Tableau-based Theorem Proving: A Unified View

Integrating and Unifying Methods of Tableau-based Theorem Proving

Bernhard Beckert

Dissertation
(PhD Thesis)

Final Version
May 29, 1998

Universität Karlsruhe (TH)
Fakultät für Informatik
Institut für Logik, Komplexität und Deduktionssysteme

The invention of tableau systems will continue, simply because they are easier to think of than other formulations.

Melvin Fitting (1998)

Preface

Background

This is the English version of my PhD thesis submitted (in German) to the Department of Computer Science, University of Karlsruhe, in June 1998.

Publication

Some of the results presented in this thesis have already been published as journal articles or in conference proceedings:

(Beckert & Goré, 1997) – a mixed variable tableau calculus for the basic modal logics (Section 4.3.7).

(Beckert & Hartmer, 1998) – tableau calculi for the fragments MLSS and MLSSF of set theory (Section 3.8).

(Beckert & Gabbay, 1998) – fibring tableau calculi (Chapter 6).

Acknowledgements

Many people encouraged me and helped me in some way or the other while I worked on this thesis; without their support it could not have been done. My sincerest thanks go to:

- Prof. Peter H. Schmitt for his excellent advise, his support, and for giving me the opportunity to work in his group with its friendly and productive atmosphere;
- Prof. Jacques Calmet for agreeing to be co-referee of my thesis and for fruitful discussions;
- my colleagues and friends Wolfgang Ahrendt, Reiner Hähnle, Christian Pape, and Joachim Posegga for all their support and many stimulating discussions;
- Prof. Dov Gabbay for giving me the opportunity to visit his group at Imperial College, for sharing with me parts of his huge knowlege in all areas of logic, and for suggesting to apply the fibring technique to tableau calculi;
- Rajeev Goré, as to work with him is always productive—even when he is on the other side of the planet;

– Ulrike Hartmer, who worked out the details of the calculi for fragments of set theory described in this thesis;
– all the many other people who gave me some advise or hints, including Prof. Domenico Cantone, Uwe Egly, Guido Governatori, and Gernot Stenz;
– the Department of Computer Science of the University of Karlsruhe for providing a good work environment;
– the Deutsche Forschungsgemeinschaft (DFG), which—as part of its Schwerpunktprogramm *Deduktion*—financed the projects that I worked on for the last years;
– the Department of Computing at Imperial College in London, for inviting me—my visit there was very profitable and productive;
– the European Union, for financing my visit at Imperial College as part of its *Training and Mobility of Researchers* (TMR) program and my participation in several workshops and conferences as part of the COST Action *Many-valued Logics*;
– and especially my parents for their support and encouragement over the years.

Karlsruhe, Mai 1998

Bernhard Beckert

Note. In the following, I often use the pronoun "we"; that is not a *pluralis majestatis* but it refers to the two of us: myself and *you*, the reader.

Contents

List of Tables

List of Figures

1 Introduction

Logicians have but ill defined
As rational the human kind.
Logic, they say, belongs to man,
But let them prove it if they can.

— OLIVER GOLDSMITH

1.1 Motivation

Automated Deduction—and in particular tableau-based theorem proving—has reached a state where it is on the verge of being successfully used in real-world applications. Up to now, however, the transition from a technique mainly used in research towards a tool that is routinely used in practice has not been accomplished.

One obstacle is that much work is put into developing very sophisticated and powerful theorem proving systems for certain logics (in particular first-order predicate logic), which are not tailor-made for certain applications. To use these systems, problems from the particular area of application have to be transformed into the logic supported by the prover.

A different and possibly more successful approach is to construct a specialised, dedicated proof system for a certain application supporting a logic that is particularly well suited for the application and that makes use of the special features of both the logic and the application to increase efficiency.

This requires the availability of uniform methods for the construction of efficient dedicated proof procedures for different logics, such that they do not have to be developed from scratch. There is, however, a gap between the two main areas into which most knowledge in the field of Automated Deduction can be separated: on the one hand, many different calculi for many different logics have been presented; these, however, are typically only of theoretical interest and not suitable for an implementation. On the other hand, proof procedures for a few important logics—predominantly classical propositional and first-order predicate logic—have been described including a huge number of special techniques and refinements to make them more efficient.

The objective of this thesis is to close that gap: To reach this goal and to facilitate the uniform construction of efficient, dedicated tableau-based theorem provers, three approaches are pursued, which complement each other:

1. the generalisation and uniform description of tableau calculi, of their refinements, and of methods for their improvement;

2. the development of uniform methods for constructing efficient deterministic proof procedures from (non-deterministic) tableau calculi;

3. the (uniform) integration of different tableau calculi and of tableau calculi and special methods for solving problems from certain domains (theory reasoning).

The first two approaches apply to all kinds of logics, whereas the third one applies mainly to logics that are extensions of classical logic (e.g., many-valued, modal and temporal logics) but only in part to logics that can be seen as restrictions of classical logics, such as linear and relevance logic.

The uniform methods presented in the following simplify the design of efficient tableau-based proof procedures. Since many of the pre-conditions for the applicability of these methods are purely syntactical and easy to check, one can—as a future goal—imagine an (at least partially) automated *meta* deduction system that, when presented with the definition of a tableau calculus, applies uniform techniques to improve the calculus and construct an efficient proof procedure, or that even constructs a tableau calculus for a given logic from scratch.

1.2 Main Results and Structure of this Thesis

Chapter 1 In this introductory chapter, the importance of methods for the uniform design of efficient tableau calculi is discussed and motivated; the main results and the structure of the thesis are described; the notion of tableau calculi is introduced; the main properties of tableau calculi and the differences to other types of calculi are explained; and, finally, a short overview of the history of tableau calculi is given.

Chapter 2 Logical systems (or logics for short) are defined in a very general way (as few restrictions as possible are made regarding syntax and semantics). The notions of terms and, based on that, substitutions and unification of terms are introduced. As examples for the description of logics within the general framework, first-order predicate logic, modal logics, and two fragments of quantifier-free set theory are presented.

Chapter 3 A uniform description of tableau calculi and their syntactical and semantical properties is given—as, for example, being analytic, monotonic, saturating, respectively being sound, complete, proof confluent—without any restriction to certain logics or normal forms. General criteria are presented for checking whether a calculus has these properties (including criteria for checking soundness and completeness). An important class of "well-behaved" tableau calculi is identified, which are called *ideal* calculi. This class turns out to be of great importance as (a) idealness is a pre-condition for the applicability of many of the uniform methods for improving the efficiency of tableau calculi described in the following, and (b) ideal calculi exist for most logics. As examples, ideal calculi for first-order predicate logic and for modal logics are presented. In addition, ideal tableau calculi are defined for the fragments of set theory introduced in Chapter 2; these calculi are shown to be more efficient than the calculi that were previously described in the literature.

Chapter 4 Techniques are presented for improving a tableau calculus for automated deduction in such a way that proof procedures based on it are more efficient. In particular, methods that have turned out to be useful for tableau-based deduction in first-order predicate logic are generalised, such that they can be used in the design of tableau-based proof procedures for arbitrary logics—including (a) the concept of rigid variables that represent terms and can be instantiated "on demand" during proof search, (b) the universal variable technique, where variables represent *all* terms simultaneously, and (c) a combination of both. The relation of rigid and universal variable calculi to ground (i.e., variable-free) calculi is explained, including uniform *lifting* methods for constructing rigid and mixed variable calculi from ground calculi. An improved version of *skolemisation* is described where instead of introducing new Skolem symbols, each premiss from which the existence of objects with certain properties can be deduced is assigned its own unique symbol. As examples for calculi making use of these techniques, mixed rigid and universal variable calculi for first-order predicate and modal logics are presented. Other methods for uniformly improving tableau calculi that are described in this chapter include the *local lemma* technique, *pruning* of redundant tableau branches, and the introduction of additional tableau rule schemata.

Chapter 5 This chapter complements Chapter 4 where methods for improving a tableau calculus are discussed such that shorter proofs can be constructed. Here, the subject is how to efficiently search for proofs in the remaining smaller search space. Techniques for turning a (non-deterministic) tableau calculus into a deterministic proof procedure are discussed and analysed. A general concept of *regularity* for arbitrary tableau calculi and the notion of *weight orderings* are introduced, which are proven to be appropriate for constructing a deterministic depth-first proof search procedure for arbitrary ideal rigid variable calculi (although it was known that such procedures exist, it was up to now an unsolved problem to actually describe a practical procedure).

Chapter 6 The *fibring* technique, which allows to combine logical systems based on combining their semantics, is extended to tableau calculi. A method is introduced for uniformly constructing a sound and complete tableau calculus for a combined logic from tableau calculi for the component logics. Since tableau calculi are readily available for most "basic" logics, calculi can be obtained for many "complex" logics that can be constructed by fibring basic logics, such as modal predicate logic, intuitionistic temporal logic, etc. As an example, a calculus for modal predicate logic is presented that is the combination of a calculus for first-order predicate logic and a calculus for a modal logic.

Chapter 7 The concept of *theory reasoning*, which allows to integrate tableau calculi with dedicated procedures for solving problems from a certain domain, is generalised and formulated using the notions introduced in previous chapters

1.3 Tableaux and Why We Use Them

A tableau, as defined in dictionaries, is a "striking or vivid representation" (Hayward & Sparkes, 1968) or a "well-arranged picture" („ein wohlgeordnetes Bild" (Drosdowski *et al.*, 1991)). In automated deduction, a tableau is a special representation of (partial) proofs. Whether this representation is indeed "striking" and "well-arranged" is of course a matter of debate; but from all the proof representations that are well-suited for computers and are thus used in automated deduction, the tableau representation is arguably the one that is easiest to understand and use for humans.

While tableau calculi have always been popular for pedagogical purposes in introductory logic texts, the deduction community became interested in them only in the 1980s. One reason was the increased demand for deduction in non-classical logics in various AI applications. For many non-classical logics tableau-like calculi are the only ones available. In addition, the proximity of tableau inference rules to semantics makes it easy to construct tableau calculi for new logics. Moreover, the introduction of unification and other refinements lead to an increase in the efficiency of tableau calculi for classical logic; today some of the most powerful automated theorem provers for first-order predicate logic are based on tableau-calculi.

There are many different definitions of the notion of tableaux in the literature; and there seem to be as many characterisations of tableau calculi and criteria that distinguish them from other types of calculi as there are experts in the field of tableau-based deduction; the most common and important of these criteria are:

1. A tableau is a tree structure whose nodes are labelled with formulae; and any calculus operating on such structures is a tableau calculus.

2. A calculus is a tableau calculus if it has all (or most) of the following properties: It proves by analysing the theorem to be proven (top-down); a proof is based on a complete case distinction; it proves by contradiction.

In this thesis, a tableau calculus is assumed to satisfy both of the above criteria. In particular the tree structure of tableaux is part of their definition (whereas some authors, e.g. (Fitting, 1998), view trees as only being one of many possible data structures for implementing tableaux). Therefore, calculi such as the connection method, which is based on a matrix representation, and sequent calculi are not considered to be tableau calculi—though they certainly are closely related to tableau calculi.[1]

To avoid confusion, the following notions have to be distinguished:

- tableaux, which are trees whose nodes are annotated with formulae;
- (partial) tableau proofs, which consist of a tableau and the information of how to construct this tableau using the rules of the calculus; this information can, for example, be given in form of a sequence of tableaux where each tableau in the sequence is constructed from the previous one by a single rule application;
- the state that the computation of a tableau proof procedure has reached, which in addition to a partial tableau proof may contain information about futile proof attempts;
- tableau calculi, which are characterised by their expansion and closure rules;
- the tableau method in general.

In the literature, "tableau" is often used inconsistently with any of the above meanings. In this thesis, however, "tableau" always refers to a tree structure, the only (rare) exception being expressions like "semantic tableaux" or "free variable tableaux", which refer to certain tableau calculi or classes of calculi.

1.4 Historical Overview

Historically, tableaux derive from Gentzen's proof theoretic work done in the 1930s, namely his *sequent calculus* introduced in (Gentzen, 1935). The term *tableau* is due to Beth (1955) who was looking for a "systematic method for constructing a counter-example." Roughly at the same time (and, like Beth, motivated by semantic concerns) Hintikka (1955) and Schütte (1956) independently came up with a similar system. These calculi still had drawbacks with respect to notation, which was tedious; but

[1] Every sequent calculus can be turned into a tableau calculus and vice versa, where the branches of tableaux in a tableau calculus correspond to the sequents in a sequent calculus.

Hintikka gave an argument in his completeness proof that is (with few modifications) still in use today (see Section 3.5.4).

The modern form (in particular the tree representation) of tableaux was conceived, again independently, by Lis (1960) and Smullyan; the latter added the device of *unifying notation* and summarised his results in (Smullyan, 1968), which became a well-known textbook.

The development of tableau-like calculi for non-classical logics began in parallel to that of calculi for classical logic—the first being a sequent calculus for intuitionistic logic described in (Gentzen, 1935). Beth presented a calculus for intuitionistic logic, that is similar to his calculus for classical logic, in (Beth, 1959). Roughly at the same time Kanger (1957) and Matsumoto and Ohnishi (1957; 1959) developed tableau-like calculi for modal logics. In Kanger's calculus for the modal logic S5, formulae were indexed with integers; this can be seen as the first tableau-like calculus using the mechanism of labelled formulae. Kripke describes in his celebrated paper (Kripke, 1959) (in which he proposes the possible world semantics for modal logics) a tableau-like calculus for modal logics in the style of Beth (1955). Kripke uses auxiliary tableaux, where a different tableau is used for each possible world, and these tableaux are interrelated by a reachability relation.

The first real tableau calculi (using tree representation) for non-classical logics, namely the modal logic S4 and intuitionistic logic, were presented in (Fitting, 1969) (later Fitting defined tableau calculi for many other modal logics in (Fitting, 1983)). A first tableau calculus for temporal logics was described in (Rescher & Urquhart, 1971); and the first tableau-like calculus for many-valued logics was Rousseau's (1967) sequent calculus (which was based on using sequents consisting of more than two sequences). Later, Suchon (1974) defined a tableau calculus for Łukasiewicz logics; and Surma (1984), Carnielli (1987) and Hähnle (**?**) presented tableau calculi for arbitrary finitely-valued logics.

In the late 1950s, the design of tableau-like calculi and proof procedures began that were tailor-made for automated proof search and thus for implementing automated theorem provers; and that area of research started to separate from the development of tableau calculi for purely theoretical or pedagogical purposes.

The possibility to automate proof search was first considered by Kanger (1957; 1963). In 1957 and '58, D. Prawitz, H. Prawitz, and Voghera implemented a sequent calculus for first-order predicate logic without function symbols (Prawitz, 1960; Prawitz *et al.*, 1960). At the same time, in 1958, Wang implemented a sequent calculus for the $\forall\exists$-fragment of first-order predicate logic on an IBM 704 (Wang, 1960); this remarkable program was able to prove all 220 propositional and 139 of the 158 first-order theorems in Russell and Whitehead's *Principia Mathematica*.

In the 1960s and '70s resolution-based calculi dominated the field of automated deduction; during that time nearly all implementations were in some way based on resolution, a notable exception being Popplestone's (1967) implementation of a Beth-style tableau calculus for first-order predicate logic. The main advantage of resolution was (at that time) that it employed unification for finding useful instantiations for universally quantified variables; whereas tableau calculi lacked this powerful method and were based on enumerating all possible instantiations. In the 1980s, tableau calculi overcame that disadvantage. The introduction of free (or dummy) variables whose instantiations are computed using unification lead to a drastic increase in the efficiency of tableau-based theorem provers.

The idea of using unification in tableau calculi was first considered in (Cohen *et al.*, 1974); the first calculi using unification were formulated (independently) in (Broda, 1980) and (Bowen, 1982). However, in these papers the problem of preserving soundness when dummy variables are instantiated with (Skolem) constants had not been solved (see Section 4.4). In *model elimination* (Andrews, 1981) and the *matrix method* (Bibel, 1982), which are calculi closely related to tableaux that use unification, the problem was avoided by assuming the input formula to be in skolemised normal form such that no Skolem constants are introduced in a proof.

First tableau calculi for first-order predicate logic that use run-time skolemisation and unification were presented by Wrightson (1984) and Reeves (1987); these solved the the soundness problem by imposing certain constraints on the unifiability of free variables and Skolem constants. Finally, the technique for preserving soundness that is mainly used today, namely the use of complex Skolem *terms* instead of Skolem *constants*, was introduced in (Schmitt, 1987) and (Fitting, 1990).

A further important improvement of tableau calculi for first-order predicate logic was the introduction of *connectedness* conditions (see Section 5.5); the notion of connectedness had been used before in *model elimination* (Andrews, 1981) and the *matrix method* (Bibel, 1982).

Today, the most comprehensive available source of information on tableau calculi is the *Handbook of Tableau Methods* (D'Agostino *et al.*, 1998). Its chapters cover the main variants of clausal and non-clausal tableau calculi for classical propositional and first-order predicate logic, as well as tableau systems for the most important families of non-classical logics.

2 Logical Systems

2.1 Syntax and Semantics of Logical Systems

We define the notion of a *logical system* in a very general way; only very basic properties of its syntax and semantics are part of the definition.

The logic has to have a model semantics that uses Kripke-style models, i.e., models consisting of (possible) *worlds* in which formulae are true or false. That includes models consisting of just one world (for example, models of classical propositional and first-order logic);[1] and there is no restriction on what the relationship between these worlds is.

Definition 2.1.1 Associated with a *logical system* $\mathbf{L}$ (a *logic* for short) is a set Sig of ($\mathbf{L}$-)signatures[2] of $\mathbf{L}$. For each signature $\Sigma \in Sig$, syntax and semantics of the instance $\mathbf{L}(\Sigma)$ of $\mathbf{L}$ are given by:

Syntax: A set $Form(\Sigma)$ of *(*$\mathbf{L}$*-)formulae* and a set $Atom(\Sigma) \subset Form(\Sigma)$ of *atomic (*$\mathbf{L}$*-)formulae ([*$\mathbf{L}$*-]atoms)* where the sets $Atom(\Sigma)$ and $Form(\Sigma)$ are decidable formal languages (not containing the empty word), i.e., atoms and formulae are words in these languages, respectively.

Semantics: A set $\mathcal{M}(\Sigma)$ of *(*$\mathbf{L}$*-)models* where each model $\mathbf{m} \in \mathcal{M}(\Sigma)$ (at least) contains (a) a set W of *worlds*, (b) an *initial world* $w^0 \in W$, and (c) a binary relation $\models$ between W and $Form(\Sigma)$.

If $w \models F$ for some world $w \in W$ and some formula $F \in Form(\Sigma)$, then F is said to be *true* in w, else it is *false* in w.

A formula $F \in Form(\Sigma)$ is *satisfied* by a model $\mathbf{m} \in \mathcal{M}(\Sigma)$ if (and only if) it is true in the initial world w^0 of $\mathbf{m}$. A set $G \subset Form(\Sigma)$ of formulae is satisfied by $\mathbf{m}$ iff all its elements are satisfied by $\mathbf{m}$.

[1] In fact, any kind of model can be considered to be a Kripke-style model with a single world (namely the model itself). However, since the labels of tableau formulae are interpreted as worlds, if there is only one world in the models of a logic, then the interpretation of all labels is the same and they become useless for the calculus.

[2] We do not further specify what a signature is; Sig can be seen as a set of indices for distinguishing different instances of the logic $\mathbf{L}$ (which *usually* differ in the symbols they use).

A formula $F \in \mathit{Form}(\Sigma)$ (a set $\mathfrak{F} \subset \mathit{Form}(\Sigma)$ of formulae) is *satisfiable* if there is a model $\mathbf{m} \in \mathcal{M}$ satisfying F (resp. $\mathfrak{F}$). □

Although usually non-atomic formulae are constructed from atomic formulae, and their truth value is determined by the truth value of the atoms they consist of, this is *not* part of the above definition. However, the existence of a "well-behaved" tableau calculus for a logic $\mathbf{L}$ implies that the truth value of a formula F is strongly related to the truth value of certain atoms (that may or may not be sub-formulae of F).

Example 2.1.2 The truth value of the formula $(\exists x)(p(x))$ in a model of first-order predicate logic is not determined by the truth value of the (only) atom $p(x)$ it contains, neither can it be computed from the truth values of all atomic formulae $p(t)$ unless all elements of the model's domain are represented by a term t (as is the case in Herbrand models). □

Tableau calculi allow to check the *satisfiability* of a formula; we only consider this property. It may or may not be possible in a certain logic to check whether a formula is valid in some model (true in all worlds) or is a tautology (valid in all models) by reducing this problem to a satisfiability problem; in many logics—though not in all—a formula is a tautology if its negation is not satisfiable.

Often, formulae are used in tableau calculi that are not built from the original but from an extended signature (e.g., formulae containing Skolem symbols):

Definition 2.1.3 Given a logic $\mathbf{L}$, a signature $\Sigma^* \in \mathit{Sig}$ is an *extension* of a signature $\Sigma \in \mathit{Sig}$ (and Σ is a *restriction* of Σ^*) if

$$\mathit{Form}(\Sigma) \subset \mathit{Form}(\Sigma^*) \text{ and } \mathit{Atom}(\Sigma) \subset \mathit{Atom}(\Sigma^*) \ .$$

In that case, a model $\mathbf{m} \in \mathcal{M}(\Sigma)$ is a *restriction* of a model $\mathbf{m}^* \in \mathcal{M}(\Sigma^*)$ (to the signature Σ), if there is a function f that assigns to each world of $\mathbf{m}$ a world of $\mathbf{m}'$ such that: (a) the initial world of $\mathbf{m}^*$ is assigned to the initial world of $\mathbf{m}$; and (b) for all formulae $F \in \mathit{Form}(\Sigma)$ and worlds w of $\mathbf{m}$: $w \models F$ iff $f(w) \models F$. □

2.2 Terms and Substitutions

2.2.1 Logical Systems with Terms and Free Variables

There is an important class of logical systems, including all predicate logics, where formulae contain *terms*. Again we make as few restrictions as possible on what a term is. In particular we do not assume that terms are of the form $f(t_1, \ldots, t_n)$; instead, a set of terms may be any decidable set of words occurring in the formulae. The only condition, that in fact is used to define the notion of terms, is that the set of formulae is closed under the replacement of terms occurring in a formula by other terms.

To be more flexible, we allow the terms to be separated into different classes, i.e., we attach *sorts* to terms. We do not use a sub-sort hierarchy; however, most notions and methods introduced in the following can easily be adapted to a more complex sort concept (see (Weidenbach, 1995) for an overview of tableau calculi for first-order logic with sorted terms).

Definition 2.2.1 A formal language L is a *language with terms* if there is

1. a non-empty set Term of *(ground) terms* that is a decidable formal language over the same alphabet as L (not containing the empty word);
2. a non-empty set S of *sorts*;
3. a function sort assigning a sort $s \in S$ to each term $t \in \mathit{Term}$ such that there is at least one term of each sort;
4. the set of terms of some sort s is closed under replacing subterms of some sort s' by subterms of the same sort s', i.e., if $vrw \in \mathit{Term}$, $r, r' \in \mathit{Term}$, and $\mathit{sort}(r) = \mathit{sort}(r')$, then $vr'w \in \mathit{Term}$ and $\mathit{sort}(vrw) = \mathit{sort}(vr'w)$;
5. the language L is closed under the replacement of terms by other terms of the same sort, i.e., if $s, t \in \mathit{Term}$, $\mathit{sort}(s) = \mathit{sort}(t)$, and the word wsw' is an element of L, then wtw' is an element of L.

If t and r are terms and t is of the form vrw (v, w may be empty), then r is a *subterm* of t. □

The closure property allows the replacement of terms by place holders or dummies that stand for arbitrary terms of a certain sort. We call these place holders *free variables*. A word containing a free variable X^s of sort s stands for a single (but unknown) element of the set of all words that are the result of replacing X^s by some term of sort s.

A non-ground term is constructed by replacing an arbitrary number of subterms of a ground term by free variables; and a non-ground word is constructed by replacing a ground term occurring in a ground word by a non-ground term.

Definition 2.2.2 Let L be a language with terms; and let Var be an infinite set of *free variables* that do not occur in L. The function sort is (arbitrarily) extended to Var such that there is an infinite number of free variables of each sort $s \in S$.

Then, the set $\mathit{Term}^{\mathrm{fv}}$ of *(non-ground) terms* is defined by:

$$\begin{aligned}
\mathit{Term}^{\mathrm{fv}}_0 &= \mathit{Term} \\
\mathit{Term}^{\mathrm{fv}}_i &= \{vXw \mid vtw \in \mathit{Term}^{\mathrm{fv}}_{i-1},\ t \in \mathit{Term},\ X \in \mathit{Var},\ \mathit{sort}(t) = \mathit{sort}(X)\} \\
\mathit{Term}^{\mathrm{fv}} &= \bigcup_{i \geq 0} \mathit{Term}^{\mathrm{fv}}_i \ .
\end{aligned}$$

The function sort is extended to (non-ground) terms in $\mathit{Term}^{\mathrm{fv}}$ by defining

$$\mathit{sort}(vXw) = \mathit{sort}(vtw)$$

for all variables X and terms t with $\mathit{sort}(X) = \mathit{sort}(t)$.

The language L^{fv} of *(non-ground) words* is defined by

$$L^{\mathrm{fv}} = \{vtw \mid vsw \in L,\ s \in \mathit{Term},\ t \in \mathit{Term}^{\mathrm{fv}},\ \mathit{sort}(s) = \mathit{sort}(t)\} \ .$$

$\square$

Definition 2.2.3 A logical system is a *logic with terms* if, for each signature $\Sigma \in \mathit{Sig}$, the sets $\mathit{Form}(\Sigma)$ of formulae and $\mathit{Atom}(\Sigma)$ atoms are languages with terms with the same set $\mathit{Term}(\Sigma)$ of terms. $\square$

In the following we use the sets $\mathit{Var} = \{X_i \mid i \geq 1\}$ and $\mathit{UVar} = \{\boldsymbol{x}_i \mid i \geq 1\}$ of free variables; we assume these variables to be different from all other occurring symbols (without mentioning that explicitly in definitions).

Free variables, which are either denoted by upper-case letters (X, Y, Z, X_i, X' etc.) or by boldface letters ($\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{x}_i, \boldsymbol{x}'$ etc.), should not be confused with object variables occurring in $\mathit{Term}(\Sigma)$, which are denoted by x, y, z, x_i, x' etc. A term that does not contain free variables is always called *ground*—even if it contains object variables that are not bound by a quantifier.

2.2.2 Substitutions

An important notion is that of *substituting* variables by terms. This concept, which is well-known from classical predicate logic, can easily be extended to our more general notion of terms.

Definition 2.2.4 Let L be a language with terms. A *substitution* is a mapping

$$\sigma : \mathit{Var} \to \mathit{Term}^{\mathrm{fv}}$$

of free variables into (non-ground) terms such that $\mathit{sort}(X) = \mathit{sort}(\sigma(X))$ for all $X \in \mathit{Var}$.

The set $dom(\sigma) = \{X \in \mathit{Var} \mid \sigma(X) \neq X\}$ of variables is called the *domain* of σ; and the set $ran(\sigma) = \{\sigma(X) \mid X \in dom(\sigma)\}$ of terms is called the *range* of σ.

If $dom(\sigma) = \{X_1, \ldots, X_n\}$ is finite, then $\{X_1 \mapsto t_1, \ldots, X_n \mapsto t_n\}$ may be used to represent σ where $t_i = \sigma(X_i)$ ($1 \leq i \leq n$).

A substitution $\{X_1 \mapsto Y_1, \ldots, X_n \mapsto Y_n\}$ that maps the variables X_i in different, pairwise distinct variables Y_j is called a *variable renaming*.

A word $w' \in L$ is a *variant* of a word $w \in L$ if there is a variable renaming π such that $w\pi = w'$.

The *application* $t\sigma$ of a substitution σ to a term t is the result of (simultaneously) replacing all occurrences of free variables X in t by $\sigma(X)$. The term $t\sigma$ is called an *instance* of t. The application of a substitution to a formula and instances of formulae are defined analogously.

The *restriction* $\sigma_{|V}$ of a substitution σ to a set $V \subset \mathit{Var}$ of variables is the substitution that is defined for all $X \in \mathit{Var}$ by:

$$\sigma_{|V}(X) = \begin{cases} \sigma(X) & \text{if } X \in V \\ X & \text{otherwise} \end{cases}$$

The *composition* $\sigma \circ \tau$ of two substitutions σ and τ is the substitution that is for all $X \in \mathit{Var}$ defined by:

$$(\sigma \circ \tau)(X) = \sigma(\tau(X)) \ .$$

The empty substitution, which has an empty domain, is denoted by id.

A substitution σ is *idempotent* if $\sigma = \sigma \circ \sigma$. The set of all idempotent substitutions is denoted by $Subst$.

A substitution τ is *grounding* for a formula F (a set $\mathfrak{F}$ of formulae) if $dom(\tau)$ is the set of free variables occurring in F (resp. $\mathfrak{F}$) and $F\tau$ (resp. $\mathfrak{F}\tau$) does not contain any free variables. □

The result of applying a composition $\sigma \circ \tau$ to a term t can be computed by first applying τ and then σ, i.e., $t(\sigma \circ \tau) = (t\tau)\sigma$.

If an *idempotent* substitution $\sigma = \{X_1 \mapsto t_1, \ldots, X_n \mapsto t_n\}$ is applied to a term, the variables do not have to be replaced simultaneously, i.e.,

$$\sigma = \{X_1 \mapsto t_1\} \circ \cdots \circ \{X_n \mapsto t_n\}$$

Example 2.2.5 The substitutions $\{X \mapsto Y, Z \mapsto Y\}$ and $\{X \mapsto a, Y \mapsto f(b)\}$ are idempotent.

The substitutions $\{X \mapsto Y, Y \mapsto a\}$ and $\{X \mapsto f(X)\}$ are *not* idempotent. □

Definition 2.2.6 Given a finite set W of free variables, a substitution $\sigma \in \mathit{Subst}(\Sigma)$ is *more general* than a substitution $\tau \in \mathit{Subst}(\Sigma)$ (on W) and τ is a specialisation of σ, denoted by $\sigma \leq^W \tau$, iff there is a substitution $\rho \in \mathit{Subst}(\Sigma)$ such that $\tau(X) = (\sigma(X))\rho$ for all $X \in W$. □

The set W contains the "relevant" free variables, i.e., those occurring in the context in which the substitutions σ and τ are used (usually those occurring in a certain tableau). It is of advantage to keep the set W as small as possible because, for example, $\sigma = \{X \mapsto f(Y)\}$ subsumes $\tau = \{X \mapsto f(c)\}$ if $Y \notin W$; otherwise, if $Y \in W$, σ subsumes the substitution $\tau' = \{Y \mapsto f(c), Y \mapsto c\}$ but not τ.

The empty substitution id is the most general of all substitutions, i.e., $id \leq^W \sigma$ for all substitutions σ and all sets W of free variables.

2.2.3 Unification

Although terms have been defined in a more general way than usual, it is still possible to define the notion of *unifiers*. And the problem of testing whether two terms are unifiable is always decidable.

Definition 2.2.7 Let L be a language with terms. Terms $r, t \in \mathit{Term}^{\mathrm{fv}}$ are *unifiable* if $r\mu = t\mu$ for some substitution $\mu \in \mathit{Subst}$. In that case, μ is called a *unifier* of r and t. □

In many logics with terms (e.g., first-order predicate logic), it is possible to represent the set of all solutions to a unification problem (all unifiers) by a single most general unifier (MGU), that is more general than all other unifiers w.r.t. the subsumption relation $\leq^W$ (Def. 2.2.6). In general however, a single MGU is not sufficient to represent all solutions. Instead, a *set* $\mathcal{U}$ of (most general) unifiers has to be used; $\mathcal{U}$ is *complete* if every solution to the given problem is subsumed by one of the unifiers in $\mathcal{U}$.

Example 2.2.8 Let $s = abc$ and $t = XY$ be terms. The two substitutions

$$\sigma_1 = \{X \mapsto ab, Y \mapsto c\} \qquad \text{and} \qquad \sigma_2 = \{X \mapsto a, Y \mapsto bc\}$$

form a complete set of unifiers of s and t; but since they are incomparable w.r.t. $\leq^W$, there is no single substitution that represents all unifiers of s and t. □

In free variable tableau calculi, the cardinality of a complete set of unifiers is closely related to the number of choice points when tableau rules involving unification are applied. Therefore, it is desirable to compute a *minimal* complete set of unifiers. Nevertheless, it is not always useful to enforce minimality since there is a trade-off between the gain of computing a minimal set and the extra cost for checking minimality and removing subsumed substitutions.

Definition 2.2.9 Let L be a language with terms; let W be a set of free variables; and let $r, t \in \mathit{Term}^{\mathrm{fv}}$ be terms. A set $\mathcal{U} \subset \mathit{Subst}$ is a *complete set of unifiers* of r and t w.r.t. W if

1. each $\sigma \in \mathcal{U}$ is a unifier of r and t *(soundness)*,
2. for each unifier τ of r and t there is a unifier $\sigma \in \mathcal{U}$ such that $\sigma \leq^{W} \tau$ *(completeness)*.

The set $\mathcal{U}$ is called a *minimal* complete set of unifiers if, in addition,

3. there are *no* $\sigma_1, \sigma_2 \in \mathcal{U}$, $\sigma_1 \neq \sigma_2$, such that $\sigma_1 \leq^{W} \sigma_2$ *(minimality)*. □

If two terms are unifiable, then there is a *finite* complete set of unifiers.

The computation of (most general) unifiers is closely related to the computation of (most general) common specialisations of (idempotent) substitutions, because ρ is a common specialisation of σ and τ iff it is a unifier of the terms $\sigma(X)$ and $\tau(X)$ for all $X \in \mathit{dom}(\sigma) \cap \mathit{dom}(\tau)$.

2.3 First-order Predicate Logic

As a first example for a logical system, we use first-order predicate logic PL1, which is a logic system *with terms*. According to Definition 2.1.1, the set $\mathit{Sig}_{\mathrm{PL1}}$ of signatures and the syntax and semantics of PL1 have to be defined.

Signatures: A signature $\Sigma = \langle P(\Sigma), F(\Sigma), \alpha_\Sigma \rangle$ in $\mathit{Sig}_{\mathrm{PL1}}$ consists of a set $P(\Sigma)$ of predicate symbols, a non-empty set $F(\Sigma)$ of function symbols, and a function α_Σ that assigns an arity $n \geq 0$ to each predicate and each function symbol. A function symbol of arity 0 is called a *constant*.

Syntax: In addition to the predicate and function symbols in signatures there is an infinite set V of *object variables* (which is disjoint from the sets Var and UVar of free variables). The *logical operators* are $\vee$ (disjunction), $\wedge$ (conjunction), $\rightarrow$ (implication), and $\neg$ (negation), and the quantifier symbols $\forall$ and $\exists$. We consider $\phi \leftrightarrow \psi$ (equivalence) to be an abbreviation for $(\phi \wedge \psi) \vee (\neg\phi \wedge \neg\psi)$.

Definition 2.3.1 Let $\Sigma \in \mathit{Sig}_{\mathrm{PL1}}$ be a signature. The set $\mathit{Term}^{nc}(\Sigma)$ of *terms* over Σ is defined by:

1. All variables $x \in V$ and all constants $c \in F(\Sigma)$ are terms over Σ.

2. If $f \in F(\Sigma)$ and $t_1, \dots, t_{\alpha_\Sigma(f)}$ are terms over Σ, then $f(t_1, \dots, t_{\alpha_\Sigma(f)})$ is a term over Σ.

By $Term^0_{\mathrm{PL1}}(\Sigma)$ we denote the set of all terms in $Term^{nc}(\Sigma)$ that do not contain object variables.

The set $Atom^{nc}_{\mathrm{PL1}}(\Sigma)$ of *atoms* over Σ is defined by: If $p \in P(\Sigma)$ and $t_1, \dots, t_{\alpha_\Sigma(f)}$ are terms over Σ, then $p(t_1, \dots, t_{\alpha_\Sigma(f)})$ is an atom over Σ.

The set $Form^{nc}_{\mathrm{PL1}}(\Sigma)$ of formulae over Σ is defined by:

1. Atoms over Σ are formulae over Σ.
2. If F is a formula over Σ, then $\neg F$ is a formula over Σ.
3. If F and G are formulae over Σ, then $F \wedge G$, $F \vee G$, and $F \rightarrow G$ are formulae over Σ.
4. If F is a formula over Σ and $x \in V$, then $(\forall x)F$ and $(\exists x)F$ are formulae over Σ.

$\square$

We define the set $Form_{\mathrm{PL1}}(\Sigma)$ of formulae of the logical system PL1 to consist of all *sentences* in $Form^{nc}_{\mathrm{PL1}}(\Sigma)$, i.e., all elements of $Form^{nc}_{\mathrm{PL1}}(\Sigma)$ in which object variables occur only bound by a quantifier; and the set $Atom_{\mathrm{PL1}}(\Sigma)$ of atoms of PL1 is the set of all atomic sentences in $Form^0_{\mathrm{PL1}}(\Sigma)$.

Note that this is a slight abuse of terminology. Usually, object variables can occur free in formulae of first-order logic. Free object variables are often used as dummy variables in free variable tableau calculi. Here, however, we have separated free (dummy) variables and object variables. One could still allow free object variables to occur in formulae; but that does not increase expressivity and complicates the design of a tableau calculus unnecessarily. Free variables in a formula whose unsatisfiability is to be proven would have to be treated as if they were existentially quantified. In addition, one would have to be careful not to introduce free variables into the scope of a quantification when free variables are instantiated with terms. To avoid these complications, we formally define the set of formulae of the logical system PL1 to consist of sentences only; formulae with free object variables are only considered as auxiliaries used in the construction of sentences. Accordingly, the logical system PL1 is a logic *with terms* w.r.t. to the set $Term^0_{\mathrm{PL1}}(\Sigma)$ of terms not containing object variables, since the set $Form_{\mathrm{PL1}}(\Sigma)$ is closed under replacement of variable-free terms by other variable-free terms.

Semantics: According to Definition 2.1.1, all models must contain a set of worlds. Thus, we define $\mathcal{M}_{\mathrm{PL1}}(\Sigma)$ to consist of models where the initial world w^0 is a first-order structure (Def. 2.3.2), and $\{w^0\}$ is the set of worlds.

Definition 2.3.2 A first-order *structure* $\langle D, \mathcal{I} \rangle$ for a signature $\Sigma \in \mathit{Sig}_{\mathrm{PL1}}$ consists of a domain D and an interpretation $\mathcal{I}$, which gives meaning to the function and predicate symbols of Σ.

A *variable assignment* is a mapping $\mu : V \to D$ from the set of object variables to the domain D.

The *evaluation function* val is defined as usual; that is, given a structure $\langle D, \mathcal{I} \rangle$ and a variable assignment μ, it assigns to each formula $F \in \mathit{Form}^{nc}_{\mathrm{PL1}}(\Sigma)$ a truth value $\mathit{val}_{\mathcal{I},\mu}(F) \in \{\mathit{true}, \mathit{false}\}$. $\square$

The relation $\models_{\mathrm{PL1}}$ is defined by: $w^0 \models_{\mathrm{PL1}} F$ if and only if, for all variable assignments μ, $\mathit{val}_{\mathcal{I},\mu}(F) = \mathit{true}$.

2.4 Modal Logics

As a second example for logical systems, we use the (basic) modal logics K, KT, KB, K4, K5, K45, KD, KDB, KD4, KD5, KD45, KB4, B, S4, and S5 (see (Goré, 1998) for an overview).

Signature: The set $\mathit{Sig}_{\mathbf{L}}$ is the same for all modal logics **L**; a signature Σ is a denumerable non-empty set of primitive propositions.

Syntax: Formulae are built using the classical connectives $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (negation), and the non-classical unary modal connectives $\Box$ (“box”) and $\Diamond$ (“diamond”).

Given a signature Σ, the set $\mathit{Form}_{\mathbf{L}}(\Sigma) = \mathit{Form}_{\mathrm{mod}}(\Sigma)$ of formulae is the same for all modal logics **L**; formulae are constructed in the usual way from the propositional variables and the logical connectives. The set $\mathit{Atom}_{\mathbf{L}}(\Sigma) = \mathit{Atom}_{\mathrm{mod}}(\Sigma)$ of atoms is identical to Σ. The modal logics are logical systems without terms.

Semantics: We use a Kripke-style possible world semantics for modal logics. Thus, the models of modal logics consist of Kripke frames:

Definition 2.4.1 A Kripke *frame* is a pair $\langle W, R \rangle$, where W is a non-empty set (of possible worlds) and R is a binary relation on W.

A *Kripke model* is a triple $\langle W, R, V \rangle$, where the valuation V is a mapping from propositional variables to sets of worlds. Thus, $V(p)$ is the set of worlds at which p is *true* under the valuation V.

Name	Axiom	Property
(K)	$\Box(A \to B) \to (\Box A \to \Box B)$	—
(T)	$\Box A \to A$	reflexive
(D)	$\Box A \to \Diamond A$	serial
(4)	$\Box A \to \Box\Box A$	transitive
(5)	$\Diamond A \to \Box\Diamond A$	euclidean
(B)	$A \to \Box\Diamond A$	symmetric

Table 2.1: Basic modal axioms and their corresponding restrictions on the reachability relation.

If wRw' (i.e., $\langle w, w' \rangle \in R$) then the world w' is *reachable* from world w, and w' is a *successor* of w.

The notion of propositions being true in a world is extended to complex formulae $F \in Form_{\mathbf{L}}(\Sigma)$ as follows: F is *true* in a world w iff:

- G is not true in w, in case $F = \neg G$,
- G_1 and G_2 are true in w, in case $F = G_1 \wedge G_2$,
- G_1 or G_2 is true in w, in case $F = G_1 \vee G_2$,
- G is true in all worlds reachable from w, in case $F = \Box G$,
- G is true in some world reachable from w, in case $F = \Diamond G$.

$\Box$

The first two columns of Table 2.2 show the axiomatisations of the 15 basic modal logics that can be formed from the axioms shown in Table 2.1.

Definition 2.4.2 Given one of the logics **L** listed in Table 2.2, a Kripke frame $\langle W, R \rangle$ is an **L**-*frame* if every formula instance of each axiom of **L** is true in all worlds of $\langle W, R \rangle$.

A Kripke model $\langle W, R, V \rangle$ is an **L**-*model* if $\langle W, R \rangle$ is an **L**-frame. $\Box$

It is well-known that the axioms listed in Table 2.1 are characterised by the properties of R listed next to them. Thus, all KT-frames have a reflexive accessibility relation R, and if a frame has a reflexive accessibility relation then it validates axiom (T). Therefore, we associate these properties with logics as well, and say, for example, that a logic **L** is serial if all **L**-frames have a serial accessibility relation. Some care is needed here: for example the axiom (D) is not an axiom of KT, but it is valid in all KT-frames since it is implied by (T). Consequently the reachability relation R of all KT-models *is* serial.

Logic	Axioms	Logic	Axioms
K	(K)	KT	(K), (T)
KB	(K), (B)	K4	(K), (4)
K5	(K), (5)	K45	(K), (4), (5)
KD	(K), (D)	KDB	(K), (D), (B)
KD4	(K), (D), (4)	KD5	(K), (D), (5)
KD45	(K), (D), (4), (5)	KB4	(K), (B), (4)
B	(K), (T), (B)	S4	(K), (T), (4)
S5	(K), (T), (5)		

Table 2.2: Axiomatic characterisations of the basic modal logics.

We can now proceed to formally define the semantics of the basic modal logics. Let $\mathbf{L}$ be one of the basic modal logics listed in Table 2.2, and let Σ be a signature in $Sig_{\mathbf{L}}$. A model $\mathbf{m}$ in $\mathcal{M}_{\mathbf{L}}(\Sigma)$ consists of a Kripke $\mathbf{L}$-model $\langle W, R, V\rangle$. One of the worlds in W is chosen to be the initial world w^0. The relation $\models_{\mathbf{L}}$ is, for all worlds $w \in W$ and formulae $F \in Form_{\mathbf{L}}(\Sigma)$ defined by: $w \models_{\mathbf{L}} F$ if and only if F is true in w.

2.5 Modal Logics Without Binary Connectives

As a further example for logical systems, we use modal logics $\widehat{\mathbf{L}}$ without binary logical connectives. That is, all formulae are modal literals, i.e., they are of the form $\circ_1 \cdots \circ_n\, p$ ($n \geq 0$), where p is a propositional variable and $\circ_i$ is one of the modalities $\Box$, $\Diamond$ or the negation symbol $\neg$; the semantics of $\widehat{\mathbf{L}}$ is the same as that of the corresponding full modal logic $\mathbf{L}$. More formally:

Signatures: The set $Sig_{\widehat{\mathbf{L}}}$ of signatures of $\widehat{\mathbf{L}}$ is the same as that of the modal logic $\mathbf{L}$, i.e., a signature is a denumerable non-empty set of propositional variables.

Syntax: Let Σ be a signature in $Sig_{\widehat{\mathbf{L}}}$. Then $Form_{\widehat{\mathbf{L}}}(\Sigma)$ is the set of formulae in $Form_{\mathbf{L}}$ that consist of a single element of Σ prefixed by a sequence of the logical operators $\Box$, $\Diamond$, and $\neg$. The set $Atom_{\widehat{\mathbf{L}}}(\Sigma)$ is identical to Σ.

Semantics: The set $\mathcal{M}_{\widehat{\mathbf{L}}}(\Sigma)$ of models of $\widehat{\mathbf{L}}$ is identical to the set $\mathcal{M}_{\mathbf{L}}$ of models of $\mathbf{L}$; and the relation $\models_{\widehat{\mathbf{L}}}$ is the restriction of $\models_{\mathbf{L}}$ to the formulae from $Form_{\mathbf{L}}(\Sigma)$ occurring in $Form_{\widehat{\mathbf{L}}}(\Sigma)$.

Every formula in $Form_{\widehat{\mathbf{L}}}(\Sigma)$ is satisfiable. Nevertheless, the logic is not trivial, because we are interested in the satisfiability of *sets* of formulae, which are implicitly conjunctively connected and can be unsatisfiable.

The logics $\widehat{\mathbf{L}}$ are used in Chapter 6 to demonstrate the benefits of fibring, i.e., combining logics and their calculi. The missing connectives can be added by fibring $\widehat{\mathbf{L}}$ with first-order predicate logic PL1; and a calculus for the resulting modal predicate logic can be constructed by fibring calculi for $\widehat{\mathbf{L}}$ and PL1.

2.6 The Fragments MLSS and MLSSF of Set Theory

As further examples for logical systems present two fragments of quantifier-free set theory; a new and improved tableau calculus for these logics is defined in Section 3.8.

Set theory is the common language of mathematics. Therefore, set theory plays an important rôle in many applications of automated deduction. For example, some of the most widely used specification languages, namely the Z and B specification languages, are completely based on set theory. For other languages, sets are at least a very important construct, frequently used in specifications either on the meta-level or as a data structure of the specified programs. Set theoretic proof obligations occur both as part of proving an implementation to be sound w.r.t. a specification and as part of immanent reasoning (such as consistency checks, proving invariants, pre- and post-conditions).

Set theoretic reasoning, i.e., employing special purpose techniques instead of using the axioms of set theory, is indispensable for automated deduction in many real world domains. Automated deduction tools can, for example, be integrated into interactive software verification systems and relieve the user from the need to interactively handle simple set theoretic problems that do not require his or her intuition but merely a combinatorial search.

Multi-level syllogistic (MLS) consists of quantifier-free formulae built using the set theoretic predicates *membership*, *equality*, *set inclusion*, the binary functions *union*, *intersection*, *set difference*, and a constant representing the empty set. In the extension MLSS of MLS, n-ary functions $\{\cdot\}_n$ can be used to construct singletons, pairs, etc. The fragment MLSSF consists of MLSS enriched with free (uninterpreted) function symbols.

The expressiveness of MLSS and MLSSF is sufficient for many applications. MLSS formulae can contain variables that are implicitly universally quantified. The main restriction is that there is no existential quantification; thus, sentences such as “there is an infinite set” cannot be formalised within MLSS.

Decision and semi-decision procedures for various extensions of MLS have been described in the literature; these, however, are not based on tableaux but are highly non-deterministic search procedures and are not suitable for implementation; an overview can be found in (Cantone & Ferro, 1995; Cantone *et al.*, 1989). Extensions of MLS that are known to be decidable include: MLS with powerset and singleton (Cantone, 1991; Cantone *et al.*, 1985), with relational constructs (Cantone & Schwartz, 1991), with unary union (Cantone *et al.*, 1987), and with a choice operator (Ferro & Omodeo, 1987).

Signatures An MLSSF signature Σ is a signature of PL1 such that

1. its set $P(\Sigma)$ of predicate symbols consists of the binary symbols $\sqsubset$ (membership), $\approx$ (equality), and $\sqsubseteq$ (set inclusion),
2. its set $F(\Sigma)$ of function symbols consists of
   (a) the binary function symbols $\sqcap$ (intersection), $\sqcup$ (union), $\setminus$ (set difference), the set constructors $\{\cdot\}_n$ with arity $n \geq 1$ (singleton, pair, etc.), and the set theoretic constant $\emptyset$ (the empty set),
   (b) function symbols that have no special set theoretic interpretation; they are called *free* function symbols.

An MLLSF signature is an MLSS signature if all free function symbols are constants, i.e., are of arity 0.

Syntax The formulae of MLSS and MLSSF are built according to the rules of first-order predicate logic using the logical connectives $\vee$ (disjunction), $\wedge$ (conjunction), $\neg$ (negation), and $\rightarrow$ (implication) but *no* quantifiers.

Definition 2.6.1 Let Σ be an MLSS (resp. MLSSF) signature; then the set of *atoms* of MLSS (MLSSF), which is denoted by $Atom_{\mathrm{MLSS}}(\Sigma)$ (resp. $Atom_{\mathrm{MLSSF}}(\Sigma)$) is the set $Atom_{\mathrm{PL1}}(\Sigma)$ of all PL1-atoms over Σ; and the set of *formulae* of MLSS (MLSSF), which is denoted by $Form_{\mathrm{MLSS}}(\Sigma)$ (resp. $Form_{\mathrm{MLSSF}}(\Sigma)$) is the set of all PL1-formulae over Σ that do not contain any quantifiers $\forall$ or $\exists$ (and, thus, no object variables). $\square$

Notation 2.6.2 To avoid confusion we use the non-standard symbols $\sqsubset, \approx, \sqsubseteq, \sqcap, \sqcup, \emptyset$ on the object level and the standard symbols $\in, =, \subset, \cap, \cup, \emptyset$ on the meta level.

As usual, the binary set theoretic function and predicate symbols are written in infix notation, and $\{\cdot\}_n$ is written in circumfix notation. $\square$

Definition 2.6.3 A term $t \in Term_{\mathrm{PL1}}(\Sigma)$ over an MLSSF signature Σ is a *set term*. It is a *pure* set terms if it does not contain free function symbols f with arity $\alpha_\Sigma(f) > 0$. A set term is called *functional* if it is of the form $f(t_1, \ldots, t_n)$ where f is a free function symbol. $\square$

Note that functional set terms can contain non-functional set terms (which are not necessarily pure) and vice versa.

Semantics We use the semantics of set theory (and thus its fragments MLSS and MLSSF) as it is defined by the ZF axiom system or, equivalently, by the von Neumann hierarchy (cumulative hierarchy) of sets (see, for example, (Jech, 1978) for a detailed discussion of the semantics of set theory).

Definition 2.6.4 Let Ord denote the class of all ordinals. The *von Neumann hierarchy* $\mathfrak{V}$ is defined by

$$\mathfrak{V} = \bigcup_{\alpha \in Ord} \mathfrak{V}_\alpha$$

where

1. $\mathfrak{V}_0 = \emptyset$,
2. $\mathfrak{V}_\alpha = \bigcup_{\beta<\alpha} \mathfrak{V}_\beta$ for each limit ordinal α, and
3. $\mathfrak{V}_{\alpha+1}$ is the powerset of $\mathfrak{V}_\alpha$ for each ordinal α. □

Definition 2.6.5 Let Σ be an MLSS or MLSSF signature. A first-order structure $\mathbf{m} = \langle D, \mathcal{I} \rangle \in \mathcal{M}_{\mathrm{PL1}}(\Sigma)$ is a *set structure* if it has the following properties:

1. The elements of D are sets in the von Neumann hierarchy $\mathfrak{V}$;
2. D is closed under the set operations $\cap$, $\cup$, $\setminus$, and $\{\cdot\}_n$ ($n \geq 1$), and it contains the empty set;
3. $\mathcal{I}$ interprets
   (a) the constant $\emptyset$ by the empty set,
   (b) the predicate symbols by their canonical interpretations, i.e., $\sqsubset$ by $\in$, $\approx$ by the identity relation, and $\sqsubseteq$ by $\subseteq$,
   (c) the set theoretic function symbols by their canonical interpretations, i.e., $\sqcup$ by $\cup$, $\sqcap$ by $\cap$, $\setminus$ by $\setminus$, and $\{\cdot\}_n$ by $\{\cdot\}_n$ ($n \geq 1$). □

As models of logical systems must contain a set of worlds, we define models of MLSS and MLSSF to consist of a single (initial) world w^0 that is a set structure.

The relations $\models_{\mathrm{MLSS}}$ and $\models_{\mathrm{MLSSF}}$ are defined in the same way as the relation $\models_{\mathrm{PL1}}$ of PL1: an MLSS-formula or MLSSF-formula F is true in the world w^0, which is a set structure, if and only if, for all variable assignments μ, $val_{\mathcal{I}}(F) = true$.

One could allow free object variables to occur in MLSS and MLSSF formulae; but that would not enhance expressivity. Since free object variables in quantifier-free formulae are implicitly universally quantified, a formula $G(x)$ is valid in MLSS or MLSSF if and only if a skolemisation $\neg G(c)$ of its negation is unsatisfiable. Thus, free object variables can be eliminated, and a tableau calculus for formulae without free object variables is sufficient.

3 Tableau Calculi

3.1 A Uniform View

It is important to distinguish the two phases into which the development of an efficient tableau-based proof procedure can be separated: the design of a tableau calculus and the construction of a proof procedure based on that calculus. A tableau calculus is mainly characterised by a collection of deduction rules that may be employed to non-deterministically construct a tableau proof; a proof procedure is a description of how to search for a proof using a certain calculus.

In the literature on tableau calculi, these two phases are often intermingled; refinements that are neither needed for soundness nor for completeness of a calculus but are intended to improve efficiency are made part of the calculus's definition. That is harmful because whatever properties the calculus is then shown to have are actually only properties of the refined calculus. In addition, refined calculi are often less "well-behaved" than their pure versions, which makes it more difficult to apply the uniform methods for constructing an efficient proof procedure or combining different calculi that are described in the following chapters.

A typical refinement that should not be made part of the definition of a calculus is the often useful heuristic that applications of non-branching rules are preferable to applications of rules introducing several new tableau (sub-)branches. If this heuristic is part of the definition of a calculus, then it is, for example, impossible to employ a more sophisticated technique for chosing the next of several possible rule applications that is based on measuring the complexity of the formulae that are added by an application.

Consequently, avoiding redundancy by making a calculus more deterministic and restricting the search space is not the main issue of this chapter; it is the topic of Chapter 5, where the design of efficient proof procedures is discussed. That notwithstanding, efficiency has to be considered when a calculus is designed; for example, heavily branching rules, such as the cut rule, should be avoided (but only if there are less-branching rules that are equally "well-behaved").

As few restrictions as possible are made regarding the type and form of tableaux and tableau calculi. But as said in the introduction (Section 1.3), a tableau is a tree whose nodes are labelled with formulae. If no further restrictions are made, nothing is known about the behaviour of tableau calculi, except that tableaux represent states of the proof search, and a tableau rule application corresponds to a state transition. In particular, nothing is known about the way in which states are represented and what the relation between tableaux is.

To be able to formulate general theorems and apply uniform methods, additional assumptions have to be made regarding the behaviour of calculi. The first of these assumptions is that tableau branches represent different cases of a proof and that they are thus implicitly disjunctively connected; closing a branch means that the corresponding case has been successfully handled. Since branches represent distinct cases, that implies that the effect of tableau rule applications are local to a branch.

The next step is to assume that the formulae on a branch are implicitly conjunctively connected and represent the knowledge that has been derived about the proof case corresponding to the branch. That entails the existence of a tableau expansion and closure rule that is monotonic and operates on the sets of formulae; the order of formulae on a branch becomes irrelevant.

The final step are semantical assumptions. A branch is considered to define a partial model. The tableau construction then corresponds to the construction of a model of the formulae on the initial tableau. A branch is closed if a contradiction is found in the partial model defined by that branch. A closed tableau then proves the fact that the formulae on the initial tableau are unsatisfiable.

To include as many different calculi as possible in our general definition of the notion of tableau calculi, labels and truth value signs are attached to formulae in tableaux. Labels have many uses, they allow to make information about formulae and about the relation between formulae explicit; labels are particularly useful in calculi for non-classical logics (e.g., many-valued, modal, and intuitionistic logics); many tableau-like calculi using labelled formulae are described in (Gabbay, 1996b). Tableau calculi with labels attached to formulae are much more powerful than calculi that encode the information otherwise contained in labels into the structure of tableaux. If information is implicitly represented by the structure of tableaux, then any changes affecting the structure may destroy soundness and completeness of the calculus; therefore, we say that such calculi are not *ideal* (the idealness property is formally defined in Section 3.3.7).

