A path characterization of validity for multimodal logics

Samuli Antti Kalervo Heilala
School of Computer Science
McGill University, Montreal

June 2009

A thesis submitted to McGill University in partial fulfilment
of the requirements of the degree of Master of Science

Contents

Abstract

Modal logics were formalized in the early to mid-20th century to pin down the notion of truth qualified by modalities such as "necessity", "possibility", and "belief", which arise in philosophy and natural languages. They have since outgrown their philosophical trappings and have found applications in type systems and as description languages in many domains of computer science and artificial intelligence.

Although they frequently exhibit redundancies in their proof spaces, analytic tableau systems in the style of Smullyan have emerged as popular proof formalisms for modal logics. Inspired by previous work on matrix characterizations by Bibel, Andrews, and Wallen, we propose a new characterization of validity for a family of multimodal logics with interacting accessibility relations which avoids the redundancies common in tableau systems. Our goal is not to produce a high-performance theorem prover for multimodal logics, but to present a fresh way of understanding and establishing multimodal logical validity.

Résumé

Les logiques modales ont été formalisées vers le milieu du 20ième siècle avec le but de comprendre la notion de vérité qualifiée par des modes tels que “la nécessité”, “la possibilité”, et “la croyance”, qui proviennent de la philosophie et des langages naturels. Ces modes, depuis évolués au-delà de leurs origines philosophiques, trouvent plusieurs applications dans la sémantique statique des langages de programmation et chez les logiques de description de plusieurs domaines de l’informatique et de l’intelligence artificielle.

Bien que leurs espaces démonstrationnels présentent des redondances, les systèmes de tableaux analytiques sont un formalisme de démonstration populaire. Nous proposons une nouvelle caractérisation de validité pour une famille de logiques multimodales définies avec des relations d’accessibilité interagissantes qui évite les redondances présentes chez les systèmes de tableaux. Nos innovations sont inspirées par les caractérisations matricielles présentées auparavant par Bibel, Andrews, et Wallen. Notre but n’est pas de produire un démonstrateur automatique de théorèmes de haute performance, mais de présenter une façon originale de comprendre et concevoir la validité logique multimodale.

Acknowledgements

First and foremost, I would like to thank my thesis supervisor, Brigitte Pientka, for her tremendous patience and understanding, as well as her keen insight and sound advice in both technical matters and details of presentation. For financial support, I express my deepest gratitude to M. André Courtemanche, the School of Computer Science at McGill, and my parents. For translating the abstract into French, I owe a debt of honour to Maja Frydrychowicz. Finally, I thank my family for their unwavering support, my friends for periodically reminding me, in no uncertain terms, to get back to work, the staff of the School of Computer Science at McGill for keeping things organized, and the internet for being a truly magnificent research and procrastination tool.

Chapter 1

Introduction

The study of modal logics as a mathematical discipline began in the early 20th century with the work of MacColl [34] and Lewis [33] in formalizing notions of qualified truth that arise in philosophy and natural languages, usually in the form of modal qualifiers or operators such as "necessarily", "possibly", "certainly", "probably", and so on. Although modal logics are sometimes still characterized as logics of necessity and possibility, they have since outgrown their philosophical trappings and have proved themselves flexible tools for working with and reasoning about many kinds of relational structures. In particular, they have found applications in type systems and as description languages in many domains of computer science and artificial intelligence, some of which we will sketch later in this chapter.

Prior to the 1960s, the study of modal logics revolved mostly around axiomatic Hilbert systems for formal languages of logic which had been extended by one or more modal operators. Around the early 1960s, ideas developed by, amongst others, Kanger [26], Hintikka [24], and Kripke [30, 31] rejuvenated the field with the introduction of a relational semantics and model theory for modal logic. This relational semantics is often called a *Kripke semantics*, and the idea that lies at its heart, which we will review formally in chapter 2, is that modal formulas make statements whose truth can

be seen as being relative to a particular world in a larger collection of possible worlds which are accessible from each other in certain ways. Such a collection of worlds, along with its accessibility relation, is sometimes called a Kripke structure. Multimodal logic enriches the language of modal logic by more modal operators. Properties of possible interactions between multiple modalities are also encoded by the Kripke semantics.

While the semantic and algebraic aspects of modal logics have generated interest for some time, modal theorem proving and the proof theory of modal and multimodal logics are still relatively young fields. Considering the increasing number of computer science domains in which modal logics are finding use, a broader understanding of proof systems for them, as well as the proof spaces they induce, seems worthwhile. Analytic tableau systems in the style of Smullyan [44], along with their cousins, sequent calculi, are amongst the most common types of proof systems proposed for modal logics, probably because they are concisely defined, relatively user-friendly, and allow large families of logics to be characterized in a more or less uniform way (see Goré [21] for an overview).

However, the proof spaces induced by tableau calculi exhibit several types of redundancies which introduce undesirable nondeterminism to the task of constructing a tableau proof. They can also cause tableau proofs to appear different from one another when the reasoning underlying them is, in fact, the same. Wallen [48] identifies several types of redundancies within the proof spaces of particular sequent calculi for various modal logics and points out that these redundancies also appear in equivalent tableau systems. We will discuss them in more detail later in this chapter and in chapter 4. Based on previous work on connection-based theorem proving by Prawitz [41], Bibel [6], and Andrews [2], Wallen goes on to describe a *matrix characterization of validity* for some modal logics. His formalism induces an alternative proof space in which the redundancies associated with sequent and tableau calculi are absent. Similar matrix characterizations have been developed for multiplicative (exponential)

linear logic by Kreitz et al. [28] and Kreitz and Mantel [27].

The key idea behind such matrix characterizations is that many of the choices that need to be made during the construction of a traditional tableau or sequent calculus derivation—attempting to establish the validity of a given query formula—can be modularly abstracted out of the derivation. The structure of the query formula dictates the order of many steps of the proof, and these steps are partially independent of choices made regarding modal qualifiers or, in the case of first-order logics, first-order quantifiers. Tableau systems can be concisely defined because all types of choices made during a derivation are thrown into the same bag. The unfortunate side-effect of this is that subtleties in the behaviours of the logics may be obscured, and the different types of choices made during the construction of a tableau derivation become inextricably intertwined.

Many of the domains in which modal logics are useful modelling tools require fairly rich modal languages, so developing alternative proof formalisms, such as matrix characterizations of validity, for a wider class of modal logics, in particular multimodal logics with interacting accessibility relations, seems an obvious next step. The main contribution of this thesis is the development of such an alternative characterization of validity for a family of multimodal logics.

Our approach is rather different from Wallen's, however, whose matrix characterization is inspired by Fitting's prefixed tableaus [17]. In order to support multimodal logics with richly interacting accessibility relations, which traditional prefixed tableau systems struggle with, we use as a basis a tableau calculus similar to those of Nerode [39] and Baldoni [4], in which modal accessibility relations are manipulated directly rather than encoded implicitly in the structure of prefixes. The resulting characterization has a flavour different from Wallen's, and we shall call it a *path characterization of validity* for reasons that will become clear in chapter 4.

Note that while proof search procedures may benefit from the cleaner proof space

induced by a matrix or path characterization, our goal is not the development of a high-performance theorem prover for multimodal logics based on a path characterization, but the characterization itself. We will touch on proof search only briefly in chapter 5.

In the rest of this chapter, we will describe computer science applications of modal logics, related work, and our contributions in more detail.

1.1 Applications

In this section, we will sketch some applications of multimodal logics in various domains of computer science. Our discussion here is informal, our aim being only to demonstrate some ways in which modal qualification of truth judgements is useful in real applications.

Knowledge and belief in multiagent modelling The modalities of necessity and possibility (called *alethic* modalities) are related to the modalities of knowledge and belief (called *epistemic* and *doxastic* modalities, respectively). A judgement of truth, such as "A is true", can be augmented by judgements of knowledge or belief, such as "I know A to be true" or "I believe A to be true". The qualified truth judgements typically also incorporate the identity of the agent doing the knowing or believing. To wit, "agent i knows A to be true" or "agent i believes A to be true".

The formal study of epistemic logic began with von Wright [46] and Hintikka [24], who gave a Kripke semantics for knowledge and belief. Note that to correctly model our intuitions regarding these modalities, the respective modal operators are required to behave in particular ways. For instance, it is generally expected—as an inherent property of knowledge, as opposed to belief—that if an agent knows something to be true, then it must indeed be true. This axiom can be stated schematically as "if agent i knows A to be true, then A is true". Belief, on the other hand, does not have this

property. Other properties of knowledge and belief may also be assumed, depending on the domain. The axiom of *positive introspection*, for instance, states that "if agent i knows A to be true, then he knows that he knows A to be true".

In the context of artificial intelligence, multimodal epistemic logic can be used to model multiagent reasoning, with interacting accessibility relations representing interacting knowledge or belief amongst agents. For concrete examples, see for instance Fagin et al. [14].

Program behaviour This category encompasses logics for reasoning explicitly about program behaviour, as well as type systems based on multimodal logics.

Examples in the first class include temporal logic (see for instance Venema [45] or Blackburn, de Rijke, and Venema [8]) and dynamic logic (see for instance Harel, Kozen, and Tiuryn [22]), which can formalize statements about program states and behaviour, as well as proofs of their truth or validity. The language of dynamic logic is richer than what we will be covering in this thesis, but it is also modal in the sense that truth is qualified, not by necessity, possibility, or belief, but by actions. To wit, judgements may have the form "every execution of program i leads to a state in which A is true" or "some execution of program i leads to a state in which A is true". Here, i is not necessarily simply an atomic program name, but could be a sequence of instructions or more complex expressions: it could be a description of the program itself.

In temporal logic, the program is fixed, so truth is qualified not by individual programs, but by the time during the execution of the fixed program. Statements of interest include ones such as "it will always be the case that A is true", "it has, at some point in the past, been the case that A was true", "A is true now", and so on. Being able to reason temporally is naturally useful in formalizing and proving properties related to program correctness.

An interesting related modal logic with applications in the formal verification of

hardware is Fairtlough and Mendler's lax logic [15]. Although lax logic enjoys only a single modality, it can potentially be combined with others, such as temporal operators, and this is where the ability to define interactions between modalities becomes interesting. Let us return briefly to multiagent knowledge and belief systems for clearer intuition. Even if the agents in our domain are defined to know and believe more or less independently, adding the ability to reason temporally, for instance, makes it possible to define or specify useful interactions between the temporal, epistemic, and doxastic modalities, such as faithful stubbornness: "if agent i believes A to be true, then he will always believe A to be true".

The class of type systems inspired by modal logics is rich, since modalities can encode a multitude of properties of interest to a programmer. If the possible worlds of a Kripke structure are interpreted as physical locations in a network, then modal formulas become types of mobile code in a distributed computing environment [37, 38], and the type-correctness of such programs guarantees their mobility, taking local resources into account. If possible worlds are treated as stages of a computation, then what emerges is a type system for reasoning about staged computation, including type-level correctness guarantees for runtime code generation and partial evaluation [13]. In the domain of information flow, possible worlds can be interpreted as security levels associated with confidential data, with an accessibility relation defining the order or structure of the security hierarchy. Modal types then encode data at various security levels, and the type system can ensure that high-security data does not leak into low-security programs [36].

Description logics Descriptions logics (DLs) are a family of knowledge representation languages used to encode domain-specific knowledge and access and reason about it in a structured and domain-independent way [3]. As such, they are of interest to artificial intelligence researchers, who strive to develop generic reasoning algorithms for description logics without being bound to particular domains. Emerging in the

1980s as descendents of semantic networks and frame-based systems—in particular the KL-ONE representation system of Brachman and Schmolze [9]—description logics lie between propositional and first-order logics in that statements of DLs generally have first-order equivalents, but the structure and constraints of the language and knowledge base can make reasoning in DLs tractable and more efficient than full-fledged first-order proof search.

Like description logics, many modal logics are also intermediate between propositional and first-order logics. In fact, the simplest description logic, called $\mathcal{ALC}$, is nothing more than a notational variant of the classical multimodal logic **K** [3]. While the expressiveness of DLs has grown far beyond $\mathcal{ALC}$, they remain multimodal languages with a more flexible syntax. Consequently, methods and representations developed for multimodal logics may find applications in the field of description logics and vice versa.

1.2 Related work

As we mentioned earlier in this chapter, the matrix or connection-based characterization, pioneered for first-order logic by Prawitz [41] and developed further by Bibel [6] and Andrews [2], was adapted to a class of modal logics, as well as to intuitionistic logic, by Wallen [47, 48] and to multiplicative linear logics by Kreitz et al. [28] and Kreitz and Mantel [27]. While the inspiration for our work comes from that of Wallen, the details are rather different, owing to the different tableau calculi that underlie his and our work.

Wallen motivates his matrix characterization by studying the redundancies of a sequent calculus system in which modal information is maintained implicitly. Technically, his characterization is closely related to Fitting's prefixed tableau calculus [18], which is well-suited for the class of unimodal logics that Wallen considers. However,

difficulties can arise when prefixed tableau systems such as Fitting's or Massacci's [35] are extended to more expressive logics, in particular to multimodal logics with interacting modalities. There are also potential complications in the unimodal case; the frame condition of symmetry, for instance, is conspicuously absent in Wallen's treatment. Baldoni [4] discusses the issues in some detail and proposes an alternative tableau system for multimodal logics, in which modal accessibility relations, encoded syntactically by prefixes in Fitting's systems, are represented and manipulated more explicitly.

In both Fitting's and Baldoni's systems, accessibility relations exist as concrete parts of tableau nodes, but the distinction between maintaining them as explicit relations rather than ones whose structures are implied by prefixes has important consequences. In a system of prefixes—or any system in which the structure of a putative countermodel is represented by complex labels representing worlds—every condition on the represented accessibility relation needs to be translated into a corresponding condition on the prefixes or complex labels, since some of the semantics has been imported into the syntax. If the relation is explicitly maintained in tableau nodes, then the "translation" is much more direct.

Moreover, since Wallen's sequent calculus maintains modal information implicitly, he must deal with *non-permutability*, the problem that modal rule applications cannot be applied in any order, with the consequence that it may be necessary to backtrack during proof construction because of previously made choices. This is one type of redundancy that Wallen's matrix characterization addresses: the sequent calculus formalism cannot inspect the deep modal structure of the query formula, while the matrix characterization can. Another type of redundancy is *irrelevance*, caused by the inability of tableau and sequent calculi to inspect the deep syntactic structure of the query formula without decomposing it one connective at a time. Our path characterization is built on a different tableau system which avoids non-permutability by

design, but the path characterization goes one step further by eliminating irrelevance and separating the modal and structural choices made in a derivation.

Goré's survey [21] provides a good overview of tableau methods for unimodal logics, though the focus is on systems in which the accessibility relation is implicit. When discussing systems where it is explicit, more time is spent on prefixed systems. The calculi most similar to ours in spirit are those of Baldoni [4] and Nerode [39] (the former of which was inpired by the latter), but our presentation also differs starkly from theirs, since our tableau calculus is designed specifically to facilitate the subsequent development of the path characterization. In some ways, our tableau calculus is a "path characterization light" formulated expressly to lead to the path characterization proper, and we will point out the parallels between our two formalisms as we develop them.

1.3 Contributions and organization

The primary contribution of this thesis is the development and proof of correctness of a path characterization of validity for a family of multimodal logics. The characterization is reminiscent of Wallen's matrix characterization [48], but is defined on a different class of logics and in a way that is closer to the relational semantics of the logics being considered. A secondary contribution is an alternative formulation of a tableau system for the multimodal logics of interest, which we use to prove the correctness of the path characterization.

The remainder of the thesis is organized as follows.

- Chapter 2 contains a brief overview of the syntax and semantics of the multimodal logics we will be dealing with, some proof theoretic preliminaries and notation, and several background results on relations and closures.

- Chapter 3 contains the development and proof of correctness of our tableau

characterization of validity for multimodal logics.

- Chapter 4 contains the development and proof of correctness of our path characterization of validity for multimodal logics.

- Chapter 5 concludes with some discussion of future work, including other logics to which our approach may be extended.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university or other institution. To the best of our knowledge, except where due reference is made, material presented in chapter 4 is original scholarship and a contribution to knowledge in the field.

Chapter 2

Preliminaries

This chapter contains a brief overview of the syntax and semantics of the multimodal logics we will be dealing with, some proof theoretic preliminaries and notation, and several background results on relations and closures.

2.1 Syntax and semantics

Much of the material in this section is standard and included for the sake of completeness. For more background, history, and comprehensive bibliographies, see for instance Blackburn, de Rijke, and Venema [8] or Hughes and Cresswell [25].

In this thesis, we will focus on propositional modal languages with a finite number of unary modal operators. Given a number $n \in \{0, 1, 2, \ldots\}$, we define the language of propositional n-modal logic in the following way.

Definition 2.1.1 (Formulas)**.** A *formula* is an expression constructed from *propositional letters* p_1, p_2, ..., parentheses, and the *logical operators* $\neg$ ("negation"), $\wedge$ ("conjunction"), $\vee$ ("disjunction"), $\supset$ ("implication"), $\Box_1$ ("1-box"), $\Box_2$ ("2-box"), ..., $\Box_n$ ("n-box"), $\Diamond_1$ ("1-diamond"), $\Diamond_2$ ("2-diamond"), ..., and $\Diamond_n$ ("n-diamond"). Formulas are defined by the following rules.

1. Every propositional letter is a formula.

2. If A is a formula, then so are $\neg A$, $\Box_1 A$, $\Box_2 A$, ..., $\Box_n A$, $\Diamond_1 A$, $\Diamond_2 A$, ..., and $\Diamond_n A$.

3. If A and B are formulas, then so are $(A \wedge B)$, $(A \vee B)$, and $(A \supset B)$.

A formula is said to be *atomic* if it is a propositional letter. Otherwise, it is *nonatomic*.

Following a common convention, we will omit parentheses in formulas whenever possible, using operator precedence rules for disambiguation. In particular, we will let the unary operators $\neg$, $\Box_1$, $\Box_2$, ..., $\Box_n$, $\Diamond_1$, $\Diamond_2$, ..., and $\Diamond_n$ all have higher precedence than $\wedge$, which in turn has higher precedence than $\vee$, which in turn has higher precedence than $\supset$. All binary operators will be considered right-associative, meaning that $p_1 \supset p_2 \supset p_3$, for instance, represents the formula $(p_1 \supset (p_2 \supset p_3))$.

As seen in definition 2.1.1, n is the number of distinct pairs of identically labelled box ($\Box$) and diamond ($\Diamond$) operators in our language. The "degenerate" modal language with $n = 0$ has no modal operators; it is nothing more than the language of propositional logic. The language with $n = 1$ has a single pair of modal operators; it is sometimes called the language of *unimodal logic*, and the labels of its modal operators are generally omitted. For consistency, we will always show the labels of modal operators, but will use i, j, and k to denote arbitrary elements of $\{1, 2, \ldots, n\}$.

The box and diamond operators are sometimes called *necessity* and *possibility* operators, respectively. We prefer "box" and "diamond", since the interpretations of the modalities in applications are not always the alethic ones of necessity and possibility, as we saw in the previous chapter. Instead of $\Box$ and $\Diamond$, some authors use L and M as modal operators [25]. In richer modal languages, such as those of dynamic logic [22] or boolean modal logic [8], where the labels of modal operators may be complex expressions rather than merely atomic identifiers, the labels are often placed inside the operators. To wit, $\Box_i$ and $\Diamond_i$ might be written as $[i]$ and $\langle i \rangle$,

respectively. Since the labels of our box and diamond operators are simple identifiers, we prefer subscript labels.

If modal logic is regarded as an extension of classical logic, then the two types of modal operators (box and diamond) are interdefinable in much the same way as conjunction and disjunction and, in first-order logics, universal and existential quantification. Consequently, some authors consider only modal operators of one type to be primitive and define the others as notational abbreviations. If box operators are taken to be primitive, for instance, then $\Diamond_i$ can be defined as the dual of $\Box_i$, i.e., as an abbreviation for $\neg\Box_i\neg$. When we discuss the semantics of multimodal logic later in this section, the dual nature of the box and diamond operators will become more evident. If modal logic is seen as an extension of intuitionistic logic, on the other hand, then the modal operators are not interdefinable in this way. We will briefly discuss intuitionistic modal logic in chapter 5, but until then, the foundations of our logics will be strictly classical. However, for the sake of perspicuity, we will still define both types of modal operators to be primitive.

To facilitate inductive proofs, we associate a *degree* with every formula. This is the number of instances of logical operators in the formula and is defined as follows.

Definition 2.1.2 (Degrees of formulas)**.** The *degree* of a formula A is a number $d \in \{0, 1, 2, \ldots\}$ determined by the following rules.

1. Every propositional letter has degree 0.
2. If A is a formula with degree d, then $\neg A$, $\Box_i A$, and $\Diamond_i A$ have degree $d + 1$.
3. If A and B are formulas with degrees d_1 and d_2, respectively, then $A \wedge B$, $A \vee B$, and $A \supset B$ have degree $d_1 + d_2 + 1$.

Our definitions of *subformulas* and related concepts are standard.

Definition 2.1.3 (Subformulas)**.** The set of *subformulas* of a formula A is the smallest set of formulas that contains A and has the following properties.

1. If $\neg B$, $\Box_i B$, or $\Diamond_i B$ is a subformula of A, then so is B.

2. If $B \wedge C$, $B \vee C$, or $B \supset C$ is a subformula of A, then so are B and C.

A subformula B of a formula A is a *proper subformula* of A if it is not identical to A. A proper subformula B of a formula A is an *immediate subformula* of A if there is no proper subformula C of A such that B is a proper subformula of C. Within a given formula, *subformula occurrences* are particular instances of subformulas. Note that a formula can have multiple occurrences of the same subformula. Subformulas of the form $\Box_i B$ or $\Diamond_i B$ are sometimes called *modal subformulas*.

Definition 2.1.4 (Positive and negative subformula occurrences)**.** A subformula occurrence B of a formula A is said to be a *positive subformula occurrence* of A or to *occur positively* in A if it is in the scope of an even number of negations in A. Otherwise, it is a *negative subformula occurence* of A or *occurs negatively* in A.

Note that being on the left side of an implication is also considered being in the scope of a negation, since a formula of the form $A \supset B$ is classically equivalent to $\neg A \vee B$.

Example 2.1.1 (Formulas and subformulas)**.** $A = \Box_1 p_1 \supset \Box_1 \Box_1 p_1 \wedge \Diamond_2 \Diamond_2 p_1 \supset \Diamond_2 p_1$ is a formula equivalent to $((\Box_1 p_1 \supset \Box_1 \Box_1 p_1) \wedge (\Diamond_2 \Diamond_2 p_1 \supset \Diamond_2 p_1))$. Its subformulas are p_1, $\Box_1 p_1$, $\Box_1 \Box_1 p_1$, $\Box_1 p_1 \supset \Box_1 \Box_1 p_1$, $\Diamond_2 p_1$, $\Diamond_2 \Diamond_2 p_1$, $\Diamond_2 \Diamond_2 p_1 \supset \Diamond_2 p_1$, and A itself. There are four occurrences of p_1, two of each of $\Box_1 p_1$ and $\Diamond_2 p_1$, and one of each remaining subformula of A. All subformula occurrences on the left sides of the two implications are negative subformula occurrences of A, while those on the right sides are positive.

We are now ready to breathe some life and meaning into formulas. We do so, as usual, by means of a Kripke semantics based on *frames* and *models*, which we define as follows.

Definition 2.1.5 (Frames)**.** A *frame* is a pair (W, R), where W is a nonempty set, and R is a set of triples of the form (i, w, x), where $i \in \{1, 2, \ldots, n\}$ and $w, x \in W$. The elements of W are called *worlds*, and those of R are called *accessibility triples* on W. R itself is called an *accessibility relation* on W. If there is an $(i, w, x) \in R$, then we say that w can *i-see*, *i-reach*, or *i-access* x, or that x is an *i-successor* of w.

The elements that we call worlds are sometimes called *possible worlds* [11], *points* [48], *states* [8], or *nodes* [8].

Definition 2.1.6 (Models)**.** A *model* is a triple (W, R, V), where (W, R) is a frame, and V is a function from propositional letters to subsets of W. V is called the *valuation* of the model. If $M = (W, R, V)$ and $F = (W, R)$, then M is said to be *based on* F, and F is called the *underlying frame* of M.

Our presentation of accessibility relations is slightly nonstandard. Usually, a multimodal frame is defined as a pair (W, R), where W is a nonempty set of worlds, and R is either a tuple $(R_1, R_2, \ldots, R_n)$ or set $\{R_1, R_2, \ldots, R_n\}$ of binary relations on W. However, it is easy to translate between the various presentations. For instance, if $(R_1, R_2, \ldots, R_n)$ is a tuple of binary relations on W, then $\bigcup_{i=1}^{n} \{(i, u, v) : (u, v) \in R_i\}$ is the corresponding set of accessibility triples on W. Conversely, if R is a set of accessibility triples on W, then every component of the corresponding tuple $(R_1, R_2, \ldots, R_n)$ of binary relations on W can be extracted as $R_i = \{(u, v) : (i, u, v) \in R\}$.

The standard presentations may lend themselves to more efficient implementations of data structures for multimodal accessibility relations, since the binary relations corresponding to different modalities are stored separately, which is not the case in our "mixed bag" approach of accessibility triples. However, our proof systems make extensive use of closures, which are discussed in section 2.3, and in this context, sets of accessibility triples generate slightly less cumbersome notation.

Definition 2.1.7 (Truth)**.** The *truth relation* $\Vdash$ is a ternary relation on models $M = (W, R, V)$, worlds $w \in W$, and formulas defined by the following rules.

1. For every propositional letter p, $M, w \Vdash p$ if and only if $w \in V(p)$.

2. $M, w \Vdash \neg A$ if and only if $M, w \nVdash A$ (i.e., it is not the case that $M, w \Vdash A$).

3. $M, w \Vdash A \wedge B$ if and only if $M, w \Vdash A$ and $M, w \Vdash B$.

4. $M, w \Vdash A \vee B$ if and only if $M, w \Vdash A$ or $M, w \Vdash B$.

5. $M, w \Vdash A \supset B$ if and only if $M, w \Vdash A$ implies that $M, w \Vdash B$ (i.e., $M, w \nVdash A$ or $M, w \Vdash B$).

6. $M, w \Vdash \Box_i A$ if and only if for every $x \in W$ such that $(i, w, x) \in R$, $M, x \Vdash A$.

7. $M, w \Vdash \Diamond_i A$ if and only if there is an $x \in W$ such that $(i, w, x) \in R$ and $M, x \Vdash A$.

If $M, w \Vdash A$, then A is said to be *true* at w in M. Otherwise, A is *false* at w in M. If $M, w \Vdash A$ for every $w \in W$, then A is said to be *globally true* in M.

