m \xrightarrow{L} p'_m \xrightarrow{\Diamond} p_{m+1}$,

$$p'_m \models (\underline{X} = \underline{\alpha}, > k) \wedge \mathrm{rightmost\_one}(\underline{X}, k)$$

and

$$p_{m+1} \models (\underline{X} = \underline{\alpha}).$$

We have to show

$$p'_m \models (\underline{X} = \mathrm{bin}_n(m+1))$$

and

$$p_{m+1} \models B \wedge (\underline{\alpha} = \mathrm{bin}_n(m+1)).$$

Due to the fact that p'_m is an element of the same cloud as p_m, and $\underline{\alpha}$ has the same value in all points in a cloud we obtain

$$p'_m \models (\underline{\alpha} = \mathrm{bin}_n(m)).$$

Together with

$$p'_m \models (\underline{X} = \underline{\alpha}, > k) \wedge \mathrm{rightmost\_one}(\underline{X}, k)$$

this implies

$$p'_m \models (\underline{X} = \mathrm{bin}_n(m+1))$$

(the values of the leading bits $\alpha_{n-1}, \ldots, \alpha_{k+1}$ of $\underline{\alpha}$ are copied to the leading bits $X_{n-1}, \ldots, X_{k+1}$ and the other bits are defined explicitly by $p'_m \models \mathrm{rightmost\_one}(\underline{X}, k)$ so that the binary value of $\underline{X}$ is $m+1$). From $p'_m \xrightarrow{\Diamond} p_{m+1}$ and the fact that in SSL propositional variables are persistent we obtain $p_{m+1} \models (B \wedge (\underline{X} = \mathrm{bin}_n(m+1)))$. Using $p_{m+1} \models (\underline{X} = \underline{\alpha})$ we obtain $p_{m+1} \models (\underline{\alpha} = \mathrm{bin}_n(m+1))$. Finally, the cross property applied to $t_m \xrightarrow{\Diamond} p_m$ and $p_m \xrightarrow{L} p'_m$ implies that there exists a point t_{m+1} with $t_m \xrightarrow{L} t_{m+1}$ and $t_{m+1} \xrightarrow{\Diamond} p'_m$. Using additionally $p_0 \xrightarrow{L} t_m$ and $p'_m \xrightarrow{\Diamond} p_{m+1}$ we obtain $p_0 \xrightarrow{L} t_{m+1}$ and $t_{m+1} \xrightarrow{\Diamond} p_{m+1}$. This ends the proof of the third assertion.

4. The proof for S4 × S5-commutator models of the formula $\text{counter}_{\text{S4}\times\text{S5},n}$ is very similar to the proof for cross axiom models of the formula $\text{counter}_{\text{SSL},n}$. For completeness sake we explicate it in detail. Suppose there are an S4 × S5-commutator model M of the formula $\text{counter}_{\text{S4}\times\text{S5},n}$ and some point $p_0 \in M$ with $M, p_0 \models \text{counter}_{\text{S4}\times\text{S5},n}$. We show by induction that the claimed sequences of points $p_1, \dots p_{2^n-1}$ and $p'_0, \dots p'_{2^n-2}$ with the claimed properties exist. In addition, we show that there exist points t_i with $p_0 \xrightarrow{L} t_i$ and $t_i \xrightarrow{\Diamond} p_i$, for $1 \leqslant i \leqslant 2^n - 1$. Note that the sequences $p_0, \dots p_{2^n-1}$ and $p'_0, \dots p'_{2^n-2}$ are supposed to form a "staircase" as in Figure 5.4. By definition, $p_0 \models \text{persistent}(\underline{X}) \wedge (\underline{\alpha} = \text{bin}_n(0))$. By induction hypothesis, let us assume that for some m with $0 \leqslant m < 2^n - 1$ there exist $p_1, \dots, p_m$ and $p'_0, \dots, p'_{m-1}$ with

$$p_0 \xrightarrow{L} p'_0 \xrightarrow{\Diamond} p_1 \xrightarrow{L} \dots \xrightarrow{L} p'_{m-1} \xrightarrow{\Diamond} p_m,$$

with

$$p_i \models (\underline{\alpha} = \text{bin}_n(i)),$$

for $0 \leqslant i \leqslant m$, and with

$$p'_i \models (\underline{X} = \text{bin}_n(i+1)),$$

for $0 \leqslant i < m$, and that there are t_i with $p_0 \xrightarrow{L} t_i$ and $t_i \xrightarrow{\Diamond} p_i$, for $1 \leqslant i \leqslant m$. Since $m < 2^n - 1$, the set $\{0, \dots, n-1\}\backslash\text{Ones}(m)$ is nonempty. With $k := \min(\{0, \dots, n-1\}\backslash\text{Ones}(m))$ we have

$$p_m \models \text{rightmost\_zero}(\underline{\alpha}, k).$$

Due to $p_0 \models \text{counter}_{\text{S4}\times\text{S5},n}$ as well as $p_0 \xrightarrow{L} t_m \xrightarrow{\Diamond} p_m$ this implies

$$p_m \models L\big((\underline{X} = \underline{\alpha}, > k) \wedge \text{rightmost\_one}(\underline{X}, k) \wedge \Diamond(\underline{X} = \underline{\alpha})\big).$$

Thus, there must exist points p'_m and p_{m+1} satisfying $p_m \xrightarrow{L} p'_m \xrightarrow{\Diamond} p_{m+1}$ as well as

$$p'_m \models (\underline{X} = \underline{\alpha}, > k) \wedge \text{rightmost\_one}(\underline{X}, k)$$

and

$$p_{m+1} \models (\underline{X} = \underline{\alpha}).$$

We have to show

$$p'_m \models (\underline{X} = \text{bin}_n(m+1))$$

and

$$p_{m+1} \models B \wedge (\underline{\alpha} = \mathrm{bin}_n(m+1)).$$

Due to the fact that p'_m is an element of the same cloud as p_m, and $\underline{\alpha}$ has the same value in all points in a cloud we obtain

$$p'_m \models (\underline{\alpha} = \mathrm{bin}_n(m)).$$

Together with

$$p'_m \models (\underline{X} = \underline{\alpha}, > k) \wedge \mathrm{rightmost\_one}(\underline{X}, k)$$

this implies

$$p'_m \models (\underline{X} = \mathrm{bin}_n(m+1))$$

(the values of the leading bits $\alpha_{n-1}, \ldots, \alpha_{k+1}$ of $\underline{\alpha}$ are copied to the leading bits $X_{n-1}, \ldots, X_{k+1}$ and the other bits are defined explicitly by $p'_m \models \mathrm{rightmost\_one}(\underline{X}, k)$ so that the binary value of $\underline{X}$ is $m+1$). From $p_0 \models \mathrm{persistent}(\underline{X})$ as well as $p_0 \xrightarrow{L} t_m \xrightarrow{\Diamond} p'_m \xrightarrow{\Diamond} p_{m+1}$ we obtain $p_{m+1} \models (\underline{X} = \mathrm{bin}_n(m+1))$. Using $p_{m+1} \models (\underline{X} = \underline{\alpha})$ we obtain $p_{m+1} \models (\underline{\alpha} = \mathrm{bin}_n(m+1))$. Finally, the left commutativity property applied to $t_m \xrightarrow{\Diamond} p_m$ and $p_m \xrightarrow{L} p'_m$ implies that there exists a point t_{m+1} with $t_m \xrightarrow{L} t_{m+1}$ and $t_{m+1} \xrightarrow{\Diamond} p'_m$. Using additionally $p_0 \xrightarrow{L} t_m$ and $p'_m \xrightarrow{\Diamond} p_{m+1}$ we obtain $p_0 \xrightarrow{L} t_{m+1}$ and $t_{m+1} \xrightarrow{\Diamond} p_{m+1}$. This ends the proof of the fourth assertion. □

5.3 Alternating Turing Machines

The concept of an *Alternating Turing Machine* (ATM) was set forth by Chandra and Stockmeyer [19] and independently by Kozen [55] in 1976, with a joint journal publication in 1981 [18]. We are going to use a variant of ATMs with a single tape as in [64]. This is justified by the fact that one-tape ATMs can efficiently simulate multi-tape ATMs; see [19, Proposition 3.4]. For an even more efficient simulation of multi-tape ATMs by one-tape ATMs see [76].

An ATM is a nondeterministic Turing machine where some configurations are “or” configurations that accept if at least one of their successors does, while other configurations are “and” configurations that accept if all of their successors accept. The mode of each configuration (“and” vs. “or”) is determined by the state of the configuration. There are two special states called q_{accept} and q_{reject}. All other states are either *universal states* or *existential states.*

For a relation $\delta \subseteq X \times Y$ and any $x \in X$ we write $\delta(x) := \{y \in Y \mid (x, y) \in \delta\}$.

Definition 5.3. An *alternating Turing Machine* M is a quintuple

$$M = (Q, \Sigma, \Gamma, q_0, \delta),$$

where

- Q, the set of *states* of M, is the disjoint union of the following four sets:
  - $Q_\exists$, a finite set, its elements are called *existential states*,
  - $Q_\forall$, a finite set, its elements are called *universal states*,
  - $\{q_{\text{accept}}\}$, a one-element set, its element is called *accepting state*,
  - $\{q_{\text{reject}}\}$, a one-element set, its element is called *rejecting state*,
- Σ is a nonempty finite set, called the *input alphabet*,
- $\Gamma \supseteq \Sigma$ is a finite set containing a *blank* symbol $\# \notin \Sigma$, we call Γ the *tape alphabet*,
- $q_0 \in Q$ is the *initial state*,
- $\delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{\mathit{left}, \mathit{right}\})$ is the *transition relation*,

and where δ satisfies the condition

$$\delta(q, a) \begin{cases} = \varnothing & \text{for } q \in \{q_{\text{accept}}, q_{\text{reject}}\} \text{ and } a \in \Gamma, \\ \neq \varnothing & \text{for } q \in Q_\exists \cup Q_\forall \text{ and } a \in \Gamma. \end{cases}$$

A *configuration* of an alternating Turing machine M is an element (q, z, γ) of

$$C_M = Q \times \mathbb{Z} \times \Gamma^{\mathbb{Z}}$$

where $q \in Q$ is the current state of the finite control, $z \in \mathbb{Z}$ is the current position of the tape head (that is, the number of the cell on which the tape head is positioned), and the function $\gamma : \mathbb{Z} \to \Gamma$ represents the current tape content and satisfies the condition $\gamma(z) = \#$ for all but finitely many $z \in \mathbb{Z}$. A configuration represents an instantaneous description of M at some point in a computation. The *initial configuration* of M on input $w = w_1 \dots w_n \in \Sigma^*$ with $w_i \in \Gamma$ for $i = 1 \dots, |w|$ is

$$\sigma_M(w) := (q_0, 0, \gamma_w)$$

where $\gamma_w : \mathbb{Z} \to \Gamma$ is defined by

$$\gamma_w(z) := \begin{cases} \# & \text{if } z \leqslant 0 \text{ or } z > |w|, \\ w_z & \text{if } z \in \{1, \dots, |w|\}. \end{cases}$$

That means that at the start of the computation the tape head is positioned on cell no. 0, and the input string w is contained in the cells with the numbers 1 to $|w|$ while all other cells contain the blank $\#$.
For two configurations c and c' we write $c \vdash c'$ and say c' *is a successor of* c, if, according to the transition relation δ, the configuration c' can be reached from the configuration c in one step (this is defined in the usual way). The reflexive-transitive closure of $\vdash$ is denoted $\vdash^*$. A *computation* or *computation path* of M on input w is a sequence $c_0 \vdash \ldots \vdash c_m$, where $c_0 = \sigma_M(w)$. In the following we will only consider ATMs M such that there exists a function $t : \mathbb{N} \to \mathbb{N}$ such that for any $n \in \mathbb{N}$ and any possible input string $w \in \Sigma^n$, any computation path of M on input w has length at most $t(n)$, that is, if $c_0, \ldots, c_m$ is a computation path of M on input w then $m \leqslant t(n)$. In this case we say that M *works in time* t. For such machines M we can define the *language* $L(M) \subseteq \Sigma^*$ *recognized by* M as follows: for $w \in \Sigma^*$

$$w \in L(M) :\Longleftrightarrow \text{there exists an "accepting tree of } M \text{ on input } w\text{"}.$$

An *accepting tree of* M *on input* w is a finite rooted and labeled tree each of whose nodes is labeled with a configuration of M such that the following five properties hold true:

I. The root of the tree is labeled with the initial configuration $\sigma_M(w)$ of M on input w.

II. If c is the label of an internal node of the tree then the labels of its successors are configurations c' satisfying $c \vdash c'$ (note that this implies that the state q of c is an element of $Q_\exists \cup Q_\forall$).

III If c is the label of an internal node of the tree then the labels of its successors are pairwise different configurations.

IV. If c is the label of an internal node of the tree and the state q of c is an element of $Q_\forall$ then for every configuration c' with $c \vdash c'$ there exists a successor node labeled with c'.

V. If c is the label of a leaf of the tree then the state of c is equal to q_{accept}.

Note that these conditions imply that, if c is the label of a node of the tree and the state q of c is an element of $Q_\exists$, then this node is an internal node, hence, it has a successor. Let the height of a rooted tree be the length of the longest path in the tree. It is clear that if there is an accepting tree of M on input w then its height is at most $t(|w|)$.
The time complexity class AEXPTIME is the class of all languages L such that there exist an alternating Turing machine M with $L = L(M)$ and a

polynomial p such that M works in time $2^{p(n)}$. We will make use of the fundamental fact AEXPTIME = EXPSPACE [19, Corollary 3.6].
For technical purposes we will also need the following notion. A *partial tree of M on input w* is a finite rooted and labeled tree each of whose nodes is labeled with a configuration of M that satisfies the same four properties I, II, III, and IV as an accepting tree of M on input w and instead of the property V the following weaker property:

V′. If c is the label of a leaf of the tree then the state of c is different from q_{reject}.

It is clear that any accepting tree of M on input w is a partial tree of M on input w. Usually we will write a partial tree of M on input w as a triple $T = (V, E, c)$ where V is the set of nodes of the tree T, where $E \subseteq V \times V$ is the set of edges of the tree T (note that the root *root* of the tree is uniquely determined by V and E: it is the only node that does not have an incoming edge), and where $c : V \to Q \times \mathbb{Z} \times \Gamma^{\mathbb{Z}}$ is the labeling of the tree. Let E^* be the reflexive-transitive closure of E. We will often need the following data associated with any computation node $v \in V$:

- The time $\mathit{time}(v)$ of v. This is the number of edges of the unique path in T from *root* to v. Note that $\mathit{time}(\mathit{root}) = 0$.
- The configuration $c(v)$ of v. This is the configuration with which the node v is labeled.
- The state $\mathit{state}(v)$ of the configuration $c(v)$.
- The position $\mathit{pos}(v)$ of the tape head in the configuration $c(v)$.
- The symbol $\mathit{read}(v)$ in the cell $\mathit{pos}(v)$ in the configuration $c(v)$.
- The predecessor $\mathit{pred}(v)$ of v in the tree T, for $v \neq \mathit{root}$.
- The symbol $\mathit{written}(v)$ that has been written in the computation step that lead to this node v, for $v \neq \mathit{root}$. Note that this is the symbol now contained in the cell $\mathit{pos}(\mathit{pred}(v))$ on which the tape head was positioned in the previous configuration.

Chapter 6

Reduction of Alternating Turing Machines Working in Exponential Time to SSL

In this chapter we prove the following theorem.

Theorem 6.1. *The satisfiability problem of* SSL *is* EXPSPACE*-hard under logarithmic space reduction.*

As explained at the beginning of the previous chapter, we are going to show this by reducing any language L recognized by an Alternating Turing Machine working in exponential time to the satisfiability problem of SSL.
In the following section we will first describe the idea of the reduction and then define the reduction function f_{SSL} in detail. In Section 6.2 we will show that it can be computed in logarithmic space. In the final two sections we are going to show that it is corrrect. First we show that in case $w \in L$ the formula $f_{\mathrm{SSL}}(w)$ is SSL-satisfiable by explicitly constructing an SSL cross axiom model for $f_{\mathrm{SSL}}(w)$. In the last section we show that if $f_{\mathrm{SSL}}(w)$ is SSL-satisfiable then w is an element of L.

6.1 Construction of the Formula

Let $L \in$ EXPSPACE be an arbitrary language over some alphabet Σ, that is, $L \subseteq \Sigma^*$. We are going to show that there is a logspace computable function f_{SSL} mapping strings to strings such that, for any $w \in \Sigma^*$,

- $f_{\mathrm{SSL}}(w)$ is a bimodal formula and
- $f_{\mathrm{SSL}}(w)$ is SSL-satisfiable if, and only if, $w \in L$.

Once we have shown this, we have shown the result. In order to define this desired reduction function f_{SSL}, we are going to make use of an Alternating Turing Machine for L. Since EXPSPACE = AEXPTIME, there exist an Alternating Turing Machine $M = (Q, \Sigma, \Gamma, q_0, \delta)$ and a univariate polynomial p such that M accepts L, that is, $L(M) = L$, and such that the time used by M on arbitrary input of length n is bounded by $2^{p(n)} - 1$. We can assume without loss of generality $Q = \{0, \ldots, |Q| - 1\}$, $\Gamma = \{0, \ldots, |\Gamma| - 1\}$, that the coefficients of the polynomial p are natural numbers and that, for all $n \in \mathbb{N}$, we have $p(n) \geqslant n$ and $p(n) \geqslant 1$. In the following, whenever we have fixed some $n \in \mathbb{N}$, we set

$$N := p(n).$$

Let us consider an input word $w \in \Sigma^n$ of length n, for some $n \in \mathbb{N}$, and let us sketch the main idea of the construction of the formula $f_{\mathrm{SSL}}(w)$. The formula $f_{\mathrm{SSL}}(w)$ will describe the possible computations of M on input w in the following sense: any cross axiom model of $f_{\mathrm{SSL}}(w)$ will essentially contain an accepting tree of M on input w, and if $w \in L(M)$ then there exists an accepting tree of M on input w and one can turn this into a cross axiom model of $f_{\mathrm{SSL}}(w)$. In such a model, any node in an accepting tree of M on input w will be modeled by a cloud (that is, by an $\xrightarrow{L}$-equivalence class) in which certain shared variables (we use the notion “shared variables” in the same sense as in Section 5.1) will have values that describe the data of the computation node that are important in this computation step. Which data are these? First of all, we need the time of the computation node. We assume that the computation starts with the initial configuration of M on input w at time 0. Since the ATM M needs at most $2^N - 1$ time steps, we can store the time of each computation node in a binary counter counting from 0 to $2^N - 1$. Since during each time step at most one additional cell either to the right or to the left of the previous cell can be visited, we can describe any configuration reachable during a computation of M on input w by the following data:

- the current content of the tape, which is a string in $\Gamma^{2\cdot(2^N-1)+1} = \Gamma^{2^{N+1}-1}$,
- the current tape head position, which is a number in $\{0, \ldots, 2^{N+1} - 2\}$.

We assume that in the initial configuration on input w the tape content is $\#^{2^N} w \#^{2^N-1-n}$ (remember that we use # for the blank symbol) and that the tape head scans the blank # to the left of the first symbol of w, that is, the position of the tape head is $2^N - 1$. If a cloud in a cross axiom model of $f_{\mathrm{SSL}}(w)$ describes a computation node of M on input w then in this cloud the following shared variables will have the following values:

- a vector $\underline{\alpha}^{\text{time}} = (\alpha^{\text{time}}_{N-1}, \ldots, \alpha^{\text{time}}_0)$ giving in binary the current time of the computation,
- a vector $\underline{\alpha}^{\text{pos}} = (\alpha^{\text{pos}}_{N}, \ldots, \alpha^{\text{pos}}_0)$ giving in binary the current position of the tape head,
- a vector $\underline{\alpha}^{\text{state}} = (\alpha^{\text{state}}_0, \ldots, \alpha^{\text{state}}_{|Q|-1})$ giving in unary the current state of the computation (here "unary" means: exactly one of the shared variables α^{state}_i will be true, namely the one with i being the current state),
- a vector $\underline{\alpha}^{\text{written}} = (\alpha^{\text{written}}_0, \ldots, \alpha^{\text{written}}_{|\Gamma|-1})$ giving in unary the symbol that has just been written into the cell that has just been left, unless the cloud corresponds to the first node in the computation tree — in that case the value of this vector is irrelevant (here "unary" means: exactly one of the variables $\alpha^{\text{written}}_i$ will be true, namely the one with i being the symbol that has just been written),
- a vector $\underline{\alpha}^{\text{read}} = (\alpha^{\text{read}}_0, \ldots, \alpha^{\text{read}}_{|\Gamma|-1})$ giving in unary the symbol in the current cell (here "unary" means: exactly one of the shared variables α^{read}_i will be true, namely the one with i being the symbol in the current cell).

The formula $f_{\text{SSL}}(w)$ has to ensure that for any possible computation step in an accepting tree starting from such a computation node there exists a cloud describing the corresponding successor node in the accepting tree. In this new cloud, the value of the counter for the time $\underline{\alpha}^{\text{time}}$ has to be incremented. This can be done by the technique described in Section 5.2 for implementing a binary counter. In parallel, we have to make sure that in this new cloud also the vectors $\underline{\alpha}^{\text{pos}}$, $\underline{\alpha}^{\text{state}}$, $\underline{\alpha}^{\text{written}}$, and $\underline{\alpha}^{\text{read}}$ are set to the right values. For the vectors $\underline{\alpha}^{\text{pos}}$, $\underline{\alpha}^{\text{state}}$, and $\underline{\alpha}^{\text{written}}$ these values can be computed using the corresponding element of the transition relation δ of the ATM. For example, $\underline{\alpha}^{\text{pos}}$ has to be decremented by one if the tape head moves to the left, and it has to be incremented by one if the tape head moves to the right. Also the new state (to be stored in $\underline{\alpha}^{\text{state}}$) and the symbol written into the cell that has just been left (to be stored in $\underline{\alpha}^{\text{written}}$) are determined by the data of the previous computation node and by the corresponding element of the transition relation δ.
But the vector $\underline{\alpha}^{\text{read}}$ is supposed to describe the symbol in the current cell. This symbol is not determined by the current computation step but has either been written the last time when this cell has been visited during this computation or, when this cell has never been visited before, the symbol in

this cell is still the one that was contained in this cell before the computation started. How can one ensure that $\underline{\alpha}^{\text{read}}$ is set to the right value? If the current cell has never been visited before, we have to make sure that the value is set to the correct value describing the inital content of this cell. Otherwise, we make use of the cross property. The point in the new cloud whose existence is enforced by the formula must have a cross point in any cloud corresponding to any previous computation node. The idea is that one of these cross points picks up the right value in the right cloud. We are going to make sure that the cloud is identified that corresponds to the configuration after the previous visit of the same cell during the computation. Then in the cloud corresponding to this configuration the value of $\underline{\alpha}^{\text{written}}$ will tell us the symbol that has been written into the current cell during the previous visit. In order to identify the correct cloud of the step after the previous visit of the current cell and to copy the value of the symbol, the formula $f_{\text{SSL}}(w)$ will ensure that any cloud describing a computation node will contain a point in which the following (persistent!) propositional variables have the following values:

- a vector $\underline{X}^{\text{time}} = (X^{\text{time}}_{N-1}, \ldots, X^{\text{time}}_0)$ giving in binary the current time of the computation,
- a vector $\underline{X}^{\text{pos}} = (X^{\text{pos}}_{N}, \ldots, X^{\text{pos}}_0)$ giving in binary the current position of the tape head, that is the position of the current cell,
- a vector $\underline{X}^{\text{read}} = (X^{\text{read}}_0, \ldots, X^{\text{read}}_{|\Gamma|-1})$ giving in unary the symbol in the current cell, (here "unary" means: exactly one of the variables X^{read}_i will be true, namely the one with i being the symbol in the current cell).
- a vector $\underline{X}^{\text{time-apv}} = (X^{\text{time-apv}}_{N-1}, \ldots, X^{\text{time-apv}}_0)$ giving in binary the time one step after the previous visit of the cell, if it has been visited before ("time-apv" stands for "time after previous visit"); otherwise this vector will have the binary value 0.

Now we come to the formal definition of the formula $f_{\text{SSL}}(w)$. The formula $f_{\text{SSL}}(w)$ will have the following structure:

$$\begin{aligned} f_{\text{SSL}}(w) \quad := \quad & K\Box \mathit{uniqueness} \\ & \wedge \mathit{start} \\ & \wedge K\Box \mathit{time\_after\_previous\_visit} \\ & \wedge K\Box \mathit{get\_the\_right\_symbol} \\ & \wedge K\Box \mathit{computation} \\ & \wedge K\Box \mathit{no\_reject}. \end{aligned}$$

The formula $f_{\mathrm{SSL}}(w)$ will contain the following propositional variables:

$$\begin{array}{l} B, \\ A^{\mathrm{time}}_{N-1}, \ldots, A^{\mathrm{time}}_{0}, \\ A^{\mathrm{pos}}_{N}, \ldots, A^{\mathrm{pos}}_{0}, \\ A^{\mathrm{state}}_{0}, \ldots, A^{\mathrm{state}}_{|Q|-1}, \\ A^{\mathrm{written}}_{0}, \ldots, A^{\mathrm{written}}_{|\Gamma|-1}, \\ A^{\mathrm{read}}_{0}, \ldots, A^{\mathrm{read}}_{|\Gamma|-1}, \\ X^{\mathrm{time}}_{N-1}, \ldots, X^{\mathrm{time}}_{0}, \\ X^{\mathrm{time\text{-}apv}}_{N-1}, \ldots, X^{\mathrm{time\text{-}apv}}_{0}, \\ X^{\mathrm{pos}}_{N}, \ldots, X^{\mathrm{pos}}_{0}, \\ X^{\mathrm{read}}_{0}, \ldots, X^{\mathrm{read}}_{|\Gamma|-1}. \end{array}$$

For *string* $\in \{\mathrm{time}, \mathrm{pos}, \mathrm{state}, \mathrm{written}, \mathrm{read}\}$ and natural numbers k we define

$$\alpha^{string}_k := L(A^{string}_k \wedge \Box LB).$$

These formulas α^{string}_k are the shared variables we talked about above. We are now going to define the subformulas of $f_{\mathrm{SSL}}(w)$. We will use the abbreviations introduced above, in Table 5.1, and in Table 6.1.
The models of the formula $f_{\mathrm{SSL}}(w)$ will contain certain "information" points that will realize an accepting tree of M on input w if, and only if, $w \in L$. Besides these information points there will also be other, "auxiliary", points (and an $\xrightarrow{L}$-equivalence class not containing any information points) whose sole purpose is to make the mechanism of shared variables work. In several formulas we need to distinguish between the information points and the other, auxiliary, points. It turns out that this can be done simply by the truth value of the propositional variable B.
The following formula makes sure that in each of the vectors of shared variables that describe in a unary way the current state respectively the written symbol respectively the current symbol, exactly one shared variable is true:

$$\mathit{uniqueness} \;\; := \;\; B \to \big(\mathrm{unique}(\underline{\alpha}^{\mathrm{state}}) \wedge \mathrm{unique}(\underline{\alpha}^{\mathrm{written}}) \wedge \mathrm{unique}(\underline{\alpha}^{\mathrm{read}})\big).$$

The vector $\underline{X}^{\mathrm{read}}$ will satisfy the same uniqueness condition automatically.
The following formula ensures that the variables in the cloud corresponding to the first node in a computation tree get the correct values. The computation starts at time 0 with the tape head at position $2^N - 1$ and in the state q_0 and with the blank symbol # in the current cell.

$$\mathit{start} := B \wedge (\underline{\alpha}^{\mathrm{time}} = \mathrm{bin}_N(0)) \wedge (\underline{\alpha}^{\mathrm{pos}} = \mathrm{bin}_{N+1}(2^N - 1)) \wedge \alpha^{\mathrm{state}}_{q_0} \wedge \alpha^{\mathrm{read}}_{\#}.$$

For	the following expression	is an abbreviation of the following formula
$l \geqslant 1$	unique($\underline{F}$)	$\bigvee_{k=0}^{l-1} F_k \wedge \bigwedge_{k=0}^{l-1} \bigwedge_{m=k+1}^{l-1} \neg(F_k \wedge F_m)$
$l \geqslant 1$	$(\underline{F} \neq \underline{G})$	$\neg(\underline{F} = \underline{G})$
$l \geqslant 1$	$(\underline{F} < \underline{G})$	$\bigvee_{k=0}^{l-1} ((\underline{F} = \underline{G}, > k) \wedge \neg F_k \wedge G_k)$
$l \geqslant 1$	$(\underline{F} \leqslant \underline{G})$	$(\underline{F} < \underline{G}) \vee (\underline{F} = \underline{G})$
$l \geqslant 1$	$(\underline{F} = \underline{G} + 1)$	$\bigvee_{k=0}^{l-1}\big((\underline{F} = \underline{G}, > k)$ $\wedge \text{rightmost\_one}(\underline{F}, k)$ $\wedge \text{rightmost\_zero}(\underline{G}, k)\big)$
$l \geqslant 1$	$(\underline{F} \neq \underline{G} + 1)$	$\neg(\underline{F} = \underline{G} + 1)$
$l \geqslant 1$, $0 \leqslant i < 2^l$	$(\underline{F} < \text{bin}_l(i))$	$\bigvee_{k \in \text{Ones}(i)} \big(\neg F_k$ $\wedge \bigwedge_{h \in \{k+1,\dots,l-1\} \setminus \text{Ones}(i)} \neg F_h\big)$
$l \geqslant 1$, $0 \leqslant i < 2^l$	$(\underline{F} \leqslant \text{bin}_l(i))$	$(\underline{F} < \text{bin}_l(i)) \vee (\underline{F} = \text{bin}_l(i))$
$l \geqslant 1$, $0 \leqslant i < 2^l$	$(\underline{F} > \text{bin}_l(i))$	$\neg(\underline{F} \leqslant \text{bin}_l(i))$

Table 6.1: Some (partially numerical) abbreviations for logical formulas, where $\underline{F} = (F_{l-1}, \dots, F_0)$ and $\underline{G} = (G_{l-1}, \dots, G_0)$ are vectors of formulas. An empty conjunction like $\bigwedge_{h=0}^{-1} F_h$ can be replaced by any propositional formula that is true always. An empty disjunction like $\bigvee_{h=0}^{-1} F_h$ can be replaced by any propositional formula that is false always.

The following formula ensures that the vector $\underline{X}^{\text{time-apv}}$ stores the time after the previous visit of the same cell, if it has been visited before. If it has never been visited before, this vector gets the binary value 0.

$$\begin{aligned}
&\mathit{time\_after\_previous\_visit}\\
&:= \quad B \rightarrow \Bigg(\Big(\underline{X}^{\text{time-apv}} \leqslant \underline{X}^{\text{time}} \Big) \\
&\qquad \wedge \Big((\underline{\alpha}^{\text{time}} < \underline{X}^{\text{time}} \wedge \underline{\alpha}^{\text{pos}} \neq \underline{X}^{\text{pos}}) \rightarrow (\underline{X}^{\text{time-apv}} \neq \underline{\alpha}^{\text{time}} + 1) \Big) \\
&\qquad \wedge \Big((\underline{\alpha}^{\text{time}} < \underline{X}^{\text{time}} \wedge \underline{\alpha}^{\text{pos}} = \underline{X}^{\text{pos}}) \rightarrow (\underline{\alpha}^{\text{time}} < \underline{X}^{\text{time-apv}}) \Big) \Bigg).
\end{aligned}$$

We explain this formula. The time $\underline{X}^{\text{time-apv}}$ after the previous visit of the current cell $\underline{X}^{\text{pos}}$ is certainly at most as large as the current time $\underline{X}^{\text{time}}$. When during the computation at an earlier time a cell has been visited that is different from the current one then one plus the time of that visit is certainly not the time after the previous visit of the current cell. When during the computation at an earlier time the current cell has been visited then the time of that visit is a strict lower bound for the time after the previous visit of the current cell. Together these conditions ensure that $\underline{X}^{\text{time-apv}}$ gets the correct value.
The following formula ensures that the vector $\underline{X}^{\text{read}}$ stores (in unary form) the symbol in the current cell.

$$\begin{aligned}
&\mathit{get\_the\_right\_symbol} := \\
&\Bigg(\Big(B \wedge (\underline{X}^{\text{time-apv}} = \text{bin}_N(0)) \Big) \rightarrow \\
&\quad \Big(\textstyle\bigwedge_{i=1}^{n} ((\underline{X}^{\text{pos}} = \text{bin}_{N+1}(2^N - 1 + i)) \rightarrow X^{\text{read}}_{w_i}) \\
&\qquad \wedge ((\underline{X}^{\text{pos}} \leqslant \text{bin}_{N+1}(2^N - 1)) \vee (\underline{X}^{\text{pos}} > \text{bin}_{N+1}(2^N - 1 + n))) \rightarrow X^{\text{read}}_{\#} \Big) \Bigg) \\
&\wedge \Bigg(\Big(B \wedge (\underline{X}^{\text{time-apv}} > \text{bin}_N(0)) \wedge (\underline{\alpha}^{\text{time}} = \underline{X}^{\text{time-apv}}) \Big) \rightarrow (\underline{X}^{\text{read}} = \underline{\alpha}^{\text{written}}) \Bigg).
\end{aligned}$$

We explain this formula. If the current cell has never before been visited (this is the case iff the vector $\underline{X}^{\text{time-apv}}$ has the binary value 0) then the vector $\underline{X}^{\text{read}}$ is forced to store in unary format the initial symbol in the current cell. This is either a symbol w_i of the input string or the blank $\#$. If the current cell has

been visited before (this is the case iff the vector $\underline{X}^{\text{time-apv}}$ has a binary value strictly greater than 0) then in the cloud corresponding to the time stored in $\underline{X}^{\text{time-apv}}$ the vector $\underline{\alpha}^{\text{written}}$ describes the symbol that has been written into the current cell during the previous visit. Therefore, this value is copied into the vector $\underline{X}^{\text{read}}$.
Next, we wish to define the formula *computation* that describes the computation steps. We have to distinguish between the two cases whether the tape head is going to move to the left or to the right. If in a computation step the symbol $\theta \in \Gamma$ is written into the current cell, if the tape head moves to the right, and if the new state after this step is the state $r \in Q$, then the following formula guarantees the existence of a point and its cloud with suitable values in the shared variables and in the persistent propositional variables.

$$
\begin{aligned}
&\mathit{compstep}_{\text{right}}(r,\theta)\\
&:= \bigwedge_{k=0}^{N-1}\bigwedge_{l=0}^{N}\Bigg(\big(B \wedge \text{rightmost\_zero}(\underline{\alpha}^{\text{time}},k) \wedge \text{rightmost\_zero}(\underline{\alpha}^{\text{pos}},l)\big)\\
&\qquad \to L\Bigg(B \wedge (\underline{X}^{\text{time}} = \underline{\alpha}^{\text{time}}, > k) \wedge \text{rightmost\_one}(\underline{X}^{\text{time}},k)\\
&\qquad\qquad \wedge (\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{pos}}, > l) \wedge \text{rightmost\_one}(\underline{X}^{\text{pos}},l)\\
&\qquad\qquad \wedge \Diamond\big((\underline{\alpha}^{\text{time}} = \underline{X}^{\text{time}}) \wedge (\underline{\alpha}^{\text{pos}} = \underline{X}^{\text{pos}}) \wedge \alpha_r^{\text{state}} \wedge \alpha_\theta^{\text{written}}\\
&\qquad\qquad\quad \wedge (\underline{\alpha}^{\text{read}} = \underline{X}^{\text{read}})\big)\Bigg)\Bigg).
\end{aligned}
$$

We explain this formula. The procedure is quite similar to the one of the formula $\text{counter}_{SSL,n}$ in Section 5.2 for a binary counter. The first three lines of the formula make sure that there is a point in the same cloud as the current point such that in this new point the binary value of the persistent variable vector $\underline{X}^{\text{time}}$ is larger by one than the binary value of the shared variable vector $\underline{\alpha}^{\text{time}}$ and that in this new point the binary value of the persistent variable vector $\underline{X}^{\text{pos}}$ is larger by one than the binary value of the shared variable vector $\underline{\alpha}^{\text{pos}}$. The last two lines ensure the existence of a $\xrightarrow{\Diamond}$-successor of this new point in which the shared variable vectors $\underline{\alpha}^{\text{time}}$, $\underline{\alpha}^{\text{pos}}$, $\underline{\alpha}^{\text{state}}$, $\underline{\alpha}^{\text{written}}$, and $\underline{\alpha}^{\text{read}}$ get the correct new values.
If in a computation step the symbol $\theta \in \Gamma$ is written into the current cell, if the tape head moves to the left, and if the new state after this step is the state $r \in Q$, then the following formula guarantees the existence of a point and its cloud with suitable values in the shared variables and in the persistent

propositional variables.

$$
\begin{aligned}
&\mathit{compstep}_{\text{left}}(r,\theta)\\
&:= \bigwedge_{k=0}^{N-1}\bigwedge_{l=0}^{N}\Bigg(\big(B \wedge \text{rightmost\_zero}(\underline{\alpha}^{\text{time}},k) \wedge \text{rightmost\_one}(\underline{\alpha}^{\text{pos}},l)\big)\\
&\quad\rightarrow L\Bigg(B \wedge (\underline{X}^{\text{time}} = \underline{\alpha}^{\text{time}}, > k) \wedge \text{rightmost\_one}(\underline{X}^{\text{time}},k)\\
&\qquad\wedge (\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{pos}}, > l) \wedge \text{rightmost\_zero}(\underline{X}^{\text{pos}},l)\\
&\qquad\wedge \Diamond\big((\underline{\alpha}^{\text{time}} = \underline{X}^{\text{time}}) \wedge (\underline{\alpha}^{\text{pos}} = \underline{X}^{\text{pos}}) \wedge \alpha_r^{\text{state}} \wedge \alpha_\theta^{\text{written}}\\
&\qquad\quad\wedge (\underline{\alpha}^{\text{read}} = \underline{X}^{\text{read}})\big)\Bigg)\Bigg).
\end{aligned}
$$

This formula is very similar to the previous one with the exception that here the binary counter for the position of the tape head is decremented.
The computation is modeled by the following subformula. Remember that Q is the disjoint union of the sets $\{q_{\text{accept}}\}$, $\{q_{\text{reject}}\}$, $Q_\forall$, $Q_\exists$.

$$
\begin{aligned}
&\mathit{computation}\\
&:= \bigwedge_{q\in Q_\forall}\bigwedge_{\eta\in\Gamma}\Bigg((\alpha_q^{\text{state}} \wedge \alpha_\eta^{\text{read}}) \rightarrow\\
&\quad\Bigg(\bigwedge_{(r,\theta,\mathit{left})\in\delta(q,\eta)} \mathit{compstep}_{\text{left}}(r,\theta) \wedge \bigwedge_{(r,\theta,\mathit{right})\in\delta(q,\eta)} \mathit{compstep}_{\text{right}}(r,\theta)\Bigg)\Bigg)\\
&\quad\wedge \bigwedge_{q\in Q_\exists}\bigwedge_{\eta\in\Gamma}\Bigg((\alpha_q^{\text{state}} \wedge \alpha_\eta^{\text{read}}) \rightarrow\\
&\quad\Bigg(\bigvee_{(r,\theta,\mathit{left})\in\delta(q,\eta)} \mathit{compstep}_{\text{left}}(r,\theta) \vee \bigvee_{(r,\theta,\mathit{right})\in\delta(q,\eta)} \mathit{compstep}_{\text{right}}(r,\theta)\Bigg)\Bigg)\Bigg).
\end{aligned}
$$

Finally, the subformula *no_reject* is defined as follows.

$$\mathit{no\_reject} := \neg\alpha_{q_{\text{reject}}}^{\text{state}}.$$

We have completed the description of the formula $f_{\text{SSL}}(w)$ for $w \in \Sigma^*$. It is clear that $f_{\text{SSL}}(w)$ is a bimodal formula, for any $w \in \Sigma^*$. We still have to show two claims:

1. The function f_{SSL} can be computed in logarithmic space.

2. For any $w \in \Sigma^*$,

$$w \in L \iff \text{the bimodal formula } f_{\text{SSL}}(w) \text{ is SSL-satisfiable.}$$

The first claim will be shown in the following section. The two directions of the second claim will be shown separately in Sections 6.3 and 6.4.