The fact that only the two truth value signs T and F are used does not imply a restriction to two-valued logic; these signs represent the fact that a formula is (resp. is not) satisfied by a model. The truth-values of a many-valued logic can be encoded into the labels attached to tableau formulae.

3.2 Syntax of Tableau Calculi

As said before, we consider the tree structure of tableaux to be an essential property; trees are *not* just a data structure for implementing tableaux. Calculi such as, for example, sequent calculi and the connection method, which operate on other data structures, are only “tableau-like”.

Definition 3.2.1 Let **L** be a logic; let $\Sigma \in \mathit{Sig}$ be a signature; and let Lab be a set of labels.

A *tableau formula* $\mathsf{S}{:}\sigma{:}F$ consists of a truth value sign $\mathsf{S} \in \{\mathsf{T}, \mathsf{F}\}$, a label $\sigma \in \mathit{Lab}$, and a formula $F \in \mathit{Form}(\Sigma)$; it is called *atomic* if $F \in \mathit{Atom}(\Sigma)$. In addition, the symbol $\bot$ is a tableau formula (which indicates branch closure). The set of all tableau formulae is denoted with $\mathit{TabForm}(\Sigma)$.

The complement $\overline{\phi}$ of a tableau formula ϕ is defined by: $\overline{\phi} = \mathsf{F}{:}\sigma{:}F$ if ϕ is of the form $\mathsf{T}{:}\sigma{:}F$, and $\overline{\phi} = \mathsf{T}{:}\sigma{:}F$ if ϕ is of the form $\mathsf{F}{:}\sigma{:}F$ (the complement of $\bot$ is undefined).

A *tableau* (over the signature Σ) is a finitely branching tree whose nodes are labelled with tableau formulae from $\mathit{TabForm}(\Sigma)$.

A *branch* of a tableau T is a maximal path in T. The set of formulae on a branch B is denoted with $\mathit{Form}(B)$. □

In the following, we often identify a node in a tableau and the formula with which it is labelled.

To keep the notion of tableau calculi as general as possible, any function that assigns to a tableau a set of possible successor tableaux is considered to be a *tableau rule*. A tableau rule can change a tableau to which it is applied in an arbitrary way. Tableau *expansion* rules are a special case of tableau rules; they are discussed in Section 3.3.3.

A tableau calculus $\mathcal{C}$ for a logic **L** has (different) “instances” $\mathcal{C}(\Sigma)$ for each signature $\Sigma \in \mathit{Sig}$. We allow formulae from an extended signature Σ^* to be used in a tableau proof. Only the tableau formulae that are tested for satisfiability have to be taken from the language of the non-extended signature Σ, which is a restriction of Σ^*; they are put on the initial tableau. The possibility to use an extended signature in the proof or, equivalently, to demand that the formulae that are tested for satisfiability are taken from the language of a restricted signature is indispensable for many tableau calculi; it allows, for example, to introduce Skolem symbols that are know not to occur in the initial tableau.

Definition 3.2.2 A tableau calculus $\mathcal{C}$ for a logic **L** is, for each signature $\Sigma \in \mathit{Sig}$, specified by:

- an extension $\Sigma^* \in \mathit{Sig}$ of the signature Σ (Def. 2.1.3);
- a set Lab of labels and an initial label $\sigma^0 \in \mathit{Lab}$;
- a tableau rule $\mathcal{R}(\Sigma)$ that assigns to each tableau T over the signature Σ^* a set of tableaux over Σ^*, which are the (possible) *successor tableaux* of T. The set $\mathcal{R}(\Sigma)(T)$ may be infinite but has to be enumerable.

□

We now have everything at hand to define what the tableaux for a set $\mathfrak{F}$ of formulae are and when a tableau is closed. The construction of tableaux for $\mathfrak{F}$ is in general a non-deterministic process, since a tableau may have any—even an infinite—number of possible successor tableaux.

Definition 3.2.3 Let $\mathcal{C}$ be a tableau calculus for a logic **L**; and let $\Sigma \in \mathit{Sig}$ be a signature. The set of all tableaux for a finite set $\Gamma \subset \mathit{TabForm}(\Sigma^*)$ of tableau formulae is inductively defined as follows:

1. A linear tableau whose nodes are labelled with the formulae in Γ is a tableau for Γ (an *initial tableau*).
2. If T is a tableau for Γ and $T' \in \mathcal{R}(\Sigma^*)(T)$, i.e., T' is a successor tableau of T, then T' is a tableau for Γ.

A tableau T is a tableau for a finite set $\mathfrak{F} \subset \mathit{Form}(\Sigma)$ of formulae if it is a tableau for the set $\{\mathsf{T}{:}\sigma^0{:}F \mid F \in \mathfrak{F}\}$ of tableau formulae. □

Some useful tableau calculi, by definition, start with an empty initial tableau and allow to (later on) extend branches of a tableau by formulae from the set $\mathfrak{F}$ whose unsatisfiability is to be proven. This can easily be modelled in our framework by introducing a special initial label $\circ$ with the meaning “is not yet on the branch” and extending the tableau rule such that $\mathsf{S}{:}\sigma^0{:}F$ can be derived from $\mathsf{S}{:}\circ{:}F$ (where σ^0 is the original initial label). That derivation then corresponds to adding the formula $\mathsf{S}{:}\sigma^0{:}F$ to the tableau. Another possibility, which is also applicable if the set $\mathfrak{F}$ is infinite, is to extend the tableau rule such that $\mathsf{S}{:}\sigma^0{:}F$ can be derived from the empty premiss.

Definition 3.2.4 A tableau branch B is *closed* if one of its nodes is labelled with $\perp$.

A tableau is *closed* if all its branches are closed. □

Intuitively, a tableau proof for a set $\mathfrak{F}$ of formulae proves the unsatisfiability of $\mathfrak{F}$ (provided that the calculus is sound).

Definition 3.2.5 Let $\mathcal{C}$ be a tableau calculus for a logic **L**; and let $\Sigma \in \mathit{Sig}$ be a signature. A *tableau proof* for a set $\mathfrak{F} \subset \mathit{Form}(\Sigma)$ of formulae is a finite sequence $T_0, \ldots, T_n$ ($n \geq 0$) of tableaux for $\mathfrak{F}$ such that

- T_0 is an initial tableau for $\mathfrak{F}$;
- T_i is a successor tableau of T_{i-1} ($1 \leq i \leq n$);

- T_n is closed.

□

Lemma 3.2.6 *Given a tableau calculus $\mathcal{C}$ for a logic* **L** *and a signature $\Sigma \in Sig$, there is a tableau proof for a set $\mathfrak{F} \subset Form(\Sigma)$ of formulae if and only if there is a closed tableau for $\mathfrak{F}$.*

Proof: This follows immediately from the definitions of tableaux for a formula set and of tableau proofs. □

3.3 Syntactical Properties

3.3.1 Non-destructive Tableau Calculi

A tableau calculus is non-destructive if applications of its tableau rule do not alter or remove formulae already on the tableau but only add new formulae.

Definition 3.3.1 A tableau calculus is *non-destructive* if all possible successor tableaux of a tableau T contain T as an initial subtree; otherwise the calculus is *destructive*. □

Example 3.3.2 A typical example for destructive calculi are those that use free variables in tableau formulae and allow the instantiation of the free variables by terms when the tableau rule is applied. □

3.3.2 Proof Confluence

A tableau calculus is *proof confluent* if there are no "dead ends" in the proof search. A certain tableau rule application may be useless for constructing a proof for a formula set $\mathfrak{F}$, but it cannot prevent the construction of a proof if the calculus is proof confluent. This property is important for reducing the size of the search space. A deterministic proof procedures for a proof confluent calculus can be constructed by ensuring *fairness* (Chapter 5) of tableau rule applications; backtracking is not needed.

Definition 3.3.3 A tableau calculus $\mathcal{C}$ for a logic **L** is *proof confluent* if each sequence $T_0, \dots, T_k$ ($k \geq 0$) of tableau for a set $\mathfrak{F} \subset Form(\Sigma)$ of formulae for which a tableau proof exists can be extended to a tableau proof $T_0, \dots, T_k, \dots, T_n$ ($n \geq k$) for $\mathfrak{F}$. □

3.3.3 Tableau Calculi with Expansion Rule

The most important syntactical property of tableau calculi forcing them to be "well-behaved" is that tableau rule applications have only *local* effects. That is, the tableau rule is non-destructive and it extends only a *single* branch of a tableau. In addition, what the possibilities for extending a branch B are, only depends on B itself; no additional pre-conditions are allowed such as, for example, the presence of formulae on other branches. A calculus has this "local effects" property if its tableau rule can be described in form of an *tableau expansion rule*.

Definition 3.3.4 Let $\mathcal{C}$ be a tableau calculus for a logic **L**; and let $\Sigma \in Sig$ be a signature.

A *branch extension* is a finite subset of tableau formulae over Σ^*.

A tableau rule *conclusion* is a finite set of branch extensions.

A tableau *expansion rule* $\mathcal{E}(\Sigma)$ is a function that assigns to each (finite) tableau branch whose nodes are labelled with formulae from $TabForm(\Sigma^*)$ a set $\mathcal{E}(\Sigma)(B)$ of (possible) conclusions, which may be infinite but has to be enumerable. □

Note that special *closure* rules are not needed. A branch is closed by extending it with the special tableau formula $\perp$. Thus, branch closure can be considered to be a special case of branch extension.

Definition 3.3.5 Let $\mathcal{C}$ be a tableau calculus for a logic **L**; and let $\Sigma \in Sig$ be a signature.

An expansion rule $\mathcal{E}(\Sigma)$ *characterises* the tableau rule $\mathcal{R}(\Sigma)$ of $\mathcal{C}$ if, for all tableaux T over Σ^*: a tableau T' is a successor tableau of T (i.e., $T' \in \mathcal{R}(\Sigma)(T)$) if and only if there is a branch B of T and a conclusion C in $\mathcal{E}(\Sigma)(B)$ such that the tableau T' can be constructed from T by extending the branch B by a new sub-branch for each extension E in C where the nodes in that sub-branch are labelled with the elements of E.

If the expansion rule $\mathcal{E}(\Sigma)$ characterises the tableau rule $\mathcal{R}(\Sigma)$ of a calculus $\mathcal{C}$ for all signatures Σ, then $\mathcal{E}$ is said to be *the* expansion rule of $\mathcal{C}$; and $\mathcal{C}$ is said to be a calculus with expansion rule $\mathcal{E}$. □

Theorem 3.3.6 *A tableau calculus with expansion rule is non-destructive.*

Proof: The theorem follows immediately from Definitions 3.3.1 and 3.3.4. □

3.3.4 Analytic Tableau Calculi

Two different notions of *analytic* calculi can be found in the literature. One is that a calculus is analytic if it "analyses" the formula to be proven, i.e., if it is a top-down procedure—as opposed to saturating calculi (or bottom-up procedures) that try to deduce the formula to be proven from given axioms. In that sense, tableau calculi are always analytic; a stronger version of this property is defined in Section 3.5 (Def. 3.5.13).

Here, however, we use the word *analytic* in its traditional, more restricted sense. A calculus is analytic, if all formulae in a successor tableau of a tableau T occur as subformulae in T.

Definition 3.3.7 A tableau calculus is *analytic* if the following holds for each tableau T over a signature Σ^* and all its successor tableaux T': if $\mathsf{S}'{:}\sigma'{:}F'$ is a tableau formula in T', then there is a tableau formula $\mathsf{S}{:}\sigma{:}F$ in T such that F' is a subformula of F. $\square$

Analytic calculi have a smaller search space than non-analytic calculi because there are less different formulae that may be introduced by a tableau rule application.

It is possible to define analytic calculi for classical and most non-classical propositional logics. Calculi for predicate logics are usually not analytic in the strict sense, because they allow, for example, to deduce $p(t)$ from $(\forall x)(p(x))$ for all terms t. Such calculi are, however, still analytic in a weaker sense, namely if not only subformulae but as well instances of subformulae occurring in a tableau T may be introduced by a tableau rule application to T.

3.3.5 Monotonic Tableau Calculi

A tableau calculus with expansion rule is *monotonic* if the set of possible conclusions for a branch B' that is an expansion of a branch B contains all possible conclusions for B.

Definition 3.3.8 Let **L** be a logic, and let $\mathcal{C}$ be a tableau calculus for **L** with expansion rule $\mathcal{E}$.

The calculus $\mathcal{C}$ is *monotonic* if $\mathcal{E}(\Sigma)(B_1) \subset \mathcal{E}(\Sigma)(B_2)$ for all branches B_1 and B_2 over a signature Σ^* such that B_1 is an initial sub-path of B_2. $\square$

If a calculus with expansion rule is monotonic, then rule applications are permutable. That is, assumed B is a branch in a some tableau over a signature Σ, and C_1 and C_2 are conclusions in $\mathcal{E}(\Sigma)(B)$, then all tableau branches that can be constructed from B by first appending an extension E_1 from C_1 and then an extension E_2 from C_2 can as well be constructed permutating the corresponding tableau rule applications and extending B first by E_2 and then by E_1.

3.3.6 Non-structural Tableau Calculi

A tableau calculus is *non-structural* if the order of formulae on a tableau branch B is irrelevant for the result of applying the expansion rule to B.

Definition 3.3.9 Let **L** be a logic; and let $\mathcal{C}$ be a tableau calculus for **L** with expansion rule $\mathcal{E}$.

The calculus $\mathcal{C}$ is *non-structural* if $\mathcal{E}(\Sigma)(B_1) = \mathcal{E}(\Sigma)(B_2)$ for all branches B_1 and B_2 over Σ^* with $Form(B_1) = Form(B_2)$. □

3.3.7 Ideal Tableau Calculi

We call a tableau calculus *ideal* if it is a (a) calculus with expansion rule, (b) monotic, and (c) non-structural. Ideal calculi are—at least syntactically—well behaved. The idealness property will turn out to be of great importance in the following chapters.

Definition 3.3.10 A tableau calculus with expansion rule that is non-structural and monotonic is called *ideal*. □

If a tableau calculus is ideal, then its expansion rule can be represented as a function on finite sets of tableau formulae (a function on *premisses*).

Lemma 3.3.11 *Let $\mathcal{C}$ be an ideal tableau calculus with expansion rule $\mathcal{E}$ for a logic* **L**. *Then, for all signatures Σ, there is a (single) function $\tilde{\mathcal{E}}(\Sigma)$ that assigns to each finite set $\Pi \subset TabForm(\Sigma^*)$ of tableau formulae (each premiss) a set $\tilde{\mathcal{E}}(\Sigma)(\Pi)$ of (possible) conclusions such that*

$$\mathcal{E}(\Sigma)(B) = \tilde{\mathcal{E}}(\Sigma)(Form(B))$$

for all tableau branches B over Σ^*.

The above lemma implies that the function $\tilde{\mathcal{E}}(\Sigma)$ on sets of tableau formulae (premisses) can be seen as an alternative characterisation of the expansion rule $\mathcal{E}$—provided that the calculus is ideal. Therefore, in the case of ideal calculi, we identify the two functions, denote them both with $\mathcal{E}$, and call them “expansion rule”.

Another important feature of ideal calculi is that they are proof confluent.

Theorem 3.3.12 *If a tableau calculus is ideal (Def. 3.3.8), then it is proof confluent (Def. 3.3.3).*

Proof: Let $\mathcal{C}$ be a ideal calculus for a logic **L**; let $\mathfrak{F} \subset Form(\Sigma)$ be a set of formulae for which a tableau proof $T'_0, \ldots, T'_m$ ($m \geq 0$) exists; and let $T_0, \ldots, T_k$ ($k \geq 0$) be a sequence of tableaux for $\mathfrak{F}$.

A tableau proof that is an extension of $T_0, \ldots, T_k$ can be constructed as follows: for each branch B_i ($1 \leq i \leq r$) of T_k, the same m tableau rule applications that were used to construct T'_m from T'_0 are used to extend B_i such that T'_m is attached as a subtableau to the end of B_i. That is possible because the initial tableaux T_0 and T'_0 contain the same formulae (though they may not be identical) and the calculus is non-structural and monotonic. The resulting tableau proof is of length $1 + k + mr$. $\square$

In practice, expansion rules of ideal calculi are often described by means of rule schemata (see Sections 3.6 and 3.7 for examples). In these schemata, the elements of the minimal premiss, that have to be present to allow the deduction of a certain conclusion, and that conclusion are separated by a horizontal bar, while vertical bars in the conclusion denote different extensions. The use of schemata for defining tableau expansion rules fits perfectly in our framework, with the exception that different rule schemata are usually considered to define (or to *be*) different rules whereas we consider rule schemata to define different sub-cases of one (single) expansion rule.

Definition 3.3.13 Let $\mathcal{C}$ be an ideal calculus for a logic **L**; and let Σ be a signature in Sig.

A set $\Pi \subset TabForm(\Sigma)$ is a *minimal premiss* of a conclusion C if it is a premiss of C, i.e. $C \in \mathcal{E}(\Sigma)(\Pi)$, and there is no subset $\Pi' \subset \Pi$, $\Pi \neq \Pi'$ such that $C \in \mathcal{E}(\Sigma)(\Pi')$. $\square$

Idealness intuitively prohibits "strange" behaviour of calculi. As will become obvious in the following chapters, to be ideal is a very important property of tableau calculi. It is a pre-condition for the applicability of many of the uniform methods described in the following. Even "slight" non-idealness should be considered harmful. Unfortunately, many calculi described in the literature and used in practice are "slightly" non-ideal. Such calculi can often be repaired with minor changes. A typical example are calculi using expansion rules that introduce *new* symbols, i.e., symbols that must not occur on the branch or even the whole tableau. As the following example shows, this type of rules can often be replaced by similar rules not violating monotonicity.

Example 3.3.14 In calculi for first-order predicate logic, often an expansion rule is used that allows to derive $F(c)$ from formulae of the form $(\exists x)(F(x))$, where c is a constant *new* to the tableau (or the branch). A calculus using such a rule is not monotonic (and thus not ideal) because the rule demands the *absence* of formulae containing c.

If instead a special constant symbol c_F is used, which does not have to be new, then the calculus is monotonic. Soundness is preserved provided that c_F is not introduced into the tableau in any other way than by skolemising $(\exists x)(F(x))$ (in particular, Skolem constants c_F must not occur in the initial tableau); see Section 4.4. $\square$

Ideal calculi exist for most logics. There are, however, some non-classical logics whose inherent properties make it difficult or even impossible to define an ideal tableau calculus; these include non-monotonic logics (since their calculi are not monotonic) and logics with resource restrictions such as linear and relevance logic (as rule applications in calculi for these logics have non-local effects).

The idealness property, in particular monotonicity, is often dropped and violated to remove redundancies when a calculus is turned into a proof procedure (see Chapter 5). That is not harmful as long as the additional restrictions that are imposed are clearly end explicitly separated from the definition of the calculus.

3.3.8 Continuity

An ideal calculus is in particular monotonic. Therefore, all conclusions that can be derived (separately) from any two premisses Π, Π' can as well be derived from their union, i.e., $\mathcal{E}(\Pi) \cup \mathcal{E}(\Pi') \subset \mathcal{E}(\Pi \cup \Pi')$. If, moreover, $\mathcal{E}(\Pi \cup \Pi') \subset \mathcal{E}(\Pi) \cup \mathcal{E}(\Pi')$ and thus $\mathcal{E}(\Pi) \cup \mathcal{E}(\Pi') = \mathcal{E}(\Pi \cup \Pi')$, then a calculus is said to be *continuous* (for Π, Π').

Virtually no calculus is continuous for all premisses. If a conclusion C has a minimal premiss Π consisting of more than one tableau formula, i.e., $\Pi = \Pi' \cup \Pi''$ where Π' and Π'' are both not empty, then the calculus is not continuous w.r.t. the subsets Π' and Π'' (because $C \notin \mathcal{E}(\Pi') \cup \mathcal{E}(\Pi'')$ as the premiss Π is minimal). Therefore, the notion of continuity is defined w.r.t. a certain premiss.

Definition 3.3.15 Let $\mathcal{C}$ be a ideal calculus for a logic **L**; let $\Sigma \in Sig$ be a signature; and $\Pi \subset TabForm(\Sigma^*)$ be a premiss.

The tableau rule of $\mathcal{C}$ is *continuous* w.r.t. Π if, for all premisses $\Pi' \subset TabForm(\Sigma^*)$:

$$\mathcal{E}(\Pi \cup \Pi') \subset \mathcal{E}(\Pi) \cup \mathcal{E}(\Pi') \ .$$

□

Example 3.3.16 The tableau rule of the calculus for first-order predicate logic described in Section 3.6 is continuous w.r.t. all premisses consisting solely of non-atomic formulae. The rule is not continuous w.r.t. premisses containing atoms because branch closure involves two complementary atoms; i.e., the minimal premiss for the conclusion $\{\{\bot\}\}$ consists of more than one tableau formula.

A tableau rule that allows to "apply" equalities, i.e., to derive $F(s)$ from $F(t)$ and the equality $s \approx t$, is not continuous w.r.t. any (non-empty) premiss. □

Continuity is a very useful property, because a premiss w.r.t. which the tableau rule is continuous can be "delete" once all its conclusions have been added to the branch resp. the tableau, i.e., it has not to be considered again; continuity is, thus, closely related to the semantical property of invertibility (Def. 3.5.11).

3.4 Semantics of Tableaux

The semantics of tableaux of a calculus for a logic $\mathbf{L}$ is based on the semantics of $\mathbf{L}$, which is given by sets $\mathcal{M}(\Sigma)$ of Kriple-style models for each signature Σ. The labels that are part of tableau formulae are assumed to represent worlds in models, and the truth value signs encode truth and falsehood of a formula.

Definition 3.4.1 Let $\mathcal{C}$ be a tableau calculus for a logic $\mathbf{L}$; and let $\Sigma \in \mathit{Sig}$ be a signature. A *tableau interpretation* for $\mathcal{C}(\Sigma)$ is a pair $\langle \mathbf{m}, I \rangle$ where

- $\mathbf{m} \in \mathcal{M}(\Sigma^*)$ is a model for the extended signature Σ^*, and
- I is a partial function that assigns to labels $\sigma \in \mathit{Lab}(\Sigma)$ worlds of $\mathbf{m}$ such that $I(\sigma^0) = w^0$ (i.e., I assigns to the initial label σ^0 the initial world w^0 of $\mathbf{m}$).

A tableau formula $\mathsf{S}{:}\sigma{:}F \in \mathit{TabForm}(\Sigma^*)$ is *satisfied* by $\langle \mathbf{m}, I \rangle$ iff

1. $I(\sigma)$ is defined, and
2. (a) $\mathsf{S} = \mathsf{T}$ and F is true in $I(\sigma)$, or
   (b) $\mathsf{S} = \mathsf{F}$ and F is false in $I(\sigma)$.

The tableau formula $\perp$ is *not* satisfied by any tableau interpretation.

A tableau branch B is *satisfied* by $\langle \mathbf{m}, I \rangle$ iff *all* tableau formulae on B are satisfied by $\langle \mathbf{m}, I \rangle$.

A tableau is *satisfied* by $\langle \mathbf{m}, I \rangle$ iff *at least one* of its branches is satisfied by $\langle \mathbf{m}, I \rangle$. $\square$

Note, that a tableau formula is satisfied by default if the interpretation function I is not defined for its label.

Often, it does not make sense to use all possible tableau interpretations to define the semantics of tableaux. An appropriate subset of the tableau interpretations has to be chosen; that choice depends on the particular calculus and the logic. On the one hand, to be useful for proving soundness of the calculus, the subset must only contain tableau interpretations for which the tableau rule preserves satisfiability (and can be proven to preserve satisfiability). On the other hand, to be useful for proving completeness, the subset must contain enough interpretations such that every satisfiable tableau is satisfied by an interpretation contained in the subset.

Definition 3.4.2 Let $\mathcal{C}$ be a tableau calculus for a logic $\mathbf{L}$. The semantics of $\mathcal{C}$ is, for each signature Σ, defined by a set $\mathit{TabInterp}(\Sigma^*)$ of tableau interpretations. $\square$

Example 3.4.3 To define the semantics of tableaux of calculi for first-order predicate logic, only tableau interpretations are used whose first part is an Herbrand model. $\square$

3.5 Semantical Properties

3.5.1 The Advantage of Semantical Properties

Semantical properties are needed, because even strong syntactical restrictions such as idealness still allow calculi to behave “strangely”. Formulae could be added to the tableau that syntactically encode knowledge derived from the premiss Π, but whose semantics (i.e., truth value) has nothing to do with that of the formulae in Π. An extreme example for this is that two symbols of the signature are used to encode the formulae in Π in a binary representation, and a tableau rule is employed that operates on that binary representation. It is impossible to apply any uniform methods to such calculi—though they may be sound and complete—because an understanding of the encoding would be required. To assure a more “conservative” behaviour one could impose additional syntactical restrictions, such as requiring a calculus to be analytic. However, the property of tableau rules that makes it possible to apply techniques such as fibring (Chapter 6) in a uniform way is essentially semantical: the result of a rule application must be semantically related to its premiss.

Two important semantical properties are already part of the definition of tableau interpretations (Def. 3.4.1), namely that the labels in tableau formulae represent worlds in models, and that the truth value signs encode truth and falsehood of a formula; signs and labels do not contain other information.

Defining a semantics for tableaux is not only important for proving soundness and completeness of a calculus; in addition, pre-conditions of many search space restrictions and other useful techniques are semantical. To replace such uniform semantical pre-conditions by uniform syntactical pre-conditions is often difficult (if not impossible). That notwithstanding, a tableau calculus can be quite useful, in particular sound and complete, if no semantics for tableaux is provided.

3.5.2 Soundness and Completeness

The most important semantical properties of tableau calculi are soundness and completeness:

Definition 3.5.1 A calculus $\mathcal{C}$ for a logic **L** is *sound* if, for all signatures $\Sigma \in Sig$ and all finite sets $\mathfrak{F} \subset Form(\Sigma)$ of formulae:

If there is a tableau proof for $\mathfrak{F}$, then $\mathfrak{F}$ is unsatisfiable.

A calculus $\mathcal{C}$ for a logic **L** is *complete* if, for all signatures $\Sigma \in Sig$ and all finite sets $\mathfrak{F} \subset Form(\Sigma)$ of formulae:

If $\mathfrak{F}$ is unsatisfiable, then there is a tableau proof for $\mathfrak{F}$.

□

3.5.3 Soundness Ensuring Properties

There are three important soundness properties; together they are sufficient to ensure soundness of a calculus.

1. An initial tableau for a satisfiable set of formulae is satisfiable;
2. satisfiability is preserved by tableau rule applications;
3. a closed tableau is unsatisfiable.

Together these properties imply that there is no tableau proof for a satisfiable set of formulae. The last of the properties does not have to be checked, as all calculi have it (according to the following lemma).

Lemma 3.5.2 *A closed tableau is unsatisfiable.*

Proof: If a tableau T is closed, then all its branches are closed, which means that they contain $\bot$. Because $\bot$ is unsatisfiable, none of the branches of T is satisfied by any tableau interpretation; therefore T is not satisfied by any tableau interpretation. $\square$

Definition 3.5.3 Let $\mathcal{C}$ be a calculus for a logic $\mathbf{L}$. Then the following *soundness properties* are defined that $\mathcal{C}$ may have:

Soundness Property 1 ([weak] appropriateness of the set of tableau interpretations): For all signatures Σ, if a set $\mathfrak{F} \subset \mathit{Form}(\Sigma)$ is satisfiable, then there is a tableau interpretation in $\mathit{TabInterp}(\Sigma^*)$ that satisfies the initial tableau for $\mathfrak{F}$.

Soundness Property 2 ([weak] soundness of expansion): For all signatures Σ, if there is a tableau interpretation in $\mathit{TabInterp}(\Sigma^*)$ satisfying a tableau T and T' is a successor tableau of T, then there is a tableau interpretation in $\mathit{TabInterp}(\Sigma^*)$ satisfying T'. $\square$

A tableau calculus that has the two soundness properties from Definition 3.5.3 can be proven to be sound (whether it is ideal or not).

Theorem 3.5.4 *A tableau calculus $\mathcal{C}$ for a logic $\mathbf{L}$ that has the two soundness properties from Definition 3.5.3 (appropriateness of the set of tableau interpretations and soundness of expansion) is sound (Def. 3.5.1).*

Proof: Let $\mathfrak{F} \subset \mathit{Form}(\Sigma)$ be a finite, satisfiable set of formulae. We prove by contradiction that there is no tableau proof for $\mathfrak{F}$, which implies soundness of the calculus.

Suppose that there is a tableau proof $T_0, \ldots, T_n$ ($n \geq 0$) for $\mathfrak{F}$. Since T_n is closed, it is unsatisfiable (Lemma 3.5.2).

However, since $\mathfrak{F}$ is satisfiable, by Property 1 (appropriateness of the set of tableau interpretations), there is a tableau interpretation $\langle \mathbf{m}, I \rangle \in \mathit{TabInterp}(\Sigma^*)$ satisfying the initial tableau T_0. Therefore, by induction on i and using Soundness Property 2 (soundness of expansion), all the tableaux T_i ($1 \leq i \leq n$) are satisfied by some tableau interpretation in $\mathit{TabForm}(\Sigma^*)$, contradicting the fact that T_n is unsatisfiable.

Thus, the assumption is wrong, and there is indeed no tableau proof for $\mathfrak{F}$. □

3.5.4 Completeness Ensuring Properties

Before properties can be formulated that establishes completeness of a calculus, the notion of a *fully expanded* tableau branch has to be defined. The definition relies on the fact that the calculus has an expansion rule (Def. 3.3.4) and is monotonic (Def. 3.3.8); without these properties, it is difficult to define the notion of fully expanded branches in a uniform way. Intuitively a branch B is fully expanded if each possible rule application creates at least one successor branch B' that contains the same formulae as B.

Definition 3.5.5 Let $\mathcal{C}$ be a monotonic tableau calculus with expansion rule for a logic **L**; and let $\Sigma \in \mathit{Sig}$ be a signature.

A (possibly infinite) tableau branch B is *fully expanded* if $E \subset \mathit{Form}(B)$ for at least one extension E in each conclusion $C \in \mathcal{R}(\Sigma)(B)$. □

Definition 3.5.6 Let $\mathcal{C}$ be a calculus for a logic **L**. Then, the following *completeness properties* are defined that $\mathcal{C}$ may have:

Completeness Property 1 ([weak] appropriateness of the set of tableau interpretations): For all signatures Σ, if there is a tableau interpretation in $\mathit{TabInterp}(\Sigma^*)$ satisfying the initial tableau for a set $\mathfrak{F} \subset \mathit{Form}(\Sigma)$ of formulae, then there is a model in $\mathcal{M}(\Sigma)$ satisfying $\mathfrak{F}$.

Completeness Property 2 (satisfiability of fully expanded branches): For all signatures Σ, if a tableau branch B is fully expanded and not closed, then there is a tableau interpretation in $\mathit{TabInterp}(\Sigma^*)$ satisfying B. □

The completeness properties from Definition 3.5.6 are sufficient to ensure completeness of an *ideal* calculus. As said at the end of Section 3.3.7, idealness is often dropped when an ideal calculus $\mathcal{C}$ is turned into an efficient proof procedure. That is not harmful, because the completeness of these non-ideal procedures follows from the completeness of $\mathcal{C}$ when the uniform methods for turning a calculus into an efficient proof procedure from Chapter 5 are employed.

Theorem 3.5.7 *A tableau calculus $\mathcal{C}$ for a logic $\mathbf{L}$ that is ideal and has the two completeness properties from Definition 3.5.6 (appropriateness of the set of tableau interpretations and satisfiability of fully expanded branches) is complete.*

Proof: Let $\mathfrak{F} \subset Form(\Sigma)$ be a finite, unsatisfiable set of formulae; we proceed to prove that there is a tableau proof for $\mathfrak{F}$. Let $(T_n)_{n \geq 0}$ be a (possibly infinite) sequence of tableaux for $\mathfrak{F}$ such that

1. T_0 is an initial tableau for $\mathfrak{F}$;
2. T_i is a successor tableau of T_{i-1} ($i \geq 0$);
3. the sequence is constructed in a fair way, i.e., all possible conclusions in $\mathcal{E}(\Pi)$ for all premisses Π occurring in one of the T_i is sooner or later used to expand every non-closed branch on which Π occurs (where corresponding branches in T_i and T_{i+j} are identified).

Because of the monotonicity and non-destructiveness of $\mathcal{C}$ such a sequence can be constructed for all formula sets $\mathfrak{F}$.

The sequence $(T_n)_{n \geq 0}$ approximates an infinite tree T_∞; according to its construction, every non-closed branch in T_∞ is fully expanded (Def. 3.5.5). Suppose there is a non-closed (and thus fully expanded) branch B in T_∞. Then, by the satisfiability of fully expanded branches (Property 2 in Def. 3.5.6), B is satisfied by a tableau interpretation $\langle \mathbf{m}^*, I \rangle \in \mathit{TabInterp}(\Sigma^*)$. Since all the formulae of the initial tableau T_0 are on B, the initial tableau is satisfied by $\langle \mathbf{m}^*, I \rangle$ as well. Therefore, by the appropriateness of the set of tableau interpretations (Property 1 in Def. 3.5.6), there is a model $\mathbf{m} \in \mathcal{M}(\Sigma)$ satisfying $\mathfrak{F}$, in contradiction to the fact that $\mathfrak{F}$ is unsatisfiable. Thus, the assumption that there is a non-closed branch in T_∞ is wrong, and indeed all branches contain a node labelled with $\perp$. Then, because (by definition) premisses are finite and the tableaux are finitely branching, König's Lemma implies that there is an $n \geq 0$ such that already the finite sub-tableau T_n of T_∞ is closed. This concludes the proof, because T_n is by construction a closed tableau for $\mathfrak{F}$. □

3.5.5 Strong Soundness and Completeness Properties

A stronger version of the soundness properties from Definition 3.5.3 can be formulated. The weak properties only require satisfiability of the initial tableau and that satisfiability is preserved by tableau rule applications, but allow that at each step a different model resp. tableau interpretation may be chosen. If a calculus has the stronger properties defined below, and the formulae for which a tableau proof is to be constructed are satisfied by a model $\mathbf{m}$, then the initial tableau and all subsequent tableaux are satisfied by one and the same tableau interpretation, which is an extension of $\mathbf{m}$.

These stronger soundness properties are not needed for proving soundness but are important for the fibring technique (Chapter 6).

Definition 3.5.8 Let $\mathcal{C}$ be a calculus for a logic $\mathbf{L}$. Then the following *strong soundness properties* are defined that $\mathcal{C}$ may have:

Strong Soundness Property 1 (appropriateness of the set of tableau interpretations): For all signatures Σ, if a set $\mathfrak{F} \subset \mathit{Form}(\Sigma)$ is satisfied by a model $\mathbf{m} \in \mathcal{M}(\Sigma)$, then there is a tableau interpretation $\langle \mathbf{m}^*, I \rangle \in \mathit{TabInterp}(\Sigma^*)$ satisfying the initial tableau for $\mathfrak{F}$ where $\mathbf{m}^*$ is an extension of $\mathbf{m}$.

Strong Soundness Property 2 soundness of expansion: For all signatures Σ, if a tableau T is satisfied by a tableau interpretation in $\mathit{TabInterp}(\Sigma^*)$ and T' is a successor tableau of T, then T' is satisfied by *the same* tableau interpretation. □

A calculus has the strong soundness of expansion property defined above (and, thus, the weaker property from Def. 3.5.3) if and only if the conclusions C in $\mathcal{E}(\Sigma)(B)$ are logical consequences of the formulae on B. An *ideal* calculus has this property if and only if a conclusion is a logical consequence of any of its minimal premisses:

Lemma 3.5.9 *An ideal calculus $\mathcal{C}$ for a logic $\mathbf{L}$ has the strong soundness of expansion property (Property 2 from Def. 3.5.8) if, for all signatures $\Sigma \in \mathit{Sig}$, all premisses $\Pi \subset \mathit{TabForm}(\Sigma^*)$, and all conclusions C such that Π is a (minimal) premiss of C the following holds:*

If a tableau interpretation satisfies Π, then it satisfies an extensions $E \in C$.

Proof: For the if-part of the proof, let $\langle \mathbf{m}, I \rangle \in \mathit{TabInterp}(\Sigma^*)$ be a tableau interpretation satisfying a tableau T and let T' be a successor tableau of T; let B be the branch in T that is expanded, and let $\Pi \subset \mathit{Form}(B)$ be a minimal premiss of the conclusion C that is used to expand B.

By assumption $\langle \mathbf{m}, I \rangle$ satisfies T; thus it satisfies some branch B^0 of T. If B^0 is different from B, then B^0 is also a branch of T' and we are through.

If, on the other hand, $B^0 = B$ and, thus, $\langle \mathbf{m}, I \rangle$ satisfies $\Pi \subset \mathit{Form}(B)$, then it satisfies one of the extensions $E \in C$. Therefore $\langle \mathbf{m}, I \rangle$ satisfies the branch of T' that has been constructed by extending B with the formulae E and, thus, it satisfies the tableau T'.

For the only-if-part, consider a tableau T that consists of a single branch with the formulae in a premiss Π. If Π is satisfied then so is T, which implies that the successor tableau T' is satisfied that is constructed from T using the conclusion C. But, by construction, T' can only be satisfied if one of the extensions in C is satisfied. $\square$

A stronger version of Completeness Property 1 (appropriateness of the set of tableau interpretations) can be defined as well.[1]

Definition 3.5.10 Let $\mathcal{C}$ be a calculus for a logic $\mathbf{L}$. Then, the following *strong completeness property* is defined that $\mathcal{C}$ may have:

Strong Completeness Property 1 (appropriateness of the set of tableau interpretations): For all signatures Σ, if $\langle \mathbf{m}^*, I\rangle \in \mathit{TabForm}(\Sigma^*)$ satisfies the initial tableau for a set $\mathfrak{F} \subset \mathit{Form}(\Sigma)$ of formulae, then $\mathbf{m}^*$ can be restricted to a model $\mathbf{m} \in \mathcal{M}(\Sigma)$ that satisfies $\mathfrak{F}$. $\square$

3.5.6 Invertible Expansion Rules

If the converse of the pre-condition of Lemma 3.5.9 holds, i.e., if each conclusion logically implies its minimal premiss, then a tableau rule is said to be *invertible*.

Definition 3.5.11 Let $\mathcal{C}$ be a ideal calculus for a logic $\mathbf{L}$; let $\Sigma \in \mathit{Sig}$ be a signature; and let $\Pi \subset \mathit{TabForm}(\Sigma^*)$ be a premiss.

The tableau rule of $\mathcal{C}$ is *invertible* w.r.t. Π if, for all conclusions $C \in \mathcal{E}(\Sigma)(\Pi)$ such that Π is a minimal premiss of C, the following holds:

If a tableau interpretation satisfies an extension $E \in C$, then it satisfies Π.

The calculus $\mathcal{C}$ is said to be *invertible* if its tableau rule is invertible for all signatures Σ and all premisses $\Pi \subset \mathit{TabForm}(\Sigma^*)$. $\square$

It is of advantage if a tableau rule is invertible, because then rule applications not only preserve satisfiability but they preserve *unsatisfiability* as well. This is important for constructing efficient proof procedures as it allows to "delete" the minimal premiss of a conclusion C from a branch that has been extended using C.

Example 3.5.12 The tableau rule of a calculus for PL1 that allows to derive the conclusion $\{\{\mathsf{T}{:}\sigma{:}F\}, \{\mathsf{T}{:}\sigma{:}G\}\}$ from the premiss $\Pi_1 = \{\mathsf{T}{:}\sigma{:}(F \vee G)\}$ is invertible with respect to Π_1 because every first-order interpretation that satisfies one of the formulae F and G satisfies $F \vee G$ as well.

However, the tableau rule that allows to derive the conclusion $\{\{\mathsf{T}{:}\sigma{:}p(t)\}\}$ from the premiss $\Pi_2 = \{\mathsf{T}{:}\sigma{:}(\forall x)(p(x))\}$ is *not* invertible w.r.t. Π_2, because an interpretation satisfying $p(t)$ does not necessarily satisfy $(\forall x)(p(x))$. $\square$

[1] A stronger version of Completeness Property 2 is not defined along the same line, as that property does not involve a transition from one model or tableau interpretation to another. In Section 3.5.7, the property of being semantically analytic is defined, which strengthens Completeness Property 2 in a different way.

3.5.7 Semantically Analytic Tableau Calculi

A stronger version of Completeness Property 2 in Definition 3.5.6 (satisfiability of fully expanded branches) can be used to ensure that a calculus is "analytic down to the atomic level". This is *not* a syntactical property and it does *not* imply that the calculus is analytic in the classical sense (Def. 3.3.7).

Definition 3.5.13 A calculus $\mathcal{C}$ for a logic $\mathbf{L}$ is (weakly) *semantically analytic* if, for all signatures Σ, the following holds: If a tableau branch B is fully expanded and not closed, then every tableau interpretation in $\mathit{TabInterp}(\Sigma^*)$ satisfying the *atoms* on B satisfies *all* formulae on B, and at least one such tableau interpretation exists. □

Example 3.5.14 If a tableau calculus for a modal logic is to be semantically analytic, it has to be possible to extend a tableau branch containing the formula $\mathsf{T}{:}\sigma{:}\Box p$ by the formulae $\mathsf{T}{:}\tau{:}p$ for all labels τ representing a world reachable from the world represented by σ. □

Example 3.5.15 In a tableau calculus for classical propositional logic that is to be semantically analytic, it must be possible to expand a branch containing $\mathsf{T}{:}(p \vee q)$ by sub-branches containing $\mathsf{T}{:}p$ and $\mathsf{T}{:}q$, respectively. Thus, the calculus is not semantically analytic any more, if the restriction from Section 5.5 is imposed that forbids to use formulae for expansion that are not *connected* to other formulae. □

If a calculus is semantically analytic, then the atoms on a fully expanded branch B explicitly represent all information about the tableau interpretations satisfying B that can be derived from the (complex) formulae on B.

However, in certain logics $\mathbf{L}$, there may be hidden (implicit) restrictions on the form of models that are not specific for models satisfying certain formulae but apply to all models of $\mathbf{L}$ and, thus, to all tableau interpretations used to define the semantics of tableaux. If, for example, a calculus is used for fibring (Chapter 6), then it is important that such hidden information, too, is explicitly represented by the atoms on a fully expanded branch. This property is formalised as follows:

Definition 3.5.16 A calculus $\mathcal{C}$ for a logic $\mathbf{L}$ is *strongly semantically analytic* if, for all signatures Σ, the following holds:

If

1. B is a fully expanded tableau branch that is not closed, and
2. $\Phi \subset \mathit{TabForm}(\Sigma^*)$ is a set of *atomic* tableau formulae such that, for *no* ϕ in Φ, both ϕ and $\overline{\phi}$ are in $\mathit{Form}(B) \cup \Phi$,

then there is a tableau interpretation $\langle \mathbf{m}, I\rangle$ in $\mathit{TabInterp}(\Sigma^*)$ such that

1. $\langle \mathbf{m}, I\rangle$ satisfies $\mathit{Form}(B) \cup \Phi$,
2. for all worlds w in $\mathbf{m}$ there is a label σ in $\mathcal{C}$ with $I(\sigma) = w$. □

Example 3.5.17 Kripke-style models that are used to define the semantics of intuitionistic logic have to satisfy the (hidden) restriction that, if a formula is true in some world w, then it is true in all successor worlds of w (if a formula is false in a world w, it may be true or false in the successor worlds of w).

Thus, if a calculus for intuitionistic logic is to be strongly semantically analytic, then it must be possible to derive $\mathsf{T}{:}\tau{:}G$ from $\mathsf{T}{:}\sigma{:}G$ for all labels τ that represent successor worlds of the world represented by σ. □

For most logics, including classical and modal logics, the properties of being weakly resp. strongly semantically analytic coincide, i.e., a calculus that is weakly semantically analytic is strongly semantically analytic as well.

3.6 An Ideal Tableau Calculus for PL1

3.6.1 Syntax

To describe our calculus $\mathcal{C}_{\mathrm{PL1}}$ for first-order predicate logic PL1, we have to define, for each signature $\Sigma \in \mathit{Sig}_{\mathrm{PL1}}$, the extension Σ^* to be used for constructing tableaux, the set of labels, the initial label, and the tableau rule.

Extended Signatures For skolemisation we use a set $F^{sko}(\Sigma)$ of function symbols that is disjoint from $F(\Sigma)$ and contains infinitely many symbols of each arity $n \geq 0$. The extension of a signature $\Sigma = \langle P(\Sigma), F(\Sigma), \alpha(\Sigma)\rangle$ is thus

$$\Sigma^* = \langle P(\Sigma), F(\Sigma) \cup F^{sko}(\Sigma), \alpha(\Sigma) \cup \alpha^{sko}(\Sigma)\rangle \ .$$

Labels The models of first-order logic consist of only one world. We use the label $*$ to represent this single world. Thus, $\mathit{Lab}(\Sigma) = \{*\}$ for all signatures Σ, and $*$ is the initial label. To simplify notation, the abbreviation $\mathsf{S}{:}G$ is used for tableau formulae of first-order predicate logic, i.e., the label $*$ is omitted.

α	α_1,	α_2
T:$*$:$(F \land G)$	T:$*$:F,	T:$*$:G
F:$*$:$(F \lor G)$	F:$*$:F,	F:$*$:G
F:$*$:$(F \to G)$	T:$*$:F,	F:$*$:G
T:$*$:$\neg F$	F:$*$:F,	F:$*$:F
F:$*$:$\neg F$	T:$*$:F,	T:$*$:F

β	β_1,	β_2
T:$*$:$(F \lor G)$	T:$*$:F,	T:$*$:G
F:$*$:$(F \land G)$	F:$*$:F,	F:$*$:G
F:$*$:$(F \to G)$	F:$*$:F,	T:$*$:G

$\gamma(x)$	$\gamma_1(x)$
T:$*$:$(\forall x)(F(x))$	T:$*$:$F(x)$
F:$*$:$(\exists x)(F(x))$	F:$*$:$F(x)$

$\delta(x)$	$\delta_1(x)$
F:$*$:$(\forall x)(F(x))$	F:$*$:$F(x)$
T:$*$:$(\exists x)(F(x))$	T:$*$:$F(x)$

Table 3.1: The four formula types of first-order predicate logic.

Tableau Rule We define an ideal calculus, i.e., a calculus with expansion rule $\mathcal{E}$.

The set of formulae in $Form(\Sigma^*) = Form^0_{\mathrm{PL1}}(\Sigma^*)$ that are not atomic is divided into four classes (Table 3.1): α for formulae of conjunctive type, β for formulae of disjunctive type, γ for quantified formulae of universal type, and δ for quantified formulae of existential type (unifying notation).

Notation 3.6.1 The letters α, β, γ, and δ are used to denote formulae of (and only of) the appropriate type. In the case of γ- and δ-formulae the object variable x bound by the (top-most) quantifier is made explicit by writing $\gamma(x)$ and $\gamma_1(x)$ (resp. $\delta(x)$ and $\delta_1(x)$); accordingly, $\gamma_1(t)$ denotes the result of replacing all occurrences of x in γ_1 by t. □

In Table 3.2, the expansion rule of the calculus $\mathcal{C}_{\mathrm{PL1}}$ is given schematically for the various formula types.

To preserve monotonicity of the calculus, we use a schema for δ-formulae that, for constructing the Skolem term, does not introduce a *new* Skolem function symbol. Rather, each equivalence class of δ-formulae identical up to variable renaming and replacement of ground terms may be assigned the same Skolem function symbol. (This is an adaptation of the δ^{++}-rule described in (Beckert *et al.*, 1993) to the ground case.)

Definition 3.6.2 Given a signature $\Sigma \in Sig_{\mathrm{PL1}}$, a *Skolem term assignment* is a function sko assigning to each δ-formula $\phi \in TabForm_{\mathrm{PL1}}(\Sigma^*)$ a term

$$sko(\phi) = f(t_1, \ldots, t_k) \in Term^0_{\mathrm{PL1}}(\Sigma^*)$$

such that

1. (a) $f \in F^{sko}(\Sigma)$,

$$\frac{\alpha}{\begin{array}{c}\alpha_1\\ \alpha_2\end{array}} \qquad \frac{\beta}{\beta_1 \mid \beta_2} \qquad \frac{\gamma(x)}{\gamma_1(t)} \qquad \frac{\delta(x)}{\delta_1(t)} \qquad \frac{\begin{array}{c}\mathsf{T}{:}{*}{:}G\\ \mathsf{F}{:}{*}{:}G\end{array}}{\perp}$$

where t is any variable-free term (for γ); where $t = sko(\delta)$ (for δ); where G is atomic (for closure).

Table 3.2: Rule schemata for first-order predicate logic.

(b) $k = \alpha^{sko}_{\Sigma}(f)$, and

(c) $t_1, \dots, t_k$ are (sub-)terms occurring in ϕ;[2]

2. for all $f' \in F^{sko}(\Sigma)$, if f' occurs in ϕ, then $f > f'$ where $>$ is an arbitrary but fixed ordering on $F^{sko}(\Sigma)$; and

3. for all δ-formulae $\psi \in Form^0_{\mathrm{PL1}}(\Sigma^*)$, if $sko(\psi) = f(t'_1, \dots t'_k)$, then ϕ and ψ are identical up to renaming of bound variables and up to replacing occurrences of terms t_i by t'_i ($1 \le i \le k$). □

The purpose of Condition 2 in the above definition of sko is to avoid cycles like: $sko(\phi) = f(t_1, \dots, t_k)$, the symbol f occurs in ψ, $sko(\psi) = f'(t'_1, \dots, t'_l)$, and f' occurs in ϕ.

According to Condition 3, it is allowed to use the same Skolem function symbol for an equivalence class of δ-formulae that are identical up to (a) renaming of bound variables and (b) replacement of variable-free terms; the variable-free terms that may be replaced have to be made arguments of the Skolem term.

Actually, it is not necessary to use complex terms for skolemisation. It is sufficient to (uniquely) assign to each δ-formula (resp. class of δ-formulae that are identical up to renaming of bound object variables) a Skolem *constant* (observing Condition 2 in Def. 3.6.2). The possibility to use Skolem *terms* plays, however, an important rôle for lifting, i.e., constructing the free variable version of the calculus $\mathcal{C}_{\mathrm{PL1}}$ (see Section 4.2.10), as the following example illustrates:

Example 3.6.3 The δ-formulae $\delta^t = \mathsf{T}{:}(\exists x)(p(t, x))$ may be assigned the same Skolem function symbol f using the Skolem *terms* $sko(\delta^t) = f(t)$ (for all terms t), in which case the expansion rule is liftable for premisses containing these δ-formulae.

If, on the other hand, Skolem *constants* $sko(\delta^t) = c^t$ are used, the expansion rule is not liftable, as $\delta^t[t \mapsto t'] = \delta^{t'}$ but $\delta^t_1[t \mapsto t'] \neq \delta^{t'}_1$. □

We now formally define the expansion (and closure) rule $\mathcal{E}_{\mathrm{PL1}}$ of the calculus $\mathcal{C}_{\mathrm{PL1}}$:

[2] Note that the terms t_i ($1 \le i \le k$) must be elements of $Term^0_{\mathrm{PL1}}(\Sigma)$, i.e, do not contain object variables.

Definition 3.6.4 For all signatures Σ and all premisses $\Pi \subset \mathit{TabForm}_{\mathrm{PL1}}(\Sigma^*)$, the set $\mathcal{E}_{\mathrm{PL1}}(\Sigma)(\Pi)$ of possible conclusions is the smallest set containing the following conclusions (where $\alpha, \beta, \gamma, \delta$ denote formulae of the corresponding type):

– $\{\{\alpha_1, \alpha_2\}\}$	for all $\alpha \in \Pi$,
– $\{\{\beta_1\}, \{\beta_2\}\}$	for all $\beta \in \Pi$,
– $\{\{\gamma_1(t)\}\}$	for all $\gamma \in \Pi$ and all terms $t \in \mathit{Term}^0_{\mathrm{PL1}}(\Sigma^*)$,
– $\{\{\delta(t)\}\}$	for all $\delta \in \Pi$ where $t = \mathit{sko}(\delta)$ (Def. 3.6.2),
– $\{\{\bot\}\}$	if $\mathsf{T}{:}\sigma{:}G, \mathsf{F}{:}\sigma{:}G \in \Pi$ for any atom $G \in \mathit{Atom}_{\mathrm{PL1}}(\Sigma^*)$.

□

Note, that the formulae in $\mathit{Form}_{\mathrm{PL1}}(\Sigma) = \mathit{Form}^0_{\mathrm{PL1}}(\Sigma)$ are first-order sentences, i.e., do not contain free object variables, and that the tableau expansion rule of $\mathcal{C}_{\mathrm{PL1}}$ does not introduce any free object variables either.

3.6.2 Semantics

To define the semantics of the tableaux of $\mathcal{C}_{\mathrm{PL1}}$, we use the set $\mathit{TabInterp}_{\mathrm{PL1}}(\Sigma^*)$ that contains all *canonical* tableau interpretations:

Definition 3.6.5 A tableau interpretation $\mathbf{m} = \langle\langle D, \mathcal{I}\rangle, I\rangle$ for a signature Σ of PL1 is *canonical* if:

1. $D = \mathit{Term}^0_{\mathrm{PL1}}(\Sigma^*)$.
2. For all δ-formulae $\delta(x) \in \mathit{TabForm}(\Sigma^*)$:
$$\text{if } \mathit{val}_{\mathcal{I}}(\delta(x)) = \mathit{true}, \text{ then } \mathit{val}_{\mathcal{I}}(\delta_1(t)) = \mathit{true},$$
where $t = \mathit{sko}(\delta)$.
3. $I(*) = w^0 = \langle D, \mathcal{I}\rangle$. □

Intuitively, in a canonical tableau interpretation, the function $\mathcal{I}$ assigns to Skolem terms $t = \mathit{sko}(\delta)$ an element for which the formula δ holds; in addition, the label $*$ is interpreted by I in the right way.

3.6.3 Soundness and Completeness

Using the set $\mathit{TabInterp}_{\mathrm{PL1}}(\Sigma^*)$ of canonical tableau interpretations as defined in Def. 3.6.5, the calculus $\mathcal{C}_{\mathrm{PL1}}$ has the soundness and completeness properties from Definitions 3.5.3 and 3.5.6. In particular, if a tableau is satisfied by a canonical tableau interpretation, then all its successor tableaux are satisfied by the same interpretation;

and every fully expanded tableau branch that is not closed is satisfied by a canonical interpretation.

Before we prove that the calculus $\mathcal{C}_{\mathrm{PL1}}$ is sound and complete using the criteria from Sections 3.5.3 and 3.5.4, we formulate and prove the appropriate version of Hintikka's Lemma.

Definition 3.6.6 A set $\Xi \subset \mathit{TabForm}_{\mathrm{PL1}}(\Sigma^*)$ of tableau formulae is a *Hintikka set* if it satisfies the following conditions:

1. There are no complementary atomic formulae $\mathsf{T}{:}G$, $\mathsf{F}{:}G$ in Ξ.
2. If $\alpha \in \Xi$, then α_1 and α_2 are in Ξ.
3. If $\beta \in \Xi$, then β_1 or β_2 is in Ξ.
4. If $\gamma(x) \in \Xi$, then $\gamma_1(t) \in \Xi$ for all ground terms t in $\mathit{Term}^0_{\mathrm{PL1}}(\Sigma^*)$.
5. If $\delta(x) \in \Xi$, then $\delta_1(t) \in \Xi$ where $t = \mathit{sko}(\delta)$. □

Lemma 3.6.7 *If Ξ is a Hintikka set (Def. 3.6.6), then*

1. *it is satisfied by some tableau interpretation in $\mathit{TabInterp}(\Sigma^*)$;*
2. *every tableau interpretation satisfying the atomic tableau formulae in Ξ satisfies Ξ.*

Proof: The second part of the lemma is easy to prove by induction on the structure of tableau formulae in Ξ.

A canonical tableau interpretation $\langle\langle D, \mathcal{I}\rangle, I\rangle$ satisfying the atomic formulae in Ξ can be defined as follows, which—using the second part of the lemma—proves the first part of the lemma:

- $\langle D, \mathcal{I}\rangle$ is an Herbrand structure, i.e., $D = \mathit{Term}^0_{\mathrm{PL1}}(\Sigma^*)$ and $\mathcal{I}(t) = t$ for all terms $t \in \mathit{Term}^0_{\mathrm{PL1}}(\Sigma^*)$;
- for all atoms $p(t_1, \ldots, t_{\alpha(p)})$ over Σ^*, put $p^{\mathcal{I}}(t_1, \ldots, t_{\alpha(p)}) = \mathit{true}$ if and only if $\mathsf{T}{:}p(t_1, \ldots, t_{\alpha(p)}) \in \Xi$;
- $I(*) = \langle D, \mathcal{I}\rangle$.

The interpretation function $\mathcal{I}$ is well-defined because of Condition 1 in the definition of Hintikka sets (Def. 3.6.6). □

Lemma 3.6.8 *The tableau calculus $\mathcal{C}_{\mathrm{PL1}}$ for PL1 has the Strong Soundness Property 1 from Definition 3.5.8 (appropriateness of the set of tableau interpretations).*

Proof: Let $\langle D, \mathcal{I}\rangle$ be a first-order Herbrand structure satisfying a set $\mathfrak{F}$ of formulae. Because the Skolem symbols in $F^{sko}(\Sigma)$ do not occur in $\mathfrak{F}$, it suffices to choose their interpretation such that the resulting structure $\langle D, \mathcal{I}^*\rangle$ satisfies Condition 2 in the definition of canonical tableau interpretations, leaving the interpretation of the symbols in Σ unchanged. A canonical tableau interpretation satisfying the initial tableau for $\mathfrak{F}$ can then be constructed combining $\langle D, \mathcal{I}^*\rangle$ with the label interpretation I defined by $I(*) = \langle D, \mathcal{I}\rangle$.