The judgement $M, w \Vdash A$ is sometimes read as A being *satisfied* [8] or *forced* [48] at w in M. The truth relation formalizes the intuitive notion, mentioned in the previous chapter, of formulas being locally true at particular worlds in a collection of possible worlds (namely the worlds of the underlying frame).

Without further restrictions, frames and models typically encode domains that are more general than necessary. In other words, if our aim is to use a modal language to encode information in and reason about some domain, then it is often the case that the accessibility relations amongst the worlds, states, or nodes in it are systematically subject to some common restrictions or *frame conditions*, which we define as follows.

Definition 2.1.8 (Frame conditions)**.** An accessibility relation R on a set W is said to be

1. *i-reflexive* if for every $w \in W$, $(i, w, w) \in R$.

2. i, j-*symmetric* if $(i, w, x) \in R$ implies that $(j, x, w) \in R$.

3. i, j, k-*transitive* if $(i, w, x) \in R$ and $(j, x, y) \in R$ together imply that $(k, w, y) \in R$, and

4. i, j, k-*euclidean* if $(i, w, x) \in R$ and $(j, w, y) \in R$ together imply that $(k, x, y) \in R$.

As a shorthand, R is said to be i-*symmetric*, i-*transitive*, or i-*euclidean* if it is i, i-symmetric, i, i, i-transitive, or i, i, i-euclidean, respectively.

The frame conditions defined above are not the only interesting ones, of course, but they are amongst the most fundamental and common. We will discuss other frame conditions in chapter 5.

For a fixed $n \in \{0, 1, 2, \ldots\}$, different modal logics are characterized by different collections of frame conditions. If **L** is a name denoting a collection of frame conditions, then a frame (W, R) is said to be an **L**-*frame* if R satisfies **L**'s frame conditions. Similarly, a model is said to be an **L**-*model* if its underlying frame is an **L**-frame.

For instance, if $n = 1$, then a logic might be characterized by the frame conditions of 1-reflexivity and 1-transitivity. These are the frame conditions of the well-known unimodal logic **S4** (see for instance Hughes and Cresswell [25]). Consequently, a frame (W, R) is an **S4**-frame if and only if R is 1-reflexive and 1-transitive. Note that since $n = 1$, the first component of every accessibility triple in R can only be 1. If $n = 2$, then a logic might be characterized by $1, 2$-symmetry and 2-transitivity, for instance. We will use **L** to denote arbitrary logics, i.e., arbitrary collections of frame conditions.

As n increases, the number of possible ways in which the n binary relations represented by an accessibility relation can interact grows rapidly, and it becomes infeasible to study each multimodal logic separately. Unimodal logics, however, are easier to classify and name, since their frame conditions can only refer to a single modality. Table 2.1 assigns names to some standard unimodal logics and gives their frame con-

L	**L's frame conditions**	**Other common names for L**
K	none	
KT	1-reflexivity	**T**
KB	1-symmetry	
K4	1-transitivity	
K5	1-euclideanness	
KTB	1-reflexivity, 1-symmetry	**B**
KT4	1-reflexivity, 1-transitivity	**S4**
KB4	1-symmetry, 1-transitivity	
K45	1-transitivity, 1-euclideanness	
KTB4	1-reflexivity, 1-symmetry, 1-transitivity	**S5**, **KT5**

Table 2.1: Some standard unimodal logics and their frame conditions.

ditions. These are, in fact, the only 10 distinct unimodal logics that can be obtained from our frame conditions. They are a subset of what are sometimes called the 15 basic modal logics (see for instance Chellas [11] or Lemmon and Scott [32]).

Applications generally suggest or dictate suitable numbers of modalities for the syntax, as well as which frame conditions are relevant for the semantics (though it is by no means always obvious which frame conditions correctly model intuition and reasoning in particular domains). With these parameters fixed, we are ready to attempt to answer questions about the domain. Most proof systems, ours included, attempt to answer questions regarding the *satisfiability* or *validity* of given formulas, sometimes called *query formulas*.

Definition 2.1.9 (**L**-satisfiability, -falsifiability, and -validity)**.** A formula A is said to be *satisfied* by a frame F if A is true at some world in some model based on F. A is said to be **L**-*satisfiable* if it is satisfied by some **L**-frame.

A formula A is said to be *falsified* or *refuted* by a frame F if A is false at some world in some model based on F. A is said to be **L**-*falsifiable* if it is falsified by some **L**-frame.

A formula A is said to be *valid* in a frame F if A is globally true in every model

based on F. A is said to be **L**-*valid* if it is valid in every **L**-frame.

Notice that if a formula is **L**-falsifiable, then it is not **L**-valid. Contrapositively, if a formula is **L**-valid, then it is not **L**-falsifiable.

Given a logic **L**, our proof systems will characterize, in several different ways, when a query formula is **L**-valid. Other questions that many proof systems deal with are ones of *logical consequence*, i.e., when does a formula "follow from" a set of assumptions? Fitting and Mendelsohn [19] discuss some of the technicalities that arise when considering logical consequence in a modal setting, but the systems we will present in chapters 3 and 4 will focus solely on answering questions of validity.

2.2 Formula trees

In this section, we will define some of the notation and proof theoretic mechanisms that our formalisms will use. Our proof systems, like most other tableau and matrix characterizations, are based on *refutation*: given a logic **L** and a query formula A, the **L**-validity of A is established from a failed attempt to find evidence of a countermodel for A. How this countermodel construction proceeds depends naturally on the structure of the query formula. The proof systems we will introduce in the following chapters will make extensive use of representations of formulas that reveal their internal structure, all based, in one way or another, on the following notion of *formula trees*.

Definition 2.2.1 (Formula trees)**.** The *formula tree* of a formula A is an ordered tree whose nodes are in one-to-one correspondence with the subformula occurrences of A. Its structure, as well as the correspondence between its nodes and the subformula occurrences of A, is defined by the following rules.

1. The root corresponds to the whole of A.

2. If a node corresponds to an atomic subformula occurrence, then it has no children.

3. If a node corresponds to a subformula occurrence of the form $\neg B$, $\Box_i B$, or $\Diamond_i B$, then it has one child, corresponding to the subformula occurrence B.

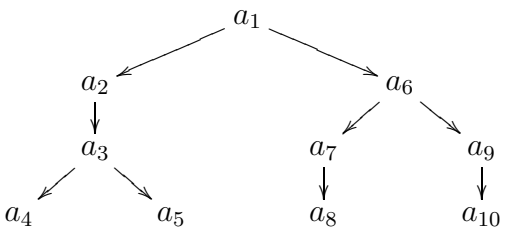
4. If a node corresponds to a subformula occurrence of the form $B \wedge C$, $B \vee C$, or $B \supset C$, then it has two children, corresponding to the subformula occurrences B and C, in that order.

We will use symbols from the alphabet $\{a_1, a_2, \ldots\}$ to name nodes of formula trees in the order in which they would be visited by a depth-first traversal of the tree, and we will speak of nodes and their names interchangeably where no ambiguity arises. A node is called *atomic* or *nonatomic* when the subformula occurrence it corresponds to is atomic or nonatomic, respectively. The *label* of a node a, denoted by $\mathrm{lab}(a)$, is the subformula of A that the node corresponds to. We will write $\ll$ to denote the transitive tree ordering of the formula tree on its nodes.

Example 2.2.1 (Formula trees)**.** The formula tree and label function of the formula $\Box_1(p_1 \supset p_2) \supset \Box_2 p_1 \supset \Box_3 p_2$ are shown in figure 2.1. In the formula tree, $a_1 \ll a_4$, $a_6 \ll a_8$, and $a_1 \ll a_6 \ll a_9 \ll a_{10}$, for instance. On the other hand, $a_3 \not\ll a_1$, since a_3 is a descendent of a_1, $a_1 \not\ll a_1$, since $\ll$ is irreflexive, and $a_7 \not\ll a_9$ (and $a_9 \not\ll a_7$), since a_7 and a_9 are siblings.

As a shorthand, if A denotes a formula, then we will write A to also refer to its formula tree. When we write, for instance, "the nodes of A", we mean the nodes of the formula tree of A.

One purpose of formally defining the formula tree is to be able to refer to individual subformula occurrences of a formula and associate information with them. As part of this program of associating information with subformula occurrences and nodes, we introduce *signed formulas* as follows.



a	$\mathrm{lab}(a)$
a_1	$\Box_1(p_1 \supset p_2) \supset \Box_2 p_1 \supset \Box_3 p_2$
a_2	$\Box_1(p_1 \supset p_2)$
a_3	$p_1 \supset p_2$
a_4	p_1
a_5	p_2
a_6	$\Box_2 p_1 \supset \Box_3 p_2$
a_7	$\Box_2 p_1$
a_8	p_1
a_9	$\Box_3 p_2$
a_{10}	p_2

Figure 2.1: The formula tree and label function of $\Box_1(p_1 \supset p_2) \supset \Box_2 p_1 \supset \Box_3 p_2$ (example 2.2.1).

Definition 2.2.2 (Signed formulas)**.** A *signed formula* is a pair of the form $(A, 0)$ or $(A, 1)$, where A is a formula. If (A, s) is a signed formula, then s is called its *sign* and stands for either 0 or 1.

Just as formulas are associated with formula trees, so signed formulas are associated with *signed formula trees*. In such a signed formula tree, the sign of the signed formula being represented determines a mapping of signs to the signed formula tree's nodes, which we will describe shortly. Like the sign of a signed formula, the sign of a signed formula tree node determined in this way is either 0 or 1. These signs of signed formula tree nodes in turn induce further information associated with the nodes in the following way.

Definition 2.2.3 (Signed formula trees)**.** The *signed formula tree* of a signed formula (A, s) is an ordered tree whose nodes, structure, and labels are identical to those of

the formula tree of A. In addition to labels, however, some of the nodes of the signed formula tree of (A, s) are associated with further information, namely *signs*, *primary types*, and *secondary types*, defined as follows.

The *sign* of a node a, denoted by $\mathrm{sgn}(a)$, is s if the subformula occurrence that a corresponds to occurs positively in A, and $(s+1) \bmod 2$ if it occurs negatively in A. In other words, in the signed formula tree of the signed formula $(A, 0)$, the sign of a node a is 0 if the subformula occurrence that a corresponds to occurs positively in A, and 1 otherwise. On the other hand, in the signed formula tree of the signed formula $(A, 1)$, the sign of a node a is 1 if the subformula occurrence that a corresponds to occurs positively in A, and 0 otherwise.

The *primary type* of a nonatomic node a, denoted by $\mathrm{ptype}(a)$, is one of α, β, ν_i, or π_i and is determined by its sign and label in the following way.

1. If $(\mathrm{lab}(a), \mathrm{sgn}(a))$ has the form $(\neg B, 0)$, $(\neg B, 1)$, $(B \wedge C, 1)$, $(B \vee C, 0)$, or $(B \supset C, 0)$, then a has primary type α.

2. If $(\mathrm{lab}(a), \mathrm{sgn}(a))$ has the form $(B \wedge C, 0)$, $(B \vee C, 1)$, or $(B \supset C, 1)$, then a has primary type β.

3. If $(\mathrm{lab}(a), \mathrm{sgn}(a))$ has the form $(\Box_i B, 1)$ or $(\Diamond_i B, 0)$, then a has primary type ν_i.

4. If $(\mathrm{lab}(a), \mathrm{sgn}(a))$ has the form $(\Box_i B, 0)$ or $(\Diamond_i B, 1)$, then a has primary type π_i.

Atomic nodes do not have primary types.

The *secondary type* of a nonroot node a, denoted by $\mathrm{stype}(a)$, is one of α_1, α_2, β_1, β_2, $\nu_{i,1}$, or $\pi_{i,1}$ and is determined by the primary type of a's parent and by a's position with respect to its parent. Specifically, if a is the left child of a node of primary type α (resp. β), then its secondary type is α_1 (resp. β_1). If a is the right child of a node of primary type α (resp. β), then its secondary type is α_2 (resp. β_2). If a is the single

a	$\mathrm{lab}(a)$	$\mathrm{sgn}(a)$	$\mathrm{ptype}(a)$	$\mathrm{stype}(a)$
a_1	$\Box_1(p_1 \supset p_2) \supset \Box_2 p_1 \supset \Box_3 p_2$	0	α	
a_2	$\Box_1(p_1 \supset p_2)$	1	ν_1	α_1
a_3	$p_1 \supset p_2$	1	β	$\nu_{1,1}$
a_4	p_1	0		β_1
a_5	p_2	1		β_2
a_6	$\Box_2 p_1 \supset \Box_3 p_2$	0	α	α_2
a_7	$\Box_2 p_1$	1	ν_2	α_1
a_8	p_1	1		$\nu_{2,1}$
a_9	$\Box_3 p_2$	0	π_3	α_2
a_{10}	p_2	0		$\pi_{3,1}$

Figure 2.2: The label, sign, and type functions of $(\Box_1(p_1 \supset p_2) \supset \Box_2 p_1 \supset \Box_3 p_2, 0)$ (example 2.2.2).

child of a node of primary type ν_i (resp. π_i), then its secondary type is $\nu_{i,1}$ (resp. $\pi_{i,1}$). The root does not have a secondary type.

Note that in this definition, we have tacitly assumed that nodes of primary types α and β have two children each. This is always true for nodes of primary type β, but a node a can have primary type α and only one child. This happens when $(\mathrm{lab}(a), \mathrm{sgn}(a))$ has the form $(s, \neg B)$, for instance. In such a case, the secondary type of the single child of a can be considered to be either α_1 or α_2.

Example 2.2.2 (Signed formula trees)**.** The signed formula tree of $(\Box_1(p_1 \supset p_2) \supset \Box_2 p_1 \supset \Box_3 p_2, 0)$ has the same structure as shown in figure 2.1. Its label, sign, and type functions are shown in figure 2.2. Note that the label function is identical to the one shown in figure 2.1.

Nodes of (signed) formula trees that correspond to modal subformula occurrences will play an important role in the sequel, so we will give them special names. In particular, we will call nodes of a signed formula tree with secondary type $\nu_{i,1}$ *ν-nodes* and nodes with secondary type $\pi_{i,1}$ *π-nodes*. The root is a special case. Although it has no secondary type, we will, for technical reasons that will become clear in chapters 3 and 4, also consider it a π-node. We will also use terms such as *nearest*

π-ancestor to refer to a node's nearest ancestor that is the root or has secondary type $\pi_{i,1}$, and *ν-child* to refer to a node's child with secondary type $\nu_{i,1}$.

Also, as a shorthand, we will sometimes speak of the *modality* of a ν-node or a nonroot π-node. This refers to the label of the top-level box or diamond operator of the subformula occurrence that the parent of the modal node corresponds to. In other words, the modality of a node of secondary type $\nu_{i,1}$ or $\pi_{i,1}$ is i. In example 2.2.2, for instance, the modality of a_3 is 1, and that of a_{10} is 3.

As with formulas and formula trees, if X denotes a signed formula, then we will write X to also refer to its signed formula tree. "The π-nodes of X", for instance, refers to the set of nodes of the signed formula tree of X containing the root and every node with secondary type $\pi_{i,1}$.

The purpose of signs and types of signed formula tree nodes is, informally, to make explicit information on how the nodes "behave" or what information they provide. If a node a of a signed formula (A, s) has primary type α, for instance, then the signed formula $(\mathrm{lab}(a), \mathrm{sgn}(a))$ is *conjunctive* in character. This turn of phrase is best demonstrated by example.

Suppose $(\mathrm{lab}(a), \mathrm{sgn}(a))$ has the form $(B \vee C, 0)$. Suppose furthermore that this signed formula represents or encodes the falsehood of the formula $B \vee C$. Now suppose there is a model $M = (W, R, V)$ and a world $w \in W$ such that $M, w \nVdash B \vee C$. By definition 2.1.7, we can conclude that both $M, w \nVdash B$ and $M, w \nVdash C$. Now notice that since a corresponds to a subformula occurrence of the form $B \vee C$, it has two children, say b and c, which correspond to the subformula occurrences B and C, respectively. Moreover, since B and C occur positively in A if and only if $B \vee C$ does, their signs are the same as that of a. Consequently, $(\mathrm{lab}(b), \mathrm{sgn}(b)) = (B, 0)$ and $(\mathrm{lab}(c), \mathrm{sgn}(c)) = (C, 0)$. These signed formulas can be throught of as representing or encoding the falsehood of B and C, respectively, exactly what $M, w \nVdash B$ and $M, w \nVdash C$ assert.

To formalize this intuition a little further, let us say that a node with label B and sign 0 *holds* at a world w in a model M if $M, w \nVdash B$, and one with label B and sign 1 holds at w in M if $M, w \Vdash B$. We can now express the idea in the previous paragraph more succinctly: if a node a of primary type α and with children b and c holds at a world w, then b and c hold at w. Node a is conjunctive because the conclusion of this implication is a conjunction. A node of primary type β, on the other hand, is *disjunctive* in character: if a node a of primary type β and with children b and c holds at a world w, then either b or c holds at w. Note that no modal information was used or needed in either of these implications. We introduced the world w only because a formula needs somewhere to be true or false.

Primary types ν_i and π_i, meanwhile, encode behaviour that is conjunctive (trivially so, since nodes of these primary types have only one child each) but depends on modal information. If a node a of primary type ν_i and with child b holds at a world w, then b holds at any world i-accessible from w. If a node a of primary type π_i and with child b holds at a world w, then there is a world i-accessible from w at which b holds. Where does the model come from in which these worlds and accessibilities exist? It is exactly the putative countermodel constructed by a derivation in a proof formalism. These derivations will be the focus of chapters 3 and 4.

2.3 L-closures

Recall that an accessibility relation on a set W is a set of accessibility triples, i.e., triples of the form (i, w, x), where $i \in \{1, 2, \ldots, n\}$ and $w, x \in W$. Our proof systems will make extensive use of **L**-*closures* of accessibility relations, which we will define and discuss in this section. Conceptually, these closures are a generalization of the familiar reflexive, symmetric, and transitive closures of binary relations on sets, with frame conditions serving as closure properties. We have collected all basic results

on **L**-closures in this section, as they will be used in various places throughout the remainder of the thesis.

Definition 2.3.1 (**L**-closures)**.** The **L**-*closure* of an accessibility relation R on a set W, denoted by $\mathrm{cl}_{\mathbf{L}}(R)$, is the smallest accessibility relation on W containing R and satisfying **L**'s frame conditions. Formally, an accessibility relation Q on W is said to have the **L**-*closure property* of R if $R \subseteq Q$ and Q satisfies **L**'s frame conditions. The **L**-closure of R is then defined as follows.

$$\mathrm{cl}_{\mathbf{L}}(R) = \bigcap \{Q : Q \text{ is an accessibility relation on } W \text{ with the } \mathbf{L}\text{-closure property of } R\}$$

Since $\mathrm{cl}_{\mathbf{L}}(R)$ is defined as the intersection of accessibility relations, each of which contains R, $\mathrm{cl}_{\mathbf{L}}(R)$ itself contains R. It is also easy to show that $\mathrm{cl}_{\mathbf{L}}(R)$ satisfies **L**'s frame conditions. This confirms that $\mathrm{cl}_{\mathbf{L}}(R)$ itself has the **L**-closure property of R, making $\mathrm{cl}_{\mathbf{L}}(R)$ the accessibility relation on W with the **L**-closure property of R that all other such accessibility relations contain.

The above definition of **L**-closures is slightly cumbersome for inductive proofs and actually computing them, so we will now characterize **L**-closures in an alternative, more constructive way using the notion of **L**-*closure sequences*.

Definition 2.3.2 (**L**-closure sequences)**.** The **L**-*closure sequence* of an accessibility relation R on a set W is an infinite sequence $R_1, R_2, \ldots$ of accessibility relations on W, where $R_1 = R$, and for every $l \in \{2, 3, \ldots\}$, R_l is the smallest accessibility relation on W with the following properties.

1. If **L** is i-reflexive and $w \in W$, then $(i, w, w) \in R_l$.

2. If **L** is i, j-symmetric and there is an $l' \in \{1, 2, \ldots, l-1\}$ such that $(i, w, x) \in R_{l'}$, then $(j, x, w) \in R_l$.

3. If **L** is i, j, k-transitive and there are $l', l'' \in \{1, 2, \ldots, l-1\}$ such that $(i, w, x) \in R_{l'}$ and $(j, x, y) \in R_{l''}$, then $(k, w, y) \in R_l$.

4. If **L** is i, j, k-euclidean and there are $l', l'' \in \{1, 2, \ldots, l-1\}$ such that $(i, w, x) \in R_{l'}$ and $(j, w, y) \in R_{l''}$, then $(k, x, y) \in R_l$.

Definition 2.3.3 (Constructive **L**-closures)**.** The *constructive* **L**-*closure* of an accessibility relation R on a set W with **L**-closure sequence R_1, R_2, ..., denoted by $\mathrm{ccl}_{\mathbf{L}}(R)$, is defined as follows.

$$\mathrm{ccl}_{\mathbf{L}}(R) = \bigcup_{l=1}^{\infty} R_l$$

The following important result establishes the equivalence of **L**-closures and constructive **L**-closures and allows us to prove properties of **L**-closures via constructive **L**-closures.

Lemma 2.3.1 (**L**-closure equality)**.** *For any logic* **L** *and accessibility relation* R, $\mathrm{cl}_{\mathbf{L}}(R) = \mathrm{ccl}_{\mathbf{L}}(R)$.

Proof. Let R be an accessibility relation on a set W with **L**-closure sequence R_1, R_2, ... We first observe that for every $l \in \{1, 2, \ldots\}$, R_l is contained in $\mathrm{cl}_{\mathbf{L}}(R)$. To see this, assume otherwise: let R_l be the first accessibility relation in the **L**-closure sequence of R that is not contained in $\mathrm{cl}_{\mathbf{L}}(R)$. In particular, let $(i, w, x) \in R_l$ be an accessibility triple such that $(i, w, x) \notin \mathrm{cl}_{\mathbf{L}}(R)$. If $l = 1$, then $R_l = R$, so $(i, w, x) \in R_l$ means that $(i, w, x) \in R$, which implies that $(i, w, x) \in \mathrm{cl}_{\mathbf{L}}(R)$. This is a contradiction. So $l \in \{2, 3, \ldots\}$ and $(i, w, x) \in R_l$ because of one of rules 1–4 of definition 2.3.2. We proceed by case analysis on each of these rules.

1. If **L** is i-reflexive and $(i, w, x) \in R_l$ because of rule 1, then $w = x$, but $(i, w, x) \notin \mathrm{cl}_{\mathbf{L}}(R)$, contradicting $\mathrm{cl}_{\mathbf{L}}(R)$'s i-reflexivity.

2. If **L** is j, i-symmetric and $(i, w, x) \in R_l$ because of rule 2, then there is an $l' \in \{1, 2, \ldots, l-1\}$ such that $(j, x, w) \in R_{l'}$. Since $R_{l'}$ is contained in $\mathrm{cl}_{\mathbf{L}}(R)$,

$(j, x, w) \in \mathrm{cl}_{\mathbf{L}}(R)$, but $(i, w, x) \notin \mathrm{cl}_{\mathbf{L}}(R)$, contradicting $\mathrm{cl}_{\mathbf{L}}(R)$'s j, i-symmetry.

3. If $\mathbf{L}$ is k, j, i-transitive and $(i, w, x) \in R_l$ because of rule 3, then there are $l', l'' \in \{1, 2, \ldots, l-1\}$ such that $(k, w, y) \in R_{l'}$ and $(j, y, x) \in R_{l''}$. Since $R_{l'}$ and $R_{l''}$ are contained in $\mathrm{cl}_{\mathbf{L}}(R)$, $(k, w, y), (j, y, x) \in \mathrm{cl}_{\mathbf{L}}(R)$, but $(i, w, x) \notin \mathrm{cl}_{\mathbf{L}}(R)$, contradicting $\mathrm{cl}_{\mathbf{L}}(R)$'s k, j, i-transitivity.

4. If $\mathbf{L}$ is k, j, i-euclidean and $(i, w, x) \in R_l$ because of rule 4, then there are $l', l'' \in \{1, 2, \ldots, l-1\}$ such that $(k, y, w) \in R_{l'}$ and $(j, y, x) \in R_{l''}$. Since $R_{l'}$ and $R_{l''}$ are contained in $\mathrm{cl}_{\mathbf{L}}(R)$, $(k, y, w), (j, y, x) \in \mathrm{cl}_{\mathbf{L}}(R)$, but $(i, w, x) \notin \mathrm{cl}_{\mathbf{L}}(R)$, contradicting $\mathrm{cl}_{\mathbf{L}}(R)$'s k, j, i-euclideanness.

For every $l \in \{1, 2, \ldots\}$, $R_l \subseteq \mathrm{cl}_{\mathbf{L}}(R)$, so $\mathrm{ccl}_{\mathbf{L}}(R) \subseteq \mathrm{cl}_{\mathbf{L}}(R)$. Next, we show that $\mathrm{ccl}_{\mathbf{L}}(R)$ satisfies $\mathbf{L}$'s frame conditions. Since $R \subseteq \mathrm{ccl}_{\mathbf{L}}(R)$ (as R_1), this will show that $\mathrm{ccl}_{\mathbf{L}}(R)$ has the $\mathbf{L}$-closure property of R. Again, we refer to the rules of definition 2.3.2.

1. If $\mathbf{L}$ is i-reflexive and $w \in W$, then for every $l \in \{2, 3, \ldots\}$, by rule 1, $(i, w, w) \in R_l$, so $(i, w, w) \in \mathrm{ccl}_{\mathbf{L}}(R)$.

2. If $\mathbf{L}$ is i, j-symmetric and $(i, w, x) \in \mathrm{ccl}_{\mathbf{L}}(R)$, then there is an $l \in \{1, 2, \ldots\}$ such that $(i, w, x) \in R_l$. By rule 2, $(j, x, w) \in R_{l+1}$, so $(j, x, w) \in \mathrm{ccl}_{\mathbf{L}}(R)$.

3. If $\mathbf{L}$ is i, j, k-transitive and $(i, w, x), (j, x, y) \in \mathrm{ccl}_{\mathbf{L}}(R)$, then there are $l, l' \in \{1, 2, \ldots\}$ such that $(i, w, x) \in R_l$ and $(j, x, y) \in R_{l'}$. Suppose that $l \geq l'$ (the other case is analogous). By rule 3, $(k, w, y) \in R_{l+1}$, so $(k, w, y) \in \mathrm{ccl}_{\mathbf{L}}(R)$.

4. If $\mathbf{L}$ is i, j, k-euclidean and $(i, w, x), (j, w, y) \in \mathrm{ccl}_{\mathbf{L}}(R)$, then there are $l, l' \in \{1, 2, \ldots\}$ such that $(i, w, x) \in R_l$ and $(j, w, y) \in R_{l'}$. Suppose that $l \geq l'$ (the other case is analogous). By rule 4, $(k, x, y) \in R_{l+1}$, so $(k, x, y) \in \mathrm{ccl}_{\mathbf{L}}(R)$.