6.2 LOGSPACE Computability of the Reduction

For the first claim, we observe that there are three kinds of subformulas of $f_{\text{SSL}}(w)$:

1. subformulas that do not depend on the input string w at all,
2. subformulas that depend only on the length n of the input string w but not on its symbols $w_1, \ldots, w_n$,
3. subformulas that depend on the particular symbols $w_1, \ldots, w_n$ of the input string w.

The subformula $K\Box$*uniqueness* is of the first type. Therefore, it can be written using only a constant amount of workspace. And there is only one subformula of the third type, the subformula $K\Box$*get_the_right_symbol*. All other subformulas are of the second type. All of them contain vectors of propositional variables of length at most $N+1$ or conjunctions or disjunctions of length at most $N+1$, where $N = p(n)$. And all of these vectors and lists of conjunctions or disjunctions have a very regular structure. This applies also to the only subformula of the third type. This regular structure makes it possible to write down these formulas using a fixed (that means: independent of the input string w) number of counters that can count up to N. But such counters can be implemented in binary using not more than $O(\log N) = O(\log n)$ space. Hence, given a string w, the whole formula $f_{\text{SSL}}(w)$ can be computed using not more than logarithmic space.

6.3 Construction of a Model

We come to the second claim. First, we show the direction from left to right. Let us assume $w \in L$. We will construct a cross axiom model $(W, \xrightarrow{L}, \xrightarrow{\Diamond}, \sigma)$ with a point $p_{root,root} \in W$ such that $p_{root,root} \models f_{\text{SSL}}(w)$. There exists an accepting tree $T = (V, E, c)$ of M on input w, where V is the set of nodes of T, where $E \subseteq V \times V$ is the set of edges, and where the function $c : V \to Q \times \{0, \ldots, 2^{N+1} - 2\} \times \Gamma^{2^{N+1}-1}$ labels each node with a configuration (remember the discussion about the description of configurations at the beginning of Section 6.1). Let $\mathit{root} \in V$ be the root of T. The set W is defined to be the (disjoint) union of the following three sets P, U, and S. We define

$$P := \{p_{v,x} \ : \ v, x \in V \text{ and } vE^*x\}.$$

For the definition of U we use the following set as an index set:

$$\begin{aligned} I \ := \ & (\{\text{time}\} \times \{0, \ldots, N-1\}) \\ & \cup(\{\text{pos}\} \times \{0, \ldots, N\}) \\ & \cup(\{\text{state}\} \times Q) \\ & \cup(\{\text{written}\} \times \Gamma) \\ & \cup(\{\text{read}\} \times \Gamma). \end{aligned}$$

We define

$$U := \{u_{v,\mathit{string},z} \ : \ v \in V \cup \{\top\}, (\mathit{string}, z) \in I\}$$

where $\top$ is a special element not contained in V. We extend the binary relation E^* on V to a binary relation $\widetilde{E}$ on $V \cup \{\top\}$ by

$$\widetilde{E} := \{(u, v) \in (V \cup \{\top\}) \times (V \cup \{\top\}) \ : \ \text{ either } (u, v \in V \text{ and } uE^*v) \text{ or } v = \top\}.$$

We define the set S by

$$\begin{aligned} S \ := \ & \{s_{v,\text{time},k} \ : \ v \in V, k \in \text{Ones}(\mathit{time}(v))\} \\ & \cup\{s_{v,\text{pos},k} \ : \ v \in V, k \in \text{Ones}(\mathit{pos}(v))\} \\ & \cup\{s_{v,\text{state},q} \ : \ v \in V, q = \mathit{state}(v)\} \\ & \cup\{s_{v,\text{written},\gamma} \ : \ v \in V\backslash\{\mathit{root}\}, \gamma = \mathit{written}(v)\} \cup \{s_{\text{root},\mathit{written},\#}\} \\ & \cup\{s_{v,\text{read},\gamma} \ : \ v \in V, \gamma = \mathit{read}(v)\}. \end{aligned}$$

As the relation $\xrightarrow{L}$ is supposed to be an equivalence relation we can define it by defining the $\xrightarrow{L}$-equivalence classes. These are the sets

$$\mathit{Cloud}_v := \{p_{v,x} \in P \ : \ x \in V\} \cup \{u_{v,i} \in U \ : \ i \in I\} \cup \{s_{v,i} \in S \ : \ i \in I\}$$

for all $v \in V$, and the set

$$\mathit{Cloud}_\top := \{u_{\top,i} \in U \ : \ i \in I\}.$$

We define the relation $\overset{\Diamond}{\to}$ by:

$$\begin{aligned}
\overset{\Diamond}{\to} \;:=\; & \{(p_{v,x}, p_{v',x'}) \in P \times P \;:\; v, v', x, x' \in V \text{ and } vE^*v' \text{ and } x = x'\} \\
& \cup\{(u_{v,i}, u_{v',i'}) \in U \times U \;:\; v, v' \in V \cup \{\top\}, i, i' \in I \text{ and } v\widetilde{E}v' \\
& \qquad\qquad\qquad\qquad\qquad \text{and } i = i'\} \\
& \cup\{(u_{v,i}, s_{v',i'}) \in U \times S \;:\; v, v' \in V, i, i' \in I \text{ and } vE^*v' \text{ and } i = i'\} \\
& \cup\{(s_{v,i}, s_{v',i'}) \in S \times S \;:\; v, v' \in V, i, i' \in I \text{ and } v = v' \text{ and } i = i'\}.
\end{aligned}$$

It is straightforward to check that $\overset{\Diamond}{\to}$ is reflexive and transitive. The cross property is satisfied as well. Thus, $(W, \overset{L}{\to}, \overset{\Diamond}{\to})$ is an cross axiom frame. Finally, we define the valuation σ as follows.

$$\sigma(B) \;:=\; P,$$

and

$$\begin{aligned}
\sigma(A_k^{\text{time}}) \;&:=\; \{u_{v,\text{time},k} \in U \;:\; v \in V \cup \{\top\}\} \cup \{s_{v,\text{time},k} \in S \;:\; v \in V\}, \\
\sigma(X_k^{\text{time}}) \;&:=\; \{p_{v,x} \in P \;:\; v, x \in V \text{ and } k \in \text{Ones}(\mathit{time}(x))\}, \\
\sigma(X_k^{\text{time-apv}}) \;&:=\; \{p_{v,x} \in P \;:\; v, x \in V \text{ and } k \in \text{Ones}(j)\}
\end{aligned}$$

$$\text{where } j := \begin{cases} 0 & \text{if on the path from } \mathit{root} \text{ to } x \text{ the cell } \mathit{pos}(x) \text{ has not been visited before the cell } x \text{ is reached,} \\ 1 + \mathit{time}(v') & \text{otherwise, where } v' \text{ is the last node on the path from } \mathit{root} \text{ to } \mathit{pred}(x) \text{ with } \mathit{pos}(v') = \mathit{pos}(x), \end{cases}$$

for $k \in \{0, \ldots, N-1\}$,

$$\begin{aligned}
\sigma(A_k^{\text{pos}}) \;&:=\; \{u_{v,\text{pos},k} \in U \;:\; v \in V \cup \{\top\}\} \cup \{s_{v,\text{pos},k} \in S \;:\; v \in V\}, \\
\sigma(X_k^{\text{pos}}) \;&:=\; \{p_{v,x} \in P \;:\; v, x \in V \text{ and } k \in \text{Ones}(\mathit{pos}(x))\},
\end{aligned}$$

for $k \in \{0, \ldots, N\}$,

$$\sigma(A_q^{\text{state}}) \;:=\; \{u_{v,\text{state},q} \in U \;:\; v \in V \cup \{\top\}\} \cup \{s_{v,\text{state},q} \in S \;:\; v \in V\},$$

for $q \in Q$,

$$\begin{aligned}
\sigma(A_\gamma^{\text{written}}) \;&:=\; \{u_{v,\text{written},\gamma} \in U \;:\; v \in V \cup \{\top\}\} \cup \{s_{v,\text{written},\gamma} \in S \;:\; v \in V\}, \\
\sigma(A_\gamma^{\text{read}}) \;&:=\; \{u_{v,\text{read},\gamma} \in U \;:\; v \in V \cup \{\top\}\} \cup \{s_{v,\text{read},\gamma} \in S \;:\; v \in V\}, \\
\sigma(X_\gamma^{\text{read}}) \;&:=\; \{p_{v,x} \in P \;:\; v, x \in V \text{ and } \gamma = \mathit{read}(x))\},
\end{aligned}$$

for $\gamma \in \Gamma$. It is obvious that all propositional variables are persistent. Thus, we have defined a cross axiom model $(W, \xrightarrow{L}, \xrightarrow{\Diamond}, \sigma)$. We claim that $p_{root,root} \models f_{\mathrm{SSL}}(w)$. For an illustration of an important detail of the structure see Figure 6.1.
We start with some preliminary observations. First, for any cloud, any shared variable has the same truth values in all points in the cloud. Secondly,

$$y \models B \iff y \in P,$$

for all $y \in W$. So, the points in P are the "information" points. On the other hand, as the cloud $Cloud_\top$ does not contain any elements from P, for all points $y \in Cloud_\top$ we have $y \models K\neg B$, hence,

$$(\forall y \in Cloud_\top)\ y \models \neg\alpha_k^{string},$$

for all *string* $\in$ {times, pos, state, written, read} and all k. That means, the truth value of any shared variable in the cloud $Cloud_\top$ is false. We claim that in the other clouds all shared variables have the values indicated by their names, namely,

$$\begin{array}{rcl} y & \models & (\underline{\alpha}^{\mathrm{time}} = \mathrm{bin}_N(time(v))), \\ y & \models & (\underline{\alpha}^{\mathrm{pos}} = \mathrm{bin}_{N+1}(pos(v))), \\ (y & \models & \alpha_q^{\mathrm{state}}) \iff q = state(v),\ \text{for } q \in Q, \\ (y & \models & \alpha_\gamma^{\mathrm{read}}) \iff \gamma = read(v),\ \text{for } \gamma \in \Gamma, \end{array}$$

for $v \in V$ and $y \in Cloud_v$,

$$(y \models \alpha_\gamma^{\mathrm{written}}) \iff \gamma = written(v),$$

for $\gamma \in \Gamma$, $v \in V \backslash \{root\}$ and $y \in Cloud_v$, and

$$(y \models \alpha_\gamma^{\mathrm{written}}) \iff \gamma = \#,$$

for $\gamma \in \Gamma$ and $y \in Cloud_{root}$. This can be checked similarly as the corresponding claim (5.1) in the proof of Proposition 5.2. We prove the assertions about $\alpha_\gamma^{\mathrm{written}}$ and leave the proofs of the other assertions to the reader. Let us fix some $\gamma \in \Gamma$. Note that, for all $p' \in P$, we have $p' \models \neg A_\gamma^{\mathrm{written}}$, hence

$$(\forall p' \in P)\ p' \models (\neg A_\gamma^{\mathrm{written}} \vee \Diamond K\neg B).$$

Furthermore, $u_{\top,i} \models K\neg B$ for all $i \in I$. Since for all $v \in V \cup \{\top\}$ and $i \in I$ we have $u_{v,i} \xrightarrow{\Diamond} u_{\top,i}$ we obtain $u_{v,i} \models \Diamond K\neg B$ for all $v \in V \cup \{\top\}$ and $i \in I$. Hence,

$$(\forall u' \in U)\ u' \models (\neg A_\gamma^{\mathrm{written}} \vee \Diamond K\neg B).$$

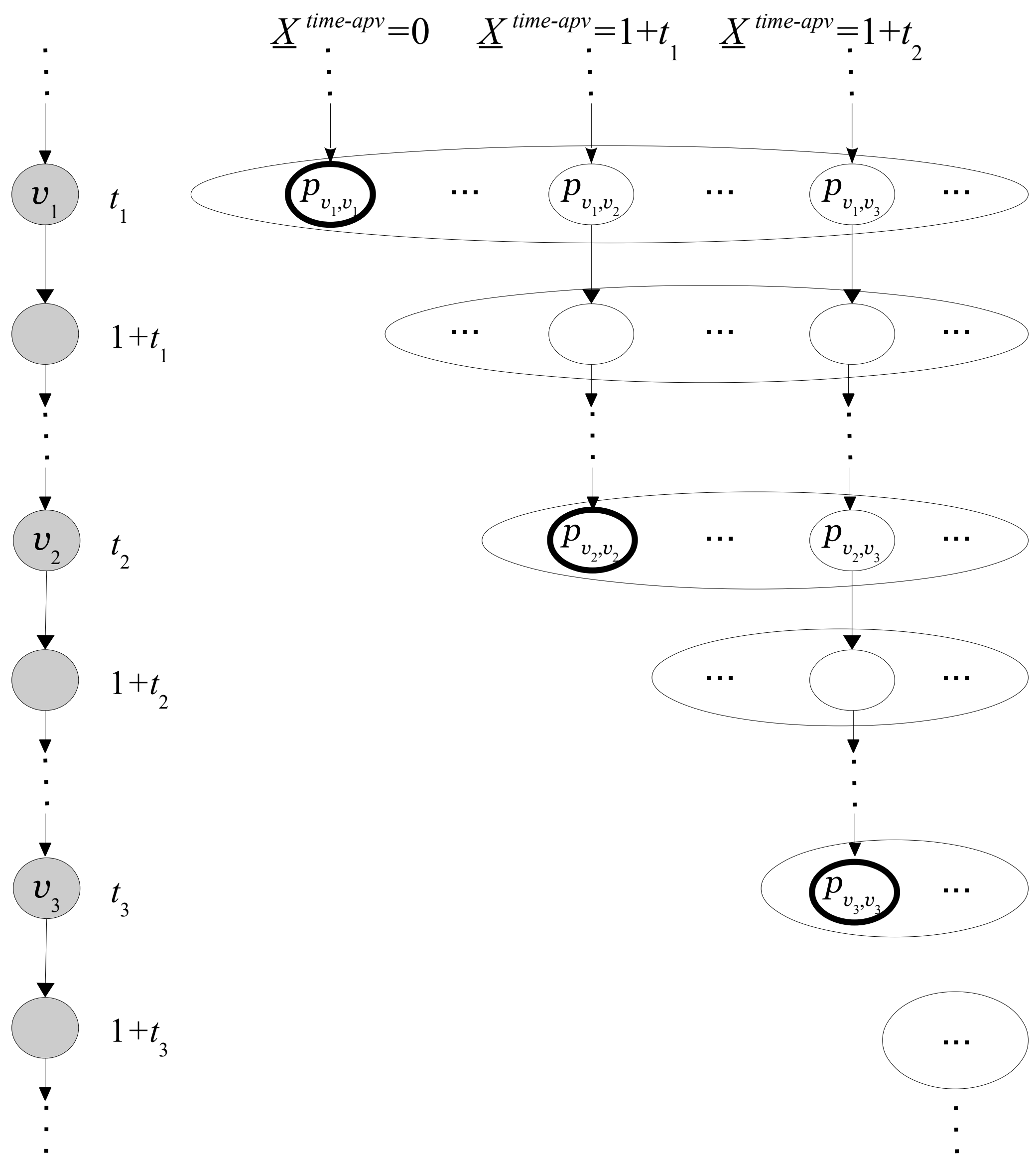


Figure 6.1: A possible detail of a cross axiom model of the formula $f_{\mathrm{SSL}}(w)$. Consider a certain cell and let us assume that v_1, v_2, v_3 are the first three computation nodes on some computation path in which this cell is visited. Let $t_i := time(v_i)$. The diagram on the left shows a part of the computation path. The diagram on the right shows the corresponding part of the cross axiom model.

This shows that, for any $v \in V$, the shared variable $\alpha_\gamma^{\text{written}} = L(A_\gamma^{\text{written}} \wedge \Box LB)$ is true in the cloud $Cloud_v$ if, and only if, there exists some $s' \in Cloud_v \cap S$ with $s' \models A_\gamma^{\text{written}} \wedge \Box LB$. As $p' \models B$ for all $p' \in P$ and any $s' \in S$ is $\xrightarrow{L}$-equivalent to some $p' \in P$, we have $s' \models LB$, for all $s' \in S$. Actually, for $s' \in S$ we even have $s' \models \Box LB$ as s' does not have any $\xrightarrow{\Diamond}$-successor besides itself. Thus, for any $v \in V$, the shared variable $\alpha_\gamma^{\text{written}} = L(A_\gamma^{\text{written}} \wedge \Box LB)$ is true in the cloud $Cloud_v$ if, and only if, there exists some $s' \in Cloud_v \cap S$ with $s' \models A_\gamma^{\text{written}}$. The elements $s' \in Cloud_v \cap S$ have the form $s' = s_{v,i}$ for some $i \in I$. On the one hand, we observe that, for $v \in V$ and $i \in I$, $s_{v,i} \models A_\gamma^{\text{written}} \iff i = (\text{written}, \gamma)$. On the other hand, for $v \in V$ we have

$$s_{v,\text{written},\gamma} \in Cloud_v \iff \begin{cases} \gamma = written(v) & \text{if } v \in V\backslash\{root\}, \\ \gamma = \# & \text{if } v = root. \end{cases}$$

Thus, we have shown the desired claims:

$$(y \models \alpha_\gamma^{\text{written}}) \iff \gamma = written(v),$$

for $\gamma \in \Gamma$, $v \in V\backslash\{root\}$ and $y \in Cloud_v$, and

$$(y \models \alpha_\gamma^{\text{written}}) \iff \gamma = \#,$$

for $\gamma \in \Gamma$ and $y \in Cloud_{root}$.

It is clear from the definition of the valuation σ that in the points in P the variable vectors $\underline{X}^{\text{time}}$, $\underline{X}^{\text{pos}}$, $\underline{X}^{\text{read}}$, and $\underline{X}^{\text{time-apv}}$ have the values indicated by their names:

$$\begin{array}{rcl} p_{v,x} & \models & (\underline{X}^{\text{time}} = \text{bin}_N(time(x)), \\ p_{v,x} & \models & (\underline{X}^{\text{pos}} = \text{bin}_{N+1}(pos(x)), \\ (p_{v,x} & \models & X_\gamma^{\text{read}}) \iff \gamma = read(x), \text{ for } \gamma \in \Gamma, \\ p_{v,x} & \models & (\underline{X}^{\text{time-apv}} = \text{bin}_N(j)), \end{array}$$

$$\text{where } j := \begin{cases} 0 & \text{if on the path from } root \text{ to } x \text{ the cell } pos(x) \text{ has not been visited before the cell } x \text{ is reached,} \\ 1 + time(v') & \text{otherwise, where } v' \text{ is the last node on the path from } root \text{ to } pred(x) \text{ with } pos(v') = pos(x), \end{cases}$$

for $v, x \in V$ satisfying vE^*x.
Now we are prepared to show $p_{root,root} \models f_{\mathrm{SSL}}(w)$. Our observations about the values of the shared variable vectors $\underline{\alpha}^{\text{state}}$, $\underline{\alpha}^{\text{read}}$, and $\underline{\alpha}^{\text{written}}$ show

$$p_{root,root} \models K\Box \mathit{uniqueness}$$

(remember that B is false in all points of the cloud $\mathit{Cloud}_\top$). Similarly, our observations about the values of the shared variable vectors $\underline{\alpha}^{\text{time}}$, $\underline{\alpha}^{\text{pos}}$, $\underline{\alpha}^{\text{state}}$, and $\underline{\alpha}^{\text{read}}$ imply

$$p_{root,root} \models \mathit{start}.$$

As the state of any node in the accepting tree T of M on input w is different from q_{reject}, our observations about the value of the vector $\underline{\alpha}^{\text{state}}$ (in any cloud Cloud_v for $v \in V$ the shared variable α_q^{state} is true if, and only if, $q = \mathit{state}(v)$, and in the cloud $\mathit{Cloud}_\top$ all shared variables are false) shows

$$p_{root,root} \models K\Box \mathit{no\_reject}.$$

Next, we show

$$p_{root,root} \models K\Box \mathit{time\_after\_previous\_visit}.$$

As the variable B is true only in the elements of P, it is sufficient to show for all $v, x \in V$ with vE^*x

$$\begin{aligned} p_{v,x} \models \quad & \Big(\underline{X}^{\text{time-apv}} \leqslant \underline{X}^{\text{time}}\Big) \\ & \wedge\Big(\big((\underline{\alpha}^{\text{time}} < \underline{X}^{\text{time}}) \wedge (\underline{\alpha}^{\text{pos}} \neq \underline{X}^{\text{pos}})\big) \to (\underline{X}^{\text{time-apv}} \neq \underline{\alpha}^{\text{time}} + 1)\Big) \\ & \wedge\Big(\big((\underline{\alpha}^{\text{time}} < \underline{X}^{\text{time}}) \wedge (\underline{\alpha}^{\text{pos}} = \underline{X}^{\text{pos}})\big) \to (\underline{\alpha}^{\text{time}} < \underline{X}^{\text{time-apv}})\Big). \end{aligned}$$

We distinguish between the two cases whether the cell $\mathit{pos}(x)$ under the tape head in the configuration $c(x)$ has been visited on the path from root to x before x is reached or not.
First let us assume that the cell $\mathit{pos}(x)$ has not been visited before. Then $p_{v,x} \models (\underline{X}^{\text{time-apv}} = \mathrm{bin}_N(0))$, hence $p_{v,x} \models (\underline{X}^{\text{time-apv}} \leqslant \underline{X}^{\text{time}})$ and $p_{v,x} \models (\underline{X}^{\text{time-apv}} \neq \underline{\alpha}^{\text{time}} + 1)$, that is, the formulas in the first two lines are true in $p_{v,x}$. And if we have $p_{v,x} \models (\underline{\alpha}^{\text{time}} < \underline{X}^{\text{time}})$ then, due to vE^*x, the point v is a point on the path from root to $\mathit{pred}(x)$. Our assumption (that the cell $\mathit{pos}(x)$ has not been visited before x is reached) implies $p_{v,x} \models \neg(\underline{\alpha}^{\text{pos}} = \underline{X}^{\text{pos}})$. Hence the formula in the third line is true in $p_{v,x}$ as well.
Now, let us assume that the cell $\mathit{pos}(x)$ has been visited on the path from root to x before x is reached. Let x' be the last node on the path from root

to $\mathit{pred}(x)$ with $\mathit{pos}(x') = \mathit{pos}(x)$. Let $i := \mathit{time}(x')$ and $j := 1 + i$. Then $i < \mathit{time}(x)$ and $j \leqslant \mathit{time}(x)$. As $p_{v,x} \models (\underline{X}^{\text{time-apv}} = \text{bin}_N(j))$, the formula in the first line above is true, that is, $p_{v,x} \models (\underline{X}^{\text{time-apv}} \leqslant \underline{X}^{\text{time}})$. For the formula in the second line, let us assume $p_{v,x} \models (\underline{\alpha}^{\text{time}} < \underline{X}^{\text{time}}) \wedge (\underline{\alpha}^{\text{pos}} \neq \underline{X}^{\text{pos}})$. Then v is a node on the path from *root* to $\mathit{pred}(x)$ with $\mathit{pos}(v) \neq \mathit{pos}(x)$. Hence, $v \neq x'$, hence, $\mathit{time}(v) \neq \mathit{time}(x')$, hence, $j \neq \mathit{time}(v) + 1$, hence, $p_{v,x} \models (\underline{X}^{\text{time-apv}} \neq \underline{\alpha}^{\text{time}} + 1)$. For the formula in the third line, let us assume $p_{v,x} \models (\underline{\alpha}^{\text{time}} < \underline{X}^{\text{time}}) \wedge (\underline{\alpha}^{\text{pos}} = \underline{X}^{\text{pos}})$. Then v is a node on the path from *root* to $\mathit{pred}(x)$ with $\mathit{pos}(v) = \mathit{pos}(x)$, that is, with the property that in this node the same cell is visited as in the node x. As x' is the last node on the path from *root* to $\mathit{pred}(x)$ with this property, we have $\mathit{time}(v) \leqslant \mathit{time}(x')$, hence, $j = 1 + i = 1 + \mathit{time}(x') > \mathit{time}(v)$, hence $p_{v,x} \models (\underline{\alpha}^{\text{time}} < \underline{X}^{\text{time-apv}})$.
Next, we show

$$p_{\mathit{root},\mathit{root}} \models K\Box \mathit{get\_the\_right\_symbol}.$$

It is sufficient to show for all $y \in W$, $y \models \mathit{get\_the\_right\_symbol}$. There are two cases to be considered. In each of them, due to the presence of the variable B, we need to consider only points $y \in P$. Let us consider elements $v, x \in V$ with vE^*x. It is sufficient to show $p_{v,x} \models \mathit{get\_the\_right\_symbol}$. We distinguish between the two cases considered in this formula. First, let us assume $p_{v,x} \models (\underline{X}^{\text{time-apv}} = \text{bin}_N(0))$. We have to show that in this case

$$\begin{aligned} p_{v,x} \models\; & \bigwedge_{i=1}^{n} \Big((\underline{X}^{\text{pos}} = \text{bin}_{N+1}(2^N - 1 + i)) \to X^{\text{read}}_{w_i} \Big) \\ & \wedge \Big(\big((\underline{X}^{\text{pos}} \leqslant \text{bin}_{N+1}(2^N - 1)) \vee (\underline{X}^{\text{pos}} > \text{bin}_{N+1}(2^N - 1 + n)) \big) \\ & \quad \to X^{\text{read}}_{\#} \Big). \end{aligned}$$

According to our observations about the value of $\underline{X}^{\text{time-apv}}$, the cell $\mathit{pos}(x)$ under the tape head in the configuration $c(x)$ has not been visited on the path from *root* to $\mathit{pred}(x)$. Thus, the symbol $\mathit{read}(x)$ is still the initial symbol in the cell $\mathit{pos}(x)$. Let us call this symbol γ. Then $p_{v,x} \models X^{\text{read}}_{\gamma}$, and

$$\gamma = \begin{cases} w_i & \text{if } \mathit{pos}(x) = 2^N - 1 + i, \text{ for some } i \in \{1, \ldots, n\}, \\ \# & \text{if } \mathit{pos}(x) \leqslant 2^N - 1 \text{ or } \mathit{pos}(x) > 2^N - 1 + n. \end{cases}$$

On the other hand, $p_{v,x} \models (\underline{X}^{\text{pos}} = \text{bin}_{N+1}(\mathit{pos}(x)))$. We have shown the assertion.
Now, let us assume

$$p_{v,x} \models ((\underline{X}^{\text{time-apv}} > \text{bin}_N(0)) \wedge (\underline{\alpha}^{\text{time}} = \underline{X}^{\text{time-apv}})).$$

We have to show that in this case

$$p_{v,x} \models (\underline{X}^{\text{read}} = \underline{\alpha}^{\text{written}}).$$

The assumption $p_{v,x} \models (\underline{X}^{\text{time-apv}} > \text{bin}_N(0))$ implies that the cell $pos(x)$ has already been visited on the path from $root$ to $pred(x)$ and that $p_{v,x} \models (\underline{X}^{\text{time-apv}} = \text{bin}_{N+1}(1+i))$ where $i = time(x')$ and x' is the last node on the path from $root$ to $pred(x)$ with $pos(x') = pos(x)$. The assumption $p_{v,x} \models (\underline{\alpha}^{\text{time}} = \underline{X}^{\text{time-apv}})$ implies $x' = pred(v)$. But then in the point v the vector $\underline{\alpha}^{\text{written}}$ encodes in unary the symbol that was written into the cell $pos(x)$ in the computation step from x' to v. If we call this symbol γ, this means $p_{v,x} \models \alpha_\gamma^{\text{written}}$. This is still the symbol in the cell $pos(x)$ when x is reached, hence, $p_{v,x} \models X_\gamma^{\text{read}}$. So, we have indeed shown

$$p_{v,x} \models (\underline{X}^{\text{read}} = \underline{\alpha}^{\text{written}}).$$

Finally, we show

$$p_{root,root} \models K\Box computation.$$

It is sufficient to show

$$y \models computation,$$

for all $y \in W$. We will separately treat the conjunctions over the set $(q,\eta) \in Q_\exists \times \Gamma$ and over the set $Q_\forall \times \Gamma$. Let us fix a pair $(q,\eta) \in Q_\exists \times \Gamma$ and let us assume that $y \in W$ is a point with $y \models (\alpha_q^{\text{state}} \wedge \alpha_\eta^{\text{read}})$. We have to show that there is an element $(r,\theta,left) \in \delta(q,\eta)$ such that $y \models compstep_{\text{left}}(r,\theta)$ or that there is an element $(r,\theta,right) \in \delta(q,\eta)$ such that $y \models compstep_{\text{right}}(r,\theta)$. As in the cloud $Cloud_\top$ the truth value of any shared variable is false, the assumption $y \models (\alpha_q^{\text{state}} \wedge \alpha_\eta^{\text{read}})$ implies that there exists some $v \in V$ with $y \in Cloud_v$. Furthermore, $q = state(v)$ and $\eta = read(v)$. As T is an accepting tree and the state q of $c(v)$ is an element of $Q_\exists$, the node v is an inner node of T, hence, it has a successor v'. Let us assume that $((q,\eta),(r,\theta,left)) \in \delta$ is the element of the transition relation δ that leads from v to v' (the case that this element is of the form $((q,\eta),(r,\theta,right))$ is treated analogously). We claim that then

$$y \models compstep_{\text{left}}(r,\theta).$$

Let us check this. Let us assume that, for some $k \in \{0,\dots,N-1\}$ and for some $l \in \{0,\dots,N\}$,

$$y \models \big(B \wedge \text{rightmost\_zero}(\underline{\alpha}^{\text{time}},k) \wedge \text{rightmost\_one}(\underline{\alpha}^{\text{pos}},l)\big).$$

The number $i := time(v)$ is an element of $\{0,\dots,2^N-2\}$ because v is an inner point of the tree T and the length of any computation path is at most

$2^N - 1$, and the number $j := \mathit{pos}(v)$ is an element of $\{1, \ldots, 2^{N+1}-3\}$ because the computation starts in cell $2^N - 1$ and because during each computation step the tape head can move at most one step to the left or to the right. We obtain $k = \min(\{0, \ldots, N-1\} \backslash \mathrm{Ones}(i))$ and $l = \min \mathrm{Ones}(j)$. We claim that the two points $p_{v,v'}$ and $p_{v',v'}$ have the properties formulated in the formula $\mathit{compstep}_{\mathrm{left}}(r, \theta)$. Indeed, we observe $y \xrightarrow{L} p_{v,v'}$ and $p_{v,v'} \xrightarrow{\Diamond} p_{v',v'}$ as well as $p_{v,v'} \models B$. The facts

$$\begin{array}{ll} \mathit{time}(v) = i, & \mathit{pos}(v) = j, \\ \mathit{time}(v') = i+1, & \mathit{pos}(v') = j-1, \end{array}$$

imply

$$\begin{array}{rcl} p_{v,v'} & \models & (\underline{\alpha}^{\mathrm{time}} = \mathrm{bin}_N(i)) \wedge (\underline{\alpha}^{\mathrm{pos}} = \mathrm{bin}_{N+1}(j)) \wedge \\ & & (\underline{X}^{\mathrm{time}} = \mathrm{bin}_N(i+1)) \wedge (\underline{X}^{\mathrm{pos}} = \mathrm{bin}_{N+1}(j-1)) \quad \text{and} \\ p_{v',v'} & \models & (\underline{\alpha}^{\mathrm{time}} = \mathrm{bin}_N(i+1)) \wedge (\underline{\alpha}^{\mathrm{pos}} = \mathrm{bin}_{N+1}(j-1)) \wedge \\ & & (\underline{X}^{\mathrm{time}} = \mathrm{bin}_N(i+1)) \wedge (\underline{X}^{\mathrm{pos}} = \mathrm{bin}_{N+1}(j-1)). \end{array}$$

We obtain

$$p_{v',v'} \models (\underline{\alpha}^{\mathrm{time}} = \underline{X}^{\mathrm{time}}) \wedge (\underline{\alpha}^{\mathrm{pos}} = \underline{X}^{\mathrm{pos}}),$$

and with

$$k = \min(\{0, \ldots, N-1\} \backslash \mathrm{Ones}(i)) \text{ and } l = \min(\mathrm{Ones}(j))$$

we obtain as well

$$\begin{array}{rcl} p_{v,v'} & \models & (\underline{X}^{\mathrm{time}} = \underline{\alpha}^{\mathrm{time}}, > k) \wedge \mathrm{rightmost\_one}(\underline{X}^{\mathrm{time}}, k) \\ & & \wedge (\underline{X}^{\mathrm{pos}} = \underline{\alpha}^{\mathrm{pos}}, > l) \wedge \mathrm{rightmost\_zero}(\underline{X}^{\mathrm{pos}}, l). \end{array}$$

Finally, our observations about the values of the shared variable vectors $\underline{\alpha}^{\mathrm{state}}$, $\underline{\alpha}^{\mathrm{written}}$, $\underline{\alpha}^{\mathrm{read}}$ and about the vector $\underline{X}^{\mathrm{read}}$ imply that also

$$p_{v',v'} \models \alpha_r^{\mathrm{state}} \wedge \alpha_\theta^{\mathrm{written}} \wedge (\underline{\alpha}^{\mathrm{read}} = \underline{X}^{\mathrm{read}}).$$

(remember that $((q, \eta), (r, \theta, \mathit{left})) \in \delta$ is the element of the transition relation δ that leads from the node v to the node v'). This ends the treatment of the conjunctions over the set $(q, \eta) \in Q_\exists \times \Gamma$ in the formula *computation*. Let us now consider a pair $(q, \eta) \in Q_\forall \times \Gamma$. Let us assume that $y \in W$ is a point such that $y \models (\alpha_q^{\mathrm{state}} \wedge \alpha_\eta^{\mathrm{read}})$. We have to show that for all elements $(r, \theta, \mathit{left}) \in \delta(q, \eta)$ we have $y \models \mathit{compstep}_{\mathrm{left}}(r, \theta)$ and for all elements $(r, \theta, \mathit{right}) \in \delta(q, \eta)$ we have $y \models \mathit{compstep}_{\mathrm{right}}(r, \theta)$. Let us consider an arbitrary element $(r, \theta, \mathit{left}) \in \delta(q, \eta)$ (the case of an element $(r, \theta, \mathit{right}) \in \delta(q, \eta)$

is treated analogously). As in the cloud $\mathit{Cloud}_\top$ the truth value of any shared variable is false, the assumption $y \models (\alpha_q^{\text{state}} \wedge \alpha_\eta^{\text{read}})$ implies that there exists some $v \in V$ with $y \in \mathit{Cloud}_v$. Furthermore, $q = \mathit{state}(v)$ and $\eta = \mathit{read}(v)$. As $q \in Q_\forall$ and T is an accepting tree, in T there is a successor v' of v such that the element $((q, \eta), (r, \theta, \mathit{left}))$ leads from v to v'. Above, we have already seen that this implies

$$y \models \mathit{compstep}_{\text{left}}(r, \theta).$$

Thus, we have shown $y \models \mathit{computation}$ for all $y \in W$. This ends the proof of the claim $p_{\mathit{root},\mathit{root}} \models f_{\text{SSL}}(w)$.