The rank $rk(f)$ of the symbols $f \in F^{sko}(\Sigma)$ is defined as follows:

- $rk(f) = 0$ if no $\delta \in \mathit{TabForm}(\Sigma^*)$ exists such that $sko(\delta) = f(t_1, \dots, t_k)$ for any $t_1, \dots, t_k \in \mathit{Term}^0_{\mathrm{PL1}}(\Sigma^*)$;
- $rk(f) = 1$ if $sko(\delta) = f(t_1, \dots, t_k)$ for some $\delta \in \mathit{TabForm}(\Sigma)$ and some terms $t_1, \dots, t_k \in \mathit{Term}^0_{\mathrm{PL1}}(\Sigma)$;
- $rk(f) = 1 + \max\{rk(f') \mid f' \in F\}$ where F is the set of all $f' \in F^{sko}(\Sigma)$ occurring in any $\delta \in \mathit{TabForm}(\Sigma^*)$ such that $sko(\delta) = f(t_1, \dots, t_k)$ for any terms $t_1, \dots, t_k \in \mathit{Term}^0_{\mathrm{PL1}}(\Sigma^*)$.

The function rk is well defined because of Condition 2 in Definition 3.6.2.

We inductively define a sequence $(\langle D, \mathcal{I}^n\rangle)_{n\geq 0}$ of first-order structures that all have the domain D, where $\langle D, \mathcal{I}^n\rangle$ is a structure over the signature Σ^n that is the restriction of Σ^* to function symbols of rank not greater than n; the interpretation $\mathcal{I}^{n+1}$ coincides with $\mathcal{I}^n$ on all symbols in $\Sigma^n \cup \Sigma$.

The initial interpretation $\mathcal{I}^0$ is defined by $f^{\mathcal{I}^0} = f^{\mathcal{I}}$ for all $f \in F(\Sigma)$, and for all $f \in F^{sko}(\Sigma)$ of rank 0 the value of $f^{\mathcal{I}^0}$ is chosen arbitrarily.

The symbols $f \in F^{sko}(\Sigma)$ of rank $r \leq n$ have already been interpreted by $\mathcal{I}^n$. Consider $f \in F^{sko}(\Sigma)$ of rank $n+1$; we define $f^{\mathcal{I}^{n+1}}(b_1, \dots, b_k)$ for $b_1, \dots, b_k \in D$ by: If there are terms $t_1, \dots, t_k \in \mathit{Term}^0(\Sigma^*)$ such that $t_i^{\mathcal{I}^n} = b_i$ $(1 \leq i \leq k)$ and there is a formula $\delta(x) \in \mathit{TabForm}(\Sigma^*)$ with $f(t_1, \dots, t_k) = sko(\delta(x))$ and $val_{\mathcal{I}^n}(\delta) = \mathit{true}$, choose an $e \in D$ with $val_{\mathcal{I}^n,\{x\mapsto e\}}(\delta_1(x)) = \mathit{true}$, and set $f^{\mathcal{I}^{n+1}}(b_1, \dots, b_k) = e$ (since f is of rank $n+1$, the symbols in δ are from the signature Σ^n). Otherwise, if no such terms $t_1, \dots, t_k$ and formula δ exist, choose $f^{\mathcal{I}^{n+1}}(b_1, \dots, b_k)$ to be an arbitrary element in D.

If other terms $t'_1, \dots, t'_k$ and another δ-formula δ' exist satisfying the above conditions, then $val_{\mathcal{I}^n}(\delta(x)) = val_{\mathcal{I}^n}(\delta'(x))$ and $val_{\mathcal{I}^n,\{x\mapsto e\}}(\delta_1(x)) = val_{\mathcal{I}^n,\{x\mapsto e\}}(\delta'_1(x))$, because $\mathcal{I}^n(t_i) = \mathcal{I}^n(t'_i) = b_i$ $(1 \leq i \leq k)$ and the formulae δ and δ' are identical up to renaming of bound variables and replacement of terms t_i by t'_i (Condition 3 in Def. 3.6.2).

We can think of the sequence $(\langle D, \mathcal{I}^n \rangle)_{n \geq 0}$ as an approximation to the first-order interpretation $\langle D, \mathcal{I}^* \rangle$ over Σ^*. The interpretation $\mathcal{I}^*$ coincides with $\mathcal{I}^n, \mathcal{I}^{n+1}, \ldots$ on the symbols in Σ^n. It satisfies Condition 2 in Definition 3.6.2 by construction. □

Lemma 3.6.9 *The tableau calculus $\mathcal{C}_{\mathrm{PL1}}$ for PL1 has Strong Soundness Property 2 from Definition 3.5.8 (soundness of expansion).*

Proof: As the calculus is ideal, we can use Lemma 3.5.9. Let $\Pi \subset \mathit{TabForm}_{\mathrm{PL1}}(\Sigma^*)$ be a minimal premiss of a conclusion C; and assume that Π is satisfied by a tableau interpretation $\langle \mathbf{m}, I \rangle = \langle \langle D, \mathcal{I} \rangle, I \rangle \in \mathit{TabInterp}_{\mathrm{PL1}}(\Sigma^*)$. We show that $\langle \mathbf{m}, I \rangle$ satisfies one of the extensions in C by cases according to the form of Π:

$\Pi = \{\alpha\}$: In that case, $C = \{\{\alpha_1, \alpha_2\}\}$, and since $\langle \mathbf{m}, I \rangle$ satisfies α we have, by the property of α-formulae (Def. 2.3.2), that $\langle \mathbf{m}, I \rangle$ satisfies α_1 and α_2.

$\Pi = \{\beta\}$: In that case, $C = \{\{\beta_1\}, \{\beta_2\}\}$, and since $\langle \mathbf{m}, I \rangle$ satisfies β we have, by the property of β-formulae (Def. 2.3.2), that $\langle \mathbf{m}, I \rangle$ satisfies β_i for some $i \in \{1, 2\}$.

$\Pi = \{\gamma\}$: In that case, $C = \{\{\gamma_1(t)\}\}$ for some $t \in \mathit{Term}^0_{\mathrm{PL1}}(\Sigma^*)$, and since $\langle \mathbf{m}, I \rangle$ satisfies $\gamma(x)$ we have, by the property of γ-formulae, that $\langle \mathbf{m}, \mathcal{I} \rangle$ satisfies $\gamma_1(t)$.

$\Pi = \{\delta\}$: In that case, $C = \{\{\delta_1(t)\}\}$ where $t = \mathit{sko}(\delta)$, and since $\langle \mathbf{m}, \mathcal{I} \rangle$ is canonical and satisfies δ, it satisfies $\delta_1(t)$.

$\Pi = \{\mathsf{T}{:}\sigma{:}F, \mathsf{F}{:}\sigma{:}F\}$: A premiss Π of this form is not satisfied by any tableau interpretation, which contradicts the assumption that Π is satisfied by $\langle \mathbf{m}, \mathcal{I} \rangle$; thus, this case cannot occur. □

Lemma 3.6.10 *The tableau calculus $\mathcal{C}_{\mathrm{PL1}}$ for PL1 has the Strong Completeness Property 1 from Definition 3.5.10 (appropriateness of the set of tableau interpretations).*

Proof: If $\langle \mathbf{m}^*, \mathcal{I} \rangle \in \mathit{TabForm}(\Sigma^*)$ satisfies an initial tableau for $\mathfrak{F}$, then any restriction of $\mathbf{m}^*$ to Σ satisfies $\mathfrak{F}$, because the symbols in $F^{sko}(\Sigma)$, which are the only additional symbols, do not occur in $\mathfrak{F}$. □

Lemma 3.6.11 *The tableau calculus $\mathcal{C}_{\mathrm{PL1}}$ for PL1 has Completeness Property 2 (satisfiability of fully expanded branches) from Definition 3.5.6; and it is strongly semantically analytic (Def. 3.5.16).*

Proof: Let B be a fully expanded branch that is not closed; and let $\Phi \subset \mathit{TabForm}(\Sigma^*)$ be a set of atomic tableau formulae such that, for *no* ϕ in Φ, both ϕ and $\overline{\phi}$ are in $\mathit{Form}(B) \cup \Phi$.

The set $\mathit{Form}(B) \cup \Phi$ is a Hintikka set and Lemma 3.6.7 implies that there is a tableau interpretation $\langle \mathbf{m}, I \rangle$ satisfying $\mathit{Form}(B) \cup \Phi$; thus, the calculus $\mathcal{C}_{\mathrm{PL1}}$ has Completeness Property 2 from Definition 3.5.6.

Since there is only a single world w^0 in $\mathbf{m}$ and, by definition $I(*) = w^0$, the second condition in Definition 3.5.16 is met as well; and the calculus is indeed strongly semantically analytic. $\square$

Theorem 3.6.12 *The tableau calculus $\mathcal{C}_{\mathrm{PL1}}$ for PL1 is sound and complete.*

Proof: According to Theorems 3.5.4 and 3.5.7, it is sufficient to prove that $\mathcal{C}_{\mathrm{PL1}}$ has the two soundness properties from Definition 3.5.3 and the two completeness properties from Definition 3.5.6. That, however, follows immediately from Lemmata 3.6.8, 3.6.9, 3.6.10. $\square$

3.7 Ideal Tableau Calculi for Modal Logics

3.7.1 Overview

The first non-structural tableau calculi for modal logics, which use labels for encoding the reachability relation between possible worlds (instead of implicitly encoding them in the structure of tableaux), were described in (Fitting, 1983). Following Fitting's work, non-structural tableau calculi for many modal logics have been defined, see (Goré, 1998) for an overview. However, all these calculi are non-monotonic because they use tableau rules that when applied to a π-formula (a formula asserting *existence* of a reachable world in which a formula is true) they skolemise the π-formula by introducing a label that has to be *new* to the branch or tableau. In addition, many of these calculi are not ideal, as *all* branches that pass through a particular premiss are extended when that premiss is used for expansion.

In Section 3.7.2, we present a tableau calculus that is ideal because each formula F is assigned its own unique label, which is a gödelisation of F itself. This unique label is used for skolemising the π-formula $\mathsf{T}{:}\sigma{:}\Diamond F$. This calculus is a ground version (i.e., a version without free variables) of the calculus described in (Beckert & Goré, 1997). It exemplifies the claim made in Section 3.3.7, that it often only requires minor modifications to turn a "slightly" non-ideal calculus into an ideal one.

Because a world in a Kripke model may have no successor, the conclusion of a ν-formula (asserting that a certain formula is true in all reachable worlds) must only contain labels representing worlds whose existence is known. This knowledge is deduced from labels of other formulae, which leads to non-continuity of the tableau rule for all premisses containing a ν-formula. We define a variant of the calculus whose rule is continuous for ν-formulae in Section 3.7.4. It uses *conditional* labels, where a conditional labels does not imply the existence of the world it represents. An additional benefit of using conditional labels is that they simplify the definition of a free variable version of the calculus. Checking the existence of worlds is made part of closing a tableau branch.

3.7.2 Labels for Modal Logic Calculi

In this section, we introduce labels consisting of natural numbers, which are frequently used in labelled tableau calculi. Labels consisting of other constituents than natural numbers can be defined in the same way.

Definition 3.7.1 Let $\mathbb{N}$ be the set of natural numbers. The set $Lab(\mathbb{N})$ of (non-conditional) *labels consisting of natural numbers* is defined by:

- The word 1 is an element of $Lab(\mathbb{N})$.
- If $\sigma \in Lab(\mathbb{N})$, then the word $\sigma.n$ is an element of $Lab(\mathbb{N})$ for all $n \in \mathbb{N}$.

The set $CondLab(\mathbb{N})$ of *conditional labels consisting of natural numbers* is defined by:

- The word 1 is an element of $CondLab(\mathbb{N})$;
- If $\sigma \in Lab(\mathbb{N})$, then the words $\sigma.n$ and $\sigma.(n)$ are elements of $Lab(\mathbb{N})$ for all $n \in \mathbb{N}$.

The *initial label* of $Lab(\mathbb{N})$ and $CondLab(\mathbb{N})$ is 1.

The *length* of a label σ is the number of dots it contains plus one, and is denoted by $|\sigma|$.

The components of a label σ are called *positions* in σ. A position is *conditional* if it is of the form (n), and a label is conditional if it contains a conditional position.

The equivalence class of all labels in $CondLab(\mathbb{N})$ that are identical to a label σ up to parentheses indicating conditional positions is denoted by $[\sigma]$.

The set of all non-empty *initial prefixes* of a label σ, excluding σ itself, is denoted by $ipr(\sigma)$. $\square$

Note that $Lab(\mathbb{N}) \subset CondLab(\mathbb{N})$; and that (1) is not an element of $CondLab(\mathbb{N})$, because 1 represents the initial world in models, which always exists.

We often do not differentiate between the labels $\sigma.n$ and $\sigma.(n)$, and we use $\sigma.[n]$ to denote that the label may be of either form.

Definition 3.7.2 A set $\Gamma \subset Lab(\mathbb{N})$ of labels is *strongly generated* if:

1. the initial label 1 is an element of Γ; and
2. $\sigma \in \Gamma$ implies $\tau \in \Gamma$ for all $\tau \in ipr(\sigma)$. $\square$

Logic	$\sigma \rhd \tau$ iff	Logic	$\sigma \rhd \tau$ iff
K	$\tau = \sigma.[n]$	KT	$\tau = \sigma.[n]$ or $\tau = \sigma$
KB	$\tau = \sigma.[n]$ or $\sigma = \tau.[m]$	K4	$\tau = \sigma.\theta$
K5	$\tau = \sigma.[n]$, or $\lvert\sigma\rvert \geq 2$, $\lvert\tau\rvert \geq 2$	K45	$\tau = \sigma.\theta$, or $\lvert\sigma\rvert \geq 2$, $\lvert\tau\rvert \geq 2$
KD	K-condition, or σ is a K-deadend and $\sigma = \tau$	KDB	KB-condition, or $\lvert\Gamma\rvert = 1$ and $\sigma = \tau = 1$
KD4	K4-condition, or σ is a K-deadend and $\sigma = \tau$	KD5	K5-condition, or $\lvert\Gamma\rvert = 1$ and $\sigma = \tau = 1$
KD45	K45-condition, or $\lvert\Gamma\rvert = 1, \sigma = \tau = 1$	KB4	$\lvert\Gamma\rvert \geq 2$
B	$\tau = \sigma$, or $\tau = \sigma.[n]$, or $\sigma = \tau.[m]$	S4	$\tau = \sigma.\theta$ or $\tau = \sigma$
S5	for all σ, τ		

Table 3.3: The accessibility relation on labels for the basic modal logics.

The labels in $Lab(\mathbb{N})$ capture a basic reachability relation between the worlds they name, where the world named by $\sigma.[n]$ is reachable from the world named by σ. A set of strongly generated labels can be viewed as a tree with root 1, where $\sigma.[n]$ is an immediate child of σ.

Definition 3.7.3 Given a modal logic $\mathbf{L}$ and a set $\Gamma \subset Lab(\mathbb{N})$ of strongly generated labels, a label $\tau \in \Gamma$ is $\mathbf{L}$-*accessible* from a label $\sigma \in \Gamma$, written as $\sigma \rhd \tau$, if the conditions set out in Table 3.3 are satisfied.

A label $\sigma \in \Gamma$ is an $\mathbf{L}$-*deadend* if no $\tau \in \Gamma$ is $\mathbf{L}$-accessible from σ. □

The following lemma shows that the $\mathbf{L}$-accessibility relation $\rhd$ on labels captures the reachability relation R of $\mathbf{L}$-frames exactly (see (Goré, 1998) for a proof). In particular, $\rhd$ has the properties like reflexivity, transitivity, etc. that are appropriate for the axioms of $\mathbf{L}$ (see Table 2.1).

Lemma 3.7.4 *Let* $\mathbf{L}$ *one of the basic modal logics. If* $\Gamma \subset Lab(\mathbb{N})$ *is a strongly generated set of labels, then* $\langle \Gamma, \rhd \rangle$ *is an* $\mathbf{L}$*-frame, where* $\rhd$ *is the* $\mathbf{L}$*-accessibility relation.*

3.7.3 Syntax and Semantics of Calculi for Modal Logics

The ideal calculi $\mathcal{C}_{\mathbf{L}}$ for the basic modal logics $\mathbf{L}$ presented in this section are based on the labelled tableau calculi described in (Fitting, 1983). The main difference is that, to ensure monotonicity of the calculi, we use an expansion rule schema for π-formulae that does not introduce a *new* labels but—similar to the schema for δ-formulae in Section 3.6—uses a symbol that is uniquely assigned to the formula to which the rule is applied.

α	$\alpha_1,$	α_2
$\mathsf{T}{:}\sigma{:}(F \wedge G)$	$\mathsf{T}{:}\sigma{:}F,$	$\mathsf{T}{:}\sigma{:}G$
$\mathsf{F}{:}\sigma{:}(F \vee G)$	$\mathsf{F}{:}\sigma{:}F,$	$\mathsf{F}{:}\sigma{:}G$
$\mathsf{T}{:}\sigma{:}\neg F$	$\mathsf{F}{:}\sigma{:}F,$	$\mathsf{F}{:}\sigma{:}F$
$\mathsf{F}{:}\sigma{:}\neg F$	$\mathsf{T}{:}\sigma{:}F,$	$\mathsf{T}{:}\sigma{:}F$

β	$\beta_1,$	β_2
$\mathsf{T}{:}\sigma{:}(F \vee G)$	$\mathsf{T}{:}\sigma{:}F,$	$\mathsf{T}{:}\sigma{:}G$
$\mathsf{F}{:}\sigma{:}(F \wedge G)$	$\mathsf{F}{:}\sigma{:}F,$	$\mathsf{F}{:}\sigma{:}G$

$\mathsf{S}{:}\sigma{:}\nu$	$\mathsf{S}{:}\sigma.n{:}\nu_K$	$\mathsf{S}{:}\sigma.n{:}\nu_4$	$\mathsf{S}{:}\sigma{:}\nu_T$
$\mathsf{T}{:}\sigma{:}\Box F$	$\mathsf{T}{:}\sigma.n{:}F$	$\mathsf{T}{:}\sigma.n{:}\Box F$	$\mathsf{T}{:}\sigma{:}F$
$\mathsf{F}{:}\sigma{:}\Diamond F$	$\mathsf{F}{:}\sigma.n{:}F$	$\mathsf{F}{:}\sigma.n{:}\Diamond F$	$\mathsf{F}{:}\sigma{:}F$

$\mathsf{S}{:}\tau.n{:}\nu$	$\mathsf{S}{:}\tau{:}\nu_{4^r}$	$\mathsf{S}{:}\tau{:}\nu_B$	$\mathsf{S}{:}\tau{:}\nu_5$
$\mathsf{T}{:}\tau.n{:}\Box F$	$\mathsf{T}{:}\tau{:}\Box F$	$\mathsf{T}{:}\tau{:}F$	$\mathsf{T}{:}\tau{:}\Box\Box F$
$\mathsf{F}{:}\tau.n{:}\Diamond F$	$\mathsf{F}{:}\tau{:}\Diamond F$	$\mathsf{F}{:}\tau{:}F$	$\mathsf{F}{:}\tau{:}\Diamond\Diamond F$

$\mathsf{S}{:}\sigma{:}\pi$	$\mathsf{S}{:}\sigma.n{:}\pi_1$
$\mathsf{T}{:}\sigma{:}\Diamond F$	$\mathsf{T}{:}\sigma.n{:}F$
$\mathsf{F}{:}\sigma{:}\Box F$	$\mathsf{F}{:}\sigma.n{:}F$

Table 3.4: The four formula types of modal logics.

Extended Signatures No extension of the signatures is necessary, thus $\Sigma = \Sigma^*$ for all modal logics and all signatures $\Sigma \in \mathit{Sig}_{\mathrm{mod}}$.

Labels The set $\mathit{Lab}(\Sigma)$ of labels is, for all signatures Σ, the set $\mathit{Lab}(\mathbb{N})$ of (non-conditional) labels consisting of natural numbers (Def. 3.7.1).

Expansion Rule There are four types of complex (non-atomic) tableau formulae: α-formulae (conjunctive) and β-formulae (disjunctive) as in calculi for classical logic, ν-formulae (which express truth of a formula in *all* reachable worlds), and π-formulae (which express truth of a formula in *some* reachable world); see Table 3.4.

Notation 3.7.5 The letters ν and π are used to denote formulae of (and only of) the appropriate type. □

The differences in the expansion rules for different modal logics are mainly in the rule schema for ν-formulae, i.e., in the conclusions of a premiss containing a ν-formula. In Table 3.5, the expansion rule is given schematically for the various formula types. Table 3.6 summarises which formulae are part of the conclusion of a premiss $\{\{\mathsf{S}{:}\sigma{:}\nu\}\}$ consisting of a ν-formula; in that table, 4^d indicates the inclusion of ν_4 in case $|\sigma| \geq 2$.

We give the formal definition of the expansion rule for the logic K; the formal definitions of the expansion rules for the other modal logics can easily be extracted from their schematical description.

$$\begin{array}{c} \alpha \\ \hline \alpha_1 \\ \alpha_2 \end{array} \qquad \begin{array}{c} \beta \\ \hline \beta_1 \mid \beta_2 \end{array} \qquad \begin{array}{c} \mathsf{S}{:}\sigma{:}\pi \\ \hline \mathsf{S}{:}\sigma.\lceil\pi\rceil{:}\pi_1 \end{array} \qquad \begin{array}{c} \mathsf{T}{:}\sigma{:}F \\ \mathsf{F}{:}\sigma{:}F \\ \hline \bot \end{array}$$

where $n = \lceil F \rceil$ if $\pi = \mathsf{T}{:}\sigma{:}\Diamond F$, and $n = \lceil \neg F \rceil$ if $\pi = \mathsf{F}{:}\sigma{:}\Box F$

$$\begin{array}{c} \mathsf{S}{:}\sigma{:}\nu \\ \hline \mathsf{S}{:}\sigma.n{:}\nu_K \end{array} \qquad \begin{array}{c} \mathsf{S}{:}\sigma{:}\nu \\ \hline \mathsf{S}{:}\sigma.n{:}\nu_4 \end{array} \qquad \begin{array}{c} \mathsf{S}{:}\sigma{:}\nu \\ \hline \mathsf{S}{:}\sigma{:}\nu_T \end{array}$$

where $\sigma.n$ occurs on the branch; for all logics

where $\sigma.n$ occurs on the branch; for K4, KD4, S4, S5, and, if $|\sigma| \geq 2$, for K5, KD5

for T, B, S4, S5.

$$\begin{array}{c} \mathsf{S}{:}\tau.n{:}\nu \\ \hline \mathsf{S}{:}\tau{:}\nu_{4^r} \end{array} \qquad \begin{array}{c} \mathsf{S}{:}\tau.n{:}\nu \\ \hline \mathsf{S}{:}\tau{:}\nu_B \end{array} \qquad \begin{array}{c} \mathsf{S}{:}\tau.n{:}\nu \\ \hline \mathsf{S}{:}\tau{:}\nu_5 \end{array}$$

for K5, KD5, K45 KD45, KB4, S5.

for KB, KDB, KB4, B.

if $\tau = 1$ for K5, KD5.

Table 3.5: Rule schemata for modal logics.

Logic	$\nu_{\mathbf{L}}$ for $\mathbf{L} =$	Logic	$\nu_{\mathbf{L}}$ for $\mathbf{L} =$
K, D	K	K45, KD45	K, 4, 4^r
T	K, T	KB4	K, B, 4, 4^r
KB, KDB	K, B	B	K, T, B
K4, KD4	K, 4	S4	K, T, 4
K5, KD5	K, 4^d, 4^r, 5	S5	K, T, 4, 4^r

Table 3.6: Elements $\nu_{\mathbf{L}}$ of the conclusion of a ν-formula.

Definition 3.7.6 For all premisses $\Pi \subset \mathit{TabForm}_{\mathrm{mod}}(\Sigma)$, the set $\mathcal{E}_{\mathrm{K}}(\Sigma)(\Pi)$ consists of the the following conclusions (where $\lceil\cdot\rceil$ is any bijection from the set $\mathit{Form}_{\mathrm{mod}}(\Sigma)$ of formulae to the set of natural numbers):

– $\{\{\alpha_1, \alpha_2\}\}$ for all $\alpha \in \Pi$,
– $\{\{\beta_1\}, \{\beta_2\}\}$ for all $\beta \in \Pi$,
– $\{\{\mathsf{T}{:}\sigma.n{:}F\}\}$ for all $\mathsf{T}{:}\sigma{:}\Box F \in \Pi$ and all labels of the form $\sigma.n$ occurring in Π,
– $\{\{\mathsf{F}{:}\sigma.n{:}F\}\}$ for all $\mathsf{F}{:}\sigma{:}\Diamond F \in \Pi$ and all labels of the form $\sigma.n$ occurring in Π,
– $\{\{\mathsf{F}{:}\sigma.n{:}F\}\}$ for all $\mathsf{F}{:}\sigma{:}\Box F \in \Pi$ where $n = \lceil\neg F\rceil$,
– $\{\{\mathsf{T}{:}\sigma.n{:}F\}\}$ for all $\mathsf{T}{:}\sigma{:}\Diamond F \in \Pi$ where $n = \lceil F\rceil$,
– $\{\{\bot\}\}$ if $\mathsf{T}{:}\sigma{:}F, \mathsf{F}{:}\sigma{:}F \in \Pi$ for any $F \in \mathit{Form}_{\mathrm{mod}}(\Sigma)$.

$\Box$

Semantics To define the semantics of tableaux for a modal logic $\mathbf{L}$, we use the set $\mathit{TabInterp}_{\mathbf{L}}(\Sigma^*)$ consisting of tableau interpretation that are (a) $\mathbf{L}$-interpretations and that are (b) canonical, i.e., interpret labels generated by rule applications to π-formulae in the right way.

Definition 3.7.7 Let $\mathbf{L}$ be one of the basic modal logics; let $\Sigma \in \mathit{Sig}$ be a signature; and let Lab be the set $\mathit{CondLab}(\mathbb{N})$ of (conditional and non-conditional) labels consisting of natural numbers (Def. 3.7.1).

A tableau interpretation $\langle \mathbf{m}, I\rangle$ is an $\mathbf{L}$-*interpretation* if $\mathbf{m} = \langle W, R, V\rangle$ is a Kripke $\mathbf{L}$-model and the label interpretation function I has the following properties:

1. $I(1) = w^0$ is the initial world of $\mathbf{m}$;
2. $I(\sigma.(n)) = I(\sigma.n)$ for all $\sigma.n$ and $\sigma.(n)$ in Lab;
3. for all $\sigma \in \mathit{Lab}$, if $I(\tau)$ is undefined for some $\tau \in \mathit{ipr}(\sigma)$, then $I(\sigma)$ is undefined;
4. for all $\sigma, \tau \in \mathit{Lab}$, if (a) $\sigma \rhd \tau$ and (b) $I(\sigma)$ and $I(\tau)$ are defined, then $I(\sigma)\ R\ I(\tau)$.

An $\mathbf{L}$-interpretation is *canonical* if, moreover:

5. For all labels $\sigma = \tau.n \in \mathit{Lab}$:

   if $I(\tau)$ is defined and $I(\tau) \models \Diamond F$, then $I(\sigma)$ is defined and $I(\sigma) \models F$,

   where F is the formula for which $n = \lceil F\rceil$ ($\lceil\cdot\rceil$ is the bijection from the set of formulae to the set of natural numbers used by the expansion rule). $\Box$

Because we deal only with strongly generated sets of labels with root 1, the twin requirements that every $\mathbf{L}$-interpretation $\langle \mathbf{m}, \mathcal{I} \rangle$ define the label 1, and Condition 3 in the above definition force the interpretation function I to "define" as many labels in $ipr(\sigma)$ as is possible. However, for a conditional label of the form $\tau.(n)$, where n is parenthesised, it is perfectly acceptable that $I(\tau.(n))$ is undefined even if $I(\tau)$ is defined.

Example 3.7.8 The interpretation function I must be defined for the labels 1, 1.1, and $1.1.1$. It need not be defined for $1.(1)$; but if it is, then $I(1.(1).1)$ must be defined as well. □

Using the set $\mathit{TabInterp}_{\mathbf{L}}(\Sigma^*)$ of canonical $\mathbf{L}$-interpretations for defining the semantics of tableaux, the calculus $\mathcal{C}_{\mathbf{L}}$ has the soundness and completeness properties from Definitions 3.5.3 and 3.5.6 (where $\mathbf{L}$ is any of the basic modal logics). If a tableau is satisfied by a canonical tableau interpretation, then all its successor tableaux are satisfied by the same interpretation; and every fully expanded tableau branch that is not closed is satisfied by a canonical $\mathbf{L}$-interpretation.

Again, the proof that fully expanded branches are satisfiable is based on an appropriate version of Hintikka's Lemma:

Definition 3.7.9 Let $\mathbf{L}$ be a basic modal logic; and let $\Sigma \in \mathit{Sig}_{\mathrm{mod}}$ be a signature. A set $\Xi \subset \mathit{TabForm}_{\mathrm{mod}}(\Sigma^*)$ of tableau formulae not containing conditional labels is a *modal* $\mathbf{L}$*-Hintikka* set if it satisfies the following conditions:

1. There are no complementary atomic formulae $\mathsf{T}{:}\sigma{:}p$ and $\mathsf{F}{:}\sigma{:}p$ in Ξ.
2. If $\alpha \in \Xi$, then α_1 and α_2 are in Ξ.
3. If $\beta \in \Xi$, then β_1 or β_2 is in Ξ.
4. If $\mathsf{S}{:}\sigma{:}\nu \in \Xi$, then $\mathsf{S}{:}\tau{:}\nu_{\mathrm{K}} \in \Xi$ for *all* $\tau \in \mathit{Lab}$ such that $\sigma \rhd \tau$.
5. If $\mathsf{S}{:}\sigma{:}\pi \in \Xi$, then $\mathsf{S}{:}\tau{:}\pi_1 \in \Xi$ for *some* $\tau \in \mathit{Lab}$ such that $\sigma \rhd \tau$. □

Lemma 3.7.10 *Let* $\mathbf{L}$ *be a basic modal logic; and let* $\Sigma \in \mathit{Sig}_{\mathrm{mod}}$ *be a signature. If* Ξ *is a modal* $\mathbf{L}$*-Hintikka set (Def. 3.7.9), then*

1. *it is satisfied by some tableau interpretation in* $\mathit{TabInterp}_{\mathbf{L}}(\Sigma^*)$*, i.e., a canonical* $\mathbf{L}$*-interpretation;*
2. *every tableau interpretation in* $\mathit{TabInterp}_{\mathbf{L}}(\Sigma^*)$ *satisfying the atomic tableau formulae in* Ξ *satisfies* Ξ*.*

Proof: The second part of the lemma is easy to prove by induction on the structure of tableau formulae in Ξ.

Because of Condition 1 in the definition of modal Hintikka sets (Def. 3.7.9), a canonical tableau interpretation $\langle\langle W, R, V\rangle, I\rangle$ satisfying the atomic formulae in Ξ can be defined by:

- Let W be the set Lab of all labels occurring in Ξ;
- let $I(\sigma) = \sigma$ if $\sigma \in W$, and let $I(\sigma)$ be undefined otherwise;
- for all $\sigma, \tau \in W$, let $\sigma \, R \, \tau$ iff $\sigma \rhd \tau$;
- let $V(p) = \{\sigma \mid \mathsf{T}{:}\sigma{:}p \in \Xi\}$. $\Box$

Lemma 3.7.11 *For all basic modal logics* $\mathbf{L}$*, the tableau calculus* $\mathcal{C}_{\mathbf{L}}$ *has Strong Soundness Property 1 from Definition 3.5.8 (appropriateness of the set of tableau interpretations).*

Proof: Let $\mathbf{m} = \langle W, R, V\rangle$ be an $\mathbf{L}$-model satisfying a set of formulae $\mathfrak{F}$. We know that $w^0 \models F$ for all $F \in \mathfrak{F}$, where w^0 is the initial world in W.

Now, for $n \in \mathbb{N}$, let F_n be the formula for which $n = \lceil F \rceil$ (where $\lceil \cdot \rceil$ is the bijection from the set of formulae to the set of natural numbers used by the expansion rule) and create I as follows: Let $I(1) = w^0$, and for every label of the form $\tau.n$:

- if there is a world $w \in W$ that is reachable from $I(\tau)$ such that $w \models F_n$, then put $I(\tau.n) = I(\tau.(n)) = w$;
- else, if there is no such world w, but there is a world w' that is reachable from $I(\tau)$, then put $I(\tau.n) = I(\tau.(n)) = w'$;
- else, if there is no world reachable from $\mathcal{I}(\tau)$, let $I(\tau.n)$ and $I(\tau.(n)$ be undefined.

The $\mathbf{L}$-interpretation $\langle \mathbf{m}, I\rangle$ is canonical by way of its definition, and in addition satisfies the tableau formulae on initial tableaux for $\mathfrak{F}$, because $I(1) = w^0 \models F$ for all $F \in \mathfrak{F}$. $\Box$

Lemma 3.7.12 *For all basic modal logics* $\mathbf{L}$*, the tableau calculus* $\mathcal{C}_{\mathbf{L}}$ *has Strong Soundness Property 2 from Definition 3.5.8 (soundness of expansion).*

Proof: As the calculus is ideal, we can use Lemma 3.5.9. Let $\Pi \subset TabForm_{\mathrm{mod}}(\Sigma^*)$ be a minimal premiss of a conclusion C; and assume that Π is satisfied by a canonical $\mathbf{L}$-interpretation $\langle \mathbf{m}, I\rangle \in TabInterp_{\mathbf{L}}(\Sigma^*)$. It can easily be checked by cases according to the form of Π that $\langle \mathbf{m}, I\rangle$ satisfies one of the extensions in C. $\Box$

Lemma 3.7.13 *For all basic modal logics* $\mathbf{L}$*, the tableau calculus* $\mathcal{C}_{\mathbf{L}}$ *has Strong Completeness Property 1 from Definition 3.5.10 (appropriateness of the set of tableau interpretations).*

Proof: The calculi have this property trivially, because the signatures have not been extended, i.e., $\Sigma^* = \Sigma$, and therefore every $\mathbf{L}$-model in $\mathcal{M}(\Sigma^*)$ is a restriction of itself to Σ. □

Lemma 3.7.14 *For all basic modal logics* $\mathbf{L}$*, the tableau calculus* $\mathcal{C}_{\mathbf{L}}$ *has Completeness Property 2 (satisfiability of fully expanded branches) from Definition 3.5.6; and it is strongly semantically analytic.*

Proof: Let B be a fully expanded branch that is not closed; and let $\Phi \subset \mathit{TabForm}(\Sigma^*)$ be a set of atomic tableau formulae such that, for *no* ϕ in Φ, both ϕ and $\overline{\phi}$ are in $\mathit{Form}(B) \cup \Phi$.

The set $\mathit{Form}(B) \cup \Phi$ is an $\mathbf{L}$-Hintikka set and Lemma 3.7.10 implies that there is a tableau interpretation $\langle \mathbf{m}, I \rangle$ satisfying $\mathit{Form}(B) \cup \Phi$; thus, the calculus $\mathcal{C}_{\mathrm{PL1}}$ has Completeness Property 2 from Definition 3.5.6.

By construction of that tableau interpretation (see the proof of Lemma 3.7.10), $I(\sigma) = \sigma$ for all $\sigma \in W$ and, thus, the second condition in Definition 3.5.16 is met as well; and the calculus is indeed strongly semantically analytic. □

Theorem 3.7.15 *For all basic modal logics* $\mathbf{L}$*, the tableau calculus* $\mathcal{C}_{\mathbf{L}}$ *for* $\mathbf{L}$ *is sound and complete.*

Proof: According to Theorems 3.5.4 and 3.5.7, it is sufficient to prove that $\mathcal{C}_{\mathbf{L}}$ has the two soundness properties from Definition 3.5.3 and the two completeness properties from Definition 3.5.6. That, however, follows immediately from Lemmata 3.7.11, 3.7.12, 3.7.13, and 3.7.14, oberving that a calculus that a semantically analytic calculus trivially has Completeness Property 2 from Definition 3.5.6. □

Example 3.7.16 We prove that

$$G = \Box(\neg p \lor q) \land \Box p \land (\Diamond \neg q \lor \Diamond \neg p)$$

is unsatisfiable in the modal logic K (and, thus, that its negation is a K-tautology). A (fully expanded) closed tableau, that is part of a tableau proof for (the K-unsatisfiability) of G is shown in Figure 3.1. The nodes of the tableau are numbered; a pair $[i; j]$ is attached to the i-th node, the number j denotes that node i has been created by applying the expansion rule to a premiss containing the formula in node j.

Note, that after formula 5 has been added to the tableau, the only possible conclusion (that is not already on the tableau) is the one consisting of formulae 6 and 7, which

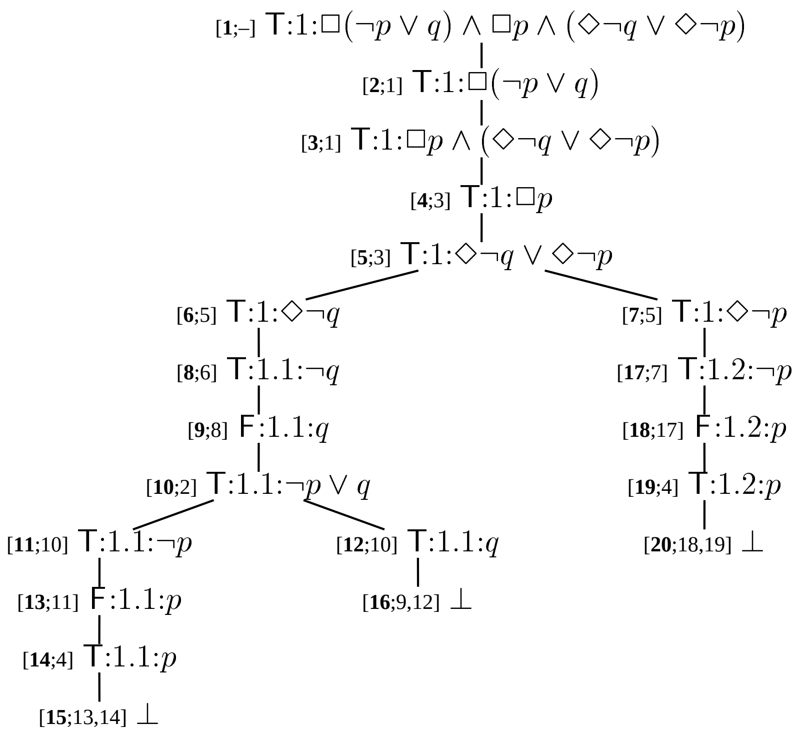

[**1**;–] T:1:$\Box(\neg p \vee q) \wedge \Box p \wedge (\Diamond\neg q \vee \Diamond\neg p)$
[**2**;1] T:1:$\Box(\neg p \vee q)$
[**3**;1] T:1:$\Box p \wedge (\Diamond\neg q \vee \Diamond\neg p)$
[**4**;3] T:1:$\Box p$
[**5**;3] T:1:$\Diamond\neg q \vee \Diamond\neg p$
[**6**;5] T:1:$\Diamond\neg q$
[**7**;5] T:1:$\Diamond\neg p$
[**8**;6] T:1.1:$\neg q$
[**17**;7] T:1.2:$\neg p$
[**9**;8] F:1.1:q
[**18**;17] F:1.2:p
[**10**;2] T:1.1:$\neg p \vee q$
[**19**;4] T:1.2:p
[**11**;10] T:1.1:$\neg p$
[**12**;10] T:1.1:q
[**20**;18,19] $\perp$
[**13**;11] F:1.1:p
[**16**;9,12] $\perp$
[**14**;4] T:1.1:p
[**15**;13,14] $\perp$

Figure 3.1: The tableau from Example 3.7.16.

is derived from 5; the two ν-formulae 3 and 4 cannot be made use of at that point, because the tableau does not contain any labels of the form $1.n$.

The labels that are introduced applying the tableau rule to premisses consisting of π-formulae 5 resp. 6 are $1.1 = 1.\lceil\neg q\rceil$ and $1.2 = 1.\lceil\neg p\rceil$. □

3.7.4 A Calculus for the Modal Logic $\mathbf{K}$ with Continuous Expansion Rule for ν-Formulae

The expansion rule of the calculus for modal logics described in the previous section is not continuous for premisses containing ν-formulae, because the condition has to be observed that the label introduced by a rule application to a ν-formula must occur in the premiss. For example, no conclusion can be deduced from the premiss $\Pi = \{\mathsf{T}{:}1{:}\Box p\}$ and nothing can be derived from $\Pi' = \{\mathsf{T}{:}1.1{:}q\}$, but $\mathsf{T}{:}1.1{:}p$ can be derived from the union of Π and Π'.

The main disadvantage of an expansion rule that is non-continuous for ν-formulae is that it makes it impossible to define a free-variable version of the calculus. The ν-formulae allow to derive many similar conclusions such as, for example, $\mathsf{T}{:}1.n{:}p$ for all $1.n$ occurring on the branch; to represent all these conclusions by a single conclusion $\mathsf{T}{:}1.X{:}p$ containing a free variable X is, however, only possible if all instances of

$\mathsf{T}{:}1.X{:}p$ are valid conclusions (in the example that is not the case for instances $\mathsf{T}{:}1.n{:}p$ where $1.n$ does *not* occur on the branch).

This problem can be avoided by dropping the pre-condition that the labels that are introduced by rule applications to ν-formulae must occur in the premiss; instead the additional position in the new labels are marked as being conditional. Thus, in the above example, $\mathsf{T}{:}1.(1){:}p$ can be derived from Π whether the label 1.1 occurs on the branch or not.

Then, however, to preserve soundness of the calculus, the existence of worlds corresponding to conditional labels has to be checked later on when branches are closed. For example, the (apparently contradictory) pair $\mathsf{T}{:}1.(1){:}p$ and $\mathsf{F}{:}1.(1){:}p$ is not necessarily inconsistent since the world $I(1.(1))$ may not exist in a tableau interpretation. Before declaring this pair to be inconsistent, we therefore have to ensure that $I(1.(1))$ is defined in all **L**-interpretations satisfying the tableau branch B that is to be closed. Fortunately, this knowledge can be deduced from other formulae on B. Thus, in our example, a formula like $G = \mathsf{T}{:}1.1{:}q$ on B would "justify" the use of the pair $\mathsf{T}{:}1.(1){:}p$ and $\mathsf{F}{:}1.(1){:}p$ for closing B since any **L**-interpretation $\langle \mathbf{m}, I\rangle$ satisfying B has to satisfy G, and, thus, $I(1.(1)) = I(1.1)$ has to be a world in the chosen model $\mathbf{m}$. The crucial point is that the label 1.1 of G is *unconditional* exactly in the *conditional* position of $1.(1)$. These observations are now extended to the general case of arbitrary labels from $CondLab(\mathbb{N})$.

Definition 3.7.17 A label $\sigma \in CondLab(\mathbb{N})$ with j-th position $[n_j]$ ($1 \le j \le |\sigma|$) is *justified* by a set $\Pi \in TabForm_{\mathrm{mod}}$ of modal tableau formulae if there is some subset Ψ of Π such that for every j:

1. some label occurring in Ψ has an unconditional but otherwise identical j-th position n_j; and
2. for all labels τ occurring in Ψ: if $|\tau| \ge j$, then the j-th position in τ is n_j or (n_j). □

The provision that the labels of complementary atomic formulae have to be justified to close a branch, makes the expansion rule more non-continuous for premisses containing such complementary formulae. That, however, is not really problematic, because for these premisses the rule is non-continuous anyway.

Except for the expansion rule and the set of labels, which is $CondLab(\mathbb{N})$ instead of $Lab(\mathbb{N})$, syntax and semantics of the new calculus $\mathcal{C}_{\mathrm{K}}^{\mathrm{con}}$ for the modal logic K is the same as that of the calculus $\mathcal{C}_{\mathrm{K}}$ defined in the previous section. The signatures are not extended; and the set $TabInterp$ of tableau interpretations defining the semantics of tableau consists of the canonical **L**-interpretations (Def. 3.7.7).

The new rule schemata for ν-formulae and for closing branches are shown in Table 3.7; and all schemata for the expansion rule of the calculus $\mathcal{C}_{\mathrm{K}}^{\mathrm{con}}$ are summarised in Table 3.8. The expansion rule $\mathcal{E}_{\mathrm{K}}^{\mathrm{con}}$ of $\mathcal{C}_{\mathrm{K}}^{\mathrm{con}}$ is formally defined as follows:

$$\frac{\sigma{:}\nu}{\nu_K(\sigma.(n))}$$
for all $n \in \mathbb{N}$

$$\frac{\begin{array}{c}\mathsf{T}{:}\sigma{:}F\\ \mathsf{F}{:}\sigma'{:}F\end{array}}{\bot}$$
where $[\sigma] = [\sigma']$, and
σ, σ' are justified
by formulae on the branch

Table 3.7: The new rule schemata of $\mathcal{C}_{\mathrm{K}}^{\mathrm{con}}$.

$$\frac{\alpha}{\begin{array}{c}\alpha_1\\ \alpha_2\end{array}} \qquad \frac{\beta}{\beta_1 \mid \beta_2} \qquad \frac{\mathsf{T}{:}\sigma{:}\Box F}{\mathsf{T}{:}\sigma.(n){:}F} \qquad \frac{\mathsf{F}{:}\sigma{:}\Diamond F}{\mathsf{F}{:}\sigma.(n){:}F} \qquad \frac{\mathsf{T}{:}\sigma{:}\Diamond F}{\mathsf{T}{:}\sigma.n{:}F}$$
for all $n \in \mathbb{N}$ where $n = \lceil F \rceil$

$$\frac{\mathsf{F}{:}\sigma{:}\Box F}{\mathsf{F}{:}\sigma.n{:}F} \qquad \frac{\mathsf{T}{:}\sigma{:}\neg F}{\mathsf{F}{:}\sigma{:}F} \qquad \frac{\mathsf{F}{:}\sigma{:}\neg F}{\mathsf{T}{:}\sigma{:}F} \qquad \frac{\begin{array}{c}\mathsf{T}{:}\sigma{:}F\\ \mathsf{F}{:}\sigma'{:}F\end{array}}{\bot}$$
where $n = \lceil \neg F \rceil$

where $[\sigma] = [\sigma']$, and
σ, σ' are justified
by formulae on the branch

Table 3.8: Expansion rule schemata for the calculus $\mathcal{C}_{\mathrm{K}}^{\mathrm{con}}$.

Definition 3.7.18 For all premisses $\Pi \subset \mathit{TabForm}_{\mathrm{mod}}$, the set $\mathcal{E}_{\mathrm{K}}^{\mathrm{con}}(\Pi)$ is the smallest set containing the following conclusions (where $\lceil\cdot\rceil$ is any bijection from $\mathit{Form}_{\mathrm{mod}}(\Sigma)$ to the set of natural numbers):

- $\{\{\alpha_1, \alpha_2\}\}$ for all $\alpha \in \Pi$,
- $\{\{\beta_1\}, \{\beta_2\}\}$ for all $\beta \in \Pi$,
- $\{\{\mathsf{T}{:}\sigma.(n){:}F\}\}$ for all $\mathsf{T}{:}\sigma{:}\Box F \in \Pi$ and all $n \in \mathbb{N}$,
- $\{\{\mathsf{F}{:}\sigma.(n){:}F\}\}$ for all $\mathsf{F}{:}\sigma{:}\Diamond F \in \Pi$ and all $n \in \mathbb{N}$,
- $\{\{\mathsf{F}{:}\sigma.n{:}F\}\}$ for all $\mathsf{F}{:}\sigma{:}\Box F \in \Pi$ where $n = \lceil \neg F \rceil$,
- $\{\{\mathsf{T}{:}\sigma.n{:}F\}\}$ for all $\mathsf{T}{:}\sigma{:}\Diamond F \in \Pi$ where $n = \lceil F \rceil$,
- $\{\{\bot\}\}$ if $\mathsf{T}{:}\sigma{:}F, \mathsf{F}{:}\sigma{:}F \in \Pi$ such that $[\sigma] = [\sigma']$, and σ, σ' are justified by Π.

□

The new expansion rule schemata for ν-formulae and branch closure can as well be used for other modal logics that are (a) serial (in which case the justification test is not needed anyway as the interpretation of all labels is defined), or that are (b) neither symmetric nor euclidean. Calculi for symmetric and euclidean logics are problematic because their expansion rules can shorten labels. For example, the tableau formula $\mathsf{T}{:}1{:}p$ is derived from $\mathsf{T}{:}1.(1){:}\Box p$ if the logic is symmetric. The semantics for serial logics guarantees that all labels define worlds, but in non-serial logics, the label 1 may be defined even though $1.(1)$ is undefined. Therefore, an additional mechanism is needed to ensure that the formula $\mathsf{T}{:}1{:}p$ is used to close a branch only if the label $1.(1)$ *is* defined. This can be achieved by attaching a set of labels to each tableau formula that all have to be justified when the formula is used to close a branch (see (Beckert & Goré, 1997)).

To prove completeness of $\mathcal{C}_{\mathrm{K}}^{\mathrm{con}}$, the definition of Hintikka sets has to take conditional labels into account (only Condition 1 differs from the definition of modal Hintikka sets without conditional labels, see Def. 3.7.9):

Definition 3.7.19 A set $\Xi \subset \mathit{TabForm}_{\mathrm{mod}}(\Sigma^*)$ of tableau formulae is a *modal* K-*Hintikka set with conditional labels* if it satisfies the following conditions:

1. There are no complementary atomic formulae $\mathsf{T}{:}\sigma{:}p$ and $\mathsf{F}{:}\sigma'{:}p$ in Ξ such that $[\sigma] = [\sigma']$ and σ, σ' are justified by Ξ.
2. If $\alpha \in \Xi$, then α_1 and α_2 are in Ξ.
3. If $\beta \in \Xi$, then β_1 or β_2 is in Ξ.
4. If $\mathsf{S}{:}\sigma{:}\nu \in \Xi$, then $\mathsf{S}{:}\sigma.(n){:}\nu_{\mathrm{K}} \in \Xi$ for *all* $n \in \mathbb{N}$.
5. If $\mathsf{S}{:}\sigma{:}\pi \in \Xi$, then $\mathsf{S}{:}\sigma.n{:}\pi_1 \in \Xi$ for *some* $n \in \mathbb{N}$. □

Lemma 3.7.20 *If Ξ is a modal* K*-Hintikka set with conditional labels, then*

1. *it is satisfied by some tableau interpretation in $\mathit{TabInterp}_{\mathrm{K}}(\Sigma^*)$, i.e., a canonical* K*-interpretation;*
2. *every tableau interpretation in $\mathit{TabInterp}_{\mathrm{K}}(\Sigma^*)$ satisfying the atomic tableau formulae in Ξ satisfies Ξ.*

Proof: Again, the second part of the lemma is easy to prove by induction on the structure of tableau formulae in Ξ.

Because of Condition 1 in the definition of modal K-Hintikka sets with conditional labels (Def. 3.7.19), a canonical tableau interpretation $\langle\langle W, R, V\rangle, I\rangle$ satisfying the atomic formulae in Ξ can be defined by:

- Let $W = \{[\sigma] \mid \sigma \in \mathit{Lab}$ is justified by $\Xi\}$;
- let $I(\sigma) = [\sigma]$ if $[\sigma] \in W$, and let $I(\sigma)$ be undefined otherwise;
- for all $\sigma, \tau \in W$, let $[\tau]$ be reachable from $[\sigma]$ iff $\tau = \sigma.n$ or $\tau = \sigma.(n)$ for some $n \in \mathbb{N}$;

- let $V(p) = \{[\sigma] \mid \mathsf{T}{:}\sigma{:}p \in \Xi\}$. □

Theorem 3.7.21 *The tableau calculus $\mathcal{C}_{\mathrm{K}}^{\mathrm{con}}$ for the modal logic* K *is sound and complete.*

Proof: The theorem can be proven in the same way as soundness and completeness of the calculus $\mathcal{C}_{\mathrm{K}}$ (Theorem 3.7.15).

The proof for Soundness Property 1 (appropriateness of the set of tableau interpretations) remains unchanged.

Proving that the calculus has the strong soundness of expansion property is somewhat more difficult, because now conditional labels may occur in a tableau. But only in the case of a premiss that allows to close a branch this leads to a real complication. In that case, the following lemma has to be applied, which follows immediately from the definitions: Let $\langle \mathbf{m}, I\rangle$ be a canonical **L**-interpretation, and let σ be a label justified by a set Π of tableau formulae. If $\langle \mathbf{m}, I\rangle$ satisfies the formulae in Π, then $I(\sigma)$ is defined.

The completeness properties are proved in the same way as for the calculus $\mathcal{C}_{\mathrm{K}}$—with the exceptions that the alternative Definition 3.7.19 of Hintikka sets and the alternative Hintikka Lemma 3.7.20 are used, which take conditional labels into account. □

Example 3.7.22 We continue from Example 3.7.22 and prove unsatisfiability of the same formula G using the new calculus $\mathcal{C}_{\mathrm{K}}^{\mathrm{con}}$. A closed tableau T^{con} for G that has been constructed using the expansion rule of $\mathcal{C}_{\mathrm{K}}^{\mathrm{con}}$ is shown in Figure 3.2.

Since the provision that labels introduced by applying the expansion rule to a ν-formula must already occur on the branch has been dropped, it is now possible to add formulae 6 and 7 although their label $1.(1)$ is new to the branch (it is, however, conditional in its second position).

The left branch B_1 of T^{con} is closed applying the tableau rule to a premiss containing the complementary pair $\mathsf{T}{:}1.(1){:}p$ and $\mathsf{F}{:}1.(1){:}p$ in nodes 7 and 14, respectively. The label $1.(1)$ of these atoms is justified on B_1 by formulae 10 and 11 whose label is 1.1. In this case, the complementary formulae contain conditional labels which are only justified by a third formula on the branch, so checking for justification is indispensable. The middle branch B_2 contains the complementary formulae $\mathsf{F}{:}1.1{:}q$ and $\mathsf{T}{:}1.(1){:}q$ in nodes 11 resp. 13. The label is again justified by formula 10 and formula 11, which in this case is part of the complementary pair. The right branch B_3 contains the pair $\mathsf{F}{:}1.2{:}p$ and $\mathsf{T}{:}1.(2){:}p$ of complementary atoms in nodes 18 resp. 19. The label $1.(2)$ of the formula in node 19 is justified by formula 18. □

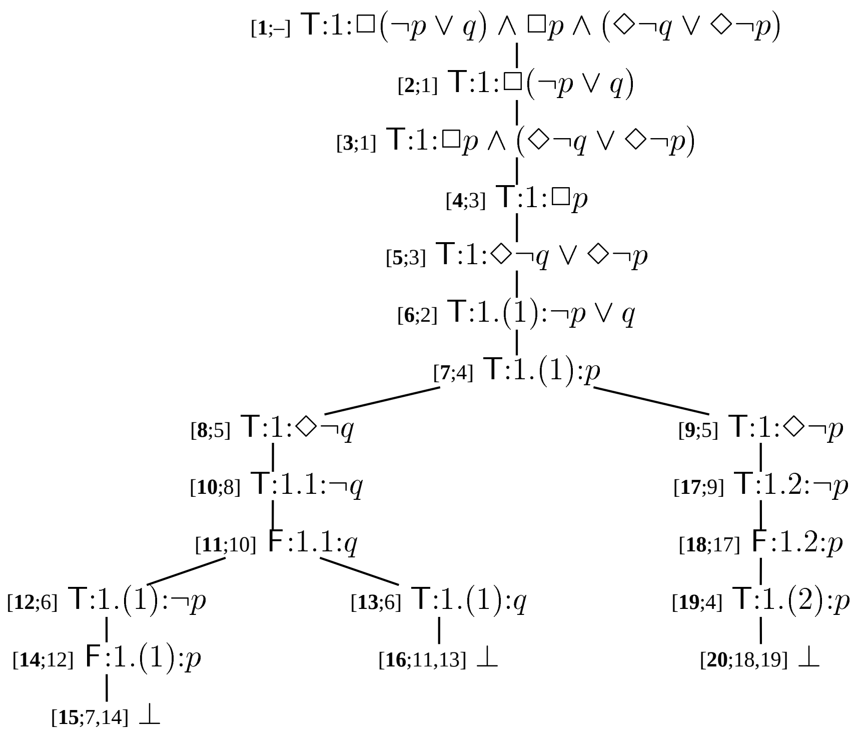

[1;–] T:1:$\Box(\neg p \vee q) \wedge \Box p \wedge (\Diamond\neg q \vee \Diamond\neg p)$
[2;1] T:1:$\Box(\neg p \vee q)$
[3;1] T:1:$\Box p \wedge (\Diamond\neg q \vee \Diamond\neg p)$
[4;3] T:1:$\Box p$
[5;3] T:1:$\Diamond\neg q \vee \Diamond\neg p$
[6;2] T:1.(1):$\neg p \vee q$
[7;4] T:1.(1):p
[8;5] T:1:$\Diamond\neg q$
[10;8] T:1.1:$\neg q$
[11;10] F:1.1:q
[12;6] T:1.(1):$\neg p$
[14;12] F:1.(1):p
[15;7,14] $\perp$
[13;6] T:1.(1):q
[16;11,13] $\perp$
[9;5] T:1:$\Diamond\neg p$
[17;9] T:1.2:$\neg p$
[18;17] F:1.2:p
[19;4] T:1.(2):p
[20;18,19] $\perp$

Figure 3.2: The tableau T^{con} from Example 3.7.22.