Since $\mathrm{cl}_{\mathbf{L}}(R)$ is contained in every accessibility relation with the $\mathbf{L}$-closure property of R, $\mathrm{cl}_{\mathbf{L}}(R) \subseteq \mathrm{ccl}_{\mathbf{L}}(R)$. We can conclude that $\mathrm{cl}_{\mathbf{L}}(R) = \mathrm{ccl}_{\mathbf{L}}(R)$. □

Notice that although constructive **L**-closures are defined as infinite unions, the **L**-closure sequence of an accessibility relation R on a set W repeats itself if W and the number of modalities are finite. In the context of this thesis, the number of modalities, n, is indeed finite, and we will only be constructing countermodels with finite underlying frames for our finite (propositional) query formulas. Consequently, we will be able to compute **L**-closures in finite time and will present a simple algorithm to do so at the end of this section.

The next few technical results are used throughout the thesis to prove the correctness of our various proof systems. It is perhaps best to skim them at first and refer back to them as needed.

Lemma 2.3.2 (**L**-closure sequence subsets)**.** *If (W, R) and (W', R') are frames, R_1, R_2, ... and R'_1, R'_2, ... are the* **L***-closure sequences of R and R', respectively, $W \subseteq W'$, and $R \subseteq R'$, then for every $l \in \{1, 2, \ldots\}$, $(i, w, x) \in R_l$ implies that $(i, w, x) \in R'_l$.*

Proof. By induction on l.

1. If $l = 1$, then $R_l = R$ and $R'_l = R'$, and $R \subseteq R'$ is given.

2. If $l \in \{2, 3, \ldots\}$, then every $(i, w, x) \in R_l$ because of one of rules 1–4 of definition 2.3.2. We proceed by case analysis on each of these rules.

   (a) If **L** is i-reflexive and $(i, w, x) \in R_l$ because of rule 1, then $w = x$, so by rule 1 and since $W \subseteq W'$, $(i, w, x) \in R'_l$.

   (b) If **L** is j, i-symmetric and $(i, w, x) \in R_l$ because of rule 2, then there is an $l' \in \{1, 2, \ldots, l-1\}$ such that $(j, x, w) \in R_{l'}$. By the induction hypothesis, $(j, x, w) \in R'_{l'}$, so by rule 2, $(i, w, x) \in R'_l$.

   (c) If **L** is k, j, i-transitive and $(i, w, x) \in R_l$ because of rule 3, then there are $l', l'' \in \{1, 2, \ldots, l-1\}$ such that $(k, w, y) \in R_{l'}$ and $(j, y, x) \in R_{l''}$. By the induction hypothesis, $(k, w, y) \in R'_{l'}$ and $(j, y, x) \in R'_{l''}$, so by rule 3, $(i, w, x) \in R'_l$.

(d) If $\mathbf{L}$ is k, j, i-euclidean and $(i, w, x) \in R_l$ because of rule 4, then there are $l', l'' \in \{1, 2, \ldots, l-1\}$ such that $(k, y, w) \in R_{l'}$ and $(j, y, x) \in R_{l''}$. By the induction hypothesis, $(k, y, w) \in R'_{l'}$ and $(j, y, x) \in R'_{l''}$, so by rule 4, $(i, w, x) \in R'_l$. □

Corollary 2.3.1 ($\mathbf{L}$-closure subsets)**.** *If (W, R) and (W', R') are frames, $W \subseteq W'$, and $R \subseteq R'$, then $(i, w, x) \in \mathrm{cl}_{\mathbf{L}}(R)$ implies that $(i, w, x) \in \mathrm{cl}_{\mathbf{L}}(R')$.*

Proof. Let R_1, R_2, ... and R'_1, R'_2, ... be the $\mathbf{L}$-closure sequences of R and R', respectively. By lemma 2.3.1, $(i, w, x) \in \mathrm{cl}_{\mathbf{L}}(R)$ implies that $(i, w, x) \in \mathrm{ccl}_{\mathbf{L}}(R)$, so there is an $l \in \{1, 2, \ldots\}$ such that $(i, w, x) \in R_l$. By lemma 2.3.2, this implies that $(i, w, x) \in R'_l$, so $(i, w, x) \in \mathrm{ccl}_{\mathbf{L}}(R')$, which by lemma 2.3.1 again, implies that $(i, w, x) \in \mathrm{cl}_{\mathbf{L}}(R')$. □

Lemma 2.3.3 (Faithfulness of $\mathbf{L}$-closure sequences)**.** *If (W, R) is a frame, R_1, R_2, ... is the $\mathbf{L}$-closure sequence of R, (W', R') is an $\mathbf{L}$-frame, and I is a function from W to W' such that $(i, w, x) \in R$ implies that $(i, I(w), I(x)) \in R'$, then for every $l \in \{1, 2, \ldots\}$, $(i, w, x) \in R_l$ implies that $(i, I(w), I(x)) \in R'$.*

Proof. By induction on l.

1. If $l = 1$, then $R_l = R$, so $(i, w, x) \in R_l$ means that $(i, w, x) \in R$. $(i, I(w), I(x)) \in R'$ then follows from the assumption on I.

2. If $l \in \{2, 3, \ldots\}$, then $(i, w, x) \in R_l$ because of one of rules 1–4 of definition 2.3.2. We proceed by case analysis on each of these rules.

   (a) If $\mathbf{L}$ is i-reflexive and $(i, w, x) \in R_l$ because of rule 1, then $w = x$, so $I(w) = I(x)$. Since R' is i-reflexive, $(i, I(w), I(x)) \in R'$.

   (b) If $\mathbf{L}$ is j, i-symmetric and $(i, w, x) \in R_l$ because of rule 2, then there is an $l' \in \{1, 2, \ldots, l-1\}$ such that $(j, x, w) \in R_{l'}$. By the induction hypothesis, $(j, I(x), I(w)) \in R'$, and since R' is j, i-symmetric, $(i, I(w), I(x)) \in R'$.

(c) If **L** is k, j, i-transitive and $(i, w, x) \in R_l$ because of rule 3, then there are $l', l'' \in \{1, 2, \ldots, l-1\}$ such that $(k, w, y) \in R_{l'}$ and $(j, y, x) \in R_{l''}$. By the induction hypothesis, $(k, I(w), I(y)) \in R'$ and $(j, I(y), I(x)) \in R'$, and since R' is k, j, i-transitive, $(i, I(w), I(x)) \in R'$.

(d) If **L** is k, j, i-euclidean and $(i, w, x) \in R_l$ because of rule 4, then there are $l', l'' \in \{1, 2, \ldots, l-1\}$ such that $(k, y, w) \in R_{l'}$ and $(j, y, x) \in R_{l''}$. By the induction hypothesis, $(k, I(y), I(w)) \in R'$ and $(j, I(y), I(x)) \in R'$, and since R' is k, j, i-euclidean, $(i, I(w), I(x)) \in R'$. □

Corollary 2.3.2 (Faithfulness of **L**-closures)**.** *If (W, R) is a frame, (W', R') is an* **L***-frame, and I is a function from W to W' such that $(i, w, x) \in R$ implies that $(i, I(w), I(x)) \in R'$, then $(i, w, x) \in \mathrm{cl}_{\mathbf{L}}(R)$ implies that $(i, I(w), I(x)) \in R'$.*

Proof. Let R_1, R_2, ... be the **L**-closure sequence of R. By lemma 2.3.1, $(i, w, x) \in \mathrm{cl}_{\mathbf{L}}(R)$ implies that $(i, w, x) \in \mathrm{ccl}_{\mathbf{L}}(R)$, so there is an $l \in \{1, 2, \ldots\}$ such that $(i, w, x) \in R_l$. By lemma 2.3.3, this implies that $(i, I(w), I(x)) \in R'$. □

Lemma 2.3.4 (**L**-closure sequences of accessibility relation unions)**.** *If (W_1, R_1), (W_2, R_2), ... is a sequence of frames such that $W_1 \subseteq W_2 \subseteq \cdots$ and $R_1 \subseteq R_2 \subseteq \cdots$, $W' = \bigcup_{m=1}^{\infty} W_m$, $R' = \bigcup_{m=1}^{\infty} R_m$, and R'_1, R'_2, ... is the* **L***-closure sequence of R', then for every $l \in \{1, 2, \ldots\}$, if $(i, w, x) \in R'_l$, then there is an $m \in \{1, 2, \ldots\}$ such that $(i, w, x) \in \mathrm{cl}_{\mathbf{L}}(R_m)$.*

Proof. By induction on l.

1. If $l = 1$, then $R'_l = R'$, so $(i, w, x) \in R'_l$ means that $(i, w, x) \in \bigcup_{m=1}^{\infty} R_m$. This means that there is an $m \in \{1, 2, \ldots\}$ such that $(i, w, x) \in R_m$, which implies that $(i, w, x) \in \mathrm{cl}_{\mathbf{L}}(R_m)$.

2. If $l \in \{2, 3, \ldots\}$, then $(i, w, x) \in R'_l$ because of one of rules 1–4 of definition 2.3.2. We proceed by case analysis on each of these rules.

(a) If $\mathbf{L}$ is i-reflexive and $(i, w, x) \in R'_l$ because of rule 1, then $w = x$, and $w \in \bigcup_{m=1}^{\infty} W_m$, so there is an $m \in \{1, 2, \ldots\}$ such that $w \in W_m$. Since $\mathrm{cl}_{\mathbf{L}}(R_m)$ is i-reflexive, $(i, w, x) \in \mathrm{cl}_{\mathbf{L}}(R_m)$.

(b) If $\mathbf{L}$ is j, i-symmetric and $(i, w, x) \in R'_l$ because of rule 2, then there is an $l' \in \{1, 2, \ldots, l-1\}$ such that $(j, x, w) \in R'_{l'}$. By the induction hypothesis, this implies that there is an $m \in \{1, 2, \ldots\}$ such that $(j, x, w) \in \mathrm{cl}_{\mathbf{L}}(R_m)$, and since $\mathrm{cl}_{\mathbf{L}}(R_m)$ is j, i-symmetric, $(i, w, x) \in \mathrm{cl}_{\mathbf{L}}(R_m)$.

(c) If $\mathbf{L}$ is k, j, i-transitive and $(i, w, x) \in R'_l$ because of rule 3, then there are $l', l'' \in \{1, 2, \ldots, l-1\}$ such that $(k, w, y) \in R'_{l'}$ and $(j, y, x) \in R'_{l''}$. By the induction hypothesis, this implies that there are $m, m' \in \{1, 2, \ldots\}$ such that $(k, w, y) \in \mathrm{cl}_{\mathbf{L}}(R_m)$ and $(j, y, x) \in \mathrm{cl}_{\mathbf{L}}(R_{m'})$. If $m \geq m'$ (the other case is analogous), then $W_{m'} \subseteq W_m$ and $R_{m'} \subseteq R_m$, so by corollary 2.3.1, $\mathrm{cl}_{\mathbf{L}}(R_{m'}) \subseteq \mathrm{cl}_{\mathbf{L}}(R_m)$. This means that $(k, w, y), (j, y, x) \in \mathrm{cl}_{\mathbf{L}}(R_m)$, and since $\mathrm{cl}_{\mathbf{L}}(R_m)$ is k, j, i-transitive, $(i, w, x) \in \mathrm{cl}_{\mathbf{L}}(R_m)$.

(d) If $\mathbf{L}$ is k, j, i-euclidean and $(i, w, x) \in R'_l$ because of rule 4, then there are $l', l'' \in \{1, 2, \ldots, l-1\}$ such that $(k, y, w) \in R'_{l'}$ and $(j, y, x) \in R'_{l''}$. By the induction hypothesis, this implies that there are $m, m' \in \{1, 2, \ldots\}$ such that $(k, y, w) \in \mathrm{cl}_{\mathbf{L}}(R_m)$ and $(j, y, x) \in \mathrm{cl}_{\mathbf{L}}(R_{m'})$. If $m \geq m'$ (the other case is analogous), then $W_{m'} \subseteq W_m$ and $R_{m'} \subseteq R_m$, so by corollary 2.3.1, $\mathrm{cl}_{\mathbf{L}}(R_{m'}) \subseteq \mathrm{cl}_{\mathbf{L}}(R_m)$. This means that $(k, y, w), (j, y, x) \in \mathrm{cl}_{\mathbf{L}}(R_m)$, and since $\mathrm{cl}_{\mathbf{L}}(R_m)$ is k, j, i-euclidean, $(i, w, x) \in \mathrm{cl}_{\mathbf{L}}(R_m)$. □

Corollary 2.3.3 ($\mathbf{L}$-closures of accessibility relation unions)**.** *If* (W_1, R_1), (W_2, R_2), *...is a sequence of frames such that* $W_1 \subseteq W_2 \subseteq \cdots$ *and* $R_1 \subseteq R_2 \subseteq \cdots$, $W' = \bigcup_{m=1}^{\infty} W_m$, *and* $R' = \bigcup_{m=1}^{\infty} R_m$, *then if* $(i, w, x) \in \mathrm{cl}_{\mathbf{L}}(R')$, *then there is an* $m \in \{1, 2, \ldots\}$ *such that* $(i, w, x) \in \mathrm{cl}_{\mathbf{L}}(R_m)$.

Proof. Let R'_1, R'_2, ...be the $\mathbf{L}$-closure sequence of R'. By lemma 2.3.1, $(i, w, x) \in$

$\mathrm{cl}_{\mathbf{L}}(R')$ implies that $(i, w, x) \in \mathrm{ccl}_{\mathbf{L}}(R')$, which means that there is an $l \in \{1, 2, \ldots\}$ such that $(i, w, x) \in R'_l$. By lemma 2.3.4, this implies that there is an $m \in \{1, 2, \ldots\}$ such that $(i, w, x) \in \mathrm{cl}_{\mathbf{L}}(R_m)$. □

In addition to facilitating the inductive proofs of various results concerning **L**-closures, constructive **L**-closures suggest a means to compute **L**-closures of accessibility relations on finite sets. Since all the accessibility relations that our proof systems need to manipulate are finite, this is sufficient. Given an accessibility relation R on a finite set W, the following procedure computes the constructive **L**-closure of R.

1 $Q \leftarrow R$

2 $Q' \leftarrow \varnothing$

3 **do**

4 $\quad Q \leftarrow Q \cup Q'$

5 $\quad Q' \leftarrow \varnothing$

6 $\quad$ **for every** frame condition c of **L do**

7 $\quad\quad$ **if** $c = i$-reflexivity **then**

8 $\quad\quad\quad$ **for every** $w \in W$ **do**

9 $\quad\quad\quad\quad Q' \leftarrow Q' \cup \{(i, w, w)\}$

10 $\quad\quad$ **else if** $c = i, j$-symmetry **then**

11 $\quad\quad\quad$ **for every** $(i, w, x) \in Q$ **do**

12 $\quad\quad\quad\quad Q' \leftarrow Q' \cup \{(j, x, w)\}$

13 $\quad\quad$ **else if** $c = i, j, k$-transitivity **then**

14 $\quad\quad\quad$ **for every** $(i, w, x), (j, x, y) \in Q$ **do**

15 $\quad\quad\quad\quad Q' \leftarrow Q' \cup \{(k, w, y)\}$

16 $\quad\quad$ **else if** $c = i, j, k$-euclideanness **then**

17 **for every** $(i, w, x), (j, w, y) \in Q$ **do**

18 $Q' \leftarrow Q' \cup \{(k, x, y)\}$

19 **while** $Q' \not\subseteq Q$

20 **return** Q

Notice that if R_1, R_2, ... is the **L**-closure sequence of R, then for every $m \in \{1, 2, \ldots\}$, at the end of the mth iteration of the **do** ... **while** loop, $Q = \bigcup_{l=1}^{m} R_l$ and $Q' = R_{m+1}$. This can be shown by straightforward induction on m. The procedure terminates when $Q' \subseteq Q$, or in other words, after the first iteration m for which $Q = \bigcup_{l=1}^{m} R_l = \bigcup_{l=1}^{m+1} R_l$. If W is finite, then so is every accessibility relation on W (since we also have finitely many modalities). Consequently, the procedure must terminate, as it would otherwise describe an arbitrarily large accessibility relation on W.

To prove the correctness of our procedure, we must show that if it terminates after the mth iteration, then $Q = \bigcup_{l=1}^{m} R_l = \bigcup_{l=1}^{\infty} R_l$. To do this, it suffices to show that if $\bigcup_{l=1}^{m} R_l = \bigcup_{l=1}^{m+1} R_l$, then for any $m' \in \{1, 2, \ldots\}$, $\bigcup_{l=1}^{m} R_l = \bigcup_{l=1}^{m+m'} R_l$. Expressed differently, this is equivalent to showing that if $R_{m+1} \subseteq \bigcup_{l=1}^{m} R_l$, then for any $m' \in \{1, 2, \ldots\}$, $R_{m+m'} \subseteq \bigcup_{l=1}^{m} R_l$, which we prove as follows.

Lemma 2.3.5. *If $m \in \{1, 2, \ldots\}$ and $R_{m+1} \subseteq \bigcup_{l=1}^{m} R_l$, then for any $m' \in \{1, 2, \ldots\}$, $R_{m+m'} \subseteq \bigcup_{l=1}^{m} R_l$.*

Proof. By induction on m'.

1. If $m' = 1$, then $R_{m+m'} = R_{m+1}$, and $R_{m+1} \subseteq \bigcup_{l=1}^{m} R_l$ is given.

2. If $m' \in \{2, 3, \ldots\}$, then every $(i, w, x) \in R_{m+m'}$ because of one of rules 1–4 of definition 2.3.2. We proceed by case analysis on each of these rules.

   (a) If **L** is i-reflexive and $(i, w, x) \in R_{m+m'}$ because of rule 1, then $w = x$, so for every $l' \in \{2, 3, \ldots\}$, $(i, w, x) \in R_{l'}$. $m + 1 \in \{2, 3, \ldots\}$, so by rule 1, $(i, w, x) \in R_{m+1}$, and since $R_{m+1} \subseteq \bigcup_{l=1}^{m} R_l$, $(i, w, x) \in \bigcup_{l=1}^{m} R_l$.

(b) If $\mathbf{L}$ is j,i-symmetric and $(i,w,x) \in R_{m+m'}$ because of rule 2, then there is an $l' \in \{1,2,\ldots,m+m'-1\}$ such that $(j,x,w) \in R_{l'}$. Either $l' \in \{1,2,\ldots,m\}$ or $l' \in \{m+1,m+2,\ldots,m+m'-1\}$, but in either case, $(j,x,w) \in \bigcup_{l=1}^{m} R_l$ (using the induction hypothesis in the latter case). By rule 2, $(i,w,x) \in R_{m+1}$, and since $R_{m+1} \subseteq \bigcup_{l=1}^{m} R_l$, $(i,w,x) \in \bigcup_{l=1}^{m} R_l$.

(c) If $\mathbf{L}$ is k,j,i-transitive and $(i,w,x) \in R_{m+m'}$ because of rule 3, then there are $l', l'' \in \{1,2,\ldots,m+m'-1\}$ such that $(k,w,y) \in R_{l'}$ and $(j,y,x) \in R_{l''}$. Either $l' \in \{1,2,\ldots,m\}$ or $l' \in \{m+1,m+2,\ldots,m+m'-1\}$, but in either case, $(k,w,y) \in \bigcup_{l=1}^{m} R_l$ (using the induction hypothesis in the latter case). Similarly, $(j,y,x) \in \bigcup_{l=1}^{m} R_l$. By rule 3, $(i,w,x) \in R_{m+1}$, and since $R_{m+1} \subseteq \bigcup_{l=1}^{m} R_l$, $(i,w,x) \in \bigcup_{l=1}^{m} R_l$.

(d) If $\mathbf{L}$ is k,j,i-euclidean and $(i,w,x) \in R_{m+m'}$ because of rule 4, then there are $l', l'' \in \{1,2,\ldots,m+m'-1\}$ such that $(k,y,w) \in R_{l'}$ and $(j,y,x) \in R_{l''}$. Either $l' \in \{1,2,\ldots,m\}$ or $l' \in \{m+1,m+2,\ldots,m+m'-1\}$, but in either case, $(k,y,w) \in \bigcup_{l=1}^{m} R_l$ (using the induction hypothesis in the latter case). Similarly, $(j,y,x) \in \bigcup_{l=1}^{m} R_l$. By rule 4, $(i,w,x) \in R_{m+1}$, and since $R_{m+1} \subseteq \bigcup_{l=1}^{m} R_l$, $(i,w,x) \in \bigcup_{l=1}^{m} R_l$. □

Although we have not designed our proof systems with high-performance proof search in mind (something we will briefly discuss in chapter 5), it is nevertheless satisfying if proof search is feasible. In this light, the ability to compute $\mathbf{L}$-closures of accessibility relations on finite sets is important, even if the procedure shown above is naive and, depending on the implementation, not particularly efficient.

Chapter 3

Tableau characterization

The family of proof systems called *analytic tableaus* has its roots in the sequent calculus of Gentzen [20], but was developed into a distinct class of formalisms by Beth [5] and Hintikka [23] in the 1950s and refined by Smullyan [44] in the 1960s. Various adaptations of tableau systems to modal logics were subsequently proposed and studied by, amongst others, Kripke [30], Zeman [50], Kanger [26], and Fitting [16, 17], who developed the popular notion of *prefixed tableaus* for modal logics. Together with sequent calculi and systems of natural deduction [42], tableau formalisms are amongst the most intuitive and user-friendly proof systems, so it is no surprise that all manner of tableau systems have been devised for modal logics.

Most tableau systems for modal logics differ in presentation and details, but share the underlying idea of attempting to refute a query formula. More concretely, for a given logic **L**, they try to find a countermodel for the query formula, i.e., an **L**-model in which the formula is false. If this attempt leads to contradictions, then the query formula is not **L**-falsifiable. In other words, there is no **L**-model in which the formula is false, meaning that it is **L**-valid. If the refutation attempt succeeds, then the tableau proof exhibits a concrete **L**-model and world in it at which the query formula is false.

Tableau systems for modal logics generally treat the classical part of the language

in the same way, but differ in how modal information is represented and used. In general, these systems for modal logics can be classified according to whether they manipulate accessibility relations explicitly or implicitly. Goré [21] provides an accessible survey of various methods of this kind, concentrating on modal tableau systems with implicit accessibility relations. Systems with explicit accessibility relations, inspired by Fitting's prefixed tableaus, are touched on only briefly. The tableau system we will describe in this chapter differs markedly from Fitting's popular modal tableau systems in that it does *not* use prefixes to encode modal accessibility information. Instead, we will encode and manipulate accessibility relations directly. This approach is closest in spirit to Nerode's [39] tableau formalism, which also inspired Baldoni's [4] calculus. By doing so, our system (along with those of Baldoni and Nerode) is closer to the relational semantics of the logics being treated than prefixed or labelled systems like Fitting's and Massacci's [35], which are more syntactic in nature.

Unlike the systems of Baldoni and Nerode, however, ours represents worlds in the putative countermodel using a special notation based on the nodes of the formula tree. The motivation for such a representation is to facilitate the adaptation of our tableau proofs to proofs in our new path characterization. We do this in order to illuminate the similarities between the two systems and to prove the correctness of the path characterization. Consequently, some features of our tableau system might appear unnecessary or redundant, and the presentation could indeed be streamlined and simplified. Rather than having to return to and augment tableaus for the purpose of proving properties of the path characterization, however, we have chosen to design the tableau system from the start with the path characterization in mind.

It should also be mentioned that our presentation of tableaus is similar to what Smullyan calls *block tableaus* [44], in which nodes of tableaus are not single formulas or pieces of information, but collections of them. In our case, tableau nodes incorporate representations of entire partial putative countermodels. In this light, our tableau

calculus could just as well be called a sequent calculus.

3.1 Definitions

We will call the nodes of our tableaus *frame images*. Defined as follows, they are representations of partial potential countermodels for the query formula, an interpretation we will explore in more detail shortly.

Definition 3.1.1 (Frame images)**.** A *frame image* of a signed formula X is a triple (U, Q, S), where U is a set of expressions of the form a^ι, where a is the name of a π-node of X, and ι is a sequence of positive integers, Q is an accessibility relation on U, and S is a set of pairs of the form (u, a^ι), where $u \in U$, a is the name of a node of X, and ι is a sequence of positive integers.

A frame image (U, Q, S) is said to be *closed* if S contains pairs (u, a^ι) and (u, b^κ) such that $\mathrm{lab}(a) = \mathrm{lab}(b)$ and $\mathrm{sgn}(a) \neq \mathrm{sgn}(b)$. If $\mathrm{lab}(a)$ and $\mathrm{lab}(b)$ are atomic, then (U, Q, S) is said to be *atomically closed*. Otherwise, it is *(atomically) open*.

Where no ambiguity arises, we will write a sequence $l_1, l_2, \ldots, l_m$ of positive integers in concatenated form: "$l_1 l_2 \cdots l_m$". We will denote the empty sequence by a single dot, "$\cdot$". If ι is a sequence of positive integers and l is a positive integer, then ι, l denotes the sequence obtained by extending ι by l. Again, we will usually write this as simply "ιl".

The intuition behind a frame image (U, Q, S) is that it represents information about a potential countermodel. The elements of U name worlds in the underlying frame, Q is the accessibility relation on them, and S provides information about which formulas are true and false at which worlds. The formulas and their respective truth or falsehood are encoded by the labels and signs of the node names, while the names of the worlds at which they are true or false are carried with them explicitly. For instance, a pair (u, a^ι) contained in S encodes the truth—if $\mathrm{sgn}(a) = 1$—or

falsehood—if $\mathrm{sgn}(a) = 0$—of the subformula $\mathrm{lab}(a)$ at the world named by u. Notice that the names of the worlds are all names of π-nodes of the query formula. This is deliberate; it is one way in which we prepare for the adaptation of the tableau characterization to the path characterization in the next chapter.

Another way in which we prepare for this translation is the presence of the sequences of positive integers we associate with node names in the tableau characterization. They are not strictly necessary for the correctness of the tableau calculus, but they will correspond exactly to bookkeeping devices called *indices*, which will play an important role in the path characterization. As a hint of things to come, indices are a way of keeping track of different concrete worlds at which modal subformula occurrences of the query formula are assumed to hold in the countermodel being constructed. Every time a modal subformula occurrence is instantiated at a world in a tableau derivation, a new sequence of integers is generated for it, making it possible to distinguish multiple instantiations of the same subformula occurrence.