6.4 Existence of an Accepting Tree

We come to the other direction. Let $w \in \Sigma^*$. We wish to show that if $f_{\text{SSL}}(w)$ is SSL-satisfiable then $w \in L$. We will show that any cross axiom model of $f_{\text{SSL}}(w)$ essentially contains an accepting tree of the Alternating Turing Machine M on input w. Of course, this implies $w \in L$.
Let us sketch the main idea. We will consider a cross axiom model of $f_{\text{SSL}}(w)$. And we will consider partial trees of M on input w as considered in Section 5.3, for any $w \in \Sigma^*$. First, we will show that a certain very simple partial tree of M on input w "can be mapped to" the model (later we will give a precise meaning to "can be mapped to"). Then we will show that any partial tree of M on input w that can be mapped to the model and that is not an accepting tree of M on input w can be properly extended to a strictly larger partial tree of M on input w that can be mapped to the model as well. If there would not exist an accepting tree of M on input w then we would obtain an infinite strictly increasing sequence of partial trees of M on input w. But we show that this cannot happen by giving a finite upper bound on the size of partial trees of M on input w.
Let $w \in \Sigma^*$ be a string such that the formula $f_{\text{SSL}}(w)$ is SSL-satisfiable. We set $n := |w|$. Let $(W, \xrightarrow{\Diamond}, \xrightarrow{L}, \sigma)$ be a cross axiom model and $r_0 \in W$ a point such that $r_0 \models f_{\text{SSL}}(w)$. The quintuple

$$\mathit{Model} := (W, \xrightarrow{\Diamond}, \xrightarrow{L}, \sigma, r_0)$$

will be important in the following. Points in W that cannot be reached from r_0 by finitely many $\xrightarrow{\Diamond}$ and $\xrightarrow{L}$-steps (in any order) can be deleted from W with no harm: the resulting smaller quintuple will still be a model of $f_{\text{SSL}}(w)$. Hence, we will assume without loss of generality that every point $x \in W$ can be reached from r_0 by finitely many $\xrightarrow{\Diamond}$ and $\xrightarrow{L}$-steps (in any order). Note that now the cross property implies that for any $x \in W$ there exists some

$x' \in W$ with $r_0 \xrightarrow{L} x'$ and $x' \xrightarrow{\Diamond} x$. Hence, if φ is a formula with $r_0 \models K\Box\varphi$ then, for all $x \in W$, we have $x \models \varphi$. For every $x \in W$, let $Cloud_x$ be the $\xrightarrow{L}$-equivalence class of x. Remember that for every $\xrightarrow{L}$-equivalence class and every shared variable α_i^{string} for $string \in \{\text{time}, \text{pos}, \text{state}, \text{written}, \text{read}\}$ and a natural number i, the truth value of this shared variable is the same in all elements of the $\xrightarrow{L}$-equivalence class.
Partial trees of M on input w as introduced in Section 5.3 will play an important role in the following. We will write a partial tree of M on input w similarly as in Section 5.3 as a triple $T = (V, E, c)$, but with the difference that we will describe configurations as at the beginning of this section: the labeling function c will be a function of the form $c : V \to Q \times \{0, \ldots, 2^{N+1} - 2\} \times \Gamma^{2^{N+1}-1}$. If $T = (V, E, c)$ is a partial tree of M on input w with root *root* then a function $\pi : V \to W$ is called a *morphism from T to Model* if it satisfies the following four conditions:

1. $\pi(root) = r_0$,
2. $(\forall v, v' \in V)$ (if vEv' then $Cloud_{\pi(v)} \overset{\xrightarrow{L}}{\xrightarrow{\Diamond}} Cloud_{\pi(v')}$),
3. $(\forall v \in V \backslash \{root\})\, \pi(v) \models \alpha^{\text{written}}_{written(v)}$,
4. $(\forall v \in V)\, \pi(v) \models \big(B \wedge (\underline{\alpha}^{\text{time}} = \text{bin}_N(time(v)))$
$\wedge(\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(pos(v))) \wedge \alpha^{\text{state}}_{state(v)} \wedge \alpha^{\text{read}}_{read(v)}\big)$.

We say that T *can be mapped to Model* if there exists a morphism from T to *Model*. Below we shall prove the following lemma.

Lemma 6.2. *If a partial tree $T = (V, E, c)$ of M on input w can be mapped to Model and is not an accepting tree of M on input w then there exists a partial tree $T = (\widetilde{V}, \widetilde{E}, \widetilde{c})$ of M on input w that can be mapped to Model and that satisfies $V \subsetneq \widetilde{V}$.*

Before we prove this lemma, we deduce the desired assertion from it. Let

$$D := \max(\{|\delta(q, \eta)| \ : \ q \in Q, \eta \in \Gamma\}).$$

Then, due to Condition III in the definition of a "partial tree of M on input w", any node in any partial tree of M on input w has at most D successors. As any computation of M on w stops after at most $2^N - 1$ steps, any partial tree of M on input w has at most

$$\widetilde{D} := (D^{2^N} - 1)/(D - 1)$$

nodes. We claim that the rooted and labeled tree

$$T_0 := (\{root\}, \emptyset, c) \text{ where } c(root) := (q_0, 2^N - 1, \#^{2^N} w \#^{2^N - 1 - n})$$

is a partial tree of M on input w and can be mapped to *Model*. Indeed, Condition I in the definition of a "partial tree of M on input w" is satisfied because the node *root* is labeled with the initial configuration of M on input w. Conditions II, III, and IV are satisfied because T_0 does not have any inner nodes. Condition V' is satisfied because $q_0 \neq q_{\text{reject}}$, and this follows from $r_0 \models \alpha^{\text{state}}_{q_0}$ (this is a part of $r_0 \models start$) and $r_0 \models \neg\alpha^{\text{state}}_{q_{\text{reject}}}$ (this follows from $r_0 \models K\Box no\_reject$). Thus, T_0 is indeed a partial tree of M on input w. Now we show that T_0 can be mapped *Model*. Of course, we define $\pi : \{root\} \to W$ by $\pi(root) := r_0$. We claim that π is a morphism from T to *Model*. We check the four conditions one by one.

1. The condition $\pi(root) = r_0$ is true by definition.

2. The second condition is satisfied trivially because the tree T_0 does not have any edges.

3. The third condition in the definition of a "morphism from T to *Model*" is satisfied trivially because T_0 has only one node, its root.

4. On the one hand, we have $time(root) = 0$, $pos(root) = 2^N - 1$, $state(root) = q_0$, and $read(root) = \#$.
   On the other hand, the condition $r_0 \models start$ implies

$$r_0 \models B \wedge (\underline{\alpha}^{\text{time}} = \text{bin}_N(0)) \wedge (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(2^N - 1)) \wedge \alpha^{\text{state}}_{q_0} \wedge \alpha^{\text{read}}_{\#}.$$

Thus, we have shown that T_0 is a partial tree of M on input w and that T_0 can be mapped to *Model*.
If there would not exist an accepting tree of M on input w then, starting with T_0 and using Lemma 6.2 we could construct an infinite sequence of partial trees $T_0, T_1, T_2, \ldots$ of M on input w that can be mapped to *Model* such that the number of nodes in these trees is strictly increasing. But we have seen that any partial tree of M on input w can have at most $\widetilde{D}$ nodes. Thus, there exists an accepting tree of M on input w. We have shown $w \in L$.

In order to complete the proof of Theorem 6.1 it remains to prove Lemma 6.2.

Proof of Lemma 6.2. Let $T = (V, E, c)$ be a partial tree of M on input w that is not an accepting tree of M on input w and that can be mapped to *Model*. Let $\pi : V \to W$ be a morphism from T to *Model*. Then T has a

leaf $\widehat{v}$ such that the state $q := \mathit{state}(\widehat{v})$ is either an element of $Q_\exists$ or of $Q_\forall$. First we treat the case that it is an element of $Q_\exists$, then the case that it is an element of $Q_\forall$.
So, let us assume that $q \in Q_\exists$. We define $\eta := \mathit{read}(\widehat{v})$. Then, because π is a morphism from T to *Model*, we have

$$\pi(\widehat{v}) \models (\alpha_q^{\text{state}} \wedge \alpha_\eta^{\text{read}}),$$

hence, due to $\pi(\widehat{v}) \models \mathit{computation}$,

$$\pi(\widehat{v}) \models \bigvee_{(r,\theta,\mathit{left})\in\delta(q,\eta)} \mathit{compstep}_{\text{left}}(r,\theta) \vee \bigvee_{(r,\theta,\mathit{right})\in\delta(q,\eta)} \mathit{compstep}_{\text{right}}(r,\theta).$$

Let us assume that there is an element $(r,\theta,\mathit{left}) \in \delta(q,\eta)$ such that $\pi(\widehat{v}) \models \mathit{compstep}_{\text{left}}(r,\theta)$ (the other case, when there is an element $(r,\theta,\mathit{right}) \in \delta(q,\eta)$ such that $\pi(\widehat{v}) \models \mathit{compstep}_{\text{right}}(r,\theta)$, is treated analogously). We claim that we can define the new tree $\widetilde{T} = (\widetilde{V}, \widetilde{E}, \widetilde{c})$ as follows:

- $\widetilde{V} := V \cup \{\widetilde{v}\}$ where $\widetilde{v}$ is a new element (not in V),
- $\widetilde{E} := E \cup \{(\widehat{v}, \widetilde{v})\}$,
- $\widetilde{c}(x) := \begin{cases} c(x) & \text{for all } x \in V, \\ c' & \text{for } x = \widetilde{v}, \\ & \text{where } c' \text{ is the configuration that is reached from} \\ & c(\widehat{v}) \text{ in the computation step given by} \\ & ((q,\eta),(r,\theta,\mathit{left})) \in \delta. \end{cases}$

Before we show that $\widetilde{T}$ is a partial tree of M on input w, we define a function $\widetilde{\pi} : \widetilde{V} \to W$ that we will show to be a morphism from $\widetilde{T}$ to *Model*.
As $\widehat{v}$ is an element of a partial tree of M on input w with $\mathit{state}(\widehat{v}) \in Q_\exists$, at least one more computation step can be done. As any computation of M on input w stops after at most $2^N - 1$ steps, we observe that the number $i := \mathit{time}(\widehat{v})$ satisfies $0 \leqslant i < 2^N - 1$. Then $\{0, \ldots, N-1\} \backslash \text{Ones}(i) \neq \emptyset$. We set $k := \min(\{0, \ldots, N-1\} \backslash \text{Ones}(i))$. The assumption that π is a morphism from T to *Model* implies $\pi(\widehat{v}) \models (\underline{\alpha}^{\text{time}} = \text{bin}_N(i))$. We conclude $\pi(\widehat{v}) \models \text{rightmost\_zero}(\underline{\alpha}^{\text{time}}, k)$. As during each computation step, the tape head can move at most one step to the left or to the right and as the computation started in position $2^N - 1$ the number $j := \mathit{pos}(\widehat{v})$ satisfies $0 < j \leqslant 2^{N+1} - 3$. Then $\text{Ones}(j) \neq \emptyset$. We set $l := \min \text{Ones}(j)$. The assumption that π is a morphism from T to *Model* implies $\pi(\widehat{v}) \models (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(j))$. We conclude

$\pi(\widehat{v}) \models \text{rightmost\_one}(\underline{\alpha}^{\text{pos}}, l)$. Furthermore, as π is a morphism from T to *Model* we have $\pi(\widehat{v}) \models B$. Summarizing this, we have

$$\pi(\widehat{v}) \models \big(B \wedge \text{rightmost\_zero}(\underline{\alpha}^{\text{time}}, k) \wedge \text{rightmost\_one}(\underline{\alpha}^{\text{pos}}, l)\big).$$

Due to $\pi(\widehat{v}) \models \mathit{compstep}_{\text{left}}(r, \theta)$ there exist an element $x \in \mathit{Cloud}_{\pi(\widehat{v})}$ and an element $y \in W$ such that $x \overset{\Diamond}{\to} y$ as well as

$$\begin{aligned} x \;\models\; & B \wedge (\underline{X}^{\text{time}} = \underline{\alpha}^{\text{time}}, > k) \wedge \text{rightmost\_one}(\underline{X}^{\text{time}}, k) \\ & \wedge (\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{pos}}, > l) \wedge \text{rightmost\_zero}(\underline{X}^{\text{pos}}, l) \end{aligned}$$

and

$$y \models (\underline{\alpha}^{\text{time}} = \underline{X}^{\text{time}}) \wedge (\underline{\alpha}^{\text{pos}} = \underline{X}^{\text{pos}}) \wedge \alpha_r^{\text{state}} \wedge \alpha_\theta^{\text{written}} \wedge (\underline{\alpha}^{\text{read}} = \underline{X}^{\text{read}}).$$

We claim that we can define the desired function $\widetilde{\pi} : \widetilde{V} \to W$ by

$$\widetilde{\pi}(v) := \begin{cases} \pi(v) & \text{if } v \in V, \\ y & \text{if } v = \widetilde{v}. \end{cases}$$

We have to show that $\widetilde{T}$ is a partial tree of M on input w and that $\widetilde{\pi}$ is a morphism from $\widetilde{T}$ to *Model*. Condition I in the definition of a "partial tree of M on input w" is satisfied because $\widetilde{T}$ has the same root as T, and the label of the root does not change. A node in $\widetilde{T}$ is an internal node of $\widetilde{T}$ if, and only if, it is either an internal node of T or equal to $\widehat{v}$. For internal nodes of T Conditions II, III, and IV are satisfied by assumption (and due to the fact that the labels of nodes in V do not change). The new internal node $\widehat{v}$ satisfies Condition II by our definition of $\widetilde{c}(\widetilde{v})$. Condition III is satisfied for $\widehat{v}$ because $\widehat{v}$ has exactly one successor. And Condition IV does not apply to $\widehat{v}$ because $\widehat{v} \in Q_\exists$. A node in $\widetilde{T}$ is a leaf if, and only if, it is either equal to $\widetilde{v}$ or a leaf in T different from $\widehat{v}$. For the leaves in T different from $\widehat{v}$ Condition V$'$ is satisfied by assumption (and due to the fact that the labels of nodes in V do not change). For the new leaf $\widetilde{v}$ in $\widetilde{T}$ Condition V$'$ says $\mathit{state}(\widetilde{v}) \neq q_{\text{reject}}$. This is true because on the one hand $\mathit{state}(\widetilde{v}) = r$ and on the other hand $y \models \alpha_r^{\text{state}}$ and $y \models \neg\alpha_{q_{\text{reject}}}^{\text{state}}$ (this follows from $r \models K\Box \mathit{no\_reject}$). We have shown that $\widetilde{T}$ is a partial tree of M on input w.

Now we show that $\widetilde{\pi}$ is a morphism from $\widetilde{T}$ to *Model*. The first condition in the definition of a "morphism from $\widetilde{T}$ to *Model*" is satisfied because $\widetilde{\pi}(\mathit{root}) = \pi(\mathit{root}) = r_0$. Let us look at the second condition and let us assume that $v, v' \in \widetilde{V}$ satisfy $v\widetilde{E}v'$. We distinguish between two different cases for v and v'.

1. Case: $v' \in V$. Then our assumption $v\widetilde{E}v'$ implies $v \in V$ and vEv'. In this case the facts $\widetilde{\pi}(v) = \pi(v)$ and $\widetilde{\pi}(v') = \pi(v')$ as well as the assumption that $\pi : V \to W$ is a morphism from T to *Model* imply the desired assertion:
$$\mathit{Cloud}_{\widetilde{\pi}(v)} = \mathit{Cloud}_{\pi(v)} \overset{L}{\xrightarrow{\Diamond}} \mathit{Cloud}_{\pi(v')}) = \mathit{Cloud}_{\widetilde{\pi}(v')}.$$

2. Case: $v' = \widetilde{v}$. Then our assumption $v\widetilde{E}v'$ implies $v = \widehat{v}$. On the one hand, we have $x \in \mathit{Cloud}_{\pi(\widehat{v})} = \mathit{Cloud}_{\widetilde{\pi}(\widehat{v})}$, on the other hand $y = \widetilde{\pi}(\widetilde{v})$, hence, $y \in \mathit{Cloud}_{\widetilde{\pi}(\widetilde{v})}$. With $x \xrightarrow{\Diamond} y$ we obtain the desired assertion $\mathit{Cloud}_{\widetilde{\pi}(\widehat{v})} \overset{L}{\xrightarrow{\Diamond}} \mathit{Cloud}_{\widetilde{\pi}(\widetilde{v})}$.

The third condition in the definition of a "morphism from $\widetilde{T}$ to *Model*" is satisfied for $v \in V\backslash\{\mathit{root}\}$ by assumption (and by $\widetilde{\pi}(v) = \pi(v)$ and $\widetilde{c}(v) = c(v)$). It is satisfied for $\widetilde{v}$ because $\mathit{written}(\widetilde{v}) = \theta$, because $\widetilde{\pi}(\widetilde{v}) = y$, and because $y \models \alpha_\theta^{\text{written}}$. We come to the fourth condition. It is satisfied for $v \in V\backslash\{\mathit{root}\}$ by assumption (and due to $\widetilde{\pi}(v) = \pi(v)$ and $\widetilde{c}(v) = c(v)$). We still need to show that it is satisfied for $\widetilde{v}$. Remember $\widetilde{\pi}(\widetilde{v}) = y$. We need to show

$$\begin{aligned} y \ \models\ & \big(B \wedge (\underline{\alpha}^{\text{time}} = \text{bin}_N(\mathit{time}(\widetilde{v}))) \wedge (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(\mathit{pos}(\widetilde{v}))) \\ & \wedge \alpha^{\text{state}}_{\mathit{state}(\widetilde{v})} \wedge \alpha^{\text{read}}_{\mathit{read}(\widetilde{v})}\big). \end{aligned}$$

This assertion consists really of five assertions. We treat them one by one.

- The condition $y \models B$ is satisfied because $x \models B$ and $x \xrightarrow{\Diamond} y$ and because B is persistent.

- In the trees T and $\widetilde{T}$ we have $\mathit{time}(\widehat{v}) = i$, and in the tree $\widetilde{T}$ we have $\mathit{time}(\widetilde{v}) = i + 1$. We wish to show $y \models (\underline{\alpha}^{\text{time}} = \text{bin}_N(i + 1))$. We have already seen $\pi(\widehat{v}) \models (\underline{\alpha}^{\text{time}} = \text{bin}_N(i))$ and $\pi(\widehat{v}) \models$ rightmost_zero$(\underline{\alpha}^{\text{time}}, k)$. As $x \in \mathit{Cloud}_{\pi(\widehat{v})}$, we obtain
$$x \models ((\underline{\alpha}^{\text{time}} = \text{bin}_N(i)) \wedge \text{rightmost\_zero}(\underline{\alpha}^{\text{time}}, k)).$$
The conditions $x \xrightarrow{\Diamond} y$ as well as
$$\begin{aligned} x \ &\models\ (\underline{X}^{\text{time}} = \underline{\alpha}^{\text{time}}, > k) \wedge \text{rightmost\_one}(\underline{X}^{\text{time}}, k) \text{ and} \\ y \ &\models\ (\underline{\alpha}^{\text{time}} = \underline{X}^{\text{time}}) \end{aligned}$$
imply $y \models (\underline{\alpha}^{\text{time}} = \text{bin}_N(i + 1))$.

- In the trees T and $\widetilde{T}$ we have $pos(\widehat{v}) = j$, and in the tree $\widetilde{T}$ we have $pos(\widetilde{v}) = j - 1$. We wish to show $y \models (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(j - 1))$. We have already seen $\pi(\widehat{v}) \models (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(j))$ and $\pi(\widehat{v}) \models \text{rightmost\_one}(\underline{\alpha}^{\text{pos}}, l)$. As $x \in \mathit{Cloud}_{\pi(\widehat{v})}$, we obtain

  $$x \models ((\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(j)) \wedge \text{rightmost\_one}(\underline{\alpha}^{\text{pos}}, l)).$$

  The conditions $x \overset{\Diamond}{\to} y$ as well as

  $$\begin{array}{lcl} x & \models & (\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{pos}}, > l) \wedge \text{rightmost\_zero}(\underline{X}^{\text{pos}}, l) \text{ and} \\ y & \models & (\underline{\alpha}^{\text{pos}} = \underline{X}^{\text{pos}}) \end{array}$$

  imply $y \models (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(j - 1))$.

- We have $state(\widetilde{v}) = r$. And we have $y \models \alpha_r^{state}$.

- Let $\gamma := read(\widetilde{v})$ in $\widetilde{T}$. We wish to show $y \models \alpha_\gamma^{\text{read}}$. As we know $y \models (\underline{\alpha}^{\text{read}} = \underline{X}^{\text{read}})$ it is sufficient to show $y \models X_\gamma^{\text{read}}$. We have already seen $y \models (\underline{\alpha}^{\text{time}} = \text{bin}_N(i + 1))$ and $y \models (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(j - 1))$. As we know $y \models (\underline{\alpha}^{\text{time}} = \underline{X}^{\text{time}}) \wedge (\underline{\alpha}^{\text{pos}} = \underline{X}^{\text{pos}})$ we conclude $y \models ((\underline{X}^{\text{time}} = \text{bin}_N(i + 1)) \wedge (\underline{X}^{\text{pos}} = \text{bin}_{N+1}(j - 1)))$. We have seen $y \models B$ as well. Furthermore, we have $y \models \mathit{time\_after\_previous\_visit}$. The first line in the formula $\mathit{time\_after\_previous\_visit}$ implies $y \models (\underline{X}^{\text{time-apv}} \leqslant \underline{X}^{\text{time}})$, hence, $y \models (\underline{X}^{\text{time-apv}} \leqslant \text{bin}_N(i + 1))$. Let $v_m \in V$ for $m = 0, \ldots, i + 1$ be the uniquely determined node on the path from $root$ to $\widetilde{v}$ with $time(v_m) = m$ (hence $v_0 = root$, $v_i = \widehat{v}$, and $v_{i+1} = \widetilde{v}$). Then

  $$v_0 \widetilde{E} v_1 \widetilde{E} \ldots \widetilde{E} v_i \widetilde{E} v_{i+1}.$$

  By the second condition in the definition of a "morphism from $\widetilde{T}$ to *Model*"

  $$\mathit{Cloud}_{\widetilde{\pi}(v_0)} \overset{\overset{L}{\to}}{\overset{\Diamond}{\to}} \mathit{Cloud}_{\widetilde{\pi}(v_1)} \overset{\overset{L}{\to}}{\overset{\Diamond}{\to}} \ldots \overset{\overset{L}{\to}}{\overset{\Diamond}{\to}} \mathit{Cloud}_{\widetilde{\pi}(v_i)} \overset{\overset{L}{\to}}{\overset{\Diamond}{\to}} \mathit{Cloud}_{\widetilde{\pi}(v_{i+1})}.$$

  By repeated application of the cross property and by starting with $z_{i+1} := y$ we conclude that for $m = i + 1, i, \ldots, 1, 0$ there exists some $z_m \in \mathit{Cloud}_{\widetilde{\pi}(v_m)}$ with $z_m \overset{\Diamond}{\to} y$. Let us consider $m \in \{0, 1, \ldots, i + 1\}$. As $y \models B$ we also have $z_m \models B$ (remember that variables are persistent in SSL). Due to $\widetilde{\pi}(v_m) \models ((\underline{\alpha}^{\text{time}} = \text{bin}_N(m)) \wedge (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(pos(v_m))))$ we obtain $z_m \models ((\underline{\alpha}^{\text{time}} = \text{bin}_N(m)) \wedge (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(pos(v_m))))$ as well. Due to $y \models ((\underline{X}^{\text{time}} = \text{bin}_N(i + 1)) \wedge (\underline{X}^{\text{pos}} = \text{bin}_{N+1}(j - 1)))$ and the persistence of variables we obtain $z_m \models ((\underline{X}^{\text{time}} = \text{bin}_N(i +$

1)) $\wedge$ ($\underline{X}^{\text{pos}} = \text{bin}_{N+1}(j-1)$)) as well. Furthermore, we have $z_m \models$ *time_after_previous_visit*. We distinguish between the two cases whether the cell $j-1$ has been visited on the path from *root* to $\widehat{v}$ or not.

Let us first consider the case when the cell $j-1$ has not been visited on the path from *root* to $\widehat{v}$. Then, on the one hand, the symbol $\gamma = \mathit{read}(\widetilde{v})$ is still the initial symbol in the cell $j-1$. On the other hand, for all $m \in \{0,\ldots,i\}$ we have $\mathit{pos}(v_m) \neq j-1$ and

$$z_m \models (\underline{\alpha}^{\text{time}} < \underline{X}^{\text{time}} \wedge \underline{\alpha}^{\text{pos}} \neq \underline{X}^{\text{pos}}) \rightarrow \underline{X}^{\text{time-apv}} \neq \underline{\alpha}^{\text{time}} + 1).$$

Together with $z_m \models ((\underline{X}^{\text{time}} = \text{bin}_N(i+1)) \wedge (\underline{X}^{\text{pos}} = \text{bin}_{N+1}(j-1)))$ and $z_m \models ((\underline{\alpha}^{\text{time}} = \text{bin}_N(m)) \wedge (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(\mathit{pos}(v_m))))$ we conclude that $z_m \models (\underline{X}^{\text{time-apv}} \neq \text{bin}_N(m+1))$ for $m \in \{0,\ldots,i\}$. The persistence of $\underline{X}^{\text{time-apv}}$ implies that $y \models (\underline{X}^{\text{time-apv}} \neq \text{bin}_N(m+1))$ for $m \in \{0,\ldots,i\}$. Together with $y \models (\underline{X}^{\text{time-apv}} \leqslant \underline{X}^{\text{time}})$ we conclude that the binary value of $\underline{X}^{\text{time-apv}}$ in y must be 0. Now the fact $y \models$ *get_the_right_symbol* implies $y \models X^{\text{read}}_{\gamma}$.

Let us consider the second case, the case when the cell $j-1$ has been visited on the path from *root* to $\widehat{v}$. Let $v_{m'}$ be the last node on this path with $\mathit{pos}(v_{m'}) = j-1$. Then $0 \leqslant m' \leqslant i$. On the one hand, then the symbol $\gamma = \mathit{read}(\widetilde{v})$ is the symbol that was written into the cell $j-1$ in the computation step from node $v_{m'}$ to node $v_{m'+1}$, and by the third condition in the definition of a "morphism from $\widetilde{T}$ to *Model*" we have $\widetilde{\pi}(v_{m'+1}) \models \alpha^{\text{written}}_{\gamma}$, hence, $z_{m'+1} \models \alpha^{\text{written}}_{\gamma}$. On the other hand, from $y \models$ *time_after_previous_visit* we get $y \models (\underline{X}^{\text{time-apv}} \leqslant \underline{X}^{\text{time}})$, hence, $y \models (\underline{X}^{\text{time-apv}} \leqslant \text{bin}_N(i+1))$. From $z_{m'} \models$ *time_after_previous_visit* we get

$$z_{m'} \models \big((\underline{\alpha}^{\text{time}} < \underline{X}^{\text{time}}) \wedge (\underline{\alpha}^{\text{pos}} = \underline{X}^{\text{pos}})\big) \rightarrow (\underline{\alpha}^{\text{time}} < \underline{X}^{\text{time-apv}}).$$

Together with $z_{m'} \models (\underline{\alpha}^{\text{time}} = \text{bin}_N(m'))$ we obtain $z_{m'} \models (\text{bin}_N(m') < \underline{X}^{\text{time-apv}})$. And similarly as in the first case, from

$$z_m \models \big((\underline{\alpha}^{\text{time}} < \underline{X}^{\text{time}}) \wedge (\underline{\alpha}^{\text{pos}} \neq \underline{X}^{\text{pos}})\big) \rightarrow (\underline{X}^{\text{time-apv}} \neq \underline{\alpha}^{\text{time}} + 1),$$

for $m = m'+1,\ldots,i$ we obtain

$$z_m \models (\underline{X}^{\text{time-apv}} \neq \text{bin}_N(m+1)).$$

All this implies $y \models (\underline{X}^{\text{time-apv}} = \text{bin}_N(1+m'))$ and $z_{m'+1} \models (\underline{X}^{\text{time-apv}} = \text{bin}_N(1+m'))$. Then $z_{m'+1} \models$ *get_the_right_symbol* implies $z_{m'+1} \models (\underline{X}^{\text{read}} = \underline{\alpha}^{\text{written}})$. With $z_{m'+1} \models \alpha^{\text{written}}_{\gamma}$ we conclude $z_{m'+1} \models X^{\text{read}}_{\gamma}$, hence, $y \models X^{\text{read}}_{\gamma}$. That was to be shown.