3.8 Ideal Calculi for the Set Logics MLSS and MLSSF

3.8.1 Overview

In this section, we present an ideal tableau calculus $\mathcal{C}_{\text{MLSS}}$ for the logic MLSS, which is a decidable fragment of set theory. Furthermore, we describe an extension $\mathcal{C}_{\text{MLSSF}}$ of our calculus for the bigger fragment MLSSF consisting of MLSS enriched with free (uninterpreted) function symbols (Section 2.6). The ideal calculi $\mathcal{C}_{\text{MLSS}}$ and $\mathcal{C}_{\text{MLSSF}}$ are slight variations of the calculi described in (Beckert & Hartmer, 1998; Hartmer, 1997), which are not ideal. They are extensions of the tableau-based calculus for MLSS described in (Cantone, 1997).

The calculus $\mathcal{C}_{\text{MLSS}}$ can be used to construct a sound and complete decision procedure for MLSS. It does not require formulae to be in normal form, whereas Cantone's calculus only contains rules for normalised MLSS atoms (which are not allowed to contain complex terms) and relies on a pre-processing transformation for normalising formulae. The handling of free function symbols in the extended calculus $\mathcal{C}_{\text{MLSSF}}$ for MLSSF employs E-unification techniques for reducing the search space by finding term pairs that, when shown to be equal, close a tableau branch.

Several other methods for handling set theory in tableau calculi or the sequent calculus (without the restriction to a certain fragment) have been proposed: Brown (1978)

$$\frac{\alpha}{\begin{array}{c}\alpha_1\\ \vdots \\ \alpha_n\end{array}} \qquad\qquad \frac{\beta}{\beta_1 \mid \cdots \mid \beta_n}$$

Table 3.9: Generalised expansion rule schemata for α- and β-formulae.

presents a first-order sequent calculus that contains special rules for many set theoretic symbols. De Nivelle (1997) and Pastre (1978) introduce sequent calculi for set theory. Shults (1997) describes a tableau calculus with special set theoretic rules. All these calculi, however, are incomplete (no semi-decision procedures).

3.8.2 The Set of Labels and the Extension of Signatures

Labels The models of MLSS and MLSSF consist of only one world. We use the label $*$ to represent this single world. Thus, $Lab = \{*\}$, and $*$ is the initial label. As in calculi for PL1, the abbreviation $\mathsf{S}{:}G$ is used for tableau formulae, i.e., the label $*$ is omitted.

Extended Signatures An inequality $\mathsf{F}{:}(s \approx t)$ implies the existence of an element that occurs in only one of the two sets s and t and not in the other. The expansion rule of our calculus makes use of that fact by introducing a Skolem constant representing the existing element when it is applied to an inequality. For skolemisation we use an infinite set $F^{sko}(\Sigma)$ of constants that is disjoint from $F(\Sigma)$.

The extension of an MLSS or MLSSF signature $\Sigma = \langle P(\Sigma), F(\Sigma), \alpha(\Sigma)\rangle$ used for constructing tableau formulae is thus

$$\Sigma^* = \langle P(\Sigma), F(\Sigma) \cup F^{sko}(\Sigma), \alpha(\Sigma) \cup \alpha^{sko}(\Sigma)\rangle \ ,$$

where $\alpha^{sko}(\Sigma)(c) = 0$ for all $c \in F^{sko}(\Sigma)$.

3.8.3 The Tableau Expansion Rule for MLSS

Schemata for non-atomic formulae The non-atomic MLSS and MLSSF formulae are divided into α- and β-formulae as usual. The expansion rule $\mathcal{E}_{\mathrm{MLSS}}$ of $\mathcal{C}_{\mathrm{MLSS}}$ is defined for premisses consisting of α- and β-formulae by the standard schemata shown in Table 3.9 (we use a generalisation where the conclusions derived from α- and β-formulae can consist of an arbitrary number of tableau formulae resp. extensions).

Name	α	$\alpha_1, \ldots, \alpha_n$
(R1)	$\mathsf{T}{:}(s \sqsubseteq t)$	$\mathsf{T}{:}(s \approx s \sqcap t)$
(R2)	$\mathsf{F}{:}(s \sqsubseteq t)$	$\mathsf{F}{:}(s \approx s \sqcap t)$
(R3)	$\mathsf{T}{:}(s \in t_1 \sqcap t_2)$	$\mathsf{T}{:}(s \in t_1),\ \mathsf{T}{:}(s \in t_2)$
(R4)	$\mathsf{T}{:}(s \in t_1 \setminus t_2)$	$\mathsf{T}{:}(s \in t_1),\ \mathsf{F}{:}(s \in t_2)$
(R5)	$\mathsf{F}{:}(s \in t_1 \sqcup t_2)$	$\mathsf{F}{:}(s \in t_1),\ \mathsf{F}{:}(s \in t_2)$
(R6)	$\mathsf{F}{:}(s \in \{t_1, \ldots, t_n\}_n)$	$\mathsf{F}{:}(s \approx t_1), \ldots, \mathsf{F}{:}(s \approx t_n)$

Name	β	$\beta_1, \ldots, \beta_n$
(R7)	$\mathsf{T}{:}(s \in t_1 \sqcup t_2)$	$\mathsf{T}{:}(s \in t_1),\ \mathsf{T}{:}(s \in t_2)$
(R8)	$\mathsf{F}{:}(s \in t_1 \sqcap t_2)$	$\mathsf{F}{:}(s \in t_1),\ \mathsf{F}{:}(s \in t_2)$
(R9)	$\mathsf{F}{:}(s \in t_1 \setminus t_2)$	$\mathsf{F}{:}(s \in t_1),\ \mathsf{T}{:}(s \in t_2)$
(R10)	$\mathsf{T}{:}(s \in \{t_1, \ldots, t_n\}_n)$	$\mathsf{T}{:}(s \approx t_1), \ldots, \mathsf{T}{:}(s \approx t_n)$

Table 3.10: Rule schemata for splitting complex set terms.

Schemata for splitting complex set terms There are ten different expansion rule schemata for splitting complex set terms; they apply simple set theoretic lemmata such as "if $s \in t_1 \cup t_2$ then $s \in t_1$ or $s \in t_2$" to (a) eliminate atoms containing the set inclusion predicate $\sqsubseteq$ and replace them with (in-)equalities, and to (b) split complex terms on the right side of the membership predicate $\in$ into their constituents. These schemata can be described as instances of the generalised schema for α- and β-formulae from Table 3.9; they are listed in Table 3.10.

Schemata for handling equalities and inequalities There are three types of special expansion rule schemata for handling the equality and inequality of sets. First, there are two schemata ((EQ1) and (EQ2) in Table 3.11) that allow to "apply" an equality $\mathsf{T}{:}(t_1 \approx t_2)$ to other atoms in a very restricted way: an equality can only be applied at the top level and only to the right side of a positive atom whose predicate symbol is $\in$. That is, an equality can only be applied to derive one of the atoms $\mathsf{T}{:}(s \in t_1)$ and $\mathsf{T}{:}(s \in t_2)$ from the other one. This restriction is important, because the possibility to apply equalities arbitrarily to other atoms would lead to a much larger search space.

Second, it is possible to derive the inquality $\mathsf{F}{:}(s_1 \approx s_2)$ from $\mathsf{T}{:}(s_1 \in t)$ and $\mathsf{F}{:}(s_2 \in t)$ ((R11) in Table 3.11). This expansion rule schema is based on the fact that two objects are different if one of them is an element of some set and the other is not.

Third, the opposite of the above holds as well: if two sets t_1 and t_2 are different, then one of them contains an element e that is not an element of the other set. Unfortunately, this leads to a branching rule schema ((R12) in Table 3.11), because e can be an element of t_1 (and not of t_2) or of t_2 (and not of t_1). Instead of introducing a *new* symbol to represent the unknown element, we use a *Skolem constant assignment* that

$$\begin{array}{c} \mathsf{T}{:}(t_1 \approx t_2) \\ \mathsf{T}{:}(s \sqsubseteq t_1) \\ \hline \mathsf{T}{:}(s \sqsubseteq t_2) \end{array} \qquad \begin{array}{c} \mathsf{T}{:}(t_1 \approx t_2) \\ \mathsf{T}{:}(s \sqsubseteq t_2) \\ \hline \mathsf{T}{:}(s \sqsubseteq t_1) \end{array} \qquad \begin{array}{c} \mathsf{T}{:}(s_1 \sqsubseteq t) \\ \mathsf{F}{:}(s_2 \sqsubseteq t) \\ \hline \mathsf{F}{:}(s_1 \approx s_2) \end{array}$$

(EQ1) (EQ2) (R11)

$$\begin{array}{c|c} \multicolumn{2}{c}{\mathsf{F}{:}(t_1 \approx t_2)} \\ \hline \mathsf{T}{:}(c \sqsubseteq t_1) & \mathsf{F}{:}(c \sqsubseteq t_1) \\ \mathsf{F}{:}(c \sqsubseteq t_2) & \mathsf{T}{:}(c \sqsubseteq t_2) \end{array} \qquad\qquad \begin{array}{c|c} \hline \mathsf{T}{:}(s \sqsubseteq t) & \mathsf{F}{:}(s \sqsubseteq t) \end{array}$$

where $c = sko_{\mathrm{MLSSF}}(\mathsf{F}{:}(t_1 \approx t_2))$
(R12)

where s resp. $\{\ldots, s, \ldots\}$ and t resp. $\{\ldots, t, \ldots\}$ are top-level terms on the branch
(Cut)

Table 3.11: Rule schemata handling equalities and inequalities, and the cut rule schema.

uniquely assigns a constant to each inequality (similar to the Skolem term assignment from Definition 3.6.2 that is used to assign a Skolem term to each δ-formula of PL1); thus, monotonicity and idealness of the expansion rule is preserved.

Definition 3.8.1 Given an MLSS or MLSSF signature Σ, a *Skolem constant assignment* (for MLSS resp. MLSSF) is a function sko_{MLSSF} assigning to each inequality $\phi = \mathsf{F}{:}(t_1 \approx t_2)$ in $\mathit{TabForm}(\Sigma^*)$ a constant $sko_{\mathrm{MLSSF}}(\phi) = c \in F^{sko}(\Sigma)$ such that, for all $c' \in F^{sko}(\Sigma)$, if c' occurs in ϕ, then $c > c'$ where $>$ is an arbitrary but fixed ordering on $F^{sko}(\Sigma)$. □

The cut rule schema The cut rule schema (Table 3.11) may be applied to extend a tableau branch B using any atom $s \sqsubseteq t$ as cut formula where the set terms s and t occur

- as top-level arguments of an atom on B, or
- as arguments on the second level if the top-level function symbol is $\{\cdot\}_n$.

In practice, the cut rule schema is rarely needed to construct a proof; it is, for example, needed to detect implicit membership cycles on a branch; see Section 3.8.3.

Example 3.8.2 If $\mathsf{T}{:}(t_1 \sqsubseteq \{t_2,\, t_3 \sqcap t_4\})$ and $\mathsf{T}{:}(t_5 \sqcap t_6 \approx t_7)$ are atoms on the branch, then $t_1, t_2, (t_3 \sqcap t_4), (t_5 \sqcap t_6), t_7$ may be used in a cut rule application and t_3, t_4, t_5, t_6 may not be used (unless that is justified by other atoms on the branch). □

Schemata for branch closure An application of the expansion rule of $\mathcal{C}_{\mathrm{MLSS}}$ adds formulae to a tableau branch being true in all set structures that are models of the expanded branch; the purpose of schemata for branch closure is to detect inconsistencies, i.e., formulae on a branch that are false in all set structures. There are four types of inconsistencies that have to be considered:

1. In no set structure both a formula ϕ and its complement $\overline{\phi}$ are true; thus, as in all calculi, a premiss containing a pair $\phi, \overline{\phi}$ allows to deduce $\perp$ (for completeness it is sufficient to only consider complementary *atoms*).
2. No object is an element of the empty set; thus, an atom of the form $\mathsf{T}{:}(t \sqsubseteq \emptyset)$ is unsatisfiable.
3. As no object is different from itself, atoms of the form $\mathsf{F}{:}(t \approx t)$ are unsatisfiable.
4. The existence of a membership cycle, i.e., of sets $u_1, \ldots, u_k$ such that $u_i \in u_{i+1}$ $(1 \leq i < k)$ and $u_k \in u_1$, would contradict the Axiom of Foundation. In fact, there are by construction no sets in the von Neumann hierarchy that form a membership cycle. Thus, atoms defining a membership cycle allow to close a branch; in particular, $\mathsf{T}{:}(t \sqsubseteq t)$ is unsatisfiable.

The following is a formal definition of the expansion rule of the calculus $\mathcal{C}_{\mathrm{MLSS}}$:

Definition 3.8.3 Let Σ be an MLSS signature.

For all premisses $\Pi \subset \mathit{TabForm}_{\mathrm{MLSS}}(\Sigma^*)$, the set $\mathcal{E}_{\mathrm{MLSS}}(\Sigma^*)(\Pi)$ consists of the the following conclusions:

– $\{\{\alpha_1, \ldots, \alpha_n\}\}$
for all $\alpha \in \Pi$ (Tables 3.1 and 3.10);

– $\{\{\beta_1\}, \ldots, \{\beta_n\}\}$
for all $\beta \in \Pi$ (Tables 3.1 and 3.10);

– $\{\{\mathsf{T}{:}(s \sqsubseteq t_2)\}\}$
for all (a) $\mathsf{T}{:}(s \sqsubseteq t_1)$ and (b) $\mathsf{T}{:}(t_1 \approx t_2)$ or $\mathsf{T}{:}(t_2 \approx t_1)$ in Π;

– $\{\{\mathsf{F}{:}(s_1 \approx s_2)\}\}$
for all $\mathsf{T}{:}(s_1 \sqsubseteq t)$ and $\mathsf{F}{:}(s_2 \sqsubseteq t)$ in Π;

– $\{\{\mathsf{T}{:}(s \sqsubseteq t)\}, \{\mathsf{F}{:}(s \sqsubseteq t)\}\}$
for all set terms s and t such that (a) s or $\{\ldots, s, \ldots\}_n$ and (b) t or $\{\ldots, t, \ldots\}_n$ occur as top-level terms in atoms in Π;

– $\{\{\mathsf{T}{:}(c \sqsubseteq t_1), \mathsf{F}{:}(c \sqsubseteq t_2)\}, \{\mathsf{F}{:}(c \sqsubseteq t_1), \mathsf{T}{:}(c \sqsubseteq t_2)\}\}$
for all $\mathsf{F}{:}(t_1 \approx t_2)$ in Π, where $c = \mathit{sko}_{\mathrm{MLSSF}}(\mathsf{F}{:}(t_1 \approx t_2))$;

– $\{\{\bot\}\}$
  if

  1. $\mathsf{T}{:}\sigma{:}G$, $\mathsf{F}{:}\sigma{:}G \in \Pi$ for any atom G,
  2. $\mathsf{T}{:}(t \sqsubseteq \emptyset) \in \Pi$,
  3. $\mathsf{F}{:}(t \approx t) \in \Pi$,
  4. for some $k \geq 1$, there are atoms $\mathsf{T}{:}(t_i \sqsubseteq t_{i+1})$ $(1 \leq i < k)$ and $\mathsf{T}{:}(t_k \sqsubseteq t_1)$ in Π.

$\square$

3.8.4 Soundness, Completeness, Termination

The calculus $\mathcal{C}^{\mathrm{MLSS}}$ defined in the previous section is sound and complete (a proof can be found in (Hartmer, 1997)). It has the soundness and completeness ensuring properties from Definitions 3.5.6 and 3.5.8.

Theorem 3.8.4 *The calculus $\mathcal{C}^{\mathrm{MLSS}}$ for* MLSS *is sound and complete.*

Without further restrictions, the calculus $\mathcal{C}^{\mathrm{MLSS}}$ is not a decision procedure. The rule schema for inequalities ((R12) in Table 3.11) introduces additional constants, and the cut rule schema can—in connection with schema (R11)—be used to construct new inequalities from these constants; the interaction of these expansion rule schemata can lead to infinite branches.

Fortunately, the calculus can easily be turned into a decision procedure, observing the completeness preserving restriction that chains $c_1, c_2, \ldots$ where c_i is added to the branch applying the schema (R12) for inequalities to an inequality that contains the constant c_{i-1} must not be infinite; their maximal length is the number of (sub-)terms the formula set whose satisfiability is to be checked.

Definition 3.8.5 Let $T_1, \ldots, T_k$ be a sequence of tableaux for a set $\mathfrak{F}$ of MLSS-formulae that has been constructed using the expansion rule $\mathcal{E}_{\mathrm{MLSS}}$ (Def. 3.8.3).

The *rank* $rk(s)$ of a set term s in this sequence of tableaux is defined as follows: If s occurs in $\mathfrak{F}$ or has been generated by an application of rule schemata (R1) and (R2), then $rk(s) = 0$; otherwise, i.e., if s is a constant that has been introduced by applying rule (R12) to an inequality $\mathsf{F}{:}(t_1 \approx t_2)$, then its rank is $rk(s) = 1 + \max\{rk(t_1), rk(t_2)\}$.

$\square$

Definition 3.8.6 A tableau T for a set $\mathfrak{F}$ of MLSS-formulae is *exhausted*, if no expansion rule application to T is possible without

- introducing a constant whose rank is greater than the number of (sub-)terms in $\mathfrak{F}$, or

- adding only tableau formulae to a branch B of T that already occur on B.

Note that a tableau that is exhausted (Def. 3.8.6) is not necessarily fully expanded (Def. 3.5.5).

Theorem 3.8.7 *There is an exhausted non-closed $\mathcal{C}_{\mathrm{MLSS}}$-tableau for a set $\mathfrak{F}$ of MLSS-formulae if and only if $\mathfrak{F}$ is satisfiable.*

Thus, if a sequence of tableaux for a set $\mathfrak{F}$ of MLSS-formulae is constructed in a *fair* way (i.e., all possible rule applications are executed sooner or later), then the construction will terminate after a finite number of steps with a tableau that is (a) closed, in which case $\mathfrak{F}$ is unsatisfiable, or (b) exhausted, in which case $\mathfrak{F}$ is satisfiable.

3.8.5 Restricting the Search Space

Although the search space for a $\mathcal{C}_{\mathrm{MLSS}}$-tableau proof is finite if the restriction described in the previous section is used, it is very large because of the indeterminism of the cut rule schema and because the number of constants that can be introduced is exponential in the size of the formula set whose unsatisfiability it to be proven.

Fortunately, it is possible to impose a strong restriction on cut rule applications, which at the same time restricts the number of constants that are introduced, because a constant c_k of rank k can only be created from an inequality containing a constant c_{k-1} of rank $k-1$ after the cut rule has been applied using an atom as cut formula that contains c_{k-1}. The idea is to apply all rule schemata except the cut rule schema until further applications do not add new formulae to branches, and then to construct a *realisation* of an open branch. The realisation of a branch B approximates a set structure satisfying the formulae on B (if the branch is satisfiable); it satisfies at least all atoms of the form $\mathsf{T}{:}(t_1 \sqsubseteq t_2)$ on B. If the realisation does not satisfy all the other atoms on B as well, it can be used to find cut rule applications that are (at least potentially) useful.

The switching between the expansion of tableau branches and the construction of possible models, and the way in which we construct models are similar to the method described in (Cantone, 1997).

Definition 3.8.8 Let Σ be an MLSS signature; and let Π be a set of MLSS-tableau formulae over Σ^*.

- $\mathcal{G}$ denotes the set of all set (sub-)terms over Σ occurring in Π;
- $\mathcal{V}$ denotes the set of (a) all set terms $t \in \mathcal{G}$ such that $\mathsf{T}{:}(t \sqsubseteq s)$ occurs in Π and (b) all constants from Σ that occur in Π;
- $\mathcal{T}$ denotes the set of all constants occurring in Π that are not in $\mathcal{V}$;

- $\sim$ denotes the equivalence relation on $\mathcal{G} \cup \mathcal{T}$ induced by the equalities in Π;
- $\mathcal{T}'$ denotes the set of all $c \in \mathcal{T}$ such that $c \not\sim s$ for all $s \in \mathcal{G}$;
- $\mathcal{V}'$ denotes the set $(\mathcal{V} \cup \mathcal{T}) \setminus \mathcal{T}'$;
- u_c is, for each $c \in \mathcal{T}'$, an element of the von-Neumann hierarchy $\mathfrak{V}$ that is different from all $u_{c'}$ with $c \neq c'$. □

Note, that $\mathcal{T}'$ contains the constants that have been introduced by applying the expansion rule schema for inequalities (R12) and that are not equal to other terms (w.r.t. the equalities on the branch). The interpretation of these constants has to be different from the interpretation of all other terms, whereas different terms in $\mathcal{V}'$ may have the same interpretation.

Definition 3.8.9 Let Σ be an MLSS signature; let Π be a set of tableau formulae over Σ^*; and let t be a set term in Π. Then the set $\mathcal{P}_\Pi(t)$ of *implicit predecessors* of t is defined by:

1. $\mathcal{P}_\Pi(\emptyset) = \emptyset$;
2. $\mathcal{P}_\Pi(c) = \{s \in \mathcal{V} \cup \mathcal{T} \mid \mathsf{T}{:}(s \sqsubseteq c) \in \Pi\}$ for constants c;
3. $\mathcal{P}_\Pi(t_1 \sqcup t_2) = \mathcal{P}_\Pi(t_1) \cup \mathcal{P}(t_2)$;
4. $\mathcal{P}_\Pi(t_1 \sqcap t_2) = \mathcal{P}_\Pi(t_1) \cap \mathcal{P}_\Pi(t_2)$;
5. $\mathcal{P}_\Pi(t_1 \setminus t_2) = \mathcal{P}_\Pi(t_1) \setminus \mathcal{P}_\Pi(t_2)$; and
6. $\mathcal{P}_\Pi(\{t_1, \ldots, t_n\}_n) = \{s \in \mathcal{V} \cup \mathcal{T} \mid \mathsf{T}{:}(s \sqsubseteq \{t_1, \ldots, t_n\}_n) \in \Pi\} \cup \{t_1, \ldots, t_n\}$. □

The sets of implicit predecessors can be used to detect implicit membership cycles. If, for example, $s \in \mathcal{P}_\Pi(t), t \in \mathcal{P}_\Pi(s)$ for some terms s, t, then the branch can be closed, and it is not necessary to apply the expansion rule (especially the cut rule schema) to make the cycle explicit. Thus, using the predecessor sets, we can strengthen the calculus by adding another rule schema for closing branches:

Definition 3.8.10 The calculus $\mathcal{C}'_{\mathrm{MLSS}}$ is identical to $\mathcal{C}_{\mathrm{MLSS}}$ except for its expansion rule $\mathcal{E}'_{\mathrm{MLSS}}$, which is defined as follows:

For all MLSS signatures Σ and all premisses $\Pi \subset \mathit{TabForm}_{\mathrm{MLSS}}(\Sigma^*)$, the set $\mathcal{E}'_{\mathrm{MLSS}}(\Sigma)(\Pi)$ consists of

– the conclusions in $\mathcal{E}_{\mathrm{MLSS}}(\Sigma)(\Pi)$ (Def. 3.8.3) and

– the conclusion $\{\{\bot\}\}$ if the sets of implicit predecessors of terms in Π contain a cycle, i.e., there are set terms $t_1, \ldots, t_n$ occurring as (sub-)terms in Π such that $t_1 \in \mathcal{P}_\Pi(t_2), \ldots, t_{n-1} \in \mathcal{P}(t_n), t_n \in \mathcal{P}_\Pi(t_1)$. $\Box$

Theorem 3.8.11 *The calculus $\mathcal{C}'_{\mathrm{MLSS}}$ for* MLSS *is sound and complete.*

The set $\mathcal{P}_\Pi(t)$ of implicit predecessors contains those terms denoting elements of the set represented by t whose membership can be deduced from atoms in Π of the form $\mathsf{T}{:}(s \sqsubseteq a)$ (where a is a constant) and applying the definition of the set operators. The *realisation* of a set of tableau formulae goes beyond that: it is a partial definition of a set structure (different terms may be interpreted by the same set).

Definition 3.8.12 Let Σ be an MLSS signature; and let Π be a set of tableau formulae over Σ^* not containing $\bot$. The *realisation* $\mathcal{R}_\Pi(t)$ of a term t occurring in Π is defined by:

1. $\mathcal{R}_\Pi(t) = \emptyset$ if $t = \emptyset$,
2. $\mathcal{R}_\Pi(t) = \{\mathcal{R}_\Pi(s) \mid s \in \mathcal{P}_\Pi(t)\} \cup \{u_t\}$ if $t \in T'$,[3] and
3. $\mathcal{R}_\Pi(t) = \{\mathcal{R}_\Pi(s) \mid s \in \mathcal{P}_\Pi(t)\}$ otherwise. $\Box$

The realisation of a term can be effectively computed and can be used to restrict the application of the cut rule schema: provided a branch B is exhausted w.r.t. all other expansion rule schemata, the cut rule schema has only to be applied to terms occurring in atoms that are not satisfied by the realisation $\mathcal{R}_\Pi(t)$ where $\Pi = \mathit{Form}(B)$ (if there is no such atom, then B is satisfiable and we are done).

If, for example, $\mathsf{F}{:}(t_1 \sqsubseteq t_2)$ is in Π but $\mathcal{R}_\Pi(t_1) \in \mathcal{R}_\Pi(t_2)$, then there has to be a term s such that (a) $\mathcal{R}_\Pi(s) = \mathcal{R}_\Pi(t_1)$, i.e., the realisation of s is the same as that of t_1, and (b) s is an implicit member of t_2, i.e., $s \in \mathcal{P}_\Pi(t_2)$—but that membership is not (yet) made explicit on the branch (there is no atom $\mathsf{T}{:}(s \sqsubseteq t_2)$ in Π). In that case, the cut rule schema is applied using the atom $\mathsf{T}{:}(s \sqsubseteq t_2)$ as cut formula.

The following theorem states that completeness of the calculus $\mathcal{C}'_{\mathrm{MLSS}}$ is preserved if realisations are used to restrict applications of the cut rule schema (a proof can be found in (Hartmer, 1997)). Note that that the calculus is not weakened, but that this restriction is a technique for making proof procedures based on $\mathcal{C}'_{\mathrm{MLSS}}$ more efficient.

Definition 3.8.13 If $\mathfrak{F}$ is an unsatisfiable set of MLSS-formulae, then there is a $\mathcal{C}'_{\mathrm{MLSS}}$-tableau proof $T_1, \ldots, T_n$ for $\mathfrak{F}$ that is constructed observing the following restriction:

The cut rule schema may only be applied to construct a tableau T_{i+1} from T_i if the branch B of T_i that is expanded

[3] One has to make sure that the u_c's are different from $\mathcal{R}_\Pi(t)$ for all terms t; it is always possible to choose such u_c's.

1. is not closed; and
2. no other expansion rule schema can be used to add a formula to B that is not already on B;

and the cut formula $\mathsf{T}{:}(s \approx t)$ used for expansion satisfies one of the following conditions, where $\Pi = \mathit{Form}(B)$:

1. (a) $\mathsf{T}{:}(t \approx t') \in \Pi$,
   (b) $\mathcal{R}_\Pi(t) \neq \mathcal{R}_\Pi(t')$, and
   (c) i. $s \in \mathcal{P}_\Pi(t)$, $s \notin \mathcal{P}_\Pi(t')$, and $\mathsf{F}{:}(s \sqsubseteq t) \notin \Pi$, or
       ii. $s \in \mathcal{P}_\Pi(t')$, $s \notin \mathcal{P}_\Pi(t)$, and $\mathsf{T}{:}(s \sqsubseteq t') \notin \Pi$,
2. (a) $\mathsf{F}{:}(t \approx t')$, $\mathsf{F}{:}(c \sqsubseteq t)$, and $\mathsf{T}{:}(c \sqsubseteq t') \in \Pi$ (for some constant c),
   (b) $\mathcal{R}_\Pi(t) = \mathcal{R}_\Pi(t')$, $\mathcal{R}_\Pi(s) = \mathcal{R}_\Pi(c)$, and
   (c) $s \in \mathcal{P}_\Pi(t)$, $s \notin \mathcal{P}_\Pi(t')$, and $\mathsf{T}{:}(s \sqsubseteq t) \notin \Pi$,
3. (a) $\mathsf{F}{:}(t' \sqsubseteq t) \in \Pi$,
   (b) $\mathcal{R}_\Pi(t') \in \mathcal{R}_\Pi(t)$, $\mathcal{R}_\Pi(s) = \mathcal{R}_\Pi(t')$, and
   (c) $s \in \mathcal{P}_\Pi(t)$, and $\mathsf{T}{:}(s \sqsubseteq t) \notin \Pi$. □

3.8.6 A Comparison with Cantone's Calculus

The calculus $\mathcal{C}_{\mathrm{MLSS}}$ for MLSS described in the previous sections is similar to that presented by Cantone (1997). The main difference is that Cantone's calculus is restricted to *normalised* atoms, i.e., atoms not containing complex set terms:

Definition 3.8.14 An atomic MLLS-tableau formula ϕ is *normalised* iff it is of the form

– $\mathsf{S}{:}(a \sqsubseteq b)$,
– $\mathsf{S}{:}(a \approx b)$,
– $\mathsf{T}{:}(a \approx b \sqcup c)$,
– $\mathsf{T}{:}(a \approx b \sqcap c)$,
– $\mathsf{T}{:}(a \approx b \setminus c)$, or
– $\mathsf{T}{:}(a \approx \{b_1, \ldots, b_n\}_n)$ ($n \geq 1$),

where a, b, c and $b_1, \ldots, b_n$ are constants. □

There is a satisfiability preserving transformation of arbitrary finite sets of MLSS-tableau formulae into sets of normalised atoms by introducing new constants as abbreviations for complex set terms. For example,

$$\mathsf{T}{:}(a \sqsubseteq (b \sqcap b'))$$

is replaced by

$$\mathsf{T}{:}(c \approx (b \sqcap b')) \text{ and } \mathsf{T}{:}(a \sqsubseteq c)$$

where c is a new constant. The overhead for computing the transformation is negligible, because its complexity is polynomial in the size of the set to be transformed. However, the introduction of new constants leads to a much bigger search space, even more so as all these new constants occur in equalities.

Our rule schemata (R7), (R3), (R4), and (R10) are—in combination with rule schemata (EQ1) and (EQ2)—extensions for handling atoms with *complex* set terms of the corresponding rule schemata in Cantone's calculus. For example, our rule schema (R3), that allows to derive

$$\mathsf{T}{:}(a \sqsubseteq b) \text{ and } \mathsf{T}{:}(a \sqsubseteq b') \quad \text{from} \quad \mathsf{T}{:}(a \sqsubseteq (b \sqcap b'))\ ,$$

corresponds to Cantone's rule schema that allows to derive

$$\mathsf{T}{:}(a \sqsubseteq b) \text{ and } \mathsf{T}{:}(a \sqsubseteq b') \quad \text{from} \quad \mathsf{T}{:}(c \approx (b \sqcap b')) \text{ and } \mathsf{T}{:}(a \sqsubseteq c)$$

(for all a, b, c).

There are no rule schemata in Cantone's calculus corresponding to our rule schemata (R5), (R8), and (R9) for atoms expressing negated membership. Consider the three atoms

$$\phi = \mathsf{F}{:}(c \sqsubseteq (b_1 \sqcup b_2) \setminus b_3),\ \ \psi_1 = \mathsf{T}{:}(c \sqsubseteq b_1),\ \ \psi_2 = \mathsf{F}{:}(c \sqsubseteq b_3)$$

whose conjunction is unsatisfiable. To close a branch containing these atoms, our rule schemata (R9) and (R5) are applied to split the atom ϕ and derive that one of the complements $\overline{\psi_1}$ and $\overline{\psi_2}$ holds, which allows to close the two resulting sub-branches (see the tableau[4] in Figure 3.3 (a)). Since no rule schemata for splitting ϕ exist in Cantone's calculus, instead schemata for positive membership atoms have to be used to derive $\overline{\phi}$ from ψ_1 and ψ_2: first, ϕ has to be normalised, the result are the atoms

$$\mathsf{F}{:}(c \sqsubseteq d_1),\ \ \mathsf{T}{:}(d_1 \approx d_2 \setminus b_3),\ \ \mathsf{T}{:}(d_2 \approx b_1 \sqcup b_2)$$

where d_1 abbreviates $(b_1 \sqcup b_2) \setminus b_3$ and d_2 abbreviates $b_1 \sqcup b_2$. Then, with two rule applications, $\mathsf{T}{:}(c \sqsubseteq d_2)$ and $\mathsf{T}{:}(c \sqsubseteq d_1)$ are derived. The latter atom can be used to close the branch; it corresponds to the non-normalised atom $\overline{\phi}$ (see the tableau Figure 3.3 (b)).

[4] The i-th node in the tableau is labelled with $[i; j; \text{(R)}]$, which indicates that its formula has been derived applying the expansion rule schema (R) of $\mathcal{C}_{\mathrm{MLSSF}}$ to a premiss consisting of the formula in the j-th node.

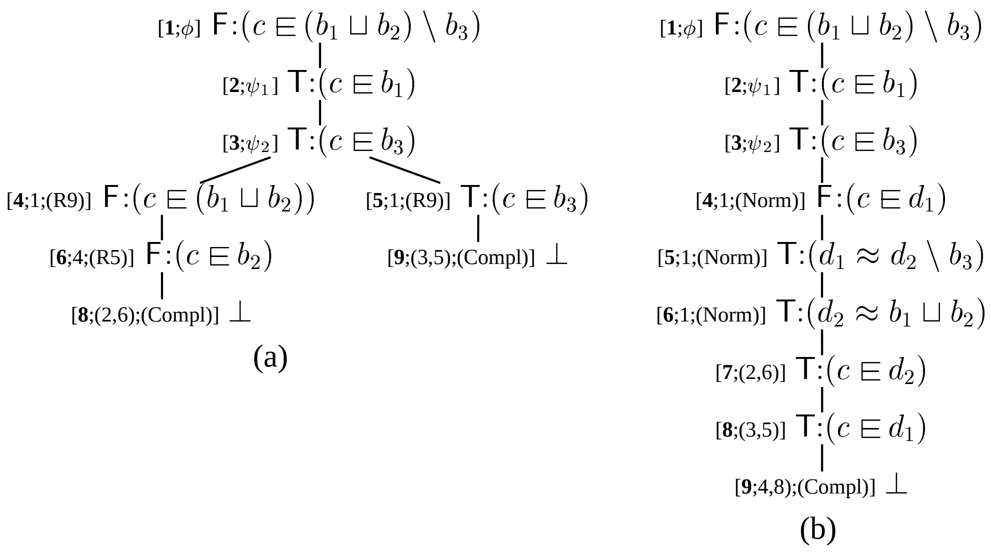

[1;ϕ] F:$(c \sqsubseteq (b_1 \sqcup b_2) \setminus b_3)$
[2;ψ_1] T:$(c \sqsubseteq b_1)$
[3;ψ_2] T:$(c \sqsubseteq b_3)$
[4;1;(R9)] F:$(c \sqsubseteq (b_1 \sqcup b_2))$
[5;1;(R9)] T:$(c \sqsubseteq b_3)$
[6;4;(R5)] F:$(c \sqsubseteq b_2)$
[9;(3,5);(Compl)] $\perp$
[8;(2,6);(Compl)] $\perp$
(a)
[1;ϕ] F:$(c \sqsubseteq (b_1 \sqcup b_2) \setminus b_3)$
[2;ψ_1] T:$(c \sqsubseteq b_1)$
[3;ψ_2] T:$(c \sqsubseteq b_3)$
[4;1;(Norm)] F:$(c \sqsubseteq d_1)$
[5;1;(Norm)] T:$(d_1 \approx d_2 \setminus b_3)$
[6;1;(Norm)] T:$(d_2 \approx b_1 \sqcup b_2)$
[7;(2,6)] T:$(c \sqsubseteq d_2)$
[8;(3,5)] T:$(c \sqsubseteq d_1)$
[9;4,8);(Compl)] $\perp$
(b)

Figure 3.3: Constructing a closed sub-tableau with (a) the calculus $\mathcal{C}_{\mathrm{MLSS}}$ and (b) Cantone's calculus.

The need (and possibility) to derive more complex terms from simpler ones leads to a larger search space. Our calculus, that splits complex terms into simpler ones, is more goal directed.

3.8.7 An Ideal Calculus for MLSSF

To extend the calculus $\mathcal{C}_{\mathrm{MLSS}}$ described in the previous sections to a calculus $\mathcal{C}_{\mathrm{MLSSF}}$ for the larger fragment MLSSF, it suffices

- to relax the restrictions on the equality rule schemata (the new schemata (EQ1') and (EQ2') are shown in Table 3.12 on the left), and
- to add a cut rule schema that uses equalities as cut formulae (the schema (Cut') in Table 3.12).

The new rule schemata only need to be applied to functional set terms. Non-functional terms, even if they are not pure, can be handled by the expansion rule schemata of $\mathcal{C}_{\mathrm{MLSS}}$. Below, the expansion rule $\mathcal{E}_{\mathrm{MLSSF}}$ of $\mathcal{C}_{\mathrm{MLSSF}}$ is formally defined.

Definition 3.8.15 Let Σ be an MLSSF signature.

For all premisses $\Pi \subset \mathit{TabForm}_{\mathrm{MLSSF}}(\Sigma^*)$, the set $\mathcal{E}_{\mathrm{MLSSF}}(\Sigma^*)(\Pi)$ consists of the following conclusions:

– $\{\{\alpha_1, \ldots, \alpha_n\}\}$
  for all $\alpha \in \Pi$ (Tables 3.1 and 3.10);

$\dfrac{\begin{array}{c}\mathsf{T}{:}(s \approx t)\\ \phi[s]\end{array}}{\phi[t]}$	$\dfrac{\begin{array}{c}\mathsf{T}{:}(t \approx s)\\ \phi[s]\end{array}}{\phi[t]}$	$\dfrac{}{\mathsf{T}{:}(t_1 \approx t_2) \mid \mathsf{F}{:}(t_1 \approx t_2)}$
where the occurrence of s in ϕ is inside a functional term (EQ1')	(EQ2')	where t_1, t_2 occur on the branch and at least one is a functional term (Cut')

Table 3.12: Additional expansion rule schemata for MLSSF.

- $\{\{\beta_1\}, \ldots, \{\beta_n\}\}$
  for all $\beta \in \Pi$ (Tables 3.1 and 3.10);
- $\{\{\phi[t]\}\}$
  for all (a) $\phi[s]$ and (b) $\mathsf{T}{:}(s \approx t)$ or $\mathsf{T}{:}(t \approx s)$ in Π such that s occurs as a proper subterm of a functional term in $\phi[s]$, where $\phi[t]$ is constructed from $\phi[s]$ by replacing one occurrence of s in $\phi[s]$ by t;
- $\{\{\mathsf{F}{:}(s_1 \approx s_2)\}\}$
  for all $\mathsf{T}{:}(s_1 \sqsubseteq t)$ and $\mathsf{F}{:}(s_2 \sqsubseteq t)$ in Π;
- $\{\{\mathsf{T}{:}(s \sqsubseteq t)\}, \{\mathsf{F}{:}(s \sqsubseteq t)\}\}$
  for all set terms s and t such that (a) s or $\{\ldots, s, \ldots\}_n$ and (b) t or $\{\ldots, t, \ldots\}_n$ occur as top-level terms in atoms in Π;
- $\{\{\mathsf{T}{:}(t_1 \approx t_2)\}, \{\mathsf{F}{:}(t_1 \approx t_2)\}\}$
  for all set terms t_1 and t_2 occurring in Π such that at least one of t_1 and t_2 is a functional set term;
- $\{\{\mathsf{T}{:}(c \sqsubseteq t_1), \mathsf{F}{:}(c \sqsubseteq t_2)\}, \{\mathsf{F}{:}(c \sqsubseteq t_1), \mathsf{T}{:}(c \sqsubseteq t_2)\}\}$
  for all $\mathsf{F}{:}(t_1 \approx t_2)$ in Π, where $c = sko_{\mathrm{MLSSF}}(\mathsf{F}{:}(t_1 \approx t_2))$
- $\{\{\bot\}\}$
  if

  1. $\mathsf{T}{:}\sigma{:}G$, $\mathsf{F}{:}\sigma{:}G \in \Pi$ for any atom G,
  2. $\mathsf{T}{:}(t \sqsubseteq \emptyset) \in \Pi$,
  3. $\mathsf{F}{:}(t \approx t) \in \Pi$,
  4. for some $k \geq 1$, $\mathsf{T}{:}(t_i \sqsubseteq t_{i+1}) \in \Pi$ $(1 \leq i < k)$ and $\mathsf{T}{:}(t_k \sqsubseteq t_1) \in \Pi$ □

The calculus $\mathcal{C}_{\mathrm{MLSSF}}$ for MLSSF is sound and complete (a proof can be found in (Hartmer, 1997)); it is, however, not a decision procedure.

Theorem 3.8.16 *The calculus $\mathcal{C}_{\mathrm{MLSSF}}$ for* MLSSF *is sound and complete.*

3.8.8 Using Rigid E-Unification to Restrict the Cut Rule Schema

The additional rule schemata of $\mathcal{C}_{\mathrm{MLSSF}}$ introduced in the previous section are highly non-deterministic. In this section, we describe an expansion rule schema for MLSSF that is much more goal-directed and leads to a smaller search space. It is based on the concept of *rigid E-unification*.

Definition 3.8.17 Let Σ be an MLSSF-signature.

A *rigid equality* is a rigid variable formula over Σ^*_{fv} of the form $t \approx t'$.

A *rigid E-unification problem* $\langle E, s, t\rangle$ consists of a finite set E of rigid equalities and terms s and t over Σ^*_{fv}.

A substitution $\sigma \in \mathit{Subst}(\Sigma^*_{\mathrm{fv}})$ is a *solution* to the problem $\langle E, s, t\rangle$ iff $s\sigma$ and $t\sigma$ are identical in the free algebra defined by $E\sigma$ where the free variables in $E\sigma$ are treated as constants. □

The problem of deciding whether a given rigid E-unification problem has a solution is decidable (it is NP-complete). In general, the number of solutions is infinite. An overview of methods for rigid E-unification can be found in (Beckert, 1998b).

The basic idea is to use rigid E-unification for handling the functional part of MLSSF and to use the MLSS expansion rule for handling the non-functional (i.e. set theoretic) part. The additional expansion rule schema, we describe in the following, forms the connecting link between the two parts.

Consider, for example, a branch B containing the two atoms

$$\mathsf{T}{:}(f(a) \approx b) \text{ and } \mathsf{T}{:}(g(f(a \sqcap (b \sqcup a))) \sqsubseteq g(b)) \ .$$

The branch B is unsatisfiable, because $a \cap (b \cup a) = a$ and, thus,

$$g(f(a \cap (b \cup a))) = g(f(a)) = g(b) \ ;$$

that implies $g(b) \in g(b)$, which is a membership cycle. To extend the branch B by a closed sub-tableau using the expansion rule of $\mathcal{C}_{\mathrm{MLSSF}}$, one first has to discover the important set theoretic identities that have to be proven (an identity is proven by using it as a cut formula; after the branch that contains its negation has been closed, it is available on the remaining open branch). In the above example, the identity that has to be proven is $a \cap (b \cup a) = a$. It is not easy to discover the right identities; it is futile, for example, to try to show that $a \cap (b \cup a) = b$.

The question of which set theoretic identities have to be proven to close a branch B is transformed into rigid E-unification problems as follows: for each pair s, t of terms that, if they were identical would allow to close the branch (e.g., terms s, t such that $\mathsf{F}{:}(s \approx t)$ is on B), one rigid E-unification problem is generated. In s and t all maximal non-functional sub-terms are replaced by rigid variables; the resulting terms s^{rv}

and t^{rv} and the equalities on the branch form a rigid E-unification problem. Each solution to the problem corresponds to identities between non-functional sub-terms that, when proven, allow to close the branch. The corresponding inequalities are used as a conclusion, i.e., they are (disjunctively connected) added to the branch.

Definition 3.8.18 Given a set Λ of atomic MLSSF-tableau formulae, the set Λ^{rv} is constructed from Λ by replacing all non-functional (sub-)terms t in Λ by a rigid variable X_t.

Let the substitution τ_Λ be defined by: $\tau(X_t) = t$ for all terms t occurring in Λ that have been replaced (i.e., τ_Λ is the inverse of the transformation that turns Λ into Λ^{rv}: $\Lambda^{\mathrm{rv}}\tau = \Lambda$). □

Example 3.8.19 If $\Lambda = \{\mathsf{T}{:}((a \sqcap c) \sqcup b \approx c),\ \mathsf{T}{:}[f(c) \sqsubseteq g(a \sqcap c, f(d \setminus e))]\}$, then the result of the transformation is $\Lambda^{\mathrm{rv}} = \{\mathsf{T}{:}(X_1 \approx X_2),\ \mathsf{T}{:}[f(X_2) \sqsubseteq g(X_3, f(X_4))]\}$. □

Definition 3.8.20 Let Σ be an MLSSF signature; let $\Pi \subset \mathit{TabForm}_{\mathrm{MLSSF}}(\Sigma^*)$ be a premiss; let Λ_Π be the set of all atoms in Π of the form $\mathsf{T}{:}(t_1 \approx t_2)$, $\mathsf{T}{:}(t_1 \sqsubseteq t_2)$, or $\mathsf{F}{:}(t_1 \sqsubseteq t_2)$; and let E^{rv}_Π be the set of all rigid variable equalities in $\Lambda^{\mathrm{rv}}_\Pi$.

Further let $\mu = \{x_1 \leftarrow r_1, \ldots, x_n \leftarrow r_n\}$ ($n \geq 1$) be a simultaneous solution[5] to

1. rigid E-unification problems $\langle E^{\mathrm{rv}}_\Pi, s_1, t_1\rangle$ and $\langle E^{\mathrm{rv}}_\Pi, s_2, t_2\rangle$ such that $\mathsf{T}{:}(s_1 \sqsubseteq t_1)$ and $\mathsf{F}{:}(s_2 \sqsubseteq t_2)$ are in $\Lambda^{\mathrm{rv}}_\Pi$, or
2. rigid E-unification problems $\langle E^{\mathrm{rv}}_\Pi, t_1, t'_1\rangle$, …, $\langle E^{\mathrm{rv}}_\Pi, t_n, t'_n\rangle$ ($n \geq 1$) such that atoms $\mathsf{T}{:}(t_1 \sqsubseteq t'_2, \ldots, t_{n-1} \sqsubseteq t'_n)$, and $\mathsf{T}{:}(t_n \sqsubseteq t'_1)$ in $\Lambda^{\mathrm{rv}}_\Pi$ form a potential membership cycle.

Then, the conclusion

$$\{\{\mathsf{F}{:}(\tau_{\Lambda_\Pi}(X_1) \approx r_1\tau_{\Lambda_B})\},\ \ldots, \{\mathsf{F}{:}(\tau_{\Lambda_B}(X_n) \approx r_n\tau_{\Lambda_B})\}\ .$$

is an *EU-conclusion* of Π. □

Using the notion of EU-conclusions, the expansion rule $\mathcal{E}^{\mathrm{EU}}_{\mathrm{MLSSF}}$ of $\mathcal{C}^{\mathrm{EU}}_{\mathrm{MLSSF}}$ is defined as follows.

[5] The substitution μ has to be the simultaneous solution of several rigid E-unification problems, i.e., a simultaneous rigid E-unification problem has to be solved. In general, *simultaneous* rigid E-unification is undecidable. But the simultaneous problems that have to be solved here belong to a decidable subclass, namely the class of simultaneous rigid E-unification problems of the form $\{\langle E, s_1, t_1\rangle, \ldots, \langle E, s_n, t_n\rangle\}$ where the sets of equalities are identical. In that case, any substitution that (a) is a solution to the non-simultaneous problem $\langle E, f(s_1, \ldots, s_n), f(t_1, \ldots, t_n)\rangle$ and (b) does not instantiate variables with terms containing f is a solution to the original problem (the function symbol f must not occur in the original problem).

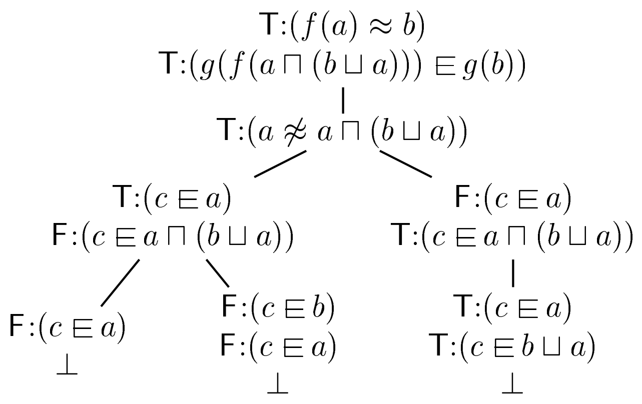

$\mathsf{T}{:}(f(a) \approx b)$
$\mathsf{T}{:}(g(f(a \sqcap (b \sqcup a))) \sqsubseteq g(b))$
$\mathsf{T}{:}(a \not\approx a \sqcap (b \sqcup a))$
$\mathsf{T}{:}(c \sqsubseteq a)$
$\mathsf{F}{:}(c \sqsubseteq a \sqcap (b \sqcup a))$
$\mathsf{F}{:}(c \sqsubseteq a)$
$\mathsf{T}{:}(c \sqsubseteq a \sqcap (b \sqcup a))$
$\mathsf{F}{:}(c \sqsubseteq a)$
$\bot$
$\mathsf{F}{:}(c \sqsubseteq b)$
$\mathsf{F}{:}(c \sqsubseteq a)$
$\bot$
$\mathsf{T}{:}(c \sqsubseteq a)$
$\mathsf{T}{:}(c \sqsubseteq b \sqcup a)$
$\bot$

Figure 3.4: A closed sub-tableau that is constructed using the expansion rule of $\mathcal{C}^{\mathrm{EU}}_{\mathrm{MLSSF}}$ (Example 3.8.22).

Definition 3.8.21 Let Σ be an MLSSF signature.

For all premisses $\Pi \subset \mathit{TabForm}_{\mathrm{MLSSF}}(\Sigma^*)$, the set $\mathcal{E}^{\mathrm{EU}}_{\mathrm{MLSSF}}(\Sigma^*)(\Pi)$ consists of the conclusions in $\mathcal{E}_{\mathrm{MLSS}}(\Sigma^*)(\Pi)$ (Def. 3.8.3) and, in addition, all EU-conclusions of Π. $\square$

Example 3.8.22 We continue the example from the beginning of this section and apply the expansion rule the new calculus of $\mathcal{C}^{\mathrm{EU}}_{\mathrm{MLSSF}}$ to construct a closed sub-tableau below a branch containing the atoms

$$\mathsf{T}{:}(f(a) \approx b) \text{ and } \mathsf{T}{:}(g(f(a \sqcap (b \sqcup a))) \sqsubseteq g(b)) \ .$$

The only rigid E-unification problem that can be extracted from these atoms is

$$\langle \{f(X_a) \approx X_b\}, g(f(X_{a\sqcap(b\sqcup a)})), g(X_b)\rangle \ .$$

A most general solution of this problem is the substitution $\{X_a \mapsto X_{a\sqcap(b\sqcup a)}\}$. Thus, the new rule schema of $\mathcal{C}^{\mathrm{EU}}_{\mathrm{MLSSF}}$ allows to add $\mathsf{F}{:}(a \approx a \sqcap (b \sqcup a))$ to the branch; the branch can then be extended to a closed sub-tableau (see Figure 3.4). $\square$

The new expansion rule schema partly overlaps with other schemata. It allows, for example, to derive $\mathsf{F}{:}(s_1 \approx s_2)$ from $\mathsf{T}{:}(s_1 \sqsubseteq t)$ and $\mathsf{F}{:}(s_2 \sqsubseteq t)$ if s_1 and s_2 are non-functional set terms. This is also possible applying the schema (R11).

It is not necessary to consider rigid E-unification problems that can be constructed from inequalities $\mathsf{F}{:}(s \approx t)$ in L^{rv}_{Π} because, when rule schema (R11) has been applied, the branch contains atoms $\mathsf{T}{:}(c \sqsubseteq s)$, $\mathsf{F}{:}(c \sqsubseteq t)$, or $\mathsf{F}{:}(c \sqsubseteq s)$, $\mathsf{T}{:}(c \sqsubseteq t)$.

The expansion rule of $\mathcal{C}^{\mathrm{EU}}_{\mathrm{MLSSF}}$ is sound, and the new schema helps to reduce the search space. It is a conjecture that $\mathcal{C}^{\mathrm{EU}}_{\mathrm{MLSSF}}$ is complete, i.e., that the schema based on rigid E-unification can replace the schemata (EQ1'), (EQ2'), and (Cut') of $\mathcal{C}_{\mathrm{MLSSF}}$, but this has not been proven this yet.

Theorem 3.8.23 *The calculus* $\mathcal{C}^{\mathrm{EU}}_{\mathrm{MLSSF}}$ *for* MLSSF *is sound.*

Conjecture 3.8.24 *The calculus* $\mathcal{C}^{\mathrm{EU}}_{\mathrm{MLSSF}}$ *for* MLSSF *is complete.*

Example 3.8.25 We proof that the formula

$$G \;=\; [a \sqsubseteq [(f(a) \setminus f(a \sqcup (b \sqcap a))) \sqcup d \sqcup e] \;\wedge\; e \sqcup b \sqsubseteq a] \;\rightarrow\; a \sqsubseteq d$$

is an MLSSF tautology; it contains the free function symbol f. Intuitively, the reason for the validity of G is the following: We assume that a is an element of (at least) one of the three sets $u = f(a) \setminus f(a \cup (b \cap a))$, d, and e, and that $e \cup b$ is an element of a. Now, the set u cannot contain a, because $a = (a \cup (b \cap a))$ and therefore u is empty for all interpretations of f; the set e cannot contain a, otherwise there would be a membership cycle $a \in (e \cup b) \in a$. Therefore, d contains a.

Figure 3.5 shows a closed tableau for $\neg G$.[6] The unsatisfiability of $\neg G$ implies the validity of G.

Formula (11) is derived from formulae (9) and (10) applying the E-unification rule schema. The substitution $\{x_a \leftarrow x_{a\sqcup(b\sqcap a)}\}$ is a solution to the rigid E-unification problems $\langle\emptyset, X_a, X_a\rangle$ and $\langle\emptyset, \langle f(X_a), f(X_{a\sqcup(b\sqcap a)})\rangle\rangle$, which is constructed from 9 and 10. Accordingly, the inequality $\mathsf{F}{:}(a \approx a \sqcup (b \sqcap a))$ is added to the branch.

When the rule schema (R12) is applied to formula (11) to derive formulae (12)–(15), the Skolem constant $c_1 = sko_{\mathrm{MLSSF}}(\mathsf{F}{:}(a \approx a \sqcup (b \sqcap a)))$ is introduced.

The branch ending in node (30) is closed by the membership cycle $e \sqcup b \sqsubseteq a$ and $a \sqsubseteq e \sqcup b$ (formulae (6) and (28)). All other branches are closed by pairs of complementary atoms.

If the rule schema for closing branches is used that relies on computing sets of implicit predecessors to detect implicit membership cycles (Def. 3.8.10), then the cut rule application that generates formulae (28) and (29) is not needed. Instead, the branch ending in node (26) can be closed immediately; it contains an implicit cycle because $a \in e$ implies $a \in e \cup b$ (this cycle is made explicit by the cut application). The set of all possible implicit predecessors for terms on the branch ending in node (26) is $\{a, b, d, e, f(a), f(a \sqcup (b \sqcap a)), (e \sqcup b)\}$. The predecessor sets of the constants are

$$\begin{aligned}
\mathcal{P}_\Pi(a) &= \{e \sqcup b\}\\
\mathcal{P}_\Pi(b) &= \emptyset\\
\mathcal{P}_\Pi(d) &= \emptyset\\
\mathcal{P}_\Pi(e) &= \{a\}
\end{aligned}$$

[6] The i-th node in the tableau is labelled with $[i; j; R]$, which indicates that its formula has been derived applying the expansion rule schema R of $\mathcal{C}^{\mathrm{EU}}_{\mathrm{MLSSF}}$ to a premiss consisting of the formula in the j-th node.