Recall that rather than manipulating subformula occurrences, however, we deal only with nodes of the query formula. Furthermore, in the case of tableaus, the query formula is, in fact, a query *signed* formula. If we are interested in the **L**-validity of a formula A, then we will attempt to construct an **L**-tableau for the signed formula $(A, 0)$. Recall from the previous chapter that signed formulas can be thought of as encoding the truth or falsehood of their component formulas, so the reason we construct an **L**-tableau for $(A, 0)$ is that we are attempting to refute A, or build a model in which it is false. The construction of tableaus themselves proceeds according to the following definition.

Definition 3.1.2 (**L**-tableaus)**.** An **L**-*tableau* for a signed formula X is an ordered tree of frame images of X. Every **L**-tableau for X is constructed from the root $(\{a_1^{\cdot}\}, \varnothing, \{(a_1^{\cdot}, a_1^{\cdot})\})$ using the following **L**-*tableau rules*.

1. If (U, Q, S) is the leaf of a branch of an **L**-tableau, S contains a pair (u, a^{ι}),

where a has primary type α and left child b, and $(u, b^\iota) \notin S$, then the (α_1) *rule* may be applied to extend the branch by $(U, Q, S \cup \{(u, b^\iota)\})$. This is called α_1-*decomposing* (u, a^ι) to (u, b^ι) and can be represented schematically as follows.

$$\frac{(U, Q, S \cup \{(u, \alpha_1^\iota)\})}{(U, Q, S)}\ (\alpha_1) \quad ((u, \alpha^\iota) \in S)$$

Here, α denotes an arbitrary node of primary type α, and α_1 denotes its left child. We will use a similar notation when presenting schematic forms of the remaining rules.

2. If (U, Q, S) is the leaf of a branch of an **L**-tableau, S contains a pair $(u, a^\iota) \in S$, where a has primary type α and right child c, and $(u, c^\iota) \notin S$, then the (α_2) *rule* may be applied to extend the branch by $(U, Q, S \cup \{(u, c^\iota)\})$. This is called α_2-*decomposing* (u, a^ι) to (u, c^ι) and can be represented schematically as follows.

$$\frac{(U, Q, S \cup \{(u, \alpha_2^\iota)\})}{(U, Q, S)}\ (\alpha_2) \quad ((u, \alpha^\iota) \in S)$$

Notice that if a has only one child, then the rules (α_1) and (α_2) are effectively the same. As in the previous chapter, we will systematically assume that nodes of primary type α have two children each. When they have only one, the two separate cases—one for each child—generally collapse into one.

3. If (U, Q, S) is the leaf of a branch of an **L**-tableau, S contains a pair (u, a^ι), where a has primary type β and children b and c, $(u, b^\iota) \notin S$, and $(u, c^\iota) \notin S$, then the (β) *rule* may be applied to fork and extend the branch by $(U, Q, S \cup \{(u, b^\iota)\})$ and $(U, Q, S \cup \{(u, c^\iota)\})$. This is called β-*decomposing* (u, a^ι) to (u, b^ι) and (u, c^ι) and can be represented schematically as follows.

$$\frac{(U, Q, S \cup \{(u, \beta_1^\iota)\}) \quad (U, Q, S \cup \{(u, \beta_2^\iota)\})}{(U, Q, S)}\ (\beta) \quad ((u, \beta^\iota) \in S)$$

4. If (U, Q, S) is the leaf of a branch of an **L**-tableau, S contains a pair (u, a^ι), where a has primary type ν_i and child b, and $(i, u, v) \in \mathrm{cl}_{\mathbf{L}}(Q)$, then the (ν) *rule* may be applied to extend the branch by $(U, Q, S \cup \{(v, b^{\iota l})\})$, where l is the smallest positive integer such that $b^{\iota l}$ does not yet occur in the tableau, meaning that the **L**-tableau does not contain a frame image (U', Q', S') where S' contains a pair of the form $(u', b^{\iota l})$. This is called *ν-decomposing* (u, a^ι) to $(v, b^{\iota l})$ and can be represented schematically as follows.

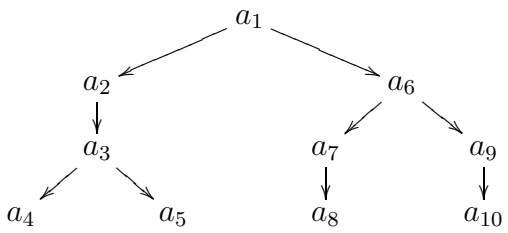
$$\frac{(U, Q, S \cup \{(v, \nu_{i,1}^{\iota l})\})}{(U, Q, S)}\ (\nu) \quad ((u, \nu_i^\iota) \in S \text{ and } (i, u, v) \in \mathrm{cl}_{\mathbf{L}}(Q))$$

5. If (U, Q, S) is the leaf of a branch of an **L**-tableau, S contains a pair (u, a^ι), where a has primary type π_i and child b, and $b^\iota \notin U$, then the (π) *rule* may be applied to extend the branch by $(U \cup \{b^\iota\}, Q \cup \{(i, u, b^\iota)\}, S \cup \{(b^\iota, b^\iota)\})$. This is called *$\pi$-decomposing* (u, a^ι) to (b^ι, b^ι) and can be represented schematically as follows.

$$\frac{(U \cup \{\pi_{i,1}^\iota\}, Q \cup \{(i, u, \pi_{i,1}^\iota)\}, S \cup \{(\pi_{i,1}^\iota, \pi_{i,1}^\iota)\})}{(U, Q, S)}\ (\pi) \quad ((u, \pi_i^\iota) \in S)$$

   Note that adding b^ι (or $\pi_{i,1}^\iota$ in the schematic form) to U is compatible with the definition of frame images, since b (or $\pi_{i,1}$, the child of π_i) has secondary type $\pi_{i,1}$ and is thus a π-node. This is also part of why we define the roots of query formulas to be π-nodes: the definition of frame images would otherwise have to include a separate clause to allow $a_1^\cdot$ to name a world in a model represented by a frame image.

A branch of an **L**-tableau is said to be (atomically) closed if its leaf is, and an **L**-tableau is said to be (atomically) closed if all of its branches are. Otherwise, they are (atomically) open.



a	lab(a)	sgn(a)	ptype(a)	stype(a)
a_1	$\Box_1(p_1 \supset p_2) \supset \Box_1 p_1 \supset \Box_1 p_2$	0	α	
a_2	$\Box_1(p_1 \supset p_2)$	1	ν_1	α_1
a_3	$p_1 \supset p_2$	1	β	$\nu_{1,1}$
a_4	p_1	0		β_1
a_5	p_2	1		β_2
a_6	$\Box_1 p_1 \supset \Box_1 p_2$	0	α	α_2
a_7	$\Box_1 p_1$	1	ν_1	α_1
a_8	p_1	1		$\nu_{1,1}$
a_9	$\Box_1 p_2$	0	π_1	α_2
a_{10}	p_2	0		$\pi_{1,1}$

Figure 3.1: The signed formula tree and label, sign, and type functions of $(\Box_1(p_1 \supset p_2) \supset \Box_1 p_1 \supset \Box_1 p_2, 0)$ (example 3.1.1).

Example 3.1.1 (**L**-tableaus)**.** In this example, we show an atomically closed **K**-tableau for the signed formula $(A, 0) = (\Box_1(p_1 \supset p_2) \supset \Box_1 p_1 \supset \Box_1 p_2, 0)$. Recall that **K** is the unimodal logic ($n = 1$) with no frame conditions. The signed formula tree and label, sign, and type functions of $(A, 0)$ are shown in figure 3.1, while an atomically closed **K**-tableau for it is shown in figure 3.2. The tableau is broken up into several parts for formatting reasons, and for clarity, we have underlined the source of every tableau rule decomposition. At the leaf of each branch, we have also underlined the two elements that atomically close it.

The **L**-tableau formalism is a proof system that provides a characterization of **L**-validity for formulas. In particular, a formula A is **L**-valid if and only if there is a closed **L**-tableau for the signed formula $(A, 0)$. For instance, the atomically closed **K**-tableau for $(\Box_1(p_1 \supset p_2) \supset \Box_1 p_1 \supset \Box_1 p_2, 0)$ shown in figure 3.2 establishes the **K**-validity of the formula $\Box_1(p_1 \supset p_2) \supset \Box_1 p_1 \supset \Box_1 p_2$. We will prove the two directions

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{A}\qquad\mathcal{B}}{(\{a_1^{\cdot},a_{10}^{\cdot}\},\{(1,a_1^{\cdot},a_{10}^{\cdot})\},\{(a_1^{\cdot},a_1^{\cdot}),(a_1^{\cdot},a_6^{\cdot}),(a_1^{\cdot},a_9^{\cdot}),(a_{10}^{\cdot},a_{10}^{\cdot}),(a_1^{\cdot},a_2^{\cdot}),\underline{(a_{10}^{\cdot},a_3^1)}\})}\,(\beta)}{(\{a_1^{\cdot},a_{10}^{\cdot}\},\{(1,a_1^{\cdot},a_{10}^{\cdot})\},\{(a_1^{\cdot},a_1^{\cdot}),(a_1^{\cdot},a_6^{\cdot}),(a_1^{\cdot},a_9^{\cdot}),(a_{10}^{\cdot},a_{10}^{\cdot}),\underline{(a_1^{\cdot},a_2^{\cdot})}\})}\,(\nu)}{(\{a_1^{\cdot},a_{10}^{\cdot}\},\{(1,a_1^{\cdot},a_{10}^{\cdot})\},\{\underline{(a_1^{\cdot},a_1^{\cdot})},(a_1^{\cdot},a_6^{\cdot}),(a_1^{\cdot},a_9^{\cdot}),(a_{10}^{\cdot},a_{10}^{\cdot})\})}\,(\alpha_1)}{(\{a_1^{\cdot}\},\varnothing,\{(a_1^{\cdot},a_1^{\cdot}),(a_1^{\cdot},a_6^{\cdot}),\underline{(a_1^{\cdot},a_9^{\cdot})}\})}\,(\pi)}{(\{a_1^{\cdot}\},\varnothing,\{(a_1^{\cdot},a_1^{\cdot}),\underline{(a_1^{\cdot},a_6^{\cdot})}\})}\,(\alpha_2)}{(\{a_1^{\cdot}\},\varnothing,\{\underline{(a_1^{\cdot},a_1^{\cdot})}\})}\,(\alpha_2)$$

where

$$\mathcal{A}=\dfrac{\dfrac{(\{a_1^{\cdot},a_{10}^{\cdot}\},\{(1,a_1^{\cdot},a_{10}^{\cdot})\},\{(a_1^{\cdot},a_1^{\cdot}),(a_1^{\cdot},a_6^{\cdot}),(a_1^{\cdot},a_9^{\cdot}),(a_{10}^{\cdot},a_{10}^{\cdot}),(a_1^{\cdot},a_2^{\cdot}),(a_{10}^{\cdot},a_3^1),\underline{(a_{10}^{\cdot},a_4^1)},(a_1^{\cdot},a_7^{\cdot}),\underline{(a_{10}^{\cdot},a_8^1)}\})}{(\{a_1^{\cdot},a_{10}^{\cdot}\},\{(1,a_1^{\cdot},a_{10}^{\cdot})\},\{(a_1^{\cdot},a_1^{\cdot}),(a_1^{\cdot},a_6^{\cdot}),(a_1^{\cdot},a_9^{\cdot}),(a_{10}^{\cdot},a_{10}^{\cdot}),(a_1^{\cdot},a_2^{\cdot}),(a_{10}^{\cdot},a_3^1),(a_{10}^{\cdot},a_4^1),\underline{(a_1^{\cdot},a_7^{\cdot})}\})}\,(\nu)}{(\{a_1^{\cdot},a_{10}^{\cdot}\},\{(1,a_1^{\cdot},a_{10}^{\cdot})\},\{(a_1^{\cdot},a_1^{\cdot}),\underline{(a_1^{\cdot},a_6^{\cdot})},(a_1^{\cdot},a_9^{\cdot}),(a_{10}^{\cdot},a_{10}^{\cdot}),(a_1^{\cdot},a_2^{\cdot}),(a_{10}^{\cdot},a_3^1),(a_{10}^{\cdot},a_4^1)\})}\,(\alpha_1)$$

and

$$\mathcal{B}=(\{a_1^{\cdot},a_{10}^{\cdot}\},\{(1,a_1^{\cdot},a_{10}^{\cdot})\},\{(a_1^{\cdot},a_1^{\cdot}),(a_1^{\cdot},a_6^{\cdot}),(a_1^{\cdot},a_9^{\cdot}),\underline{(a_{10}^{\cdot},a_{10}^{\cdot})},(a_1^{\cdot},a_2^{\cdot}),(a_{10}^{\cdot},a_3^1),\underline{(a_{10}^{\cdot},a_5^1)}\})$$

Figure 3.2: An atomically closed **K**-tableau for $(\Box_1(p_1\supset p_2)\supset\Box_1p_1\supset\Box_1p_2,0)$ (example 3.1.1).

of this claim—the soundness and completeness of the tableau characterization—in sections 3.2 and 3.3, respectively.

3.2 Soundness

This section consists of a proof of the soundness of the tableau characterization, stated as follows.

Theorem 3.2.1 (Soundness of the tableau characterization)**.** *If there is a closed* **L***-tableau for a signed formula* $(A, 0)$*, then* A *is* **L***-valid.*

Suppose, for the sake of deriving a contradiction, that there is a closed **L**-tableau for $X = (A, 0)$, but A is not **L**-valid. Then there is an **L**-model $M = (W, R, V)$ and a $w \in W$ such that $M, w \nVdash A$. This means that the root $(\{a_1^{\cdot}\}, \varnothing, \{(a_1^{\cdot}, a_1^{\cdot})\})$ of our closed **L**-tableau for X is **L**-*satisfiable*, a property defined as follows.

Definition 3.2.1 (**L**-satisfiability of frame images)**.** A frame image (U, Q, S) of a signed formula is said to be *satisfied* by a model $M = (W, R, V)$ and function I from U to W if the following conditions hold.

1. If $(i, u, v) \in Q$, then $(i, I(u), I(v)) \in R$.

2. If $(u, a^{\iota}) \in S$ and $\mathrm{sgn}(a) = 0$, then $M, I(u) \nVdash \mathrm{lab}(a)$.

3. If $(u, a^{\iota}) \in S$ and $\mathrm{sgn}(a) = 1$, then $M, I(u) \Vdash \mathrm{lab}(a)$.

I is called an *interpretation*. (U, Q, S) is said to be **L**-*satisfiable* if there is an **L**-model and interpretation that satisfy it.

Notice that we are here formalizing a notion we hinted at in the previous chapter. Every node a of a signed formula is associated with a signed formula of its own, $(\mathrm{lab}(a), \mathrm{sgn}(a))$, and $\mathrm{sgn}(a)$ being 0 or 1 represents or encodes the falsehood or

truth, respectively, of $\mathrm{lab}(a)$ at some world of a countermodel. The second and third conditions of definition 3.2.1 capture this encoding.

The next result shows that if an **L**-tableau node is **L**-satisfiable, then at least one of its children, if it has any, is also **L**-satisfiable. A consequence of this is that since the root of our closed **L**-tableau for X is assumed to be **L**-satisfiable, it has at least one **L**-satisfiable leaf.

Lemma 3.2.1 (Preservation of **L**-satisfiability)**.** *Suppose (U, Q, S) is an* **L***-satisfiable frame image of a signed formula.*

1. *If S contains a pair (u, a^ι), where a has primary type α and children b and c, then $(U, Q, S \cup \{(u, b^\iota)\})$ and $(U, Q, S \cup \{(u, c^\iota)\})$ are* **L***-satisfiable.*

2. *If S contains a pair (u, a^ι), where a has primary type β and children b and c, then $(U, Q, S \cup \{(u, b^\iota)\})$ or $(U, Q, S \cup \{(u, c^\iota)\})$ is* **L***-satisfiable.*

3. *If S contains a pair (u, a^ι), where a has primary type ν_i and child b, and $(i, u, v) \in \mathrm{cl}_{\mathbf{L}}(Q)$, then for any positive integer l, $(U, Q, S \cup \{(v, b^{\iota l})\})$ is* **L***-satisfiable.*

4. *If S contains a pair (u, a^ι), where a has primary type π_i and child b, and $b^\iota \notin U$, then $(U \cup \{b^\iota\}, Q \cup \{(i, u, b^\iota)\}, S \cup \{(b^\iota, b^\iota)\})$ is* **L***-satisfiable.*

Proof. We show some representative cases. The omitted ones are analogous. In each case, let $M = (W, R, V)$ and I be an **L**-model and interpretation that satisfy (U, Q, S).

1. Suppose $(\mathrm{lab}(a), \mathrm{sgn}(a))$ has the form $(B \wedge C, 1)$, so $(\mathrm{lab}(b), \mathrm{sgn}(b)) = (B, 1)$ and $(\mathrm{lab}(c), \mathrm{sgn}(c)) = (C, 1)$. Since (U, Q, S) is **L**-satisfied by M and I, and $(u, a^\iota) \in S$, $M, I(u) \Vdash B \wedge C$, which means that $M, I(u) \Vdash B$ and $M, I(u) \Vdash C$. In other words, M and I also satisfy $(U, Q, S \cup \{(u, b^\iota)\})$ and $(U, Q, S \cup \{(u, c^\iota)\})$.

2. Suppose $(\mathrm{lab}(a), \mathrm{sgn}(a))$ has the form $(B \supset C, 1)$, so $(\mathrm{lab}(b), \mathrm{sgn}(b)) = (B, 0)$ and $(\mathrm{lab}(c), \mathrm{sgn}(c)) = (C, 1)$. Since (U, Q, S) is **L**-satisfied by M and I, and

$(u, a^\iota) \in S$, $M, I(u) \Vdash B \supset C$, which means that $M, I(u) \nVdash B$ or $M, I(u) \Vdash C$. In other words, M and I also satisfy either $(U, Q, S \cup \{(u, b^\iota)\})$ or $(U, Q, S \cup \{(u, c^\iota)\})$.

3. Suppose $(\mathrm{lab}(a), \mathrm{sgn}(a))$ has the form $(\Box_i B, 1)$, so $(\mathrm{lab}(b), \mathrm{sgn}(b)) = (B, 1)$. Since (U, Q, S) is **L**-satisfied by M and I, and $(u, a^\iota) \in S$, $M, I(u) \Vdash \Box_i B$, meaning that for every $w \in W$ such that $(i, I(u), w) \in R$, $M, w \Vdash B$. Since $(i, u, v) \in \mathrm{cl}_{\mathbf{L}}(Q)$, $(i, I(u), I(v)) \in R$ by corollary 2.3.2, so $M, I(v) \Vdash B$. In other words, for any positive integer l, M and I also satisfy $(U, Q, S \cup \{(v, b^{\iota l})\})$. We can say "for any positive integer" here, since the definition of **L**-satisfiability of frame images ignores the sequences of positive integers associated with node names.

4. Suppose $(\mathrm{lab}(a), \mathrm{sgn}(a))$ has the form $(\Diamond_i B, 1)$, so $(\mathrm{lab}(b), \mathrm{sgn}(b)) = (B, 1)$. Since (U, Q, S) is **L**-satisfied by M and I, and $(u, a^\iota) \in S$, $M, I(u) \Vdash \Diamond_i B$, meaning that there is a $w \in W$ such that $(i, I(u), w) \in R$ and $M, w \Vdash B$. Since $b^\iota \notin U$, I is not defined on b^ι. Let I' be an interpretation from $U \cup \{b^\iota\}$ to W obtained by extending I by letting $I'(b^\iota) = w$. Now $(i, I'(u), I'(b^\iota)) \in R$ and $M, I'(b^\iota) \Vdash B$. In other words, M and I' satisfy $(U \cup \{b^\iota\}, Q \cup \{(i, u, b^\iota)\}, S \cup \{(b^\iota, b^\iota)\})$. □

If our closed **L**-tableau for X has an **L**-satisfiable leaf, then this **L**-satisfiable leaf must also be closed. Suppose (U, Q, S) is a leaf satisfied by an **L**-model $M = (W, R, V)$ and interpretation I and also closed by $(u, a^\iota) \in S$ and $(u, b^\kappa) \in S$. If $\mathrm{sgn}(a) = 0$ and $\mathrm{sgn}(b) = 1$ (the other case is analogous), then $M, I(u) \nVdash \mathrm{lab}(a)$ and $M, I(u) \Vdash \mathrm{lab}(b)$. Since $\mathrm{lab}(a) = \mathrm{lab}(b)$, however, this is a contradiction, completing the proof of the soundness of the tableau characterization.

3.3 Completeness

This section consists of a proof of the completeness of the tableau characterization, stated as follows.

Theorem 3.3.1 (Completeness of the tableau characterization)**.** *If a formula A is* **L***-valid, then there is an atomically closed* **L***-tableau for* $(A, 0)$*.*

Notice that this result is not quite the converse of the soundness result of the previous section. Here, we are claiming that the **L**-validity of a formula A implies not just the existence of a closed **L**-tableau for $(A, 0)$, but of an *atomically* closed one. The converse of the soundess of the tableau characterization is a weaker result, which is however implied by theorem 3.3.1.

The completeness of tableau systems is typically proved using a *systematic construction procedure*, a deterministic procedure that, given a query formula, either constructs a closed tableau for it, establishing its validity, or generates evidence of a countermodel, implying its invalidity. In our case, given a query formula A, our procedure will attempt to construct an atomically closed **L**-tableau for $(A, 0)$. Such an attempt has one of three outcomes.

1. The procedure terminates and returns an atomically closed **L**-tableau for $(A, 0)$.

2. The procedure terminates and returns an atomically open **L**-tableau for $(A, 0)$.

3. The procedure does not terminate.

The procedure must be defined in such a way that the existence of a countermodel (an **L**-model in which A is false at some world) follows from the second and third outcomes. If the systematic construction procedure has this property and A is **L**-valid, then the procedure must terminate and return an atomically closed **L**-tableau for $(A, 0)$. Notice that this makes the systematic construction procedure a semidecision procedure for

L-validity, albeit a naive one. Before we describe the procedure, however, some new terminology is in order.

Definition 3.3.1 (**L**-viability)**.** Let (U, Q, S) be a frame image of a signed formula. A pair $(u, a^\iota) \in S$ is said to be **L**-*viable* if

1. a has primary type α and children b and c, and $(u, b^\iota) \notin S$ or $(u, c^\iota) \notin S$,
2. a has primary type β and children b and c, and $(u, b^\iota) \notin S$ and $(u, c^\iota) \notin S$,
3. a has primary type ν_i and child b, and there is a $v \in U$ such that $(i, u, v) \in \mathrm{cl}_{\mathbf{L}}(Q)$ and there is no positive integer l such that $(v, b^{\iota l}) \in S$, or
4. a has primary type π_i and child b, and $b^\iota \notin U$, $(i, u, b^\iota) \notin Q$, or $(b^\iota, b^\iota) \notin S$.

(U, Q, S) is said to be **L**-viable if there is an **L**-viable element in S. A branch of an **L**-tableau is said to be **L**-viable if its leaf is **L**-viable. If a frame image or a branch of an **L**-tableau is not **L**-viable, then it is **L**-*unviable*.

L-viability pins down when a tableau branch can be extended further to generate more information that could be used in closing it. An **L**-viable tableau branch has a leaf containing elements that can be decomposed further, so the systematic construction procedure will use **L**-viability as a test of whether an element of a frame image in an intermediate tableau is to be decomposed further.

In addition to using the notion of **L**-viability, the systematic construction procedure will associate some bookkeeping information with each leaf of every intermediate **L**-tableau it constructs. In particular, if (U, Q, S) is a leaf of an intermediate **L**-tableau, then we will associate with it a total order on S, which we will represent as a sequence whose elements are those of S. We will call this sequence the *bookeeping sequence* of (U, Q, S).

The purpose of this sequence is to guarantee fairness when choosing elements to decompose. If (U, Q, S) is associated with the bookkeeping sequence $s_1, s_2, \ldots, s_m$,

for instance, where each $s_l \in S$, then we will choose, as the element to decompose, the first **L**-viable element s_l of the sequence. With any new children of (U, Q, S) introduced as a result of the decomposition, we will associate a bookkeeping sequence in which s_l, the most recently decomposed element, is pushed to the back of the sequence, behind any new elements introduced to S and the bookkeeping sequence by the decomposition itself. We will specify this behaviour more precisely in the following definition.

Definition 3.3.2 (Systematic **L**-tableau construction procedure)**.** The *systematic* **L***-tableau construction procedure* for a signed formula X proceeds in stages. Some bookkeeping information is associated with each frame image (U, Q, S) in the form of a *bookkeeping sequence*, an ordering on S. This sequence determines the order in which elements of S are chosen for decomposition.

In stage 1, let the root of the **L**-tableau be $(\{a_1^{\cdot}\}, \varnothing, \{(a_1^{\cdot}, a_1^{\cdot})\})$. The bookkeeping sequence contains only $(a_1^{\cdot}, a_1^{\cdot})$. After stage n is complete, terminate if the **L**-tableau has no atomically open **L**-viable branches. Otherwise, carry out stage $n+1$ as follows.

Let the branch to be extended be the leftmost shortest atomically open **L**-viable branch, (U, Q, S) its leaf, s_1, s_2, ..., s_m the bookkeeping sequence of (U, Q, S), and $s_l = (u, a^\iota)$ the first **L**-viable element of S.

1. If a has primary type α and children b and c, and (u, a^ι) is **L**-viable because $(u, b^\iota) \notin S$, then apply rule (α_1) to α_1-decompose (u, a^ι) to (u, b^ι). Let the bookkeeping sequence of $(U, Q, S \cup \{(u, b^\iota)\})$ be s_1, s_2, ..., s_{l-1}, s_{l+1}, ..., s_m, (u, b^ι), (u, a^ι).

2. If a has primary type α and children b and c, and (u, a^ι) is **L**-viable because $(u, c^\iota) \notin S$, then apply rule (α_2) to α_2-decompose (u, a^ι) to (u, c^ι). Let the bookkeeping sequence of $(U, Q, S \cup \{(u, c^\iota)\})$ be s_1, s_2, ..., s_{l-1}, s_{l+1}, ..., s_m, (u, c^ι), (u, a^ι).

3. If a has primary type β and children b and c, and (u, a^ι) is **L**-viable because $(u, b^\iota) \notin S$ and $(u, c^\iota) \notin S$, then apply rule (β) to β-decompose (u, a^ι) to (u, b^ι) and (u, c^ι). Let the bookkeeping sequence of $(U, Q, S \cup \{(u, b^\iota)\})$ be s_1, s_2, ..., s_{l-1}, s_{l+1}, ..., s_m, (u, b^ι), (u, a^ι) and that of $(U, Q, S \cup \{(u, c^\iota)\})$ be s_1, s_2, ..., s_{l-1}, s_{l+1}, ..., s_m, (u, c^ι), (u, a^ι).