Thus, $\widetilde{T}$ is not only a partial tree of M on input w but can also be mapped to *Model*. This ends the treatment of the case $q \in Q_\exists$.
Now we consider the other case, the case $q \in Q_\forall$. We define $\eta := \mathit{read}(\widehat{v})$. Let

$$(r_1, \theta_1, \mathit{dir}_1), \ldots, (r_d, \theta_d, \mathit{dir}_d)$$

be the elements of $\delta(q,\eta)$ where $d \geqslant 1$ and $\mathit{dir}_m \in \{\mathit{left}, \mathit{right}\}$, for $m = 1, \ldots, d$. We claim that we can define the new tree $\widetilde{T} = (\widetilde{V}, \widetilde{E}, \widetilde{c})$ as follows:

- $\widetilde{V} := V \cup \{\widetilde{v}_1, \ldots, \widetilde{v}_d\}$ where $\widetilde{v}_1, \ldots, \widetilde{v}_d$ are new (not in V) pairwise different elements,
- $\widetilde{E} := E \cup \{(\widehat{v}, \widetilde{v}_1), \ldots, (\widehat{v}, \widetilde{v}_d)\}$,
- $\widetilde{c}(x) := \begin{cases} c(x) & \text{for all } x \in V, \\ c'_m & \text{for } x = \widetilde{v}_m, \\ & \text{where } c'_m \text{ is the configuration that is reached from} \\ & c(\widehat{v}) \text{ in the computation step given by} \\ & ((q,\eta),(r_m,\theta_m,\mathit{dir}_m)) \in \delta. \end{cases}$

Before we show that $\widetilde{T}$ is a partial tree of M on input w, we define a function $\widetilde{\pi} : \widetilde{V} \to W$ that we will show to be a morphism from $\widetilde{T}$ to *Model*. Since π is a morphism from T to *Model*), we have

$$\pi(\widehat{v}) \models \big(B \wedge (\underline{\alpha}^{\text{time}} = \text{bin}_N(\mathit{time}(\widehat{v}))) \wedge (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(\mathit{pos}(\widehat{v}))) \\ \wedge \alpha^{\text{state}}_{\mathit{state}(\widehat{v})} \wedge \alpha^{\text{read}}_{\mathit{read}(\widehat{v})}\big),$$

hence, due to $\pi(\widehat{v}) \models \mathit{computation}$,

$$\pi(\widehat{v}) \models \bigwedge_{(r,\theta,\mathit{left}) \in \delta(q,\eta)} \mathit{compstep}_{\text{left}}(r,\theta) \wedge \bigwedge_{(r,\theta,\mathit{right}) \in \delta(q,\eta)} \mathit{compstep}_{\text{right}}(r,\theta).$$

As in the case $q \in Q_\exists$ one shows that the numbers $i := \mathit{time}(\widehat{v})$ and $j := \mathit{pos}(\widehat{v})$ satisfy $0 \leqslant i < 2^N - 1$ and $0 < j \leqslant 2^{N+1} - 3$, and one defines

$$k := \min(\{0, \ldots, N-1\} \backslash \text{Ones}(i)),$$

$$l_{\text{left}} := \min(\text{Ones}(j)), \text{ and}$$

$$l_{\text{right}} := \min(\{0, \ldots, N\} \backslash \text{Ones}(j)).$$

As in the case $q \in Q_\exists$ one obtains

$$\pi(\widehat{v}) \models \big(B \wedge \text{rightmost\_zero}(\underline{\alpha}^{\text{time}}, k) \\ \wedge \text{rightmost\_one}(\underline{\alpha}^{\text{pos}}, l_{\text{left}}) \wedge \text{rightmost\_zero}(\underline{\alpha}^{\text{pos}}, l_{\text{right}})\big).$$

Let us now consider some $m \in \{1 \ldots, d\}$. If $dir_m = left$ then, due to $\pi(\widehat{v}) \models compstep_{\text{left}}(r, \theta)$, there exist an element $x_m \in Cloud_{\pi(\widehat{v})}$ and an element $y_m \in W$ such that $x_m \xrightarrow{\Diamond} y_m$ as well as

$$\begin{aligned} x_m \models\ & B \wedge (\underline{X}^{\text{time}} = \underline{\alpha}^{\text{time}}, > k) \wedge \text{rightmost\_one}(\underline{X}^{\text{time}}, k) \\ & \wedge (\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{pos}}, > l_{\text{left}}) \wedge \text{rightmost\_zero}(\underline{X}^{\text{pos}}, l_{\text{left}}) \end{aligned}$$

and

$$y_m \models (\underline{\alpha}^{\text{time}} = \underline{X}^{\text{time}}) \wedge (\underline{\alpha}^{\text{pos}} = \underline{X}^{\text{pos}}) \wedge \alpha_r^{\text{state}} \wedge \alpha_\theta^{\text{written}} \wedge (\underline{\alpha}^{\text{read}} = \underline{X}^{\text{read}}).$$

Similarly, if $dir_m = right$ then, due to $\pi(\widehat{v}) \models compstep_{\text{right}}(r, \theta)$, there exist an element $x_m \in Cloud_{\pi(\widehat{v})}$ and an element $y_m \in W$ such that $x_m \xrightarrow{\Diamond} y_m$ as well as

$$\begin{aligned} x_m \models\ & B \wedge (\underline{X}^{\text{time}} = \underline{\alpha}^{\text{time}}, > k) \wedge \text{rightmost\_one}(\underline{X}^{\text{time}}, k) \\ & \wedge (\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{pos}}, > l_{\text{right}}) \wedge \text{rightmost\_one}(\underline{X}^{\text{pos}}, l_{\text{right}}) \end{aligned}$$

and

$$y_m \models (\underline{\alpha}^{\text{time}} = \underline{X}^{\text{time}}) \wedge (\underline{\alpha}^{\text{pos}} = \underline{X}^{\text{pos}}) \wedge \alpha_r^{\text{state}} \wedge \alpha_\theta^{\text{written}} \wedge (\underline{\alpha}^{\text{read}} = \underline{X}^{\text{read}}).$$

We claim that we can define the desired function $\widetilde{\pi} : \widetilde{V} \to W$ by

$$\widetilde{\pi}(v) := \begin{cases} \pi(v) & \text{if } v \in V, \\ y_m & \text{if } v = \widetilde{v}_m, \text{ for some } m \in \{1, \ldots, d\}. \end{cases}$$

Similarly as in the case $q \in Q_\exists$ one shows that $\widetilde{T}$ is a partial tree of M on input w. Note that also Condition IV is satisfied for $\widehat{v}$. Finally, similarly as in the case $q \in Q_\exists$ one shows that $\widetilde{\pi}$ is a morphism from $\widetilde{T}$ to *Model*. This ends the treatment of the case $q \in Q_\forall$. We have proved Lemma 6.2. □

Chapter 7

Reduction of Alternating Turing Machines Working in Exponential Time to S4 × S5

In this chapter we prove the following theorem.

Theorem 7.1. *The satisfiability problem of* S4 × S5 *is* EXPSPACE-*hard under logarithmic space reduction.*

As explained at the beginning of Chapter 5, we are going to show this by reducing any language L recognized by an Alternating Turing Machine working in exponential time to the satisfiability problem of S4 × S5. The proof is quite similar to the reduction of Alternating Turing Machines working in exponential time to the satisfiability problem of SSL presented in the previous chapter. But, there are also some important differences between the two reductions. On the one hand, we are going to use certain "shared variables" as well, but the mechanism of shared variables in S4 × S5 is much easier than in SSL. On the other hand, the fact that in S4 × S5 the right commutativity property and the left commutativity property hold true (while in SSL only the left commutativity property has to hold true) causes problems that were not present in our treatment of SSL in Chapter 6. Similarly as in that section, for S4 × S5 we will model nodes in an accepting tree of an Alternating Turing Machine by clouds in an S4 × S5-product model. But the commutativity properties have the consequence that for every point in every cloud there exists a copy in every other cloud. The result is that in a cloud modeling a certain node on some path in an accepting tree there are also points that come from other nodes on other computation paths, perhaps even with the same time stamp. This makes the isolation of the correct points carrying the required information (in particular the information about the symbol to be

read at a certain time on some computation path) more difficult. For details see Section 7.1 where the reduction function is defined and explained.
In the first of the following four sections we first give an outline of the definition of the reduction function and then define the reduction function $f_{\mathrm{S4}\times\mathrm{S5}}$ formally. In the second section we show that the reduction function $f_{\mathrm{S4}\times\mathrm{S5}}$ can be computed in logarithmic space. The final two sections are devoted to the correctness proof of the reduction. First we show that in the case $w \in L$ the formula $f_{\mathrm{S4}\times\mathrm{S5}}(w)$ is S4 × S5-satisfiable by explicitly constructing an S4 × S5-product model for $f_{\mathrm{S4}\times\mathrm{S5}}(w)$. In the last section we show that if $f_{\mathrm{S4}\times\mathrm{S5}}(w)$ is S4 × S5-satisfiable then w is an element of L.
In Chapter 8 we are going to show that the satisfiability problem of SSL can be reduced in logarithmic space to the satisfiability problem of S4 × S5. Together with the EXPSPACE-hardness of the satisfiability problem of SSL (Theorem 6.1) this gives another proof of the EXPSPACE-hardness of the satisfiability problem of S4 × S5.

7.1 Construction of the Formula

Let $L \in$ EXPSPACE be an arbitrary language over some alphabet Σ, that is, $L \subseteq \Sigma^*$. We are going to show that there is a logspace computable function $f_{\mathrm{S4}\times\mathrm{S5}}$ mapping strings to strings such that, for any $w \in \Sigma^*$,

- $f_{\mathrm{S4}\times\mathrm{S5}}(w)$ is a bimodal formula and
- $f_{\mathrm{S4}\times\mathrm{S5}}(w)$ is S4 × S5-satisfiable if, and only if, $w \in L$.

Once we have shown this, we have shown the result.
In order to define this desired reduction function $f_{\mathrm{S4}\times\mathrm{S5}}$, we are going to make use of an Alternating Turing Machine for L. Since EXPSPACE = AEXPTIME, there exist an Alternating Turing Machine $M = (Q, \Sigma, \Gamma, q_0, \delta)$ and a univariate polynomial p such that M accepts L, that is, $L(M) = L$, and such that the time used by M on arbitrary input of length n is bounded by $2^{p(n)} - 1$. We can assume without loss of generality $Q = \{0, \ldots, |Q| - 1\}$, $\Gamma = \{0, \ldots, |\Gamma| - 1\}$, that the coefficients of the polynomial p are natural numbers and that, for all $n \in \mathbb{N}$, we have $p(n) \geqslant n$ and $p(n) \geqslant 1$. In the following, whenever we have fixed some $n \in \mathbb{N}$, we set

$$N := p(n).$$

Let us consider an input string $w \in \Sigma^n$ of length n, for some $n \in \mathbb{N}$, and let us sketch the main idea of the construction of the formula $f_{\mathrm{S4}\times\mathrm{S5}}(w)$. The formula $f_{\mathrm{S4}\times\mathrm{S5}}(w)$ will describe the possible computations of M on input w in the following sense: any S4 $\times$ S5-product model of $f_{\mathrm{S4}\times\mathrm{S5}}(w)$ will essentially contain an accepting tree of M on input w, and if there exists an accepting tree of M on input w then one can turn this into an S4 $\times$ S5-product model of $f_{\mathrm{S4}\times\mathrm{S5}}(w)$. In such a model, any node in an accepting tree of M on input w will be modeled by a cloud (that is, by an $\xrightarrow{L}$-equivalence class) in which certain shared variables (we use the notion "shared variables" in the same sense as in Section 5.1) will have values that describe the data of the computation node that are important in this computation step. Which data are these? First of all, we need the time of the computation node. We assume that the computation starts with the initial configuration of M on input w at time 0. Since the ATM M needs at most $2^N - 1$ time steps, we can store the time of each computation node in a binary counter counting from 0 to $2^N - 1$. Since during each time step at most one additional cell either to the right or to the left of the previous cell can be visited, we can describe any configuration reachable during a computation of M on input w by the following data:

- the state $q \in Q$ of the configuration,
- the current content of the tape, given by a string in $\Gamma^{2\cdot(2^N-1)+1} = \Gamma^{2^{N+1}-1}$,
- the current position of the tape head, given by a number in $\{0, \ldots, 2^{N+1} - 2\}$.

We assume that in the initial configuration on input w the tape content is $\#^{2^N} w \#^{2^N-1-n}$ (remember that we use # for the blank symbol) and that the tape head scans the blank # to the left of the first symbol of w, that is, the position of the tape head is $2^N - 1$. If a cloud in an S4 $\times$ S5-product model of $f_{\mathrm{S4}\times\mathrm{S5}}(w)$ describes a computation node of M on input w then in this cloud the following shared variables will have the following values:

- a vector $\underline{\alpha}^{\text{time}} = (\alpha^{\text{time}}_{N-1}, \ldots, \alpha^{\text{time}}_0)$ giving in binary the current time of the computation,
- a vector $\underline{\alpha}^{\text{pos}} = (\alpha^{\text{pos}}_{N}, \ldots, \alpha^{\text{pos}}_0)$ giving in binary the current position of the tape head,
- a vector $\underline{\alpha}^{\text{state}} = (\alpha^{\text{state}}_0, \ldots, \alpha^{\text{state}}_{|Q|-1})$ giving in unary the current state of the computation (here "unary" means: exactly one of the shared variables α^{state}_i will be true, namely the one with i being the current state),

- a vector $\underline{\alpha}^{\text{read}} = (\alpha_0^{\text{read}}, \ldots, \alpha_{|\Gamma|-1}^{\text{read}})$ giving in unary the symbol in the current cell (here "unary" means: exactly one of the shared variables α_i^{read} will be true, namely the one with i being the symbol in the current cell),
- a vector $\underline{\alpha}^{\text{written}} = (\alpha_0^{\text{written}}, \ldots, \alpha_{|\Gamma|-1}^{\text{written}})$ giving in unary the symbol that has just been written into the cell that has just been left, unless the cloud corresponds to the first node in the computation tree — in that case the value of this vector is irrelevant (here "unary" means: exactly one of the variables $\alpha_i^{\text{written}}$ will be true, namely the one with i being the symbol that has just been written),
- a vector $\underline{\alpha}^{\text{prevpos}} = (\alpha_N^{\text{prevpos}}, \ldots, \alpha_0^{\text{prevpos}})$ giving in binary the previous position of the tape head, that is, the position of the cell that has just been left, unless the cloud corresponds to the first node in the computation tree — in that case the value of this vector is irrelevant.

The formula $f_{\text{S4}\times\text{S5}}(w)$ has to ensure that for any possible computation step starting from such a computation node in an accepting tree there exists a cloud describing the corresponding successor node in the accepting tree. In this new cloud, the value of the counter for the time $\underline{\alpha}^{\text{time}}$ has to be incremented. This can be done by the technique for implementing a binary counter in S4 × S5 that was described in Section 5.2. In parallel, we have to make sure that in this new cloud also the vectors $\underline{\alpha}^{\text{pos}}$, $\underline{\alpha}^{\text{state}}$, $\underline{\alpha}^{\text{read}}$, $\underline{\alpha}^{\text{written}}$, and $\underline{\alpha}^{\text{prevpos}}$ are set to the right values. The vector $\underline{\alpha}^{\text{prevpos}}$ is a copy of the vector $\underline{\alpha}^{\text{pos}}$ in the previous cloud (we will see below how one can copy it). For the vectors $\underline{\alpha}^{\text{pos}}$, $\underline{\alpha}^{\text{state}}$, and $\underline{\alpha}^{\text{written}}$ these values can be computed using the corresponding element of the transition relation δ of the ATM. For example, $\underline{\alpha}^{\text{pos}}$ has to be decremented by one if the tape head moves to the left, and it has to be incremented by one if the tape head moves to the right. Also the new state (to be stored in $\underline{\alpha}^{\text{state}}$) and the symbol written into the cell that has just been left (to be stored in $\underline{\alpha}^{\text{written}}$) are determined by the data of the previous computation node and by the corresponding element of the transition relation δ.
But the vector $\underline{\alpha}^{\text{read}}$ is supposed to describe the symbol in the current cell. This symbol is not determined by the current computation step but has either been written the last time when this cell has been visited during this computation or, when this cell has never been visited before, the symbol in this cell is still the one that was contained in this cell before the computation started. How can one ensure that $\underline{\alpha}^{\text{read}}$ is set to the right value? We will do this by using persistent variables and a variable B^{active} that is neither shared nor persistent. We will ensure that any cloud corresponding to a node in the

computation tree in which the cell i is being visited contains a point with a persistent position vector $\underline{X}^{\text{pos}} = \text{bin}_{N+1}(i)$ and with a persistent time vector $\underline{X}^{\text{time-apv}}$ that will be forced to contain in binary form the time one step after the previous visit of the cell i or, if the cell i has not been visited before, to contain the value 0. Furthermore, there will be a third persistent vector $\underline{X}^{\text{read}}$. If the cell i has not been visited before then we will make sure that the vector $\underline{X}^{\text{read}}$ encodes the initial symbol contained in cell i. If the cell i has been visited before then we will make sure that the vector $\underline{X}^{\text{read}}$ encodes the symbol that has been written into the cell during the previous visit of the cell. In order to do that we need to determine the time of the previous visit. For that the Boolean variable B^{active} is used and the fact that due to the left commutativity property the point has $\overset{\Diamond}{\rightarrow}$-predecessors in all clouds corresponding to any node on the path in the accepting tree from the root to the current cloud. Then, when one knows this time, one can look at the shared variable vector $\underline{\alpha}^{\text{written}}$ in the cloud corresponding to the step after the previous visit of this cell (this shared variable vector encodes the symbol that has just been written into the cell) and can copy its value to the persistent variable vector $\underline{X}^{\text{read}}$. In order to implement all this we use the following persistent variables:

- a vector $\underline{X}^{\text{time-apv}} = (X_{N-1}^{\text{time-apv}}, \ldots, X_0^{\text{time-apv}})$ containing in binary the time one step after the previous visit of the current cell (the exponent of $X_i^{\text{time-apv}}$ stands for "time after previous visit") or, if the current cell has not been visited before, containing in binary the number 0,
- a vector $\underline{X}^{\text{pos}} = (X_N^{\text{pos}}, \ldots, X_0^{\text{pos}})$ containing in binary the current position of the tape head, that is, the position of the current cell,
- a vector $\underline{X}^{\text{read}} = (X_0^{\text{read}}, \ldots, X_{|\Gamma|-1}^{\text{read}})$ giving in unary the symbol in the cell described by $\underline{X}^{\text{pos}}$ (here "unary" means: exactly one of the shared variables α_i^{read} will be true, namely the one with i being the symbol in the cell described by $\underline{X}^{\text{pos}}$),

In fact, the current cell may have been visited several times already. Of course only the $\overset{\Diamond}{\rightarrow}$-successor of the point corresponding to the previous visit (that is, corresponding to the very last visit of the cell before the current visit, not earlier visits) should be allowed to copy the value of its vector $\underline{\alpha}^{\text{written}}$ to $\underline{X}^{\text{read}}$. In order to determine the right point, we are going to use an additional Boolean

- variable B^{active} (which is neither persistent nor shared) saying whether the current point is active or not.

We shall explain later how all this works. Finally, there are two more vectors of persistent variables that are needed for changing the values of the shared variable vectors $\underline{\alpha}^{\text{time}}$ and $\underline{\alpha}^{\text{pos}}$:

- a vector $\underline{X}^{\text{prevtime}} = (X^{\text{prevtime}}_{N-1}, \ldots, X^{\text{prevtime}}_0)$ containing in binary the current time of the computation minus one, unless the cloud corresponds to the first node in the computation tree — in that case the value of this vector is irrelevant,
- a vector $\underline{X}^{\text{prevpos}} = (X^{\text{prevpos}}_N, \ldots, X^{\text{prevpos}}_0)$ containing in binary the previous position of the tape head, that is, the position of the cell that has just been left, unless the cloud corresponds to the first node in the computation tree — in that case the value of this vector is irrelevant.

Now we come to the formal definition of the formula $f_{\mathrm{S4}\times\mathrm{S5}}(w)$. The formula $f_{\mathrm{S4}\times\mathrm{S5}}(w)$ will have the following structure:

$$\begin{aligned} f_{\mathrm{S4}\times\mathrm{S5}}(w) \quad := \quad & \mathit{persistence} \\ & \wedge K\Box \mathit{uniqueness} \\ & \wedge \mathit{start} \\ & \wedge K\Box \mathit{initial\_symbols} \\ & \wedge K\Box \mathit{written\_symbols} \\ & \wedge K\Box \mathit{read\_a\_symbol} \\ & \wedge K\Box \mathit{computation} \\ & \wedge K\Box \mathit{no\_reject}. \end{aligned}$$

The formula $f_{\mathrm{S4}\times\mathrm{S5}}(w)$ will contain the following propositional variables:

$$\begin{aligned} & A^{\text{time}}_{N-1}, \ldots, A^{\text{time}}_0, \\ & A^{\text{pos}}_N, \ldots, A^{\text{pos}}_0, \\ & A^{\text{state}}_0, \ldots, A^{\text{state}}_{|Q|-1}, \\ & A^{\text{written}}_0, \ldots, A^{\text{written}}_{|\Gamma|-1}, \\ & A^{\text{read}}_0, \ldots, A^{\text{read}}_{|\Gamma|-1}, \\ & A^{\text{prevpos}}_N, \ldots, A^{\text{prevpos}}_0, \\ & X^{\text{prevtime}}_{N-1}, \ldots, X^{\text{prevtime}}_0, \\ & X^{\text{prevpos}}_N, \ldots, X^{\text{prevpos}}_0, \\ & X^{\text{pos}}_N, \ldots, X^{\text{pos}}_0, \\ & X^{\text{time-apv}}_{N-1}, \ldots, X^{\text{time-apv}}_0, \\ & X^{\text{read}}_0, \ldots, X^{\text{read}}_{|\Gamma|-1}, \\ & B^{\text{active}}. \end{aligned}$$

For *string* $\in$ {time, pos, state, read, prevpos, written} and natural numbers i we use α_i^{string} as an abbreviation for LA_i^{string}. These formulas α_i^{string} are the shared variables we talked about above.
We are now going to define the subformulas of $f_{\mathrm{S4}\times\mathrm{S5}}(w)$. We will use the abbreviations introduced above, in Table 5.1, and in Table 6.1.
The following formula makes sure that certain propositional variables are persistent:

$$\begin{aligned}&\mathit{persistence}\\ &\quad := \text{persistent}(\underline{X}^{\text{prevtime}}) \wedge \text{persistent}(\underline{X}^{\text{prevpos}})\\ &\qquad \wedge \text{persistent}(\underline{X}^{\text{pos}}) \wedge \text{persistent}(\underline{X}^{\text{time-apv}}) \wedge \text{persistent}(\underline{X}^{\text{read}}).\end{aligned}$$

The following formula makes sure that in each of the vectors of shared or persistent variables that describe in a unary way the current state respectively the written symbol exactly one variable is true:

$$\mathit{uniqueness} \ := \ \text{unique}(\underline{\alpha}^{\text{state}}) \wedge \text{unique}(\underline{\alpha}^{\text{written}}) \wedge \text{unique}(\underline{X}^{\text{read}}).$$

The vector $\underline{\alpha}^{\text{read}}$ will satisfy the same uniqueness condition automatically due to another formula (due to the formula *read_a_symbol*).
The following formula ensures that the shared variables in the cloud corresponding to the first node in a computation tree have the correct values. The computation starts at time 0 with the tape head at position $2^N - 1$ and in the state q_0. The vector $\underline{\alpha}^{\text{read}}$ will automatically get the correct value # due to the formulas *read_a_symbol* and *initial_symbols*, that will be introduced next. For the vectors $\underline{\alpha}^{\text{prevpos}}$ and $\underline{\alpha}^{\text{written}}$ we do not need to fix any values.

$$\mathit{start} \ := \ (\underline{\alpha}^{\text{time}} = \text{bin}_N(0)) \wedge (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(2^N - 1)) \wedge \alpha_{q_0}^{\text{state}}.$$

The following formula ensures that whenever an initial symbol on the tape is requested (by a point with persistent time $\underline{X}^{\text{time-apv}}$ equal to 0) it is stored in a persistent vector $\underline{X}^{\text{read}}$ of this point.

$$\begin{aligned}&\mathit{initial\_symbols}\\ &\quad := (\underline{X}^{\text{time-apv}} = \text{bin}_N(0))\\ &\qquad \to \Bigg(\bigwedge_{i=1}^{n} \big((\underline{X}^{\text{pos}} = \text{bin}_{N+1}(2^N - 1 + i)) \to X_{w_i}^{\text{read}}\big)\\ &\qquad\quad \wedge \Big(\big((\underline{X}^{\text{pos}} \leqslant \text{bin}_{N+1}(2^N - 1)) \vee (\underline{X}^{\text{pos}} > \text{bin}_{N+1}(2^N - 1 + n))\big)\\ &\qquad\qquad \to X_{\#}^{\text{read}}\Big)\Bigg).\end{aligned}$$

We explain this formula. By another formula we will ensure that whenever a cell is visited for the first time there will be an "active" point storing the position of this cell in a persistent vector $\underline{X}^{\text{pos}}$ and such that its persistent time vector $\underline{X}^{\text{time-apv}}$ has the binary value 0. The formula above ensures that the persistent vector $\underline{X}^{\text{read}}$ in this point stores the correct initial symbol in this cell. This is either a symbol w_i of the input string $w = w_1 \dots w_n$ or the blank #.
The following formula ensures that whenever a symbol that has just been written on the tape is requested (by an active point with the correct persistent time $\underline{X}^{\text{time-apv}}$) then it is copied into a persistent vector $\underline{X}^{\text{read}}$ of this point.

$$\begin{aligned}&\mathit{written\_symbols}\\ &\quad := \big((\underline{X}^{\text{time-apv}} > \text{bin}_N(0)) \wedge (\underline{X}^{\text{time-apv}} = \underline{\alpha}^{\text{time}}) \wedge B^{\text{active}}\big)\\ &\qquad \to (\underline{X}^{\text{read}} = \underline{\alpha}^{\text{written}}).\end{aligned}$$

The following formula ensures that the shared variable vector $\underline{\alpha}^{\text{read}}$ describes the symbol in the current cell.

$$\begin{aligned}&\mathit{read\_a\_symbol}\\ &\quad := \mathit{existence\_of\_a\_reading\_point}\\ &\qquad \wedge \mathit{time\_of\_previous\_visit}\\ &\qquad \wedge \mathit{becoming\_inactive}\\ &\qquad \wedge \mathit{staying\_inactive}\\ &\qquad \wedge \mathit{storing\_the\_read\_symbol},\end{aligned}$$

where

$$\begin{aligned}&\mathit{existence\_of\_a\_reading\_point}\\ &\quad := L\big((\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{pos}}) \wedge (\underline{X}^{\text{time-apv}} \leqslant \underline{\alpha}^{\text{time}}) \wedge B^{\text{active}}\big),\\ &\mathit{time\_of\_previous\_visit}\\ &\quad := \Big(\big((\underline{X}^{\text{time-apv}} > \text{bin}_N(0)) \wedge (\underline{X}^{\text{time-apv}} = \underline{\alpha}^{\text{time}}) \wedge B^{\text{active}}\big)\\ &\qquad \to (\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{prevpos}})\Big),\\ &\mathit{becoming\_inactive}\\ &\quad := \Big(\big((\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{prevpos}}) \wedge (\underline{X}^{\text{time-apv}} < \underline{\alpha}^{\text{time}})\big) \to \neg B^{\text{active}}\Big),\\ &\mathit{staying\_inactive}\\ &\quad := \Big(\neg B^{\text{active}} \to \Box\neg B^{\text{active}}\Big),\\ &\mathit{storing\_the\_read\_symbol}\\ &\quad := \Big(\big((\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{pos}}) \wedge (\underline{X}^{\text{time-apv}} \leqslant \underline{\alpha}^{\text{time}}) \wedge B^{\text{active}}\big) \to (\underline{\alpha}^{\text{read}} = \underline{X}^{\text{read}})\Big).\end{aligned}$$

We explain these formulas. The formula *existence_of_a_reading_point* enforces the existence of an "active" point in the current cloud that contains in the persistent vector $\underline{X}^{\text{pos}}$ the number of the current cell and that we wish to force to contain in the persistent vector $\underline{X}^{\text{time-apv}}$ the time one step after the previous visit of this cell, if this cell has been visited before, or the value 0, otherwise. How can we enforce that? We make essential use of the left commutativity property. The point whose existence is ensured by the formula *existence_of_a_reading_point* has $\overset{\Diamond}{\to}$-predecessors in all clouds corresponding to the nodes on the path in the accepting tree from the root to the node corresponding to the current cloud. Since we demand $\underline{X}^{\text{time-apv}} \leqslant \underline{\alpha}^{\text{time}}$ the time stored in binary in the persistent variable $\underline{X}^{\text{time-apv}}$ must be identical with the time of one of the clouds corresponding to a node on this path. The formula *time_of_previous_visit* ensures that if it is positive then it can be equal to the time $\underline{\alpha}^{\text{time}}$ of a cloud corresponding to a node on this path only when $\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{prevpos}}$, that is, only when in the step leading to this node something has been written into the current cell. So, if the current cell has not been visited before, the binary value of $\underline{X}^{\text{time-apv}}$ must be 0. Then the formula *initial_symbols* will ensure that the persistent variable $\underline{X}^{\text{read}}$ gets the correct value. Otherwise, when the cell has been visited before, in fact, it may have been visited several times already. In this case, we have to make sure that the number stored in $\underline{X}^{\text{time-apv}}$ is the time after the very last visit to this cell immediately before the current visit. This is ensured by the formulas *becoming_inactive* and *staying_inactive*. If the number stored in $\underline{X}^{\text{time-apv}}$ were strictly smaller than the the time after the very last visit to this cell immediately before the current visit then, due to *becoming_inactive* the corresponding point would be set to "inactive", and, due to *staying_inactive*, also its $\overset{\Diamond}{\to}$-successor would stay inactive. But this would apply also to the point in the current cloud which is supposed to be active and to contain the correct value in $\underline{X}^{\text{time-apv}}$ according to formula *existence_of_a_reading_point*. Thus, the first four subformulas of *read_a_symbol* ensure that the current cloud contains an active point that contains in the persistent vector $\underline{X}^{\text{pos}}$ the number of the current cell and that contains in the persistent vector $\underline{X}^{\text{time-apv}}$ the time one step after the previous visit of this cell, if this cell has been visited before, or the value 0, otherwise. Finally, the formula *storing_the_read_symbol* makes sure that the value of $\underline{X}^{\text{read}}$ (which contains the symbol written into the current cell during the previous visit of this cell, or, if the current cell has not been visited before, the initial symbol in this cell) is copied into the shared variable vector $\underline{\alpha}^{\text{read}}$.

Next, we wish to define the formula *computation* that describes the computation steps. We have to distinguish between the two cases whether the tape head is going to move to the left or to the right. If in a computation step the symbol $\theta \in \Gamma$ is written into the current cell, if the tape head moves to the right, and if the new state after this step is the state $r \in Q$, then the following formula $compstep_{\text{right}}(r, \theta)$ guarantees the existence of a point and its cloud with suitable values in the shared variable vectors $\underline{\alpha}^{\text{time}}, \underline{\alpha}^{\text{pos}}, \underline{\alpha}^{\text{prevpos}}, \underline{\alpha}^{\text{state}}, \underline{\alpha}^{\text{written}}$.

We explain this formula. The first four lines of this formula take care that the two binary counters $\underline{\alpha}^{\text{time}}$ and $\underline{\alpha}^{\text{pos}}$ for the current time and for the current position of the tape head are incremented at the same time. This is similar to the formula $\text{counters}_{\text{S4}\times\text{S5},n}$ in Section 5.2. Furthermore, also the persistent vectors $\underline{X}^{\text{prevtime}}$ (that we will not need otherwise) and $\underline{X}^{\text{prevpos}}$ (that we use again in the fifth line of this formula) get the correct values. The fifth line ensures that the shared variable vectors $\underline{\alpha}^{\text{prevpos}}$, $\underline{\alpha}^{\text{state}}$ and $\underline{\alpha}^{\text{written}}$ of the current point get the correct values. The shared variable vector $\underline{\alpha}^{\text{read}}$ will get the correct value due to the formulas *initial_symbols*, *written_symbols*, and *read_a_symbol*.

$$
\begin{aligned}
&compstep_{\text{right}}(r, \theta) \\
&:= \bigwedge_{k=0}^{N-1} \bigwedge_{l=0}^{N} \Bigg(\big(\text{rightmost\_zero}(\underline{\alpha}^{\text{time}}, k) \wedge \text{rightmost\_zero}(\underline{\alpha}^{\text{pos}}, l)\big) \\
&\qquad \to L\Bigg((\underline{X}^{\text{prevtime}} = \underline{\alpha}^{\text{time}}) \wedge (\underline{X}^{\text{prevpos}} = \underline{\alpha}^{\text{pos}}) \\
&\qquad\qquad \wedge \Diamond\Big((\underline{\alpha}^{\text{time}} = \underline{X}^{\text{prevtime}}, > k) \wedge \text{rightmost\_one}(\underline{\alpha}^{\text{time}}, k) \\
&\qquad\qquad\quad \wedge (\underline{\alpha}^{\text{pos}} = \underline{X}^{\text{prevpos}} > l) \wedge \text{rightmost\_one}(\underline{\alpha}^{\text{pos}}, l) \\
&\qquad\qquad\quad \wedge (\underline{\alpha}^{\text{prevpos}} = \underline{X}^{\text{prevpos}}) \wedge \alpha_r^{\text{state}} \wedge \alpha_\theta^{\text{written}} \Big)\Bigg)\Bigg).
\end{aligned}
$$

If in a computation step the symbol $\theta \in \Gamma$ is written into the current cell, if the tape head moves to the left, and if the new state after this step is the state $r \in Q$, then the following formula guarantees the existence of a point and its cloud with suitable values in the shared variables.

$$\begin{aligned}
&\mathit{compstep}_{\text{left}}(r,\theta)\\
&:= \bigwedge_{k=0}^{N-1}\bigwedge_{l=0}^{N}\Bigg(\big(\text{rightmost\_zero}(\underline{\alpha}^{\text{time}},k)\wedge \text{rightmost\_one}(\underline{\alpha}^{\text{pos}},l)\big)\\
&\quad\to L\Bigg((\underline{X}^{\text{prevtime}}=\underline{\alpha}^{\text{time}})\wedge(\underline{X}^{\text{prevpos}}=\underline{\alpha}^{\text{pos}})\\
&\qquad\wedge\Diamond\Big((\underline{\alpha}^{\text{time}}=\underline{X}^{\text{prevtime}},>k)\wedge \text{rightmost\_one}(\underline{\alpha}^{\text{time}},k)\\
&\qquad\quad\wedge(\underline{\alpha}^{\text{pos}}=\underline{X}^{\text{prevpos}},>l)\wedge \text{rightmost\_zero}(\underline{\alpha}^{\text{pos}},l)\\
&\qquad\quad\wedge(\underline{\alpha}^{\text{prevpos}}=\underline{X}^{\text{prevpos}})\wedge\alpha_r^{\text{state}}\wedge\alpha_\theta^{\text{written}}\Big)\Bigg)\Bigg).
\end{aligned}$$

This formula is very similar to the previous one with the exception that here the binary counter for the position of the tape head is decremented.
The computation is modeled by the following subformula. Remember that Q is the disjoint union of the sets $\{q_{\text{accept}}\}$, $\{q_{\text{reject}}\}$, $Q_\forall$, $Q_\exists$.

$$\begin{aligned}
&\mathit{computation}\\
&:= \bigwedge_{q\in Q_\forall}\bigwedge_{\eta\in\Gamma}\Bigg((\alpha_q^{\text{state}}\wedge\alpha_\eta^{\text{read}})\to\\
&\quad\Bigg(\bigwedge_{(r,\theta,\mathit{left})\in\delta(q,\eta)}\mathit{compstep}_{\text{left}}(r,\theta)\wedge\bigwedge_{(r,\theta,\mathit{right})\in\delta(q,\eta)}\mathit{compstep}_{\text{right}}(r,\theta)\Bigg)\Bigg)\\
&\quad\wedge\bigwedge_{q\in Q_\exists}\bigwedge_{\eta\in\Gamma}\Bigg((\alpha_q^{\text{state}}\wedge\alpha_\eta^{\text{read}})\to\\
&\quad\Bigg(\bigvee_{(r,\theta,\mathit{left})\in\delta(q,\eta)}\mathit{compstep}_{\text{left}}(r,\theta)\vee\bigvee_{(r,\theta,\mathit{right})\in\delta(q,\eta)}\mathit{compstep}_{\text{right}}(r,\theta)\Bigg)\Bigg)\Bigg).
\end{aligned}$$

Finally, the subformula *no_reject* is defined as follows.

$$\mathit{no\_reject} := \neg\alpha^{\text{state}}_{q_{\text{reject}}}.$$

We have completed the description of the formula $f_{\text{S4}\times\text{S5}}(w)$ for $w\in\Sigma^*$. It is clear that $f_{\text{S4}\times\text{S5}}(w)$ is a bimodal formula, for any $w\in\Sigma^*$. We still have to show two claims:

1. The function $f_{\text{S4}\times\text{S5}}$ can be computed in logarithmic space.

2. For any $w \in \Sigma^*$,

$$w \in L \iff \text{the bimodal formula } f_{\mathrm{S4}\times\mathrm{S5}}(w) \text{ is } \mathrm{S4}\times\mathrm{S5}\text{-satisfiable.}$$

The first claim is shown in the following section. The two directions of the equivalence in the second claim are shown afterwards in separate sections.

7.2 LOGSPACE Computability of the Reduction

We wish to show that the function $f_{\mathrm{S4}\times\mathrm{S5}}$ can be computed in logarithmic space. This is shown by the same argument as the corresponding claim for the function f_{SSL} in Section 6.2.