The predecessor set of $e \sqcup b$ is

$$\mathcal{P}_{\Pi}(e \sqcup b) = \mathcal{P}_{\Pi}(e) \cup \mathcal{P}_{\Pi}(b) = \{a\} \ .$$

Thus, we have $a \in \mathcal{P}_{\Pi}(e \sqcup b)$ and $e \sqcup b \in \mathcal{P}_{\Pi}(a)$, which indicates the presence of an implicit membership cycle. $\square$

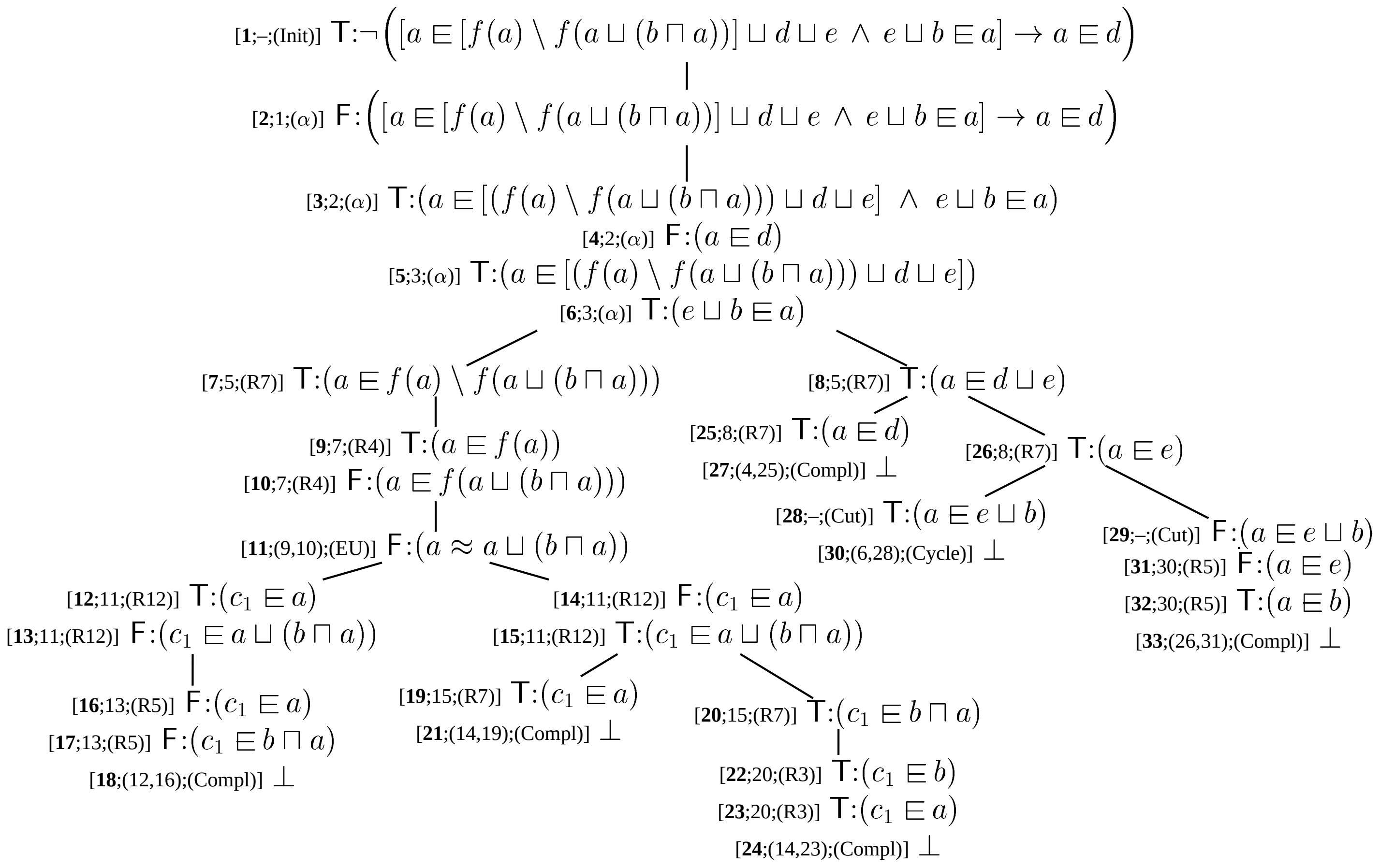

[**1**;–;(Init)] $\mathsf{T}{:}\neg\Big([a \sqsubseteq [f(a) \setminus f(a \sqcup (b \sqcap a))] \sqcup d \sqcup e \ \wedge\ e \sqcup b \sqsubseteq a] \to a \sqsubseteq d\Big)$
[**2**;1;(α)] $\mathsf{F}{:}\Big([a \sqsubseteq [f(a) \setminus f(a \sqcup (b \sqcap a))] \sqcup d \sqcup e \ \wedge\ e \sqcup b \sqsubseteq a] \to a \sqsubseteq d\Big)$
[**3**;2;(α)] $\mathsf{T}{:}(a \sqsubseteq [(f(a) \setminus f(a \sqcup (b \sqcap a))) \sqcup d \sqcup e] \ \wedge\ e \sqcup b \sqsubseteq a)$
[**4**;2;(α)] $\mathsf{F}{:}(a \sqsubseteq d)$
[**5**;3;(α)] $\mathsf{T}{:}(a \sqsubseteq [(f(a) \setminus f(a \sqcup (b \sqcap a))) \sqcup d \sqcup e])$
[**6**;3;(α)] $\mathsf{T}{:}(e \sqcup b \sqsubseteq a)$
[**7**;5;(R7)] $\mathsf{T}{:}(a \sqsubseteq f(a) \setminus f(a \sqcup (b \sqcap a)))$
[**8**;5;(R7)] $\mathsf{T}{:}(a \sqsubseteq d \sqcup e)$
[**9**;7;(R4)] $\mathsf{T}{:}(a \sqsubseteq f(a))$
[**10**;7;(R4)] $\mathsf{F}{:}(a \sqsubseteq f(a \sqcup (b \sqcap a)))$
[**11**;(9,10);(EU)] $\mathsf{F}{:}(a \approx a \sqcup (b \sqcap a))$
[**12**;11;(R12)] $\mathsf{T}{:}(c_1 \sqsubseteq a)$
[**13**;11;(R12)] $\mathsf{F}{:}(c_1 \sqsubseteq a \sqcup (b \sqcap a))$
[**16**;13;(R5)] $\mathsf{F}{:}(c_1 \sqsubseteq a)$
[**17**;13;(R5)] $\mathsf{F}{:}(c_1 \sqsubseteq b \sqcap a)$
[**18**;(12,16);(Compl)] $\bot$
[**14**;11;(R12)] $\mathsf{F}{:}(c_1 \sqsubseteq a)$
[**15**;11;(R12)] $\mathsf{T}{:}(c_1 \sqsubseteq a \sqcup (b \sqcap a))$
[**19**;15;(R7)] $\mathsf{T}{:}(c_1 \sqsubseteq a)$
[**21**;(14,19);(Compl)] $\bot$
[**20**;15;(R7)] $\mathsf{T}{:}(c_1 \sqsubseteq b \sqcap a)$
[**22**;20;(R3)] $\mathsf{T}{:}(c_1 \sqsubseteq b)$
[**23**;20;(R3)] $\mathsf{T}{:}(c_1 \sqsubseteq a)$
[**24**;(14,23);(Compl)] $\bot$
[**25**;8;(R7)] $\mathsf{T}{:}(a \sqsubseteq d)$
[**27**;(4,25);(Compl)] $\bot$
[**26**;8;(R7)] $\mathsf{T}{:}(a \sqsubseteq e)$
[**28**;–;(Cut)] $\mathsf{T}{:}(a \sqsubseteq e \sqcup b)$
[**30**;(6,28);(Cycle)] $\bot$
[**29**;–;(Cut)] $\mathsf{F}{:}(a \sqsubseteq e \sqcup b)$
[**31**;30;(R5)] $\mathsf{F}{:}(a \sqsubseteq e)$
[**32**;30;(R5)] $\mathsf{T}{:}(a \sqsubseteq b)$
[**33**;(26,31);(Compl)] $\bot$

Figure 3.5: Tableau proof for an MLSSF formula (Example 3.8.25).

4 Enhancements

4.1 Overview

In this chapter, we present techniques for improving a tableau calculus for automated deduction in such a way that proof procedure based on it are more efficient. Only those techniques are the subject of this chapter that require the calculus to be changed; heuristics and techniques for organising the proof search, i.e., for constructing an efficient proof procedure based on a given calculus are discussed in the next chapter.

Strengthening the calculus A calculus is strengthened if it is changed in such a way that shorter proofs for at least some formulae exist (for example, by enhancing its expansion rule such that it allows the deduction of additional conclusions from certain premisses). In most cases, strengthening adds non-determinism, i.e., there are more possibilities to proceed at each expansion step. Thus, there is a trade-off between the advantage of shorter proofs and the disadvantage that these short proofs may be harder to find, because there are more choice points in the search space.

Unfortunately, there is no general rule for judging whether a certain enhancement is useful for automated deduction. Some automated deduction systems even use restricted calculi that have less choice points than the standard version of the calculus at the expense of an increase in the size of smallest proofs (including, for example, proof procedures for first-order clause logic with the strong connectedness condition, see Section 5.5).

The non-analytic cut rule, which allows to deduce the conclusion $\{\{\sigma{:}\mathsf{T}{:}\phi\}, \{\sigma{:}\mathsf{F}{:}\phi\}\}$ from the empty premiss for all labels σ and formulae ϕ, is a typical example for an enhancement where the disadvantage of additional non-determinism outweighs the advantage of the existence of shorter proofs. Therefore, in automated deduction systems the cut rule is *never* used unrestrictedly. There are, however, restricted version of the cut rule, such as the generation of local lemmata (Section 4.5), that still reduce the size of the smallest proofs for certain formula classes exponentially, but do not lead to too many additional choice points.

A method for strengthening tableau calculi that is always useful is the *pruning* technique described in Section 4.6, as it reduces the size of proofs without introducing additional choice points.

Another very useful way of strengthening tableau calculi is the introduction of *universal variables* (Section 4.3); this technique can lead to exponentially smaller proofs, and adds only very few additional choice points.

Post-poning choice points Often, it is possible to uniformly represent different tableaux—and thus different parts of the search space—by a single tableau using additional syntactical devices. A typical example is the *rigid variable* technique (Section 4.2), where tableaux that are identical up to the replacement of terms by other terms are all represented by a single tableau in which a free (or dummy) variable is used as a place holder for the different terms.

In a certain sense, post-poning choice points leads to a breadth-first search where different parts of the search space are investigated simultaneously, until additional information has been gathered that allows to make an informed decision about which part of the search space should be further investigated. For example, if a tableau rule application requires a rigid variable X to be instantiated with a certain term t, then the decision to instantiate X with t is informed in the sense that the instantiation is known to be useful as an expansion rule application exists which is not possible without it.

Techniques that post-pone choice points are always of advantage if the only measure considered is the number of expansions steps that have to be executed to find a tableau proof. These techniques can, however, be difficult to implement. For example, if a rigid variable is instantiated with a term that later turns out to be the wrong choice, then it might become much more difficult to find a tableau proof (though not impossible provided that the calculus is proof confluent); thus, it is of advantage not to instantiate rigid variables as long as it is not absolutely certain that a certain instantiation is useful. An instantiation that may or may not be useful should instead be used as a heuristic, i.e., expansion rule applications that are compatible with such an instantiation are preferred but are not necessarily the only ones that are considered. However, because this is difficult to implement, virtually all tableau-based deduction systems for first-order predicate logic apply a rigid variable substitution to the whole tableau as soon as it is found to allow the closure of a single branch—instead of first searching for closing substitutions for all branches, and then trying to combine these substitutions to construct a single substitution that allows to simultaneously close all branches.

Another disadvantage of techniques for post-poning choice points is that they make it more difficult for the deduction system to interact with a human user (where the reason for interaction may either be that the system is not fully automated or that a proof that has been constructed is communicated to the user). Humans prefer calculi with a simple tableau expansion rule; its applications should be syntactically (not necessarily semantically) simple, i.e., each application should be easy to validate, the pre-conditions should be easy to check, and rule applications should not have any side effects. Enhancements for post-poning choice points, however, typically result in a calculus that is syntactically more complicated; rule applications may have non-local side effects such as the instantiation of free variables (the problem of user interaction is discussed for the case of calculi for first-order predicate logic in (**?**)).

4.2 Rigid Variable Tableau Calculi

4.2.1 The Idea of Rigid Variables

The concept of rigid variables[1], which is well known from theorem proving in first-order predicate logic, is based on the observation that the expansion rule of many calculi for logics with terms is stable for certain premisses Π w.r.t. the replacement of terms in Π by other terms; in particular, there are premisses that allow to derive conclusions $C(t)$ for all terms t.

Example 4.2.1 A typical example is the expansion rule of the calculus $\mathcal{C}_{\mathrm{PL1}}$ from Section 3.6. It is stable for premisses consisting of α-, β-, γ-, and δ-formulae. For example, the conclusion $\{\{\alpha_1[t \mapsto t'], \alpha_2[t \mapsto t']\}\}$ can, for all terms t', be derived from a premiss containing an α-formula $\alpha[t \mapsto t']$. Premisses containing a γ-formula γ allow to derive the conlcusion $C(t) = \{\{\gamma_1(t)\}\}$ for all terms t.

The expansion rule of $\mathcal{C}_{\mathrm{PL1}}$ is only unstable for premisses that allow to deduce $\perp$, i.e., to close a branch; for example, the premiss $\{\mathsf{T}{:}\phi(s),\ \mathsf{F}{:}\phi(t)\}[t \mapsto t']$ allows the deduction of $\perp = \perp[t \mapsto t']$ only in case $t' = s$. $\square$

If a calculus has an expansion rule that is stable for certain premisses, then instead of guessing terms when they are introduced, they can be represented by a rigid variable that is later instantiated "on demand" when the expansion rule is applied to a premiss w.r.t. which it is not stable, i.e., in case the application requires the instantiation of the rigid variable. Usually unification is used to find a *most general* substitution that allows a certain expansion rule application. Using rigid variables reduces the number of choice points in the construction of a tableau proof and thus the size of the search space.

Intuitively, a formula containing a rigid variable X stands for a single (but unknown) element of the set of all formulae that are the result of replacing X by some term (cf. Section 4.3 where rigid variables are introduced that represent *all* term and that are, thus, called *universal* variables). All occurrences of a rigid variable in a tableau have to be instantiated by the same term (which is why these variables are called "rigid"). Thus, instantiating a rigid variable to make a non-stable expansion rule application

[1] In the literature—in particular on calculi for first-order predicate logic—, several other names have been used for free and, in particular, for rigid variables, including *parameter*, *dummy variable*, and *meta variable*.

possible can make other expansions impossible, which is particularly problematic if the calculus is not proof confluent.

Rigid variable tableau calculi are usually constructed by "lifting" a ground tableau calculus. Unfortunately, this is often done in an *ad hoc* manner. Starting point is the observation that the expansion rule of a calculus is stable under replacement of terms by terms for some or most premisses. Then, as a first step, the expansion rule is lifted for those premisses that allow the deduction of a conclusion $C(t)$ for all terms t; then, in the rigid variable version, the conclusion $C(X)$ can be deduced from such premisses where X is an arbitrary rigid variable. In a second step, the rigid variable expansion rule for other premisses is designed in such a way that the calculus is sound and complete; that, however, can be difficult and there may be hidden traps—in particular if the ground calculus is not ideal. One has to carefully check for which premisses the ground expansion rule is stable and for which it is not.

Example 4.2.2 The fact that lifting a ground calculus can be difficult is exemplified by the history of rigid variable tableau calculi for first-order predicate logic: When the first versions were defined, some authors missed to note the fact that the ground rule they used was not stable for δ-formulae, as it demanded the Skolem constant introduced by an application to a δ-formula to be *new*. Other authors solved (resp. avoided) this problem by designing an expansion rule schema for δ-formulae that does not result from lifting the ground version (see Section 4.4 for a discussion of improving the expansion rule schema for δ-formulae). $\square$

Such hidden traps can be avoided by altering the ground calculus and making it as stable as possible before it is lifted. The design problem is then to make the rule of the ground calculus stable; an example is the ground expansion rule from Section 3.6, which might be somewhat non-intuitive but is stable for δ-formulae.

If a rigid variable calculus $\mathcal{C}^{\text{fv}}$ has been constructed by lifting a stable ground calculus $\mathcal{C}^{\text{gd}}$, then soundness and completeness of $\mathcal{C}^{\text{fv}}$ follow from soundness and completeness of $\mathcal{C}^{\text{gd}}$ and do not have to be proven separately. In particular, as the relation between the rigid variable and ground calculus is purely syntactical, one does not have to come up separately with an appropriate semantics for the rigid variable tableaux. The semantics of a tableau containing rigid variables can be defined based on the semantics of its ground instances (Section 4.2.9).

4.2.2 Syntax of Free Variable Tableau Calculi

The replacement of terms by free variables is not restricted to the formula part G of a tableau formula $\mathsf{S}{:}\sigma{:}G$, but they can as well be introduced into the label σ and the truth value sign S. That is, we assume that the set of tableau formulae is a language with terms. Labels and formulae may or may not contain free variables of the same sort (and, thus, can have variables in common). Free variables used as truth value signs have to be of a special sort s, such that the only terms of sort s are T and F.

Definition 4.2.3 A calculus $\mathcal{C}^{\mathrm{fv}}$ for a logic $\mathbf{L}$ is a *free variable calculus* (w.r.t. the set Var of free variables) if, for each signature $\Sigma \in \mathit{Sig}$, the extension Σ^*_{fv} of Σ that is used to build tableau formulae is an extension of a signature $\Sigma^*_{\mathrm{gd}} \in \mathit{Sig}$ (that is an extension of Σ, too) such that:

1. the set $\mathit{TabForm}(\Sigma^*_{\mathrm{gd}})$ of tableau formulae over Σ_{gd} is a language with terms (Def. 2.2.1);
2. the set $\mathit{TabForm}(\Sigma^*_{\mathrm{fv}})$ of tableau formulae over Σ_{fv} is the free variable language constructed from $\mathit{TabForm}(\Sigma^*_{\mathrm{gd}})$ and the set Var of free variables (according to Definition 2.2.2), i.e.,

$$\mathit{TabForm}(\Sigma^*_{\mathrm{fv}}) = (\mathit{TabForm}(\Sigma^*_{\mathrm{gd}}))^{\mathrm{fv}} \enspace .$$

The sets of terms of $\mathit{TabForm}(\Sigma^*_{\mathrm{fv}})$ and $\mathit{TabForm}(\Sigma^*_{\mathrm{gd}})$ are denoted by $\mathit{TabTerm}(\Sigma^*_{\mathrm{fv}})$ resp. $\mathit{TabTerm}(\Sigma^*_{\mathrm{gd}})$.

A calculus $\mathcal{C}^{\mathrm{gd}}$ for a logic $\mathbf{L}$ is a *ground calculus* (w.r.t. the set Var of free variables) if its tableau formulae do not contain any of the free variables in Var. □

Example 4.2.4 One of the rare examples where free variables in truth value signs are useful, is the design of an expansion rule that allows to deduce the non-branching conclusion $\{\{X{:}F,\ X{:}G\}\}$ from premiss containing the equivalency $\{\mathsf{T}{:}(F \leftrightarrow G)\}$. □

The class of free variable tableau calculi is only characterised by the fact that they use tableau formulae containing free variables. All definitions from Chapter 3 of notions such as branch, tableau, tableau proof, proof confluence, monotonicity, etc. remain unchanged. The notions of calculi with expansion rule and of idealness as defined in Chapter 3, however, are not appropriate and for the rigid variable case and have to be adapted (see Sections 4.2.3 and 4.2.5).

4.2.3 Rigid Variable Tableau Calculi with Expansion Rule

The notions of a calculus with expansion rule (Section 3.3.3)—and, thus, that of an ideal calculus (Section 3.3.7)—are not appropriate for rigid variable calculi, because in a calculus with expansion rule all tableau formulae on a tableau T must remain unchanged when it is expanded; it is not allowed to instantiate rigid variables occurring in T. Therefore, we introduce the notion of *rigid variable expansion rules*, where a conclusion C contains (besides a finite set of extensions) a substitution that is applied to the whole tableau when C is used for expansion. Apart from the application of that substitution, tableau rule applications are still only allowed to have only *local* effects. That is, a tableau rule application extends only a *single* branch of a tableau, and, what the possibilities for extending a branch B are, only depends on B itself; no additional pre-conditions are allowed such as the presence of certain formulae on other branches.

Definition 4.2.5 Let $\mathcal{C}$ be a free variable tableau calculus for a logic **L**; and let $\Sigma \in \mathit{Sig}$ be a signature.

A *rigid variable conclusion* is a pair $\langle C, \sigma \rangle$ consisting of a finite set C of branch extensions (Def. 3.3.4) and a substitution $\sigma \in \mathit{Subst}(\Sigma^*_{\mathrm{fv}})$ such that $C = C\sigma$.

A *rigid variable expansion rule $\mathcal{E}(\Sigma)$* is a function that assigns to each (finite) tableau branch whose nodes are labelled with formulae from $\mathit{TabForm}(\Sigma^*_{\mathrm{fv}})$ a set $\mathcal{E}(\Sigma)(B)$ of (possible) rigid variable conclusions, which may be infinite but has to be enumerable. □

Definition 4.2.6 Let $\mathcal{C}$ be a free variable tableau calculus for a logic **L**; and let $\Sigma \in \mathit{Sig}$ be a signature.

A rigid variable expansion rule $\mathcal{E}(\Sigma)$ *characterises* the tableau rule $\mathcal{R}(\Sigma)$ of $\mathcal{C}$ if, for all tableaux T over Σ^*_{fv}: a tableau T' is a successor tableau of T (i.e., $T' \in \mathcal{R}(\Sigma)(T)$) if and only if there is a branch B of T and a rigid variable conclusion $\langle C, \sigma \rangle$ in $\mathcal{E}(\Sigma)(B)$ such that the tableau T' can be constructed from T by

1. extending the branch B by a new sub-branch for each extension E in C where the nodes in that sub-branch are labelled with the elements of E, and
2. applying the substitution σ to the tableau.

If the rigid variable expansion rule $\mathcal{E}(\Sigma)$ characterises the tableau rule $\mathcal{R}(\Sigma)$ of $\mathcal{C}$ for all signatures Σ, then $\mathcal{E}$ is said to be *the* rigid variable expansion rule of $\mathcal{C}$; and $\mathcal{C}$ is said to be a *rigid variable calculus with expansion rule $\mathcal{E}$*. □

4.2.4 Rigid Variable Expansion Rules that are Monotonic w.r.t. Substitution

The intuition behind the substitution σ that is part of a rigid variable conclusion $\langle C, \sigma \rangle$ is that its application is a pre-condition for the derivation of C from the formulae on the branch. If a (ground) calculus is monotonic as defined in Section 3.3.5, the pre-condition for deriving a certain conclusion is the presence of certain formulae on the branch that is expanded—and not the absence of formulae. Accordingly a notion of monotonicity w.r.t. the application of substitutions can be defined: Only that a certain variable *is* resp. *can be* instantiated in a certain way is allowed as a pre-condition—and not that the variable is *not* instantiated that way. Note, however, that a different, incompatible instantiation can very well prevent the derivation.

Definition 4.2.7 Let $\mathcal{C}$ be a rigid variable tableau calculus with expansion rule $\mathcal{E}$ for a logic $\mathbf{L}$.

The expansion rule $\mathcal{E}$ and the calculus $\mathcal{C}$ are called *monotonic w.r.t. substitution* if for all signatures $\Sigma \in \mathit{Sig}$ and all branches B over Σ^* the following holds: if $\langle C, \sigma \rangle \in \mathcal{E}(B)$ and $\rho, \tau \in \mathit{Subst}(\Sigma^*)$ are substitutions such that $\sigma = \rho \circ \tau$, then $\langle C, \rho \rangle \in \mathcal{E}(B\tau)$. $\square$

Example 4.2.8 Assume that—in a calculus for PL1 that is monotonic w.r.t. substitution—$\langle \{\{\mathsf{T}{:}p(a, b)\}\}, \{X \mapsto a, Y \mapsto b\} \rangle$ is a possible conclusion for a branch branch $B(X, Y)$, then $\langle \{\{\mathsf{T}{:}p(a, b)\}\}, \{Y \mapsto b\} \rangle$ is a possible conclusion for $B(a, Y)$, and $\langle \{\{\mathsf{T}{:}p(a, b)\}\}, \mathit{id} \rangle$ is a possible conclusion for $B(a, b)$. $\square$

4.2.5 Ideal Rigid Variable Tableau Calculi

Idealness, the important property of tableau making them well-behaved, is defined for rigid variable calculi in a similar same way as for. A ground calculus is ideal if it is non-structural, monotonic, and is a calculus with expansion rule. A rigid variable calculus is ideal, if it is non-structural (as in the ground case), it is montonic (as in the ground case) and moreover is monotonic w.r.t. substitution, and it has a *rigid variable* expansion rule allowing the application of substitutions (instead of a ground expansion rule).

Definition 4.2.9 A rigid variable tableau calculus with expansion rule that is (a) non-structural, (b) monotonic, and (c) monotonic w.r.t. substitution is called *ideal*. $\square$

As in the ground case (see Lemma 3.3.11), the expansion rule $\mathcal{E}$ of an ideal rigid variable calculus can be represented as a function $\tilde{\mathcal{E}}$ on premisses; and we identify the expansion rule $\mathcal{E}$ and the function $\tilde{\mathcal{E}}$.

Lemma 4.2.10 *Let $\mathcal{C}$ be an ideal rigid variable tableau calculus with expansion rule $\mathcal{E}$ for a logic $\mathbf{L}$. Then, for all signatures Σ, there is a (single) function $\tilde{\mathcal{E}}(\Sigma)$ that assigns to each (finite) premiss $\Pi \subset \mathit{TabForm}(\Sigma^*_{\mathrm{fv}})$ a set $\tilde{\mathcal{E}}(\Sigma)(\Pi)$ of (possible) rigid variable conclusions such that*

$$\mathcal{E}(\Sigma)(B) = \tilde{\mathcal{E}}(\Sigma)(\mathit{Form}(B))$$

for all tableau branches B over Σ^*_{fv}.

The notion of *minimal* premisses of a rigid variable conclusion is defined as in the ground case (Def. 3.3.13).

Expansion rules of ideal rigid variable calculi can be described by means of rule schemata with an explanation attached to them that describes how to compute the substitution that is to be applied when an instance of the schema is used to extend a tableau branch (see Sections 4.2.10 and 4.2.11 for examples).

4.2.6 A Subsumption Relation on Rigid Variable Conclusions

The subsumption (or is-more-general-than) relation $\leq^W$ defined on substitutions can be extended to rigid variable conclusions as follows:

Definition 4.2.11 Let $\mathcal{C}$ be an ideal rigid variable calculus for a logic **L**; let $\langle C, \sigma\rangle$ and $\langle C', \sigma'\rangle$ be rigid variable conclusions over the same signature $\Sigma^*_{\text{fv}} \in \mathit{Sig}$; and let $W \subset \mathit{Var}$ be a finite set of rigid variables.

The conclusion $\langle C, \sigma\rangle$ *subsumes* the conclusion $\langle C', \sigma'\rangle$, which is denoted by

$$\langle C, \sigma\rangle \leq^W \langle C', \sigma'\rangle \enspace ,$$

if there is a substitution $\rho \in \mathit{Subst}(\Sigma^*_{\text{fv}})$ such that

1. $\sigma \leq^W \rho \leq^W \sigma'$ where $\leq^W$ is the subsumption relation on substitutions from Definition 2.2.6;
2. $C\rho = C'$.

□

Lemma 4.2.12 *The subsumption relation $\leq^W$ on rigid variable conclusions is transitive.*

Proof: Assume that $\langle C, \sigma\rangle \leq^W \langle C', \sigma'\rangle$ and $\langle C', \sigma'\rangle \leq^W \langle C'', \sigma''\rangle$.

The substitutions ρ and ρ' that exist according to the definition of the relation $\leq^W$ satisfy the conditions $\sigma \leq^W \rho \leq^W \sigma'$ and $\sigma' \leq^W \rho' \leq^W \sigma''$; as the relation $\leq^W$ is transitive on substitutions, that implies $\sigma \leq^W \rho' \leq^W \sigma''$.

In addition, we have $C\rho' = (C\rho)\rho' = C'\rho' = C''$. Thus, $\langle C, \sigma\rangle \leq^W \langle C'', \sigma''\rangle$, as the substitution ρ' has the required properties. □

Lemma 4.2.13 *If $\langle C, \sigma\rangle$ and $\langle C, \sigma'\rangle$ are rigid variable conclusions that only differ in their substitutions and $\sigma \leq^W \sigma'$, then $\langle C, \sigma\rangle \leq^W \langle C, \sigma'\rangle$.*

Example 4.2.14 A conclusion with a more general substitution is more general, i.e.,

$$\langle\{\{\bot\}\}, \mathit{id}\rangle \text{ subsumes } \langle\{\{\bot\}\}, \{X \mapsto a\}\rangle \enspace ,$$

and, if $Y \notin W$, then

$$\langle\{\{\bot\}\}, \{X \mapsto f(Y)\}\rangle \text{ subsumes } \langle\{\{\bot\}\}, \{X \mapsto f(a)\}\rangle \enspace .$$

□

Intuitively, completeness of a an ideal rigid variable calculus is preserved if only most general conclusions w.r.t. $\leq^W$ are derived from a given premiss. However, the set W of variables has to be chosen carefully. If the context is known (e.g., a certain tableau), then it is sufficient if W contains all variables occurring in that context; otherwise, one has to make sure that for any finite set W of variables that may occur in the context, completeness is preserved.

Example 4.2.15 If $X \notin W$, then the conclusion $C_X = \langle\{\{p(X)\}\}, id\rangle$ subsumes the conclusion $C_t = \langle\{\{p(t)\}\}, id\rangle$ for all terms t, in which case the single conclusion C_X can replace all the conclusions C_t.

If, however, the context is not known and, thus, the set W may or may not contain the variable X, one has to be careful as C_X does *not* subsume C_t if $X \in W$. In that case, one can use the set $\{C_X \mid X \in \mathit{Var}\}$ of conclusions to replace the set $\{C_t \mid t \in \mathit{TabTerm}\}$, because the number of variable in a given context is always finite and there are, thus, always variables $X \in \mathit{Var}$ not occurring in the context. □

4.2.7 The Rigid Variable Version of an Ideal Ground Tableau Calculus

As said before, a rigid variable tableau calculus $\mathcal{C}^{\mathrm{rv}}$ is constructed by lifting a ground calculus $\mathcal{C}^{\mathrm{gd}}$. The following definition clarifies the relationship between $\mathcal{C}^{\mathrm{rv}}$ and $\mathcal{C}^{\mathrm{gd}}$.

Definition 4.2.16 Let $\mathcal{C}^{\mathrm{rv}}$ be an ideal rigid variable calculus for a logic **L**, and let $\mathcal{C}^{\mathrm{gd}}$ be an ideal ground calculus for **L** such that, for all signatures $\Sigma \in \mathit{Sig}$, the extended signature Σ^*_{gd} used by $\mathcal{C}^{\mathrm{gd}}$ is the signature for which

$$\mathit{TabForm}(\Sigma^*_{\mathrm{fv}}) = (\mathit{TabForm}(\Sigma^*_{\mathrm{gd}}))^{\mathrm{fv}}$$

holds, that has to exist according to the definition of free variable tableau calculi (Def. 4.2.3).

The calculus $\mathcal{C}^{\mathrm{rv}}$ is a *rigid variable version* of the calculus $\mathcal{C}^{\mathrm{gd}}$ (and $\mathcal{C}^{\mathrm{gd}}$ is a *ground version* of $\mathcal{C}^{\mathrm{rv}}$) if, for all (rigid variable) premisses $\Pi_{\mathrm{rv}} \subset \mathit{TabForm}(\Sigma^*_{\mathrm{fv}})$, all (ground) premisses $\Pi_{\mathrm{gd}} \subset \mathit{TabForm}(\Sigma^*_{\mathrm{gd}})$, and all substitutions $\tau \in \mathit{Subst}(\Sigma^*_{\mathrm{fv}})$ with finite domain such that

$$\Pi_{\mathrm{fv}}\tau = \Pi_{\mathrm{gd}} \ ,$$

the following holds, where $\mathcal{E}^{\mathrm{gd}}$ and $\mathcal{E}^{\mathrm{rv}}$ are the expansion rules of $\mathcal{C}^{\mathrm{gd}}$ resp. $\mathcal{C}^{\mathrm{rv}}$, and $W = \mathit{dom}(\tau)$:

$$\mathcal{E}^{\mathrm{gd}}(\Pi_{\mathrm{gd}}) = \{C_{\mathrm{gd}} \mid \text{there is a } \langle C_{\mathrm{rv}}, \sigma\rangle \in \mathcal{E}^{\mathrm{rv}}(\Pi_{\mathrm{rv}}) \text{ such that } \langle C_{\mathrm{rv}}, \sigma\rangle \leq^W \langle C_{\mathrm{gd}}, \tau\rangle\} \ .$$

□

	Π	C	
gd	$\{\mathsf{T}{:}{*}{:}(\forall x)(p(x,c))\}$	$\{\{\mathsf{T}{:}{*}{:}p(t,c)\}\}$	for all terms t
rv	$\{\mathsf{T}{:}{*}{:}(\forall x)(p(x,Y))\}$	$\langle\{\{\mathsf{T}{:}{*}{:}p(X,Y)\}\}, id\rangle$	for all $X \in Var$
	$\tau = \{Y \mapsto c\}$	$\rho = \{Y \mapsto c,\, X \mapsto t\}$	

	Π	C
gd	$\{\mathsf{T}{:}{*}{:}p(t) \wedge q(t)\}$	$\{\{\mathsf{T}{:}{*}{:}p(t), \mathsf{T}{:}{*}{:}q(t)\}\}$
rv	$\{\mathsf{T}{:}{*}{:}p(X) \wedge q(X)\}$	$\langle\{\{\mathsf{T}{:}{*}{:}p(X), \mathsf{T}{:}{*}{:}q(X)\}\}, id\rangle$
	$\tau = \{X \mapsto t\}$	$\rho = \{X \mapsto t\}$

	Π	C
gd	$\{\mathsf{T}{:}{*}{:}p(t), \mathsf{F}{:}{*}{:}p(t)\}$	$\{\{\bot\}\}$
rv	$\{\mathsf{T}{:}{*}{:}p(t'), \mathsf{F}{:}{*}{:}p(t'')\}$	$\langle\{\{\bot\}\}, \sigma\rangle$ where σ an MGU of t', t''
	τ such that $t = t'\tau = t''\tau$	$\rho = \tau$

Table 4.1: Examples for the relationship between the expansion rules of a ground calculus for PL1 and its rigid variable version (see Example 4.2.17).

Example 4.2.17 Table 4.1 shows examples from first-order predicate logic for the three main types of ground premisses (and their conclusions) for which an expansion rule is liftable and their rigid variable versions.

In each case, the ground expansion rule (see Section 3.6) allows to derive the conclusion C_{gd} from the premiss Π_{gd}, and the rigid variable expansion rule (see Section 4.2.10) allows to derive the conclusion C_{rv} from the premiss Π_{rv}.

Using the notation from Definition 4.2.16, the relationship is indicated by the substitutions τ for which $\Pi_{\mathrm{rv}}\tau = \Pi_{\mathrm{gd}}$ and the substitution ρ for which $C_{\mathrm{rv}}\rho = C_{\mathrm{gd}}$.

In the first example, it is *not* sufficient if $\langle\{\{\mathsf{T}{:}{*}{:}p(X,Y)\}\}, id\rangle$ is a conclusion of $\{\mathsf{T}{:}{*}{:}(\forall x)(p(x,Y))\}$ for just some rigid variable $X \in Var$, because a different substitution $\tau' = \{Y \mapsto c, X \mapsto d\}$ might instantiate X, in which case $X \in W$ and, thus,

$$\langle\{\{\mathsf{T}{:}{*}{:}p(X,Y)\}\}, id\rangle \not\lesssim^{W} \langle\{\{\mathsf{T}{:}{*}{:}p(t,c)\}\}, \{Y \mapsto c, X \mapsto d\}\rangle \ .$$

□

A ground version of any rigid variable calculus can easily be constructed by defining that, if some pair $\langle C_{\mathrm{rv}}, \sigma\rangle$ is a rigid variable conclusion of some premiss Π_{rv} and ρ is some substitution grounding for C_{rv} and Π_{rv} such that $\sigma \leq \rho$, then $C_{\mathrm{gd}} = C_{\mathrm{rv}}\rho$ is a ground conclusion of the premiss $\Pi_{\mathrm{gd}} = \Pi_{\mathrm{rv}}$.

The advantage of using lifting for constructing a rigid variable calculus $\mathcal{C}^{\mathrm{rv}}$ from a ground calculus $\mathcal{C}^{\mathrm{gd}}$ is that soundness and completeness of $\mathcal{C}^{\mathrm{rv}}$ follows from soundness and completeness of $\mathcal{C}^{\mathrm{gd}}$.

Theorem 4.2.18 *Let $\mathcal{C}^{\mathrm{rv}}$ be an ideal rigid variable calculus for a logic* $\mathbf{L}$*, and let $\mathcal{C}^{\mathrm{gd}}$ be an ideal ground calculus for* $\mathbf{L}$ *such that $\mathcal{C}^{\mathrm{rv}}$ is a rigid variable version of $\mathcal{C}^{\mathrm{gd}}$ (Def. 4.2.16).*

Then, for all signatures $\Sigma \in \mathit{Sig}$ and all finite sets $G \subset \mathit{Form}(\Sigma)$ of formulae, there is a $\mathcal{C}^{\mathrm{rv}}$-tableau proof for G if and only if there is a $\mathcal{C}^{\mathrm{gd}}$-tableau proof for G.

Proof: *If-part:* Assume that $T_0^{\mathrm{gd}}, \dots, T_n^{\mathrm{gd}}$ $(n \geq 0)$ is a tableau proof for G constructed using the ground calculus $\mathcal{C}^{\mathrm{gd}}$.

By induction on i, we prove that there is a rigid variable tableau proof $T_0^{\mathrm{rv}}, \dots, T_n^{\mathrm{rv}}$ such that $T_i^{\mathrm{rv}}\tau_i = T_i^{\mathrm{gd}}$ for substitutions $\tau_i \in \mathit{Subst}(\Sigma^*_{\mathrm{fv}})$ $(0 \leq i \leq n)$.

$i = 0$: Since G does not contain rigid variables, and $\mathcal{C}_{\mathrm{rv}}$ is non-structural, $T_0^{\mathrm{rv}}\tau_0 = T_0^{\mathrm{gd}}$ for $T_0^{\mathrm{rv}} = T_0^{\mathrm{gd}}$ and $\tau_0 = id$.

$i \to i+1$: If T_{i+1}^{gd} has been constructed from T_i^{gd} applying the expansion rule of $\mathcal{C}^{\mathrm{gd}}$ to a premiss Π^{gd} on a branch B_i^{gd} of T_i^{gd} and deriving a conclusion C^{gd}, then there is a premiss Π^{rv} on a branch B_i^{rv} of T_i^{rv} such that $\Pi_i^{\mathrm{rv}}\tau_i = \Pi^{\mathrm{gd}}$ and $B_i^{\mathrm{rv}}\tau_i = B_i^{\mathrm{gd}}$. Thus, according to the relationship between a ground calculus and its rigid variable version (Def. 4.2.16), there is a rigid variable conclusion $\langle C^{\mathrm{rv}}, \sigma\rangle$ that is derivable from Π^{rv} using the expansion rule of $\mathcal{C}^{\mathrm{rv}}$, and there is a substitution ρ_i such that $C^{\mathrm{rv}}\rho_i = C^{\mathrm{gd}}$ and $\tau_i \leq^W \rho_i$ where W is the domain of τ_i (that contains all rigid variables occurring in T_i^{rv}), which implies $T_i^{\mathrm{rv}}\rho_i = T_i^{\mathrm{gd}}$. Thus, the substitution $\tau_{i+1} = \rho_i$ and the tableau T_{i+1}^{rv} that is derived from T_i^{rv} expanding the branch B_i^{rv} of T_i^{rv} using the conclusion C^{rv} satisfy the condition $T_{i+1}^{\mathrm{rv}}\tau_{i+1} = T_{i+1}^{\mathrm{gd}}$.

Only-if-part: Assume that $T_0^{\mathrm{rv}}, \dots, T_n^{\mathrm{rv}}$ $(n \geq 0)$ is a tableau proof for G constructed using the rigid variable calculus $\mathcal{C}^{\mathrm{rv}}$.

Again, by induction on i, we prove that there is a ground tableau proof $T_0^{\mathrm{gd}}, \dots, T_n^{\mathrm{gd}}$ such that $T_i^{\mathrm{gd}} = T_i^{\mathrm{rv}}\tau_i$ for substitutions $\tau_i \in \mathit{Subst}(\Sigma^*_{\mathrm{fv}})$ $(0 \leq i \leq n)$.

$i = 0$: Since G does not contain rigid variables, and $\mathcal{C}_{\mathrm{gd}}$ is non-structural, $T_0^{\mathrm{gd}} = T_0^{\mathrm{rv}}\tau_0$ for $T_0^{\mathrm{gd}} = T_0^{\mathrm{rv}}$ and $\tau_0 = id$.

$i \to i+1$: If T_{i+1}^{rv} has been constructed from T_i^{rv} applying the rigid variable expansion rule of $\mathcal{C}^{\mathrm{rv}}$ to a premiss Π^{rv} on a branch B_i^{rv} of T_i^{fv} and deriving a rigid variable conclusion $\langle C^{\mathrm{rv}}, \sigma\rangle$, then there is a premiss Π^{gd} on a branch B_i^{gd} of T_i^{gd} such that $\Pi^{\mathrm{gd}} = \Pi_i^{\mathrm{rv}}\tau_i$ and $B_i^{\mathrm{gd}} = B_i^{\mathrm{rv}}\tau_i$. Thus, according to the relationship between a ground calculus and its rigid variable version (Def. 4.2.16), there is a substitution ρ such that $\sigma \leq^W \tau_i \leq^W \rho_i$ and the ground conclusion $C^{\mathrm{gd}} = C^{\mathrm{rv}}\rho_i$ is derivable from Π^{gd} using the expansion rule of C^{gd}. Since $\sigma \leq^W \tau_i \leq^W \rho_i$, we have $T_i^{\mathrm{gd}} = T_i\tau_i = T_i\sigma\rho_i$ and $C^{\mathrm{gd}} = C\rho_i = C\sigma\rho_i$. Thus, the substitution $\tau_{i+1} = \rho_i$ and the tableau T_{i+1}^{gd} that is derived from T_i^{gd} expanding the branch B_i^{gd} of T_i^{gd} using the conclusion C^{gd} satisfy the condition $T_{i+1}^{\mathrm{gd}} = T_{i+1}^{\mathrm{rv}}\tau_{i+1}$. $\Box$

Corollary 4.2.19 *Let $\mathcal{C}^{\mathrm{rv}}$ be an ideal rigid variable calculus for a logic $\mathbf{L}$, and let $\mathcal{C}^{\mathrm{gd}}$ be an ideal ground calculus for $\mathbf{L}$ such that $\mathcal{C}^{\mathrm{rv}}$ is a rigid variable version of $\mathcal{C}^{\mathrm{gd}}$.*

Then, $\mathcal{C}^{\mathrm{rv}}$ is sound and complete if and only if $\mathcal{C}^{\mathrm{gd}}$ is sound and complete.

Theorem 4.2.18 also implies that soundness and completeness of a rigid variable calculus $\mathcal{C}_1^{\mathrm{rv}}$ follows from soundness and completeness of a rigid variable calculus $\mathcal{C}_2^{\mathrm{rv}}$ if $\mathcal{C}_1^{\mathrm{rv}}$ and $\mathcal{C}_2^{\mathrm{rv}}$ have the same ground version $\mathcal{C}^{\mathrm{gd}}$.

4.2.8 Lifting: Constructing a Rigid Variable Calculus

In this section, we discuss how to actually construct a rigid variable version $\mathcal{C}^{\mathrm{rv}}$ of a ground calculus $\mathcal{C}^{\mathrm{gd}}$.

A trivial rigid variable version $\mathcal{C}^{\mathrm{rv}}$ can easily be constructed by defining that, given some ground conclusion C_{gd} of some ground premiss Π_{gd} and some substitution ρ that is grounding for a rigid variable premiss Π_{rv} such that $\Pi_{\mathrm{rv}}\rho = \Pi_{\mathrm{gd}}$, then $\langle C_{\mathrm{gd}}, \rho\rangle$ is a rigid variable conclusion of Π_{rv}.

Consider the following example from first-order predicate logic: As

$$C_{\mathrm{gd}} = \{\{\mathsf{T}{:}p(a)\}, \{\mathsf{T}{:}q(a)\}\}$$

is a ground conclusion of

$$\Pi_{\mathrm{gd}} = \{\mathsf{T}{:}(p(a) \vee q(a))\} \ ,$$

and $\Pi_{\mathrm{rv}}\rho = \Pi_{\mathrm{gd}}$ for $\rho = \{X \mapsto a\}$ and

$$\Pi_{\mathrm{rv}} = \{\mathsf{T}{:}(p(X) \vee q(X))\}$$

we conclude that

$$\langle C_{\mathrm{rv}}, \rho\rangle = \langle\{\{\mathsf{T}{:}p(a)\}, \{\mathsf{T}{:}q(a)\}\}, \{X \mapsto a\}\rangle$$

is a rigid variable conclusion of Π_{rv} in the rigid variable calculus. Such trivial rigid variable conclusions are, of course, not what we want. We intend to construct a rigid variable calculus where the conclusion of the above premiss Π_{rv} is the more general conclusion

$$C'_{\mathrm{rv}} = \langle\{\{\mathsf{T}{:}p(X)\}, \{\mathsf{T}{:}q(X)\}\}, id\rangle \ .$$

Thus, the general idea of lifting is to (repeatedly) replace conclusions (or sets of conclusions) of the trivial rigid variable calculus by more general conclusions in such a way that soundness and completeness is preserved.

Theorem 4.2.20 *Let $\mathcal{C}$ and $\mathcal{C}'$ be ideal rigid variable calculi for a logic* $\mathbf{L}$ *that are identical except for their expansion rules $\mathcal{E}$ resp. $\mathcal{E}'$.*

If $\mathcal{C}$ is sound and complete and, for all signatures $\Sigma \in Sig$, all rigid variable premisses Π over Σ^*_{fv}, and all finite sets W of rigid variables,

1. *for each rigid variable conclusion $\langle C', \tau' \rangle \in \mathcal{E}'(\Sigma)(\Pi)$, if C^{gd} is any ground conclusion over Σ^*_{gd} such that $\langle C', \tau' \rangle \leq^W \langle C^{\mathrm{gd}}, id \rangle$, then there is a rigid variable conclusion $\langle C, \tau \rangle \in \mathcal{E}(\Sigma)(\Pi)$ such that $\langle C, \tau \rangle \leq^W \langle C^{\mathrm{gd}}, id \rangle$* (soundness), *and*

2. *for each rigid variable conclusion $\langle C, \tau \rangle \in \mathcal{E}(\Sigma)(\Pi)$ there is a rigid variable conclusion $\langle C', \tau' \rangle \in \mathcal{E}'(\Sigma)(\Pi)$ such that $\langle C', \tau' \rangle \leq^W \langle C, \tau \rangle$* (completeness),

then the calculus $\mathcal{C}'$ is sound and complete as well.

The above theorem states the properties that a set of conclusions replacing a set of less general conclusions must have to preserve soundness and completeness of the calculus. It does not answer the question of how to construct such an appropriate set of more general conclusions. Unfortunately, there is no uniform method for constructing an optimal set containing only most general conclusions. There is, however, a method that yields good results in most cases; it is the method that is usually used (in an informal way) when a ground calculus is lifted in an *ad hoc* manner. The idea is to replace all occurrences of terms in a certain ground premiss Π^{gd} and in one of its ground conclusions C^{gd} by arbitrary terms containing rigid variables and then to check what most general substitution τ has to be applied and what other properties the new terms have to have to make sure that using the resulting rigid variable conclusion $\langle C^{\mathrm{rv}}, \tau \rangle$ as a possible conclusion for the resulting rigid variable premiss Π^{rv} leads to an expansion rule satisfying the conditions of Theorem 4.2.20 and, thus, to a sound and complete calculus.

Example 4.2.21 Assume that $\{\{\mathsf{T}{:}p(f(b))\}\}$ is a ground conclusion of the ground premiss $\{\mathsf{T}{:}(a \approx b),\ \mathsf{T}{:}p(f(a))\}$, i.e., the expansion rule allows the "application" of equalities to terms in formulae.

We replace the term occurrences in the premiss and the conclusion by arbitrary rigid variable terms, namely the occurrence of $f(a)$ by t, the occurrence of a in $a \approx b$ by t', the occurrence of b in $a \approx b$ by s, and the occurrence of $f(b)$ in the conclusion by s'.

Now, it is easy to check that using $\langle \{\{\mathsf{T}{:}p(s')\}\}, \tau \rangle$ as a conclusion for the premiss $\{\mathsf{T}{:}(t' \approx s),\ \mathsf{T}{:}p(t)\}$ preserves soundness and completeness according to Theorem 4.2.20 if τ is a unifier of t' and a subterm t'' of t and s' is the result of replacing t'' in t' by $s\tau$. □

In some cases a more general conclusion $\langle C', \tau' \rangle$ cannot be used, because it just slightly violates the soundness condition in Theorem 4.2.20, i.e., for many or even most—but not for all—ground conclusion C^{gd} such that $\langle C', \tau' \rangle \leq^{W} \langle C^{\mathrm{gd}}, id \rangle$ there is a (less general) rigid variable conclusion $\langle C, \tau \rangle$ with $\langle C, \tau \rangle \leq^{W} \langle C^{\mathrm{gd}}, id \rangle$. This problem can be overcome if decidable symbolic (i.e., syntactical) *constraints* can be formulated that separate the "good" from the "bad" ground conclusions C^{gd}.

The syntactical objects that are constituents of symbolic constraints have to be included in the extended signatures Σ^* that are used to construct tableau formulae, which immediately implies that no constraints are attached to (a) the formulae in an initial tableau and (b) the special tableau formula $\perp$. The constraints are made part of the label of a tableau formula. A special set of tableau interpretations is used to define the semantics of the ground calculus with constraints; in these interpretations all labels containing a constraint that evaluates to false represent a special world in which all formulae are false (thus, if the constraint in a label σ evaluates to false, a tableau formula of the form $\sigma{:}\mathsf{T}{:}\phi$ is *not* satisfied by *any* tableau interpretation and a tableau formula of the form $\sigma{:}\mathsf{F}{:}\phi$ is satisfied by *all* tableau interpretations).

Example 4.2.22 Assume that the ground conclusion $\{\{\mathsf{T}{:}\phi(\mathit{succ}(\mathit{pred}(n)))\}\}$ can be derived from the premiss $\{\{\mathsf{T}{:}\phi(\mathit{pred}(\mathit{succ}(n)))\}\}$ for all terms n that cannot (syntactically) be reduced to 0. In that case, it is not sound to derive the rigid variable conclusion $\{\{\mathsf{T}{:}\phi(\mathit{succ}(\mathit{pred}(X)))\}\}$ from the premiss $\{\{\mathsf{T}{:}\phi(\mathit{pred}(\mathit{succ}(X)))\}\}$.

If, however, the restriction that n cannot (syntactically) be reduced to 0 can be formulated as a symbolic constraint $\{n \not\equiv 0\}$, then it is sound for *all* terms n to derive the ground conclusion $\{\{\mathsf{T}{:}\{n \not\equiv 0\}{:}\phi(\mathit{succ}(\mathit{pred}(n)))\}\}$ from $\{\{\mathsf{T}{:}\phi(\mathit{pred}(\mathit{succ}(n)))\}\}$; and it is, thus, sound to derive the rigid variable conclusion $\{\{\mathsf{T}{:}\{X \not\equiv 0\}{:}\phi(\mathit{succ}(\mathit{pred}(X)))\}\}$ from the rigid variable premiss $\{\{\mathsf{T}{:}\phi(\mathit{pred}(\mathit{succ}(X)))\}\}$. $\square$

The symbolic constraints attached to tableau formulae that are necessary to preserve soundness of expansion rule applications have to be clearly separated from constraints that are used to organise proof search such as, for example, ordering constraints that implement selection functions (see Section 5.5).

4.2.9 Semantics of Rigid Variable Tableaux

Rigid variable calculi usually do not have a model semantics based on tableau interpretations, because the truth of different tableau formulae containing the same rigid variable is interrelated. Nevertheless, it is possible to define a semantics for rigid variable tableaux based on the semantics of the ground version of a rigid variable calculus $\mathcal{C}^{\mathrm{rv}}$; this semantics can be used to prove soundness of $\mathcal{C}^{\mathrm{rv}}$ by showing that expansion rule applications preserve satisfiability. For proving completeness, however, one cannot use this semantics, because the completeness criteria from Definition 3.5.6

and Theorem 3.5.7 only apply to non-destructive calculi, and rigid variable calculi are inherently destructive.

Definition 4.2.23 Let $\mathcal{C}^{\mathrm{rv}}$ be an ideal rigid variable calculus for a logic $\mathbf{L}$, and let $\mathcal{C}^{\mathrm{gd}}$ be an ideal ground calculus for $\mathbf{L}$ such that $\mathcal{C}^{\mathrm{rv}}$ is a rigid variable version of $\mathcal{C}^{\mathrm{gd}}$.

A rigid variable tableau T^{rv} over a signature Σ^*_{rv} is *satisfied* by a tableau interpretation $\langle \mathbf{m}, I\rangle \in \mathit{TabInterp}(\Sigma^*_{\mathrm{gd}})$ of $\mathcal{C}^{\mathrm{gd}}$ iff for all substitutions $\tau \in \mathit{Subst}(\Sigma^*_{\mathrm{fv}})$ that are grounding for T^{rv}, the ground tableau $T^{\mathrm{gd}} = T^{\mathrm{rv}}\tau$ is satisfied by $\langle \mathbf{m}, I\rangle$ (Def. 3.4.1). $\square$

Definition 4.2.24 Let $\mathcal{C}^{\mathrm{rv}}$ be an ideal rigid variable calculus for a logic $\mathbf{L}$, and let $\mathcal{C}^{\mathrm{gd}}$ be an ideal ground calculus for $\mathbf{L}$ such that $\mathcal{C}^{\mathrm{rv}}$ is a rigid variable version of $\mathcal{C}^{\mathrm{gd}}$.

The calculus $\mathcal{C}^{\mathrm{rv}}$ has the *strong soundness of expansion property for rigid variable calculi* if, for all signatures Σ_{fv} and all tableau T, T' over Σ^*_{fv}: if T' is a successor tableau of T, then T' is satisfied by the all tableau interpretations in $\mathit{TabInterp}(\Sigma^*_{\mathrm{gd}})$ that satisfy T (Def. 4.2.23). $\square$

Lemma 4.2.25 *Let $\mathcal{C}^{\mathrm{rv}}$ be an ideal rigid variable calculus for a logic $\mathbf{L}$, and let $\mathcal{C}^{\mathrm{gd}}$ be an ideal ground calculus for $\mathbf{L}$ such that $\mathcal{C}^{\mathrm{rv}}$ is a rigid variable version of $\mathcal{C}^{\mathrm{gd}}$.*

The calculus $\mathcal{C}^{\mathrm{rv}}$ has the strong soundness of expansion property for rigid variable calculi (Def. 4.2.24) if and only if $\mathcal{C}^{\mathrm{gd}}$ has the strong soundness of expansion property for ground calculi (Property 2 in Def. 3.5.8).

Proof: The lemma follows trivially from the definitions of the soundness properties and the relation between a rigid variable calculus and its ground version. $\square$

Theorem 4.2.26 *Let $\mathcal{C}^{\mathrm{rv}}$ be an ideal rigid variable calculus for a logic $\mathbf{L}$, and let $\mathcal{C}^{\mathrm{gd}}$ be an ideal ground calculus for $\mathbf{L}$ such that $\mathcal{C}^{\mathrm{rv}}$ is a rigid variable version of $\mathcal{C}^{\mathrm{gd}}$.*

If $\mathcal{C}^{\mathrm{rv}}$ has Property 1 from Definition 3.5.8 (appropriateness of the set of tableau interpretations) and the strong soundness of expansion property for rigid variable calculi (Def. 4.2.24), then $\mathcal{C}^{\mathrm{rv}}$ is sound.

Proof: As $\mathcal{C}^{\mathrm{rv}}$ has Property 1 from Definition 3.5.8, then $\mathcal{C}^{\mathrm{gd}}$ has that property as well, because the initial tableaux for a set $\mathfrak{F}$ of formulae are the same in both calculi. Since $\mathcal{C}^{\mathrm{rv}}$ has the strong soundness of expansion property for rigid variable calculi, $\mathcal{C}^{\mathrm{gd}}$ has the strong soundness of expansion property for ground calculi (Property 2 in Def. 3.5.8) according to Lemma 4.2.25.

Thus, $\mathcal{C}^{\mathrm{gd}}$ is sound (Theorem 3.5.4), which implies that $\mathcal{C}^{\mathrm{rv}}$ is sound as well (Corollary 4.2.19). $\square$

4.2.10 An Ideal Rigid Variable Calculus for PL1

In this section, as an example, an ideal rigid variable calculus $\mathcal{C}^{\mathrm{rv}}_{\mathrm{PL1}}$ for first-order predicate logic is defined, which is a rigid variable version of the ground calculus $\mathcal{C}^{\mathrm{gd}}_{\mathrm{PL1}}$ from Section 3.6 and can constructed from $\mathcal{C}^{\mathrm{gd}}_{\mathrm{PL1}}$ using the lifting technique described in Section 4.2.8.

For each first-order signature Σ (see Section 2.3), the extended signature Σ^*_{fv} contains the free variables from Var as constants (see Section 2.3) and, in addition, the set $F^{sko}(\Sigma)$ of Skolem function symbols containing infinitely many symbols of each arity $n \geq 0$. Consequently, the set $\mathit{TabForm}(\Sigma^*_{\mathrm{fv}})$ is a language with the set $\mathit{Term}(\Sigma^*_{\mathrm{fv}})$ of terms (where all free variables and terms are of the same sort), and it is easy to verify that

$$\mathit{TabForm}(\Sigma^*_{\mathrm{fv}}) = (\mathit{TabForm}(\Sigma^*_{\mathrm{gd}})^{\mathrm{fv}}$$

where Σ^*_{gd} is the extensions of Σ by the set $F^{sko}(\Sigma)$ of Skolem function symbols.