4. If a has primary type ν_i and child b, and (u, a^ι) is **L**-viable because there is a $v \in U$ such that $(i, u, v) \in \mathrm{cl}_{\mathbf{L}}(Q)$ and there is no positive integer l such that $(v, b^{\iota l}) \in S$, then for every such $v \in U$, apply rule (ν) to ν-decompose (u, a^ι) to $(v, b^{\iota l})$ for some positive integer l. Let $\{v_1, v_2, \ldots, v_k\} \subseteq U$ be all such worlds (since U is finite, there are finitely many of them, so this operation can be carried out in a finite number of steps), and for each $k' \in \{1, 2, \ldots, k\}$, let (u, a^ι) be ν-decomposed to $(v_{k'}, b^{\iota l_{k'}})$. Let the bookkeeping sequence of $(U, Q, S \cup \{(v_1, b^{\iota l_1}), (v_2, b^{\iota l_2}), \ldots, (v_k, b^{\iota l_k})\})$ be s_1, s_2, ..., s_{l-1}, s_{l+1}, ..., s_m, $(v_1, b^{\iota l_1})$, $(v_2, b^{\iota l_2})$, ..., $(v_k, b^{\iota l_k})$, (u, a^ι).

5. If a has primary type π_i and child b, and (u, a^ι) is **L**-viable because $b^\iota \notin U$, $(i, u, b^\iota) \notin Q$, or $(b^\iota, b^\iota) \notin S$, then since the tableau rules guarantee that only a π-decomposition to (b^ι, b^ι) can have any of these effects, such a π-decomposition has not yet taken place along this tableau branch, so $b^\iota \notin U$, and we can apply rule (π) to π-decompose (u, a^ι) to (b^ι, b^ι). Let the bookkeeping sequence of $(U \cup \{b^\iota\}, Q \cup \{(i, u, b^\iota)\}, S \cup \{(b^\iota, b^\iota)\})$ be s_1, s_2, ..., s_{l-1}, s_{l+1}, ..., s_m, (b^ι, b^ι), (u, a^ι).

The bookkeeping sequence of a leaf (U, Q, S) determines the order in which elements of S are decomposed, so informally, it implements a first-in-first-out queue of **L**-viable elements. It contains all the elements of S, but we ignore all but the **L**-viable ones. Intuitively, this makes the systematic construction procedure *fair*: if a branch of an intermediate **L**-tableau has a leaf with an **L**-viable element, then that element will

eventually be chosen for decomposition, unless the branch becomes atomically closed or the element becomes **L**-unviable. Moreover, **L**-tableaus are finitely branching, and by choosing to extend one of the shortest atomically open **L**-viable branches at every stage, we guarantee that no such branch is neglected indefinitely.

Recall the three possible outcomes of the systematic construction procedure:

1. The procedure terminates and returns an atomically closed **L**-tableau.

2. The procedure terminates and returns an atomically open **L**-tableau.

3. The procedure does not terminate.

The first case occurs if all branches of the constructed **L**-tableau are atomically closed. The second case occurs if each branch of the constructed **L**-tableau is either atomically closed or **L**-unviable, but at least one of them is atomically open by virtue of having an atomically open leaf. Any such atomically open leaf is still **L**-unviable, however, and any such atomically open **L**-unviable leaf encodes a countermodel for A. We will prove this shortly.

The third case is the most interesting one. Every iteration of the systematic construction procedure can be carried out in a finite number of steps and increases the length of a branch of the intermediate **L**-tableau by at least one, possibly forking it. Here, we use the assumptions that our logic has finitely many modalities and that formulas must be finite, so **L**-closures can be computed in finite time. This means that a nonterminating systematic construction procedure describes an "infinite open branch", i.e., an infinite sequence (U_1, Q_1, S_1), (U_2, Q_2, S_2), ... of atomically open frame images, where (U_1, Q_1, S_1) is the root $(\{a_1^{\cdot}\}, \varnothing, \{(a_1^{\cdot}, a_1^{\cdot})\})$, and for every $l \in \{1, 2, \ldots\}$, $(U_{l+1}, Q_{l+1}, S_{l+1})$ is the child of an application of a branch extension rule to (U_l, Q_l, S_l). What we will do is let $U = \bigcup_{l=1}^{\infty} U_l$, $Q = \bigcup_{l=1}^{\infty} Q_l$, and $S = \bigcup_{l=1}^{\infty} S_l$, and then show that (U, Q, S) is a countermodel for A. Note that the systematic construction procedure does not actually produce or return this potential countermodel,

but its existence is a consequence of the nontermination of the procedure.

We are now in a position to formally prove our claims from above. To this end, we require one more technical notion, namely that of **L**-*Hintikka frame images*. These are a variant of Hintikka sets [44], also known as downward saturated sets [16].

Definition 3.3.3 (**L**-Hintikka frame images)**.** A frame image (U, Q, S) of a signed formula is said to be an **L**-*Hintikka frame image* if it has the following properties.

1. It is atomically open.

2. If $(u, a^\iota) \in S$, where a has primary type α and children b and c, then $(u, b^\iota) \in S$ and $(u, c^\iota) \in S$.

3. If $(u, a^\iota) \in S$, where a has primary type β and children b and c, then $(u, b^\iota) \in S$ or $(u, c^\iota) \in S$.

4. If $(u, a^\iota) \in S$, where a has primary type ν_i and child b, then for every $v \in U$ such that $(i, u, v) \in \mathrm{cl}_{\mathbf{L}}(Q)$, there is a positive integer l such that $(v, b^{\iota l}) \in S$.

5. If $(u, a^\iota) \in S$, where a has primary type π_i and child b, then $b^\iota \in U$, $(i, u, b^\iota) \in \mathrm{cl}_{\mathbf{L}}(Q)$, and $(b^\iota, b^\iota) \in S$.

Notice the similarities between the clauses in definitions 3.3.1 (**L**-viability) and 3.3.3 (**L**-Hintikka frame images). The negation of each clause of definition 3.3.1 implies the conditions for the corresponding clause of definition 3.3.3, so if a tableau node (U, Q, S) is atomically open and **L**-unviable, then it is an **L**-Hintikka frame image.

The following lemma asserts that an **L**-Hintikka frame image is also **L**-satisfiable.

Lemma 3.3.1 (**L**-Hintikka frame image **L**-satisfiability)**.** *Every* **L**-*Hintikka frame image is* **L**-*satisfiable.*

Proof. We will show that an **L**-Hintikka frame image (U, Q, S) is satisfied by the model $M = (U, \mathrm{cl}_{\mathbf{L}}(Q), V)$ and the identity function I on U, where

$$V(p) = \{u \in U : (u, a^{\iota}) \in S, \mathrm{lab}(a) = p, \text{ and } \mathrm{sgn}(a) = 1\},$$

and that M is an **L**-model. First, notice that if $(i, u, v) \in Q$, then since $I(u) = u$, $I(v) = v$, and $Q \subseteq \mathrm{cl}_{\mathbf{L}}(Q)$, $(i, I(u), I(v)) \in \mathrm{cl}_{\mathbf{L}}(Q)$. Next, the claims that

1. if $(u, a^{\iota}) \in S$ and $\mathrm{sgn}(a) = 0$, then $M, I(u) \nVdash \mathrm{lab}(a)$, and
2. if $(u, a^{\iota}) \in S$ and $\mathrm{sgn}(a) = 1$, then $M, I(u) \Vdash \mathrm{lab}(a)$

are proved by simultaneous induction on the degree of $\mathrm{lab}(a)$. We show some representative cases. The omitted ones are analogous. In each case, we use one of rules 1–5 of definition 3.3.3.

1. If $(\mathrm{lab}(a), \mathrm{sgn}(a))$ has the form $(p, 0)$, then by rule 1, there is no $(u, b^{\kappa}) \in S$ such that $\mathrm{lab}(b) = \mathrm{lab}(a) = p$ and $\mathrm{sgn}(b) = 1$. This means that $u \notin V(p)$, so $M, u \nVdash p$, or $M, I(u) \nVdash \mathrm{lab}(a)$.

2. If $(\mathrm{lab}(a), \mathrm{sgn}(a))$ has the form $(p, 1)$, then $u \in V(p)$, so $M, u \Vdash p$, or $M, I(u) \Vdash \mathrm{lab}(a)$.

3. If $(\mathrm{lab}(a), \mathrm{sgn}(a))$ has the form $(B \wedge C, 1)$ and a has children b and c, then by rule 2, $(u, b^{\iota}) \in S$ and $(u, c^{\iota}) \in S$. $\mathrm{sgn}(b) = \mathrm{sgn}(c) = 1$, so by the induction hypothesis, $M, I(u) \Vdash \mathrm{lab}(b)$ and $M, I(u) \Vdash \mathrm{lab}(c)$, or $M, I(u) \Vdash B$ and $M, I(u) \Vdash C$, so $M, I(u) \Vdash B \wedge C$, or $M, I(u) \Vdash \mathrm{lab}(a)$.

4. If $(\mathrm{lab}(a), \mathrm{sgn}(a))$ has the form $(B \supset C, 1)$ and a has children b and c, then by rule 3, $(u, b^{\iota}) \in S$ or $(u, c^{\iota}) \in S$. $\mathrm{sgn}(b) = 0$ and $\mathrm{sgn}(c) = 1$, so by the induction hypothesis, $M, I(u) \nVdash \mathrm{lab}(b)$ or $M, I(u) \Vdash \mathrm{lab}(c)$, or $M, I(u) \nVdash B$ or $M, I(u) \Vdash C$, so $M, I(u) \Vdash B \supset C$, or $M, I(u) \Vdash \mathrm{lab}(a)$.

5. If $(\mathrm{lab}(a), \mathrm{sgn}(a))$ has the form $(\Box_i B, 1)$ and a has child b, then by rule 4, for every $v \in U$ such that $(i, u, v) \in \mathrm{cl}_{\mathbf{L}}(Q)$, $(v, b^{\iota l}) \in S$ for some positive integer l. $\mathrm{sgn}(b) = 1$, so by the induction hypothesis, $M, I(v) \Vdash \mathrm{lab}(b)$ or $M, v \Vdash B$ for every $v \in U$ such that $(i, u, v) \in \mathrm{cl}_{\mathbf{L}}(Q)$, so $M, u \Vdash \Box_i B$, or $M, I(u) \Vdash \mathrm{lab}(a)$.

6. If $(\mathrm{lab}(a), \mathrm{sgn}(a))$ has the form $(\Diamond_i B, 1)$ and a has child b, then by rule 5, $b^{\iota} \in U$, $(i, u, b^{\iota}) \in \mathrm{cl}_{\mathbf{L}}(Q)$, and $(b^{\iota}, b^{\iota}) \in S$. $\mathrm{sgn}(b) = 1$, so by the induction hypothesis, $M, I(b^{\iota}) \Vdash \mathrm{lab}(b)$, or $M, b^{\iota} \Vdash B$. Since $(i, u, b^{\iota}) \in \mathrm{cl}_{\mathbf{L}}(Q)$, this means that $M, u \Vdash \Diamond_i B$, or $M, I(u) \Vdash \mathrm{lab}(a)$.

We have shown that (U, Q, S) is satisfied by M and I. Now, since $\mathrm{cl}_{\mathbf{L}}(Q)$ satisfies all of **L**'s frame conditions by construction, M is an **L**-model, completing the proof that (U, Q, S) is **L**-satisfiable. □

To complete the proof of completeness, we will show that if the systematic construction procedure terminates and returns an atomically open tableau, or does not terminate, then there is an **L**-Hintikka frame image (U, Q, S) with $(a_1^{\cdot}, a_1^{\cdot}) \in S$. By the previous lemma, this frame image is **L**-satisfiable, so it falsifies A. Consequently, if A is **L**-valid, then the only possible outcome of the systematic construction procedure is an atomically closed **L**-tableau for A.

In the first case, if the systematic construction procedure terminates and returns an atomically open tableau, then it has an atomically open leaf (U, Q, S). Recall that this leaf must be an **L**-Hintikka frame image because it is **L**-unviable and atomically open. S contains $(a_1^{\cdot}, a_1^{\cdot})$, so A is not **L**-valid.

If the procedure does not terminate, then recall that (U, Q, S) is defined by an infinite open branch (U_1, Q_1, S_1), (U_2, Q_2, S_2), ... of atomically open frame images: $U = \bigcup_{l=1}^{\infty} U_l$, $Q = \bigcup_{l=1}^{\infty} Q_l$, and $S = \bigcup_{l=1}^{\infty} S_l$. Moreover, for every $l \in \{1, 2, \ldots\}$, $U_l \subseteq U_{l+1}$, $Q_l \subseteq Q_{l+1}$, and $S_l \subseteq S_{l+1}$. We will show shortly that (U, Q, S) is an **L**-Hintikka frame image. To do so, however, we require one more technical result,

namely that two elements (u, a^ι) and (v, a^ι), where $u \neq v$, cannot occur in the same tableau. There are two cases to consider.

1. Suppose $\iota = \iota' l$. This means that a has a nearest ν-ancestor a', since only ν-decompositions can extend the sequence of positive integers associated with node names. Since (u, a^ι) occurs in the tableau, a ν-decomposition of a pair of the form $(u', a'^{\iota'})$ is performed along the branch in which it occurs. Similarly, since (v, a^ι) occurs in the tableau, a ν-decomposition of a pair of the form $(v', a'^{\iota'})$ is performed along the branch in which it occurs. The proviso on the (ν) rule implies that regardless of how the tableau was constructed, the same positive integer l cannot have been used in both of these ν-decompositions, violating the legality of the tableau.

2. Suppose $\iota = \cdot$, meaning that a has no nearest ν-ancestor. From the tableau rules, it is clear that u and v are necessarily both the nearest π-ancestor of a with the integer sequence $\cdot$. This is because no ν-decompositions can have taken place along the relevant branches, and other than ν-decompositions, only π-decompositions affect the world names that nodes are associated with. Since the nearest π-ancestor of a is unique, $u = v$.

We can now show that (U, Q, S) is an **L**-Hintikka frame image as follows.

1. It is atomically open. If it were not, then since $S_l \subseteq S_{l+1}$ for every $l \in \{1, 2, \ldots\}$, there would be an atomically closed frame image along the branch.

2. If $(u, a^\iota) \in S$, where a has primary type α and children b and c, and $(u, b^\iota) \notin S$ or $(u, c^\iota) \notin S$, then (u, a^ι) is **L**-viable at some point along the branch, and the fairness of the systematic construction procedure guarantees that it is decomposed to (u, b^ι) and (u, c^ι) at some point, meaning that $(u, b^\iota) \in S$ and $(u, c^\iota) \in S$. This is a contradiction.

3. If $(u, a^\iota) \in S$, where a has primary type β and children b and c, and $(u, b^\iota) \notin S$ and $(u, c^\iota) \notin S$, then (u, a^ι) is **L**-viable at some point along the branch, and the fairness of the systematic construction procedure guarantees that it is decomposed to (u, b^ι) and (u, c^ι) at some point, meaning that $(u, b^\iota) \in S$ or $(u, c^\iota) \in S$. This is a contradiction.

4. If $(u, a^\iota) \in S$, where a has primary type ν_i and child b, and there is a $v \in U$ such that $(i, u, v) \in \mathrm{cl}_{\mathbf{L}}(Q)$, and there is no positive integer l such that $(v, b^{\iota l}) \in S$, then by corollary 2.3.3, there is an $m \in \{1, 2, \ldots\}$ such that $(i, u, v) \in \mathrm{cl}_{\mathbf{L}}(Q_m)$. There is also an $m' \in \{1, 2, \ldots\}$ such that $(u, a^\iota) \in S_{m'}$.

   If $m \geq m'$, then $S_{m'} \subseteq S_m$, so (u, a^ι) is **L**-viable at (U_m, Q_m, S_m). The fairness of the systematic construction procedure guarantees that it is decomposed to $(v, b^{\iota l})$ for some positive integer l at some point above (U_m, Q_m, S_m), meaning that $(v, b^{\iota l}) \in S$. This is a contradiction.

   If $m' \geq m$, then $U_m \subseteq U_{m'}$ and $Q_m \subseteq Q_{m'}$, so by corollary 2.3.1 $(i, u, v) \in \mathrm{cl}_{\mathbf{L}}(Q_{m'})$, so (u, a^ι) is **L**-viable at $(U_{m'}, Q_{m'}, S_{m'})$. The fairness of the systematic construction procedure guarantees that it is decomposed to $(v, b^{\iota l})$ for some positive integer l at some point above $(U_{m'}, Q_{m'}, S_{m'})$, meaning that $(v, b^{\iota l}) \in S$. This is a contradiction.

5. If $(u, a^\iota) \in S$, where a has primary type π_i and child b, and $b^\iota \notin U$, $(i, u, b^\iota) \notin \mathrm{cl}_{\mathbf{L}}(Q)$, or $(b^\iota, b^\iota) \notin S$, then there is an $m \in \{1, 2, \ldots\}$ such that $(u, a^\iota) \in S_m$ and is **L**-viable at (U_m, Q_m, S_m). To see this, consider the following three cases.

   (a) If $b^\iota \notin U$, then for every $m \in \{1, 2, \ldots\}$, $b^\iota \notin U_m$.

   (b) If $(i, u, b^\iota) \notin \mathrm{cl}_{\mathbf{L}}(Q)$, then $(i, u, b^\iota) \notin Q$, so for every $m \in \{1, 2, \ldots\}$, $(i, u, b^\iota) \notin Q_m$. This means that a ν-decomposition of (u, a^ι) to (b^ι, b^ι) does not occur anywhere along the branch, implying that for every $m \in$

$\{1, 2, \ldots\}$, $b^\iota \notin U_m$. There can also be no decomposition of (v, a^ι) to (b^ι, b^ι), where $u \neq v$, thanks to our previous observation.

(c) If $(b^\iota, b^\iota) \notin S$, then for every $m \in \{1, 2, \ldots\}$, $(b^\iota, b^\iota) \notin S_m$. This means that a ν-decomposition to (b^ι, b^ι) does not occur anywhere along the branch, implying that for every $m \in \{1, 2, \ldots\}$, $b^\iota \notin U_m$.

In each case, $(u, a^\iota) \in S_m$ for some $m \in \{1, 2, \ldots\}$, so (u, a^ι) is **L**-viable at (U_m, Q_m, S_m), and the fairness of the systematic construction procedure guarantees that it is decomposed to (b^ι, b^ι) at some point above (U_m, Q_m, S_m), meaning that $b^\iota \in U$, $(i, u, b^\iota) \in \mathrm{cl}_{\mathbf{L}}(Q)$, and $(b^\iota, b^\iota) \in S$. This is a contradiction.

Since (U, Q, S) is an **L**-Hintikka frame image, it is **L**-satisfiable. Once again, S contains $(a_1^{\cdot}, a_1^{\cdot})$, meaning that A is not **L**-valid, completing the proof of the completeness of the tableau characterization.

Because theorem 3.3.1 is stronger than the converse of theorem 3.2.1, a characterization of **L**-validity based on atomically closed **L**-tableaus is sound and complete with respect to one based on closed **L**-tableaus that are not necessarily atomically closed. One direction of this result is obvious, but the other is worth stating explicitly.

Corollary 3.3.1. *If there is a closed* **L***-tableau for a signed formula* $X = (A, 0)$*, then there is an atomically closed one.*

Proof. By theorem 3.2.1, the existence of a closed **L**-tableau for X implies that A is **L**-valid. By theorem 3.3.1, this implies that there is an atomically closed **L**-tableau for X. □

An alternative proof of this result involves defining a procedure that converts a closed **L**-tableau into an atomically closed one.

Chapter 4

Path characterization

The tableau characterization of the previous chapter is a convenient and powerful formalism, but like most tableau systems, it suffers from a problem Wallen calls *irrelevance* [48]. This is best demonstrated by example.

During the construction of a tableau derivation, certain disjunctive choices must be made. If (U, Q, S) is the leaf of a tableau, then we may have a choice as to which element of S to decompose. However, since tableau inference rules focus only on the primary types of nodes of the query formula, it is possible to make choices that are not relevant to producing a proof. Suppose a partial tableau derivation has produced an intermediate frame image (U, Q, S), for instance, where S contains two pairs (u, a^ι) and (u, b^κ) with the following properties:

1. $(\mathrm{lab}(a), \mathrm{sgn}(a)) = (B \wedge p_1, 1)$, where B is a large, complex subformula.

2. $(\mathrm{lab}(b), \mathrm{sgn}(b)) = (p_1, 0)$.

Applying rule (α_2) to decompose (u, a^ι) to $(u, {a'_2}^\iota)$, where $(\mathrm{lab}(a'_2), \mathrm{sgn}(a'_2)) = (p_1, 1)$, would close the branch. However, applying (α_1) to decompose (u, a^ι) to $(u, {a'_1}^\iota)$, where $(\mathrm{lab}(a'_1), \mathrm{sgn}(a'_1)) = (B, 1)$, could result in the branch becoming much longer before it closes. In other words, proof search procedures based on tableau inference rules cannot

inspect the deeper structure of the query formula in order to make better decisions, which may result in large portions of a derivation being essentially irrelevant.

Note that this is a problem with the tableau proof space rather than particular proof search strategies, since knowing which tableau rules to apply to approach a complete proof requires knowledge of the deeper structure of the query formula, which, unless the proof search procedure performs some more involved analysis, simply cannot be obtained without actually applying the intermediate decompositions. Although matrix and path characterizations also inspect the entire structure of a query formula, they do so without having to make the disjunctive choices that tableau systems are typically forced to make.

Wallen also identifies another important problem that plagues many sequent calculus and related tableau systems, which he calls *non-permutability*. Informally, this means that tableau rules cannot be applied in any order due to dependencies between different parts of the derivation. By applying rules in a particular order, the construction of a derivation might run into a dead end, and backtracking might be required during proof construction. Our tableau system does not suffer from this problem, since our logics are propositional, and the provisos on our modal inference rules are conservative in that they prevent rule applications that would lead to dead ends. In Wallen's words, there is no *order dependence* amongst the propositional rules [48], and although it is possible to perform irrelevant ν-decompositions in our tableau system, they do not prevent relevant ones from being carried out in the future.

However, our tableau derivations do mix different types of choices even though they are conceptually quite distinct: the structural choices that lead to irrelevance are of one kind, while the *modal* choices of which worlds to instantiate ν-nodes at are distinct concerns, and the goal of our path characterization is to explicitly separate them.

4.1 Definitions

Nodes of a signed formula tree represent subformula occurrences of the query formula. Informally, the path characterization uses nodes of a formula tree to denote not just subformula occurrences of a query formula, but also modal information about them. In a tableau, the π-decomposition of a node creates a new world (or a new name for a world required to exist) and adds it to the information known about the putative countermodel. In the path characterization, π-nodes, then, will have a special role as representing the concrete worlds for whose introduction they are responsible. This reflects our convention, in the tableau characterization, of using formula tree node names as world names (augmented by sequences of positive integers, which we will come to shortly). Meanwhile, in a tableau, the ν-decomposition of a node creates an association between the subformula occurrence that the ν-node represents and a world in the putative countermodel previously named by a node as a result of a π-decomposition. In the path characterization, then, ν-nodes will be *mapped to* π-nodes.

However, if we use a signed formula tree to perform this sort of node-based modal association, then we quickly run into the problem that each ν-node can be associated with only a single π-node. This does not sit comfortably with the fact that in tableaus, a node of primary type ν_i can be decomposed several times to "instantiate" its ν-child at more than one world of the countermodel. This issue is tackled by using not a signed formula tree for our node-based modal association, but an extension in which subtrees have been replicated or pruned according to a *multiplicity*, defined as follows.

Definition 4.1.1 (Multiplicities)**.** A *multiplicity* for a signed formula X is a partial function μ from tuples of the form (a, ι) to nonnegative integers, where a is the name of a ν-node of X, and ι is a sequence of positive integers, with the following properties.

1. If a is a ν-node with no ν-ancestors, then μ is defined on $(a, \cdot)$.

2. If a is a ν-node with a nearest ν-ancestor a', and μ is defined on (a', ι), then for

every $l \in \{1, 2, \ldots, \mu(a', \iota)\}$, μ is defined on $(a, \iota l)$.

A multiplicity μ is defined on a pair (a, ι) only if warranted by one of these two rules. Note that the second clause does not produce a circularity, since ι is shorter than ιl.

The purpose of a multiplicity, as its name might suggest, is to encode in how many ways a ν-node of X can, in the parlance of tableaus, be introduced through ν-decomposition. In other words, it tells us at how many potentially different worlds a ν-node can be instantiated. This is the intended high-level interpretation. At a more immediate level, a multiplicity encodes how parts of the signed formula tree of X are to be *replicated* or *pruned*. Our definition of multiplicities provides a very fine grain of control over this replication and pruning, and while its flexibility is not strictly necessary, it allows us to create a very close correspondence between proofs of validity in the tableau and path characterizations. This correspondence will be explored in greater detail in section 4.3, in which we prove the completeness of the path characterization with respect to the tableau characterization.

To formally implement the intended effect of a multiplicity μ on a signed formula X, we define an extension of the signed formula X and its signed formula tree in the following way.

Definition 4.1.2 (Indexed signed formulas)**.** An *indexed signed formula* is a pair (X, μ), where X is a signed formula, and μ is a multiplicity for X.

Definition 4.1.3 (Indexed signed formula trees)**.** The set of *indexed node names* of an indexed signed formula (X, μ) is a set of expressions of the form a^ι, where a is the name of a node of X, and ι is a sequence of positive integers, called an *index*. These expressions will serve as names of nodes in the *indexed signed formula tree* of (X, μ). With this in mind, we can simultaneously define both the indexed node names and the structure of the indexed signed formula tree of (X, μ) with the following rules.

1. $a_1^{\cdot}$ is an indexed node name, and it names the root of the indexed signed formula tree.

2. If a node a of X has a ν-child b, and a^{ι} is an indexed node name, then for every $l \in \{1, 2, \ldots, \mu(b, \iota)\}$, $b^{\iota l}$ is also an indexed node name, and it names a child of a^{ι} in the indexed signed formula tree.

3. If a node a of X has a non-ν-child b (i.e., a child b of secondary type other than $\nu_{i,1}$), and a^{ι} is an indexed node name, then b^{ι} is also an indexed node name, and it names a child of a^{ι} in the indexed signed formula tree.

In the second clause, we know μ to be defined on (b, ι) from its definition as a multiplicity. The following lines of reasoning show why this is so.

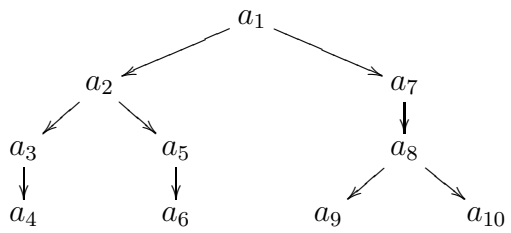
1. If b is a ν-node with no ν-ancestors, then the first and third clauses above imply that ι is the empty sequence $\cdot$, and clause 1 of definition 4.1.1 guarantees that μ is defined on $(b, \cdot)$, or (b, ι).