7.3 Construction of a Model

In this section we show for any $w \in \Sigma^*$, if $w \in L$ then the bimodal formula $f_{\mathrm{S4}\times\mathrm{S5}}(w)$ is S4 × S5-satisfiable. Let us assume $w \in L$. We are going to explicitly define an S4 × S5-product model of $f_{\mathrm{S4}\times\mathrm{S5}}(w)$.
There exists an accepting tree $T = (V, E, c)$ of M on input w, where V is the set of nodes of T, where $E \subseteq V \times V$ is the set of edges, and where the function $c : V \to Q \times \{0, \dots, 2^{N+1} - 2\} \times \Gamma^{2^{N+1}-1}$ labels each node with a configuration (remember the discussion about the description of configurations at the beginning of Section 7.1). Let $root \in V$ be the root of T. We will now construct an S4 × S5-product model of $f_{\mathrm{S4}\times\mathrm{S5}}(w)$. We define an $S4$-frame $(W_1, R_\Diamond)$ by

$$W_1 := V \quad \text{and} \quad R_\Diamond := \text{ the reflexive-transitive closure of } E.$$

That means, for any $v, v' \in W_1$ we have $vR_\Diamond v'$ iff in the tree T there is a path from v to v'. We define an $S5$-frame (W_2, R_L) by

$$W_2 := V \quad \text{and} \quad R_L := W_2 \times W_2.$$

Then the product frame $(W, \xrightarrow{\Diamond}, \xrightarrow{L})$ with $W := W_1 \times W_2$ and with $\xrightarrow{\Diamond}$ and $\xrightarrow{L}$ defined as in Definition 3.10.2 is an S4 × S5-product frame. We still need to define a suitable valuation σ. We define the valuation σ as follows.

$$\begin{array}{rcl}
\sigma(A_k^{\text{time}}) & := & \{(v,x) \;:\; v,x \in V \text{ and } k \in \text{Ones}(\mathit{time}(v))\}, \\
\sigma(X_k^{\text{prevtime}}) & := & \{(v,x) \;:\; v \in V, x \in V \backslash \{\mathit{root}\} \text{ and } k \in \text{Ones}(\mathit{time}(x)-1)\}, \\
\sigma(X_k^{\text{time-apv}}) & := & \{(v,x) \;:\; v \in V, x \in V \backslash \{\mathit{root}\} \text{ and there exists a node} \\
& & y \neq \mathit{root} \text{ on the path from } \mathit{root} \text{ to } x \text{ such that} \\
& & \mathit{pos}(\mathit{pred}(y)) = \mathit{pos}(x) \text{ and } k \in \text{Ones}(\mathit{time}(y)) \text{ and, for all} \\
& & \text{nodes } z \text{ on thepath from } y \text{ to } x \text{ with } z \neq y, \\
& & \mathit{pos}(\mathit{pred}(z)) \neq \mathit{pos}(x)\},
\end{array}$$

for $k \in \{0, \ldots, N-1\}$,

$$\begin{array}{rcl}
\sigma(A_k^{\text{pos}}) & := & \{(v,x) \;:\; v \in V, x \in V \text{ and } k \in \text{Ones}(\mathit{pos}(v))\}, \\
\sigma(X_k^{\text{pos}}) & := & \{(v,x) \;:\; v \in V, x \in V \text{ and } k \in \text{Ones}(\mathit{pos}(x))\}, \\
\sigma(A_k^{\text{prevpos}}) & := & \{(v,x) \;:\; v \in V \backslash \{\mathit{root}\}, x \in V \text{ and } k \in \text{Ones}(\mathit{pos}(\mathit{pred}(v)))\} \\
\sigma(X_k^{\text{prevpos}}) & := & \{(v,x) \;:\; v \in V, x \in V \backslash \{\mathit{root}\} \text{ and } k \in \text{Ones}(\mathit{pos}(\mathit{pred}(x)))\},
\end{array}$$

for $k \in \{0, \ldots, N\}$,

$$\sigma(A_q^{\text{state}}) := \{(v,x) \;:\; v \in V, x \in V \text{ and } q = \mathit{state}(v)\}$$

for $q \in Q$,

$$\begin{array}{rcl}
\sigma(A_\eta^{\text{read}}) & := & \{(v,x) \;:\; v,x \in V \text{ and } \eta = \mathit{read}(v)\}, \\
\sigma(A_\eta^{\text{written}}) & := & \{(v,x) \;:\; (v \in V \backslash \{\mathit{root}\}, x \in V \text{ and } \eta = \mathit{written}(v)) \\
& & \quad \text{or } (v = \mathit{root} \text{ and } x \in V \text{ and } \eta = \#)\}, \\
\sigma(X_\eta^{\text{read}}) & := & \{(v,x) \;:\; v,x \in V \text{ and } \eta = \mathit{read}(x))\},
\end{array}$$

for $\eta \in \Gamma$,

$$\begin{array}{rcl}
\sigma(B^{\text{active}}) & := & \{(v,x) \;:\; v,x \in V \text{ and } v \text{ is an element of the path} \\
& & \quad \text{from } \mathit{root} \text{ to } x\}.
\end{array}$$

We have defined an S4 × S5-product model $(W_1 \times W_2, \xrightarrow{\Diamond}, \xrightarrow{L})$. We claim that in this model $(\mathit{root}, \mathit{root}) \models f_{\text{S4}\times\text{S5}}(w)$. For an illustration of an important detail of the model see Figure 7.1.

First, we observe that for $\mathit{string} \in \{\text{prevtime}, \text{prevpos}, \text{pos}, \text{time-apv}, \text{written}\}$ and natural numbers i as well as for $(v,x) \in V \times V$ the truth value of the variable X_i^{string} in the point (v,x) does not depend on v. Hence, all these variables are persistent. So,

$$(\mathit{root}, \mathit{root}) \models \mathit{persistence}.$$

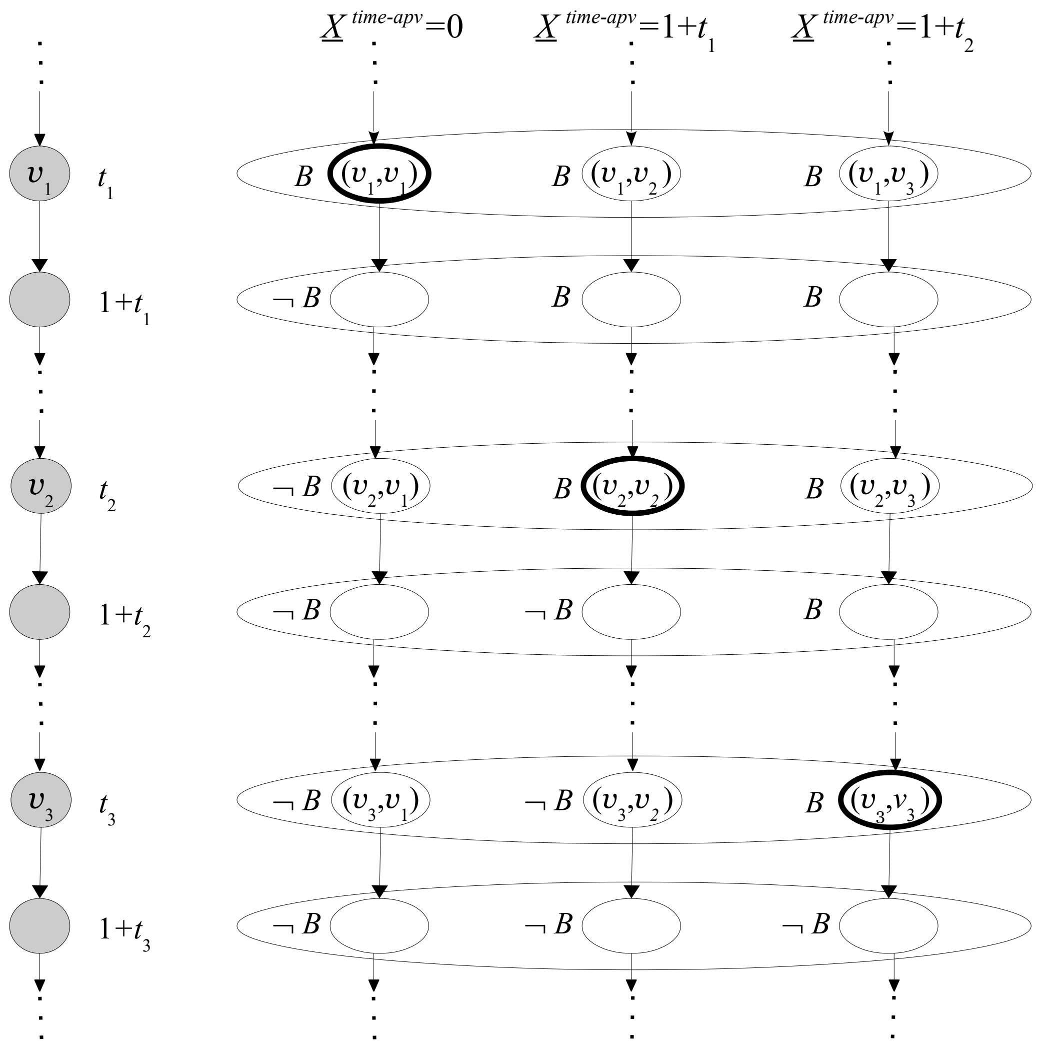


Figure 7.1: A possible detail of an S4 × S5-product model of the formula $f_{S4\times S5}(w)$. Consider a certain cell and let us assume that v_1, v_2, v_3 are the first three computation nodes on some computation path in which this cell is visited. Let $t_i := \mathit{time}(v_i)$. The diagram on the left shows a part of the computation path. The diagram on the right shows the corresponding part of the S4 × S5-product model. Here B stands for B^{active}.

Similarly, for *string* $\in$ {time, pos, state, read, prevpos, written} and natural numbers i as well as for $(v,x) \in V \times V$ the truth value of the variable A_i^{string} in the point (v,x) does not depend on x. Note that for $v \in V$ the set

$$\mathrm{Cloud}(v) := \{(v,x) \ : \ v \in V, x \in V\}$$

is the $\xrightarrow{L}$-equivalence class of (v,v). All of the vectors of shared variables have the expected values: for $v \in V$ and $s \in \mathrm{Cloud}(v)$. We see

$$\begin{array}{rcl} s & \models & (\underline{\alpha}^{\mathrm{time}} = \mathrm{bin}_N(\mathit{time}(v))), \\ s & \models & (\underline{\alpha}^{\mathrm{pos}} = \mathrm{bin}_{N+1}(\mathit{pos}(v))), \\ (s & \models & \alpha_q^{\mathrm{state}}) \iff q = \mathit{state}(v), \text{ for } q \in Q, \\ (s & \models & \alpha_\eta^{\mathrm{read}}) \iff \eta = \mathit{read}(v), \text{ for } \eta \in \Gamma, \end{array}$$

and for $v \in V \backslash \{\mathit{root}\}$ and $s \in \mathrm{Cloud}(v)$ we see

$$\begin{array}{rcl} s & \models & (\underline{\alpha}^{\mathrm{prevpos}} = \mathrm{bin}_{N+1}(\mathit{pos}(\mathit{pred}(v)))), \\ (s & \models & \alpha_\eta^{\mathrm{written}}) \iff \eta = \mathit{written}(v), \text{ for } \eta \in \Gamma. \end{array}$$

Similarly, for the vectors of persistent variables we see, for $v, x \in V$:

$$\begin{array}{rcl} (v,x) & \models & (\underline{X}^{\mathrm{prevtime}} = \mathrm{bin}_N(t)), \text{ where} \\ & & t = \begin{cases} 0, & \text{if } x = \mathit{root} \\ \mathit{time}(\mathit{pred}(x)), & \text{otherwise,} \end{cases} \\ (v,x) & \models & (\underline{X}^{\mathrm{prevpos}} = \mathrm{bin}_{N+1}(p)), \text{ where} \\ & & p = \begin{cases} 0, & \text{if } x = \mathit{root} \\ \mathit{pos}(\mathit{pred}(x)), & \text{otherwise,} \end{cases} \\ (v,x) & \models & (\underline{X}^{\mathrm{pos}} = \mathrm{bin}_{N+1}(\mathit{pos}(x))), \\ (v,x) & \models & (\underline{X}^{\text{time-apv}} = \mathrm{bin}_N(t)), \text{ where} \\ & & t = \begin{cases} 0, & \text{if the cell } \mathit{pos}(x) \text{ has not been visited before} \\ & \text{on the computation path from } \mathit{root} \text{ to } x \\ & \text{(this is in particular true in the case } x = \mathit{root}), \\ \multicolumn{2}{l}{1 + \text{the time of the previous visit of the cell } \mathit{pos}(x),} \\ & \text{otherwise.} \end{cases} \\ ((v,x) & \models & X_\eta^{\mathrm{read}}) \iff \eta = \mathit{read}(x), \text{ for } \eta \in \Gamma, \end{array}$$

It is straightforward to check that for all $(v,x) \in W$

$$(v,x) \models \mathit{uniqueness}$$

(in fact, in order to achieve $(\mathit{root}, x) \models \mathrm{unique}(\underline{\alpha}^{\mathrm{written}})$ we made a somewhat arbitrary choice for the value of $A_\eta^{\mathrm{written}}$ in (root, x), for $x \in V$ and $\eta \in \Gamma$). Hence, we have

$$(\mathit{root}, \mathit{root}) \models K\Box \mathit{uniqueness}.$$

It is clear as well that

$$(\mathit{root}, \mathit{root}) \models \mathit{start}.$$

As none of the nodes in the accepting tree T is labeled with a configuration with the state q_{reject} we have $(v, x) \models \mathit{no\_reject}$, for all $(v, x) \in W$, hence

$$(\mathit{root}, \mathit{root}) \models K\Box \mathit{no\_reject}.$$

We still need to show that

$$\begin{aligned}(\mathit{root}, \mathit{root}) \models\ & K\Box \mathit{initial\_symbols} \wedge K\Box \mathit{written\_symbols} \\ & \wedge K\Box \mathit{read\_a\_symbol} \wedge K\Box \mathit{computation}.\end{aligned}$$

It is sufficient to show that, for all $(v, x) \in W$,

$$(v, x) \models \mathit{initial\_symbols} \wedge \mathit{written\_symbols} \wedge \mathit{read\_a\_symbol} \wedge \mathit{computation}.$$

First, let us consider the formula *initial_symbols*. We wish to show that

$$(v, x) \models \mathit{initial\_symbols},$$

for all $(v, x) \in W$. Nothing needs to be shown if $(v, x) \models (\underline{X}^{\text{time-apv}} = \mathrm{bin}_N(0))$ is not true. So, let us assume that $(v, x) \models (\underline{X}^{\text{time-apv}} = \mathrm{bin}_N(0))$. We noted above that the assumption $(v, x) \models (\underline{X}^{\text{time-apv}} = \mathrm{bin}_N(0))$ implies that the cell $\mathit{pos}(x)$ has not been visited before on the computation path from root to x. Hence, $\mathit{read}(x) = \eta$, where η is the initial symbol in the cell x. We obtain $(v, x) \models X_\eta^{\mathrm{read}}$. So, if the number $i := \mathit{pos}(x) - (2^N - 1)$ satisfies $1 \leqslant i \leqslant n$ then $(v, x) \models X_{w_i}^{\mathrm{read}}$, otherwise $(v, x) \models X_{\#}^{\mathrm{read}}$. Remember that $(v, x) \models (\underline{X}^{\mathrm{pos}} = \mathrm{bin}_{N+1}(\mathit{pos}(x))$. We have shown $(v, x) \models \mathit{initial\_symbols}$. Next, we consider the formula *written_symbols* and show that

$$(v, x) \models \mathit{written\_symbols}$$

for all $(v, x) \in W$. Let us assume

$$(v, x) \models ((\underline{X}^{\text{time-apv}} > \mathrm{bin}_N(0)) \wedge (\underline{X}^{\text{time-apv}} = \underline{\alpha}^{\mathrm{time}}) \wedge B^{\mathrm{active}})$$

(otherwise, nothing needs to be shown). The condition $(v, x) \models (\underline{X}^{\text{time-apv}} > \mathrm{bin}_N(0))$ implies $x \in V \backslash \{\mathit{root}\}$ and that the binary value t of the vector $\underline{X}^{\text{time-apv}}$ in the point (v, x) is equal to 1+ the time of the previous visit of the

cell *pos*(x) on the computation path from *root* to x. Note that $t \leqslant \mathit{time}(x)$. The condition $(v, x) \models (\underline{X}^{\text{time-apv}} = \underline{\alpha}^{\text{time}})$ means $t = \mathit{time}(v)$. Now the condition $(v, x) \models B^{\text{active}}$ implies that v is an element of the path from *root* to x. Actually, due to $t = \mathit{time}(v)$ the node v is exactly the computation node on the computation path from *root* to x after the previous visit of the cell *pos*(x). Thus, the symbol written during this visit and described by the value of $\underline{\alpha}^{\text{written}}$ at the point (v, x) is just the symbol still contained in the same cell when the computation node x is reached. Hence, we have $(v, x) \models (\underline{X}^{\text{read}} = \underline{\alpha}^{\text{written}})$. For later purposes we note that for the same reason we have $(v, x) \models (\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{prevpos}})$ as well. We have shown $(v, x) \models$ *written_symbols*.
Next, we consider the formula *read_a_symbol*. We wish to show that

$$(v, x) \models \mathit{read\_a\_symbol},$$

for all $(v, x) \in W$. We show this separately for the five subformulas of *read_a_symbol*. First, we show

$$(v, x) \models \mathit{existence\_of\_a\_reading\_point},$$

that is,

$$(v, x) \models L\big((\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{pos}}) \wedge (\underline{X}^{\text{time-apv}} \leqslant \underline{\alpha}^{\text{time}}) \wedge B^{\text{active}}\big).$$

Indeed, it is clear that $(v, x) \xrightarrow{L} (v, v)$ and that $(v, v) \models (\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{pos}})$ and $(v, v) \models B^{\text{active}}$. Furthermore, the binary value of $\underline{X}^{\text{time-apv}}$ in the point (v, v) is either 1+ the time of the previous visit of the cell *pos*(v) on the path from *root* to v, if this cell has been visited before v on this path, or 0, otherwise. In any case we obtain $(v, v) \models (\underline{X}^{\text{time-apv}} \leqslant \underline{\alpha}^{\text{time}})$. Next, we show

$$(v, x) \models \mathit{time\_of\_previous\_visit},$$

that is,

$$\begin{aligned}(v, x) \models \ & \Big(\big((\underline{X}^{\text{time-apv}} > \mathrm{bin}_N(0)) \wedge (\underline{X}^{\text{time-apv}} = \underline{\alpha}^{\text{time}}) \wedge B^{\text{active}}\big) \\ & \quad \rightarrow (\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{prevpos}})\Big).\end{aligned}$$

Actually, we have seen this already in the proof of $(v, x) \models$ *written_symbols* above. Next, we show

$$(v, x) \models \mathit{becoming\_inactive},$$

that is,

$$(v,x) \models \Big(\big((\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{prevpos}}) \wedge (\underline{X}^{\text{time-apv}} < \underline{\alpha}^{\text{time}}) \big) \to \neg B^{\text{active}} \Big)$$

For the sake of a contradiction, let us assume $(v,x) \models ((\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{prevpos}}) \wedge (\underline{X}^{\text{time-apv}} < \underline{\alpha}^{\text{time}}))$ and $(v,x) \models B^{\text{active}}$. Then v is a node on the path from *root* to x. Due to $(v,x) \models (\underline{X}^{\text{time-apv}} < \underline{\alpha}^{\text{time}})$ we have $\mathit{time}(v) > 0$. And due to $(v,x) \models (\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{prevpos}})$ the node v must have the property $\mathit{pos}(\mathit{pred}(v)) = \mathit{pos}(x)$. That means that the cell $\mathit{pos}(x)$ has been visited before x on the path from *root* to x. But under these circumstances, the binary value t of $\underline{X}^{\text{time-apv}}$ at the point (v,x) is equal to $\mathit{time}(u)$ where u is the last node on the path from *root* to x with the property $\mathit{pos}(\mathit{pred}(u)) = \mathit{pos}(x)$. Note that v is a node on the path from *root* to x with the property $\mathit{pos}(\mathit{pred}(u)) = \mathit{pos}(x)$. On the other hand, the condition $(v,x) \models (\underline{X}^{\text{time-apv}} < \underline{\alpha}^{\text{time}})$ implies $t < \mathit{time}(v)$. That is a contradiction. We have shown $(v,x) \models \mathit{becoming\_inactive}$. Next,we show

$$(v,x) \models \mathit{staying\_inactive},$$

that is,

$$(v,x) \models \Big(\neg B^{\text{active}} \to \Box \neg B^{\text{active}} \Big).$$

This is clear from the definition of $\sigma(B^{\text{active}})$.
We come to the last subformula of *read_a_symbol* and show

$$(v,x) \models \mathit{storing\_the\_read\_symbol},$$

that is,

$$(v,x) \models \Big(\big((\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{pos}}) \wedge (\underline{X}^{\text{time-apv}} \leqslant \underline{\alpha}^{\text{time}}) \wedge B^{\text{active}} \big) \to (\underline{\alpha}^{\text{read}} = \underline{X}^{\text{read}}) \Big).$$

The condition $(v,x) \models B^{\text{active}}$ implies that v is a node on the path from *root* to x. The condition $(v,x) \models (\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{pos}})$ says that $\mathit{pos}(x) = \mathit{pos}(v)$. We claim that $v = x$. Once we have shown this, of course, we obtain $(v,x) \models (\underline{\alpha}^{\text{read}} = \underline{X}^{\text{read}})$. For the sake of a contradiction, let us assume $v \neq x$. Then the cell $\mathit{pos}(x)$ has been visited before x on the path from *root* to x. Let u be the last node before x on the path from *root* to x with $\mathit{pos}(u) = \mathit{pos}(x)$. We obtain $\mathit{time}(v) \leqslant \mathit{time}(u)$. For the binary value t of $\underline{X}^{\text{time-apv}}$ in (v,x) we obtain $t = 1 + \mathit{time}(u)$. Finally, the condition $(v,x) \models (\underline{X}^{\text{time-apv}} \leqslant \underline{\alpha}^{\text{time}})$ implies $t \leqslant \mathit{time}(v)$. By putting all this together we arrive at the following contradiction:

$$\mathit{time}(v) \leqslant \mathit{time}(u) < 1 + \mathit{time}(u) = t \leqslant \mathit{time}(v).$$

We have shown $(v, x) \models \mathit{storing\_the\_read\_symbol}$.
Finally, we have to show that

$$(v, x) \models \mathit{computation},$$

for all $(v, x) \in W$. We will separately treat the conjunctions over the set $(q, \eta) \in Q_\exists \times \Gamma$ and over the set $Q_\forall \times \Gamma$. Let us first fix a pair $(q, \eta) \in Q_\exists \times \Gamma$ and let us assume that $(v, x) \in W$ is a point with $(v, x) \models (\alpha_q^{\text{state}} \wedge \alpha_\eta^{\text{read}})$. We have to show that there is an element $(r, \theta, \mathit{left}) \in \delta(q, \eta)$ such that $(v, x) \models \mathit{compstep}_{\text{left}}(r, \theta)$ or that there is an element $(r, \theta, \mathit{right}) \in \delta(q, \eta)$ such that $(v, x) \models \mathit{compstep}_{\text{right}}(r, \theta)$. As T is an accepting tree and the state q of $c(v)$ is an element of $Q_\exists$, the node v is an inner node of T, hence, it has a successor v'. Let us assume that $((q, \eta), (r, \theta, \mathit{left})) \in \delta$ is the element of the transition relation δ that leads from v to v' (the case that this element is of the form $((q, \eta), (r, \theta, \mathit{right}))$ is treated analogously). We claim that then

$$(v, x) \models \mathit{compstep}_{\text{left}}(r, \theta).$$

In fact, we observe $(v, x) \xrightarrow{L} (v, v')$ and $(v, v') \xrightarrow{\Diamond} (v', v')$. We claim that the two points (v, v') and (v', v') have the properties formulated in the formula $\mathit{compstep}_{\text{left}}(r, \theta)$. Let us check this. Let us assume that, for some $k \in \{0, \ldots, N-1\}$ and for some $l \in \{0, \ldots, N\}$,

$$(v, x) \models \big(\text{rightmost\_zero}(\underline{\alpha}^{\text{time}}, k) \wedge \text{rightmost\_one}(\underline{\alpha}^{\text{pos}}, l)\big).$$

The number $i := \mathit{time}(v)$ is an element of $\{0, \ldots, 2^N - 2\}$ because v is an inner point of the tree T and the length of any computation path is at most $2^N - 1$. The number $j := \mathit{pos}(v)$ is an element of $\{1, \ldots, 2^{N+1} - 3\}$ because the computation starts in cell $2^N - 1$ and during each computation step the tape head can move at most one step to the left or to the right. We obtain $k = \min(\{0, \ldots, N-1\} \backslash \text{Ones}(i))$ and $l = \min \text{Ones}(j)$. As $v = \mathit{pred}(v')$ we obtain

$$(v, v') \models (\underline{X}^{\text{prevtime}} = \underline{\alpha}^{\text{time}})$$

and

$$(v, v') \models (\underline{X}^{\text{prevpos}} = \underline{\alpha}^{\text{pos}}).$$

And as $\mathit{time}(v') = \mathit{time}(v) + 1 = i + 1$ and $\mathit{pos}(v') = \mathit{pos}(v) - 1 = j - 1$ we conclude that

$$\begin{aligned}(v', v') \quad \models \quad \Big(& (\underline{\alpha}^{\text{time}} = \underline{X}^{\text{prevtime}}, > k) \wedge \text{rightmost\_one}(\underline{\alpha}^{\text{time}}, k) \\ & \wedge (\underline{\alpha}^{\text{pos}} = \underline{X}^{\text{prevpos}}, > l) \wedge \text{rightmost\_zero}(\underline{\alpha}^{\text{pos}}, l)\Big).\end{aligned}$$

Finally, the condition

$$(v', v') \models (\underline{\alpha}^{\text{prevpos}} = \underline{X}^{\text{prevpos}})$$

is obviously satisfied and the condition

$$(v', v') \models \alpha_r^{\text{state}} \wedge \alpha_\theta^{\text{written}}$$

follows from the fact that $((q, \eta), (r, \theta, \mathit{left})) \in \delta$ is the element of the transition relation δ that leads from v to v'. This ends the treatment of the conjunctions over the set $(q, \eta) \in Q_\exists \times \Gamma$ in the formula *computation*. Let us now consider a pair $(q, \eta) \in Q_\forall \times \Gamma$. Let us assume that $(v, x) \in W$ is a point such that $(v, x) \models (\alpha_q^{\text{state}} \wedge \alpha_\eta^{\text{read}})$. We have to show that for all elements $(r, \theta, \mathit{left}) \in \delta(q, \eta)$ we have $(v, x) \models \mathit{compstep}_{\text{left}}(r, \theta)$ and for all elements $(r, \theta, \mathit{right}) \in \delta(q, \eta)$ we have $(v, x) \models \mathit{compstep}_{\text{right}}(r, \theta)$. Let us consider an arbitrary element $(r, \theta, \mathit{left}) \in \delta(q, \eta)$ (the case of an element $(r, \theta, \mathit{right}) \in \delta(q, \eta)$ is treated analogously). As $q \in Q_\forall$ and T is an accepting tree, in T there is a successor v' of v such that the element $((q, \eta), (r, \theta, \mathit{left}))$ leads from v to v'. Above, we have already seen that this implies

$$(v, x) \models \mathit{compstep}_{\text{left}}(r, \theta).$$

Thus, we have shown $(v, x) \models \mathit{computation}$ for all $(v, x) \in W$. This ends the proof of the claim that in the S4 × S5-product model $(W_1 \times W_2, \xrightarrow{\Diamond}, \xrightarrow{L}, \sigma)$ that we constructed for $w \in L$ we have $(\mathit{root}, \mathit{root}) \models f_{\text{S4}\times\text{S5}}(w)$.

7.4 Existence of an Accepting Tree

We come to the other direction. Let $w \in \Sigma^*$. We wish to show that if $f_{\text{S4}\times\text{S5}}(w)$ is S4 × S5-satisfiable then $w \in L$. We will show that any S4 × S5-product model essentially contains an accepting tree of the Alternating Turing Machine M on input w. Of course, this implies $w \in L$.

Let us sketch the main idea. We will consider an S4 × S5-product model of $f_{\text{S4}\times\text{S5}}(w)$. And we will consider partial trees of M on input w as considered in Section 5.3, for any $w \in \Sigma^*$. First, we will show that a certain very simple partial tree of M on input w "can be mapped to" the model (later we will give a precise meaning to "can be mapped to"). Then we will show that any partial tree of M on input w that can be mapped to the model and that is not an accepting tree of M on input w can be properly extended to a strictly larger partial tree of M on input w that can be mapped to the model as well. If there would not exist an accepting tree of M on input w then we would

obtain an infinite strictly increasing sequence of partial trees of M on input w. But we show that this cannot happen by giving a finite upper bound on the size of partial trees of M on input w.
Let $w \in \Sigma^*$ be a string such that the formula $f_{\mathrm{S4}\times\mathrm{S5}}(w)$ is S4 × S5-satisfiable. We set $n := |w|$. Let $(W_1, R_\Diamond)$ be an $S4$-frame, let (W_2, R_L) be an $S5$-frame, let $\sigma : AT \to \mathcal{P}(W_1 \times W_2)$ be a function such that the quadruple $(W_1 \times W_2, \overset{\Diamond}{\to}, \overset{L}{\to}, \sigma)$ where $\overset{\Diamond}{\to}$ and $\overset{L}{\to}$ are defined as in Definition 3.10 is an S4 × S5-product model, and let $(r_1, r_2) \in W_1 \times W_2$ be a point with $(r_1, r_2) \models f_{\mathrm{S4}\times\mathrm{S5}}(w)$. The quintuple

$$\mathit{Model} := (W_1 \times W_2, \overset{\Diamond}{\to}, \overset{L}{\to}, \sigma, (r_1, r_2))$$

will be important in the following. We claim that we can assume without loss of generality that $r_1 \overset{\Diamond}{\to} x$ for all $x \in W_1$ and $R_L = W_2 \times W_2$. Otherwise, instead of W_1 we could consider the set $W_1' := \{v \in W_1 \mid r_1 \overset{\Diamond}{\to} v\}$ and instead of W_2 we could consider the set $W_2' :=$ the R_L-equivalence class of r_2 and the restrictions $\overset{L}{\to}'$ resp. $\overset{\Diamond}{\to}'$ resp. σ' of $\overset{L}{\to}$ resp. $\overset{\Diamond}{\to}$ resp. σ to $W_1' \times W_2'$. By structural induction one shows that for any bimodal formula φ and for any $(v, x) \in W_1' \times W_2'$,

$$(W_1 \times W_2, \overset{\Diamond}{\to}, \overset{L}{\to}, \sigma), (v, x) \models \varphi \iff (W_1' \times W_2', \overset{\Diamond}{\to}', \overset{L}{\to}', \sigma'), (v, x) \models \varphi.$$

So, we shall assume that $r_1 \overset{\Diamond}{\to} x$, for all $v \in W_1$, and $R_L = W_2 \times W_2$. Note that this implies that if φ is a formula with $(r_1, r_2) \models K\Box\varphi$ then, for all $(v, x) \in W_1 \times W_2$, we have $(v, x) \models \varphi$.
For every $v \in W_1$, the set

$$\mathrm{Cloud}(v) := \{(v, x) \ : \ v \in W_1, x \in W_2\}$$

is the $\overset{L}{\to}$-equivalence class (short: *cloud*) of any element $y \in W_1 \times W_2$ whose first component is v. Remember that for every $\overset{L}{\to}$-equivalence class and every shared variable $\alpha_i^{\mathit{string}}$ for *string* $\in$ {time, pos, state, read, prevpos, written} and natural numbers i, the truth value of this shared variable is the same in all elements of the $\overset{L}{\to}$-equivalence class.
Partial trees of M on input w as introduced in Section 5.3 will play an important role in the following. We will write a partial tree of M on input w similarly as in Section 5.3 as a triple $T = (V, E, c)$, but with the difference that we will describe configurations as at the beginning of Section 7.1: the labeling function c will be a function of the form $c : V \to Q \times \{0, \ldots, 2^{N+1} - 2\} \times \Gamma^{2^{N+1}-1}$. If $T = (V, E, c)$ is a partial tree of M on input w with root *root* then a function $\pi : V \to W_1$ is called a *morphism from T to Model* if it satisfies the following four conditions:

1. $\pi(root) = r_1$.
2. $\{(\pi(v), \pi(v')) \ : \ v, v' \in V \text{ and } vEv'\} \subseteq R_\Diamond$.
3. $(\forall v \in V \backslash \{root\}) \, (\exists x \in W_2)$.
   $(\pi(v), x) \models \big((\underline{\alpha}^{\text{prevpos}} = \text{bin}_{N+1}(pos(pred(v)))) \wedge \alpha^{\text{written}}_{written(v)}\big)$
4. $(\forall v \in V) \, (\exists x \in W_2)$
   $$(\pi(v), x) \models \big((\underline{\alpha}^{\text{time}} = \text{bin}_N(time(v))) \wedge (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(pos(v))) \wedge \alpha^{\text{state}}_{state(v)} \wedge \alpha^{\text{read}}_{read(v)}\big).$$

We say that T *can be mapped to Model* if there exists a morphism from T to *Model*. Below we shall prove the following lemma.

Lemma 7.2. *If a partial tree $T = (V, E, c)$ of M on input w can be mapped to Model and is not an accepting tree of M on input w then there exists a partial tree $T = (\widetilde{V}, \widetilde{E}, \widetilde{c})$ of M on input w that can be mapped to Model and that satisfies $V \subsetneq \widetilde{V}$.*

Before we prove this lemma, we deduce the desired assertion from it. Let

$$D := \max(\{|\delta(q, \eta)| \ : \ q \in Q, \eta \in \Gamma\}).$$

Then, due to Condition III in the definition of a “partial tree of M on input w”, any node in any partial tree of M on input w has at most D successors. As any computation of M on input w stops after at most $2^N - 1$ steps, any partial tree of M on input w has at most

$$\widetilde{D} := (D^{2^N} - 1)/(D - 1)$$

nodes.

We claim that the rooted and labeled tree

$$T_0 := (\{root\}, \emptyset, c) \text{ where } c(root) := (q_0, 2^N - 1, \#^{2^N} w \#^{2^N - 1 - n})$$

is a partial tree of M on input w and that it can be mapped to *Model*. Indeed, Condition I in the definition of a “partial tree of M on input w” is satisfied because the node *root* is labeled with the initial configuration of M on input w. Conditions II, III, and IV are satisfied because T_0 does not have any inner nodes. Condition V′ is satisfied because $state(root) = q_0$ and, due to $(r_1, r_2) \models \alpha^{\text{state}}_{q_0}$ (this is a part of $(r_1, r_2) \models start$) and $(r_1, r_2) \models \neg\alpha^{\text{state}}_{q_{\text{reject}}}$ (this follows from $(r_1, r_2) \models K\Box no\_reject$) we obtain $q_0 \neq q_{\text{reject}}$. Thus, T_0

is indeed a partial tree of M on input w. Now we show that T_0 can be mapped to *Model*. Of course, we define the function $\pi : \{\mathit{root}\} \to W_1$ by $\pi(\mathit{root}) := r_1$.

1. The condition $\pi(\mathit{root}) = r_1$ is true by definition.

2. The tree T_0 does not have any edges, that is, its set E of edges is empty. So, the second condition is satisfied.

3. The third condition does not apply to the tree T_0 because T_0 has only one node, its root.

4. On the one hand, we have

$$\mathit{time}(\mathit{root}) = 0, \qquad \mathit{pos}(\mathit{root}) = 2^N - 1,$$

$$\mathit{state}(\mathit{root}) = q_0, \qquad \text{and } \mathit{read}(\mathit{root}) = \#.$$

On the other hand, the condition $(r_1, r_2) \models \mathit{start}$ says

$$(r_1, r_2) \models \big((\underline{\alpha}^{\text{time}} = \text{bin}_N(0)) \wedge (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(2^N - 1)) \wedge \alpha^{\text{state}}_{q_0}.$$

We still wish to show $(r_1,r_2) \models \alpha^{\text{read}}_{\#}$. We have $(r_1,r_2) \models \mathit{read\_a\_symbol}$. This implies $(r_1, r_2) \models \mathit{existence\_of\_a\_reading\_point}$, and this implies that there exists some $y \in W_2$ with

$$(r_1, y) \models \big((\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{pos}}) \wedge (\underline{X}^{\text{time-apv}} \leqslant \underline{\alpha}^{\text{time}}) \wedge B^{\text{active}}\big),$$

hence, with

$$(r_1, y) \models \big((\underline{X}^{\text{pos}} = \text{bin}_{N+1}(2^N - 1)) \wedge (\underline{X}^{\text{time-apv}} = \text{bin}(0)) \wedge B^{\text{active}}\big).$$

Now $(r_1, y) \models \mathit{initial\_symbols}$ implies $(r_1, y) \models X^{\text{read}}_{\#}$. Finally, the condition $(r_1, y) \models \mathit{storing\_the\_read\_symbol}$ implies $(r_1, y) \models \alpha^{\text{read}}_{\#}$, and this implies $(r_1, r_2) \models \alpha^{\text{read}}_{\#}$.