Example 4.2.27 If a is a constant of the signatur Σ but it is not a Skolem constant, then the atom $p(a)$ is a formula over Σ and the extended signatures Σ^*_{gd} and Σ^*_{fv}. If c is a Skolem constant, then $p(c)$ is a formula over Σ^*_{gd} and Σ^*_{fv} but not over Σ. The atoms $p(X)$ and $p(c, X)$ are formulae only over Σ^*_{fv}. Note that $p(c, x)$ is not a formula over any signature, as formulae must not contain free object variables. $\square$

The set of labels and the initial label of $\mathcal{C}^{\mathrm{rv}}$ are the same as that of $\mathcal{C}^{\mathrm{gd}}$, i.e., the label $*$ represents the single world of PL1-models.

To define the expansion rule of $\mathcal{C}^{\mathrm{rv}}$, we again use *unifying notation*, i.e., tableau formulae with rigid variables are devided into α-, β-, δ-, and γ-formulae according to Table 3.1 in the same way as ground tableau formulae.

The expansion rule schemata of $\mathcal{C}^{\mathrm{rv}}$ are constructed lifting the schemata of $\mathcal{C}^{\mathrm{gd}}$. The schemata for premisses consisting of α-, β-, and γ-formulae are obviously stable w.r.t. replacement of terms by terms; new rigid variables are introduced by rule applications to premisses containing γ-formulae.

The schema for premisses consisting of δ-formulae can be lifted as well, provided that an appropriate version of the schema is used. As is easy to check, the version of the schema is stable where all ground terms occurring in a formula $\delta(x)$ are made arguments of the Skolem term that replaces the bound object variable x. Thus, the main difficulty in designing an ideal rigid variable calculus for PL1 has already been overcome by designing a stable ground expansion rule schema for δ-formulae.

Definition 4.2.28 Given a signature $\Sigma \in \mathit{Sig}_{\mathrm{PL1}}$, a *free variable Skolem term assignment* is a function $\mathit{sko}_{\mathrm{fv}}$ assigning to each δ-formula $\phi \in \mathit{TabForm}_{\mathrm{PL1}}(\Sigma^*_{\mathrm{fv}})$ a term

$$\mathit{sko}_{\mathrm{fv}}(\phi) = f(X_1, \ldots, X_k) \in \mathit{Term}^0_{\mathrm{PL1}}(\Sigma^*_{\mathrm{fv}})$$

such that

1. (a) $f \in F^{sko}(\Sigma)$,
   (b) $k = \alpha_{\Sigma}^{sko}(f)$, und
   (c) $X_1, \ldots, X_k$ are the free variables occurring in ϕ;
2. for all $f' \in F^{sko}(\Sigma)$, if f' occurs in ϕ, then $f > f'$ where $>$ is an arbitrary but fixed ordering on $F^{sko}(\Sigma)$; and
3. for all δ-formulae $\psi \in \mathit{TabForm}_{\mathrm{PL1}}(\Sigma^*)$, if $\mathit{sko}(\psi) = f(X'_1, \ldots X'_k)$, then the formulae ϕ and ψ are identical up to renaming of bound object variables and up to replacing all occurrences of free variables X_i by X'_i ($1 \leq i \leq k$). □

The expansion rule of the ground calculus $\mathcal{C}^{\mathrm{gd}}$ is only unstable for premisses allowing to close a branch. It requires the formulae in the two complementary atoms $\mathsf{T}{:}{*}{:}G$ and $\mathsf{F}{:}{*}{:}G$ that form the minimal premiss for the deduction of $\perp$ to be identical. Thus, the rigid variable version of the rule schema for closing branches is that a premiss $\Pi = \{\mathsf{T}{:}{*}{:}G,\ \mathsf{F}{:}{*}{:}G'\}$ allows the deduction of a conclusion $\langle\{\{\perp\}\}, \mu\rangle$ if μ is a unifier of G and G'. As it is sufficient to use set of most general conclusions (Section 4.2.8), the restriction to conclusions $\langle\{\{\perp\}\}, \mu\rangle$ where μ is a *most general* unifier of G and G' preserves completeness.

In Table 4.2, the rigid variable expansion rule $\mathcal{E}^{\mathrm{rv}}_{\mathrm{PL1}}$ of $\mathcal{C}^{\mathrm{rv}}_{\mathrm{PL1}}$ is given schematically; and the following is its formal definition:

Definition 4.2.29 The expansion rule $\mathcal{E}^{\mathrm{rv}}_{\mathrm{PL1}}$ of $\mathcal{C}^{\mathrm{rv}}_{\mathrm{PL1}}$ is, for all signatures $\Sigma \in \mathit{Sig}_{\mathrm{PL1}}$ and all premisses $\Pi \subset \mathit{TabForm}_{\mathrm{PL1}}(\Sigma^*_{\mathrm{rv}})$, defined by: the set $\mathcal{E}(\Sigma)^{\mathrm{rv}}_{\mathrm{PL1}}(\Pi)$ of possible conclusions is the smallest set containing the following rigid variable conclusions:

– $\{\langle\{\alpha_1, \alpha_2\}, \mathit{id}\rangle\}$	for all $\alpha \in \Pi$,
– $\{\langle\{\beta_1\}, \{\beta_2\}, \mathit{id}\rangle\}$	for all $\beta \in \Pi$,
– $\{\langle\{\gamma_1(X)\}, \mathit{id}\rangle\}$	for all $\gamma \in \Pi$ and all rigid variables $X \in \mathit{Var}$,
– $\{\langle\{\delta(t)\}, \mathit{id}\rangle\}$	for all $\delta \in \Pi$ where $t = \mathit{sko}_{\mathrm{fv}}(\delta)$ (Def. 4.2.28),
– $\{\langle\{\perp\}, \mu\rangle\}$	if $\mathsf{T}{:}\sigma{:}G, \mathsf{F}{:}\sigma{:}G' \in \Pi$ such that $G, G' \in \mathit{Atom}_{\mathrm{PL1}}(\Sigma^*_{\mathrm{fv}})$ are unifiable atoms, and μ is an MGU of G and G'

□

It is easy to check that the calculus $\mathcal{C}^{\mathrm{rv}}_{\mathrm{PL1}}$ is a rigid variable version of the sound and complete ground calculus $\mathcal{C}_{\mathrm{PL1}}$ defined in Section 3.6, which (using Corollary 4.2.19) implies soundness and completeness of $\mathcal{C}^{\mathrm{rv}}_{\mathrm{PL1}}$.

Theorem 4.2.30 *The calculus $\mathcal{C}^{\mathrm{rv}}_{\mathrm{PL1}}$ is sound and complete.*

$$\frac{\alpha}{\begin{array}{c}\alpha_1\\ \alpha_2\end{array}} \qquad \frac{\beta}{\beta_1 \mid \beta_2} \qquad \frac{\gamma(x)}{\gamma_1(X)} \qquad \frac{\delta(x)}{\delta_1(t)}$$

where X is any rigid variable

where $t = \mathit{sko}_{\mathrm{fv}}(\delta)$ (see Def. 4.2.28)

$$\begin{array}{c}\phi\\ \hline \psi\\ \hline \bot\end{array}$$

where ϕ and ψ are unifiable atomic formulae and an MGU of ϕ and ψ is applied to the tableau

Table 4.2: Rigid variable rule schemata for first-order predicate logic.

4.2.11 An Ideal Rigid Variable Tableau for the Modal Logic K

As a second example, we define a rigid variable version $\mathcal{C}_{\mathrm{K}}^{\mathrm{rv}}$ of the ground calculus $\mathcal{C}_{\mathrm{K}}^{\mathrm{con}}$ for the modal logic K from Section 3.7.4, which an expansion rule that is continuous w.r.t. premisses containing ν-formulae and that is, thus, for these premisses liftable.

Since $\mathit{Form}_{\mathrm{mod}}(\Sigma)$ is a not a language with terms, rigid variables are not introduced into the formula part of a tableau formulae but into labels; the set of labels of $\mathcal{C}_{\mathrm{K}}^{\mathrm{con}}$ is a language with terms (where terms are natural numbers). Thus, the set of labels of $\mathcal{C}_{\mathrm{K}}^{\mathrm{rv}}$ is $\mathit{Lab}_{\mathrm{fv}} = \mathit{CondLab}(\mathbb{N} \cup \mathit{Var})$ (where $\mathit{CondLab}(\mathbb{N} \cup \mathit{Var})$ is defined analogously to $\mathit{CondLab}(\mathbb{N})$, see Definition 3.7.1), with the initial label 1. Using this set $\mathit{Lab}_{\mathrm{fv}}$ of labels, we trivially have $\mathit{TabForm}_{\mathrm{fv}}(\Sigma_{\mathrm{mod}}) = (\mathit{TabForm}_{\mathrm{gd}}(\Sigma_{\mathrm{mod}}))^{\mathrm{fv}}$ (the signatures are not extended by additional [Skolem] symbols).

As in calculi for first-order predicate logic, the expansion rule schemata is obviously stable for premisses consisting of α- and β-formulae. The rôle of the schema for γ-formulae in $\mathcal{C}_{\mathrm{PL1}}^{\mathrm{rv}}$, however, is now played by the schema for the ν-formulae of modal logics. This schema allows, for example, to derive from a premiss Π containing $\mathsf{T}{:}1{:}\Box G$ the conclusion $\{\{\mathsf{T}{:}1.(n){:}p\}\}$ for *all* $n \in \mathbb{N}$. The schema is liftable, and its rigid variable version allows to deduce from Π the conclusion $\{\{\mathsf{T}{:}1.(X){:}p\}\}$ for all rigid variables X.

When a branch is closed, all rigid variables occurring in the labels of the involved complementary atoms are instantiated because the labels have to be justified by other labels on the branch, and all non-conditional positions in labels are ground (i.e., they consist of natural numbers and not of variables). For example, complementary atoms $\phi_1 = \mathsf{T}{:}1.(X){:}p$ and $\phi_2 = \mathsf{F}{:}1.(Y){:}p$ can only be used to close a branch if the branch contains formula such as $\psi = \mathsf{T}{:}1.1{:}q$ whose label justifies (instances of) the label $1.(X)$ and $1.(Y)$. Thus, the conclusion $\langle\{\{\bot\}\}, \{X \mapsto 1, Y \mapsto 1\}$ is derived from the premiss $\{\phi_1, \phi_2, \psi\}$.

In Table 4.2, the rigid variable expansion rule $\mathcal{E}_{\mathrm{mod}}^{\mathrm{rv}}$ of $\mathcal{C}_{\mathrm{mod}}^{\mathrm{rv}}$ is given schematically; and

$$\frac{\alpha}{\begin{matrix}\alpha_1\\ \alpha_2\end{matrix}} \qquad \frac{\beta}{\beta_1 \mid \beta_2} \qquad \frac{\mathsf{T}{:}\sigma{:}\Box G}{\mathsf{T}{:}\sigma.(X){:}G} \qquad \frac{\mathsf{F}{:}\sigma{:}\Diamond G}{\mathsf{F}{:}\sigma.(X){:}G}$$

for all $X \in \mathit{Var}$

$$\frac{\mathsf{T}{:}\sigma{:}\Diamond F}{\mathsf{T}{:}\sigma.n{:}G} \qquad \frac{\mathsf{F}{:}\sigma{:}\Box G}{\mathsf{F}{:}\sigma.n{:}G} \qquad \frac{\mathsf{T}{:}\sigma{:}\neg G}{\mathsf{F}{:}\sigma{:}G} \qquad \frac{\mathsf{F}{:}\sigma{:}\neg G}{\mathsf{T}{:}\sigma{:}G}$$

where $n = \lceil G \rceil$ where $n = \lceil \neg G \rceil$

$$\frac{\begin{matrix}\mathsf{T}{:}\sigma{:}G\\ \mathsf{F}{:}\sigma'{:}G\end{matrix}}{\bot}$$

if there is a substitution μ such that $[\sigma\mu] = [\sigma'\mu]$ and
that label is justified by formulae on the branch
after μ has been applied;
a most general such substitution μ has to be applied to the tableau

Table 4.3: Expansion rule schemata of the rigid variable calculus $\mathcal{C}_{\mathrm{K}}^{\mathrm{rv}}$ for the modal logic K].

it is formally defined as follows:

Definition 4.2.31 For all premisses $\Pi \subset \mathit{TabForm}_{\mathrm{mod}}$, the set $\mathcal{E}_{\mathrm{K}}^{\mathrm{rv}}(\Sigma)(\Pi)$ is the smallest set containing the following conclusions (where $\lceil \cdot \rceil$ is any bijection from the set $\mathit{Form}_{\mathrm{mod}}(\Sigma)$ of modal formulae to the set of natural numbers):

– $\langle\{\{\alpha_1, \alpha_2\}\}$	for all $\alpha \in \Pi$,
– $\langle\{\{\beta_1\}, \{\beta_2\}\}, \mathit{id}\rangle$	for all $\beta \in \Pi$,
– $\langle\{\{\mathsf{T}{:}\sigma.(X){:}G\}\}, \mathit{id}\rangle$	for all $\mathsf{T}{:}\sigma{:}\Box G \in \Pi$ and all $X \in \mathit{Var}$,
– $\langle\{\{\mathsf{F}{:}\sigma.(X){:}G\}\}, \mathit{id}\rangle$	for all $\mathsf{F}{:}\sigma{:}\Diamond G \in \Pi$ and all $X \in \mathit{Var}$,
– $\langle\{\{\mathsf{F}{:}\sigma.n{:}G\}\}, \mathit{id}\rangle$	for all $\mathsf{F}{:}\sigma{:}\Box G \in \Pi$ where $n = \lceil \neg G \rceil$,
– $\langle\{\{\mathsf{T}{:}\sigma.n{:}G\}\}, \mathit{id}\rangle$	for all $\mathsf{T}{:}\sigma{:}\Diamond G \in \Pi$ where $n = \lceil G \rceil$,
– $\langle\{\{\bot\}\}, \mu\rangle$	if $\mathsf{T}{:}\sigma{:}G, \mathsf{F}{:}\sigma'{:}G \in \Pi$ and μ is a most general substitution such that $[\sigma\mu] = [\sigma'\mu]$, and $\sigma\mu, \sigma'\mu$ are justified by $\Pi\mu$.

□

There may be different ways to instantiate variables in a label such that it is justified on a tableau branch.

Example 4.2.32 Consider the premiss

$$\Pi = \{\mathsf{T}{:}1.1.1{:}q,\ \mathsf{T}{:}1.2{:}r,\ \mathsf{T}{:}1.(X){:}p\ \mathsf{F}{:}1.(X){:}p\}\ .$$

The set $\mathcal{E}^{\text{rv}}_{\text{mod}}(\Pi)$ of consists of the conclusions

$$\langle\{\{\bot\}\},\{X \mapsto 1\}\rangle \text{ and } \langle\{\{\bot\}\},\{X \mapsto 2\}\rangle \ .$$

□

The calculus $\mathcal{C}^{\text{rv}}_{\text{K}}$ is sound and complete, because it is a rigid variable version of the sound and complete calculus $\mathcal{C}^{\text{con}}_{\text{K}}$ defined in Section 3.7.4.

Theorem 4.2.33 *The calculus $\mathcal{C}^{\text{rv}}_{\text{K}}$ for the logic* K *is sound and complete.*

Example 4.2.34 As an example, we again prove unsatisfiability of the formula G from Examples 3.7.16 and 3.7.22, now using the rigid variable calculus $\mathcal{C}^{\text{rv}}_{\text{K}}$ defined above. A closed rigid variable tableau $T^{\text{con}}_{\text{rv}}$ for G that has been constructed using the expansion rule of $\mathcal{C}^{\text{con}}_{\text{K}}$ is shown in Figure 4.1 (the figure shows the tableau formulae with uninstantiated rigid variables; the substitutions that have to be applied during the construction of the tableau proof are listed separately).

The proof has the same structure as that shown in Figure 3.2, which is constructed using the ground version of the calculus. The difference is that, when the expansion rule schema for ν-formulae is applied to add formulae 6, 7, and 19, the label that is introduced does not have to be "guessed". Instead, the rigid variable labels $1.(X_1)$, $1.(X_2)$, and $1.(X_3)$ are introduced, respectively. The rigid variables are instantiated later on when the expansion rule is applied to close the left and the right branch of the tableau.

When the left branch is closed, it is not sufficient to apply the substitution $\{X_1 \mapsto X_2\}$, i.e., $\langle\{\{\bot\}\},\{X_1 \mapsto X_2\}\rangle$ is not a valid conclusion of any premiss on the left branch of the tableau, because the label $1.(X_2)$ is not justified. Both X_1 and X_2 have to be instantiated with 1, such that the justified label $1.(1)$ is created.

After $\sigma_1 = \{X_1 \mapsto 1, X_2 \mapsto 1\}$ has been applied to the tableau, closing the middle branch does not require a further instantiation of rigid variables. However, to close the right branch, the substitution σ_2 has to be applied, that instantiates X_3 with 2.

This example demonstrates the advantage of using rigid variables. When the branches are closed, there is only one most general substitution that can be applied (in this example), i.e., in each case the choice of instantiations for free variables is deterministic. Contrary to that, when the ground version of the calculus is used, new labels indeed have to be guessed because at the points where labels are introduced, it is not obvious whether the label $1.(1)$ or the label $1.(2)$ should be used, which corresponds to the alternative of instantiating any of the rigid variables with either 1 or 2. □

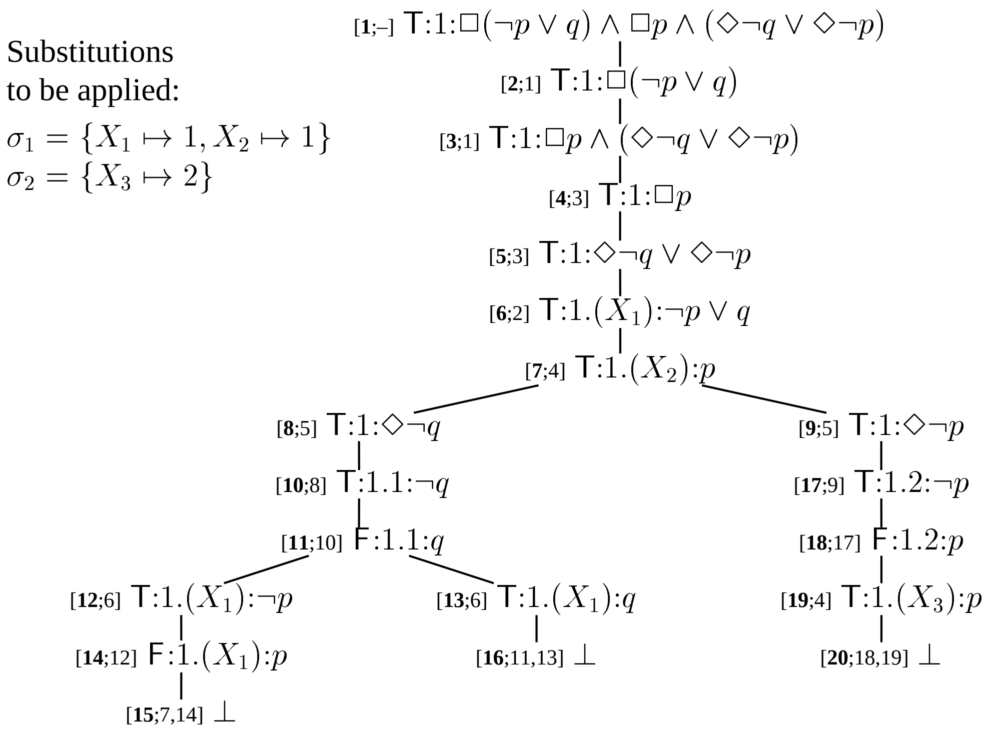

Substitutions to be applied:
$\sigma_1 = \{X_1 \mapsto 1, X_2 \mapsto 1\}$
$\sigma_2 = \{X_3 \mapsto 2\}$
[1;–] T:1:$\Box(\neg p \vee q) \wedge \Box p \wedge (\Diamond\neg q \vee \Diamond\neg p)$
[2;1] T:1:$\Box(\neg p \vee q)$
[3;1] T:1:$\Box p \wedge (\Diamond\neg q \vee \Diamond\neg p)$
[4;3] T:1:$\Box p$
[5;3] T:1:$\Diamond\neg q \vee \Diamond\neg p$
[6;2] T:1.(X_1):$\neg p \vee q$
[7;4] T:1.(X_2):p
[8;5] T:1:$\Diamond\neg q$
[9;5] T:1:$\Diamond\neg p$
[10;8] T:1.1:$\neg q$
[17;9] T:1.2:$\neg p$
[11;10] F:1.1:q
[18;17] F:1.2:p
[12;6] T:1.(X_1):$\neg p$
[13;6] T:1.(X_1):q
[19;4] T:1.(X_3):p
[14;12] F:1.(X_1):p
[16;11,13] $\perp$
[20;18,19] $\perp$
[15;7,14] $\perp$

Figure 4.1: The tableau $T^{\text{con}}_{\text{rv}}$ from Example 4.2.34 is the result of applying the listed substitutions to the above tree.

4.3 Universal Variable Calculi

4.3.1 The Idea of Universal Variable Calculi

Under certain conditions, there is an alternative use of free variables for strengthening a tableau calculus. Instead of using a free variable to represent a single but unknown term, it can be used as well to represent *all* terms. Then, a formula containing such a free variable $\boldsymbol{x}$ stands for the set of *all* formulae that are the result of replacing $\boldsymbol{x}$ by some term. Intuitively, these free variables can be seen as being universally quantified on the meta-level; accordingly they are called *universal variables*. In the following, to clearly separate rigid and universal variables, we use the variables in the set Var only as rigid variables; universal variables are taken from the separate set $UVar = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots\}$ that is disjoint from $Var = \{X_1, X_2, \ldots\}$. Universal variables are never instantiated; no substitutions are applied to tableaux containing universal (and no rigid) variables.

Definition 4.3.1 Let **L** be a logic. A free variable calculus $\mathcal{C}^{\text{uv}}$ for **L** is a *universal variable calculus* (for **L**) if its tableau formulae only contain free variables from the set $UVar$ and not from the set Var. □

Definition 4.3.2 Let $\mathcal{C}^{\mathrm{uv}}$ be a universal variable calculus for a logic **L**; let $\Sigma \in \mathit{Sig}$ be a signature; and let $\Phi \subset \mathit{TabForm}(\Sigma^*_{\mathrm{fv}})$ be a set of tableau formulae.

The set $\mathit{Inst}(\Phi) \subset \mathit{TabForm}(\Sigma^*_{\mathrm{gd}})$ of *instances* of formulae in Φ is defined by:

$$\mathit{Inst}(\Phi) = \{\phi\tau \mid \tau \in \mathit{Subst}^{\mathrm{fv}}(\Sigma^*) \text{ is grounding for } \phi\} \ .$$

□

Idealness and other syntactical notions such as conclusions and expansion rules are defined for universal variable calculi in the same way as for ground calculi.

The advantage of using universal variables is the following: Often several different instances of a tableau formula containing free variables have to be used to close a branch (or a subtableau). In rigid variable calculi the mechanism to do so is to apply the expansion rule more than once to premisses that allow the introduction of new rigid variables to generate variants of the tableau formulae. Rigid variables are *not* implicitly universally quantified (as it is, for instance, the case with variables in clauses when using a resolution calculus). Suppose a tableau branch B contains a formula $\phi(X)$; assume further that the expansion of the tableau then proceeds with creating new branches. Some of these new (sub-)branches contain occurrences of the rigid variable X; when X is instantiated, the same substitution for X has to be used on all of them. In particular situations, however, it may be possible—without destroying soundness of the calculus—to add the formula $\phi(\boldsymbol{x})$ to B. In such cases, different instances $\phi(t)$ of $\phi(\boldsymbol{x})$ can be used to expand the branch B—without first generating variants $\phi(X'), \phi(X''), \ldots$ of $\phi(X)$. Recognising such situations and exploiting them yields shorter tableau proofs, and in most cases reduces the search space. If both universal and rigid variables are used in a calculus, then more general substitutions can be used in rigid variable conclusions as compared to the corresponding calculus that uses only rigid variables.

Intuitively, if a branch B contains a tableau formula $\phi(\boldsymbol{x})$, that means that one could add $\phi(t)$ to B for arbitrary terms t without creating any new non-closed branches (this intuition, however, is only appropriate if no information is hidden in the structure of a tableau branch, i.e., if the calculus is ideal).

Example 4.3.3 Figure 4.2 shows an example for the usefulness of universal variables. The tableau T^{rv}_1 (top left in the figure) for the set $\mathfrak{F} = \{(\forall x)(p(x)), \neg p(a) \vee \neg p(b)\}$ of PL1-formulae cannot be closed immediately as no single substitution for X allows to add $\perp$ to both branches. To find a proof, the expansion rule has to be applied again to the γ-formula $\mathsf{T}{:}(\forall x)(p(x))$ to add a variant $\phi(X') = \mathsf{T}{:}p(X')$ of $\phi(X) = \mathsf{T}{:}p(X)$. Then, the closed tableau T^{rv}_2 (top right in the figure) can be deduced.

The tableau T^{uv}_1 (bottom left in the figure), however, that contains the formula $\mathsf{T}{:}p(\boldsymbol{x})$ instead of $\mathsf{T}{:}p(X)$ can expanded to a closed tableau T^{uv}_2 (bottom right in the figure) without applying a substitution, because $\mathsf{T}{:}p(\boldsymbol{x})$ represents all formulae of the form $\mathsf{T}{:}p(t)$, including $\mathsf{T}{:}p(a)$ and $\mathsf{T}{:}p(b)$. □

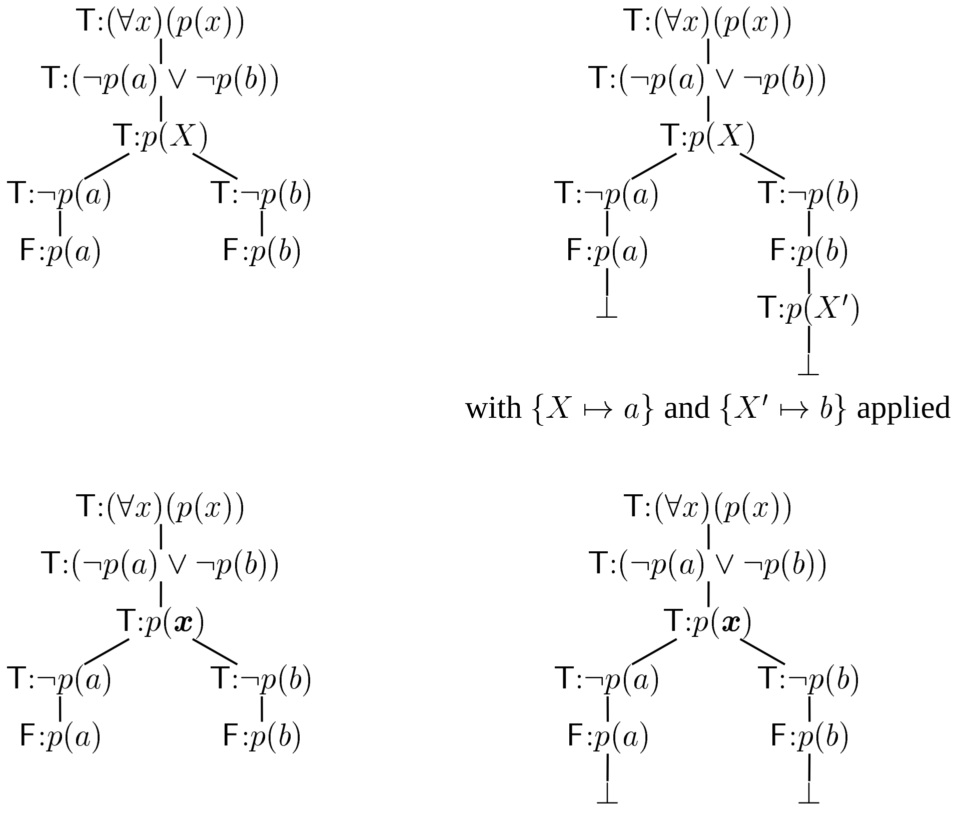


Figure 4.2: Example for the usefulness of universal variables; the tableaux T_1^{rv} (top left), T_2^{rv} (top right), T_1^{uv} (bottom left), T_2^{uv} (bottom right) from Example 4.3.3.

An additional advantage of universal variables is that they help to avoid redundancies inherent to rigid variable calculi. If, for example, a rigid variable tableau branch contains the formulae

$$\begin{aligned}\phi(X_1) &= \mathsf{T}{:}p(X_1)\\ \phi(X_2) &= \mathsf{T}{:}p(X_2)\\ \psi_1 &= \mathsf{F}{:}p(a)\\ \psi_2 &= \mathsf{F}{:}p(b)\end{aligned}$$

then there are four different possibilities to close the branch. If, however, the branch contains the universal variable formula $\phi(\boldsymbol{x}) = \mathsf{T}{:}p(\boldsymbol{x})$ instead of $\phi(X_1)$ and $\phi(X_2)$, then there is only one conclusion that closes the branch, namely $\{\{\bot\}\}$, and not variable has to be instantiated.

Universal variables in a tableau can be renamed arbitrarily, as long as all occurrences of a variable in the same tableau formula are replaced by the same new variable. Therefore, a premiss containing the complementary atoms $\mathsf{T}{:}p(\boldsymbol{x})$ and $\mathsf{F}{:}p(f(\boldsymbol{x}))$ can be used to close a branch, as the universal variable $\boldsymbol{x}$ can be renamed in one of the two atoms.

The method of using universal variables in a tableau calculus for first-order predicate logic has first been described in (Beckert & Hähnle, 1992) and has been further improved in (Beckert & Hähnle, 1998). A universal variable calculus for modal logics has been described in (Beckert & Goré, 1997). In (Bibel, 1982), a technique called *splitting by need* has been proposed for the connection method; it is—like the universal variable method—based on the idea to avoid copying a universally quantified formula in cases where it is sound to use a single copy with different instantiations for its variables.

4.3.2 The Universal Variable Version of a Ground Calculus

Universal variable calculi $\mathcal{C}^{\mathrm{uv}}$ are usually constructed by lifting a ground calculus $\mathcal{C}^{\mathrm{gd}}$ (similar to rigid variable calculi), such that soundness and completeness of $\mathcal{C}^{\mathrm{uv}}$ follows from soundness and completeness of $\mathcal{C}^{\mathrm{gd}}$.

Definition 4.3.4 Let $\mathcal{C}^{\mathrm{uv}}$ be an ideal universal variable calculus for a logic $\mathbf{L}$; and let $\mathcal{C}^{\mathrm{gd}}$ be an ideal ground calculus for $\mathbf{L}$ such that, for all signatures $\Sigma \in \mathit{Sig}$, the extended signature Σ^*_{gd} used by $\mathcal{C}^{\mathrm{gd}}$ is the signature for which

$$\mathit{TabForm}(\Sigma^*_{\mathrm{fv}}) = (\mathit{TabForm}(\Sigma^*_{\mathrm{gd}}))^{\mathrm{fv}}$$

holds, that has to exist according to the definition of free variable tableau calculi (Def. 4.2.3).

The calculus $\mathcal{C}^{\mathrm{uv}}$ is a *universal variable version* of $\mathcal{C}^{\mathrm{gd}}$ (and $\mathcal{C}^{\mathrm{gd}}$ is a *ground version* of $\mathcal{C}^{\mathrm{uv}}$) if, for all (universal variable) premisses $\Pi_{\mathrm{uv}} \subset \mathit{TabForm}(\Sigma^*_{\mathrm{fv}})$, the sets

$$\bigcup\{\mathcal{E}^{\mathrm{gd}}(\Pi_{\mathrm{gd}}) \mid \Pi_{\mathrm{gd}} \text{ is a finite subset of } \mathit{Inst}(\Pi_{\mathrm{uv}})\}$$

and

$$\begin{aligned}\{\{E_1\tau_1, \ldots, E_n\tau_n\} \mid \{E_1, \ldots, E_n\} \in \mathcal{E}^{\mathrm{uv}}(\Pi_{\mathrm{uv}}) \text{ and, for } 1 \le i \le n,\\ \tau_i \in \mathit{Subst}^{\mathrm{fv}}(\Sigma^*_{\mathrm{fv}}) \text{ is a substitution that is grounding for } E_i\}\end{aligned}$$

are identical. □

The following theorem relates soundness and completeness of an ideal universal variable calculus and soundness and completeness of its ground version. The proof ot the theorem is constructive; thus, it provides an algorithm for constructing a ground tableau proof from a universal variable tableau proof. As the ground tableau proof may be exponentially larger, the only-if part of the proof is somewhat tricky; there is no one-to-one correspondence between expansion rule applications in the ground an the universal variable proof (in Example 4.3.7 an example for the transformation is given).

Theorem 4.3.5 *Let $\mathcal{C}^{\mathrm{uv}}$ be an ideal universal variable calculus for a logic* $\mathbf{L}$*; and let $\mathcal{C}^{\mathrm{gd}}$ be an ideal ground calculus for* $\mathbf{L}$ *such that $\mathcal{C}^{\mathrm{uv}}$ is a universal variable version of $\mathcal{C}^{\mathrm{gd}}$ (Def. 4.3.4).*

Then, for all signatures $\Sigma \in \mathit{Sig}$ and all finite sets $G \subset \mathit{Form}(\Sigma)$ of formulae, there is a $\mathcal{C}^{\mathrm{uv}}$-tableau proof for G if and only if there is a $\mathcal{C}^{\mathrm{gd}}$-tableau proof for G.

Proof: *If-part:* Assume that $T^{\mathrm{gd}}_1, \ldots, T^{\mathrm{gd}}_n$ ($n \ge 1$) is a tableau proof for G constructed using the ground calculus $\mathcal{C}^{\mathrm{gd}}$.

By induction on i, we prove that a sequence $T^{\mathrm{uv}}_1, \ldots, T^{\mathrm{uv}}_n$ of universal variable tableaux for G exists such that for each branch B^{uv} of T^{uv}_i there is a branch B^{gd} of T^{gd}_i with

$$\mathit{Form}(B^{\mathrm{gd}}) \subset \mathit{Inst}(\mathit{Form}(B^{\mathrm{uv}})) \ .$$

That implies, as all branches of the closed tableau T^{gd}_n contain $\bot$, that all branches of T^{uv}_n contain $\bot$. Therefore T^{uv}_n is closed.

$i = 1$: An arbitrary initial tableau T^{uv}_1 for G does not contain any universal variables; therefore, $\mathit{Form}(B^{\mathrm{gd}}) = \mathit{Form}(B^{\mathrm{uv}}) = \mathit{Inst}(\mathit{Form}(B^{\mathrm{uv}}))$.

$i \to i+1$: Let B^{gd}_i be the branch of T^{gd}_i that has been expanded using a premiss $\Pi \subset \mathit{Form}(B^{\mathrm{gd}}_i)$ and a conclusion $C^{\mathrm{gd}} = \{E^{\mathrm{gd}}_1, \ldots, E^{\mathrm{gd}}_k\}$. According to the definition of the relationship between universal variable calculi and their ground version, there has to be a premiss Π^{uv} on each branch B^{uv}_i with $\mathit{Form}(B^{\mathrm{gd}}_i) \subset \mathit{Inst}(\mathit{Form}(B^{\mathrm{uv}}_i))$ such that a universal variable conclusion $C^{\mathrm{uv}} = \{E^{\mathrm{uv}}_1, \ldots, E^{\mathrm{uv}}_k\}$ can be derived from the premiss Π^{uv} where $E_j^{\mathrm{gd}} \subset \mathit{Inst}(E_j^{\mathrm{uv}})$ $(1 \leq j \leq k)$. Let the tableau T_{i+1}^{uv} be constructed from the tableau T_i^{uv} by extending all such branches using the premiss Π^{uv} and the conclusion C^{uv}.

Now, let B_{i+1}^{uv} be an arbitrary branch in T_{i+1}^{uv}. The only interesting case is where $\mathit{Form}(B_{i+1}^{\mathrm{uv}}) = \mathit{Form}(B_i^{\mathrm{uv}}) \cup E_j^{\mathrm{uv}}$ for some $j \in \{1, \ldots, k\}$. But then the branch B_{i+1}^{gd} in T_{i+1}^{gd} that has been constructed by extending B_i^{gd} with E_j^{gd} satisfies the condition in the induction hypothesis, because $\mathit{Form}(B_i^{\mathrm{gd}}) \subset \mathit{Inst}(\mathit{Form}(B_i^{\mathrm{uv}}))$ and, therefore,

$$\begin{array}{rcl} \mathit{Form}(B_{i+1}^{\mathrm{gd}}) & = & \mathit{Form}(B_i^{\mathrm{gd}}) \cup E_j^{\mathrm{gd}} \\ & \subset & \mathit{Inst}(\mathit{Form}(B_i^{\mathrm{uv}})) \cup \mathit{Inst}(E_j^{\mathrm{uv}}) \\ & = & \mathit{Inst}(\mathit{Form}(B_i^{\mathrm{uv}}) \cup E_j^{\mathrm{uv}}) \\ & = & \mathit{Inst}(\mathit{Form}(B_{i+1}^{\mathrm{uv}})) \ . \end{array}$$

Only-if part: Assume that $T_1^{\mathrm{uv}}, \ldots, T_n^{\mathrm{uv}}$ $(n \geq 1)$ is a tableau proof for G constructed using the universal variable calculus $\mathcal{C}^{\mathrm{uv}}$.

By induction on i, starting from $i = n$, we prove the following:

Induction hypothesis: For each branch B_i^{uv} in T_i^{uv} there is a set $\Phi^{\mathrm{gd}} \subset \mathit{Inst}(B_i^{\mathrm{uv}})$ of ground tableau formulae such that every ground tableau branch B^{gd} containing these formulae can be expanded to a closed (sub-)tableau.

The use a kind of "backward" induction, starting from $i = n$, reflects the fact that one first has to know which instances of universal variable formulae are needed as premisses to derive the leaves of the ground tableau, which then allows to compute the instances needed in the premisses of the premisses, etc.

Once the induction hypothesis has been proven to hold for $i = 1$, we can conclude that the single branch B_1^{gd} of the initial ground tableau can be extended to a closed ground tableau, as $\mathit{Form}(B_1^{\mathrm{gd}}) = \mathit{Form}(B_1^{\mathrm{uv}}) = \mathit{Inst}(\mathit{Form}(B_1^{\mathrm{uv}}))$ where B_1^{uv} is the single branch of the initial universal variable tableau T_1^{uv}.

$i = n$: As T_n^{uv} is closed, each of its branches B_n^{uv} contains $\bot$; therefore, the set Φ^{gd} can be chosen to be $\{\bot\}$. Every ground tableau branch containing $\bot$ trivially can be extended to a closed sub-tableau.

$i+1 \to i$: Let B_i^{uv} be an arbitrary branch of T_i^{uv}. If B_i^{uv} is a branch of T_{i+1}^{uv} as well, we are done. Otherwise, T_{i+1}^{uv} has been constructed from T_i^{uv} extending the branch B_i^{uv} using some premiss $\Pi^{\mathrm{uv}} \subset \mathit{Form}(B_i^{\mathrm{uv}})$ and a conclusion $C^{\mathrm{uv}} = \{E_1^{\mathrm{uv}}, \ldots, E_k^{\mathrm{uv}}\}$.

Let $\Phi_{i+1,j}^{\mathrm{gd}} \subset \mathit{Form}(B_{i+1,j}^{\mathrm{uv}})$ be the set of ground tableau formulae that exists according to the induction hypothesis for the branch $B_{i+1,j}^{\mathrm{uv}}$ of T_{i+1}^{uv} that has been constructed by expanding B_i^{uv} by E_j^{uv} $(1 \leq j \leq k)$. Further, for all $j \in \{1, \ldots, k\}$, let $\tau_1^j, \ldots, \tau_{l_j}^j$ be substitutions grounding for E_j^{uv} such that all tableau formulae in $\mathit{Inst}(E_j^{\mathrm{uv}})$ that occur in $\Phi_{i+1,j}^{\mathrm{gd}}$ are in $\bigcup_{r=1}^{l_j} E_j^{\mathrm{uv}}\tau_r^j$.

Choose $\Phi_i^{\mathrm{gd}} \subset \mathit{Inst}(\mathit{Form}(B_i^{\mathrm{uv}}))$ to be the smallest set such that:

1. containing all ground formulae in $\mathit{Inst}(\mathit{Form}(B_i^{\mathrm{uv}}))$ that occur in any of the sets $\Phi_{i+1,j}^{\mathrm{gd}}$ ($1 \leq j \leq k$), and
2. containing all minimal premisses $\Pi^{\mathrm{gd}} \subset \mathit{Inst}(\mathit{Form}(B_i^{\mathrm{uv}}))$ as subsets that are necessary to derive the conclusions $C^{\mathrm{gd}} = \{E_1^{\mathrm{uv}}\tau_{r_1}^1, \ldots, E_k^{\mathrm{uv}}\tau_{r_k}^1\}$ where $r_j \in \{1, \ldots, l_j\}$ for $1 \leq j \leq k$. Such premisses Π^{gd} exist as subsets of $\mathit{Inst}(\mathit{Form}(B_i^{\mathrm{uv}}))$ according to the definition of the relationship between a universal variable calculus and its ground version.

Now, every ground tableau branch B_i^{gd} such that $\Phi_i^{\mathrm{gd}} \subset \mathit{Form}(B_i^{\mathrm{gd}})$ can be extended to a closed sub-tableau as follows: Expand B_i^{gd} repeatedly using all the conclusions $C^{\mathrm{gd}} = \{E_1^{\mathrm{uv}}\tau_{r_1}^1, \ldots, E_k^{\mathrm{uv}}\tau_{r_k}^1\}$ in such a way that in the construction of each resulting new sub-branch each of these conclusions has been used once. There are $\prod_{j=1}^{k} l_j$ of these conclusions; thus, the number of new sub-branches is exponential in $\prod_{j=1}^{k} l_j$. The pigeon-hole principle implies that each of the new sub-branches contains for some $j \in \{1, \ldots .k\}$ the formulae of *all* the extensions $E_j^{\mathrm{uv}}\tau_r^j$ ($r \in \{1, \ldots, l_j\}$); as otherwise there would be a branch that does not contain some $E_j^{\mathrm{uv}}\tau_{r_j}^j$ for all $j \in \{1, \ldots, k\}$, in contradiction to the assumption that all the conclusions $\{E_1^{\mathrm{uv}}\tau_{r_1}^1, \ldots, E_k^{\mathrm{uv}}\tau_{r_k}^1\}$ have been used in the expansion of all new sub-branches.

We can conclude that each of the new sub-branches contains all tableau formulae in $\Phi_{i+1,j}^{\mathrm{gd}}$ for some $j \in \{1, \ldots, k\}$, because a formulae in

$$\Phi_{i+1,j}^{\mathrm{gd}} \subset \mathit{Inst}(\mathit{Form}(B_{i+1,j}^{\mathrm{uv}})) = \mathit{Inst}(\mathit{Form}(B_i^{\mathrm{uv}})) \cup \mathit{Inst}(E_j^{\mathrm{uv}})$$

is either an element of $\mathit{Inst}(\mathit{Form}(B_i^{\mathrm{uv}}))$, in which case it occurs in Φ_i^{gd} according to condition 1 in the definition of Φ_i^{gd} and thus on B_i^{gd}, or it is an element of $\mathit{Inst}(E_j^{\mathrm{uv}})$ and occurs, thus, by construction on the new sub-branch.

The induction hypothesis applies to each of the new sub-branches as they all contain the formulae of one of the sets $\Phi_{i+1,j}^{\mathrm{gd}}$, and they can all be extended to a closed sub-tableau. Therefore, the branch B_i^{gd} can be extended to a closed sub-tableau. $\square$

Corollary 4.3.6 *Let $\mathcal{C}^{\mathrm{uv}}$ be an ideal universal variable calculus for a logic* $\mathbf{L}$*; and let $\mathcal{C}^{\mathrm{gd}}$ be an ideal ground calculus for* $\mathbf{L}$ *such that $\mathcal{C}^{\mathrm{uv}}$ is a universal variable version of $\mathcal{C}^{\mathrm{gd}}$ (Def. 4.3.4).*

Then, $\mathcal{C}^{\mathrm{uv}}$ is sound and complete if and only if $\mathcal{C}^{\mathrm{gd}}$ is sound and complete.

Example 4.3.7 As the ground tableau proofs that are constructed from universal variable proofs are exponentially larger, we can only present a small example for one step in that construction.

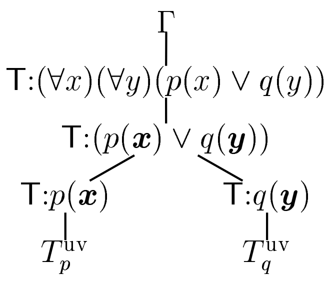

Γ
$\mathsf{T}{:}(\forall x)(\forall y)(p(x) \lor q(y))$
$\mathsf{T}{:}(p(\boldsymbol{x}) \lor q(\boldsymbol{y}))$
$\mathsf{T}{:}p(\boldsymbol{x})$
$\mathsf{T}{:}q(\boldsymbol{y})$
T_p^{uv}
T_q^{uv}

Figure 4.3: A universal variable tableau proof (see Example 4.3.7).

Consider the universal variable tableau proof shown in Figure 4.3 (it is constructed using the universal variable calculus $\mathcal{C}_{\mathrm{PL1}}^{\mathrm{uv}}$ for PL1 presented in Section 4.3.6); assume that the sub-tableaux T_p^{uv} and T_q^{uv} are closed (the set Γ of tableau formulae in the initial tableau might, for example, contain the formulae $\mathsf{T}{:}(\neg p(a) \lor \neg p(b))$ and $\mathsf{T}{:}(\neg q(c) \lor \neg q(d))$).

Assume further that the construction of a ground tableau proof (according to the proof of Theorem 4.3.5) has already been applied to all expansion steps necessary to generate the sub-tableaux T_p^{uv} and T_q^{uv}. We now consider the expansion step in which the universal variable conclusion $\{\{\mathsf{T}{:}p(\boldsymbol{x})\}, \{\mathsf{T}{:}p(\boldsymbol{y})\}\}$ is derived from the premiss $\{\mathsf{T}{:}(p(\boldsymbol{x}) \lor p(\boldsymbol{y}))\}$. Let B^{uv} be the single branch of the tableau before and B_p^{uv} and B_q^{uv} the two branches of the tableau after that expansion rule application. The construction of the ground tableau proof has already proceeded to the step where we have closed sub-tableaux T_p^{gd} and T_q^{gd} that can be generated by expanding any ground tableau branches containing certain instances of the formulae on B_p^{uv} and B_q^{uv}, respectively. Assume that $\Phi_p^{\mathrm{gd}} = \{\mathsf{T}{:}p(a), \mathsf{T}{:}p(b)\}$ and $\Phi_q^{\mathrm{gd}} = \{\mathsf{T}{:}q(c), \mathsf{T}{:}q(d)\}$ are these sets of instances.

Figure 4.4 shows the completed ground tableau proof. It is easy to check that on each of the new sub-branches either both $\mathsf{T}{:}p(a)$ and $\mathsf{T}{:}p(b)$ or both $\mathsf{T}{:}q(c)$ and $\mathsf{T}{:}q(d)$ occur.[2]. $\square$

For many logics, the same decrease in the length of shortest proofs that results from using the universal variable technique can be achieved using a non-analytic cut rule. In the above example, one could use the cut formula

$$G = (\forall x)(\forall y)(p(x) \lor q(y)) \to (\forall x)(p(x)) \lor (\forall y)(q(y)) \ .$$

Then, the resulting branch that contains $\mathsf{F}{:}G$ could be closed using a sub-tableau proof of constant size, whereas the branch containing $\mathsf{T}{:}G$ could be closed using a sub-tableau proof of the same size as the universal variable tableau proof. Thus, in some sense, the universal variable technique can be seen as a restricted application of the

[2] Note that both in the ground and the universal variable case, an expansion rule schema for γ-formulae is used that allows to derive $\mathsf{T}{:}(p(t) \lor q(t'))$ in one step from $\mathsf{T}{:}(\forall x)(\forall y)(p(x) \lor q(y))$ without deriving first an intermediate formula $\mathsf{T}{:}(\forall y)(p(t) \lor q(y))$

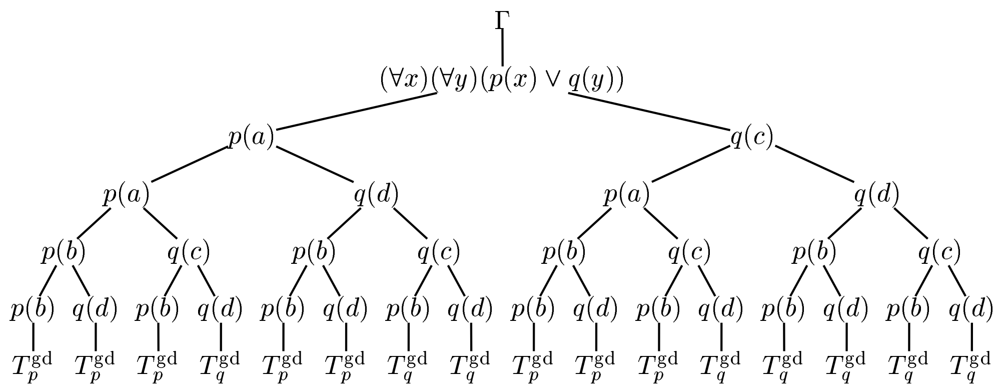

Γ
(∀x)(∀y)(p(x) ∨ q(y))
p(a)
q(c)
p(a) q(d) p(a) q(d)
p(b) q(c) p(b) q(c) p(b) q(c) p(b) q(c)
p(b) q(d) p(b) q(d) p(b) q(d) p(b) q(d) p(b) q(d) p(b) q(d) p(b) q(d) p(b) q(d)
T_p^{gd} T_p^{gd} T_p^{gd} T_q^{gd} T_p^{gd} T_p^{gd} T_q^{gd} T_q^{gd} T_p^{gd} T_p^{gd} T_p^{gd} T_q^{gd} T_q^{gd} T_q^{gd} T_q^{gd} T_q^{gd}

Figure 4.4: A ground tableau proof constructed from the universal variable tableau proof in Figure 4.3 (see Example 4.3.7); to enhance readability, the labels and truth-value signs are omitted (which are $*$ resp. T in all cases).

non-analytic cut rule (using the cut rule in an unrestricted way leads to an explosion in the size of the search space).

In (Stenz, 1997), a transformation of tableau proofs constructed using a universal variable calculus for PL1 into ground tableau proofs is described. That transformation, however, only works if universal variables do never occur in in a conclusion with more than one extension in the original proof, in which case a ground tableau proof can be constructed whose size is polynomial in the size of the universal variable proof.

4.3.3 Constructing a Universal Variable Calculus

In this section, we discuss the problem of how to construct a universal variable version $\mathcal{C}^{\mathrm{uv}}$ of a ground calculus $\mathcal{C}^{\mathrm{gd}}$.

One possibility is to turn an ideal rigid variable version $\mathcal{C}^{\mathrm{rv}}$ of $\mathcal{C}^{\mathrm{gd}}$ that has been constructed using the lifting technique described in Section 4.2 into a universal variable calculus. That can be done as follows: Given a universal variable premiss Π_{uv}, replace the universal variables in Π_{uv} by rigid variables to construct a rigid variable premiss Π_{rv}, where occurrences of a universal variable $\boldsymbol{x}$ in the same formula are replaced by the same rigid variable, but occurrences of $\boldsymbol{x}$ in different formulae are replaced by different rigid variables; and different universal variables are alway replaced by different rigid variables. Then, the set of possible universal variable conclusions for Π_{uv} consists of all C_{uv} that can be constructed from rigid variable conclusions $\langle C_{\mathrm{rv}}, \tau \rangle$ of Π_{rv} by

1. replacing each rigid variable that occurs in only one extension of C_{rv} by a universal variable, and

2. replacing each rigid variable that occurs in more than one extension of C_{rv} by a ground term $t \in \mathit{TabTerm}$.

All occurrences of a rigid variable are replaced by the same universal variable resp. the same term. Note that the substitution τ in the free variable conclusion does not play any rôle in the construction of the universal variable conclusion. The resulting universal variable calculus is by construction ideal.

Example 4.3.8 Table 4.4 shows examples for the most important phenomena that may occur when the above method for computing universal variable conclusions is used.

(a) A new universal variable is introduced.

(b) A variable is distributed over two extensions; it looses its universal power and has to replaced by ground terms.

(c) A variable occurs in only one extension and, therefore, remains universal.

(d) A combination of cases (b) and (c); the variable $\boldsymbol{x}$ has to be replaced by ground terms whereas the variable $\boldsymbol{y}$ remains universal.

(e) In the rigid variable version a substitution has to be applied to the tableau; in the universal variable version it is not applied as the variable that would have to be instantiated is universal.

(f) In this case, the method for constructing a universal variable conclusion does not lead to an optimal result. Each of the two universal variables is distributed over both extensions and they are therefore replaced by ground terms. However, the alternative schema

$$\frac{\mathsf{T}{:}p(\boldsymbol{x}) \leftrightarrow q(\boldsymbol{y})}{\begin{array}{c|c} \mathsf{T}{:}p(\boldsymbol{x}) & \mathsf{F}{:}p(\boldsymbol{x}) \\ \mathsf{T}{:}q(\boldsymbol{y}) & \mathsf{F}{:}q(\boldsymbol{y}) \end{array}}$$

for PL1-formulae of the form $p(\boldsymbol{x}) \leftrightarrow q(\boldsymbol{y})$ describes a sound expansion rule as well; the two veriables remain universal although they are distributed over two extensions. This schema is sound provided that the universal variables $\boldsymbol{x}$ and $\boldsymbol{y}$ each occur in only one subformula of the premiss.[3] $\square$

As Cases (a) and (b) in the above example demonstrate, in universal variable calculi, different variants (or instances) of a conclusion are generated by branching rule schemata and not by the schemata that introduce new variables as in rigid variable calculi.

For certain formula classes, using a universal variable calculus $\mathcal{C}^{\mathrm{uv}}$ constructed from a free variable calculus $\mathcal{C}^{\mathrm{fv}}$ can lead to proofs that are exponentially smaller than the shortest proofs built using $\mathcal{C}^{\mathrm{fv}}$ or its ground version $\mathcal{C}^{\mathrm{gd}}$. That notwithstanding, as

[3] Intuitively, this more liberal schema is sound because, if $p(t)$ and $q(t')$ are equivalent for all $t, t' \in \mathit{Term}$, then $p(t)$ and $q(t)$ are either both true for all $t \in \mathit{Term}$ or both false for all $t \in \mathit{Term}$.

$$\begin{array}{rc}
\Pi_{\mathrm{uv}} = & \mathsf{T}{:}(\forall x)(p(x,\boldsymbol{y})) \\ \cline{2-2}
\Pi_{\mathrm{rv}} = & \mathsf{T}{:}(\forall x)(p(x,Y)) \\ \cline{2-2}
\langle C_{\mathrm{rv}},\tau\rangle = & \langle \mathsf{T}{:}p(X,Y)\ ,\ id\rangle \\ \cline{2-2}
C_{\mathrm{uv}} = & \mathsf{T}{:}p(\boldsymbol{x},\boldsymbol{y})
\end{array}$$

(a)

$$\begin{array}{rc|c}
\Pi_{\mathrm{uv}} = & \multicolumn{2}{c}{\mathsf{T}{:}p(\boldsymbol{x}) \lor q(\boldsymbol{x})} \\ \cline{2-3}
\Pi_{\mathrm{rv}} = & \multicolumn{2}{c}{\mathsf{T}{:}p(X) \lor q(X)} \\ \cline{2-3}
\langle C_{\mathrm{rv}},\tau\rangle = & \langle \mathsf{T}{:}p(X) & \mathsf{T}{:}q(X)\ ,\ id\rangle \\ \cline{2-3}
C_{\mathrm{uv}} = & \mathsf{T}{:}p(t) & \mathsf{T}{:}q(t)
\end{array}$$

for all $t \in \mathit{Term}$

(b)

$$\begin{array}{rc}
\Pi_{\mathrm{uv}} = & \mathsf{T}{:}p(\boldsymbol{x}) \land q(\boldsymbol{x}) \\ \cline{2-2}
\Pi_{\mathrm{rv}} = & \mathsf{T}{:}p(X) \land q(X) \\ \cline{2-2}
\langle C_{\mathrm{rv}},\tau\rangle = & \left\langle \begin{array}{c} \mathsf{T}{:}p(X) \\ \mathsf{T}{:}q(X) \end{array} ,\ id\right\rangle \\ \cline{2-2}
C_{\mathrm{uv}} = & \begin{array}{c} \mathsf{T}{:}p(\boldsymbol{x}) \\ \mathsf{T}{:}q(\boldsymbol{x}) \end{array}
\end{array}$$

(c)

$$\begin{array}{rc|c}
\Pi_{\mathrm{uv}} = & \multicolumn{2}{c}{\mathsf{T}{:}p(\boldsymbol{x}) \lor q(\boldsymbol{x},\boldsymbol{y})} \\ \cline{2-3}
\Pi_{\mathrm{rv}} = & \multicolumn{2}{c}{\mathsf{T}{:}p(X) \lor q(X,Y)} \\ \cline{2-3}
\langle C_{\mathrm{rv}},\tau\rangle = & \langle \mathsf{T}{:}p(X) & \mathsf{T}{:}q(X,Y)\ ,\ id\rangle \\ \cline{2-3}
C_{\mathrm{uv}} = & \mathsf{T}{:}p(t) & \mathsf{T}{:}q(t,\boldsymbol{y})
\end{array}$$

for all $t \in \mathit{Term}$

(d)

$$\begin{array}{rc}
\Pi_{\mathrm{uv}} = & \begin{array}{c} \mathsf{T}{:}p(f(a)) \\ \mathsf{T}{:}f(\boldsymbol{x}) \approx \boldsymbol{x} \end{array} \\ \cline{2-2}
\Pi_{\mathrm{rv}} = & \begin{array}{c} \mathsf{T}{:}p(f(a)) \\ \mathsf{T}{:}f(X) \approx X \end{array} \\ \cline{2-2}
\langle C_{\mathrm{rv}},\tau\rangle = & \langle \mathsf{T}{:}p(a)\ ,\ \{X \mapsto a\}\rangle \\ \cline{2-2}
C_{\mathrm{uv}} = & \mathsf{T}{:}p(a)
\end{array}$$

(e)

$$\begin{array}{rc|c}
\Pi_{\mathrm{uv}} = & \multicolumn{2}{c}{\mathsf{T}{:}p(\boldsymbol{x}) \leftrightarrow q(\boldsymbol{y})} \\ \cline{2-3}
\Pi_{\mathrm{rv}} = & \multicolumn{2}{c}{\mathsf{T}{:}p(X) \leftrightarrow q(Y)} \\ \cline{2-3}
\langle C_{\mathrm{rv}},\tau\rangle = & \langle \begin{array}{c} \mathsf{T}{:}p(X) \\ \mathsf{T}{:}q(Y) \end{array} & \begin{array}{c} \mathsf{F}{:}p(X) \\ \mathsf{F}{:}q(Y) \end{array} ,\ id\rangle \\ \cline{2-3}
C_{\mathrm{uv}} = & \begin{array}{c} \mathsf{T}{:}p(t) \\ \mathsf{T}{:}q(t') \end{array} & \begin{array}{c} \mathsf{F}{:}p(t) \\ \mathsf{F}{:}q(t') \end{array}
\end{array}$$

for all $t, t' \in \mathit{Term}$

(f)

Table 4.4: Examples for the construction of universal variable conclusions (see Example 4.3.8).