2. If b has a nearest ν-ancestor b', then the first and third clauses above again imply that b'^{ι} is also an indexed node name. Let ι have the form $\iota' l'$ (since b' is a ν-node, ι cannot be the empty index, by the second clause above), and let a' be the parent of b'. For b'^{ι} to be an indexed node name, $a'^{\iota'}$ must be one as well, and μ must be defined on (b', ι'), with $\mu(b', \iota') \geq l'$. So by clause 2 of definition 4.1.1, for every $l'' \in \{1, 2, \ldots, \mu(b', \iota')\}$, μ is defined on $(b, \iota' l'')$. Since $\mu(b', \iota') \geq l'$, μ is defined on $(b, \iota' l')$, or (b, ι).

The nodes of indexed signed formula trees inherit the labels, signs, and types of their progenitors:

$$\mathrm{lab}(a^{\iota}) = \mathrm{lab}(a) \qquad \mathrm{ptype}(a^{\iota}) = \mathrm{ptype}(a)$$

$$\mathrm{sgn}(a^{\iota}) = \mathrm{sgn}(a) \qquad \mathrm{stype}(a^{\iota}) = \mathrm{stype}(a)$$



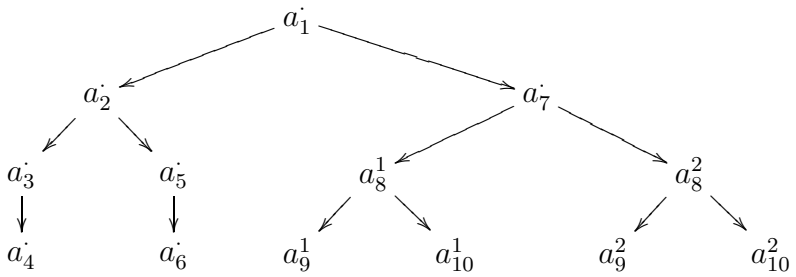
a	lab(a)	sgn(a)	ptype(a)	stype(a)
a_1	$\Diamond_1 p_1 \vee \Diamond_1 p_2 \supset \Diamond_1(p_1 \vee p_2)$	0	α	
a_2	$\Diamond_1 p_1 \vee \Diamond_1 p_2$	1	β	α_1
a_3	$\Diamond_1 p_1$	1	π_1	β_1
a_4	p_1	1		$\pi_{1,1}$
a_5	$\Diamond_1 p_2$	1	π_1	β_2
a_6	p_2	1		$\pi_{1,1}$
a_7	$\Diamond_1(p_1 \vee p_2)$	0	ν_1	α_2
a_8	$p_1 \vee p_2$	0	α	$\nu_{1,1}$
a_9	p_1	0		α_1
a_{10}	p_2	0		α_2

Figure 4.1: The signed formula tree and label, sign, and type functions of $(\Diamond_1 p_1 \vee \Diamond_1 p_2 \supset \Diamond_1(p_1 \vee p_2), 0)$ (example 4.1.1).

The terminology of ν- and π-nodes from signed formula trees, as well as the modalities of nodes, also carries over to indexed signed formula trees. If $\ll$ is the tree ordering of the signed formula tree of X, then we will denote the corresponding relation on nodes of the indexed signed formula tree of (X, μ) by $\ll^\mu$. However, we will systematically omit the superscript wherever we can safely do so without causing confusion.

Example 4.1.1 (Indexed signed formula trees)**.** The signed formula tree and label, sign, and type functions of the signed formula $X = (\Diamond_1 p_1 \vee \Diamond_1 p_2 \supset \Diamond_1(p_1 \vee p_2), 0)$ are shown in figure 4.1. If μ is defined to be a multiplicity for X such that $\mu(a_8, \cdot) = 2$, then the indexed signed formula tree and label, sign, and type functions of the indexed signed formula (X, μ) are shown in figure 4.2. Notice the duplication of the subtree rooted at a^8 and how the indices distinguish between the copies.

The indices of the indexed node names of the indexed signed formula tree of (X, μ)

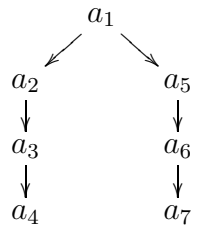


a	$\mathrm{lab}(a)$	$\mathrm{sgn}(a)$	$\mathrm{ptype}(a)$	$\mathrm{stype}(a)$
$a_1^{\cdot}$	$\Diamond_1 p_1 \vee \Diamond_1 p_2 \supset \Diamond_1(p_1 \vee p_2)$	0	α	
$a_2^{\cdot}$	$\Diamond_1 p_1 \vee \Diamond_1 p_2$	1	β	α_1
$a_3^{\cdot}$	$\Diamond_1 p_1$	1	π_1	β_1
$a_4^{\cdot}$	p_1	1		$\pi_{1,1}$
$a_5^{\cdot}$	$\Diamond_1 p_2$	1	π_1	β_2
$a_6^{\cdot}$	p_2	1		$\pi_{1,1}$
$a_7^{\cdot}$	$\Diamond_1(p_1 \vee p_2)$	0	ν_1	α_2
a_8^1	$p_1 \vee p_2$	0	α	$\nu_{1,1}$
a_9^1	p_1	0		α_1
a_{10}^1	p_2	0		α_2
a_8^2	$p_1 \vee p_2$	0	α	$\nu_{1,1}$
a_9^2	p_1	0		α_1
a_{10}^2	p_2	0		α_2

Figure 4.2: The indexed signed formula tree and label, sign, and type functions of $((\Diamond_1 p_1 \vee \Diamond_1 p_2 \supset \Diamond_1(p_1 \vee p_2), 0), \mu)$, where $\mu(a_8, \cdot) = 2$ (example 4.1.1).

provide a form of multiplicity-context. The multiplicity causes subtrees rooted at ν-nodes to be copied or deleted (this mechanism is driven by clause 2 of definition 4.1.3), and the indices of the copied nodes are a bookkeeping tool that allow us to identify which copied subtree they belong to.

The expression "copying subtrees" in the context of multiplicities should be taken with a grain of salt, however, since if a subtree rooted at a ν-node a is "copied" by virtue of μ being defined on and greater than one for some pair (a, ι), then each "copy" may be different if the subtree contains further ν-nodes. This is one important way in which our definition of multiplicities differs from Wallen's, whose multiplicities are simply functions from ν-nodes to nonnegative integers. The reason we have opted



a	lab(a)	sgn(a)	ptype(a)	stype(a)
a_1	$\Diamond_1\Diamond_2 p_1 \supset \Diamond_2\Diamond_1 p_1$	0	α	
a_2	$\Diamond_1\Diamond_2 p_1$	1	π_1	α_1
a_3	$\Diamond_2 p_1$	1	π_2	$\pi_{1,1}$
a_4	p_1	1		$\pi_{2,1}$
a_5	$\Diamond_2\Diamond_1 p_1$	0	ν_2	α_2
a_6	$\Diamond_1 p_1$	0	ν_1	$\nu_{2,1}$
a_7	p_1	0		$\nu_{1,1}$

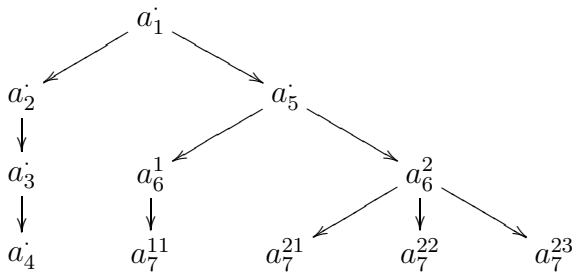
Figure 4.3: The signed formula tree and label, sign, and type functions of $(\Diamond_1\Diamond_2 p_1 \supset \Diamond_2\Diamond_1 p_1, 0)$ (example 4.1.2).

for this slightly more complicated definition of multiplicities is that it makes the relationship between our tableau and path formalisms (and the translation of proofs from one to the other) more direct. We will witness this in the proof of completeness of our path characterization later in this chapter.

Example 4.1.2 (Indexed signed formula trees)**.** The signed formula tree and label, sign, and type functions of the signed formula $X = (\Diamond_1\Diamond_2 p_1 \supset \Diamond_2\Diamond_1 p_1, 0)$ are shown in figure 4.3. If μ is defined to be a multiplicity for X such that $\mu(a_6, \cdot) = 2$, $\mu(a_7, 1) = 1$, and $\mu(a_7, 2) = 3$, then the indexed signed formula tree and label, sign, and type functions of the indexed signed formula (X, μ) are shown in figure 4.4. Notice that the subtrees rooted at a_6^1 and a_6^2 are quite different, since $\mu(a_7, 1) \neq \mu(a_7, 2)$.

The node-based modal association of nodes of the indexed signed formula tree can now be represented using a *realization*.

Definition 4.1.4 (Realizations)**.** A *realization* for an indexed signed formula Z is a pair (σ, ρ), where σ is a function mapping ν-nodes of Z to π-nodes of Z, and ρ is a



a	lab(a)	sgn(a)	ptype(a)	stype(a)
$a_1^{\cdot}$	$\Diamond_1\Diamond_2 p_1 \supset \Diamond_2\Diamond_1 p_1$	0	α	
$a_2^{\cdot}$	$\Diamond_1\Diamond_2 p_1$	1	π_1	α_1
$a_3^{\cdot}$	$\Diamond_2 p_1$	1	π_2	$\pi_{1,1}$
$a_4^{\cdot}$	p_1	1		$\pi_{2,1}$
$a_5^{\cdot}$	$\Diamond_2\Diamond_1 p_1$	0	ν_2	α_2
a_6^1	$\Diamond_1 p_1$	0	ν_1	$\nu_{2,1}$
a_7^{11}	p_1	0		$\nu_{1,1}$
a_6^2	$\Diamond_1 p_1$	0	ν_1	$\nu_{2,1}$
a_7^{21}	p_1	0		$\nu_{1,1}$
a_7^{22}	p_1	0		$\nu_{1,1}$
a_7^{23}	p_1	0		$\nu_{1,1}$

Figure 4.4: The indexed signed formula tree and label, sign, and type functions of $((\Diamond_1\Diamond_2 p_1 \supset \Diamond_2\Diamond_1 p_1, 0), \mu)$, where $\mu(a_6, \cdot) = 2$, $\mu(a_7, 1) = 1$, and $\mu(a_7, 2) = 3$ (example 4.1.2).

function mapping ν-nodes of Z to sets of π-nodes of Z such that for every ν-node z of Z, $\sigma(z) \in \rho(z)$.

A realization $\Omega = (\sigma, \rho)$ induces a partial ordering $\sqsubset$ on the nodes of Z by the rule that $z \sqsubset z'$ if and only if $z \in \rho(z')$. This is called the *realization ordering* of Ω. Given a realization Ω for an indexed signed formula Z, we can combine the tree ordering $\ll$ of Z and the realization ordering $\sqsubset$ of Ω into a single ordering $\lhd = (\ll \cup \sqsubset)^*$ (the transitive closure of the union of $\ll$ and $\sqsubset$). This is called the *reduction ordering* of Z and Ω.

The function σ maps ν-nodes to π-nodes, and we will make use of an extension σ^* of σ that maps arbitrary nodes to π-nodes. It is obtained directly from σ in the following way.

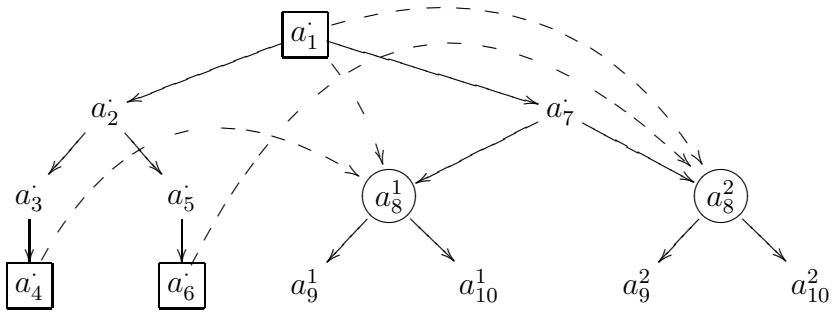


Figure 4.5: The indexed signed formula tree of $((\Diamond_1 p_1 \vee \Diamond_1 p_2 \supset \Diamond_1(p_1 \vee p_2), 0), \mu)$, where $\mu(a_8, \cdot) = 2$. π- and ν-nodes are enclosed in squares and circles, respectively, and dashed arrows represent a realization ordering $\sqsubset$ (example 4.1.3).

1. If z is a π-node, then $\sigma^*(z) = z$.

2. If z is a ν-node, then $\sigma^*(z) = \sigma(z)$.

3. If z is neither a π- nor ν-node, then $\sigma^*(z) = \sigma^*(z')$, where z' is the parent of z.

Note that the function σ^* is well-defined, since the root is defined to be a π-node.

Example 4.1.3 (Realizations)**.** Let $Z = (X, \mu)$ be the indexed signed formula described in example 4.1.1. If σ and ρ are such that $\sigma(a_8^1) = a_4^{\cdot}$, $\sigma(a_8^2) = a_6^{\cdot}$, $\rho(a_8^1) = \{a_1^{\cdot}, a_4^{\cdot}\}$, and $\rho(a_8^2) = \{a_1^{\cdot}, a_6^{\cdot}\}$, then (σ, ρ) is a realization for Z. In this case, $\sigma^*(a_1^{\cdot}) = a_1^{\cdot}$, $\sigma^*(a_3^{\cdot}) = a_1^{\cdot}$, $\sigma^*(a_6^{\cdot}) = a_6^{\cdot}$, $\sigma^*(a_8^2) = a_6^{\cdot}$, and $\sigma^*(a_9^2) = a_6^{\cdot}$, for instance.

Figure 4.5 shows the indexed signed formula tree of Z with π- and ν-nodes enclosed in squares and circles, respectively, and dashed arrows representing the realization ordering $\sqsubset$ of Ω. The transitive closure of the relation represented by all arrows in the figure thus represents the reduction ordering $\lhd$ of Z and Ω.

A key insight in understanding the relationship between tableau and path characterizations is that the nodes of an indexed signed formula tree can be thought of as representing tableau inference rules. Each tableau rule is a decomposition that introduces a node of the query formula in some modal context, i.e., at some world of the assumed countermodel. The nodes of an indexed signed formula can represent

the rule applications that introduce the nodes of the signed query formula from which they are obtained. The modal context, i.e., the world at which the decomposition is taking place, is provided by σ^* (recall that the names of π-nodes represent the worlds for whose "creation" they carry responsibility in the setting of tableaus).

What is the role of the ordering induced by the realization, then? Why do we require a function ρ to induce a reduction ordering on the nodes of an indexed signed formula? The answer is that tableau rule applications can depend on others. The provisos on tableau rules guarantee that no world is used before it is created, and these provisos translate naturally into requirements for rules represented by indexed signed formula tree nodes to be applied in an order that respects the reduction ordering. This explains why $\sigma(z)$, where z is a ν-node, should "precede" z in some way: $\sigma(z)$ is the π-node responsible for introducing the world that z is instantiated at.

The function ρ exists because the dependence amongst rule applications is not always this simple. Before a ν-node can be instantiated, its "target" world, which some π-node is responsible for, must not only exist, but must be accessible from the world at which the ν-node's parent has been instantiated (this is formalized later in this section, in definition 4.1.8). Since our frame conditions can encode interacting modalities and these interactions can be quite complex, this accessibility might depend on other π-nodes having been created before the instantiation of the ν-node takes place. Consequently, the set $\rho(z)$ contains those π-nodes whose worlds are needed for z to be introduced by a ν-decomposition. This includes $\sigma(z)$, the "target", which is why $\sigma(z) \in \rho(z)$.

The way we have defined the extended function σ^* guarantees that it is compatible with the reduction ordering in the following way. This stems directly from the fact that σ is compatible with the reduction ordering by design.

Lemma 4.1.1. *If Ω is a realization for an indexed signed formula Z, and $\lhd$ is their reduction ordering, then for every node z of Z, $\sigma^*(z) \trianglelefteq z$.*

Proof. By induction on the distance of z from the root.

1. If z is π-modal, then $\sigma^*(z) = z$.

2. If z is ν-modal, then $\sigma^*(z) = \sigma(z)$, and $\sigma(z) \in \rho(z)$, so $\sigma(z) \sqsubset z$.

3. If z is neither π- nor ν-modal, then $\sigma^*(z) = \sigma^*(z')$, where z' is the parent of z. By the induction hypothesis, $\sigma^*(z') \trianglelefteq z'$, and $z' \ll z$. □

We have alluded to the correspondence between the nodes of indexed signed formulas and tableau rule applications, suggesting that a reduction ordering implements the ordering on tableau rule applications caused by the provisos on tableau rules in the context of indexed signed formula tree nodes. Taking the correspondence further, a set of indexed signed formula tree nodes can be thought of as encoding a sequence of tableau rule applications (in other words, a tableau branch), as long as the set of nodes *respects the reduction ordering* in the following way.

Definition 4.1.5 (Respecting the reduction ordering)**.** Let Ω be a realization for an indexed signed formula Z, and let $\lhd$ be their reduction ordering. A subset ϕ of the nodes of Z is said to *respect* $\lhd$ if for every node $z \in \phi$, ϕ contains every node of Z that is $\lhd$-smaller than z.

We have encoded the possibility of multiple instantiations of ν-nodes using multiplicities and the instantiations themselves using realizations. Next, we introduce Ω-*paths*, sets of nodes of an indexed signed formula that capture all the information that may be used in a countermodel. They correspond exactly to tableau branches.

Definition 4.1.6 (Ω-paths)**.** Let Ω be a realization for an indexed signed formula Z, and let $\lhd$ be their reduction ordering. An Ω-*path through* Z is a subset of the nodes of Z. The set of Ω-paths through Z is defined as follows.

1. $\{a_1^{\cdot}\}$ is an Ω-path.

2. If ϕ is an Ω-path containing a node z of primary type α, as well as every node of Z that is $\lhd$-smaller than the left child z' of z, but not z' itself, then $\phi' = \phi \cup \{z'\}$ is an Ω-path, and ϕ is said to be α_1-*reducible* to ϕ'.

3. If ϕ is an Ω-path containing a node z of primary type α, as well as every node of Z that is $\lhd$-smaller than the right child z' of z, but not z' itself, then $\phi' = \phi \cup \{z'\}$ is an Ω-path, and ϕ is said to be α_2-*reducible* to ϕ'.

4. If ϕ is an Ω-path containing a node z of primary type β, as well as every node of Z that is $\lhd$-smaller than the children z'_1 and z'_2 of z, but not z'_1 and z'_2 themselves, then $\phi'_1 = \phi \cup \{z'_1\}$ and $\phi'_2 = \phi \cup \{z'_2\}$ are Ω-paths, and ϕ is said to be β-*reducible* to ϕ'_1 and ϕ'_2.

5. If ϕ is an Ω-path containing a node z of primary type ν_i, as well as every node of Z that is $\lhd$-smaller than a child z' of z, but not z' itself, then $\phi' = \phi \cup \{z'\}$ is an Ω-path, and ϕ is said to be ν-*reducible* to ϕ'.

6. If ϕ is an Ω-path containing a node z of primary type π_i, as well as every node of Z that is $\lhd$-smaller than the child z' of z, but not z' itself, then $\phi' = \phi \cup \{z'\}$ is an Ω-path, and ϕ is said to be π-*reducible* to ϕ'.

If an Ω-path is not reducible, then it is *irreducible*.

Notice that, by construction, all Ω-paths respect $\lhd$. An Ω-path ϕ through Z is said to be *closed* if it contains two nodes z and z' such that $\mathrm{lab}(z) = \mathrm{lab}(z')$, $\mathrm{sgn}(z) \neq \mathrm{sgn}(z')$, and $\sigma^*(z) = \sigma^*(z')$. If $\mathrm{lab}(z)$ and $\mathrm{lab}(z')$ are atomic, then ϕ is said to be *atomically closed*. This overloading of tableau-terminology (concerning frame images, in particular) is quite deliberate. Every tableau branch is represented by its leaf, and Ω-paths correspond naturally to tableau branches. Irreducible Ω-paths correspond to maximally extended tableau branches, i.e., branches whose leaves cannot fruitfully be decomposed any further.

We can make several useful observations about irreducible paths. For instance, if ϕ is an irreducible Ω-path through Z, then the following statements are all true.

1. If ϕ contains a node z of primary type α, then ϕ also contains all of its children. This is because every node of Z that is $\lhd$-smaller than a child of z is also $\lhd$-smaller than or equal to z itself (since $\sqsubset$ is only defined on ν- and π-nodes), so ϕ already contains every node $\lhd$-smaller than any child of z. If it did not contain all of z's children, then it would be reducible.

2. If ϕ contains a node z of primary type β, then by similar reasoning as in the previous case, ϕ also contains one of its children.

3. If ϕ contains a node z of primary type ν_i and every node $\sqsubset$-smaller than a child z' of z, then ϕ also contains z'. A node z'' of Z can only be $\lhd$-smaller than z' if there is a node z''' that is $\ll$-smaller than z' and $z'' \trianglelefteq z'''$, or z''' is $\sqsubset$-smaller than z' and $z'' \trianglelefteq z'''$. Since z is the parent of z', if z''' is $\ll$-smaller than z', then $z''' \trianglelefteq z$, so $z''' \in \phi$ (since ϕ respects $\lhd$ and $z \in \Omega$). In the other case, ϕ contains z''' by assumption. In either case, $z'' \in \phi$, so every node that is $\lhd$-smaller than z' is already an element of ϕ. If z' were not an element of ϕ, then ϕ would be reducible.

4. If ϕ contains a node z of primary type π_i, then by similar reasoning as in the first two cases, ϕ also contains z's child.

These properties of irreducible paths will be used in the proof of completeness of the path characterization later in this chapter.

Example 4.1.4 (Ω-paths)**.** Let $Z = (X, \mu)$ be the indexed signed formula described in example 4.1.1, and let Ω be the realization for Z described in example 4.1.3. The irreducible Ω-paths of Z are

$$\{a_1^{\cdot}, a_2^{\cdot}, a_3^{\cdot}, a_4^{\cdot}, a_7^{\cdot}, a_8^1, a_9^1, a_{10}^1\}$$

and

$$\{a_1^{\cdot}, a_2^{\cdot}, a_5^{\cdot}, a_6^{\cdot}, a_7^{\cdot}, a_8^2, a_9^2, a_{10}^2\}.$$

We can make the connection between Ω-paths and frame images more vivid by defining an association between Ω-paths and frames.

Definition 4.1.7 (Associated frames)**.** For any set ϕ of nodes of Z that respects $\lhd$, we can associate a frame (U, Q). In particular, let U be the set of π-nodes in ϕ, and define Q using the rule that if $z \in U$ is a nonroot π-node of Z with parent z' and modality i, then $(i, \sigma^*(z'), z) \in Q$. Since $\sigma^*(z') \trianglelefteq z' \ll z$ and ϕ respects $\lhd$, $\sigma^*(z') \in U$.

A useful property of associated frames is that if ϕ and ϕ' are sets of nodes of Z that respect $\lhd$, with associated frames (U, Q) and (U', Q'), respectively, and $\phi \subseteq \phi'$, then $U \subseteq U'$ and $Q \subseteq Q'$. That $U \subseteq U'$ is trivial. If $(i, \sigma^*(z'), z) \in Q$, where i is the modality and z' the parent of $z \in U$, then $z \in U'$, and its modality and parent are still i and z' in ϕ', so $(i, \sigma^*(z'), z) \in Q'$. This holds for every accessibility triple in Q, so $Q \subseteq Q'$.

The realization encodes instantiations of ν-nodes to π-nodes (or rather, to the worlds they are responsible for introducing), but we have not yet introduced any mechanism for guaranteeing that these instantions are *legal*. In other words, if a ν-node z is associated with a π-node z', i.e., $\sigma(z) = z'$, the realization by itself does not require that in some putative countermodel, the world associated with the parent of z can i-access the world created by z', where i is the modality of z. This is a property of the logic under consideration, and so far, we have not even mentioned any distinctions between logics.

In a way, it is a testament to the modularity of the path characterization that our development so far has been independent of the logic being considered. However, it does enter into the game now in the form of a property of **L**-*admissiblity* defined on realizations.

Definition 4.1.8 (**L**-admissibility of realizations)**.** A realization Ω for an indexed signed formula Z is said to be **L**-*admissible* if, for every ν-modal node z of Z with parent z' and modality i, $(i, \sigma^*(z'), \sigma^*(z)) \in \mathrm{cl}_{\mathbf{L}}(Q)$, where Q is the accessibility relation of the frame associated with the set of nodes of Z that are $\lhd$-smaller than z.

Example 4.1.5 (**L**-admissibility of realizations)**.** Let $Z = (X, \mu)$ be the indexed signed formula described in example 4.1.1, and let Ω be the realization for Z described in example 4.1.3. The ν-modal nodes of Z are a_8^1 and a_8^2, and the sets of nodes of Z that are $\lhd$-smaller than a_8^1 and a_8^2 are $\{a_1^{\cdot}, a_2^{\cdot}, a_3^{\cdot}, a_4^{\cdot}, a_7^{\cdot}\}$ and $\{a_1^{\cdot}, a_2^{\cdot}, a_5^{\cdot}, a_6^{\cdot}, a_7^{\cdot}\}$, respectively. The frames they are associated with are $(\{a_1^{\cdot}, a_4^{\cdot}\}, \{(1, a_1^{\cdot}, a_4^{\cdot})\})$ and $(\{a_1^{\cdot}, a_6^{\cdot}\}, \{(1, a_1^{\cdot}, a_6^{\cdot})\})$, respectively.

Because $(1, \sigma^*(a_7), \sigma^*(a_8^1)) = (1, a_1^{\cdot}, a_4^{\cdot})$ and $(1, \sigma^*(a_7), \sigma^*(a_8^2)) = (1, a_1^{\cdot}, a_6^{\cdot})$, it is the case that for every ν-modal node z of Z with parent z' and modality i, $(i, \sigma^*(z'), \sigma^*(z)) \in Q$, where Q is the accessibility relation of the frame associated with the set of nodes of Z that are $\lhd$-smaller than z. For any logic **L**, $Q \subseteq \mathrm{cl}_{\mathbf{L}}(Q)$, so this demonstrates that Ω is **L**-admissible for any **L** (with $n \geq 1$).

Notice that for a realization to be **L**-admissible, the required accessibility for a ν-node z is forced in the frame associated with the set of nodes $\lhd$-smaller than z. This is why ρ, which defines $\sqsubset$ and thus affects $\lhd$, is defined in the way that it is. If $\Omega = (\rho, \sigma)$ is **L**-admissible, then $\rho(z)$ describes exactly which worlds associated with π-nodes are needed for an instantiation to succeed. So ρ and the intended meaning behind **L**-admissibility go hand in hand.