If there would not exist an accepting tree of M on input w then, starting with T_0 and using Lemma 7.2, we could construct an infinite sequence of partial trees $T_0, T_1, T_2, \ldots$ of M on input w that can be mapped to *Model* such that the number of nodes in these trees is strictly increasing. But we have seen that any partial tree of M input w can have at most $\widetilde{D}$ nodes. Thus, there exists an accepting tree of M on input w. We have shown $w \in L$.

In order to complete the proof of Theorem 7.1 it remains to prove Lemma 7.2.

Proof of Lemma 7.2. Let $T = (V, E, c)$ be a partial tree of M on input w that is not an accepting tree of M on input w and that can be mapped to *Model*. Then T has a leaf $\widehat{v}$ such that the state $q := \mathit{state}(\widehat{v})$ is either an element of $Q_\exists$ or of $Q_\forall$. First we treat the case that it is an element of $Q_\exists$, then the case that it is an element of $Q_\forall$. Let $\pi : V \to W_1$ be a morphism from T to *Model*.
So, let us assume that $q \in Q_\exists$. We define $\eta := \mathit{read}(\widehat{v})$. As $\pi : V \to W_1$ is a morphism from T to *Model* there exists an $x' \in W_2$ with

$$(\pi(\widehat{v}), x') \models (\underline{\alpha}^{\text{time}} = \text{bin}_N(\mathit{time}(\widehat{v}))) \wedge (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(\mathit{pos}(\widehat{v}))) \wedge \alpha_q^{\text{state}} \wedge \alpha_\eta^{\text{read}}.$$

Hence, due to $(\pi(\widehat{v}), x') \models \mathit{computation}$,

$$(\pi(\widehat{v}), x') \models \bigvee_{(r,\theta,\mathit{left}) \in \delta(q,\eta)} \mathit{compstep}_{\text{left}}(r, \theta) \vee \bigvee_{(r,\theta,\mathit{right}) \in \delta(q,\eta)} \mathit{compstep}_{\text{right}}(r, \theta).$$

Let us assume that there is an element $(r, \theta, \mathit{left}) \in \delta(q, \eta)$ such that $(\pi(\widehat{v}), x') \models \mathit{compstep}_{\text{left}}(r, \theta)$ (the other case, when there is an element $(r, \theta, \mathit{right}) \in \delta(q, \eta)$ such that $(\pi(\widehat{v}), x') \models \mathit{compstep}_{\text{right}}(r, \theta)$, is treated analogously). We claim that we can define the new tree $\widetilde{T} = (\widetilde{V}, \widetilde{E}, \widetilde{c})$ as follows:

- $\widetilde{V} := V \cup \{\widetilde{v}\}$ where $\widetilde{v}$ is a new element (not in V),
- $\widetilde{E} := E \cup \{(\widehat{v}, \widetilde{v})\}$,
- $\widetilde{c}(x) := \begin{cases} c(x) & \text{for all } x \in V, \\ c' & \text{for } x = \widetilde{v}, \\ & \text{where } c' \text{ is the configuration that is reached from } c(\widehat{v}) \\ & \text{in the computation step given by } ((q, \eta), (r, \theta, \mathit{left})) \in \delta. \end{cases}$

We have to show that $\widetilde{T}$ is a partial tree of M on input w. Condition I in the definition of "a partial tree of M on input w" is satisfied because $\widetilde{T}$ has the same root as T, and the label of the root does not change. A node in $\widetilde{T}$ is an internal node of $\widetilde{T}$ if, and only if, it is either an internal node of T or equal to $\widehat{v}$. For internal nodes of T Conditions II, III, and IV are satisfied by assumption (and due to the fact that the labels of nodes in V do not change when moving from T to $\widetilde{T}$). The new internal node $\widehat{v}$ satisfies Condition II by our definition of $\widetilde{c}(\widetilde{v})$. Condition III is satisfied for $\widehat{v}$ because $\widehat{v}$ has exactly one successor. And Condition IV does not apply to $\widehat{v}$ because $\widehat{v} \in Q_\exists$. A node in $\widetilde{T}$ is a leaf if, and only if, it is either equal to $\widetilde{v}$ or a leaf in T different from $\widehat{v}$. For the leaves in T different from $\widehat{v}$ Condition V′ is satisfied by assumption (and due to the fact that the labels of nodes in V do not change). Finally, we have to show that Condition V′ is satisfied for the new leaf $\widetilde{v}$ in $\widetilde{T}$ as well. We postpone this until after the definition of a morphism from $\widetilde{T}$ to *Model*.

We also have to show that $\widetilde{T}$ can be mapped to *Model*. Let us define a function $\widetilde{\pi} : \widetilde{V} \to W_1$ that we will show to be a morphism from $\widetilde{T}$ to *Model*. As $\widehat{v}$ is an element of a partial tree of M on input w with $\mathit{state}(\widehat{v}) \in Q_\exists$, at least one more computation step can be done. As any computation of M on input w stops after at most $2^N - 1$ steps, we observe that the number $i := \mathit{time}(\widehat{v})$ satisfies $0 \leqslant i < 2^N - 1$. Then $\{0, \ldots, N-1\} \backslash \mathrm{Ones}(i) \neq \varnothing$. We set $k := \min(\{0, \ldots, N-1\} \backslash \mathrm{Ones}(i))$. Together with $(\pi(\widehat{v}), x') \models (\underline{\alpha}^{\mathrm{time}} = \mathrm{bin}_N(i))$ we conclude $(\pi(\widehat{v}), x') \models \mathrm{rightmost\_zero}(\underline{\alpha}^{\mathrm{time}}, k)$. As during each computation step, the tape head can move at most one step to the left or to the right and as the computation started in position $2^N - 1$ the number $j := \mathit{pos}(\widehat{v})$ satisfies $0 < j \leqslant 2^{N+1} - 3$. Then $\mathrm{Ones}(j) \neq \varnothing$. We set $l := \min \mathrm{Ones}(j)$. Together with $(\pi(\widehat{v}), x') \models (\underline{\alpha}^{\mathrm{pos}} = \mathrm{bin}_{N+1}(j))$ we conclude $(\pi(\widehat{v}), x') \models \mathrm{rightmost\_one}(\underline{\alpha}^{\mathrm{pos}}, l)$. Thus, we have

$$(\pi(\widehat{v}), x') \models \big(\mathrm{rightmost\_zero}(\underline{\alpha}^{\mathrm{time}}, k) \wedge \mathrm{rightmost\_one}(\underline{\alpha}^{\mathrm{pos}}, l)\big).$$

Due to $(\pi(\widehat{v}), x') \models \mathit{compstep}_{\mathrm{left}}(r, \theta)$ there exist an element $y \in W_2$ and an element $x \in W_1$ such that $\pi(\widehat{v}) R_\Diamond x$ as well as

$$(\pi(\widehat{v}), y) \models \big((\underline{X}^{\mathrm{prevtime}} = \underline{\alpha}^{\mathrm{time}}) \wedge (\underline{X}^{\mathrm{prevpos}} = \underline{\alpha}^{\mathrm{pos}})\big)$$

and

$$\begin{aligned}(x, y) \models \quad & \Big((\underline{\alpha}^{\mathrm{time}} = \underline{X}^{\mathrm{prevtime}}, > k) \wedge \mathrm{rightmost\_one}(\underline{\alpha}^{\mathrm{time}}, k) \\ & \wedge (\underline{\alpha}^{\mathrm{pos}} = \underline{X}^{\mathrm{prevpos}}, > l) \wedge \mathrm{rightmost\_zero}(\underline{\alpha}^{\mathrm{pos}}, l) \\ & \wedge (\underline{\alpha}^{\mathrm{prevpos}} = \underline{X}^{\mathrm{prevpos}}) \wedge \alpha_r^{\mathrm{state}} \wedge \alpha_\theta^{\mathrm{written}}\Big)\end{aligned}$$

We claim that we can define the desired function $\widetilde{\pi} : \widetilde{V} \to W_1$ by

$$\widetilde{\pi}(v) := \begin{cases} \pi(v) & \text{if } v \in V, \\ x & \text{if } v = \widetilde{v}. \end{cases}$$

Before we show that $\widetilde{\pi}$ is a morphism from $\widetilde{T}$ to *Model*, let us complete the proof that $\widetilde{T}$ is a partial tree of M on input w. We still need to show that Condition V′ is satisfied for the new leaf $\widetilde{v}$ in $\widetilde{T}$ as well. It is sufficient to show that the state r of the configuration c' is not the rejecting state q_{reject}. But this follows from $(x, y) \models \alpha_r^{\mathrm{state}}$ and $(x, y) \models \neg\alpha_{q_{\mathrm{reject}}}^{\mathrm{state}}$ (this follows from

$(r_1, r_2) \models K\Box no\_reject)$. We have shown that $\widetilde{T}$ is a partial tree of M on input w.
Now we show that $\widetilde{\pi}$ is a morphism from $\widetilde{T}$ to *Model*. The first condition in the definition of a "morphism from $\widetilde{T}$ to *Model*" is satisfied because $\widetilde{\pi}(root) = \pi(root) = r_1$. For the second condition let us consider $v, v' \in \widetilde{V}$ with $v\widetilde{E}v'$. We have to show $\pi(v)R_\Diamond\pi(v')$. There are two possible cases.

- In the case $v, v' \in V$ we have vEv' and, hence, $\pi(v)R_\Diamond\pi(v')$. As $\widetilde{\pi}(v) = \pi(v)$ and $\widetilde{\pi}(v') = \pi(v')$ we obtain $\widetilde{\pi}(v)R_\Diamond\widetilde{\pi}(v')$.

- The other possible case is $v = \widehat{v}$ and $v' = \widetilde{v}$. But we know $\widetilde{\pi}(\widehat{v}) = \pi(\widehat{v})$, $\widetilde{\pi}(\widetilde{v}) = x$, and $\pi(\widehat{v})R_\Diamond x$.

Next, we verify that the third condition in the definition of a "morphism from $\widetilde{T}$ to *Model*" is satisfied. For $v \in V\backslash\{root\}$ it is satisfied by assumption (and by $\widetilde{\pi}(v) = \pi(v)$ and $\widetilde{c}(v) = c(v)$). For $\widetilde{v}$ it is sufficient to show that

$$(x, y) \models \big((\underline{\alpha}^{\text{prevpos}} = \text{bin}_{N+1}(pos(pred(\widetilde{v})))) \wedge \alpha^{\text{written}}_{written(\widetilde{v})}\big)$$

(remember that $\widetilde{\pi}(\widetilde{v}) = x$). These are really two conditions. We prove them separately.

- In the tree $\widetilde{T}$ we have $pos(pred(\widetilde{v})) = pos(\widehat{v}) = j$, and in T we have $pos(\widehat{v}) = j$ as well. The assumption $(\pi(\widehat{v}), x') \models (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(j))$, for some $x' \in W_2$, implies $(\pi(\widehat{v}), y) \models (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(j))$, and the conditions $(\pi(\widehat{v}), y) \models (\underline{X}^{\text{prevpos}} = \underline{\alpha}^{\text{pos}})$ and $(x, y) \models (\underline{\alpha}^{\text{prevpos}} = \underline{X}^{\text{prevpos}})$ as well as the persistence of $\underline{X}^{\text{prevpos}}$ imply $(x, y) \models (\underline{\alpha}^{\text{prevpos}} = \text{bin}_{N+1}(j))$.

- In $\widetilde{T}$ we have $written(\widetilde{v}) = \theta$. And we have $(x, y) \models \alpha^{\text{written}}_\theta$, hence, $(x, y) \models \alpha^{\text{written}}_{written(\widetilde{v})}$.

We come to the fourth condition in the definition of a "morphism from $\widetilde{T}$ to *Model*". It is satisfied for $v \in V$ by assumption (and due to $\widetilde{\pi}(v) = \pi(v)$ and $\widetilde{c}(v) = c(v)$). We still need to show that it is satisfied for $v = \widetilde{v}$. Remember $\widetilde{\pi}(\widetilde{v}) = x$. It is sufficient to show

$$\begin{aligned}(x, y) \models \; & (\underline{\alpha}^{\text{time}} = \text{bin}_N(time(\widetilde{v}))) \wedge (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(pos(\widetilde{v}))) \\ & \wedge \alpha^{\text{state}}_{state(\widetilde{v})} \wedge \alpha^{\text{read}}_{read(\widetilde{v})}.\end{aligned}$$

This assertion consists really of four assertions. We treat them one by one.

- In the trees T and $\widetilde{T}$ we have $time(\widehat{v}) = i$, and in the tree $\widetilde{T}$ we have $time(\widetilde{v}) = i + 1$. We have already seen $(\pi(\widehat{v}), x') \models (\underline{\alpha}^{\text{time}} = \text{bin}_N(i))$ and $(\pi(\widehat{v}), x') \models \text{rightmost\_zero}(\underline{\alpha}^{\text{time}}, k)$. Of course, we get

  $$(\pi(\widehat{v}), y) \models ((\underline{\alpha}^{\text{time}} = \text{bin}_N(i)) \wedge \text{rightmost\_zero}(\underline{\alpha}^{\text{time}}, k)).$$

  The conditions

  $$\begin{aligned} (\pi(\widehat{v}), y) &\models (\underline{X}^{\text{prevtime}} = \underline{\alpha}^{\text{time}}), \\ (x, y) &\models (\underline{\alpha}^{\text{time}} = \underline{X}^{\text{prevtime}}, > k) \wedge \text{rightmost\_one}(\underline{\alpha}^{\text{time}}, k) \end{aligned}$$

  and the persistence of $\underline{X}^{\text{prevtime}}$ imply $(x, y) \models (\underline{\alpha}^{\text{time}} = \text{bin}_N(i + 1))$.

- In the trees T and $\widetilde{T}$ we have $pos(\widehat{v}) = j$, and in the tree $\widetilde{T}$ we have $pos(\widetilde{v}) = j - 1$. We have already seen $(\pi(\widehat{v}), x') \models (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(j))$, and $(\pi(\widehat{v}), x') \models rightmost\_one(\underline{\alpha}^{\text{pos}}, l)$. Of course, we get

  $$(\pi(\widehat{v}), y) \models ((\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(j)) \wedge \text{rightmost\_one}(\underline{\alpha}^{\text{pos}}, l)).$$

  The conditions

  $$\begin{aligned} (\pi(\widehat{v}), y) &\models (\underline{X}^{\text{prevpos}} = \underline{\alpha}^{\text{pos}}), \\ (x, y) &\models (\underline{\alpha}^{\text{pos}} = \underline{X}^{\text{prevpos}}, > l) \wedge \text{rightmost\_zero}(\underline{\alpha}^{\text{pos}}, l) \end{aligned}$$

  and the persistence of $\underline{X}^{\text{prevpos}}$ imply $(x, y) \models (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(j - 1))$.

- We have $state(\widetilde{v}) = r$. And we have $(x, y) \models \alpha_r^{state}$.

- Let $\gamma := read(\widetilde{v})$ in $\widetilde{T}$. We wish to show $(x, y) \models \alpha_\gamma^{\text{read}}$. We remark that the proof is a formal version of the informal explanation after the definition of the formula *read_a_symbol*. It is sufficient to show that there is some $z \in W_2$ with $(x, z) \models \alpha_\gamma^{\text{read}}$. The condition $(x, y) \models$ *existence_of_a_reading_point* implies that there exists some $z \in W_2$ such that

  $$(x, z) \models \big((\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{pos}}) \wedge (\underline{X}^{\text{time-apv}} \leqslant \underline{\alpha}^{\text{time}}) \wedge B^{\text{active}}\big).$$

  Remember that the binary value of $\underline{\alpha}^{\text{time}}$ in (x, y) and, hence, also in (x, z), is equal to $i + 1$ and that the binary value of $\underline{\alpha}^{\text{pos}}$ in (x, y) and, hence, also in (x, z), is equal to $j - 1$. Hence, the binary value of $\underline{X}^{\text{pos}}$ in (x, z) is equal to $j - 1$. Let t be the unique number in $\{0, \ldots, i + 1\}$ with $(x, z) \models (\underline{X}^{\text{time-apv}} = \text{bin}_N(t))$. Let v_t be the unique node in the computation path in $\widetilde{T}$ from *root* to $\widetilde{v}$ with $time(v_t) = t$.

First, we claim that $(\widetilde{\pi}(u), z) \models B^{\text{active}}$ for all nodes u on the path from *root* to $\widetilde{v}$. Any such u satisfies $uE^*\widetilde{v}$. We obtain $\widetilde{\pi}(u)R_\Diamond\widetilde{\pi}(\widetilde{v})$, hence, $\widetilde{\pi}(u)R_\Diamond x$. If $(\widetilde{\pi}(u), z) \models \neg B^{\text{active}}$ then due to $(\widetilde{\pi}(u), z) \models$ *staying_inactive*, we would obtain $(x, z) \models \neg B^{\text{active}}$. But this is a contradiction to the condition $(x, z) \models B^{\text{active}}$ with which we started. Thus, for all nodes on the path from *root* to $\widetilde{v}$ we have $(\widetilde{\pi}(u), z) \models B^{\text{active}}$.

Can there be a node $u \neq v_t$ in the path from v_t to $\widetilde{v}$ with $\mathit{pos}(\mathit{pred}(u)) = j - 1 = \mathit{pos}(\widetilde{v})$? We claim that this is not the case. Indeed, if there were such a u then for this node u we would have $(\widetilde{\pi}(u), z) \models (\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{prevpos}}) \wedge (\underline{X}^{\text{time-apv}} < \underline{\alpha}^{\text{time}})$. But then $(\widetilde{\pi}(u), z) \models$ *becoming_inactive* would imply $(\widetilde{\pi}(u), z) \models \neg B^{\text{active}}$ in contradiction to what we have just shown. Hence, we have shown that there is no node $u \neq v_t$ in the path from v_t to $\widetilde{v}$ with $\mathit{pos}(\mathit{pred}(u)) = j-1 = \mathit{pos}(\widetilde{v})$. Let us now distinguish the two cases $t = 0$ and $t > 0$.

First we treat the case $t = 0$. Then $v_t = \mathit{root}$. We have just seen that the cell $j - 1$ has not been visited before $\widetilde{v}$ on the path from *root* to $\widetilde{v}$. Hence, the initial symbol in the cell $j - 1$ is still the symbol in this cell when the node $\widetilde{v}$ is reached. Thus γ is the initial symbol in this cell. Due to $(x, z) \models$ *initial_symbols* we obtain $(x, z) \models X_\gamma^{\text{read}}$. Due to $(x, z) \models$ *storing_the_read_symbol* we obtain $(x, z) \models \alpha_\gamma^{\text{read}}$.

Finally, we treat the case $t > 0$. Then $v_t \neq \mathit{root}$. And we have just seen that the cell $j - 1$ has not been visited before $\widetilde{v}$ on the path from v_t to $\widetilde{v}$. But we claim that it has been visited in the predecessor of v_t. Indeed, we have $(\widetilde{\pi}(v_t), z) \models \big((\underline{X}^{\text{time-apv}} > \text{bin}_N(0)) \wedge (\underline{X}^{\text{time-apv}} = \underline{\alpha}^{\text{time}}) \wedge B^{\text{active}}\big)$. Hence, due to $(\widetilde{\pi}(v_t), z) \models$ *time_of_previous_visit*, we obtain $(\widetilde{\pi}(v_t), z) \models (\underline{X}^{\text{pos}} = \underline{\alpha}^{\text{prevpos}})$. Above we have seen that the binary value of $\underline{X}^{\text{pos}}$ in (x, z) is $j - 1$. As $\underline{X}^{\text{pos}}$ is persistent, the binary value of $\underline{X}^{\text{pos}}$ in $(\widetilde{\pi}(v_t), z)$ is $j - 1$ as well. Hence, the binary value of $\underline{\alpha}^{\text{prevpos}}$ in $(\widetilde{\pi}(v_t), z)$ is $j - 1$ as well. That implies $\mathit{pos}(\mathit{pred}(v_t)) = j-1$. We have shown that the predecessor of the node v_t is the last node before $\widetilde{v}$ in which the cell $j - 1 = \mathit{pos}(\widetilde{v})$ has been visited. Hence, the symbol γ that is read when the node $\widetilde{v}$ is reached, has been written in the computation step from $\mathit{pred}(v_t)$ to v_t. Hence, we have $(\widetilde{\pi}(v_t), z) \models \alpha_\gamma^{\text{written}}$. Due to $(\widetilde{\pi}(v_t), z) \models$ *written_symbols* we obtain $(\widetilde{\pi}(v_t), z) \models X_\gamma^{\text{written}}$. As above in the other case, due to $(x, z) \models$ *storing_the_read_symbol*, we finally obtain $(x, z) \models \alpha_\gamma^{\text{read}}$.

Thus, $\widetilde{T}$ is not only a partial tree of M on input w but can also be mapped to *Model*. This ends the treatment of the case $q \in Q_\exists$.
Now we consider the other case, the case $q \in Q_\forall$. We define $\eta := \mathit{read}(\widehat{v})$.

Let

$$(r_1, \theta_1, dir_1), \ldots, (r_d, \theta_d, dir_d)$$

be the elements of $\delta(q, \eta)$ where $d \geqslant 1$ and $dir_m \in \{\mathit{left}, \mathit{right}\}$, for $m = 1, \ldots, d$. We claim that we can define the new tree $\widetilde{T} = (\widetilde{V}, \widetilde{E}, \widetilde{c})$ as follows:

- $\widetilde{V} := V \cup \{\widetilde{v}_1, \ldots, \widetilde{v}_d\}$ where $\widetilde{v}_1, \ldots, \widetilde{v}_d$ are new (not in V) pairwise different elements,
- $\widetilde{E} := E \cup \{(\widehat{v}, \widetilde{v}_1), \ldots, (\widehat{v}, \widetilde{v}_d)\}$,
- $\widetilde{c}(x) := \begin{cases} c(x) & \text{for all } x \in V, \\ c'_m & \text{for } x = \widetilde{v}_m, \\ & \text{where } c'_m \text{ is the configuration that is reached from} \\ & c(\widehat{v}) \text{ in the computation step given by} \\ & ((q, \eta), (r_m, \theta_m, dir_m)) \in \delta. \end{cases}$

Before we show that $\widetilde{T}$ is a partial tree of M on input w, we define a function $\widetilde{\pi} : \widetilde{V} \to W_1$ that we will show to be a morphism from $\widetilde{T}$ to *Model*. Since T can be mapped to *Model*, we have

$$(\pi(\widehat{v}), x') \models (\underline{\alpha}^{\text{time}} = \text{bin}_N(\mathit{time}(\widehat{v}))) \wedge (\underline{\alpha}^{\text{pos}} = \text{bin}_{N+1}(\mathit{pos}(\widehat{v}))) \wedge \alpha_q^{\text{state}} \wedge \alpha_\eta^{\text{read}}.$$

for some $x' \in W_2$, hence, due to $(\pi(\widehat{v}), x') \models \mathit{computation}$,

$$(\pi(\widehat{v}), x') \models \bigwedge_{(r,\theta,\mathit{left}) \in \delta(q,\eta)} \mathit{compstep}_{\text{left}}(r, \theta) \wedge \bigwedge_{(r,\theta,\mathit{right}) \in \delta(q,\eta)} \mathit{compstep}_{\text{right}}(r, \theta).$$

As in the case $q \in Q_\exists$ one shows that the numbers $i := \mathit{time}(\widehat{v})$ and $j := \mathit{pos}(\widehat{v})$ satisfy $0 \leqslant i < 2^N - 1$ and $0 < j \leqslant 2^{N+1} - 3$, and one defines

$$\begin{aligned} k &:= \min(\{0, \ldots, N-1\} \backslash \text{Ones}(i)), \\ l_{\text{left}} &:= \min \text{Ones}(j), \\ l_{\text{right}} &:= \min(\{0, \ldots, N\} \backslash \text{Ones}(j)). \end{aligned}$$

As in the case $q \in Q_\exists$ one obtains

$$\begin{aligned} (\pi(\widehat{v}), x') \models \big(&\text{rightmost\_zero}(\underline{\alpha}^{\text{time}}, k) \\ &\wedge \text{rightmost\_one}(\underline{\alpha}^{\text{pos}}, l_{\text{left}}) \wedge \text{rightmost\_zero}(\underline{\alpha}^{\text{pos}}, l_{\text{right}})\big). \end{aligned}$$

Let us now consider some $m \in \{1 \ldots, d\}$. If $dir_m = \mathit{left}$ then, due to $(\pi(\widehat{v}), x') \models \mathit{compstep}_{\text{left}}(r, \theta)$, there exist an element $y_m \in W_2$ and an element $x_m \in W_1$ such that $\pi(\widehat{v}) R_\Diamond x_m$ as well as

$$(\pi(\widehat{v}), y_m) \models ((\underline{X}^{\text{prevtime}} = \underline{\alpha}^{\text{time}}) \wedge (\underline{X}^{\text{prevpos}} = \underline{\alpha}^{\text{pos}}))$$

and

$$\begin{aligned}(x_m, y_m) \models\ & \Big((\underline{\alpha}^{\text{time}} = \underline{X}^{\text{prevtime}}, > k) \wedge \text{rightmost\_one}(\underline{\alpha}^{\text{time}}, k) \\ & \wedge (\underline{\alpha}^{\text{pos}} = \underline{X}^{\text{prevpos}}, > l_{\text{left}}) \wedge \text{rightmost\_zero}(\underline{\alpha}^{\text{pos}}, l_{\text{left}}) \\ & \wedge (\underline{\alpha}^{\text{prevpos}} = \underline{X}^{\text{prevpos}}) \wedge \alpha_r^{\text{state}} \wedge \alpha_\theta^{\text{written}}\Big).\end{aligned}$$

Similarly, if $dir_m = right$ then, due to $(\pi(\hat{v}), x') \models compstep_{\text{right}}(r, \theta)$, there exist an element $y_m \in W_2$ and an element $x_m \in W_1$ such that $\pi(\hat{v}) R_\Diamond x_m$ as well as

$$(\pi(\hat{v}), y_m) \models ((\underline{X}^{\text{prevtime}} = \underline{\alpha}^{\text{time}}) \wedge (\underline{X}^{\text{prevpos}} = \underline{\alpha}^{\text{pos}}))$$

and

$$\begin{aligned}(x_m, y_m) \models\ & \Big((\underline{\alpha}^{\text{time}} = \underline{X}^{\text{prevtime}}, > k) \wedge \text{rightmost\_one}(\underline{\alpha}^{\text{time}}, k) \\ & \wedge (\underline{\alpha}^{\text{pos}} = \underline{X}^{\text{prevpos}}, > l_{\text{right}}) \wedge \text{rightmost\_one}(\underline{\alpha}^{\text{pos}}, l_{\text{right}}) \\ & \wedge (\underline{\alpha}^{\text{prevpos}} = \underline{X}^{\text{prevpos}}) \wedge \alpha_r^{\text{state}} \wedge \alpha_\theta^{\text{written}}\Big).\end{aligned}$$

We claim that we can define the desired function $\tilde{\pi} : \tilde{V} \to W_1$ by

$$\tilde{\pi}(v) := \begin{cases} \pi(v) & \text{if } v \in V, \\ x_m & \text{if } v = \tilde{v}_m, \text{ for some } m \in \{1, \ldots, d\}. \end{cases}$$

Similarly as in the case $q \in Q_\exists$ one shows that $\tilde{T}$ is a partial tree of M on input w. Note that also Condition IV is satisfied for $\hat{v}$. Finally, similarly as in the case $q \in Q_\exists$ one shows that $\tilde{\pi}$ is a morphism from $\tilde{T}$ to *Model*. This ends the treatment of the case $q \in Q_\forall$. We have proved Lemma 7.2. □

Chapter 8

Reduction of SSL to S4 × S5

We now present an alternative proof of the EXPSPACE-hardness of S4 × S5 by showing the following result. Remember that according to Theorem 6.1 the satisfiability problem of SSL is EXPSPACE-hard.

Theorem 8.1. *The satisfiability problem of the bimodal logic* SSL *can be reduced in logarithmic space to the satisfiability problem of the bimodal logic* S4 × S5.

To prove this theorem we proceed in four steps:

1. We start with the definition of the reduction function.

2. The main work is the proof of its correctness.

3. We prepare the estimate of the space needed to compute this function by investigating the space needed to decide languages of formulas.

4. Finally, we show that the reduction function can be computed using not more than logarithmic space.

8.1 The Reduction Function

We show that the satisfiability problem of SSL can be reduced to the satisfiability problem of S4 × S5. To this end we define a translation $\widehat{T}$ of bimodal formulas in the language $\mathcal{L}$ to bimodal formulas in $\mathcal{L}$ such that for all $\varphi \in \mathcal{L}$

$$\varphi \text{ is SSL-satisfiable} \iff \widehat{T}(\varphi) \text{ is S4} \times \text{S5-satisfiable.}$$

For the reduction we face two main problems.

1. The first problem is that in cross axiom models literals are persistent. To handle this we make sure that the translation $\widehat{T}(\varphi)$ contains a subformula postulating the persistence of literals.

2. The second problem is that in general cross axiom models do not satisfy right commutativity. To handle this we add to each cloud in a cross-axiom model a special "new point" serving as successor point for all points in a predecessor cloud that fail to have in the original model a successor point in this cloud. In order to distinguish the new points from the original ones we use a special propositional variable *main* which is false exactly at the new points.

We define the desired function $\widehat{T} : \mathcal{L} \to \mathcal{L}$ in four steps:

Definition 8.2 (Translation $\widehat{T}$)**.** 1. For every $\varphi \in \mathcal{L}$ let main $\in AT$ by the alphabetically first propositional variable that is not a subformula of φ.

2. Recursively, we define a function $T : \mathcal{L} \to \mathcal{L}$ as follows:

$$\begin{array}{lcl} T(A) & := & A \\ T(\neg\psi) & := & \neg T(\psi) \\ T((\psi_1 \wedge \psi_2)) & := & (T(\psi_1) \wedge T(\psi_2)) \\ T(K\psi) & := & K\neg(\text{main} \wedge \neg T(\psi)) \\ T(\Box\psi) & := & \Box\neg(\text{main} \wedge \neg T(\psi)) \end{array}$$

for all $A \in AT$ and for all $\psi, \psi_1, \psi_2 \in \mathcal{L}$. (Note that $K\neg(\text{main} \wedge \neg T(\psi))$ is equivalent to $K(\text{main} \to T(\psi))$ and that $\Box\neg(\text{main} \wedge \neg T(\psi))$ is equivalent to $\Box(\text{main} \to T(\psi))$.)

3. For $\varphi \in \mathcal{L}$ we define

$$persistent_{main} := \bigwedge_{A \in AT \cap \text{sf}(\varphi)} K(\Box(\text{main} \to A) \vee \Box(\text{main} \to \neg A)).$$

4. For $\varphi \in \mathcal{L}$ we define a function $\widehat{T} : \mathcal{L} \to \mathcal{L}$ by

$$\widehat{T}(\varphi) := \text{main} \wedge K\Box(\neg\text{main} \to \Box\neg\text{main}) \wedge persistent_{main} \wedge T(\varphi).$$

We claim that the function $\widehat{T}$ is indeed a reduction function from the satisfiability problem of SSL to the satisfiability problem of S4 × S5.

Proposition 8.3. *The function $\widehat{T} : \mathcal{L} \to \mathcal{L}$ satisfies, for all $\varphi \in \mathcal{L}$,*

$$\varphi \textit{ is } \mathrm{SSL}\textit{-satisfiable} \iff \widehat{T}(\varphi) \textit{ is } \mathrm{S4} \times \mathrm{S5}\textit{-satisfiable.}$$

The next section is dedicated to the proof of Proposition 8.3. We treat both directions of the claimed equivalence separately.

8.2 Correctness

Lemma 8.4. *Let $\varphi \in \mathcal{L}$. Then*

$$\varphi \textit{ is } \mathrm{SSL}\textit{-satisfiable} \Rightarrow \widehat{T}(\varphi) \textit{ is } \mathrm{S4} \times \mathrm{S5}\textit{-satisfiable.}$$

Proof. Let $\varphi \in \mathcal{L}$ be SSL–satisfiable. Then there are a cross axiom model $M = (W, \xrightarrow{\Diamond}, \xrightarrow{L}, \sigma)$ and a point $w \in W$ such that $M, w \models \varphi$. Let C_i, for $i \in I$, where I is a suitable index set, be the $\xrightarrow{L}$-equivalence classes in W. We construct an S4 × S5-commutator model $M' = (W', \xrightarrow{\Diamond'}, \xrightarrow{L'}, \sigma')$ for $\widehat{T}(\varphi)$ as follows. Let $newpoint_i$ for $i \in I$ be pairwise different "new" points that are not elements of W. We define

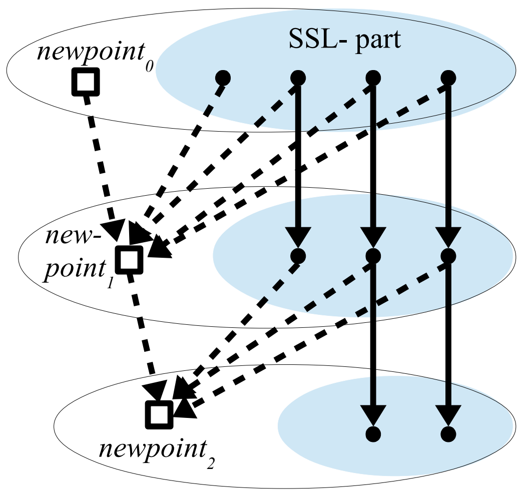
$$W' := W \cup \{newpoint_i \mid i \in I\}.$$

We define the equivalence relation $\xrightarrow{L'}$ on W' by demanding that the following sets C_i' for $i \in I$ are the $\xrightarrow{L'}$-equivalence classes of W':

$$C_i' := C_i \cup \{newpoint_i\},$$

for $i \in I$. Let $\xrightarrow{\Diamond}^{\xrightarrow{L}}$ be the relation on the set of clouds in M induced by $\xrightarrow{\Diamond}$; compare Definition 3.6. We define

$$\xrightarrow{\Diamond'} \;:=\; \xrightarrow{\Diamond} \cup \{(p, newpoint_j) \mid j \in I \text{ and } \exists i \text{ with } p \in C_i' \text{ and } C_i \xrightarrow{\Diamond}^{\xrightarrow{L}} C_j\}.$$



Finally we define

$$\begin{aligned} \sigma'(\text{main}) &:= W, \\ \sigma'(A) &:= \sigma(A), \qquad \text{for } A \in AT \backslash \{\text{main}\}. \end{aligned}$$

We wish to show the following:

1. M' is an S4 × S5-commutator model,
2. $M', w \models \widehat{T}(\varphi)$.

We prove the first claim.
Clearly, $\xrightarrow{L'}$ is an equivalence relation.