Case (f) in Example 4.3.8 shows, this construction not always yields an optimal universal variable calculus. In fact, the question whether a universal variable occurring in the premiss can remain universal in the conclusion or has to be replaced by ground terms is undecidable in general (for the case of first-order predicate logic, this problem is discussed in (Beckert & Hähnle, 1998)).

4.3.4 Semantics of Universal Variable Tableaux

It is usually not possible to define model semantics for universal variable calculi if both truth value signs F and T can occur in tableaux. The reason is that, if one defines $\phi(\boldsymbol{x})$ to be true in $I(\sigma)$ if $\phi(t)$ is true in $I(\sigma)$ for all terms t, then $\phi(\boldsymbol{x})$ is false in $I(\sigma)$ if there is a single term t such that $\phi(t)$ is false in $I(\sigma)$, i.e., $\mathsf{F}{:}\sigma{:}\phi(\boldsymbol{x})$ is satisfied if there is a single term t such that $\mathsf{F}{:}\sigma{:}\phi(t)$ is satisfied, which is not the intended semantics of the universal variable $\boldsymbol{x}$.

Therefore, we define semantics tableaux of a universal variable calculus $\mathcal{C}^{\mathrm{uv}}$ based on the semantics of the tableaux of the ground version $\mathcal{C}^{\mathrm{gd}}$ of $\mathcal{C}^{\mathrm{uv}}$; where the relationship, however, is different from that between the semantics of *rigid* variable and ground tableaux.

Definition 4.3.9 Let $\mathcal{C}^{\mathrm{uv}}$ be an ideal universal variable calculus for a logic $\mathbf{L}$, and let $\mathcal{C}^{\mathrm{gd}}$ be an ideal ground calculus for $\mathbf{L}$ such that $\mathcal{C}^{\mathrm{uv}}$ is a universal variable version of $\mathcal{C}^{\mathrm{gd}}$.

A universal variable tableau formula $\phi \in \mathit{TabForm}(\Sigma^*_{\mathrm{uv}})$ is *satisfied* by a tableau interpretation $\langle \mathbf{m}, I\rangle \in \mathit{TabInterp}(\Sigma^*)$ of $\mathcal{C}^{\mathrm{gd}}$ iff all ground tableau formulae in $\mathit{Inst}(\{\phi\})$ are satisfied by $\langle \mathbf{m}, I\rangle$. □

As in the ground case, a branch B_{uv} of a universal variable tableau is satisfied by a tableau interpretation $\langle \mathbf{m}, I\rangle$ if it satisfies all formulae on B_{uv} (or, equivalently, if it satisfies all formulae in $\mathit{Inst}(B_{\mathrm{uv}})$); and, $\langle \mathbf{m}, I\rangle$ satisfies a universal variable tableau T_{uv} if it satisfies at least one branch of T_{uv}.

Lemma 4.3.10 *Let $\mathcal{C}^{\mathrm{uv}}$ be an ideal universal variable calculus for a logic $\mathbf{L}$, and let $\mathcal{C}^{\mathrm{gd}}$ be an ideal ground calculus for $\mathbf{L}$ such that $\mathcal{C}^{\mathrm{uv}}$ is a universal variable version of $\mathcal{C}^{\mathrm{gd}}$.*

The calculus $\mathcal{C}^{\mathrm{uv}}$ has the strong soundness of expansion property w.r.t. the tableau interpretations of $\mathcal{C}^{\mathrm{gd}}$ if and only if $\mathcal{C}^{\mathrm{gd}}$ has the strong soundness of expansion property (Property 2 in Def. 3.5.8).

Proof: The lemma follows trivially from the definitions of the soundness property and the relation between a universal variable calculus and its ground version. □

Theorem 4.3.11 *Let $\mathcal{C}^{\mathrm{uv}}$ be an ideal universal variable calculus for a logic* $\mathbf{L}$*, and let $\mathcal{C}^{\mathrm{gd}}$ be an ideal ground calculus for* $\mathbf{L}$ *such that $\mathcal{C}^{\mathrm{uv}}$ is a universal variable version of $\mathcal{C}^{\mathrm{gd}}$.*

If $\mathcal{C}^{\mathrm{uv}}$ has the soundness properties from Definition 3.5.8 (appropriateness of the set of tableau interpretations and strong soundness of expansion), then $\mathcal{C}^{\mathrm{uv}}$ is sound.

Proof: As $\mathcal{C}^{\mathrm{uv}}$ has Property 1 (strong appropriateness of the set of tableau interpretations) from Definition 3.5.8, then $\mathcal{C}^{\mathrm{gd}}$ has that property as well, because the initial tableaux for a set $\mathfrak{F}$ of formulae are the same in both calculi. Since $\mathcal{C}^{\mathrm{uv}}$ has Property 2 (strong soundness of expansion), $\mathcal{C}^{\mathrm{gd}}$ has that property as well (according to Lemma 4.3.10).

Thus, $\mathcal{C}^{\mathrm{gd}}$ is sound (Theorem 3.5.4), which implies that $\mathcal{C}^{\mathrm{uv}}$ is sound as well (Corollary 4.3.6). $\square$

4.3.5 Mixing Rigid and Universal Variables

A universal variable calculus can syntactically be considered to be a ground calculus (as it does not contain rigid variables) and can thus be lifted and enriched by introducing rigid variables. Then, both universal and free variables occur in tableaux. We call such a calculus *mixed* variable calculus.

The method from Section 4.3.3 for constructing a *universal* variable version of a ground calculus $\mathcal{C}^{\mathrm{gd}}$ from a rigid variable version $\mathcal{C}^{\mathrm{rv}}$ of $\mathcal{C}^{\mathrm{gd}}$ can be adapted such that it can be used to construct a mixed variable calculus $\mathcal{C}^{\mathrm{mv}}$ that is a rigid variable version of a universal variable version of $\mathcal{C}^{\mathrm{gd}}$ and thus uses both the universal and the rigid variable technique.

The construction starts in the same way as for building a pure universal variable calculus: Given a mixed variable premiss Π_{mv} (i.e., a premiss containing both rigid and universal variables), construct pure rigid variable premiss Π_{rv} by replacing the universal variables in Π_{mv} by rigid variables, where—as before—occurrences of a universal variable $\boldsymbol{x}$ in the same formula are replaced by the same rigid variable, but occurrences of $\boldsymbol{x}$ in different formulae are replaced by different rigid variables; and different universal variables are alway replaced by different rigid variables. In addition, we now assume that the new rigid variables that are introduced are different from the rigid variables that already occur in Π_{mv}. Then, the set of possible mixed variable conclusions for Π_{mv} consists of all C_{mv} that can be constructed from a rigid variable conclusion $\langle C_{\mathrm{rv}}, \tau \rangle$ of Π_{rv} by

1. replacing all rigid variables in C_{rv} that have been introduced as a replacement for a universal variable $\boldsymbol{x}$ and occur in only one extension of C_{rv} by the original variable $\boldsymbol{x}$,

2. replacing all rigid variables in C_{rv} that have been introduced as a replacement for a universal variable and occur in more than one extension of C_{rv} by an arbitrary rigid variable,

3. restricting the substitution τ to the rigid variables that occur in the original mixed variable premiss Π_{mv}.

The second step above may seem redundant; but it is needed because, for example, $\{\{\mathsf{T}{:}p(X)\}, \mathsf{T}{:}q(X)\}\}$ must be a conclusion of $\{\{\mathsf{T}{:}p(\boldsymbol{x}) \lor q(\boldsymbol{x})\}\}$ for *all* rigid variables X. Intuitively, expansion rule applications that destroy universality of a variable $\boldsymbol{x}$ must allow the deduction of an arbitrary number of variants of the conclusion containing different rigid variables X instead of the universal variable $\boldsymbol{x}$.

Example 4.3.12 Tables 4.5 and 4.6 show examples for mixed variable premisses and the mixed variable conclusions that are computed for these premisses using the method described above (cf. Table 4.4 and Example 4.3.8 where similar examples are used to demonstrate the construction of (pure) universal variable conclusions).

(a) A new universal variable $\boldsymbol{x}$ is introduced. The rigid variable Z is not affected.

(b) The universal variable $\boldsymbol{x}$ is distributed over two extensions; it looses its universal power and has to replaced by an arbitrary rigid variable X. The rigid variable Y is not affected.

(c) The universal variable $\boldsymbol{x}$ occurs in only one extension and, therefore, remains universal. The rigid variable Y is not affected.

(d) A combination of cases (b) and (c); the universal variable $\boldsymbol{x}$ has to be replaced by an arbitrary rigid variable X whereas the variable $\boldsymbol{y}$ remains universal. The rigid variable Z is not affected.

(e) In the rigid variable version, a substitution has to be applied to the tableau; one of the instantiated variables (the rigid variable X) has been introduced as a replacement for the universal variable $\boldsymbol{x}$, the other instantiated variable (the rigid variable Y) already occurred in the mixed variable premiss. In the mixed variable conclusion only the rigid variable Y that occurred in the premiss is instantiated.

(f) This example demonstrates that a universal variable $\boldsymbol{x}$ looses its universal power and has to be replaced by an arbitrary rigid variable X if it occurs in a term in the range of the substitution that is part of the free variable conclusion. □

As shown in Case (f) in the above example, a universal variable has to be replaced by a rigid variable if it occurs in the range of a substitution that is applied to the tableau. Consequently, if one has the choice to either instantiate $\boldsymbol{x}$ with Y or instantiate Y with (a rigid replacement) for $\boldsymbol{x}$, it is better to choose the former possibility.

$$\begin{array}{rc}
\Pi_{\mathrm{mv}} = & \mathsf{T}{:}(\forall x)(p(x,\boldsymbol{y},Z)) \\ \cline{2-2}
\Pi_{\mathrm{rv}} = & \mathsf{T}{:}(\forall x)(p(x,Y,Z)) \\ \cline{2-2}
\langle C_{\mathrm{rv}},\tau\rangle = & \langle \mathsf{T}{:}p(X,Y,Z)\ ,\ id\rangle \\ \cline{2-2}
\langle C_{\mathrm{mv}},\tau\rangle = & \langle \mathsf{T}{:}p(\boldsymbol{x},\boldsymbol{y},Z)\ ,\ id\rangle
\end{array}$$

(a)

$$\begin{array}{rc}
\Pi_{\mathrm{mv}} = & \mathsf{T}{:}p(\boldsymbol{x},Y) \lor q(\boldsymbol{x},Y) \\ \cline{2-2}
\Pi_{\mathrm{rv}} = & \mathsf{T}{:}p(X,Y) \lor q(X,Y) \\ \cline{2-2}
\langle C_{\mathrm{rv}},\tau\rangle = & \langle \mathsf{T}{:}p(X,Y) \mid \mathsf{T}{:}q(X,Y)\ ,\ id\rangle \\ \cline{2-2}
\langle C_{\mathrm{mv}},\tau\rangle = & \langle \mathsf{T}{:}p(X,Y) \mid \mathsf{T}{:}q(X,Y)\ ,\ id\rangle \\
 & \text{for all } X \in \mathit{Var}
\end{array}$$

(b)

$$\begin{array}{rc}
\Pi_{\mathrm{mv}} = & \mathsf{T}{:}p(\boldsymbol{x},Y) \land q(\boldsymbol{x},Y) \\ \cline{2-2}
\Pi_{\mathrm{rv}} = & \mathsf{T}{:}p(X,Y) \land q(X,Y) \\ \cline{2-2}
\langle C_{\mathrm{rv}},\tau\rangle = & \langle \begin{array}{l}\mathsf{T}{:}p(X,Y)\\ \mathsf{T}{:}q(X,Y)\end{array}\ ,\ id\rangle \\ \cline{2-2}
\langle C_{\mathrm{mv}},\tau\rangle = & \langle \begin{array}{l}\mathsf{T}{:}p(\boldsymbol{x},Y)\\ \mathsf{T}{:}q(\boldsymbol{x},Y)\end{array}\ ,\ id\rangle
\end{array}$$

(c)

$$\begin{array}{rc}
\Pi_{\mathrm{mv}} = & \mathsf{T}{:}p(\boldsymbol{x},Z) \lor q(\boldsymbol{x},\boldsymbol{y},Z) \\ \cline{2-2}
\Pi_{\mathrm{rv}} = & \mathsf{T}{:}p(X,Z) \lor q(X,Y,Z) \\ \cline{2-2}
\langle C_{\mathrm{rv}},\tau\rangle = & \langle \mathsf{T}{:}p(X,Z) \mid \mathsf{T}{:}q(X,Y,Z)\ ,\ id\rangle \\ \cline{2-2}
\langle C_{\mathrm{mv}},\tau\rangle = & \langle \mathsf{T}{:}p(X,Z) \mid \mathsf{T}{:}q(X,\boldsymbol{y},Z)\ ,\ id\rangle \\
 & \text{for all } X \in \mathit{Var}
\end{array}$$

(d)

Table 4.5: Examples for the construction of mixed variable conclusions (first part), see see Example 4.3.12.

$$\begin{array}{rc}
\Pi_{\mathrm{mv}} = & \begin{array}{c}\mathsf{T}{:}p(f(a,b)) \\ \mathsf{T}{:}f(\boldsymbol{x},Y) \approx \boldsymbol{x}\end{array} \\ \hline
\Pi_{\mathrm{rv}} = & \begin{array}{c}\mathsf{T}{:}p(f(a,b)) \\ \mathsf{T}{:}f(X,Y) \approx X\end{array} \\ \hline
\langle C_{\mathrm{rv}}, \tau\rangle = & \langle \mathsf{T}{:}p(a)\ ,\ \{X \mapsto a, Y \mapsto b\}\rangle \\ \hline
\langle C_{\mathrm{mv}}, \tau\rangle = & \langle \mathsf{T}{:}p(a)\ ,\ \{Y \mapsto b\}\rangle
\end{array}$$

(e)

$$\begin{array}{rc}
\Pi_{\mathrm{mv}} = & \begin{array}{c}\mathsf{T}{:}p(f(\boldsymbol{x})) \\ \mathsf{F}{:}p(Y)\end{array} \\ \hline
\Pi_{\mathrm{rv}} = & \begin{array}{c}\mathsf{T}{:}p(f(X)) \\ \mathsf{F}{:}p(Y)\end{array} \\ \hline
\langle C_{\mathrm{rv}}, \tau\rangle = & \langle \bot\ ,\ \{Y \mapsto f(X)\}\rangle \\ \hline
\langle C_{\mathrm{mv}}, \tau\rangle = & \langle \bot\ ,\ \{Y \mapsto f(X)\}\rangle \\
 & \text{for all } X \in \mathit{Var}
\end{array}$$

(f)

Table 4.6: Examples for the construction of mixed variable conclusions (second part), see Example 4.3.12.

In the design of mixed variable calculi, a generalised notion of unification, which we call *uv-unification*, plays an important rôle. In the rigid variable as well as in the universal variable case, standard unification as defined in Section 2.2.3 is sufficient (the difference is that in the universal variable case unifiers are not applied to the tableau). In the mixed variable case, however, both types of variables can occur simultaneously, and uv-unification has to be used, which takes the different nature of both types of variables into account.

Definition 4.3.13 Let ϕ and ψ be tableau formulae over a signature Σ^*_{mv} containing both rigid and universal variables; and let

$$\pi = \{\boldsymbol{x}_1 \mapsto X_1, \ldots, \boldsymbol{x}_k \mapsto X_k\} \text{ and } \rho = \{\boldsymbol{y}_1 \mapsto Y_1, \ldots, \boldsymbol{y}_k \mapsto Y_l\}$$

be substitutions that replace all universal variables in ϕ resp. ψ by new rigid variables, i.e.,

1. $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ are all the universal variables occurring in ϕ, and $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_l$ are all the universal variables occurring in ψ,

2. $X_1, \ldots, X_k, Y_1, \ldots, Y_l$ are pairwise distinct rigid variables neither occurring in ϕ nor in ψ.

A substitution $\mu \in \mathit{Subst}(\Sigma^*_{\mathrm{mv}})$ is a *universal variable unifier (uv-unifier)* of ϕ and ψ if it is the restriction of a unifier of $\phi\pi$ and $\psi\rho$ to the rigid variables occurring in ϕ and/or ψ. □

Note that both the domain and the range of a uv-unifier contain only rigid and no universal variables.

Example 4.3.14 The substitution $\{Y \mapsto b\}$ is a uv-unifier of the tableau formulae $\mathsf{T}{:}p(\boldsymbol{x}, Y)$ and $\mathsf{T}{:}p(a, b)$; the empty substitution id is a uv-unifier of $\mathsf{T}{:}p(\boldsymbol{x})$ and $\mathsf{T}{:}p(f(\boldsymbol{x}))$; and $\{X \mapsto f(Y)\}$ is a uv-unifier of $\mathsf{T}{:}p(X)$ and $\mathsf{T}{:}p(f(\boldsymbol{y}))$. □

The definitions of the semantics of rigid and universal variable tableaux (Def. 4.2.23 and Def. 4.3.9) can easily be combined to define a semantics for mixed variable tableaux:

Definition 4.3.15 Let $\mathcal{C}^{\mathrm{mv}}$ be an ideal mixed variable calculus for a logic $\mathbf{L}$; and let $\mathcal{C}^{\mathrm{gd}}$ be an ideal ground calculus for $\mathbf{L}$ such that $\mathcal{C}^{\mathrm{mv}}$ is a rigid variable version of a universal variable version of $\mathcal{C}^{\mathrm{gd}}$.

A mixed variable tableau T^{mv} is satisfied by a tableau interpretation $\langle \mathbf{m}, I\rangle$ of the ground calculus $\mathcal{C}^{\mathrm{gd}}$ iff, for all substitution $\tau \in \mathit{Subst}(\Sigma^*_{\mathrm{rv}})$ such that $T^{\mathrm{mv}}\tau$ does not contain rigid variables, the tableau $T^{\mathrm{mv}}\tau$ is satisfied by $\langle \mathbf{m}, I\rangle$ according to Def. 4.3.9 (satisfiability of universal variable tableaux), i.e., if there is a branch B of $T^{\mathrm{mv}}\tau$ such that $\langle \mathbf{m}, I\rangle$ satisfies all (ground) tableau formulae in $\mathit{Inst}(\mathit{Form}(B))$. □

4.3.6 An Ideal Mixed Variable Calculus for PL1

In this section, we define a mixed variable calculus $\mathcal{C}^{\mathrm{mv}}_{\mathrm{PL1}}$ for PL1. It is constructed from the pure rigid variable calculus $\mathcal{C}^{\mathrm{rv}}_{\mathrm{PL1}}$ described in Section 4.2.10 using the method for computing mixed variable conclusions from Section 4.3.5.

For each first-order signature Σ (see Section 2.3), the extended signatures Σ^*_{mv} now contains the rigid variables from Var and the universal variables from UVar as constants (see Section 2.3) and, in addition, the set $F^{sko}(\Sigma)$ of Skolem function symbols containing infinitely many symbols of each arity $n \geq 0$. All free (rigid and universal) variables are of the same sort.

The set of labels and the initial label of $\mathcal{C}^{\mathrm{mv}}$ are the same as that of $\mathcal{C}^{\mathrm{gd}}$ and $\mathcal{C}^{\mathrm{rv}}$, i.e., the label $*$ represents the single world of PL1-models.

As before, unifying notation is used to describe the expansion rule of $\mathcal{C}^{\mathrm{mv}}$, and the set of tableau formulae divided into α-, β-, γ-, and δ-formulae according to Table 3.1.

As said above, the expansion rule schemata of $\mathcal{C}^{\mathrm{mv}}$ are constructed from the schemata of $\mathcal{C}^{\mathrm{rv}}$ using the method described in the previous section. The schemata of $\mathcal{C}^{\mathrm{mv}}$

$$\frac{\alpha}{\begin{array}{c}\alpha_1\\ \alpha_2\end{array}} \qquad \frac{\beta}{\beta_1\rho \mid \beta_2\rho}$$

for all substitutions $\rho = \{\boldsymbol{x}_1 \mapsto X_1, \ldots, \boldsymbol{x}_k \mapsto X_k\}$
where $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ are the universal variables occurring
in both β_1 and β_2 and $\{X_1, \ldots, X_k\}$ are rigid variables

$$\frac{\gamma(x)}{\gamma_1(\boldsymbol{x})} \qquad \frac{\delta(x)}{\delta_1(t)}$$

for some universal variable $\boldsymbol{x}$ not occurring in γ

where $t = sko_{\mathrm{fv}}(\delta)$ (see Def. 4.2.28)

$$\frac{\begin{array}{c}\phi\\ \overline{\psi}\end{array}}{\bot}$$

where ϕ and ψ are unifiable atomic formulae; and
a most general uv-unifier of ϕ, ψ is applied to the tableau

Table 4.7: Rule schemata for first-order predicate logic using both universal and rigid variables.

differ notably in two ways from the corresponding schemata of the rigid variable calculus $\mathcal{C}^{\mathrm{rv}}$: First. it is not the schema for γ-formulae any more that introduces variants of a formulae with different rigid variables but the schema for β-formulae. Second, the schema for closing branches applies substitutions only requires that *rigid* variables are instaniated (and no universal variables); that is, it applies a most general uv-unifier (Def. 4.3.13) of the complementary atoms.

The same Skolem terms are used as in the rigid variable case (Def. 4.2.28); however, all free variables, i.e., both universal and rigid variables, are made arguments of the Skolem term.

In Table 4.7, the expansion rule for premisses that contain both rigid and universal variables is given schematically for the different formula types.

The following is the formal definition of the expansion rule $\mathcal{E}^{\mathrm{mv}}_{\mathrm{PL1}}$ of the mixed variable calculus $\mathcal{C}^{\mathrm{mv}}_{\mathrm{PL1}}$.

Definition 4.3.16 The expansion rule $\mathcal{E}^{\mathrm{mv}}_{\mathrm{PL1}}$ of $\mathcal{C}^{\mathrm{mv}}_{\mathrm{PL1}}$ is, for all signatures $\Sigma \in Sig_{\mathrm{PL1}}$ and all premisses $\Pi \subset TabForm_{\mathrm{PL1}}(\Sigma^*_{\mathrm{mv}})$, defined by: the set $\mathcal{E}(\Sigma)^{\mathrm{mv}}_{\mathrm{PL1}}(\Pi)$ of possible conclusions is the smallest set containing the following mixed variable conclusions:

– $\{\langle\{\alpha_1,\alpha_2\}, id\rangle\}$	for all $\alpha \in \Pi$,
– $\{\langle\{\beta_1\rho\},\{\beta_2\rho\}, id\rangle\}$	for all $\beta \in \Pi$ and all $\rho = \{\boldsymbol{x}_1 \mapsto X_1, \dots, \boldsymbol{x}_k \mapsto X_k\}$ where $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k$ are the universal variables occurring in both β_1 and β_2 and $\{X_1, \dots, X_k\}$ are rigid variables,
– $\{\langle\{\gamma_1(\boldsymbol{x})\}, id\rangle\}$	for all $\gamma \in \Pi$ where $\boldsymbol{x}$ is a universal variable not occurring in γ,
– $\{\langle\{\delta(t)\}, id\rangle\}$	for all $\delta \in \Pi$ where $t = sko_{\mathrm{fv}}(\delta)$ (Def. 4.2.28),
– $\{\langle\{\bot\}, \mu\rangle\}$	if $\mathsf{T}{:}\sigma{:}G, \mathsf{F}{:}\sigma{:}G' \in \Pi$ such that $G, G' \in Atom_{\mathrm{PL1}}(\Sigma^*_{\mathrm{mv}})$ are unifiable atoms; μ is a uv-unifier of G and G'.

□

Theorem 4.3.17 *The calculus $\mathcal{C}^{\mathrm{mv}}_{\mathrm{PL1}}$ is a rigid variable version of a universal variable version of the calculus $\mathcal{C}_{\mathrm{PL1}}$ defined in Section 3.6.*

Corollary 4.3.18 *The calculus $\mathcal{C}^{\mathrm{mv}}_{\mathrm{PL1}}$ is sound and complete.*

4.3.7 An Ideal Mixed Variable Calculus for the Modal Logic K

In this section, we define a mixed variable version $\mathcal{C}^{\mathrm{mv}}_{\mathrm{K}}$ of the calculus $\mathcal{C}^{\mathrm{con}}_{\mathrm{K}}$ for the modal logic K from Section 3.7.4, i.e., a calculus with continuous expansion rule schema for ν-formulae. As in the rigid variable version, free variables are introduced into labels (and not in the formula part of tableau formulae). The set of labels of the mixed variable calculus is

$$Lab_{\mathrm{mv}} = CondLab(\mathbb{N} \cup Var \cup UVar)$$

with the initial label 1.

The relationship between the mixed variable calculus $\mathcal{C}^{\mathrm{mv}}_{\mathrm{K}}$ and the rigid variable calculus $\mathcal{C}^{\mathrm{rv}}_{\mathrm{K}}$ is similar to that between the mixed variable calculus $\mathcal{C}^{\mathrm{mv}}_{\mathrm{PL1}}$ and the rigid variable calculus $\mathcal{C}^{\mathrm{rv}}_{\mathrm{PL1}}$ for PL1: In $\mathcal{C}^{\mathrm{rv}}_{\mathrm{PL1}}$, variants of a formula are generated by applying the expansion rule to γ-formulae, whereas in $\mathcal{C}^{\mathrm{mv}}_{\mathrm{PL1}}$ such rule application introduce universal variables, and variants are generated by expansion rule applications to β-formulae that distribute universal variables over different branches. Similarly, in $\mathcal{C}^{\mathrm{rv}}_{\mathrm{K}}$, variants of tableau formulae are generated by applying the expansion rule to ν-formulae, whereas in $\mathcal{C}^{\mathrm{mv}}_{\mathrm{K}}$ universal variables are introduced by rule applications to ν-formulae, and variants are generated by expansion rule applications to β-formulae when universal variables are distributed over different branches and loose their universal power.

In the rigid variable calculus $\mathcal{C}^{\mathrm{rv}}_{\mathrm{K}}$, because of the justification test, all variables occurring in a pair of complementary atoms are instantiated when the pair is used to close a branch. For example, a pair $\mathsf{T}{:}1.(X){:}p$, $\mathsf{F}{:}1.(Y){:}p$ can only be used to close a branch if formulae such as $\mathsf{T}{:}1.1{:}q$ are available to justify the labels of the complementary pair.

$$\frac{\alpha}{\begin{array}{c}\alpha_1\\ \alpha_2\end{array}} \qquad \frac{\beta}{\beta_1\rho \mid \beta_2\rho}$$

for all substitutions $\rho = \{\boldsymbol{x}_1 \mapsto X_1, \dots, \boldsymbol{x}_k \mapsto X_k\}$
where $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k$ are the universal variables occurring
in both β_1 and β_2 and $\{X_1, \dots, X_k\}$ are rigid variables

$$\frac{\mathsf{T}{:}\sigma{:}\Box G}{\mathsf{T}{:}\sigma.(\boldsymbol{x}){:}G} \qquad \frac{\mathsf{F}{:}\sigma{:}\Diamond G}{\mathsf{F}{:}\sigma.(\boldsymbol{x}){:}G} \qquad \frac{\mathsf{T}{:}\sigma{:}\Diamond G}{\mathsf{T}{:}\sigma.n{:}G} \qquad \frac{\mathsf{F}{:}\sigma{:}\Box G}{\mathsf{F}{:}\sigma.n{:}G}$$

for some $\boldsymbol{x} \in \mathit{UVar}$ not in σ where $n = \lceil G \rceil$ where $n = \lceil \neg G \rceil$

$$\frac{\mathsf{T}{:}\sigma{:}\neg G}{\mathsf{F}{:}\sigma{:}G} \qquad \frac{\mathsf{F}{:}\sigma{:}\neg G}{\mathsf{T}{:}\sigma{:}G}$$

$$\frac{\begin{array}{c}\mathsf{T}{:}\sigma{:}G\\ \mathsf{F}{:}\sigma'{:}G\end{array}}{\bot}$$

if there are a substitution $\mu \in \mathit{Subst}_{\mathrm{rv}}$ and substitutions $\rho, \rho' \in \mathit{Subst}_{\mathrm{uv}}$
such that $[\sigma\rho\mu] = [\sigma'\rho'\mu]$ and
the labels $\sigma\rho\mu$ and $\sigma'\rho'\mu$ are justified by formulae in $\mathit{Inst}(B\mu)$
where B is the branch being expanded; a most general such substitution μ is to be applied to the tablea

Table 4.8: Expansion rule schemata for the calculus $\mathcal{C}_{\mathrm{K}}^{\mathrm{mv}}$.

Thus, the substitution $\{X \mapsto 1, Y \mapsto 1\}$ has to be applied in that case. If, however, complementary atoms containing universal variables are used, such as $\mathsf{T}{:}1.(\boldsymbol{x}){:}p$ and $\mathsf{F}{:}1.(\boldsymbol{y}){:}p$, then one still has to check that there are formulae on the branch that justify an instance $1.(n)$ of $1.(\boldsymbol{x})$ and $1.(\boldsymbol{y})$, but it is not necessary to actually instantiate the universal variables $\boldsymbol{x}$ and $\boldsymbol{y}$ to close the branch.

In contrast to the expansion rule of the mixed variable calculus $\mathcal{C}_{\mathrm{PL1}}^{\mathrm{mv}}$ for PL1, the expansion rule of the mixed variable calculus $\mathcal{C}_{\mathrm{K}}^{\mathrm{mv}}$ for K is not defined using the notion of uv-unification, because the free variables in labels are not instantiated with complex terms but only with other free variables or with natural numbers. Mixed variable labels are unifiable if there are substitutions instantiating the universal variables they contain, respectively, and another common substitution instantiating their rigid variables, such that the combination of these substitutions is a unifier of the labels. The substitutions instantiation universal variables and the substitution instantiation rigid variables cannot influence each other, and only the rigid variable substitution is applied to the tableau.

The expansion rule $\mathcal{E}_{\mathrm{K}}^{\mathrm{mv}}$ of $\mathcal{C}_{\mathrm{K}}^{\mathrm{mv}}$ is shown schematically in Table 4.8. Formally it is defined as follows:

Definition 4.3.19 The expansion rule $\mathcal{E}_{\mathrm{K}}^{\mathrm{mv}}$ of $\mathcal{C}_{\mathrm{K}}^{\mathrm{mv}}$ is, for all signatures $\Sigma \in \mathit{Sig}_{\mathrm{mod}}$ and all premisses $\Pi \subset \mathit{TabForm}_{\mathrm{mod}}(\Sigma)$, defined by: the set $\mathcal{E}_{\mathrm{K}}^{\mathrm{rv}}(\Sigma)(\Pi)$ is the smallest set containing the following conclusions (where $\lceil\cdot\rceil$ is any bijection from $\mathit{Form}_{\mathrm{mod}}(\Sigma)$ to the set of natural numbers):

– $\langle\{\{\alpha_1, \alpha_2\}\}$	for all $\alpha \in \Pi$,
– $\langle\{\{\beta_1\rho\}, \{\beta_2\rho\}\}, id\rangle$	for all $\beta \in \Pi$ and all $\rho = \{\boldsymbol{x}_1 \mapsto X_1, \ldots, \boldsymbol{x}_k \mapsto X_k\}$ where $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ are the universal variables occurring in both β_1 and β_2 and $\{X_1, \ldots, X_k\}$ are rigid variables,
– $\langle\{\{\mathsf{T}{:}\sigma.(\boldsymbol{x}){:}G\}\}, id\rangle$	for all $\mathsf{T}{:}\sigma{:}\Box G \in \Pi$ where $\boldsymbol{x} \in \mathit{UVar}$ is a universal variable not occurring in σ,
– $\langle\{\{\mathsf{F}{:}\sigma.(\boldsymbol{x}){:}G\}\}, id\rangle$	for all $\mathsf{F}{:}\sigma{:}\Diamond G \in \Pi$ where $\boldsymbol{x} \in \mathit{UVar}$ is a universal variable not occurring in σ,
– $\langle\{\{\mathsf{F}{:}\sigma.n{:}G\}\}, id\rangle$	for all $\mathsf{F}{:}\sigma{:}\Box G \in \Pi$ where $n = \lceil\neg G\rceil$,
– $\langle\{\{\mathsf{T}{:}\sigma.n{:}G\}\}, id\rangle$	for all $\mathsf{T}{:}\sigma{:}\Diamond G \in \Pi$ where $n = \lceil G\rceil$,
– $\langle\{\{\bot\}\}, \mu\rangle$	if $\mathsf{T}{:}\sigma{:}G, \mathsf{F}{:}\sigma'{:}G \in \Pi$ and μ is a most general substitution in $\mathit{Subst}_{\mathrm{rv}}$ for which there are substitutions $\rho, \rho' \in \mathit{Subst}_{\mathrm{uv}}$ such that $[\sigma\rho\mu] = [\sigma'\rho'\mu]$ and the labels $\sigma\rho\mu$ and $\sigma'\rho'\mu$ are justified by $\Pi\mu$.

□

Theorem 4.3.20 *The calculus $\mathcal{C}_{\mathrm{K}}^{\mathrm{mv}}$ is a rigid variable version of a universal variable version of the calculus $\mathcal{C}_{\mathrm{K}}$ defined in Section 3.7.4.*

Corollary 4.3.21 *The calculus $\mathcal{C}_{\mathrm{K}}^{\mathrm{mv}}$ is sound and complete.*

Example 4.3.22 We continue from Examples 3.7.22 and 4.2.34 and again prove the formula

$$G = \Box(\neg p \vee q) \wedge \Box p \wedge (\Diamond\neg q \vee \Diamond\neg p)$$

to be K-unsatisfiable, now using the mixed variable calculus $\mathcal{C}_{\mathrm{K}}^{\mathrm{mv}}$ defined above. A closed mixed variable tableau $T_{\mathrm{mv}}^{\mathrm{con}}$ for G is shown in Figure 4.5.[4]

When the expansion rule is applied to the ν-formulae 2 and 4 to add formulae 6 resp. 7 to the tableau, the universal variables $\boldsymbol{x}_1$ resp. $\boldsymbol{x}_2$ are introduced (instead of introducing rigid variables).

When the expansion rule is applied to formula 6, and formulae 12 and 13 are added to the tableau, the universal variable $\boldsymbol{x}$ in the premiss loses its universal power as it is distributed over both extensions; in 12 and 13 it is replaced by the rigid variable X_1.

In the rigid variable version of the proof, the substitution $\{X_1 \mapsto 1, X_2 \mapsto 1\}$ has to be applied to close the left branch of the tableau. Now, however, the tableau contains the

[4] The figure shows the tableau formulae with uninstantiated rigid variables; the substitution $\{X_1 \mapsto 1\}$ that has to be applied during the construction of the proof is listed separately.

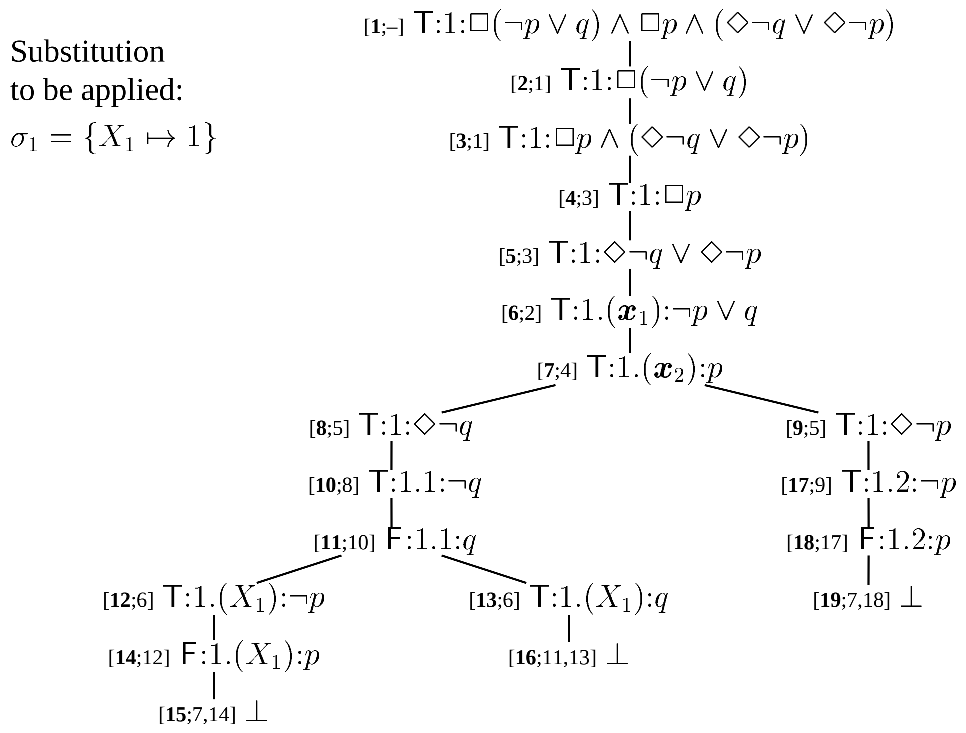

Substitution to be applied: $\sigma_1 = \{X_1 \mapsto 1\}$
[1;–] T:1:$\Box(\neg p \vee q) \wedge \Box p \wedge (\Diamond\neg q \vee \Diamond\neg p)$
[2;1] T:1:$\Box(\neg p \vee q)$
[3;1] T:1:$\Box p \wedge (\Diamond\neg q \vee \Diamond\neg p)$
[4;3] T:1:$\Box p$
[5;3] T:1:$\Diamond\neg q \vee \Diamond\neg p$
[6;2] T:1.($\boldsymbol{x}_1$):$\neg p \vee q$
[7;4] T:1.($\boldsymbol{x}_2$):p
[8;5] T:1:$\Diamond\neg q$
[9;5] T:1:$\Diamond\neg p$
[10;8] T:1.1:$\neg q$
[17;9] T:1.2:$\neg p$
[11;10] F:1.1:q
[18;17] F:1.2:p
[12;6] T:1.(X_1):$\neg p$
[13;6] T:1.(X_1):q
[19;7,18] $\perp$
[14;12] F:1.(X_1):p
[16;11,13] $\perp$
[15;7,14] $\perp$

Figure 4.5: The tableau $T_{\mathrm{mv}}^{\mathrm{con}}$ from Example 4.2.34 is the result of applying the substitution σ_1 to the above tree.

universal variable $\boldsymbol{x}_2$ instead of X_2. Thus, it is sufficient to check that an instantiation of $\boldsymbol{x}_2$ *exists* that allows to close the branch. Only the substitution $\{X_1 \mapsto 1\}$ that instantiates the rigid variable X_1 is applied to the tableau.

As in the rigid variable case, after $\{X_1 \mapsto 1\}$ has been applied, closing the middle branch does not require a further instantiation. But now, as the variable $\boldsymbol{x}_2$ is universal and has *not* been instantiated with 1 when closing the left branch, it is not necessary to generate a second variant of formula 7. The right branch can be closed using the complementary atoms 7 and 18, because $\boldsymbol{x}_2$ could be instantiated with 2 in which case the label $1.(\boldsymbol{x}_2)$ of formula 7 were justified by formulae on the right branch (namely formula 18); again, it is not necessary to actually apply the substitution $\{\boldsymbol{x}_2 \mapsto 2\}$ to the tableau, which instantiates a universal variable; its existence is sufficient. $\Box$

4.4 Improved Skolemisation

Skolemisation is a satisfiability preserving deduction of the following general form: When a formula (in our framework a premiss) Π is given that implies the existence of objects with certain properties in all models of Π, then a Skolem symbol, a Skolem term, or other syntactical construct is introduced to represent an arbitrary one of these objects whose existence is known, and a formula is deduced expressing the fact that the object represented by the Skolem symbol has the property it is known to have.

Example 4.4.1 A formula $\delta = (\exists x)(\phi(x))$ of first-order predicate logic implies the existence of elements d in the domain of all first-oder structure $\langle D, \mathcal{I} \rangle$ satisfying δ that have the property that $val_{\mathcal{I},[x \mapsto d]}(\phi(x)) = true$. A Skolem constant c or a term t is introduced that represents d; and the formula $\phi(c)$ resp. $\phi(t)$ is deduced. $\square$

Example 4.4.2 An inequality $\mathsf{F}{:}(s \approx t)$ implies, if s and t are interpreted as sets, the existence of an element d that occurs in only one of the two sets and not in the other.

Thus, the expansion rule of the calculus $\mathcal{C}_{\mathrm{MLSS}}$ from Section 3.8 for the fragment MLSS of set theory introduces a constant c representing the existing element d when it is applied to an inequality. $\square$

The objects whose existence is known do not have to be elements of a universe or domain. They can as well be functions, relations, or possible worlds (as is the case in calculi for modal logics).

Tableau calculi for many logics have to use some sort of skolemisation. Often, however, the “standard” way of skolemising does not yield optimal results. These calculi can be improved as follows: Instead of introducing a new but arbitrary Skolem symbol, each premiss from which the existence of objects with certain properties can be deduced is assigned its own unique symbol, constant, or term. When the same premiss is used again for expansion, the same Skolem symbol is used. Since the set of all premisses is enumerable, they can only express enumerably many different properties such that an enumerable set of Skolem symbols is sufficient.

Such an improved skolemisation rule is used by the ideal tableau calculi for first-order predicate logic PL1 and for the modal logic K described in Chapter 3 and in Sections 4.2 and 4.3. It was first described for the case of first-order predicate logic in (Beckert *et al.*, 1993), namely in form of a liberalised expansion rule schema for δ-formulae. An improved version of the expansion rule schema for π-formulae in modal logics was first introduced in (Beckert & Goré, 1997). Improved skolemisation in first-order predicate logic in not a new idea; using a Skolem term assigned to the formula to be skolemised, resembles the ϵ-terms first defined in (Hilbert & Bernays, 1939) (a good introduction to epsilon logic can be found in (Meyer Viol, 1995)).

Employing this improved skolemisation in tableau calculi for automated deduction has several important advantages: First, it preserves the monotonicity of expansion rules, whereas a rule that introduces symbols that have to be *new* is non-monotonic (and calculi using such a rule are not ideal). Second, using the improved skolemisation restricts the search space as the number of different Skolem symbols that are used in a proof is smaller. And third, calculi using standard skolemisation usually do not have the *strong* soundness of expansion property (Property 2 in Def. 3.5.8), i.e, they do not necessarily preserve satisfiability by the *same* tableau interpretation. The reason is that a conclusion constructed using standard skolemisation is only satisfied by tableau interpretations in which the new Skolem symbol is interpreted in the right way, namely by one of the objects known to exist. If the improved Skolemisation is used, however, it is known beforehand what the properties of the object ist that is represented by a certain Skolem symbol. Thus, it is possible to choose the set of tableau interpretations such that it only contains *canonical* interpretations in which the Skolem symbols are interpreted by object that have the appropriate properties. If such canonical interpretations are used, then (improved) skolemisation not only preserves satisfiability, but the premiss and the conclusion are satisfied by the *same* (canonical) tableau interpretations. Of course, to prove soundness, one has to guarantee in addition that every initial tableau for a satisfiable formula set is satisfied by a *canonical* interpretation. That, usually, is only possible, if the Skolem symbols are known not to occur in an initial tableau—which is the main reason why we allow the use of an extended signature for the construction of tableau proofs, as the additional symbols of the extended signature do not occur in initial tableaux.

Often—depending on the expressivity of the particular logic—it is also possible to *syntactically* prove soundness of an expansion rule using the improved version of skolemisation, based on soundness of an expansion rule using standard skolemisation. For example, Egly (1998) proved that in a tableau calculus for PL1 with standard skolemisation rule and non-analytic cut (since the cut rule is trivially sound, using it does not impair the argument), it is possible to derive (in several steps) from the empty premiss the tableau formula

$$\mathsf{T}{:}(\forall x)((\exists y)(G(x,y) \rightarrow G(x, f(x)))) \ ,$$

where $f = sko_{\mathrm{fv}}((\exists y)(G(X,y))$, for all formulae G. That implies soundness of the improved skolemisation rule for first-order predicate logic, as this formula allows to derive $\mathsf{T}{:}G(X, f(X))$ from $\mathsf{T}{:}(\exists y)G(X,y))$ anywhere in a tableau proof.

Besides the question of which Skolem symbol to use there is the problem of ensuring soundness of skolemisation in the presence of free variables (both rigid and universal). One has to make sure that the instantiation of rigid variables with Skolem symbols (or terms that contain Skolem symbols) does not lead to undesired results.

Example 4.4.3 For all instantiations $\{Y \mapsto t\}$ of the rigid variable Y, the formula $(\exists x)(p(x,Y))\{Y \mapsto t\}$ implies the existence of an object d for which

$$val_{\mathcal{I},[x \mapsto d]}(p(x,Y)\{Y \mapsto t\}) = true \ .$$

In general, however, there is not a single d with this property for all t. □

The problem exemplified in the above example is usually solved by using a complex Skolem term instead of a Skolem constant and using the free variable on which the choice of the object d depends as arguments of that Skolem term, i.e., in the above example, the formula $p(c(Y), Y)$ is derived instead of $p(c, Y)$.

The difficulty in designing a sound rule schema with skolemisation is to find out what exactly influences what the objects are whose existence is known. All free variables whose instantiations may have an influence have to be used as arguments of the Skolem term. On the other hand, if too many variables are included in the Skolem terms, the calculus is weakened and the size of the search space increased.

For example, earlier versions of rigid variable calculi for PL1 used a skolemisation where, when the expansion rule is applied to a premiss containing a δ-formula δ, all rigid variables occurring on the tableau branch being expanded were used as arguments of the new Skolem term. Later, it was realised that what the objects are in the domain of a tableau interpretation whose existence is implied by δ only depends on the instantiation of the rigid variables occurring in δ and not on the instantiation of other rigid variables on the branch and that, thus, it is sufficient to use the rigid variables in δ as arguments of the Skolem term (Hähnle & Schmitt, 1994).

Of course, one can get around the problems resulting from the use of free variables by skolemising the input formulae in a preprocessing step, as it is done in all calculi using clausal normal form. However, in some logics that is not possible such as, for example, in intuitionistic predicate logic. In addition, skolemisation results in a loss of information (and early skolemisation in an early loss of information). Consider, for example, the formula $\phi = \mathsf{T}{:}(\exists x)(p(x))$ and its skolemised version $\phi' = \mathsf{T}{:}p(c)$. If a local lemma $\overline{\phi}$ resp. $\overline{\phi'}$ is generated (see Section 4.5), then the lemma $\overline{\phi}$, which is equivalent to $\mathsf{T}{:}(\forall x)(\neg p(x))$, is much more useful than the lemma $\overline{\phi'}$, which is equivalent to $\mathsf{T}{:}\neg p(c)$. In fact, tableau calculi for PL1 with lemma generation and dynamic skolemisation have non-elementary shorter proofs for certain formula classes than calculi with lemma generation and skolemisation as a preprocessing step (Egly, 1998).

4.5 Local Lemmata

A simple and in many cases useful way of strengthening tableau calculi with expansion rule is to make sure that the extensions of a conclusion do not *intersect semantically*:

Definition 4.5.1 Let $\mathcal{C}$ be a (ground) tableau calculus for a logic **L**; and let $\Sigma \in \mathit{Sig}$ be a signature.

Two branch extensions $E_1, E_2 \subset \mathit{TabForm}(\Sigma^*)$ *intersect semantically* if there is a tableau interpretation in $\mathit{TabInterp}(\Sigma^*)$, that satisfies both E_1 and E_2. □

Example 4.5.2 Consider the tableau calculus $\mathcal{C}_{\mathrm{PL1}}$ for PL1 from Section 3.6. The two extensions $\{\mathsf{T}{:}p\}$ and $\{\mathsf{T}{:}q\}$ of the conclusion for the premiss $\{\mathsf{T}{:}(p \vee q)\}$ intersect semantically, as $\mathsf{T}{:}p$ and $\mathsf{T}{:}q$ can both be satisfied by a single tableau interpretation of $\mathcal{C}_{\mathrm{PL1}}$. □

The application of an expansion rule using a conclusion with semantically intersecting extensions adds, in some sense, less information to the tableau. Intuitively, to close a tableau, one has to show that no tableau interpretation satisfies any of its branches; and if there are branches that are satisfied by the same tableau interpretations, then these interpretations have to be considered more than once.

Extensions can be made intersection-free by adding tableau formulae; this may require to use additional extensions, and one has to be careful to preserve soundness.

Theorem 4.5.3 *Let $\mathcal{C}$ be a (ground) ideal tableau calculus for a logic $\mathbf{L}$ that (1) has the soundness properties from Definition 3.5.3 (and is thus sound) and that is (2) complete.*

For each conclusion $C = \{E_1, \ldots, E_k\}$ ($k \geq 1$) over some signature $\Sigma^* \in \mathit{Sig}$ where $E_i = \{\phi^i_1, \ldots, \phi^i_{r_i}\}$ ($1 \leq i \leq k$), the conclusion $C^!$ is defined by

$$C^! = \{E_i \cup \bigcup_{j=1}^{i-1} \{\overline{\phi^j_{l_j}}, \phi^j_1, \ldots, \phi^j_{l_j-1}\} \mid 1 \leq i \leq k, \textit{ and } 1 \leq l_j \leq r_j \textit{ for } 1 \leq j \leq i\} \ .$$

Let the calculus $\mathcal{C}^!$ be defined as follows: The expansion rule $\mathcal{E}^!$ of $\mathcal{C}^!$ is, for all signatures $\Sigma \in \mathit{Sig}$ and all premisses $\Pi \in \mathit{TabForm}(\Sigma^*)$, given by

$$\mathcal{E}^!(\Sigma)(\Pi) = \{C^! \mid C \in \mathcal{E}(\Sigma)(\Pi)\}$$

where $\mathcal{E}$ is the expansion rule of $\mathcal{C}$; and $\mathcal{C}^!$ is identical to $\mathcal{C}$ except for the expansion rules.

Then,

1. *the calculus $\mathcal{C}^!$ has the soundness properties from Definition 3.5.3 (and is thus sound);*
2. *the calculus $\mathcal{C}^!$ is complete;*
3. *for all signatures $\Sigma \in \mathit{Sig}$ and all premisses $\Pi \in \mathit{TabForm}(\Sigma^*)$, any two extensions in a conclusion in $\mathcal{E}^!(\Sigma)(\Pi)$ do* not *intersect semantically.*

Proof: *Soundness properties:* The first soundness property from Definition 3.5.3 is trivially preserved as $\mathcal{C}$ and $\mathcal{C}^!$ only differ in their expansion rule.

To prove that the second soundness property (soundness of expansion) is preserved, we can make use of Lemma 3.5.9. It suffices to show that if one of the extensions in a conclusion $C = \{E_1, \ldots, E_k\}$ is satisfied by some tableau interpretation $\langle \mathbf{m}, I \rangle$, then one of the extensions in $C^!$ is satisfied by $\langle \mathbf{m}, I\rangle$. Let E_i be the first extension in C that is satisfied by $\langle \mathbf{m}, I\rangle$, i.e., $\langle \mathbf{m}, I\rangle$ does not satisfy any of the $E_1, \dots, E_{i-1}$; and let $\phi^j_{l_j}$ be the first tableau formulae in E_j that is not satisfied ($1 \le j \le i-1$), i.e., $\phi^j_1, \dots, \phi^j_{l_j-1}$ are satisfied by $\langle \mathbf{m}, I\rangle$. Consequently, the extension

$$E_i \cup \bigcup_{j=1}^{i-1} \{\overline{\phi^j_{l_j}}, \phi^j_1, \dots, \phi^j_{l_j-1}\} \ ,$$

which is an element of $C^!$, is satisfied by $\langle \mathbf{m}, I\rangle$.

Completeness: The completeness of $\mathcal{C}^!$ follows trivially from the completeness of $\mathcal{C}$ as both calculi are ideal and, therefore, the additional formulae in the extensions of $\mathcal{C}^!$ do not impede the construction of a $\mathcal{C}^!$-tableau proof using the same rule application that are used to construct a $\mathcal{C}$-tableau proof (note that for each extension $E^!$ in a conclusion $C^!$ there is an extension $E \in C$ such that $E \subset E^!$.

No semantical intersection: Let E_i and $E_{i'}$ be two different extensions in $C^!$; then, by construction of $C^!$, there is a tableau formula ϕ such that $\phi \in E_i$ and $\overline{\phi} \in E_{i'}$, which implies that E_i and $E_{i'}$ do not intersect semantically. □

Example 4.5.4 Assume that p, q, r, s, t are tableau formulae and that

$$\begin{array}{c|c|c}
\multicolumn{3}{c}{\Pi} \\ \hline
p & r & t \\
q & s &
\end{array}$$

is an expansion rule schema of a tableau calculus $\mathcal{C}$. Then, the corresponding expansion rule schema without semantical intersection of the calculus $\mathcal{C}^!$ (Theorem 4.5.3) is

$$\begin{array}{c|c|c|c|c|c|c}
\multicolumn{7}{c}{\Pi} \\ \hline
p & r & r & t & t & t & t \\
q & s & s & & & & \\
 & \overline{p} & \overline{q} & \overline{p} & \overline{p} & \overline{q} & \overline{q} \\
 & & p & & & p & p \\
 & & & \overline{r} & \overline{s} & \overline{r} & \overline{s} \\
 & & & & r & & r
\end{array}$$

□

The formulae that are added to extensions to make them intersection-free can be considered to be *local lemmata*. If in the above example, a branch B has been extended by $C^!$ and the new sub-branch containing p and q has been closed, one can conclude that in all tableau interpretations satisfying B either p or q is not satisfied and, moreover, either (a) p is not satisfied or (b) p is satisfied and q is not satisfied. Thus, these formulae can be added as “lemmata” to the subsequent extensions. The lemmata are “local”, because they are only used in the local conclusion.

As Example 4.5.4 demonstrates, making sure that all extensions in all conclusions that an expansion rule generates are intersection-free may lead to a drastic increase in the number of extensions per conclusion, and is therefore not always of advantage. Nevertheless, the method is useful, because one is free to either use the original conclusion C or the intersection-free conclusion $C^!$ constructed according to Theorem 4.5.3 depending on the number of extensions in C resp. $C^!$; in many cases the conclusion $C^!$ can be simplified as some of its extensions are inconsistent.

Example 4.5.5 Consider the following expansion rule schema that may be used by a calculus for the three-valued Łukasiewicz logic $\mathcal{L}_3$:

$$\begin{array}{c|c} \multicolumn{2}{c}{\mathsf{T}{:}\{\frac{1}{2}\}{:}F \lor G} \\ \hline \mathsf{T}{:}\{0, \frac{1}{2}\}{:}F & \mathsf{T}{:}\{\frac{1}{2}\}{:}F \\ \mathsf{T}{:}\{\frac{1}{2}\}{:}G & \mathsf{T}{:}\{0, \frac{1}{2}\}{:}G \end{array}$$

The labels are subsets of the set $\{0, \frac{1}{2}, 1\}$ of truth-values. Appropriate tableau interpretations for this calculus can be constructed as follows: For each model of Łukasiewicz logic, there is a tableau interpretation $\langle \mathbf{m}, I\rangle$ with a world $I(\sigma)$ for each possible label; such that $I(\sigma) \models F$ iff the truth value of F in the corresponding many-valued model is an element of σ.