The definitions and obervations in this section, although they might seem a little ad hoc at first glance, come together to form a characterization of **L**-validity for formulas. In particular, a formula A is **L**-valid if and only if there is a multiplicity μ for the signed formula $(A, 0)$ and an **L**-admissible realization Ω for the indexed signed formula $Z = (X, \mu)$ such that every irreducible Ω-path through Z is atomically closed. This is the *path characterization of validity*. We will prove the two directions of this

claim—the soundness and completeness of the path characterization—in sections 4.2 and 4.3, respectively.

Example 4.1.6 (Path characterization of validity)**.** Let $Z = (X, \mu)$ be the indexed signed formula described in example 4.1.1, and let Ω be the realization for Z described in example 4.1.3. As we saw in example 4.1.5, Ω is **L**-admissible for any **L**. In example 4.1.4, we saw that the irreducible Ω-paths through Z are

$$\{a_1^{\cdot}, a_2^{\cdot}, a_3^{\cdot}, a_4^{\cdot}, a_7^{\cdot}, a_8^1, a_9^1, a_{10}^1\}$$

and

$$\{a_1^{\cdot}, a_2^{\cdot}, a_5^{\cdot}, a_6^{\cdot}, a_7^{\cdot}, a_8^2, a_9^2, a_{10}^2\}.$$

The first is atomically closed by $a_4^{\cdot}$ and a_9^1, and the second is atomically closed by $a_6^{\cdot}$ and a_{10}^2. We can conclude that $\Diamond_1 p_1 \vee \Diamond_1 p_2 \supset \Diamond_1(p_1 \vee p_2)$ is **L**-valid for any **L** (with $n \geq 1$).

Readers familiar with Wallen's matrix characterizations may have expected the definition of **L**-admissibility of a realization Ω with reduction ordering $\lhd$ to include a clause requiring $\lhd$ to be acyclic (or irreflexive, equivalently). The reason we do not require such a clause is that our paths differ from Wallen's in that they are defined using $\lhd$. A cycle in $\lhd$ would preclude the possibility of any nodes on the cycle appearing in an Ω-path, rendering them useless for the purpose of closing any paths and thus completing a proof.

4.2 Soundness

This section consists of a proof of the soundness of the path characterization, stated as follows.

Theorem 4.2.1 (Soundness of the path characterization)**.** *If there is a multiplicity μ for a signed formula $X = (A, 0)$ and an* **L***-admissible realization Ω for the indexed signed formula $Z = (X, \mu)$ such that every irreducible Ω-path through Z is atomically closed, then A is* **L***-valid.*

One way of proving this theorem is to use the soundness of the tableau system presented in chapter 3. In particular, μ and Ω can be used to construct an atomically closed **L**-tableau for A, and theorem 3.2.1 asserts that the existence of such a tableau implies that A is **L**-valid. Translations of this kind from matrix proofs to sequent-style proofs have been explored by Kreitz and Schmitt [29]. Since the parallels between our tableau and path characterizations are strong, the translation would be quite direct.

The proof we give in this section avoids such an explicit detour through tableaus, but follows the same pattern as the proof of the soundness of the tableau characterization. We will also use the notion of frame images, but not in the context of tableaus. More specifically, we will associate a frame image (U, Q, S) with every Ω-path ϕ through Z. In particular, let U be the set of π-nodes in ϕ, let Q be defined by the rule that for every nonroot π-node $z \in U$ with modality i and parent z', $(i, \sigma^*(z'), z) \in Q$, and for every $z \in \phi$, let $(\sigma^*(z), z) \in S$. Notice that (U, Q) is simply the frame associated with ϕ, as defined in the previous section, and that if $z \in \phi$, then $\sigma^*(z) \in U$, since $\sigma^*(z)$ is a π-node $\lhd$-smaller than or equal to z.

Having associated frame images with Ω-paths, we can borrow terminology related to frame images and apply it to paths. We will say, for instance, that an Ω-path ϕ is **L**-satisfiable if its associated frame image is **L**-satisfiable. The following lemma is then the path-equivalent of lemma 3.2.1.

Lemma 4.2.1 (Preservation of **L**-satisfiability for paths)**.** *Let Ω be an* **L***-admissible realization for an indexed signed formula Z, and let ϕ be an* **L***-satisfiable Ω-path through Z.*

1. *If ϕ is α_1-, α_2-, ν-, or π-reducible to ϕ', then ϕ' is* **L***-satisfiable.*

2. *If ϕ is β-reducible to ϕ'_1 and ϕ'_2, then either ϕ'_1 or ϕ'_2 is* **L***-satisfiable.*

Proof. We show some representative cases. The omitted ones are analogous. In each case, let (U, Q, S) be the frame image associated with ϕ, and let $M = (W, R, V)$ and I be the **L**-model and interpretation that satisfy it.

1. Suppose ϕ is α_1-reducible to $\phi \cup \{z\}$, z has parent $z' \in \phi$, and $(\mathrm{lab}(z'), \mathrm{sgn}(z'))$ has the form $(B \wedge C, 1)$. Since $(\sigma^*(z'), z') \in S$, we know that $M, I(\sigma^*(z')) \Vdash \mathrm{lab}(z')$, or $M, I(\sigma^*(z')) \Vdash B \wedge C$. This means that $M, I(\sigma^*(z')) \Vdash B$, or $M, I(\sigma^*(z')) \Vdash \mathrm{lab}(z)$. Since z is neither a π- nor a ν-node (its secondary type is α_1), $\sigma^*(z) = \sigma^*(z')$, so $M, I(\sigma^*(z)) \Vdash \mathrm{lab}(z)$. In other words, since $\mathrm{sgn}(z) = 1$, M and I satisfy $(U, Q, S \cup \{(\sigma^*(z), z)\})$, which is the frame image associated with $\phi \cup \{z\}$.

2. Suppose ϕ is β-reducible to $\phi \cup \{z_1\}$ and $\phi \cup \{z_2\}$, z_1 and z_2 have parent $z' \in \phi$, and $(\mathrm{lab}(z'), \mathrm{sgn}(z'))$ has the form $(B \supset C, 1)$. Since $(\sigma^*(z'), z') \in S$, we know that $M, I(\sigma^*(z')) \Vdash \mathrm{lab}(z')$, or $M, I(\sigma^*(z')) \Vdash B \supset C$. This means that either $M, I(\sigma^*(z')) \nVdash B$ or $M, I(\sigma^*(z')) \Vdash C$, or either $M, I(\sigma^*(z')) \nVdash \mathrm{lab}(z_1)$ or $M, I(\sigma^*(z')) \Vdash \mathrm{lab}(z_2)$. Since z_1 and z_2 are neither π- nor ν-nodes (their secondary types are β_1 and β_2, respectively), $\sigma^*(z_1) = \sigma^*(z_2) = \sigma^*(z')$, so either $M, I(\sigma^*(z_1)) \nVdash \mathrm{lab}(z_1)$ or $M, I(\sigma^*(z_2)) \Vdash \mathrm{lab}(z_2)$. In other words, since $\mathrm{sgn}(z_1) = 0$ and $\mathrm{sgn}(z_2) = 1$, M and I satisfy $(U, Q, S \cup \{(\sigma^*(z_1), z_1)\})$ or $(U, Q, S \cup \{(\sigma^*(z_2), z_2)\})$, which are the frame images associated with $\phi \cup \{z_1\}$ and $\phi \cup \{z_2\}$, respectively.

3. Suppose ϕ is ν-reducible to $\phi \cup \{z\}$, z has parent $z' \in \phi$, and $(\mathrm{lab}(z'), \mathrm{sgn}(z'))$ has the form $(\Box_i B, 1)$. Since $(\sigma^*(z'), z') \in S$, we know that $M, I(\sigma^*(z')) \Vdash \mathrm{lab}(z')$, or $M, I(\sigma^*(z')) \Vdash \Box_i B$. This means that for every $w \in W$ such that $(i, I(\sigma^*(z')), w) \in R$, $M, w \Vdash B$.

   Since Ω is **L**-admissible, $(i, \sigma^*(z'), \sigma^*(z)) \in \mathrm{cl}_{\mathbf{L}}(Q')$, where Q' is the accessibility

relation of the frame (U', Q') associated with the set of nodes of Z that are $\lhd$-smaller than z. Since ϕ is ν-reducible to $\phi \cup \{z\}$, ϕ contains all nodes of Z that are $\lhd$-smaller than z, so as we observed in the previous section, $U' \subseteq U$ and $Q' \subseteq Q$. By lemma 2.3.1, $(i, \sigma^*(z'), \sigma^*(z)) \in \mathrm{cl}_{\mathbf{L}}(Q')$ implies that $(i, \sigma^*(z'), \sigma^*(z)) \in \mathrm{cl}_{\mathbf{L}}(Q)$, and by lemma 2.3.2, $(i, I(\sigma^*(z')), I(\sigma^*(z))) \in R$.

So $M, I(\sigma^*(z))) \Vdash B$, or $M, I(\sigma^*(z)) \Vdash \mathrm{lab}(z)$. In other words, since $\mathrm{sgn}(z) = 1$, M and I satisfy $(U, Q, S \cup \{(\sigma^*(z), z)\})$, which is the frame image associated with $\phi \cup \{z\}$.

4. Suppose ϕ is π-reducible to $\phi \cup \{z\}$, z has parent $z' \in \phi$, and $(\mathrm{lab}(z'), \mathrm{sgn}(z'))$ has the form $(\Diamond_i B, 1)$. Since $(\sigma^*(z'), z') \in S$, we know that $M, I(\sigma^*(z')) \Vdash \mathrm{lab}(z')$, or $M, I(\sigma^*(z')) \Vdash \Diamond_i B$. This means that there is a $w \in W$ such that $(i, I(\sigma^*(z)), w) \in R$ and $M, w \Vdash B$.

   $z \notin \phi$, so $z \notin U$. This means that I is not defined on z. Let I' be an interpretation from $U \cup \{z\}$ to W obtained by extending I by defining $I'(z) = w$. Since z is a π-node, $\sigma^*(z) = z$, so $M, I'(\sigma^*(z)) \Vdash B$, or $M, I'(\sigma^*(z)) \Vdash \mathrm{lab}(z)$. In other words, M and I' satisfy $(U \cup \{z\}, Q \cup \{(i, \sigma^*(z'), z)\}, S \cup \{(\sigma^*(z), z)\})$, which is the frame image associated with $\phi \cup \{z\}$. □

An immediate consequence of this lemma is that if the Ω-path $\{a_1^i\}$ is **L**-satisfiable, then there is an irreducible Ω-path through Z which is **L**-satisfiable, since every path reduction preserves **L**-satisfiability for at least one of the obtained offspring paths.

We can now prove theorem 4.2.1 by contradiction. Suppose A is not **L**-valid. Then there is an **L**-model $M = (W, R, V)$ and a $w \in W$ such that $M, w \nVdash A$. This means that the Ω-path $\{a_1^i\}$ is **L**-satisfied by M and any interpretation I that maps a_1^i to w. By lemma 4.2.1, this means that there is at least one **L**-satisfiable irreducible Ω-path ϕ through Z.

Since ϕ, like every irreducible Ω-path through Z, is assumed to be atomically

closed, there are nodes $z, z' \in \phi$ such that $\mathrm{lab}(z) = \mathrm{lab}(z')$, $\mathrm{sgn}(z) \neq \mathrm{sgn}(z')$, and $\sigma^*(z) = \sigma^*(z')$. If $\mathrm{sgn}(z) = 1$ and $\mathrm{sgn}(z') = 0$ (the other case is analogous), then $M, I(\sigma^*(z)) \Vdash \mathrm{lab}(z)$ and $M, I(\sigma^*(z') \nVdash \mathrm{lab}(z')$. Since $\mathrm{lab}(z) = \mathrm{lab}(z')$ and $\sigma^*(z) = \sigma^*(z')$, this is a contradiction, completing the proof of the soundness of the path characterization.

4.3 Completeness

This section consists of a proof of the completeness of the path characterization, stated as follows.

Theorem 4.3.1 (Completeness of the path characterization)**.** *If a formula A is* **L***-valid, then there is a multiplicity μ for the signed formula $X = (A, 0)$ and an* **L***-admissible realization Ω for the indexed signed formula $Z = (X, \mu)$ such that every irreducible Ω-path through Z is atomically closed.*

To prove this result, we will use the completeness of the tableau characterization from section 3.3. By theorem 3.3.1, if A is **L**-valid, then there is an atomically closed **L**-tableau for $X = (A, 0)$. Starting with this **L**-tableau, we will carry out the following steps.

1. Extract a multiplicity μ for X from the tableau.
2. Extract a realization Ω for $Z = (X, \mu)$ from the tableau.
3. Show that Ω is **L**-admissible.
4. Show that every irreducible Ω-path through Z is atomically closed.

Throughout the proof, we will show concrete examples in which we perform the above steps on the atomically closed **K**-tableau for $\Box_1(p_1 \supset p_2) \supset \Box_1 p_1 \supset \Box_1 p_2$ shown in figure 3.2 (example 3.1.1).

The first step, extracting a multiplicity for X, is quite straightforward. In particular, let μ be a partial function from pairs of the form (a, ι) to nonnegative integers, where a is a ν-node of X, and ι is a sequence of positive integers, defined as follows.

1. If a is a ν-node of X with no ν-ancestors, then let $\mu(a, \cdot)$ be the largest positive integer l such that a^l occurs in the tableau, or 0 if there is no such occurrence. Recall that a^ι *occurs in* a tableau if the tableau contains a frame image (U, Q, S) such that S contains a pair of the form (u, a^ι).

2. If a is a ν-node of X with a nearest ν-ancestor a', and a'^ι occurs in the tableau, then let $\mu(a, \iota)$ be the largest positive integer l such that $a^{\iota l}$ occurs in the tableau, or 0 if there is no such occurrence.

We must of course show that μ, defined in this way, is indeed a multiplicity. That the first clause of definition 4.1.1 holds is easy to see. As for the second, observe that if a is a ν-node with a nearest ν-ancestor a', and μ is defined on (a', ι), then $a'^{\iota 1}$, $a'^{\iota 2}$, ..., and $a'^{\iota l'}$ occur in the tableau, where $l' = \mu(a', \iota)$. This follows from the construction of μ and from the fact that if $a^{\iota l'}$ occurs in the tableau, then for every $l \in \{1, 2, \ldots, l' - 1\}$, so does $a^{\iota l}$. Next, from the second clause of the above construction of μ, it follows that μ is defined on $(a, \iota 1)$, $(a, \iota 2)$, ..., $(a, \iota\mu(a', \iota))$. In particular, for every $l \in \{1, 2, \ldots, \mu(a', \iota)\}$, $\mu(a, \iota l)$ is the largest positive integer l'' such that $a^{\iota l l''}$ occurs in the tableau.

This shows that μ is defined on every pair it needs to be defined on to be a multiplicity for Z. It is also not defined on any others. To see this, first notice that if a is a ν-node, and $a^{\iota' l}$ occurs in the tableau, then μ is defined on (a, ι'), and $\mu(a, \iota') \geq l$. This follows from a straightforward induction on the distance of a from the root (or on the length of ι'). Now suppose there were a pair (a, ι) that μ is defined on without needing to be, according to definition 4.1.1. If $\iota = \cdot$, then a has no ν-ancestors, since the second clause of the above definition of μ cannot account for definition on pairs

of the form $(a, \cdot)$, as expressions of the form $a^{\cdot}$, where a is a ν-node, cannot occur in a tableau. But every multiplicity must be defined on $(a, \cdot)$, so this definition of μ on $(a, \cdot)$ is not unwarranted. If $\iota = \iota' l$, then a has a nearest ν-ancestor a' such that a'^{ι} occurs in the tableau, meaning that by our previous observation, μ is defined on (a', ι'), and $\mu(a', \iota') \geq l$. By clause 2 of definition 4.1.1, every multiplicity must be defined on $(a, \iota' l)$ under these circumstances.

Example 4.3.1 (Extracting a multiplicity from a tableau)**.** Let $X = (\Box_1(p_1 \supset p_2) \supset \Box_1 p_1 \supset \Box_1 p_2, 0)$. The multiplicity μ for X extracted from the tableau shown in figure 3.2 (example 3.1.1) is defined on $(a_3, \cdot)$ and $(a_8, \cdot)$. In particular, $\mu(a_3, \cdot) = \mu(a_8, \cdot) = 1$.

Now that we have established that μ is a multiplicity for X, we consider the indexed signed formula (X, μ). μ was defined in a very particular way. The indexed node names that occur in tableaus are not induced by a multiplicity, but by the tableau rules. However, they implicitly describe a multiplicity, which we have now formally extracted. To show that our extraction of μ is correct, we will prove the following two multiplicity-correctness lemmas.

Lemma 4.3.1 (Completeness of μ)**.** *If a^{ι} occurs in the tableau, then a^{ι} is a node of (X, μ).*

Proof. By induction on the distance of a from the root of X.

1. If $a^{\iota} = a_1^{\cdot}$, then the result follows trivially.

2. If a is a ν-node, then by inspection of the tableau rules, a has a parent a', ι has the form $\iota' l$, and $a'^{\iota'}$ occurs in the tableau. By the induction hypothesis, $a'^{\iota'}$ is a node of (X, μ), and $\mu(a, \iota') \geq l$ (by the construction of μ and our previous observation), so by the definition of indexed node names, $a^{\iota' l} = a^{\iota}$ is a node of (X, μ).

3. If a is a nonroot non-ν-node, then a has a parent a', and a'^{ι} occurs in the tableau. By the induction hypothesis, a'^{ι} is a node of (X, μ), so by the definition of indexed node names, a^{ι} is a node of (X, μ). □

Lemma 4.3.2 (Soundness of μ)**.** *If a^{ι} is a ν-node of (X, μ), then a^{ι} occurs in the tableau.*

Proof. By induction on the distance of a from the root of X.

1. If a has no ν-ancestors, then $\iota = l$ for some positive integer l. For a^l to be a ν-node of (X, μ), $\mu(a, \cdot)$ must be greater than or equal to l. Because μ is obtained from the tableau, $a^{\mu(a,\cdot)}$ occurs in it, so by the tableau rules, so does a^l.

2. If a has a nearest ν-ancestor a', then ι has the form $\iota' l$, and $a'^{\iota'}$ is a ν-node of (X, μ). By the induction hypothesis, $a'^{\iota'}$ occurs in the tableau. Note that for a^{ι} to be a node of (X, μ), it must be the case that μ is defined on (a, ι'), and $\mu(a, \iota') \geq l$. Since $a'^{\iota'}$ occurs in the tableau and $\mu(a, \iota') \geq l$, $a^{\iota' l}$ must also occur in the tableau, since μ is defined by the tableau. □

Why does it matter that the indexed ν-nodes occurring in the tableau are exactly the ones of (X, μ)? Because this allows us to carry out the second step of our proof, namely extracting a realization Ω from the tableau. For every ν-node a^{ι} of (X, μ), a^{ι} occurs in the tableau. By the proviso on the (ν) rule, there is exactly one (ν) rule application which ν-decomposes some pair to one of the form (u, a^{ι}):

$$\frac{(U, Q, S \cup \{(u, a^{\iota})\})}{(U, Q, S)} \ (\nu)$$

Here, for every $u \in U$, u occurs in the tableau, making it a π-node of (X, μ) by lemma 4.3.1. Consequently, we may let $\sigma(a^{\iota}) = u$ and $\rho(a^{\iota}) = U$. We have shown that a^{ι} is a ν-node of (X, μ) and that every $u \in U$ is a π-node of (X, μ). We have also

shown that every ν-node of (X, μ) occurs in the tableau, meaning that $\Omega = (\rho, \sigma)$ is a properly defined realization for $Z = (X, \mu)$.

Example 4.3.2 (Extracting a realization from a tableau)**.** Let $X = (\Box_1(p_1 \supset p_2) \supset \Box_1 p_1 \supset \Box_1 p_2, 0)$, and let μ be the multiplicity for X extracted in example 4.3.1. The realization $\Omega = (\rho, \sigma)$ for $Z = (X, \mu)$ extracted from the tableau shown in figure 3.2 (example 3.1.1) is defined by $\sigma(a_3^1) = \sigma(a_8^1) = a_{10}^{\cdot}$ and $\rho(a_3^1) = \rho(a_8^1) = \{a_1^{\cdot}, a_{10}^{\cdot}\}$.

As the third step of our proof, we must show that Ω is **L**-admissible. Before doing so, however, we will make the relationship between (X, μ), Ω, and the frame images of our closed tableau more explicit. In particular, we will associate with every frame image of the tableau a set of nodes of $Z = (X, \mu)$ that respects $\lhd$, the reduction ordering of Z and Ω. The set associated with the root is $\{a_1^{\cdot}\}$, and for each tableau rule, we will describe how to obtain the set associated with a child frame image from the one associated with the parent frame image.

1. If the (α_1) rule is applied to a frame image (U, Q, S) associated with a node set ϕ to α_1-decompose (u, a^ι) to (u, b^ι), then associate $(U, Q, S \cup \{(u, b^\iota)\})$ with the node set $\phi \cup \{b^\iota\}$.

2. If the (α_2) rule is applied to a frame image (U, Q, S) associated with a node set ϕ to α_2-decompose (u, a^ι) to (u, b^ι), then associate $(U, Q, S \cup \{(u, b^\iota)\})$ with the node set $\phi \cup \{b^\iota\}$.

3. If the (β) rule is applied to a frame image (U, Q, S) associated with a node set ϕ to β-decompose (u, a^ι) to (u, b^ι) and (u, c^ι), then associate $(U, Q, S \cup \{(u, b^\iota)\})$ with the node set $\phi \cup \{b^\iota\}$ and $(U, Q, S \cup \{(u, c^\iota)\})$ with the node set $\phi \cup \{c^\iota\}$.

4. If the (ν) rule is applied to a frame image (U, Q, S) associated with a node set ϕ to ν-decompose (u, a^ι) to $(v, b^{\iota l})$, then associate $(U, Q, S \cup \{(v, b^{\iota l})\})$ with the node set $\phi \cup \{b^{\iota l}\}$.

5. If the (π) rule is applied to a frame image (U, Q, S) associated with a node set ϕ to π-decompose (u, a^ι) to (b^ι, b^ι), then associate $(U \cup \{b^\iota\}, Q \cup \{(i, u, b^\iota)\}, S \cup \{(b^\iota, b^\iota)\})$ with the node set $\phi \cup \{b^\iota\}$.

That every node set associated with a frame image does indeed respect $\lhd$ is shown by induction. Before proving it, though, notice that if a frame image (U, Q, S) of the tableau is associated with a node set ϕ, then U consists of exactly the π-nodes of ϕ. This is easy to prove by a straightforward induction on the distance of the frame image and associated node set from the root, but is a useful and important property of our frame-image-node-set association. It also allows us to prove that every node set associated with a frame image respects $\lhd$. It is obviously true of the root, and the remaining cases are fairly straightforward. We show a few representative ones:

1. If the (α_1) rule is applied to a frame image (U, Q, S) associated with a node set ϕ to α_1-decompose (u, a^ι) to (u, b^ι), then $(U, Q, S \cup \{(u, b^\iota)\})$ is associated with the node set $\phi \cup \{b^\iota\}$. By the induction hypothesis, ϕ respects $\lhd$, and every node $\lhd$-smaller than b^ι is also $\lhd$-smaller than a^ι (with the exception of a^ι itself, of course), so $\phi \cup \{b^\iota\}$ respects $\lhd$.

2. If the (ν) rule is applied to a frame image (U, Q, S) associated with a node set ϕ to ν-decompose (u, a^ι) to $(v, b^{\iota l})$, then $(U, Q, S \cup \{(v, b^{\iota l})\})$ is associated with the node set $\phi \cup \{b^{\iota l}\}$. By the induction hypothesis, ϕ respects $\lhd$, and if a node z of Z is $\lhd$-smaller than $b^{\iota l}$, then it is either $\lhd$-smaller than or equal to a^ι (in which case it is an element of ϕ), or $\sqsubset$-smaller than $b^{\iota l}$. If a node z is $\sqsubset$-smaller than $b^{\iota l}$, then $z \in \rho(b^{\iota l})$ by construction. Since we have defined $\rho(b^{\iota l})$ to be U, and ϕ contains the nodes of U, ϕ contains z. In other words, ϕ contains every node $\lhd$-smaller than $b^{\iota l}$, so $\phi \cup \{b^{\iota l}\}$ respects $\lhd$.

3. If the (π) rule is applied to a frame image (U, Q, S) associated with a node set ϕ to π-decompose (u, a^ι) to (b^ι, b^ι), then $(U \cup \{b^\iota\}, Q \cup \{(i, u, b^\iota)\}, S \cup \{(b^\iota, b^\iota)\})$ is

associated with the node set $\phi \cup \{b^\iota\}$. By the induction hypothesis, ϕ respects $\lhd$, and every node $\lhd$-smaller than b^ι is also $\lhd$-smaller than a^ι (with the exception of a^ι itself, of course), so $\phi \cup \{b^\iota\}$ respects $\lhd$.

This result has the interesting consequence that every node set associated with a frame image is, in fact, an Ω-path through Z. Before using this result, however, we can prove some further invariants concerning the association of sets of nodes with frame images. We have already seen the first clause of the following lemma, but we restate it for the sake of completeness.

Lemma 4.3.3. *If a tableau frame image (U, Q, S) is associated with a set ϕ of nodes of (X, μ), then*

1. $U = \{z : z \in \phi$ *is a* π*-node*$\}$,

2. $Q = \{(i, \sigma^*(z'), z) : z \in U$ *is a nonroot* π*-node with modality* i *and parent* $z'\}$, *and*

3. $S = \{(\sigma^*(z), z) : z \in \phi\}$.

Proof. By induction on the distance of the frame image from the root. In all but the base case, let (U', Q', S') be the parent frame image of (U, Q, S), and let ϕ' be the set of nodes of (X, μ) associated with (U', Q', S').

1. If $(U, Q, S) = (\{a_1^i\}, \varnothing, \{(a_1^i, a_1^i)\})$, then $\phi = \{a_1^i\}$, and the results are trivial.

2. (a) Suppose (U, Q, S) was obtained through an α_1-decomposition of (u, a^ι) to (u, b^ι), so $U = U'$, $Q = Q'$, $S = S' \cup \{(u, b^\iota)\}$, and $\phi = \phi' \cup \{b^\iota\}$.

   By the induction hypothesis, the invariants hold for U', Q', and S' (with respect to ϕ'). Since b^ι is not a π-node, they also immediately hold for U and Q. Since $S' = \{(\sigma^*(z), z) : z \in \phi'\}$ and $(u, a^\iota) \in S'$, $\sigma^*(a^\iota) = u$. Since

b^ι is neither a ν- nor a π-node, $\sigma^*(b^\iota) = \sigma^*(a^\iota) = u$. Consequently,

$$S = S' \cup \{(u, b^\iota)\} = \{(\sigma^*(z), z) : z \in \phi' \cup \{b^\iota\}\},$$

meaning that the invariant also holds for S.