The relations $\xrightarrow{\Diamond}$ and $\xrightarrow{\Diamond}^{\xrightarrow{L}}$ are preorders (by assumption respectively by Corollary 3.9), hence, both of them are reflexive and transitive. It is clear that this implies that the relation $\xrightarrow{\Diamond'}$ is reflexive as well. For transitivity of $\xrightarrow{\Diamond'}$, assume that $p \xrightarrow{\Diamond'} q$ and $q \xrightarrow{\Diamond'} r$. If $r \in W$ then the definition of $\xrightarrow{\Diamond'}$ implies that also $q \in W$ and $p \in W$ and $p \xrightarrow{\Diamond} q$ as well as $q \xrightarrow{\Diamond} r$, hence, $p \xrightarrow{\Diamond} r$ by transitivity of $\xrightarrow{\Diamond}$. This implies $p \xrightarrow{\Diamond'} r$. If $r \notin W$ then we let $i, j, k \in I$ be the indices with $p \in C_i'$, $q \in C_j'$, and $r \in C_k'$. Note that in this case $r = newpoint_k$. We observe that $p \xrightarrow{\Diamond'} q$ and $q \xrightarrow{\Diamond'} r$ imply $C_i \xrightarrow{\Diamond}^{\xrightarrow{L}} C_j$ and $C_j \xrightarrow{\Diamond}^{\xrightarrow{L}} C_k$. Transitivity of $\xrightarrow{\Diamond}^{\xrightarrow{L}}$ implies $C_i \xrightarrow{\Diamond}^{\xrightarrow{L}} C_k$, hence, $p \xrightarrow{\Diamond'} newpoint_k = r$. Thus, $\xrightarrow{\Diamond'}$ is transitive as well. We have shown that $\xrightarrow{\Diamond'}$ is a preorder.
Next, we wish to show left commutativity. Let us consider some $p, q, r \in W'$ with $p \xrightarrow{\Diamond'} q$ and $q \xrightarrow{L'} r$. Let i, j be the indices with $p \in C_i'$ and $q, r \in C_j'$. It is

sufficient to show that there exists some $s \in C'_i$ with $s \overset{\Diamond'}{\to} r$. The condition $p \overset{\Diamond'}{\to} q$ implies $C_i \overset{\overset{L}{\to}}{\overset{\Diamond}{\to}} C_j$. If $r \in W$ then the left commutativity of M gives us an $s \in C_i$ with $s \overset{\Diamond}{\to} r$, hence, with $s \overset{\Diamond'}{\to} r$. If $r \notin W$ then $r = \mathit{newpoint}_j$, and $s := p$ does the job.
In order to prove right commutativity let us consider some $p, q, r \in W'$ with $p \overset{L'}{\to} q$ and $q \overset{\Diamond'}{\to} r$. Let i, j be the indices with $p, q \in C'_i$ and $r \in C'_j$. It is sufficient to show that there exists some $s \in C'_j$ with $p \overset{\Diamond'}{\to} s$. The condition $q \overset{\Diamond'}{\to} r$ implies $C_i \overset{\overset{L}{\to}}{\overset{\Diamond}{\to}} C_j$. Hence, we obtain $p \overset{\Diamond'}{\to} \mathit{newpoint}_j$. This proves the first claim, that M' is an S4 $\times$ S5-commutator model.

We come to the second claim, $M', w \models \widehat{T}(\varphi)$.
From the definition of $\sigma'(\text{main})$ we obtain

$$M', w \models \text{main}.$$

Exactly at the points in $\{\mathit{newpoint}_i \mid i \in I\}$ the propositional variable main is false. From the fact that all $\overset{\Diamond'}{\to}$-successors of points in this set are elements of this set as well, we conclude

$$M', w \models K\Box(\neg\text{main} \to \Box\neg\text{main}).$$

Next, we observe that for all $v \in W$ and for all propositional variables $A \in \text{sf}(\varphi)$ we have by definition of σ'

$$M, v \models A \iff M', v \models A.$$

Since all literals are persistent in M and main is true exactly at the points in $W \subseteq W'$ we obtain

$$M', w \models \mathit{persistent}_{main}.$$

Finally, we have to show $M', w \models T(\varphi)$. By induction on the structure of ψ we show the stronger assertion:

$$M, v \models \psi \iff M', v \models T(\psi),$$

for all $v \in W$ and for all $\psi \in \text{sf}(\varphi)$. We distinguish the following cases:

- Case $\psi \in AT$. We already mentioned that for all $v \in W$ and for all $A \in AT \cap \text{sf}(\varphi)$ we have $M, v \models A \iff M', v \models A$.

- Case $\psi = \neg\chi$. Then the following four assertions are equivalent, the second and the third by induction hypothesis:
  1. $M, v \models \psi$,
  2. $M, v \not\models \chi$,
  3. $M', v \not\models T(\chi)$,
  4. $M', v \models T(\psi)$.
- Case $\psi = (\chi_1 \wedge \chi_2)$. This case is treated in the same way.
- Case $\psi = K\chi$. We wish to show
$$M, v \models K\chi \iff M', v \models K(\text{main} \to T(\chi)).$$
First, let us assume $M, v \models K\chi$. Let us consider an arbitrary $v' \in W'$ such that $v \xrightarrow{L'} v'$. It is sufficient to show that
$$M', v' \models (\text{main} \to T(\chi)).$$
If $v' \in W$ then we obtain $v \xrightarrow{L} v'$ and $M, v' \models \chi$. By induction hypothesis we obtain $M', v' \models T(\chi)$, hence $M', v' \models (\text{main} \to T(\chi))$. If $v' \notin W$ then $M', v' \not\models \text{main}$, hence in this case $M', v' \models (\text{main} \to T(\chi))$ as well.
Now, for the other direction, let us assume $M', v \models K(\text{main} \to T(\chi))$. Consider some $u \in W$ with $v \xrightarrow{L} u$ (remember that this implies $v \xrightarrow{L'} u$). It is sufficient to show that
$$M, u \models \chi.$$
But for $u \in W$ we have $M', u \models \text{main}$. The condition $M', v \models K(\text{main} \to T(\chi))$ implies $M', u \models (\text{main} \to T(\chi))$. We obtain $M', u \models T(\chi)$. Finally, by induction hypothesis we obtain $M, u \models \chi$.
- Case $\psi = \Box\chi$. This case can be treated in the same way as the previous case. In fact, it is sufficient to copy the argument for the case $\psi = K\chi$ and to replace K by $\Box$, $\xrightarrow{L}$ by $\xrightarrow{\Diamond}$, and $\xrightarrow{L'}$ by $\xrightarrow{\Diamond'}$.

□

Now we turn to the other direction of Proposition 8.3.

Lemma 8.5. *Let $\varphi \in \mathcal{L}$. Then*
$$\widehat{T}(\varphi) \textit{ is } \mathrm{S4} \times \mathrm{S5}\textit{-satisfiable} \Rightarrow \varphi \textit{ is } \mathrm{SSL}\textit{-satisfiable.}$$

Proof. Let $\widehat{T}(\varphi)$ be S4 × S5–satisfiable. This implies that there exist an S4 × S5–commutator model $M' = (W', \xrightarrow{\Diamond'}, \xrightarrow{L'}, \sigma')$ and a point $w \in W'$ such that $M', w \models \widehat{T}(\varphi)$, that is

$$M', w \models \text{main} \wedge K\Box(\neg\text{main} \rightarrow \Box\neg\text{main}) \wedge \mathit{persistent}_{main} \wedge T(\varphi).$$

We construct a cross axiom model $M := (W, \xrightarrow{\Diamond}, \xrightarrow{L}, \sigma)$ for φ as follows. We construct M as a rooted model, where all points are reachable from the point w:

$$W := \{v \in W' \mid M', v \models \text{main and } (\exists w')\,(w \xrightarrow{L'} w' \text{ and } w' \xrightarrow{\Diamond'} v)\}.$$

We define the relations $\xrightarrow{L}$ and $\xrightarrow{\Diamond}$ on W simply by

$$\begin{aligned} \xrightarrow{L} &:= \xrightarrow{L'} \cap\, (W \times W), \\ \xrightarrow{\Diamond} &:= \xrightarrow{\Diamond'} \cap\, (W \times W). \end{aligned}$$

Finally,

$$\sigma(A) := \begin{cases} \sigma'(A) \cap W & \text{if } A \in \text{sf}(\varphi) \cup \{\text{main}\}, \\ \varnothing & \text{if } A \notin \text{sf}(\varphi) \cup \{\text{main}\}, \end{cases}$$

for all $A \in AT$.
First, we observe that $M', w \models$ main and the reflexivity of $\xrightarrow{L'}$ and $\xrightarrow{\Diamond'}$ imply $w \in W$. We wish to show the following:

1. M is a cross axiom model,
2. $M, w \models \varphi$.

We prove the first claim.
It is obvious that the relation $\xrightarrow{L}$ inherits reflexivity, symmetry and transitivity from $\xrightarrow{L'}$ and that the relation $\xrightarrow{\Diamond}$ inherits reflexivity and transitivity from $\xrightarrow{L'}$. Thus, the relation $\xrightarrow{L}$ is an equivalence relation, and the relation $\xrightarrow{\Diamond}$ is a preorder.
Next we wish to show that M has the left commutativity property. So let us consider points $p, q, r \in W$ with $p \xrightarrow{\Diamond} q$ and $q \xrightarrow{L} r$. By definition of the relations $\xrightarrow{\Diamond}$ and $\xrightarrow{L}$ we obtain that also $p \xrightarrow{\Diamond'} q$ and $q \xrightarrow{L'} r$. Since M' has the left commutativity property there is some point $p' \in W'$ with $p \xrightarrow{L'} p'$ and $p' \xrightarrow{\Diamond'} r$. The definition of W and $p \in W$ imply that there exists some $w' \in W'$ with $w \xrightarrow{L'} w'$ and $w' \xrightarrow{\Diamond'} p$. The left commutativity of M' and $w' \xrightarrow{\Diamond'} p$

as well as $p \xrightarrow{L'} p'$ imply that there exists some $w'' \in W'$ with $w' \xrightarrow{L'} w''$ and $w'' \xrightarrow{\Diamond'} p'$. So, we have $w \xrightarrow{L'} w''$ and $w'' \xrightarrow{\Diamond'} p'$. In addition to that, $M', w \models K\Box(\neg\text{main} \rightarrow \Box\neg\text{main})$ implies $M', w'' \models \Box(\neg\text{main} \rightarrow \Box\neg\text{main})$, hence, $M', p' \models \neg\text{main} \rightarrow \Box\neg\text{main}$. This, together with $p' \xrightarrow{\Diamond'} r$ and $M, r \models \text{main}$, implies $M', p' \models \text{main}$. Hence $p' \in W$ and $p \xrightarrow{L} p'$ as well as $p' \xrightarrow{\Diamond} r$. This shows that M has the left commutativity property.
Finally, we claim that propositional variables are persistent in M. For all $A \in AT\backslash(\{\text{main}\} \cup \text{sf}(\varphi))$ we have $\sigma(A) = \emptyset$. Hence, all $A \in AT\backslash(\{\text{main}\} \cup \text{sf}(\varphi))$ are persistent in M. Furthermore, we have $\sigma(\text{main}) = W$. Hence, main is persistent in M as well. Let us consider $u, v \in W$ with $u \xrightarrow{\Diamond} v$. Due to the definition of W there exists some $w' \in W'$ with $w \xrightarrow{L'} w'$ and $w' \xrightarrow{\Diamond'} u$. Then, $M', w \models \mathit{persistent}_{main}$ implies, for all $A \in AT \cap \text{sf}(\varphi)$, $M', w' \models \Box(\text{main} \rightarrow A) \vee \Box(\text{main} \rightarrow \neg A)$. Note that $M', u \models \text{main}$ and $M', v \models \text{main}$. So, $M', u \models A$ if, and only if, $M', v \models A$. Due to $\sigma(A) = \sigma'(A) \cap W$ the same holds true with M' replaced by M. This shows that all $A \in AT \cap \text{sf}(\varphi)$ are persistent in M. We have shown that M is a cross axiom model.
We come to the second claim, $M, w \models \varphi$. Due to $M', w \models T(\varphi)$ it is sufficient to prove for all $v \in W$ and for all $\psi \in \text{sf}(\varphi)$:

$$M', v \models T(\psi) \quad \Longleftrightarrow \quad M, v \models \psi.$$

We show this by induction on the structure of ψ. We distinguish the following cases:

- Case $\psi = A \in AT$. In this case the claim is true due to the definition of σ.
- Case $\psi = \neg\chi$. Then the following four assertions are equivalent, the second and the third by induction hypothesis:
  1. $M, v \models \psi$,
  2. $M, v \not\models \chi$,
  3. $M', v \not\models T(\chi)$,
  4. $M', v \models T(\psi)$.
- Case $\psi = (\chi_1 \wedge \chi_2)$. This case is treated in the same way.
- Case $\psi = K\chi$.
  Let us first assume $M', v \models T(K\chi)$, that is $M', v \models K(\text{main} \rightarrow T(\chi))$. We wish to show $M, v \models K\chi$. Consider an arbitrary $v' \in W$ with $v \xrightarrow{L} v'$. It is sufficient to show $M, v' \models \chi$. Note that the definition of $\xrightarrow{L}$

implies $v \xrightarrow{L'} v'$. Hence, we have $M', v' \models (\text{main} \to T(\chi))$. Furthermore, due to $v' \in W$ we have $M', v' \models \text{main}$. We obtain $M', v' \models T(\chi)$. By induction hypothesis we obtain $M, v' \models \chi$.

For the other direction let us assume that $M, v \models K\chi$. We wish to show $M', v \models K(\text{main} \to T(\chi))$. Consider an arbitrary $v' \in W'$ with $v \xrightarrow{L'} v'$. It is sufficient to show $M', v' \models (\text{main} \to T(\chi))$. If $v' \in W$ then $v \xrightarrow{L'} v'$ implies $v \xrightarrow{L} v'$, and $M, v \models K\chi$ implies $M, v' \models \chi$. By induction hypothesis we obtain $M', v' \models T(\chi)$, hence, $M', v' \models (\text{main} \to T(\chi))$. If $v' \notin W$ then we claim that $M', v' \not\models \text{main}$, hence, $M', v' \models (\text{main} \to T(\chi))$. Indeed, the assumption $v \in W$ implies that there exists some $w' \in W'$ with $w \xrightarrow{L'} w'$ and $w' \xrightarrow{\Diamond'} v$. Together with $v \xrightarrow{L'} v'$ and left commutativity of M' and transitivity of $\xrightarrow{L'}$ we can conclude that there exists some $w'' \in W'$ with $w \xrightarrow{L'} w''$ and $w'' \xrightarrow{\Diamond'} v'$. Hence, v' is reachable from w. By definition of W the assumption $v' \notin W$ indeed implies $M', v' \not\models \text{main}$.

- Case $\psi = \Box\chi$. This case can be treated similarly as the previous case. In fact, for the first part (the proof that $M', v \models T(\Box\chi)$ implies $M, v \models \Box\chi$) one can copy the argument for the corresponding part in the case $\psi = K\chi$ and replace K by $\Box$, $\xrightarrow{L}$ by $\xrightarrow{\Diamond}$, and $\xrightarrow{L'}$ by $\xrightarrow{\Diamond'}$.

  For the other direction let us assume that $M, v \models \Box\chi$. We wish to show $M', v \models \Box(\text{main} \to T(\chi))$. Consider an arbitrary $v' \in W'$ with $v \xrightarrow{\Diamond'} v'$. It is sufficient to show $M', v' \models (\text{main} \to T(\chi))$. If $v' \in W$ then $v \xrightarrow{\Diamond'} v'$ implies $v \xrightarrow{\Diamond} v'$, and $M, v \models \Box\chi$ implies $M, v' \models \chi$. By induction hypothesis we obtain $M', v' \models T(\chi)$, hence, $M', v' \models (\text{main} \to T(\chi))$. If $v' \notin W$ then we claim that $M', v' \not\models \text{main}$, hence, $M', v' \models (\text{main} \to T(\chi))$. Indeed, the assumption $v \in W$ implies that there exists some $w' \in W'$ with $w \xrightarrow{L'} w'$ and $w' \xrightarrow{\Diamond'} v$. Together with $v \xrightarrow{\Diamond'} v'$ we conclude $w' \xrightarrow{\Diamond'} v'$. Hence, $v' \notin W$ indeed implies $M', v' \not\models \text{main}$. □

8.3 LOGSPACE Decidability of Languages of Formulas

We wish to show that the language $\mathcal{L}$ of bimodal formulas can be decided in logarithmic space. We are going to show the slightly stronger result that $\mathcal{L}$ is even in ALOGTIME, where ALOGTIME is the set of all languages that can be decided in logarithmic time by an alternating Turing machine with “random access” to the input; compare Buss [17, Page 124], Ibarra, Jiang, and Rivakumar [52], Clote [20, Def. 2.3]. What “alternating” means has already been explained in Section 5.3. An alternating Turing machines with *random access* to the input string may be described as follows. Such a machine has a special work tape called *index query tape*, a special state called *input query state* q_{query} and for every symbol $a \in \Sigma \cup \{\sqcup\}$ (where Σ is the input alphabet and $\sqcup$ is a special symbol not contained in the input alphabet) an *answer state* $q_{\text{answer},a}$. The index query tape should contain at any time a binary string $\text{bin}(k)$, for some natural number k. Let us assume that the input string is a string $a_0, \ldots, a_{n-1}$ with $a_i \in \Sigma$, for all $i \in \{0, \ldots, n-1\}$ and for some $n \in \mathbb{N}$. When the machine is in the input query state then in the next step it enters the state

- q_{answer,a_k} if $k \in \{0, \ldots, n-1\}$,
- $q_{\text{answer},\sqcup}$ if $k \geqslant n$.

The set ALOGTIME is the set of all languages that are accepted in the sense of Section 5.3 by alternating Turing machines that work in time logarithmic in the input length and that have random access to the input.
We are going to show the following fact.

Proposition 8.6. *The language $\mathcal{L}$ of bimodal formulas is an element of* ALOGTIME.

Corollary 8.7. *The language $\mathcal{L}$ of bimodal formulas can be decided in logarithmic space.*

Proof. It is well-known that ALOGTIME is a subset of LOGSPACE; see [20, P. 601]. □

We still need to prove Prop. 8.6. Concerning this, we remark that Buss [17, Pages 124, 125] has shown that a certain language of Boolean formulas is in ALOGTIME. But our syntax of formulas is slightly different from the one used by Buss. Therefore, we cannot directly use his result. But we are going to proceed in a similar manner.
Remember that we represent formulas by strings over a finite alphabet. In order to do this we represent propositional variables by strings of the form $x\text{bin}(i)$ where x is a fixed symbol and $\text{bin}(i)$ is defined by

$$\text{bin}(i) := \begin{cases} 0 & \text{if } i = 0, \\ \text{the binary representation of } i \text{ without leading zeros} & \text{if } i > 0, \end{cases}$$

for $i \in \mathbb{N}$. Let ε denote the empty string over any alphabet.

A context free grammar is a quadruple $G = (V, \Sigma, \to, S)$ where V, the set of nonterminal symbols, is a nonempty set, where Σ, the set of terminal symbols, is a nonempty set with $V \cap \Sigma = \varnothing$, where $\to$ is a subset of $V \times (V \cup \Sigma)^*$, and where $S \in V$ is the start symbol. The elements of $\to$ are called *productions* or *rules*. Usually an element (D, u) of $\to$ is written in the form $D \to u$. If we wish to list several elements of $\to$, say two elements $D \to u_1$ and $D \to u_2$, we may write this as $D \to u_1 \mid u_2$. Let

$$\Gamma := V \cup \Sigma.$$

As usual we extend the relation $\to$ to a relation $\to$ between strings in Γ^* by

$$\begin{array}{rcl} w \to w' & :\Longleftrightarrow & \text{there are } u, v, v' \in \Gamma^* \text{ and } D \in V \text{ with} \\ & & (w = uDv \text{ and } w' = uv'v \text{ and } D \to v'), \end{array}$$

for $w, w' \in \Gamma^*$. Furthermore, let $\to^*$ be the reflexive-transitive closure of the relation $\to$ on Γ^*, and define

$$\begin{array}{rcl} S(G) & := & \{w \in \Gamma^* \mid S \to^* w\}, \\ L(G) & := & S(G) \cap \Sigma^*. \end{array}$$

The language $\mathcal{L}$ of bimodal formulas was defined in Definition 3.4. Actually, first we will show that a similarly defined set of Boolean formulas is in ALOGTIME. Let $G = (V, \Sigma, \to, S)$ be the following context free grammar:

- $V := \{S, B\}$,
- $\Sigma := \{(,), \neg, \wedge, x, 0, 1\}$,
- $\to$ is the following set of context free production rules:

$$\begin{array}{rcl} S & \to & \neg S \mid (S \wedge S) \mid x0 \mid x1 \mid x1B \\ B & \to & 0 \mid 1 \mid 0B \mid 1B \end{array}$$

Then $L(G)$ is the set of Boolean formulas. Let $\Gamma := \Sigma \cup V$.

Proposition 8.8. *The language $L(G)$, the language of Boolean formulas, is in* ALOGTIME.

Before we prove this, let us deduce Proposition 8.6 from it.

Proof of Proposition 8.6. Let h be the function mapping strings over $\Sigma \cup \{\Box, K\}$ to strings over Σ that replaces every occurrence of $\Box$ or K by $\neg$. It is clear that $\mathcal{L} = h^{-1}(L(G))$. It is also clear that h can be computed by a Turing machine (one does not even need alternation) with random access to the input in logarithmic time. Thus, the assertion follows directly from Proposition 8.8 and [17, Theorem 4], which says that if B is a language in ALOGTIME and h is an ALOGTIME-reduction of a language A to the language B then A is in ALOGTIME as well. □

We still need to show Proposition 8.8. We are going to make use of the fact that all productions with one exception in the grammar G are right-linear. The exception is, of course, the production $S \to (S \wedge S)$. This leads to the following description of the strings in $S(G)$. Let us call a string in Σ^* *balanced* if it contains exactly as many opening brackets as closing brackets as occurrences of the symbol $\wedge$.

Lemma 8.9. *Let $w = a_0 \ldots, a_{n-1} \in \Gamma^*$ be a string with $a_i \in \Gamma$, for $i \in \{0, \ldots, n-1\}$. Then $w \in S(G)$ if, and only if, all of the following seven conditions are satisfied:*

1. *w is not the empty string,*
2. *w is balanced,*
3. *$a_0 \in \{S, \neg, (, x\}$,*
4. *$a_{n-1} \in \{S,), B, 0, 1\}$,*
5. *for all $i \in \{0, \ldots, n-2\}$, $a_i a_{i+1} \notin P$ where the set P of forbidden pairs of symbols is defined by*

$$\begin{array}{rcl} P & := & \{(, \neg, \wedge\}\ \{B, \wedge,), 0, 1\} \\ & \cup & \{S, B,)\}\ \{S, B, (, \neg, x, 0, 1\} \\ & \cup & \{x\}\ \{S, B, (,), \neg, \wedge, x\} \\ & \cup & \{0, 1\}\ \{S, \neg, (, x\}, \end{array}$$

6. *for all $i \in \{0, \ldots, n-3\}$, $a_i a_{i+1} a_{i+2} \notin \{x00, x01, x0B\}$,*
7. *for all $i \in \{0, \ldots, n-1\}$, if $a_i = ($ then there exist $j, k \in \mathbb{N}$ with the following properties:*
   (a) *$i + 1 < j < k - 1 < n - 1$,*
   (b) *$a_j = \wedge$ and $a_k =)$,*
   (c) *the string $a_{i+1} \ldots a_{j-1}$ is balanced,*

(d) the string $a_{j+1} \dots a_{k-1}$ is balanced.

Proof. Both directions of the claimed equivalence can be shown by lengthy but straightforward inductions, the direction "$\Rightarrow$" by induction over the length of a shortest derivation of a string w in $S(G)$, and the direction "$\Leftarrow$" by induction over the length of a string w satisfying the seven conditions, where a modified length function is used. For completeness sake we give the full proofs.
"$\Rightarrow$": That any $w \in S(G)$ satisfies all of the listed conditions follows by a straightforward induction over the length of a shortest derivation of w in the grammar G. For completeness sake we give the full proof.
If the length of the derivation is zero then $w = S$, and S obviously satisfies the first four conditions, and the remaining three conditions do not apply. Let us now consider a shortest derivation of w in the grammar G and let us assume that the length of the derivation is positive. Let us assume that the last rule applied in the derivation led from a string $w' \in S(G)$ to w. By induction hypothesis w' satisfies all seven conditions.
First, w' is not the empty string. As none of the rules in G decreases the length of a string, w is not the empty string either.
Secondly, w' is balanced. There is only one rule that adds the symbol $\wedge$ or any brackets at all, the rule $S \to (S \wedge S)$, and this rule adds exactly one opening bracket, one symbol $\wedge$, and one closing bracket. Hence, w is balanced as well.
Thirdly, the first symbol of w' is one of the symbols $S, \neg, (, x$. In particular, it is not the symbol B. Hence, the first symbol of w' is either not changed, or it is changed from S to $\neg$ or (or x. Thus, the first symbol of w is one of the symbols $S, \neg, (, x$ as well.
The last symbol of w' is one of the symbols $S,), B, 0, 1$. As the last symbol in the right hand side of any of the rules in G is one of these symbols as well, so is the last symbol of w.
We come to the fifth condition. Let us first assume that $w' = uSv$, for some $u, v \in \Gamma^*$, and that $w = uyv$ for some string $y \in \{\neg S, (S \wedge S), x0, x1, x1B\}$. Let $a_i a_{i+1}$ be a substring of w of length two. If $a_i a_{i+1}$ is a substring of u or of v then $a_i a_{i+1} \notin P$ by induction hypothesis. It is also clear that none of the substrings of y of length two is an element of P. If a_i is the last symbol in u then in w' it is followed by S. As w' satisfies the fifth condition by induction hypothesis we conclude $a_i \in \{(, \neg, \wedge\}$. As the first symbol of y is either $\neg$ or (or x, we conclude that $a_i a_{i+1} \in \{(, \neg, \wedge\}\ \{\neg, (, x\}$, hence, $a_i a_{i+1} \notin P$. Finally, if a_{i+1} is the first symbol of v then in w' it is preceded by S. As w' satisfies the fifth condition by induction hypothesis we conclude $a_{i+1} \in \{\wedge,)\}$. As the last symbol of y is either S or B or) or 0 or 1, we

conclude that $a_i a_{i+1} \in \{S, B,), 0, 1\} \{\wedge,)\}$, hence, $a_i a_{i+1} \notin P$.
Now let us assume that $w' = uBv$, for some $u, v \in \Gamma^*$, and that $w = uyv$ for some string $y \in \{0, 1, 0B, 1B\}$. Let $a_i a_{i+1}$ be a substring of length two of w. If $a_i a_{i+1}$ is a substring of u or of v then $a_i a_{i+1} \notin P$ by induction hypothesis. It is also clear that none of the substrings of y of length two is an element of P. If a_i is the last symbol in u then in w' it is followed by B. As w' satisfies the fifth condition by induction hypothesis we conclude $a_i \in \{0, 1\}$. As the first symbol of y is either 0 or 1, we conclude that $a_i a_{i+1} \in \{0, 1\} \{0, 1\}$, hence, $a_i a_{i+1} \notin P$. Finally, if a_{i+1} is the first symbol of v then in w' it is preceded by B. As w' satisfies the fifth condition by induction hypothesis we conclude $a_{i+1} \in \{\wedge,)\}$. As the last symbol of y is either 0 or 1 or B, we conclude that $a_i a_{i+1} \in \{0, 1, B\} \{\wedge,)\}$, hence, $a_i a_{i+1} \notin P$. Thus, w satisfies the fifth condition.
We come to the sixth condition. Let us assume that $w' = uUv$, for some $u, v \in \Gamma^*$ and $U \in \{S, B\}$ and that $w = uyv$ for some string $y \in \{\neg S, (S \wedge S), x0, x1, x1B, 0, 1, 0B, 1B\}$. Let z be an element of the set

$$T := \{x00, x01, x0B\}$$

of forbidden triplets of symbols. By induction hypothesis, z is not a substring of w', and we wish to show that z is not a substring of w. Indeed, by induction hypothesis z is not a substring of u or of v. It is clear that z is not a substring of y. The symbol x cannot be the last symbol of u because then in w' is would be followed by U, hence, by S or by B, which is impossible by induction hypothesis. Furthermore $x0$ cannot be the last pair of symbols in u because then w' would contain $0S$ or $x0B$ as a substring, which is not true by induction hypothesis. Neither 0 nor 1 nor B can be the first symbol in v because then in w' it would be preceded by S or by B, which is not the case by induction hypothesis. Thus, z is not a substring of w.
Finally, it is clear that w satisfies the seventh condition as well because all right hand sides of productions in G are balanced.
"$\Leftarrow$": We shall use the following modified length function $\ell : \Gamma^* \to \mathbb{N}$ defined by

$$\ell(w) := \#_V(w) + 2 \cdot \#_\Sigma(w) = |w| + \#_\Sigma(w),$$

for $w \in \Gamma^*$. The definition of ℓ guarantees that replacing a nonterminal symbol by a terminal symbol strictly increases the value of ℓ. Hence, all productions in G are strictly length increasing with respect to ℓ, that is, if $w' \to w$ then $\ell(w') < \ell(w)$.
We assume that some string $w \in \Gamma^*$ satisfies all seven conditions, and we wish to show that w is an element of $S(G)$. We are going to show this by induction over $\ell(w)$. By the first condition w is not the empty string. We distinguish a series of cases.

1. w contains a pair $b_1b_2 \in \{00, 01, 10, 11\}$ that is not followed by 0 or 1 or B.

   Then w has the form ub_1b_2v with $u, v \in \Gamma^*$ such that v is either the empty string or its first symbol is different from $0, 1, B$. Then the string $w' := ub_1Bv$ satisfies on the one hand $w' \to w$ and on the other hand $\ell(w') < \ell(w)$. Furthermore, the assumption that w satisfies all of the seven conditions implies that w' satisfies all of the seven conditions as well. Indeed, w' is not the empty string, and w' is balanced. The first symbol of w' is identical with the first symbol of w. The last symbol of w' is identical with the last symbol of w or equal to B. The string w' does not contain any of the pairs in P as a substring because w does not contain any of the pairs in P as a substring and because either v is empty or its first symbol is different from 0 and from 1 and from B. The string w' does not contain any of the triplets in T as a substring because w does not contain any of the triplets in T. Finally, it is clear that w' satisfies the seventh condition because w satisfies the seventh condition. By induction hypothesis w' is an element of $S(G)$. But then $w' \to w$ implies that w is an element of $S(G)$ as well.

2. w does not satisfy the condition in the previous case, but it contains a triplet b_1b_2B with $b_1b_2 \in \{00, 01, 10, 11\}$.

   Then w has the form ub_1b_2Bv with $u, v \in \Gamma^*$. Then the string $w' := ub_1Bv$ satisfies on the one hand $w' \to w$ and on the other hand $\ell(w') < \ell(w)$. Furthermore, the assumption that w satisfies all of the seven conditions implies that w' satisfies all of the seven conditions as well. By induction hypothesis w' is an element of $S(G)$. But then $w' \to w$ implies that w is an element of $S(G)$ as well.

From now on we can assume that w does not satisfy the conditions of the previous two cases. Note that this implies that w does not contain any pair $b_1b_2 \in \{00, 01, 10, 11\}$ of bits.

3. w does not satisfy the conditions of the previous two cases, and w contains a symbol B.

   Let us fix an occurrence of B in w. Then B cannot be the first symbol of w. It must come after some 0 or some 1. The bit before B must be preceded by some 0 or some 1 or some x. As w does not contain any consecutive pair of bits, the bit before B must be preceded by some x. Finally, the triplet $x0B$ is not allowed. So, w has the form $ux1Bv$ with $u, v \in \Gamma^*$. Then the string $w' := uSv$ satisfies on the one hand $w' \to w$ and on the other hand $\ell(w') < \ell(w)$. Furthermore, the assumption that

w satisfies all of the seven conditions implies that w' satisfies all of the seven conditions as well. By induction hypothesis w' is an element of $S(G)$. But then $w' \to w$ implies that w is an element of $S(G)$ as well.

4. w does not satisfy the conditions of the previous three cases, and w contains a bit $c \in \{0, 1\}$.

   This bit c must be preceded by the symbol x. Hence, w has the form $w = uxcv$ with $u, v \in \Gamma^*$. Then the string $w' := uSv$ satisfies on the one hand $w' \to w$ and on the other hand $\ell(w') < \ell(w)$. Furthermore, w' satisfies all of the seven conditions as well. This follows from the assumption that w satisfies all of the seven conditions and from the assumption that w does not satisfy the conditions of the previous three cases, hence, either v is empty or its first symbol is different from $0, 1, B$. By induction hypothesis w' is an element of $S(G)$. But then $w' \to w$ implies that w is an element of $S(G)$ as well.

From now on we can assume that w does not satisfy the conditions of the previous four cases. Then w does not contain any of the symbols $0, 1, B$. Hence, it cannot contain x either as x cannot be the last symbol of w and any occurrence of x can only be followed by 0 or 1.

5. w does not contain any of the symbols $0, 1, B, x$, but it contains a pair $\neg S$. Then w has the form $w = u\neg Sv$ with $u, v \in \Gamma^*$. In this case the string $w' := uSv$ satisfies on the one hand $w' \to w$ and on the other hand $\ell(w') < \ell(w)$. Furthermore, w' satisfies all of the seven conditions as well. This follows from the assumption that w satisfies all of the seven conditions and from the assumption that w does not contain any of the symbols $0, 1, B, x$. By induction hypothesis w' is an element of $S(G)$. But then $w' \to w$ implies that w is an element of $S(G)$ as well.

6. w does not contain any of the symbols $0, 1, B, x$, but it contains a substring of the form $(S \wedge S)$. Then w has the form $w = u(S \wedge S)v$ with $u, v \in \Gamma^*$. In this case the string $w' := uSv$ satisfies on the one hand $w' \to w$ and on the other hand $\ell(w') < \ell(w)$. Furthermore, w' satisfies all of the seven conditions as well. This follows from the assumption that w satisfies all of the seven conditions and from the assumption that w does not contain any of the symbols $0, 1, B, x$. By induction hypothesis w' is an element of $S(G)$. But then $w' \to w$ implies that w is an element of $S(G)$ as well.

7. w does not satisfy any of the previous six cases. Then, in particular, w does not contain any of the symbols $0, 1, B, x$. We claim that $w = S$.

First, let us show that w does not contain any opening bracket (. For the sake of a contradiction let us assume that w contains an opening bracket. Let us consider the rightmost occurrence of an opening bracket in w. Then, by the seventh condition, w has the form $w = u(v \wedge v')z$, for some $u, v, v', z \in \Gamma^*$ where v and v' are balanced and nonempty. As the bracket before u is assumed to be the rightmost bracket in w, the strings v and v' do not contain any opening bracket. As they are balanced they contain neither any bracket at all nor the symbol $\wedge$. The rightmost symbol of v and of v' must be an S. If v or v' would contain some further symbol, then the symbol before this S would have to be (or $\neg$ or $\wedge$. But neither v nor v' contain any of the two symbols (and $\wedge$, and we assume that w does not satisfy the fifth case, hence, w does not contain a pair $\neg S$. We conclude that $v = v' = S$. But then w contains the string $(S \wedge S)$ as a substring in contradiction to our assumption that w does not satisfy the sixth condition. We conclude that w does not contain any opening bracket. As w is balanced it contains neither any bracket at all nor the symbol $\wedge$. But then it cannot contain the symbol $\neg$ either. Indeed, $\neg$ cannot be the last symbol of w. So, the last occurrence of $\neg$ in w could only be followed by some S, but, again, that would contradict our assumption that w does not satisfy the fifth case. Thus, the only symbol that may appear in w is the symbol S. We conclude $w = S$. Finally, indeed, S is an element of $S(G)$. □

Corollary 8.10. *Let $w = a_0 \ldots, a_{n-1} \in \Sigma^*$ be a string with $a_i \in \Sigma$, for $i \in \{0, \ldots, n-1\}$. Then $w \in L(G)$ if, and only if, all of the following seven conditions are satisfied:*

1. *w is not the empty string,*
2. *w is balanced,*
3. *$a_0 \in \{\neg, (, x\}$,*
4. *$a_{n-1} \in \{), 0, 1\}$,*
5. *for all $i \in \{0, \ldots, n-2\}$, $a_i a_{i+1} \notin Q$ where the set Q of forbidden pairs of symbols is defined by*

$$\begin{aligned} P \quad := \quad & \{(, \neg, \wedge\}\ \{\wedge,), 0, 1\} \\ \cup\ & \{)\}\ \{S, B, (, \neg, x, 0, 1\} \\ \cup\ & \{x\}\ \{(,), \neg, \wedge, x\} \\ \cup\ & \{0, 1\}\ \{\neg, (, x\}, \end{aligned}$$

6. *for all* $i \in \{0, \ldots, n-3\}$, $a_i a_{i+1} a_{i+2} \notin \{x00, x01, x0B\}$,
7. *for all* $i \in \{0, \ldots, n-1\}$, *if* $a_i = ($ *then there exist* $j, k \in \mathbb{N}$ *with the following properties:*
   (a) $i + 1 < j < k - 1 < n - 1$,
   (b) $a_j = \wedge$ *and* $a_k =)$,
   (c) *the string* $a_{i+1} \ldots a_{j-1}$ *is balanced,*
   (d) *the string* $a_{j+1} \ldots a_{k-1}$ *is balanced.*

Proof. This follows from $L(G) = S(G) \cap \Sigma^*$. □

Proof of Proposition 8.8. It is fundamental and well-known that one can count the number of occurrences of a specific symbol in a given string by an alternating Turing machine with random access to the input string in logarithmic time; see [52] or [17] for a precise statement. In particular, for any two different symbols a and b, the problem to check whether an input string w contains the same number of a's as b's is in ALOGTIME (see also [52, Page 113]). We claim that, using this, one can check all of the seven conditions formulated in the previous corollary by an alternating Turing machine with random access to the input string in logarithmic time. By using universal states, one ensures that for the acceptance of the input string w all seven conditions must be satisfied. Let us go through the conditions one by one. It is clear that one can check even in constant time whether the input string w is empty or not. By counting the number of occurrences of opening brackets, of closing brackets and of the symbol $\wedge$, one can check wether the input string is balanced or not. The condition concerning the leftmost symbol in the input string can also be checked in constant time. For the condition concerning the rightmost symbol one should first compute the length n of the input string (using again a counting subroutine). Then, using the computed binary representation of the number $n-1$ one can check whether the symbol a_{n-1} of the input string $w = a_0 \ldots a_{n-1}$ is an element of the set $\{), 0, 1\}$ or not. For the fifth condition one first guesses (using universal states) a binary string of length approximately $\log(n)$. If it is equal to $\mathrm{bin}(i)$ for some i with $0 \leqslant i < n-2$ then one checks the fifth condition for the pair $a_i a_{i+1}$. Similarly the sixth condition is treated. Finally, for the seventh condition, one first guesses (using universal states) a binary string of length approximately $\log(n)$. If it is equal to $\mathrm{bin}(i)$ for some i with $0 \leqslant i < n-1$ then one guesses (using existential states) two binary strings of length approximately $\log(n)$. If they are equal to $\mathrm{bin}(j)$ and $\mathrm{bin}(k)$, respectively, satisfying $i + 1 < j < k - 1 < n - 1$ then one checks whether $a_j = \wedge$ and $a_k =)$

and (using a counting subroutine again) whether the strings $a_{i+1} \dots a_{j-1}$ and $a_{j+1} \dots a_{k-1}$ are balanced. Note such an alternating Turing machine accepts exactly the strings in $L(G)$, and it works in logarithmic time. $\square$

8.4 LOGSPACE Computability of the Reduction Function

In this section we show that the satisfiability problem of the logic SSL can be reduced in logarithmic space to the satisfiability problem of S4 × S5.
Let $\Sigma := \{(,), \neg, \square, K, \wedge, x, 0, 1\}$ be the alphabet over which bimodal formulas are defined. In order to prove the assertion we have to show that there is a logspace computable function $\widetilde{T} : \Sigma^* \to \Sigma^*$ such that, for all $\varphi \in \Sigma^*$,

$$\begin{aligned} &(\varphi \in \mathcal{L} \text{ and } \varphi \text{ is SSL-satisfiable}) \\ &\quad\iff \quad (\widetilde{T}(\varphi) \in \mathcal{L} \text{ and } \widetilde{T}(\varphi) \text{ is S4} \times \text{S5-satisfiable}). \end{aligned}$$

We define such a function $\widetilde{T}$ by formulating an algorithm for computing it that works in logarithmic space.
So, let $\varphi \in \Sigma^*$ be the input string. According to Corollary 8.7 we can first check in logarithmic space whether φ is a bimodal formula or not, that is, whether φ is an element of $\mathcal{L}$ or not. If not then the algorithm outputs $\widetilde{T}(\varphi) := \wedge$ (which is certainly not a bimodal formula). If, on the other hand, φ is a bimodal formula then we wish to compute and print $\widehat{T}(\varphi)$ as defined in Section 8.1.
First, we compute the smallest natural number i such that $x\text{bin}(i)$ is not a subformula of φ. We do this by starting with $j = 0$, increasing j step by step, and checking in each step whether $x\text{bin}(j)$ is a subformula of φ. Note that a string $x\text{bin}(j)$ for some $j \in \mathbb{N}$ is a subformula of φ if, and only if, there exists an occurrence of the string $x\text{bin}(j)$ as a substring of φ that is not followed by a 0 or a 1. This algorithm works in logarithmic space because φ can contain only less than $|\varphi|$ many subformulas of the form $x\text{bin}(j)$. Thus, we need to check whether $x\text{bin}(j)$ is a subformula of φ only for $j < |\varphi|$. For all these j the binary representation $\text{bin}(j)$ can be stored in logarithmic space. Finally, we can also store the string main $:= x\text{bin}(i)$ in logarithmic space.
Now we wish to compute and output $\widehat{T}(\varphi)$. It is straightforward to print $\text{main} \wedge K\square(\neg\text{main} \to \square\neg\text{main})$. Next we wish to print $\mathit{persistent}_{main}$. In order to do this we must identify all $j \in \mathbb{N}$ such that $x\text{bin}(j)$ is a subformula of φ, and for each such j we must print

$$K(\square(\text{main} \to x\text{bin}(j)) \vee \square(\text{main} \to \neg x\text{bin}(j))).$$

This can be done as follows. We read the string φ from left to right. Whenever we read an x we use two binary counters in order to mark the beginning and the end of the subformula $x\text{bin}(j)$ that begins with this occurrence of x. Then we check, again using binary counters, whether the same subformula $x\text{bin}(j)$ has appeared already further to the left in φ. If it has then we just move on. If it has not appeared before, then we print out $K(\Box(\text{main} \rightarrow x\text{bin}(j)) \vee \Box(\text{main} \rightarrow \neg x\text{bin}(j)))$, and then we move on. It is clear that all this can be done in logarithmic space.
Finally, we wish to output $T(\varphi)$. In order to compute $T(\varphi)$ one has to replace every occurrence of "K" by "$K\neg(\text{main} \wedge \neg$", and every occurrence of "$\Box$" by "$\Box\neg(\text{main} \wedge \neg$", and one has to add an additional closing bracket after each subformula $K\psi$ and each subformula $\Box\psi$ of φ. Besides that, all other symbols from φ can simply be copied. The only nontrivial part here is the addition of a closing bracket after each occurrence of a subformula of the form $K\psi$ or $\Box\psi$ of φ. In order to do this, for each position in the string φ one has to count how many subformulas of the form $K\psi$ or $\Box\psi$ of φ end in this position and then one has to print so many additional closing brackets. So, how can one count how many subformulas of the form $K\psi$ or $\Box\psi$ of φ end in the current position in the string φ? If the symbol in this position is an element of $\{(, \neg, \Box, K, \wedge, x\}$ then no subformula ends in this position. The same is true if the symbol in this position is either 0 or 1 and this is followed by a bit 0 or 1 as well. There are only the following possible cases for the last symbol of a subformula.

- If the symbol in the current position is a bit, so 0 or 1, and this is not followed by a bit, then a subformula of the form $x\text{bin}(j)$ for some j ends in this position. Then we move to the left of the corresponding occurrence of x and count the number of occurrences of K and $\Box$ until we read a symbol not in $\{K, \Box, \neg\}$.

- If the symbol in the current position is a closing bracket) then we go to the left step by step until we have found the corresponding opening bracket (. This can be done by using a binary counter that is increased by 1 for each closing bracket and decreased by 1 for each opening bracket. One stops when this counter is back to its initial value 0. Once we have found the corresponding opening bracket we move to the left of it and count the number of occurrences of K and $\Box$ until we read a symbol not in $\{K, \Box, \neg\}$.

By using binary counters all this can be done in logarithmic space. This ends the description of the computation in logarithmic space of the described reduction function $\widetilde{T}$.

Chapter 9

Reduction of S4 × S5 to K4 × S5

The EXPSPACE-hardness of K4 × S5 follows from the EXPSPACE-hardness of S4 × S5 and from the following result.

Theorem 9.1. *The satisfiability problem of the bimodal logic* S4 × S5 *can be reduced in logarithmic space to the satisfiability problem of the bimodal logic* K4 × S5*.*

We start with the definition of the reduction function $\widehat{T}$ translating bimodal formulas in the language $\mathcal{L}$ to bimodal formulas in $\mathcal{L}$ such that for all $\varphi \in \mathcal{L}$:

$$\varphi \text{ is S4} \times \text{S5-satisfiable} \iff \widehat{T}(\varphi) \text{ is K4} \times \text{S5-satisfiable.}$$

The problem that we face is that in general K4 × S5-models are not reflexive. To handle this we add to the original formula φ a formula that implies that all those instances of the reflexivity axiom scheme $\Box\psi \to \psi$ where $\Box\psi$ is a subformula of φ must hold true in all reachable points.

Definition 9.2 (Translation $\widehat{T}$)**.** For $\varphi \in \mathcal{L}$ we define a function $\widehat{T} : \mathcal{L} \to \mathcal{L}$ by

$$\widehat{T}(\varphi) := \varphi \wedge \bigwedge_{\Box\psi \,\in\, \mathrm{sf}(\varphi)} K((\Box\psi \to \psi) \wedge \Box(\Box\psi \to \psi)).$$

We claim that the function $\widehat{T}$ is indeed a reduction function from the satisfiability problem of S4 × S5 to the satisfiability problem of K4 × S5.

Proposition 9.3. *The function* $\widehat{T} : \mathcal{L} \to \mathcal{L}$ *satisfies, for all* $\varphi \in \mathcal{L}$*,*

$$\varphi \textit{ is } \text{S4} \times \text{S5-}\textit{satisfiable} \iff \widehat{T}(\varphi) \textit{ is } \text{K4} \times \text{S5-}\textit{satisfiable.}$$

Proof. Let φ be S4 × S5–satisfiable. Then there are an S4 × S5-commutator model $M = (W, \xrightarrow{\Diamond}, \xrightarrow{L}, \sigma)$ and a point $w \in W$ such that $M, w \models \varphi$. Since

the relation $\xrightarrow{\Diamond}$ in M is reflexive we have for all $w' \in W$ that $M, w' \models \bigwedge_{\Box\psi \,\in \mathrm{sf}(\varphi)}(\Box\psi \to \psi)$. Hence, $M, w \models \bigwedge_{\Box\psi \,\in \mathrm{sf}(\varphi)} K((\Box\psi \to \psi) \wedge \Box(\Box\psi \to \psi))$, in other words, $M, w \models \widehat{T}(\varphi)$. Furthermore, M satisfies the conditions for K4 × S5-commutator models because the relation $\xrightarrow{\Diamond}$ in M is transitive as well. Hence, $\widehat{T}(\varphi)$ is K4 × S5-satisfiable.
For the other direction of the equivalence let us assume that $\widehat{T}(\varphi)$ is K4 × S5-satisfiable. This implies that there exist a K4 × S5–commutator model $M' = (W', \xrightarrow{\Diamond'}, \xrightarrow{L'}, \sigma')$ and a point $w \in W'$ such that $M', w \models \widehat{T}(\varphi)$, that is

$$M', w \models \varphi \wedge \bigwedge_{\Box\psi \,\in \mathrm{sf}(\varphi)} K((\Box\psi \to \psi) \wedge \Box(\Box\psi \to \psi)).$$

We construct an S4 × S5–commutator model $M := (W, \xrightarrow{\Diamond}, \xrightarrow{L}, \sigma)$ for φ as follows. We construct M as a rooted model, where all points are reachable from the point w:

$$W := \{v \in W' \mid w \xrightarrow{L'} v \text{ or } (\exists w')\,(w \xrightarrow{L'} w' \text{ and } w' \xrightarrow{\Diamond'} v)\}.$$

We define the relations $\xrightarrow{L}$ and $\xrightarrow{\Diamond}$ on W simply by

$$\begin{aligned} \xrightarrow{L} &:= \xrightarrow{L'} \cap (W \times W), \\ \xrightarrow{\Diamond} &:= (\xrightarrow{\Diamond'} \cap (W \times W)) \cup \{(v, v) \mid v \in W\}. \end{aligned}$$

Finally,

$$\sigma(A) := \sigma'(A) \cap W,$$

for all $A \in AT$.
It is clear that $w \in W$, and it is straightforward to see that M is an S4 × S5-commutator model. We claim that $M, w \models \varphi$. By induction on the structure of ψ we show the stronger assertion:

$$M, v \models \psi \iff M', v \models \psi,$$

for all $v \in W$ and for all $\psi \in \mathrm{sf}(\varphi)$. We distinguish the following cases:

- Case $\psi = A \in AT$. In this case the claim follows directly from the definition of σ.
- Case $\psi = \neg\chi$ or $\psi = (\chi_1 \wedge \chi_2)$. In both cases the claim follows directly from the induction hypothesis.

- Case $\psi = K\chi$. Let us first assume $M', v \models K\chi$. We wish to show $M, v \models K\chi$. Consider an arbitrary $v' \in W$ with $v \xrightarrow{L} v'$. It is sufficient to show $M, v' \models \chi$. Note that the definition of $\xrightarrow{L}$ implies $v \xrightarrow{L'} v'$. Hence, we have $M', v' \models \chi$. By induction hypothesis we obtain $M, v' \models \chi$.

  For the other direction let us assume that $M, v \models K\chi$. We wish to show $M', v \models K\chi$. Consider an arbitrary $v' \in W'$ with $v \xrightarrow{L'} v'$. It is sufficient to show $M', v' \models \chi$. Using left commutativity of M', from $v \in W$ and $v \xrightarrow{L'} v'$ we conclude $v' \in W$ and $v \xrightarrow{L} v'$. Hence, $M, v \models K\chi$ implies $M, v' \models \chi$. By induction hypothesis we obtain $M', v' \models \chi$.

- Case $\psi = \Box\chi$. Let us first assume $M', v \models \Box\chi$. We wish to show $M, v \models \Box\chi$. Consider an arbitrary $v' \in W$ with $v \xrightarrow{\Diamond} v'$. It is sufficient to show $M, v' \models \chi$. Note that the definition of $\xrightarrow{\Diamond}$ implies that $v \xrightarrow{\Diamond'} v'$ or $v = v'$. In the first case, $v \xrightarrow{\Diamond'} v'$, the assumption $M', v \models \Box\chi$ directly implies $M', v' \models \chi$. By induction hypothesis we obtain $M, v' \models \chi$. In the second case, $v = v'$, we use the fact that $M', w \models K((\Box\chi \to \chi) \wedge \Box(\Box\chi \to \chi))$ and $v \in W$ imply $M', v \models (\Box\chi \to \chi)$. Together with $M', v \models \Box\chi$ this implies $M', v \models \chi$, hence, $M', v' \models \chi$. By induction hypothesis we obtain $M, v' \models \chi$ as well.

  For the other direction let us assume that $M, v \models \Box\chi$. We wish to show $M', v \models \Box\chi$. Consider an arbitrary $v' \in W'$ with $v \xrightarrow{\Diamond'} v'$. It is sufficient to show $M', v' \models \chi$. But from $v \in W$ and $v \xrightarrow{\Diamond'} v'$ we conclude $v' \in W$ and $v \xrightarrow{\Diamond} v'$. Hence, $M, v \models \Box\chi$ implies $M, v' \models \chi$. By induction hypothesis we obtain $M', v' \models \chi$.

$\Box$

Proof of Theorem 9.1. Let $\Sigma =: \{(,), \neg, \Box, K, \wedge, x, 0, 1\}$ be the alphabet over which bimodal formulas are defined. In order to prove the assertion we have to show that there is a logspace computable function $\widetilde{T} : \Sigma^* \to \Sigma^*$ such that, for all $\varphi \in \Sigma^*$,

$$\begin{aligned} &(\varphi \in \mathcal{L} \text{ and } \varphi \text{ is S4} \times \text{S5-satisfiable}) \\ &\quad \iff \quad (\widetilde{T}(\varphi) \in \mathcal{L} \text{ and } \widetilde{T}(\varphi) \text{ is K4} \times \text{S5-satisfiable}). \end{aligned}$$

We define such a function $\widetilde{T}$ by formulating an algorithm for computing it that works in logarithmic space.
So, let $\varphi \in \Sigma^*$ be the input string. According to Corollary 8.7 we can first check in logarithmic space whether φ is a bimodal formula or not, that is,

whether φ is an element of $\mathcal{L}$ or not. If not then the algorithm outputs $\widetilde{T}(\varphi) := \wedge$ (which is certainly not a bimodal formula). If, on the other hand, φ is a bimodal formula then we wish to compute and print $\widetilde{T}(\varphi) := \widehat{T}(\varphi)$ as defined in Definition 9.2. The algorithm that prints $\widehat{T}(\varphi)$ works as follows.

1. It prints φ.
2. For each $\Box\psi \in \mathrm{sf}(\varphi)$ it prints the string
$$\wedge K((\Box\psi \to \psi) \wedge \Box(\Box\psi \to \psi)).$$

Of course, for the second part, for every occurrence of the symbol $\Box$ in φ one has to determine the uniquely determined formula ψ beginning in φ immediately to the right of this occurrence of $\Box$. Then one has to print the string above. All this can be done using several binary counters. First one sets a binary counter called *StartOfPsi* to the value of the position to the right of the current occurrence of $\Box$. In order to compute the correct value of a binary counter *EndOfPsi* that is supposed to be the position of the rightmost symbol in ψ one proceeds as follows. Starting from the position *StartOfPsi* one reads the given string from left to right. As long as the read symbol is $\neg$ or $\Box$ or K one continues reading. At some stage one will either read an x or an opening bracket (. If one reads an x then *EndOfPsi* is set to the position of the rightmost bit, that is, the rightmost 0 or 1, such that between this symbol and the just read occurrence of x there are only bits. If one reads an opening bracket then *EndOfPsi* is set to the position of the first closing bracket to the right of this opening bracket such that the string between these two brackets contains as many closing as opening brackets. Note that all this can be done in logarithmic space.
Finally, using the two binary counters *StartOfPsi* and *EndOfPsi* containing the positions of the first and the last symbol of ψ it is clear that one can print the string "$\wedge K((\Box\psi \to \psi) \wedge \Box(\Box\psi \to \psi))$", again using additional binary counters that use only logarithmic space. This ends the description of the computation in logarithmic space of the described reduction function $\widetilde{T}$. $\Box$

Chapter 10

Conclusion

We have investigated the complexity of three combined logics with similar properties.

- One component of the regarded combinations is the logic S5 with excellent algorithmic behavior. This is due to the fact that in models of S5 the accessibility relation is an equivalence relation.
- The second component is in all of the considered cases a logic such that the models have a transitive accessibility relation.
- These components are combined in such a way that their models have at least the left commutativity property.

We used recursive tableau algorithms to show that the satisfiability problems of the considered logics are in ESPACE. To show that these satisfiability problems are EXPSPACE-hard we proved that any language recognized by an Alternating Turing Machines working in exponential time can be reduced in logarithmic space to the satisfiability problem of SSL (and also to the satisfiability problem of $S4 \times S5$). Then we proved that the satisfiability problem of SSL can be reduced in logarithmic space to the satisfiability problem of $S4 \times S5$ and that this problem can be reduced in logarithmic space to the satisfiability problem of $K4 \times S5$.
As further work one could investigate whether the techniques used here and the achieved results can help to determine the complexity of the satisfiability problem of other logics that combine S5 with a transitive logic in such a way that at least left commutativity holds true. We think especially of extensions of SSL with additional axioms as suggested by Dabrowski, Moss and Parikh [22]:

WD	$\Diamond\Box\varphi \rightarrow \Box\Diamond\varphi$ sound for weakly directed spaces
Un	$\Diamond\varphi \wedge L\Diamond\psi \rightarrow \Diamond[\Diamond\varphi \wedge L\Diamond\psi \wedge K\Diamond L(\varphi \vee \psi)]$ sound for spaces closed under binary unions
topologic	SSL + (WD) + (Un) sound for lattice spaces, complete for topological spaces – even for complete lattice spaces

And there are also many extensions of SSL developed by Heinemann for which the problem of their computational complexity is still open.

Bibliography

[1] H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic, 27*, pages 217–274., 1998.

[2] F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001.

[3] P. Balbiani, H. van Ditmarsch, and A. Kudinov. Subset space logic with arbitrary announcements. In *Indian Conference on Logic and Its Applications*, pages 233–244. Springer, 2013.

[4] A. Baltag, V. Fiutek, and S. Smets. DDL as an "internalization" of dynamic belief revision. In *Krister Segerberg on Logic of Actions*, pages 253–280. Springer, 2014.

[5] A. Baltag, L. S. Moss, and S. Solecki. The logic of public announcements, common knowledge, and private suspicions. In *Readings in Formal Epistemology*, pages 773–812. Springer, 2016.

[6] A. Baltag, A. Özgün, and A. L. V. Sandoval. Topo-logic as a dynamic-epistemic logic. In *International Workshop on Logic, Rationality and Interaction*, pages 330–346. Springer, 2017.

[7] A. Baltag, A. Özgün, and A. L. V. Sandoval. APAL with memory is better. In *International Workshop on Logic, Language, Information, and Computation*, pages 106–129. Springer, 2018.

[8] A. Baltag and B. Renne. Dynamic epistemic logic. *Stanford Encyclopedia of Philosophy*, 2016(Fall), 2016.

[9] A. Baltag, H. van Ditmarsch, and L. S. Moss. Epistemic logic and information update. In *Handbook of the Philosophy of Information*, pages 361–456. Elsevier, Amsterdam, 2008.

[10] C. Başkent. *Topics in Subset Space Logic An Introduction to the Geometry of Dynamic Epistemology.* VDM Verlag Dr. Müller, Saarbrücken, 2010.

[11] C. Başkent. Public announcement logic in geometric frameworks. *Fundamenta Informaticae*, 118(3):207–223, 2012.

[12] E. W. Beth. *Semantic Entailment and Formal Derivability.* North-Holland, Amsterdam, 1955.

[13] A. Bjorndahl. Subset space public announcement logic revisited. *arXiv preprint arXiv:1302.4009*, 2013.

[14] A. Bjorndahl. Topological subset space models for public announcements. In *Jaakko Hintikka on Knowledge and Game-Theoretical Semantics*, pages 165–186. Springer, 2018.

[15] P. Blackburn, M. De Rijke, and Y. Venema. *Modal Logic.* Cambridge University Press, 2001.

[16] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem.* Springer, Berlin, 1997.

[17] S. R. Buss. The Boolean formula value problem is in ALOGTIME. In A. V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, New York*, pages 123–131. ACM, 1987.

[18] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *ACM*, 28(1):114–133, Jan. 1981.

[19] A. K. Chandra and L. J. Stockmeyer. Alternation. In *17th Annual Symposium on Foundations of Computer Science, 1976*, pages 98–108. IEEE, 1976.

[20] P. Clote. Computation models and function algebras. In *Handbook of Computability Theory*, volume 140 of *Stud. Logic Found. Math.*, pages 589–681. North-Holland, Amsterdam, 1999.

[21] S. A. Cook. The complexity of theorem-proving procedures. In M. A. Harrison, R. B. Banerji, and J. D. Ullman, editors, *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158. ACM, 1971.

[22] A. Dabrowski, L. S. Moss, and R. Parikh. Topological reasoning and the logic of knowledge. *Ann. Pure Appl. Logic*, 78:73–110, 1996.

[23] R. Fagin, J. Y. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. MIT press, 1995.

[24] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.

[25] M. Fitting. *Proof Methods for Modal and Intuitionistic Logics*, volume 169. Springer Science & Business Media, 1983.

[26] M. Fitting. Modal proof theory. In *Handbook of Modal Logic*, volume 3 of *Stud. Log. Pract. Reason.*, pages 85–138. Elsevier B. V., Amsterdam, 2007.

[27] D. M. Gabbay and V. B. Shehtman. Products of modal logics, part 1. *Logic Journal of IGPL*, 6(1):73–146, 1998.

[28] D. Gabelaia, A. Kurucz, F. Wolter, and M. Zakharyaschev. Products of transitive modal logics. *The Journal of Symbolic Logic*, 70(03):993–1021, 2005.

[29] K. Georgatos. Knowledge theoretic properties of topological spaces. In *International Conference Logic at Work on Knowledge Representation and Reasoning Under Uncertainty*, pages 147–159. Springer-Verlag, 1992.

[30] K. Georgatos. *Modal Logics for Topological Spaces*. PhD thesis, The City University of New York, 1993.

[31] K. Georgatos. Knowledge on treelike spaces. *Studia Logica*, 59(2):271–301, 1997.

[32] K. Georgatos. Updating knowledge using subsets. *Journal of Applied Non-Classical Logics*, 21(3-4):427–441, 2011.

[33] K. Gödel. Eine Interpretation des intuitionistischen Aussagenkalküls, Ergebnisse eines mathematischen Kolloquiums 4 (1933) 39-40. reprinted and translated in: S. Feferman et al.(eds.), Kurt Gödel. Collected Works. Vol. 1, 1986.

[34] R. Goré. Tableau methods for modal and temporal logics. In *Handbook of Tableau Methods*, pages 297–396. Kluwer Acad. Publ., Dordrecht, 1999.

[35] G. Governatori. Labelled modal tableaux. In *Advances in Modal Logic,*, volume 7, pages 87–110, 2008.

[36] E. Grädel. Why are modal logics so robustly decidable? In G. Paun, G. Rozenberg, and A. Salomaa, editors, *Current Trends in Theoretical Computer Science. Entering the 21st Century*, pages 393–408. World Scientific, 2001.

[37] G. Grätzer. *Lattice Theory: Foundation.* Birkhäuser/Springer Basel AG, Basel, 2011.

[38] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, 1992.

[39] D. Harel, D. Kozen, and J. Tiuryn. Dynamic logic. In *Handbook of Philosophical Logic*, pages 99–217. Springer, 2001.

[40] B. Heinemann. Topological modal logics satisfying finite chain conditions. *Notre Dame Journal of Formal Logic*, 39(3):406–421, 1998.

[41] B. Heinemann. Temporal aspects of the modal logic of subset spaces. *Theoretical Computer Science*, 224(1-2):135–155, 1999.

[42] B. Heinemann. Regarding overlaps in 'topologic'. *Advances in Modal Logic*, 6:259–277, 2006.

[43] B. Heinemann. A hybrid logic for reasoning about knowledge and topology. *Journal of Logic, Language and Information*, 17(1):19–41, 2008.

[44] B. Heinemann. Topology and knowledge of multiple agents. In *Ibero-American Conference on Artificial Intelligence*, pages 1–10. Springer, 2008.

[45] B. Heinemann. The Cantor space as a generic model of topologically presented knowledge. In *International Computer Science Symposium in Russia*, pages 169–180. Springer, 2010.

[46] B. Heinemann. Logics for multi-subset spaces. *Journal of Applied Non-Classical Logics*, 20(3):219–240, 2010.

[47] B. Heinemann. Subset spaces modeling knowledge-competitive agents. In *International Conference on Knowledge Science, Engineering and Management*, pages 3–14. Springer, 2015.

[48] B. Heinemann. Augmenting subset spaces to cope with multi-agent knowledge. In *International Symposium on Logical Foundations of Computer Science*, pages 130–145. Springer, 2016.

[49] B. Heinemann. A subset space perspective on agents cooperating for knowledge. In *International Conference on Knowledge Science, Engineering and Management*, pages 503–514. Springer, 2016.

[50] J. Hintikka. Form and content in quantification theory. *Acta Philosophica Fennica*, 8(7):55, 1955.

[51] M. Huth and M. D. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems (2. ed.).* Cambridge University Press, 2004.

[52] O. H. Ibarra, T. Jiang, and B. Ravikumar. Some subclasses of context-free languages in NC1. *Information Processing Letters*, 29(3):111–117, 1988.

[53] S. Kanger. *Provability in Logic.* Stockholm Studies in Philosophy 1. Almqvist and Wiksell, Stockholm, 1957.

[54] A. S. Kechris. *Classical Descriptive Set Theory.* Graduate Texts in Mathematics. 156. Berlin: Springer-Verlag, 1995.

[55] D. Kozen. On parallelism in Turing machines. In *17th Annual Symposium on Foundations of Computer Science*, pages 89–97. IEEE, 1976.

[56] S. A. Kripke. A completeness theorem in modal logic. *The Journal of Symbolic Logic*, 24:1–14, 1959.

[57] S. A. Kripke. Semantical analysis of modal logic I, normal modal propositional calculi. *Zeitschrift für mathemathische Logik und Grundlagen der Mathematik*, 9(5-6):67–96, 1963.

[58] G. Krommes. A new proof of decidability for the modal logic of subset spaces. In *Eighth ESSLLI Student Session*, pages 137–148. Citeseer, 2003.

[59] G. Krommes. Untersuchungen zur modalen Logik von Mengenräumen: Vollständigkeit, Entscheidbarkeit, Komplexität. Master's thesis, Fern-Universität Hagen, 2003.

[60] A. Kurucz. Combining modal logics. In *Handbook of Modal Logic*, volume 3 of *Stud. Log. Pract. Reason.*, pages 869–924. Elsevier B. V., Amsterdam, 2007.

[61] A. Kurucz, F. Wolter, M. Zakharyaschev, and D. M. Gabbay. *Many-Dimensional Modal Logics: Theory and Applications, Volume 148 (Studies in Logic and the Foundations of Mathematics).* North Holland, 2003.

[62] A. Kurucz and M. Zakharyaschev. A note on relativised products of modal logics. *Advances in Modal Logic*, 4:221–242, 2002.

[63] R. E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing*, 6(3):467–480, 1977.

[64] M. Lange and C. Lutz. 2-EXPTIME lower bounds for propositional dynamic logics with intersection. *The Journal of Symbolic Logic*, 70(4):1072–1086, 2005.

[65] C. I. Lewis and C. H. Langford. *Symbolic Logic.* Dover publications New York, 1959.

[66] H. MacColl. Symbolic reasoning. *Mind, 5:54*, 1880.

[67] M. Marx. Complexity of products of modal logics. *Journal of Logic and Computation*, 9(2):197–214, 1999.

[68] J. C. C. McKinsey. A solution of the decision problem for the lewis systems s2 and s4, with an application to topology. *The Journal of Symbolic Logic*, 6(4):pp. 117–134, 1941.

[69] J. C. C. McKinsey and A. Tarski. The algebra of topology. *Annals of Mathematics*, pages 141–191, 1944.

[70] J. C. C. McKinsey and A. Tarski. Some theorems about the sentential calculi of Lewis and Heyting. *The Journal of Symbolic Logic*, 13(1):1–15, 1948.

[71] L. S. Moss and R. Parikh. Topological reasoning and the logic of knowledge. In *Proceedings of the Fourth Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 95–105. Morgan Kaufmann Publishers Inc., 1992.

[72] I. Németi. Decidable versions of first order logic and cylindric-relativized set algebras. In D. G. L. Csirmaz and M. de Rijke, editors, *Logic Colloquium*, volume 92, pages 171–241. CSLI Publications, 1995.

[73] E. Pacuit and R. Parikh. The logic of communication graphs. In *International Workshop on Declarative Agent Languages and Technologies*, pages 256–269. Springer, 2004.

[74] C. H. Papadimitriou. *Computational Complexity.* Amsterdam: Addison-Wesley Publishing Company, 1994.

[75] R. Parikh, L. S. Moss, and C. Steinsvold. Topology and epistemic logic. In *Handbook of Spatial Logics*, pages 299–341. Springer, 2007.

[76] W. J. Paul, E. J. Prauß, and R. Reischuk. On alternation. *Acta Informatica*, 14(3):243–255, 1980.

[77] J. Plaza. Logics of public communications. *Synthese*, 158(2):165–179, 2007.

[78] V. R. Pratt. Semantical considerations on Floyd-Hoare logic. In *17th Annual Symposium on Foundations of Computer Science*, pages 109–121. IEEE, 1976.

[79] W. Rautenberg. Modal tableau calculi and interpolation. *Journal of Philosophical Logic*, 12(4):403–423, 1983.

[80] B. Schröder. *Ordered sets.* Birkhäuser/Springer, second edition, 2016. An introduction with connections from combinatorics to topology.

[81] K. Schütte. Ein System des verknüpfenden Schliessens. *Archiv für mathematische Logik und Grundlagenforschung*, 2(2-4):55–67, 1956.

[82] K. Segerberg. Two-dimensional modal logic. *Journal of Philosophical logic*, 2(1):77–96, 1973.

[83] V. B. Shehtman. Two-dimensional modal logic. *Matematicheskie Zametki*, 23(5):759–772, 1978.

[84] R. R. Smullyan. *First-Order Logic*, volume 43. Springer Science & Business Media, 2012.

[85] E. Spaan. *Complexity of Modal Logics.* PhD thesis, Universiteit van Amsterdam, 1993.

[86] R. H. Thomason. Combinations of tense and modality. In *Handbook of Philosophical Logic*, pages 135–165. Springer, 1984.

[87] J. van Benthem. *Modal Correspondence Theory.* PhD thesis, Mathematical Institute, University of Amsterdam, 1976.

[88] J. van Benthem. Correspondence theory. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic.* Reidel, Dordrecht, 1984.

[89] J. van Benthem. *Logical Dynamics of Information and Interaction.* Cambridge University Press, 2011.

[90] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*, volume 337. Springer Science & Business Media, 2007.

[91] M. Y. Vardi. Why is modal logic so robustly decidable? *Descriptive Complexity and Finite Models, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 31, AMS:149–184, 1997.

[92] Y. N. Wáng. A two-dimensional hybrid logic of subset spaces. In *Indian Conference on Logic and Its Applications*, pages 196–209. Springer, 2009.

[93] Y. N. Wáng and T. Ågotnes. Multi-agent subset space logic. In *IJCAI*, pages 1155–1161, 2013.

[94] Y. N. Wáng and T. Ågotnes. Subset space public announcement logic. In *Indian Conference on Logic and Its Applications*, pages 245–257. Springer, 2013.

[95] M. A. Weiss and R. Parikh. Completeness of certain bimodal logics for subset spaces. *Studia Logica*, 71(1):1–30, 2002.

[96] X. Wen, H. Liu, and F. Huang. An alternative logic for knowability. In *International Workshop on Logic, Rationality and Interaction*, pages 342–355. Springer, 2011.