(b) Suppose (U, Q, S) was obtained through a β-decomposition of (u, a^ι) to (u, b^ι) and (u, c^ι), so $U = U'$, $Q = Q'$, $S = S' \cup \{(u, b^\iota)\}$, and $\phi = \phi' \cup \{b^\iota\}$.

By the induction hypothesis, the invariants hold for U', Q', and S' (with respect to ϕ'). Since b^ι is not a π-node, they also immediately hold for U and Q. Since $S' = \{(\sigma^*(z), z) : z \in \phi'\}$ and $(u, a^\iota) \in S'$, $\sigma^*(a^\iota) = u$. Since b^ι is neither a ν- nor a π-node, $\sigma^*(b^\iota) = \sigma^*(a^\iota) = u$. Consequently,

$$S = S' \cup \{(u, b^\iota)\} = \{(\sigma^*(z), z) : z \in \phi' \cup \{b^\iota\}\},$$

meaning that the invariant also holds for S. This case covers one branch of the fork caused by the β-decomposition. An analogous argument is used to show that the invariants hold for the frame image $(U', Q', S' \cup \{(u, c^\iota)\})$ associated with the node set $\phi' \cup \{c^\iota\}$.

(c) Suppose (U, Q, S) was obtained through a ν-decomposition of (u, a^ι) to $(v, b^{\iota l})$, so $U = U'$, $Q = Q'$, $S = S' \cup \{(v, b^{\iota l})\}$, and $\phi = \phi' \cup \{b^{\iota l}\}$.

By the induction hypothesis, the invariants hold for U', Q', and S' (with respect to ϕ'). Since $b^{\iota l}$ is not a π-node, they also immediately hold for U and Q. Since $S' = \{(\sigma^*(z), z) : z \in \phi'\}$ and $(u, a^\iota) \in S'$, $\sigma^*(a^\iota) = u$. Since $b^{\iota l}$ is a ν-node, $\sigma^*(b^{\iota l}) = \sigma(b^{\iota l}) = v$ (recall that this this is how σ is defined on $b^{\iota l}$). Consequently,

$$S = S' \cup \{(v, b^{\iota l})\} = \{(\sigma^*(z), z) : z \in \phi' \cup \{b^{\iota l}\}\},$$

meaning that the invariant also holds for S.

(d) Suppose (U, Q, S) was obtained through a π-decomposition of (u, a^ι) to (b^ι, b^ι), so $U = U' \cup \{b^\iota\}$, $Q = Q' \cup \{(i, u, b^\iota)\}$, $S = S' \cup \{(b^\iota, b^\iota)\}$, and $\phi = \phi' \cup \{b^\iota\}$.

By the induction hypothesis, the invariants hold for U', Q', and S' (with respect to ϕ'). Since b^ι is a π-node,

$$U = U' \cup \{b^\iota\} = \{z : z \in \phi' \cup \{b^\iota\} \text{ is a } \pi\text{-node}\},$$

so the invariant holds for U. Also,

$$\begin{aligned} Q &= Q' \cup \{(i, u, b^\iota)\} \\ &= \{(i, \sigma^*(z'), z) : z \in U' \cup \{b^\iota\} \text{ is a nonroot} \\ &\qquad \pi\text{-node with modality } i \text{ and parent } z'\}, \end{aligned}$$

since the parent of b^ι is a^ι, and $\sigma^*(a^\iota) = u$ (we know this since the source of the π-decomposition is $(u, a^\iota) \in S'$), so the invariant holds for Q. That the modality of b^ι is i follows from the fact that the accessibility triple added to Q' by the π-decomposition is (i, u, b^ι), and the modality of this accessibility triple is determined by the modality of b^ι. Finally, since b^ι is a π-node, $\sigma^*(b^\iota) = b^\iota$, so

$$S = S' \cup \{(b^\iota, b^\iota)\} = \{(\sigma^*(z), z) : z \in \phi' \cup \{b^\iota\}\},$$

meaning that the invariant also holds for S. □

This is a rather striking result. We have shown that with μ and Ω extracted from a tableau, we can express the entire tableau in terms of sets of nodes of (X, μ), since the frame images can be reconstructed from these sets. Perhaps not surprisingly, the

next step in the proof of completeness, that Ω is **L**-admissible, follows fairly naturally. We have associated sets of nodes of Z with tableau frame images. Recall also that every ν-node of Z occurs in the tableau.

So for every ν-node of Z with modality i and parent z', let (U, Q, S) be the frame image at which it is introduced through a ν-decomposition. In other words, (U, Q, S) is a frame image of the tableau, and S contains a pair (u, a^ι) which is ν-decomposed into $(v, b^{\iota l})$. Let $z = b^{\iota l}$ and $z' = a^\iota$. By the proviso on the (ν) tableau rule, $(i, u, v) \in \mathrm{cl}_{\mathbf{L}}(Q)$. Let (U', Q') be the frame associated with the set of nodes of Z that are $\lhd$-smaller than z. We defined $\rho(z) = U$, so for every $u \in U$, $u \lhd z$. In other words, $U \subseteq U'$. By the previous lemma, Q is determined by U in exactly the same way that Q' is determined by U', so it is also the case that $Q \subseteq Q'$. So by corollary 2.3.1, we know that $(i, u, v) \in \mathrm{cl}_{\mathbf{L}}(Q)$ implies that $(i, u, v) \in \mathrm{cl}_{\mathbf{L}}(Q')$. Finally, by the invariant on S, we know that $\sigma^*(a^\iota) = u$ and $\sigma^*(b^{\iota l}) = v$, from which we obtain that $(i, \sigma^*(z'), \sigma^*(z)) \in \mathrm{cl}_{\mathbf{L}}(Q')$. This argument can be repeated for every ν-node of Z to establish the **L**-admissibility of Ω.

One step remains in the proof of theorem 4.3.1, namely that every irreducible Ω-path through Z is atomically closed. We have seen that every node set associated with a frame image is an Ω-path, and as the reader may have already noticed, if a frame image is atomically closed, then its associated Ω-path is also atomically closed. This follows immediately from the invariants established by the previous lemma. As a result, every Ω-path associated with a leaf of our tableau is atomically closed.

We require a slightly stronger result, however. We need to show that *every irreducible* Ω-path, regardless of whether or not it is associated with a leaf of our atomically closed tableau, is atomically closed. As it turns out, our result is already sufficiently strong. Consider the Ω-path $\{a_1^\cdot\}$ associated with the root of the tableau. It is a subset of every irreducible Ω-path through Z. Any nodes added to this Ω-path by α_1-, α_2-, ν-, or π-decompositions are contained in every irreducible Ω-path because

they are irreducible. When the tableau branches because of a β-decomposition, every irreducible Ω-path through Z contains, as a subset, the Ω-path associated with one of the child frame images of the fork. This continues, and the invariant is maintained that every irreducible Ω-path through Z contains, as a subset, one of the Ω-paths associated with a leaf of an intermediate tableau.

Formally, we will associate with every frame image (U, Q, S) of the tableau a set Φ of irreducible Ω-paths through Z, maintaining the invariant that if ϕ is the Ω-path associated with the frame image, then it is a subset of every irreducible Ω-path contained in Φ. We will prove this invariant by induction on the distance of (U, Q, S) from the root, for which it holds trivially (with Φ the set of all irreducible Ω-paths through Z).

1. If the (α_1) rule is applied to a frame image (U, Q, S) associated with ϕ and Φ to α_1-decompose (u, a^ι) to (u, b^ι), then $(U, Q, S \cup \{(u, b^\iota)\})$ is associated with $\phi \cup \{b^\iota\}$ and Φ. By induction, every $\phi' \in \Phi$ contains a^ι and, being irreducible, b^ι, so $\phi \cup \{b^\iota\} \subseteq \phi'$.

2. If the (α_2) rule is applied to a frame image (U, Q, S) associated with ϕ and Φ to α_2-decompose (u, a^ι) to (u, b^ι), then $(U, Q, S \cup \{(u, b^\iota)\})$ is associated with $\phi \cup \{b^\iota\}$ and Φ. By induction, every $\phi' \in \Phi$ contains a^ι and, being irreducible, b^ι, so $\phi \cup \{b^\iota\} \subseteq \phi'$.

3. If the (β) rule is applied to a frame image (U, Q, S) associated with ϕ and Φ to β-decompose (u, a^ι) to (u, b^ι) and (u, c^ι), then $(U, Q, S \cup \{(u, b^\iota)\})$ is associated with $\phi \cup \{b^\iota\}$ and Φ_1, while $(U, Q, S \cup \{(u, c^\iota)\})$ is associated with $\phi \cup \{c^\iota\}$ and Φ_2. Φ_1 is the subset of Φ whose members contain b^ι, while Φ_2 is the subset of Φ whose members contain c^ι. By induction, every path in Φ contains a^ι and is irreducible, so Φ_1 and Φ_2 partition Φ. Also, for every $\phi' \in \Phi_1$, $\phi \cup \{b^\iota\} \subseteq \phi'$, and for every $\phi' \in \Phi_2$, $\phi \cup \{c^\iota\} \subseteq \phi'$.

4. If the (ν) rule is applied to a frame image (U, Q, S) associated with ϕ and Φ to ν-decompose (u, a^ι) to $(v, b^{\iota l})$, then $(U, Q, S \cup \{(v, b^{\iota l})\})$ is associated with $\phi \cup \{b^{\iota l}\}$ and Φ. By induction, every $\phi' \in \Phi$ contains a^ι and, since ϕ' is irreducible and contains every node of Z $\lhd$-smaller than $b^{\iota l}$ (ϕ does, and by induction, $\phi \subseteq \phi'$), $b^{\iota l}$, so $\phi \cup \{b^{\iota l}\} \subseteq \phi'$.

5. If the (π) rule is applied to a frame image (U, Q, S) associated with ϕ and Φ to π-decompose (u, a^ι) to (b^ι, b^ι), then $(U \cup \{b^\iota\}, Q \cup \{(i, u, b^\iota)\}, S \cup \{(b^\iota, b^\iota)\})$ is associated with $\phi \cup \{b^\iota\}$ and Φ. By induction, every $\phi' \in \Phi$ contains a^ι and, being irreducible, b^ι, so $\phi \cup \{b^\iota\} \subseteq \phi'$.

Every irreducible Ω-path through Z is associated with the root, and none are lost as we traverse the tableau towards the leaves. Consequently, every irreducible Ω-path through Z is associated with a leaf of the tableau, meaning that each one contains, as a subset, an atomically closed Ω-path. Therefore, every irreducible Ω-path through Z is atomically closed, completing the proof of the completeness of the path characterization.

Example 4.3.3 (Associating Ω-paths with frame images)**.** Let $X = (\Box_1(p_1 \supset p_2) \supset \Box_1 p_1 \supset \Box_1 p_2, 0)$, let μ be the multiplicity for X extracted in example 4.3.1, and let Ω be the realization for $Z = (X, \mu)$ extracted in example 4.3.2. The Ω-paths associated with the leaves of the **K**-tableau shown in figure 3.2 (example 3.1.1) are

$$\{a_1^{\cdot}, a_2^{\cdot}, a_3^1, a_4^1, a_6^{\cdot}, a_7^{\cdot}, a_8^1, a_9^{\cdot}, a_{10}^{\cdot}\},$$

atomically closed by a_4^1 and a_8^1, and

$$\{a_1^{\cdot}, a_2^{\cdot}, a_3^1, a_5^1, a_6^{\cdot}, a_9^{\cdot}, a_{10}^{\cdot}\},$$

atomically closed by a_5^1 and $a_{10}^{\cdot}$. The irreducible Ω-paths through Z are

$$\{a_1^{\cdot}, a_2^{\cdot}, a_3^1, a_4^1, a_6^{\cdot}, a_7^{\cdot}, a_8^1, a_9^{\cdot}, a_{10}^{\cdot}\},$$

which contains the first atomically closed Ω-path mentioned above as a subset, and

$$\{a_1^{\cdot}, a_2^{\cdot}, a_3^1, a_5^1, a_6^{\cdot}, a_7^{\cdot}, a_8^1, a_9^{\cdot}, a_{10}^{\cdot}\},$$

which contains the second atomically closed Ω-path mentioned above as a subset, so all irreducible Ω-paths through Z are atomically closed.

Chapter 5

Conclusion

In this thesis, we have developed two proof formalisms and characterizations of validity for propositional multimodal logics with interacting accessibility relations. Our systems are more semantically motivated than the prefixed tableau calculi of Fitting [16] and the matrix characterizations of Wallen [48], and while our tableau calculus is similar in spirit to those of Nerode [39] and Baldoni [4], it is more specifically designed to allow us to separate some of the structural and modal choices made in a derivation. The proofs of correctness of the path characterization reveal, however, that our tableau and path formalisms are quite alike. This is perhaps not surprising, since the path characterization is, in some ways, merely a different way of presenting and understanding the information that exists in our tableau proofs. This, of course, is by design. We have attempted to emphasize this by pointing out correspondences between the two systems as they were developed.

How is the path characterization significantly different from the tableau characterization, then? A key selling point is that tableau derivations that are slightly different as a result of transposed decompositions of the structure of the query formula correspond to the same proof in the path characterization, since the structural decomposition of the query formula is concealed in the identities of the paths through

it.

In other words, multiplicities and realizations contain all the modal decisions that are made in a corresponding tableau derivation, while the structural decisions disappear completely. They are determined by whether or not the paths through the query formula are atomically closed. It is worth pointing out that since the identities of the paths are dependent on the chosen multiplicity and realization, the separation between modal and structural concerns is not absolute. However, while the concerns remain subtly related, all disjunctive nondeterminism disappears from the structural concerns.

A further important difference between both our formalisms and previous work is that in our systems, the properties of the logics of interest are extracted almost entirely from the definitions of the formalisms. They are present only in the form of closures that need to be computed. As a consequence, our formalisms can easily be extended to logics for whose frame conditions closures can be computed. In this chapter, we will tie up some loose ends while discussing possible future work of this kind.

5.1 Future work

Other frame conditions The frame conditions we defined in section 2.1 are multimodal generalizations of some of the most well-known unimodal frame conditions. However, in chapters 3 and 4 it became clear that our characterizations are defined in a way that is very independent of the logics under consideration; they only make appearances in the form of closures of frame conditions. This means that our approach extends easily to other frame conditions that are easy to close. If R is an accessibility relation on a set W, then we might call it, for instance,

1. i, j-*subsuming* if $(i, w, x) \in R$ implies that $(j, w, x) \in R$,

2. i, j, k-*subsuming* if $(i, w, x) \in R$ and $(j, w, x) \in R$ together imply that $(k, w, x) \in R$,

3. i, j, k-*reverse-transitive* if $(i, w, x) \in R$ and $(j, x, y) \in R$ together imply that $(k, y, w) \in R$, and

4. i, j, k, l-*transitive* if $(i, w, x) \in R$, $(j, x, y) \in R$, and $(k, y, z) \in R$ together imply that $(l, w, z) \in R$.

These frame conditions, along with all of the ones defined in section 2.1, are called *propagation frame conditions*, meaning that they imply further accessibilities amongst existing worlds. They do not postulate the existence of further, hitherto unreferenced worlds. Frame conditions that do, called *structural frame conditions*, include, for instance,

1. i-*seriality*, meaning that for every $w \in W$, there is an $x \in W$ such that $(i, w, x) \in R$,

2. i, j, k-*denseness*, meaning that $(i, w, y) \in R$ implies that there is an $x \in W$ such that $(j, w, x) \in R$ and $(k, x, y) \in R$, and

3. i, j, k, l-*confluence*, meaning that $(i, w, x) \in R$ and $(j, w, y) \in R$ together imply that there is a $z \in W$ such that $(k, x, z) \in R$ and $(l, y, z) \in R$.

This distinction between propagation and structural frame conditions (the terminology is due to Castilho et al. [10]) is important. Adding support for structural frame conditions to a tableau characterization is relatively straightforward, as it is typically sufficient to add a new tableau rule for each type of structural frame condition. However, this means that the nature of the logic under consideration is no longer confined to computing its closures: properties of the logic have spilled into custom tableau rules. This means that the translation between the tableau and path characterizations would no longer be as direct, since thrown into the tableau formalism's already

mixed bag of structural and modal rules is a different class of modal rules implementing the structural frame conditions. An additional layer of abstraction between the tableau and path characterizations thus becomes necessary.

This is one significant manifestation of the differences between our characterization and Wallen's. Wallen's framework for unimodal logic supports seriality, while ours, without modification, does not. On the other hand, our framework supports symmetry and other arbitrary propagation conditions for unimodal (and multimodal) logics, while Wallen's does not. The approaches are somewhat complementary, which is perhaps to be expected, considering that our approach is more semantically motivated, while Wallen's is essentially syntactic. Wallen's goal is, after all, efficient proof search for unimodal logics, while ours is a deeper proof theoretic understanding of multimodal logical validity.

Proof search Proof search in Wallen's system—and modal connection-based theorem proving in general—is typically performed using string unification on prefixes [48]. Our systems have no prefixes, so the efficient proof search mechanisms developed by Wallen do not have an immediate analogue in our framework. A naive proof search procedure in the path characterization would first have to choose or construct a multiplicity and then a realization, and finally test whether or not all paths through the query indexed signed formula are atomically closed.

Proof search in this kind of proof space is bound to be quite different from the incremental approach used in theorem provers based on tableau calculi, and it is clear that the disjunctive choices that must be made in exploring this proof space are essentially modal. Structural choices, i.e., choices related to how and in what order to decompose parts of the query formula, disappear, replaced by the test of atomic closure for all paths. Although the paths depend on the chosen realization, no disjunctive choices need to be made to determine what they are and whether or not they are closed.

Intuitionistic modal logics As we emphasized in section 2.1, the foundations of our logics are strictly classical. The idea of intuitionistic modal logic (IML) is still relatively young, and a number of logics have been explored and proposed to capture the essence of modalities in a constructive or intuitionistic setting (see for instance Simpson [43], Bierman and de Paiva [7], and Pfenning and Davies [40]).

Our work was partly inspired by the interpretation of intuitionistic unimodal logic as a fragment of classical bimodal logic, i.e., a classical modal logic with two modalities—one "intuitionistic" and one "modal" in a more conventional sense—interacting in particular ways (see for instance Wolter and Zakharyaschev [49] or Alechina et al. [1]). Which interaction axioms are necessary to correctly model intuitionism is debatable, but given a translation from IML to classical bimodal logic, our path characterization also works, in principle, for IML, provided that our system supports the required frame conditions and interactions.

In theory, making matrix or path characterizations work for various proposals or variants of IML is "only" a question of being able to define the correct frame conditions, so understanding frame conditions proof theoretically and designing frameworks in which frame conditions can be treated modularly and abstractly seems very desirable. This, along with the sheer number of ways in which frame conditions can be combined, was the motivation behind designing our systems to be as independent of frame conditions as possible, requiring only the ability to compute closures. Of course, computing closures can be an expensive operation in itself, but this is an issue of implementation.

Different modal languages The richness of modal languages themselves suggests several interesting extensions.

Our multimodal logics feature strictly unary modalities, but the semantics of multimodal logics extend easily to modalities of arbitrary arity. The binary relation implicitly associated with each modality is simply replaced by a relation of higher

arity. This raises the question of how these richer logics would behave in a path characterization. In general, proof formalisms for extensions to the language of multimodal logic do not seem to be thoroughly explored, perhaps because of the lack of convincing applications for them.

A more convincing case might be made for first-order modal logic, and most treatments of modal logic do indeed consider the first-order extension. The combination of modal operators and first-order quantifiers introduces several semantic considerations (see Fitting and Mendelsohn [19] for a discussion of the issues) which must naturally be reflected in any proof theoretic formalism. We have restricted ourselves to propositional multimodal logics in this thesis, but for practical modelling, first-order multimodal logics remain of great interest.

We briefly mentioned description logics (DLs) in chapter 1 and pointed out that a simple description logic can be thought of as a notational variant of a multimodal logic. The vast majority of description logics in use, however, are more expressive. A question that presents itself is thus what sorts of alternative proof theoretic formalisms can be developed for description logics (see for instance Clément [12] for recent proof theoretic work on constructive DLs) and whether or not reasoning algorithms for DLs can be developed on the basis of matrix or path characterizations designed specifically for them.

Bibliography

[1] Natasha Alechina, Michael Mendler, Valeria de Paiva, and Eike Ritter. Categorical and Kripke semantics for constructive S4 modal logic. In *Proceedings of the 15th International Workshop on Computer Science Logic*, pages 292–307. Springer-Verlag, 2001.

[2] Peter Andrews. Theorem proving via general matings. *Journal of the ACM*, 28(2):193–214, 1981.

[3] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[4] Matteo Baldoni. Normal multimodal logics with interaction axioms: A tableau calculus and some (un)decidability results. In *Labelled Deduction*, pages 33–57. Kluwer Academic Publishers, 2000.

[5] Evert W. Beth. *The Foundations of Mathematics: A Study in the Philosophy of Science*. North-Holland, 1959.

[6] Wolfgang Bibel. On matrices with connections. *Journal of the ACM*, 28(4):633–645, 1981.

[7] Gavin M. Bierman and Valeria de Paiva. On an intuitionistic modal logic. *Studia Logica*, 65(3):383–416, 2000.

[8] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge University Press, 2001.

[9] Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[10] Marcos A. Castilho, Luis Fariñas del Cerro, Olivier Gasquet, and Andreas Herzig. Modal tableaux with propagation rules and structural rules. *Fundamenta Informaticae*, 32(3–4):281–297, 1997.

[11] Brian F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980.

[12] Ian P. Clément. Proof theoretical foundations for constructive description logic. Master's thesis, McGill University, 2008.

[13] Rowan Davies and Frank Pfenning. A modal analysis of staged computation. *Journal of the ACM*, 48(3):555–604, 2001.

[14] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[15] Matt Fairtlough and Michael Mendler. Propositional lax logic. *Information and Computation*, 137(1):1–33, 1997.

[16] Melvin Fitting. Tableau methods of proof for modal logics. *Notre Dame Journal of Formal Logic*, 13(2):237–247, 1972.

[17] Melvin Fitting. *Proof Methods for Modal and Intuitionistic Logics*. D. Reidel, 1983.

[18] Melvin Fitting. First-order modal tableaux. *Journal of Automated Reasoning*, 4(2):191–213, 1988.

[19] Melvin Fitting and Richard L. Mendelsohn. *First-Order Modal Logic*. Kluwer Academic Publishers, 1998.

[20] Gerhard Gentzen. Untersuchungen über das logische schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.

[21] Rajeev Goré. Tableau methods for modal and temporal logics. In *Handbook of Tableau Methods*, pages 297–396. Kluwer Academic Publishers, 1999.

[22] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT Press, 2000.

[23] K. Jaakko J. Hintikka. Form and content in quantification theory. *Acta Philosophica Fennica*, 8:7–55, 1955.

[24] K. Jaakko J. Hintikka. *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. Cornell University Press, 1962.

[25] G. E. Hughes and M. J. Cresswell. *A New Introduction to Modal Logic*. Routledge, 1996.

[26] Stig Kanger. *Provability in Logic*. Almqvist and Wiksell, 1957.

[27] Christoph Kreitz and Heiko Mantel. A matrix characterization for multiplicative exponential linear logic. *Journal of Automated Reasoning*, 32(2):121–166, 2004.

[28] Christoph Kreitz, Heiko Mantel, Jens Otten, and Stephan Schmitt. Connection-based proof construction in linear logic. In *Proceedings of the 14th International Conference on Automated Deduction*, pages 207–221. Springer-Verlag, 1997.

[29] Christoph Kreitz and Stephan Schmitt. A uniform procedure for converting matrix proofs into sequent-style systems. *Information and Computation*, 162(1–2):226–254, 2000.

[30] Saul Kripke. A completeness theorem in modal logic. *Journal of Symbolic Logic*, 24(1):1–14, 1959.

[31] Saul Kripke. Semantic analysis of modal logic I: Normal propositional calculi. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963.

[32] E. J. Lemmon and Dana Scott. *The 'Lemmon Notes': An Introduction to Modal Logic*. Blackwell, 1977.

[33] Clarence Irving Lewis. *A Survey of Symbolic Logic*. University of California Press, 1918.

[34] Hugh MacColl. *Symbolic Logic and Its Applications*. Longmans, Green, and Co., 1906.

[35] Fabio Massacci. Strongly analytic tableaux for normal modal logics. In *Proceedings of the 12th International Conference on Automated Deduction*, pages 723–737. Springer-Verlag, 1994.

[36] Kenji Miyamoto and Atsushi Igarashi. A modal foundation for secure information flow. In *Proceedings of the Workshop on Foundations of Computer Security*, pages 187–203. Turku Centre for Computer Science, 2004.

[37] Jonathan Moody. Modal logic as a basis for distributed computation. Technical Report CMU-CS-03-194, Carnegie Mellon University, 2003.

[38] Tom Murphy VII, Karl Crary, Robert Harper, and Frank Pfenning. A symmetric modal lambda calculus for distributed computing. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science*, pages 286–295. IEEE Computer Society, 2004.

[39] Anil Nerode. Some lectures on modal logic. In *Logic, Algebra, and Computation*, pages 281–334. Springer-Verlag, 1991.

[40] Frank Pfenning and Rowan Davies. A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11(4):511–540, 2001.

[41] Dag Prawitz. An improved proof procedure. *Theoria*, 26(2):102–139, 1960.

[42] Dag Prawitz. *Natural Deduction: A Proof-Theoretical Study*. Almqvist and Wiksell, 1965.

[43] Alex K. Simpson. *The Proof Theory and Semantics of Intuitionistic Modal Logic*. PhD thesis, University of Edinburgh, 1994.

[44] Raymond M. Smullyan. *First-Order Logic*. Springer-Verlag, 1968.

[45] Yde Venema. Temporal logic. In *The Blackwell Guide to Philosophical Logic*, pages 203–223. Blackwell, 2001.

[46] Georg Henrik von Wright. *An Essay on Modal Logic*. North-Holland, 1951.

[47] Lincoln A. Wallen. Matrix proof methods for modal logics. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 917–923. Morgan Kaufmann, 1987.

[48] Lincoln A. Wallen. *Automated Proof Search in Non-Classical Logics: Efficient Matrix Proof Methods for Modal and Intuitionistic Logics*. MIT Press, 1990.

[49] Frank Wolter and Michael Zakharyaschev. Intuitionistic modal logics as fragments of classical bimodal logics. In *Logic at Work*, pages 168–186. Springer-Verlag, 1998.

[50] J. Jay Zeman. *Modal Logic: The Lewis-Modal Systems*. Clarendon Press, 1973.