Applying the construction from Theorem 4.5.3 yields the new intersection-free schema

$$\begin{array}{c|c|c} \multicolumn{3}{c}{\mathsf{T}{:}\{\frac{1}{2}\}{:}F \lor G} \\ \hline \mathsf{T}{:}\{0, \frac{1}{2}\}{:}F & \mathsf{T}{:}\{\frac{1}{2}\}{:}F & \mathsf{T}{:}\{\frac{1}{2}\}{:}F \\ \mathsf{T}{:}\{\frac{1}{2}\}{:}G & \mathsf{T}{:}\{0, \frac{1}{2}\}{:}G & \mathsf{T}{:}\{0, \frac{1}{2}\}{:}G \\ & \mathsf{F}{:}\{0, \frac{1}{2}\}{:}F & \mathsf{F}{:}\{\frac{1}{2}\}{:}G \\ & & \mathsf{T}{:}\{0, \frac{1}{2}\}{:}F \end{array}$$

The second extension in this schema is unsatisfiable as the truth value of F cannot be both an element of $\{\frac{1}{2}\}$ and *not* be an element of $\{0, \frac{1}{2}\}$. The third extension can be simplified as the first of its formulae subsumes the last one; and the second and third formula together imply that the truth value of G is 0. The result is the following schema that is intersection-free and has the same number of extensions as the original rule schema:

$$\begin{array}{c|c} \multicolumn{2}{c}{\mathsf{T}{:}\{\frac{1}{2}\}{:}F \lor G} \\ \hline \mathsf{T}{:}\{0, \frac{1}{2}\}{:}F & \mathsf{T}{:}\{\frac{1}{2}\}{:}F \\ \mathsf{T}{:}\{\frac{1}{2}\}{:}G & \mathsf{T}{:}\{0\}{:}G \end{array}$$

□

The above example has been taken from (Hähnle, 1993), where an algorithm is described for constructing intersection free expansion rules for arbitrary many-valued logics.

Making extensions intersection-free is also useful for free variable and universal variable calculi; but, since these do not have semantics defined by tableau interpretations, the construction from Theorem 4.5.3 is not applicable immediately. Instead one has to make sure that the extensions in the ground version of a calculus are intersection-free before the calculus is lifted, and its free variable version is constructed.

Example 4.5.6 The expansion rule of the calculus $\mathcal{C}_{\mathrm{PL1}}$ for PL1 allows to derive conclusions from β-formulae that are not intersection-free. An alternative version of the schema

$$\frac{\beta}{\beta_1 \mid \beta_2}$$

for β-formulae is the schema

$$\frac{\beta}{\beta_1 \;\Bigg|\; \begin{array}{c}\beta_2\\ \hline \beta_1\end{array}}$$

which preduces intersection-free conclusions; it can be used in rigid variable versions of $\mathcal{C}_{\mathrm{PL1}}$ as well.

Using this new schema has been shown to non-elementary reduce the size of the shortest proofs for certain classes of PL1-formulae (Egly, 1998). □

The following example shows that care has to be taken if local lemmata are generated that contain universal variables, and that the result of the uniform construction from Theorem 4.5.3 is not always optimal.

Example 4.5.7 Consider the rule schema

$$\frac{\mathsf{T}{:}(F(t) \vee G)}{\mathsf{T}{:}F(t) \mid \mathsf{T}{:}G}$$
$$\text{for all terms } t$$

It can be lifted to construct the universal variable schema

$$\frac{\mathsf{T}{:}(F(\boldsymbol{x}) \vee G)}{\mathsf{T}{:}F(\boldsymbol{x}) \mid \mathsf{T}{:}G}$$
$$\text{provided that } \boldsymbol{x} \text{ does not occur in } G$$

Unfortunately, if the ground schema is made intersection-free by adding the complement of $\mathsf{T}{:}F(t)$ as a lemma to the right extension, then the resulting schema

$$\frac{\mathsf{T}{:}(F(t) \vee G)}{\mathsf{T}{:}F(t) \;\Bigg|\; \begin{array}{c}\mathsf{T}{:}G\\ \mathsf{F}{:}F(t)\end{array}}$$
$$\text{for all terms } t$$

does not have a universal variable version anymore, because the term t now occurs in both extension.

It is, however, possible to use the lemma $\mathsf{F}{:}(\forall x)(F(x))$ instead of $\mathsf{F}{:}F(t)$. Then, the rule produces conclusions that are not not completely intersection-free but there are less tableau interpretation satisfying both extensions and the schema has a universal variable version:

$$\frac{\mathsf{T}{:}(F(\boldsymbol{x}) \lor G)}{\begin{array}{c|c} \mathsf{T}{:}F(\boldsymbol{x}) & \mathsf{T}{:}G \\ & \mathsf{F}{:}(\forall x)(F(x)) \end{array}}$$

provided that $\boldsymbol{x}$ does not occur in G

$\square$

4.6 Pruning

The pruning method, which is closely related to the technique called *condensing* (Oppacher & Suen, 1988), allows to reduce both the size of the search space and the size of generated tableau proofs.

Suppose a branch B of a tableau is extended using a conclusion $C = \{E_1, \dots, E_k\}$, and subsequently a closed sub-tableau is constructed below one of the extensions E_i where *none* of the tableau formulae in E_i has been used in a premiss for the construction of that sub-tableau; then the sub-tableau can be appended to any of the other sub-branches below any of the extensions E_j ($j \neq i$) or even immediately to the branch B.

There are several possible ways of making use of such situations:

1. The calculus can be strengthened by changing the tableau rule in such a way that $\bot$ can be added to all sub-branches below any of the extensions E_j ($j \neq i$); this, however, makes the calculus non-ideal, as $\bot$ in that case is not deduced from premisses on the branches that it is added to.

2. One can change the calculus in such a way that it is possible to post-pone the decision of whether a conclusion should be used to expand a tableau branch. The conclusion C is used preliminarily for expansion of B. Only if later on formulae of all its extensions are actually used to close a branch, the decision is made that C has indeed to be used for expanding B; otherwise, the decision is made that C should not be used, the preliminary expansion is undone, and the closed sub-tableau below E_i is attached immediately to B. This, however, again makes the calculus non-ideal, as in an ideal calculus formulae cannot be deleted once they have been added to a tableau.

3. The calculus remains unchanged; the information that the closed sub-tableau below E_i can be constructed without using formulae from E_i in premisses is

made use attaching this sub-tableau to all open branches below any of the extensions E_j ($j \neq i$). Then, the calculus remains ideal; and pruning is merely considered to be a technique for deterministically constructing closed sub-tableaux. Of course, an implementation does not have to actually construct the sub-tableau repeatedly; the information that this would be possible (and how) is sufficient.[5]

Definition 4.6.1 Let $\mathcal{C}$ be an ideal calculus for a logic **L**; let $\Sigma \in \mathit{Sig}$ be a signature; and let $T_0, \dots, T_n$ be a sequence of tableaux for a set $G \subset \mathit{Form}(\Sigma)$ of formulae such that T_{i+1} is a successor tableau of T_i ($1 \leq i < n$).

An occurrence of a tableau formula ϕ in some node N of a tableau T_i has been *used* for the construction of $T_{i+1}, \dots, T_n$ if

1. $\phi = \bot$ (i.e., all occurrences of the tableau formula $\bot$ are used); or
2. there is a tableau T_j ($i \leq j \leq n-1$) and a branch B_j through the node N_j in T_j that corresponds to N such that T_{j+1} has been constructed from T_j expanding the branch B_j using a premiss containing ϕ and a conclusion $C_j = \{E_1, \dots, E_k\}$; and an occurrence in T_{j+1} of at least one formula in each of the extensions $E_1, \dots, E_k$ is used for the construction of $T_{j+1}, \dots, T_n$. □

If there are several occurrences of the same formula on a branch B one of which has to be used to expand B, then, by definition, they are all used. One could instead only consider one of the occurrences to be used (preferably the one closer to the root of the tableau); however, too many occurrences of the same formula on a branch should be avoided anyway (see Section 5.2).

Example 4.6.2 Consider the tableau shown in Figure 4.6 (a); and assume that the tableau has been constructed by first using the conclusion $\{E_1, E_2\}$ of the premiss Π, then using the conclusion $\{E'_1, E'_2, E'_3\}$ of the premiss $\Pi' = E_1$, and then adding a closed sub-tableau T_1 to the branch below E'_1 and the closed sub-tableau T_2 to the branch below E'_2; the branches containing E'_3 and E_2 are still open.

Further assume that none of the formulae in E_1 has been used for the construction of the sub-tableau T_1, and that none of the formulae in E_1 and E'_2 has been used for the construction of T_2. Then, by definition, all the formulae in Π are considered to be unused. Whether formulae from E'_1 have been used for the construction of T_1 is irrelevant.

If the first of the possibilities for using pruning is employed, then $\bot$ can be added to the remaining open branches (Figure 4.6 (b)). If the second possibility is employed, i.e., all branch extensions that turn out to be unnecessary are deleted, the tableau in

[5] Note, however, that the sub-tableau has to be assumed to be (at least implicitly) present below each of the E_j the regularity of an expansion rule application is checked (see Section 5.2).

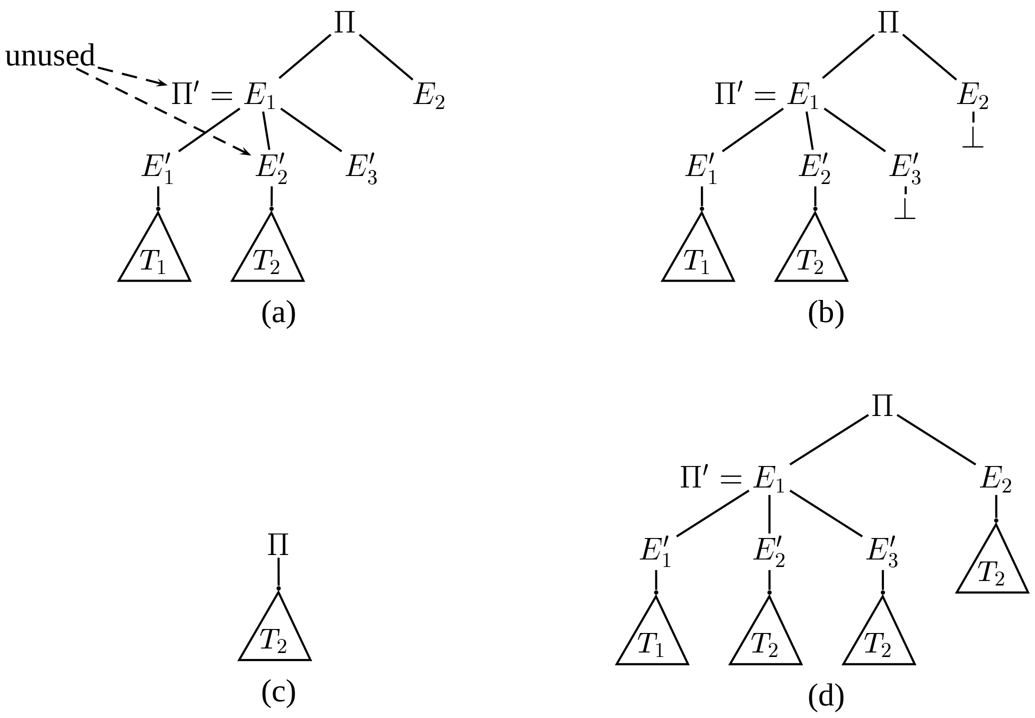


Figure 4.6: Using the pruning technique (Example 4.6.2).

Figure 4.6 (c) can be constructed. And the tableau in Figure 4.6 (c) results from employing the third possibility for using pruning, which preserves idealness. Note that the different pruning techniques all have to be applied twice, namely once for each of the two expansion rule applications that add the (unused) extensions E_1 resp. E'_2 to the tableau. $\square$

4.7 Additional Rule Schemata

A simple but often effective way of strengthening a calculus is to use additional rule schemata. If a certain sequence of rule applications occurs frequently, the expansion rule is enhanced such that a one-step rule application that has the same effect as the frequently occurring sequence of applications.

Example 4.7.1 Assume that in a certain application domain frequently formulae of the form $\mathsf{T}{:}(\mathit{true} \to \phi)$ occur, which may happen if the formulae to be proven are generated automatically.

Then, it is useful to enhance the expansion rule of $\mathcal{C}_{\mathrm{PL1}}$ with the additional schema

$$\frac{\mathsf{T}{:}(\mathit{true} \to \phi)}{\mathsf{T}{:}\phi}$$

such that $\mathsf{T}{:}\phi$ can be derived in one step without first extending the branch with the conclusion $\{\{\mathsf{F}{:}true\}, \{\mathsf{T}{:}\phi\}\}$ and closing one of the two new sub-branches by deriving $\perp$ from the premiss $\{\mathsf{F}{:}true\}$. $\square$

5 Constructing an Efficient Proof Procedure

5.1 Overview

5.1.1 Search Trees

In the previous chapter we discussed methods for improving a tableau calculus such that shorter proofs can be constructed. The subject of this chapter is how to efficiently search for proofs in the remaining smaller search space.

The proof search space can be visualised as a search tree where each possible choice of the next expansion rule application to a tableaux T creates a node with as many successor nodes as T has different successor tableaux.

There are two main concepts for proof search: *breadth first* and *depth first* search. Depth first search requires that either there are no paths in the search tree that do not contain proofs or it is possible to avoid such paths using fairness strategies for the construction of tableaux.

5.1.2 Breadth First Proof Search and Iterative Deepening

As fairness strategies that allow depth first search are difficult to construct for rigid variable calculi, breadth first search is used by most automated deduction systems. Breadth first search allows to find shorter proofs than depth first search because all paths of the search tree are considered whereas, using depth first search, paths in the search tree that contain short proofs may be missed; fairness strategies only guarantee that some proof is found but it may not the shortest one. However, the length of found proofs is not of great importance in automated deduction (the only advantage of short proofs is that they require less expansion rule applications and are thus easier to find); and breadth first search is very expensive as compared to depth first search because neighbouring paths in the search tree contain many similar or even identical tableaux that using breadth first search all have to be considered. This disadvantage of breadth first search by far outweighs any advantages it may have.

For all (practical) completion modes, i.e., (monotone) functions m from $\mathbb{N}$ to sets of tableaux such that $\bigcup_{i\in\mathbb{N}} m(i)$ includes all constructible tableaux, the size $|m(i)|$ of the search tree grows exponentially in i. It is—even for small i—usually not possible to store all tableaux in $m(i)$ in the memory of a machine. Therefore, most implementations (see, for example, (Beckert & Posegga, 1995)) use *depth first iterative deepening* (DFID) (Korf, 1985): the partial, finite search space consisting of all tableaux in $M(i) = \bigcup_{j \leq i} m(j)$ for some $i \in \mathbb{N}$ is searched for proofs in a depth first manner using backtracking, and if it turns out not to contain a proof, then i is increased. The tableaux in $M(i)$ are not available for the construction of the tableaux in $M(i+1)$; they have to be constructed again from scratch, which, however, merely causes polynomial overhead as compared to a breadth first search at the "right" level i because $M(i+1)$ is exponentially larger than $M(i)$. Although DFID search leads to acceptable performance of tableau-based automated theorem provers, it should be stressed that it is only a compromise used when no completeness preserving fairness strategy for depth first search is available.

5.1.3 Depth First Proof Search and Fairness Strategies

The advantage of depth first proof search is that the information represented by the tableaux that are constructed, increases at each proof step; no information is lost since there is no backtracking. In addition, considering similar tableaux or sequences of tableaux repeatedly that different paths of the search tree may contain are avoided.

In the case of ground tableau calculi, it is relatively easy to use depth first proof search. Their rules are not destructive; thus it suffices to systematically add all possible conclusions until all branches of the tableau that is constructed are either fully expanded or closed.

The situation is much more complicated in rigid variable calculi, which are destructive even if they are proof confluent. Applying a substitution may destroy formulae on a tableau that are needed for the proof such that they have to be deduced again from their premisses.

Up to now there was no practical solution to the problem of constructing deterministic proof procedures for rigid variable calculi that performs depth first search and are complete, i.e., that never fail to find a proof. Such procedure were only known for the special case of non-destructive rigid variable calculi, where branches are expanded without instantiating variables and only a single substitution is finally applied that is known to allow to close all branches simultaneously.

In this chapter, we analyse the problem of constructing a deterministic proof procedure for rigid variable calculi; and we present a solution for the general case of arbitrary rigid variable calculi that are ideal (and, in particluar, proof confluent). No other assumptions are made; in particular, the method is not restricted to calculi for certain logics or formulae in certain normal forms (such as clausal normal form).

The deterministic search strategy we propose is based on *regularity* (Section 5.2) to make sure that there are no "cycles" in the search (it is not possible to deduce the same formulae or sub-tableau again and again), and *weight orderings* (Section 5.3), i.e., each tableau formula is assigned a “weight” in such a way that there are only finitely many different formulae (up to variable renaming) of a certain weight; thus, if tableau formulae with lesser weight are deduced first, then sooner or later each conclusion is added to all branches containing its premiss, i.e., the strategy is *fair*.

The main difficulty is to define a regularity condition that on the one hand is restrictive enough to avoid all cycles in the proof construction and on the other hand is not too restrictive such that completeness is preserved.[1]

Our fairness strategy considers the whole tableau (and not only a single branch) both for checking regularity and for choosing a conclusion of minimal weight; a procedure based on this strategy may extend any branch of a tableau at any time. Note that this does not imply a large memory consumption; at least it is not worse than that of proof strategies where a “current” branch is extended until it is closed before other branches are considered and where DFID-based breadth first search is used to ensure completeness, as in that case all closed branches have to be stored for backtracking.

5.1.4 When is a Proof Procedure Practical?

As said above, no *practical* deterministic proof procedures for rigid variable calculi were known up to now. By “practical” we mean that the computational complexity of deciding what the next expansion rule application should be in each situation has to be reasonably low. In addition, the number of expansion steps that are necessary to find a proof has to be reasonably small as compared to the number of necessary steps when a breadth first search strategy is used.

There is trivially a (non-practical) deterministic proof procedure for all monotonic tableau calculi—namely the trivial procedure that performs a *breadth first* search in the background. Until a proof is found (in the background), it chooses arbitrary (foreground) expansion rule applications. When a proof has been found, it is appended as a sub-tableau to all open branches, which is possible because the calculus is monotonic. Such a proof procedure is of course completely useless. Note, that such a procedure can even be defined if the computational complexity of each expansion step is, for example, restricted to be polynomial in the size of the tableau that is expanded.

If the fairness strategy we present in the following sections is used, then the complexity of deciding what the next expansion step should be is in the worst case quadratic in the size of the tableau to be expanded (resp. the size of its possible successor tableaux). In the average case the complexity is much lower as only those parts of a tableau have to be considered that are affected by one of the possible expansion rule applications.

The size of the tableau proofs that are found (and thus the number of expansion steps)

[1] Baumgartner (1998) suggests a regularity condition that is (just slightly) too strong, such that cycles in the proof search are avoided but depth first search with *iterative deepening* has to be used to ensure completeness.

is at most that of the tableau proof constructed using DFID in the worst case (i.e., if coincidently all paths in the search tree not containing a proof are considered first).

5.2 Regularity

The notion of *regularity* is well known from tableau calculi for classical logic in clausal normal form (**?**) and negation normal form (Hähnle & Klingenbeck, 1996; Hähnle *et al.*, 1997); it is extended to the non-clausal case in (Beckert & Hähnle, 1998).

In the following, we define a concept of regularity for arbitrary ideal rigid variable calculi. Our concept differs from the classical notion in that it takes the whole tableau into concern and not only a single branch; it turns out to be appropriate for constructing a deterministic proof procedure for rigid variable calculi.[2]

Assume that a sequence $T_1, \dots, T_n$ of tableaux has already been constructed. We define an expansion rule application to T_n to be irregular if the successor tableau T_{n+1} is *contained* in one of the predecessor tableaux T_j—in particular, if T_{n+1} is contained in T_n. In that case, the sequence $T_j, \dots, T_{n+1}$ constitutes a cycle in the proof search because T_{n+1} does not contain any information that is not already in T_j. A tableau T_j contains T_{n+1} if each branch of T_j contains (up to renaming of free variables) all formulae from one of the branches of T_{n+1}. Intuitively, the tableau T_{n+1} is redundant if it is contained in T_j because, if closed sub-tableaux can be constructed below all branches of T_{n+1}, it is possible to construct the same closed sub-tableaux below all branches of T_j.

Note that checking whether an expansion rule application is regular or not does not involve unifiability tests because rigid variables may only be renamed but not instantiated with terms, i.e., checking regularity is not as complex as the problem of checking whether a tableau T_j *subsumes* T_{n+1} (which is NP-complete).

An important class of irregular expansion steps is the following: Assume a branch B_1 of a tableau T is extended using a rigid variable conclusion $\langle C, \sigma \rangle$, and a branch $B'_2\sigma$ in the resulting tableau T' is contained in all branches B of T that are affected by the expansion step, i.e., the branch B_1 (which is extended) and all other branches containing rigid variables that are instantiation by applying σ. This is in particular the case if $B'_2\sigma$ is contained in an initial sub-branch S_0 of T that ends above the first occurrence of any rigid variable in the domain of σ. The branch $B'_2\sigma$ of T' may be one of the branches that result from adding one of the extensions in C to B_1 or it may be any other branch that is affected by applying the substitution σ.

Example 5.2.1 Assume that the expansion rule of the rigid variable calculus for PL1 from Section 4.2.10 is used to close the branch B_1 of the tableau T shown in Fig-

[2] We do not consider universal variable calculi as they behave like ground calculi w.r.t. regularity.

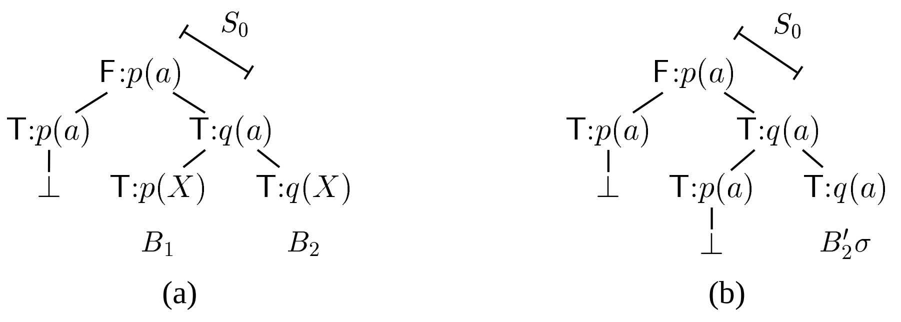


Figure 5.1: An irregular expansion rule application (Example 5.2.1).

ure 5.1 (a). This is done by deriving the conclusion $\langle\{\{\bot\}\},\ \{X \mapsto a\}\rangle$ from the premiss $\{\mathsf{F}{:}p(a),\ \mathsf{T}{:}p(X)\}$; the tableau T' shown in Figure 5.1 (b) is constructed.

This expansion rule application belongs to the class of easy to detect irregular applications described above. The right branch $B'_2\sigma$ of T' whose nodes are labelled with the formulae $\mathsf{F}{:}p(a)$ and twice $\mathsf{T}{:}q(a)$ is contained in the initial sub-branch S_0 of T whose nodes are labelled with $\mathsf{F}{:}p(a)$ and $\mathsf{T}{:}q(a)$; and S_0 ends above the first occurrence of X in T which is the only variable instantiated by σ.

Intuitively, this expansion rule application is useless because any closed sub-tableau that can be constructed below $B'_2\sigma$ can be constructed as well below both B_1 and B_2.
$\square$

An expansion rule application as described above is irregular according to the definition of regularity that is usually given in the literature (e.g. (Beckert & Hähnle, 1998)), if the branch $B'_2\sigma$ contains the same branch extension multiply; its relation to other branches is irrelevant.

Example 5.2.2 Figure 5.2 illustrates the difference between our and the classical notion of regularity. The situation is very similar to that shown in Figure 5.1 and explained in Example 5.2.1. But now, the initial sub-branch S_0 of T that contains the branch $B'_2\sigma$ of the tableau T' does not end above the first occurrence of the variable X, which is instantiated by σ. Thus, this expansion rule application does not belong to the class of easy to detect irregular applications; moreover, it is indeed *regular* according to our definition of regularity, because the branch B_1 of T does *not* contain any of the branches of T'.

According to the classical definition of regularity, however, this expansion rule application is *irregular* because the branch $B'_2\sigma$ of T' contains the extension $\{\mathsf{T}{:}q(a)\}$ twice. $\square$

As Example 5.2.2 demonstrates, the classical notion of regularity is more restrictive

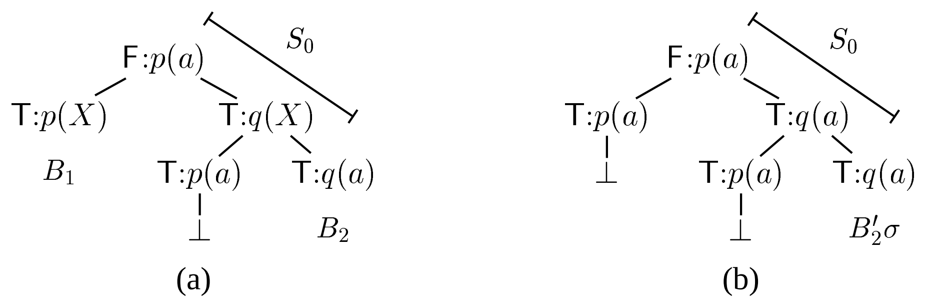


Figure 5.2: Example for the difference between the new and the classical notion of regularity.

than our notion in certain cases, which may destroy proof confluence.[3] In the ground, case where no variables are instantiated, there is no difference between the two notions of regularity.

According to the regularity condition described above, it is not irregular to add an arbitrary number of different *variants* of a formula to a tableau branch, which of course is useless and must be considered a violation of regularity. Thus, we enhance our definition such that expansion steps are irregular that add redundant variants of formulae already occurring on a tableau branch. We have to be careful, however, because it is *not* sufficient to allow only one variant of each formula to occur on a branch. The solution is to define the maximal number of variants of a formula that are not considered to be redundant to be the maximal size $k \in \mathbb{N}$ of minimal premisses in the calculus (how to handle a calculus with an expansion rule where minimal premisses can be of any size is discussed at the end of Section 5.4).

Example 5.2.3 Minimal premisses in the calculus from Section 3.6 for PL1 consist of at most $k = 2$ formulae. Thus, the number of different variants of a tableau formula that a branch contains at any given time can be restricted to two.

It may indeed be necessary to add two variants of a formula; for example, a branch containing only one variant of $\phi = \mathsf{T}{:}\neg p(X) \wedge p(f(X))$ cannot be closed; a second variant ϕ' is needed. □

Definition 5.2.4 Let $\mathcal{C}$ be a rigid variable calculus; let Σ_{fv} be a signature; and let $k \in \mathbb{N}$ be a natural number. Further let T and T' be tableaux; let $B'_1, \ldots, B'_m$ be the branches of T'; and let $\Psi_1, \ldots, \Psi_m$ be the sets of tableau formulae on these branches, i.e., $\Psi_i = \mathit{Form}(B'_i)$ $(1 \leq i \leq m)$.

The tableau T k-contains the tableau T', denoted by $T \subseteq_k T'$, if

[3] Whether using the classical, more restrictive notion of regularity does indeed destroy proof confluence, depends on which other restrictions are imposed; however, nobody has been able to define a deterministic proof procedure for rigid variable calculi using the classical notion. And the reason may very well be that it is too restrictive to preserve proof confluence when it is combined with other restrictions for ensuring fairness of the strategy (such as weight orderings).

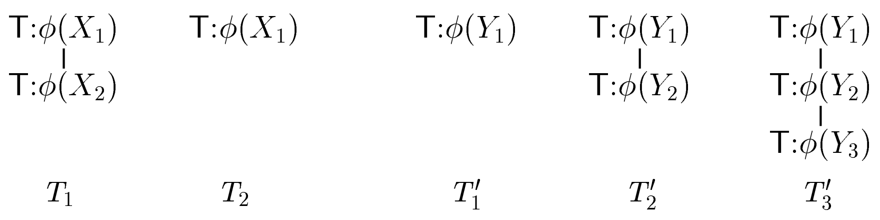


Figure 5.3: The tableaux from Example 5.2.6.

1. there is a variable renaming $\pi \in \mathit{Subst}(\Sigma^*_{\text{fv}})$, and
2. there are subsets $\Psi^k_i \subseteq \Psi_i$ ($1 \le i \le m$) that contain of each formula $\psi \in \Psi_i$
   - k variants or
   - as many variants as there are in Ψ_i,

   whatever is less; i.e., the number of variants of each ψ in Ψ^k_i is the minimum of k and the number of variants of ψ in Ψ_i,

such that for each branch B of T there is a branch B'_i of T' ($1 \le i \le m$) with

$$\Psi^k_i \subseteq \mathit{Form}(B\pi) \ .$$

□

Intuitively, a tableau T k-contains a tableau T' if each branch B of T contains *up to renaming of rigid variables* the formulae in some branch B' of T' where, however, it is sufficient if B only contains k different variants of each formula in B'. If two branches B_1 and B_2 "contain" the same branch B' of T', then they have to contain the *same* k variants of each formula in B'.

The following lemma follows immediately from the definition of $\subseteq_k$.

Lemma 5.2.5 *The relation $\subseteq_k$ on tableaux (Def. 5.2.4) is transitive and reflexive.*

Example 5.2.6 Consider the tableaux shown in Figure 5.3. If $k = 2$, then the tableau T_1 contains all the tableaux T'_1, T'_2, T'_3. The tableau T_2, however, only contains T'_1.
□

Example 5.2.7 The tableaux T_1 and T_2 shown in Figure 5.4 (a) and (b), respectively, do *not* 1-contain each other.

If T_1 was considered to contain T_2 and, thus, the construction of T_2 from T_1 was considered to be irregular, then completeness would be impaired because the tableau T_2 can be closed but T_1 cannot be closed.

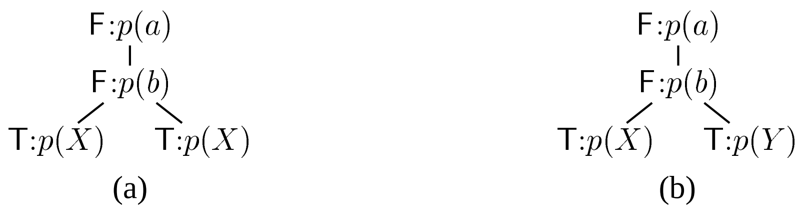


Figure 5.4: Nearly identical tableaux *not* containing each other (Example 5.2.7).

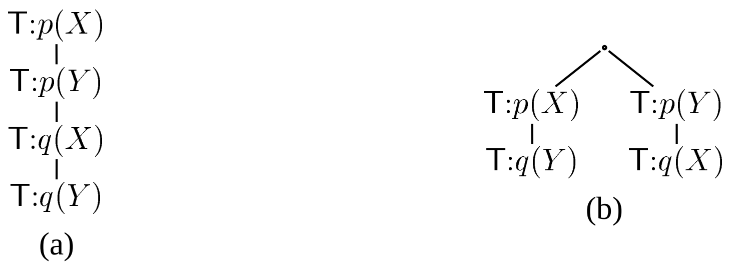


Figure 5.5: Two tableaux one of which 1-contains the other (Example 5.2.8).

If T_2 was considered to contain T_1, then neither soundness nor completeness would be affected. However, if the definition was changed such that a tableau T contains a tableau T' if an *instance* of T contains T' (according to the current definition)—in which case T_2 would 1-contain T_1—, then the computational complexity of checking regularity would be too high. □

Example 5.2.8 The tableaux T_1 shown in Figure 5.5 (a) 1-contains the tableau T_2 shown in Figure 5.5 (b). However, T_2 does *not* 1-contain T_1 although both branches of T_2 contain one variant of each of the formulae in the single branch of T_1; the reason why $T_1 \not\subseteq_k T_2$ is that the branches of T_2 contain *different* variants and would have to contain *the same* variants. For example, the left branch contains T:$p(X)$ whereas the right branch contains T:$p(Y)$ but they would have to contain either both T:$p(X)$ or both T:$p(Y)$. □

Based on the concept of a tableau k-containing another tableau, we can now formally define our notion of regularity.

Definition 5.2.9 Let $\mathcal{C}$ be an ideal rigid variable calculus; let Σ be a signature; and let $k \in \mathbb{N}$ be a natural number.

A (finite or infinite) sequence $(T_i)_{i \geq 0}$ of tableaux (and in particular a tableau proof $T_1, \ldots, T_n$) for a set $\mathfrak{F} \subset Form(\Sigma)$ of formulae is *k-regular* if it does *not* contain tableaux T_j and T_i where $j < i$ such that T_j k-contains T_i (Def. 5.2.4).

A sequence of tableaux that is not k-regular is *k-irregular*. □

If a sequence $T_1, \dots, T_n$ of tableaux is k-regular, T_{n+1} is a successor tableau of T_n, and the sequence $T_1, \dots, T_n, T_{n+1}$ is k-irregular, then the expansion rule application that is used to construct T_{n+1} from T_n is said to be k-irregular (because it causes the irregularity). Whether an expansion rule application is regular or not depends on the context in which it is used.

To check whether an expansion rule application is regular, it is sufficient to only consider those parts of the expanded tableau that are affected, i.e., the branch that is extended and the branches containing rigid variables that are instantiated.

If the notion of regularity as defined above is used to restrict proof search by only allowing regular sequences of tableaux, then completeness is preserved; i.e., our notion of regularity is not too restrictive.

Theorem 5.2.10 *Let $\mathcal{C}$ be an ideal rigid variable calculus that has a ground instance; let Σ be a signature; and let $k \in \mathbb{N}$ be the maximal size of the minimal premisses of all possible conclusions in $\mathcal{C}$.*

If there is a tableau proof for a set $\mathfrak{F} \subset Form(\Sigma^*)$ of formulae, then there is a regular tableau proof for $\mathfrak{F}$.

Proof: This theorem follows immediately from Theorem 5.4.4, which states the existence of complete deterministic proof procedures that only construct regular tableau proofs for all ideal rigid variable calculi that have a ground instance. □

The existence of a complete deterministic proof procedures that construct regular proofs (Theorem 5.4.4) not only implies Theorem 5.2.10 above, it also indicates that our notion of regularity is restrictive enough to serve its purpose. The proof of Theorem 5.4.4 makes use of the following lemma in which the restrictiveness of regularity is formalised. An infinite regular sequence of tableaux contains infinitely many different formulae or, equivalently, if a regular sequence of tableaux only contains (up to renaming of rigid variables) finitely many different tableau formulae, then it is finite.

Lemma 5.2.11 *Let $\mathcal{C}$ be an ideal rigid variable calculus; let Σ be a signature; and let $k \in \mathbb{N}$ be a natural number. Further let $\Gamma \subset TabForm(\Sigma^*)$ be a finite set of tableau formulae.*

Then, there is no infinite regular sequence $(T_i)_{i \geq 0}$ of tableaux such that, for all $i \geq 0$, the tableau formulae in T_i are variants of tableau formulae in Γ.

Proof: Let the equivalence relation $\sim_k$ on tableaux be defined by

$$T \sim_k T' \quad \text{iff} \quad T \subseteq_k T' \text{ and } T' \subseteq_k T \ .$$

We proceed to prove that there are only finitely many equivalence classes w.r.t. $\sim_k$ of tableaux that consist of variants of formulae in Γ. Let Γ^k be a set containing k different variants of each formula in Γ such that no two formulae in Γ^k have a rigid variable in common. Since Γ is finite, Γ^k is finite as well.

Let $\mathcal{T}_{\Gamma^k}$ be the set of all tableaux consisting of formulae from Γ^k where no branch contains the same formula more than once. The length of branches in these tableaux is at most $|\Gamma^k|$, which implies that there are only finitely many of them; thus, $\mathcal{T}_{\Gamma^k}$ is finite. It is easy to see that every tableau consisting of variants of formulae in Γ is equivalent w.r.t. $\sim_k$ to a tableau in $\mathcal{T}_{\Gamma^k}$. Therefore, $\mathcal{T}_{\Gamma^k}$ contains representatives of all equivalence classes of tableaux w.r.t. $\sim_k$, and there can be only finitely many such classes.

To complete the proof of the lemma, assume that the number of equivalence classes of tableaux (as defined above) is $n \in \mathbb{N}$. There cannot be an infinite regular sequence of tableaux consisting of variants of formulae in Γ, because in a sequence $T_1, \ldots, T_{n+1}$ of length $n+1$, there have to be at least two different tableaux T_j and T_i belonging to the same equivalence class, i.e., we have both $T_j \subseteq_k T_i$ and $T_i \subseteq_k T_j$. As either $j < i$ or $i < j$, that renders the sequence irregular. □

5.3 Weight Orderings

Weight orderings are the second important concept (besides regularity) on which our fairness strategy is based, that allows to construct deterministic proof procedures for arbitrary ideal rigid variable tableau calculi.

The important properties an ordering on tableau formulae that is used to ensure fairness has to have are the following:

1. It is a well-ordering on the set of tableau formulae (up to renaming of rigid variables), i.e., it is well founded and there are only finitely many tableau formulae that are incomparable to a given tableau formula.

2. Proper instances of a tableau formula ϕ have a higher weight than ϕ.

Intuitively, these are typical properties of orderings on tableau formulae that are defined by assigning a "weight" to the symbols of a signature (which is why we call them *weight* orderings).

Definition 5.3.1 Let $\mathcal{C}$ be a rigid variable calculus; and let Σ be a signature of $\mathcal{C}$.

A *weight function* w assignes to each tableau formulae in $\mathit{TabForm}(\Sigma^*)$ a natural number (its *weight*) such that the the following conditions are satisfied:

1. Given any tableau formula ϕ, there is *no infinite* set Ψ of tableau formulae such that

   (a) Ψ does not contain any ψ, ψ' that are identical up to renaming of rigid variables (i.e., all formulae in Ψ are really different),

(b) $w(\psi) \leq w(\phi)$ for all $\psi \in \Psi$.

2. If $X \in \mathit{Var}$ occurs in $\phi \in \mathit{TabForm}(\Sigma^*)$ and $t \in \mathit{Term}_{\mathrm{fv}}(\Sigma^*)$, $t \notin \mathit{Var}$, then
$$w(\phi) < w(\phi\{X \mapsto t\}) \ .$$

Given a weight function w, the *weight ordering* $\leq_w$ on tableau formulae (that is induced by w) is, for all $\phi, \psi \in \mathit{TabForm}(\Sigma^*)$, defined by
$$\phi \leq_w \psi \ \text{ iff } \ w(\phi) \leq w(\psi) \ .$$

□

A weight ordering is extended to *sets* of tableau formulae by comparing the *maximal* weight of the formulae they contain. This extension is a well-ordering as well, provided the sets that are compared are only allowed to contain a certain number of variants of each tableau formula.

Example 5.3.2 Let Σ be a signature of PL1. Assume that a positive weight is assigned to each function and each predicate symbol in Σ^* in such a way that only a finite number of function and/or predicate symbols have the same weight (rigid variables are implicitly assigned the weight 0); and let $w(\phi)$ be the sum of the weights of all function and predicate symbols occurring $\phi \in \mathit{TabForm}(\Sigma^*)$ (multiple occurrences of the same symbol count multiply). Then, w is a weight function according to Definition 5.3.1.

The condition that only a finite number of symbols must be assigned the same weight is not important in practice, as only a finite number of different function and predicate symbols can occur in the tableaux for a given set of PL1-formulae if the improved skolemisation as described in Section 4.4 is used; thus, all actually occurring symbols can be assigned the same weight. □

The purpose of Condition 1 in the definition of weight functions is obvious; it ensure that if infinitely many different formulae are added to a tableau branch, then the maximal weight of formulae on the branch will sooner or later reach each value. Condition 2 might need some more explanation. It makes sure that the weight increases in case no new formulae are appended to a branch but it is changed again and again by instantiating rigid variables; note that such a sequence of applications does not violate regularity because a new formula is created by each instantiation.

Example 5.3.3 Assume that an expansion rule allows to derive from each premiss of the form $\{p(f(\cdots(X)\cdots))\}$ the rigid variable conclusion $\langle\{\{q\},\{r\}\},\{X \mapsto f(X')\}$ where X' is any rigid variable different from X. Then, the (sub-)tableau shown in Figure 5.6 can be constructed for all n from the (sub-)tableau consisting of a single node marked with $\mathsf{T}{:}p(X_1)$.

Condition 2 in the definition of weight functions makes sure that the maximal weight of the formulae in the tableaux T_n increases. □

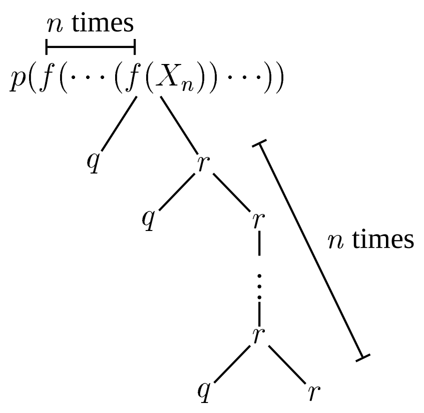


Figure 5.6: A tableau illustrating the necessity of Condition 2 in the definition of weight functions (see Example 5.3.3).

5.4 Deterministic Proof Procedures for Rigid Variable Calculi

In this section, we show that complete deterministic proof procedures can be defined for arbitrary ideal rigid variable calculi that have a ground instance; such proof procedures can be used to perform depth first search for tableau proofs (see the discussion in Section 5.1). They are constructed using the notions of regularity and weight orderings as described in Sections 5.2 and 5.3. For example, a deterministic proof procedure can be defined in this way for the rigid and the mixed variable calculi for PL1 from Sections 4.2.10 and 4.3.6.

Idealness of the calculus is indispensable. It has to be non-structural because otherwise the order in which formulae are added to the tableau is not irrelevant, and completeness may be lost when they are added in the order of their weight. The calculus has to be monotonic since a deterministic procedure may execute redundant proof steps between useful ones, and the redundant formulae that are added must not prevent useful expansion rule applications later on. We demand in addition that the calculus has a ground version because then it has the following property: if $\langle C, \sigma\rangle$ is a conclusion of some premiss Π, then $\langle C\sigma\tau, id\rangle$ is a conclusion of $\Pi\tau$ for all substitutions τ.

We only consider calculi in which minimal premisses are not of arbitrary size, i.e., do not contain more than k formulae for some fixed $k \in \mathbb{N}$. How to handle calculi that do not have this property is discussed at the end of this section.

To ensure that a proof procedure constructs a tableau proof (provided a proof exists), the sequences of tableaux that are constructed must satisfy the following two conditions:

1. The sequence of tableaux that is constructed must be k-regular, i.e., expansion rule applications creating a tableau that is k-contained in one of its predecessors are forbidden.

2. At each step, from all the possible expansion rule applications not violating the

regularity condition, one is chosen that creates a successor tableau in which the maximal weight of formulae is as small as possible (i.e., successor tableaux are compared according to the maximal weight of the formulae they contain).

If several possible expansion rule applications satisfy the above conditions, a proof procedure may employ arbitrary heuristics to choose one of them. A typical heuristic is, for example, to prefer conclusions that contain few extensions such that less new sub-branches are created.

Note that formulae are not necessarily added to a tableau branch in the order defined by their weight, because a formula ϕ can only be added when its premiss Π is present on the branch and weight of the formulae in Π may be higher than that of ϕ.

To comply with the condition that *all* expansion rule applications adding formulae of less weight have to be executed before formulae of higher weight are added to a tableau, it may be necessary to expand branches that are already closed. That is not always redundant, because closed branches still contain useful information and can influence other branches by the substitutions that are applied when they are expanded (the first substitution that is applied to close a branch is not necessarily the "right one" that allows to complete the proof). If a closed branch has no rigid variables in common with other branches, it needs not be further expanded.

Unfortunately, the regularity condition as defined above is still very difficult to implement; it requires to compare a tableau T_{n+1} with *all* its predecessors $T_1, \ldots, T_n$ and not only with the tableau T_n from which it is constructed. Such a regularity check is prohibitively expensive w.r.t. both space and time. Moreover, if an irregularity is encountered, i.e., if T_{n+1} is contained in one of the predecessor tableaux T_j, then other successor tableaux of T_j (besides T_{j+1}) have to be considered, which in a certain sense amounts to backtracking. The reason for this is the following: A tableau T_{n+1} that is contained in a tableau T_j does not have to be considered for proof search because all the proofs that may be constructed from T_{n+1} can be constructed from T_j. Now, if $j = n$, then we can just exclude the successor tableau T_{n+1} and be sure that if there is a proof derivable from T_{n+1} is is derivable from T_n without considering T_{n+1}. If, however, $j \neq n$, then the tableau proof that is known to be derivable from T_{n+1} and thus from T_j may not involve T_n but require to procceed with an alternative successor tableau T'_{j+1} different from T_{j+1}.

All these problems stem from the fact that a tableau T_j is not necessarily contained in its successor tableau T_{j+1} because rigid variable calculi are destructive and formulae occurring in T_j may not occur in T_{j+1} any more. Indeed, an irregular infinite sequence of tableaux can contain a cycle consisting of infinitely many tableaux T_{n_i} that all contain each other, where none of the irregular expansion steps is easy to detect, i.e., there is no tableau in the sequence that is contained in its immediate predecessor.

Example 5.4.1 Assume that the following symmetric rule schemata characterise the expansion rule of a rigid variable calculus:

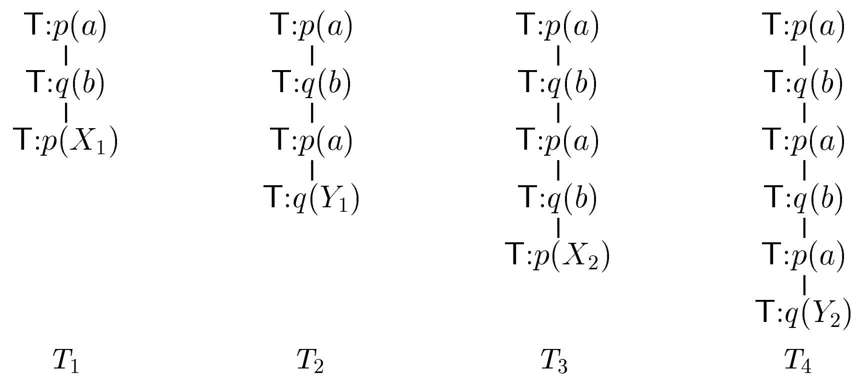


Figure 5.7: An irregular sequence of tableaux containing a cycle (Example 5.4.1).

$$\frac{\mathsf{T}{:}p(t)}{\mathsf{T}{:}q(Y)}$$

if t is unifiable with the constant a;
Y is an arbitrary rigid variable;
a unifier of t and a hat to be applied

$$\frac{\mathsf{T}{:}q(t)}{\mathsf{T}{:}p(X)}$$

if t is unifiable with the constant b;
X is an arbitrary rigid variable;
a unifier of t and b hat to be applied

Then, starting from an initial tableau T_1 consisting of the tableau formulae $\mathsf{T}{:}p(a)$, $\mathsf{T}{:}q(b)$, and $\mathsf{T}{:}p(X_1)$, a sequence $T_1, T_2, \ldots$ of tableaux can be derived such that each tableau T_i in the sequence 1-contains the tableau T_{i+2} but no tableau T_i contains the tableau T_{i+1} ($i \geq 0$). The first four tableaux in the sequence are shown in Figure 5.7. □

The above example illustrates the reason why cycles may occur that are difficult to detect, but it is rather artificial and the cycle could be avoided by always using the most specific premiss that allows to derive a certain conclusion (which is a good heuristic anyway). That is, instead of deriving the conclusion $\langle\{\{\mathsf{T}{:}p(Y)\}\}, \{X \mapsto a\}\rangle$ from the premiss $\{\mathsf{T}{:}p(X)\}$, the conclusion $\langle\{\{\mathsf{T}{:}p(Y)\}\}, id\rangle$ is derived from the premiss $\{\mathsf{T}{:}p(a)\}$. There are, however, more complicated situations where using most specific premisses does not help, as the following example demonstrates.

Example 5.4.2 Figures 5.8 and 5.9 show one cycle in an infinite irregular sequence of tableaux that is constructed using the mixed variable calculus for PL1 from Section 4.3.6.

To construct this sequence, the following expansion rule applications are repeated again and again:

– A conclusion of the form $\{\{\mathsf{T}{:}r(Y)\}, \{\mathsf{T}{:}s(Y)\}\}$ is derived from $\{\mathsf{T}{:}(r(\boldsymbol{y}) \vee s(\boldsymbol{y}))\}$ and is used to expand the rightmost branch of the tableau (for example, to derive tableau T_6 from tableau T_5).

– The third branch from the right is closed and a variable X is instantiated with a (for example, to derive T_7 from T_6).
– A conclusion of the form $\{\{\mathsf{T}{:}p(X)\}, \{\mathsf{T}{:}q(X)\}\}$ is derived from $\{\mathsf{T}{:}(p(\boldsymbol{x}) \lor q(\boldsymbol{x}))\}$ and is used to expand the rightmost branch of the tableau (for example, to derive T_8 from T_7).
– The third branch from the right is closed and a variable Y is instantiated with b (for example, to derive T_9 from T_8).

When these four steps have been executed, the cycle is complete, i.e., each tableau T_i in the sequence contains the tableau T_{i+4} for $i \geq 5$. No tableau in the sequence contains its immediate successor tableau, such that the cycle is difficult to detect. Note that for each expansion rule application, the most specific premiss is used. □

The problems discussed above that all result from the fact that a tableau T_{i+1} may not contain its predecessor tableau T_i, can be solved by making the proof procedure less destructive in the following sense: We demand that immediately after an expansion step that destroys formulae, the expansion steps that are needed to recreate the destroyed formulae are executed. In the worst case, a new copy of the sub-tableau that was affected by the instantiation is created and appended to all sub-branches that have been affected. To execute such a reconstruction step is always possible if the calculus has a ground version. The result is a tableau T^+_{i+1} that contains both T_i and T_{i+1} and all the tableaux that occur as intermediate results during the reconstruction.

Example 5.4.3 The left branch of the tableau T_i in Figure 5.10 (a) is closed using the conclusion $\{\{\bot\}\}, \{X \mapsto a\}$. The result is the tableau T_{i+1} in Figure 5.10 (b), in which all formulae containing the rigid variable X have been destroyed. They are reconstructed by appending a copy of the sub-tableau $S(X)$ that consists of all formulae in T_i in which X occurs to all the branches in T_{i+1} from which formulae are missing; the resulting tableau T^+_{i+1} (Figure 5.10 (c)) contains both T_i and T_{i+1}. □

If a deterministic proof procedure executes a reconstruction step after each expansion rule application, then a sequence $T^+_1, T^+_2, \ldots$ of tableaux is constructed where T^+_{i+1} is constructed from T^+_i by applying a regular expansion step and then reconstruct the destroyed formulae. To check whether such a sequence is regular, it is sufficient to test whether the immediate successor tableau T_{i+1} of T^+_i is contained in T^+_i. The earlier predecessors do not have to be considered as they are all contained in T^+_i.

Theorem 5.4.4 *Let $\mathcal{C}$ be an ideal rigid variable calculus that has a ground instance; let Σ be a signature; let $k \in \mathbb{N}$ be the maximal size of the minimal premisses of all possible conclusions in $\mathcal{C}$; and let w be a weight function.*

If there is a tableau proof for a set $\mathfrak{F} \subset Form(\Sigma^*)$ of formulae, then every sequence $(T^+_i)_{i\geq 1}$ of tableaux for $\mathfrak{F}$ that is constructed as described below contains a closed tableau T^+_n ($n \in \mathbb{N}$):

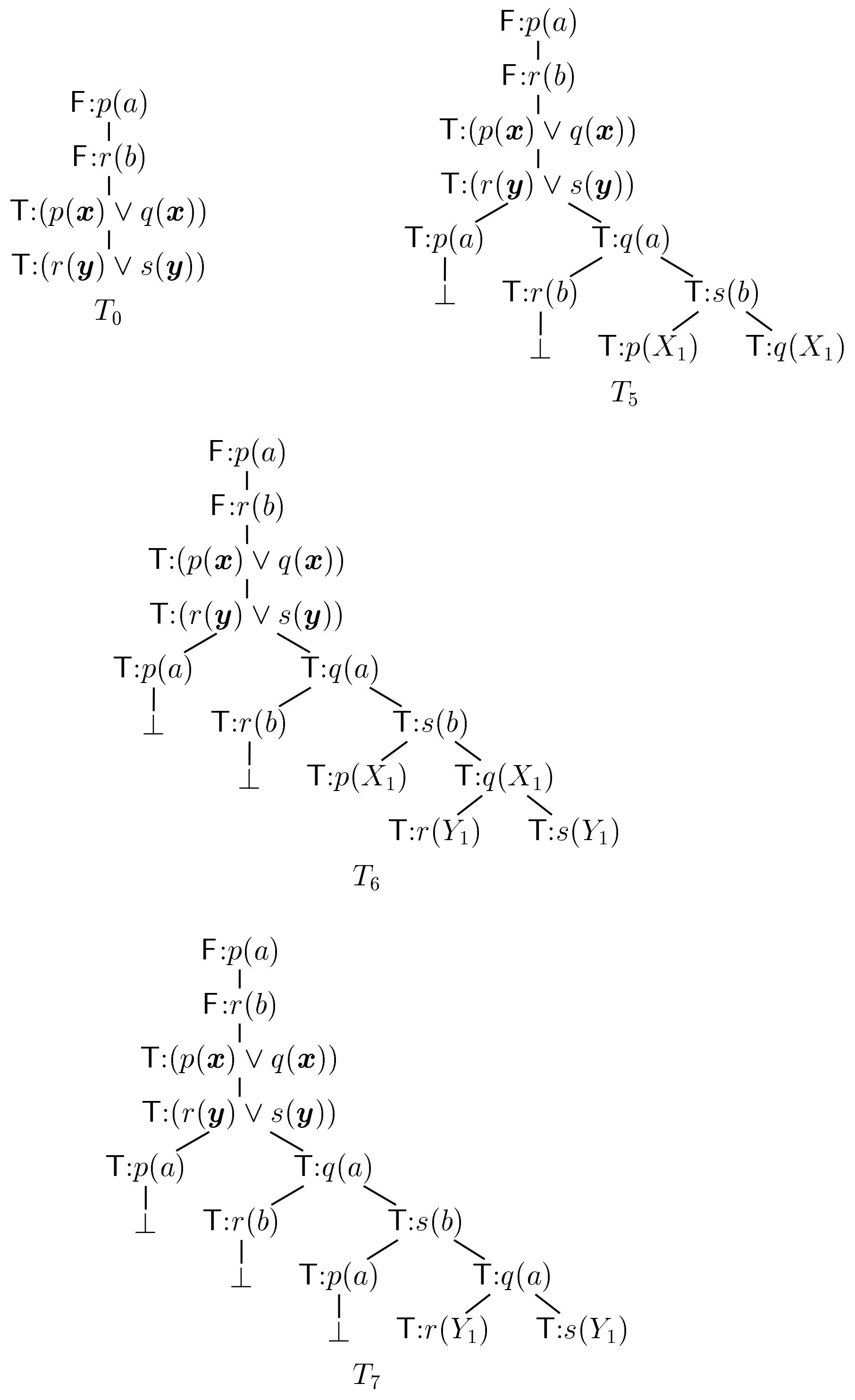
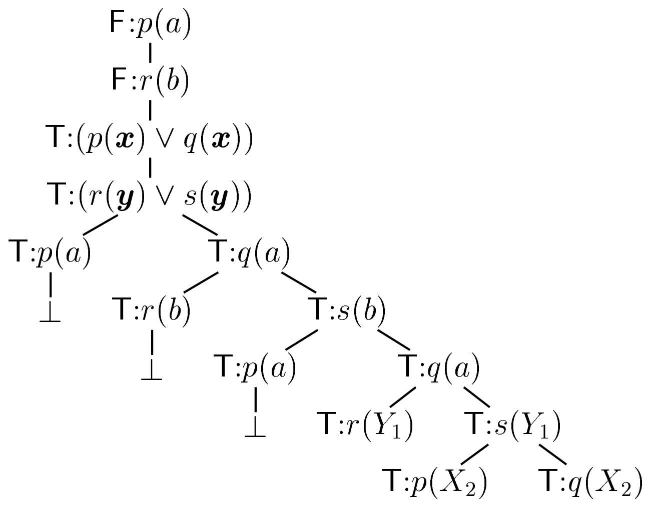
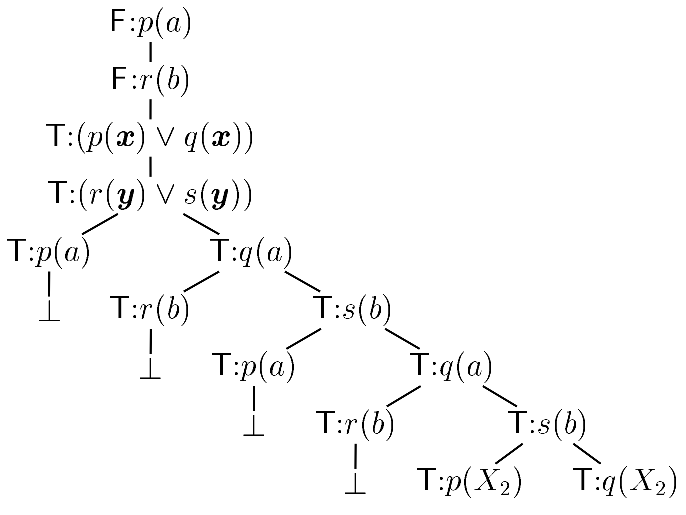


Figure 5.8: A more complex irregular sequence of tableaux containing a cycle (part 1); see Example 5.4.2.



T_8



T_9

Figure 5.9: A more complex irregular sequence of tableaux containing a cycle (part 2); see Example 5.4.2.

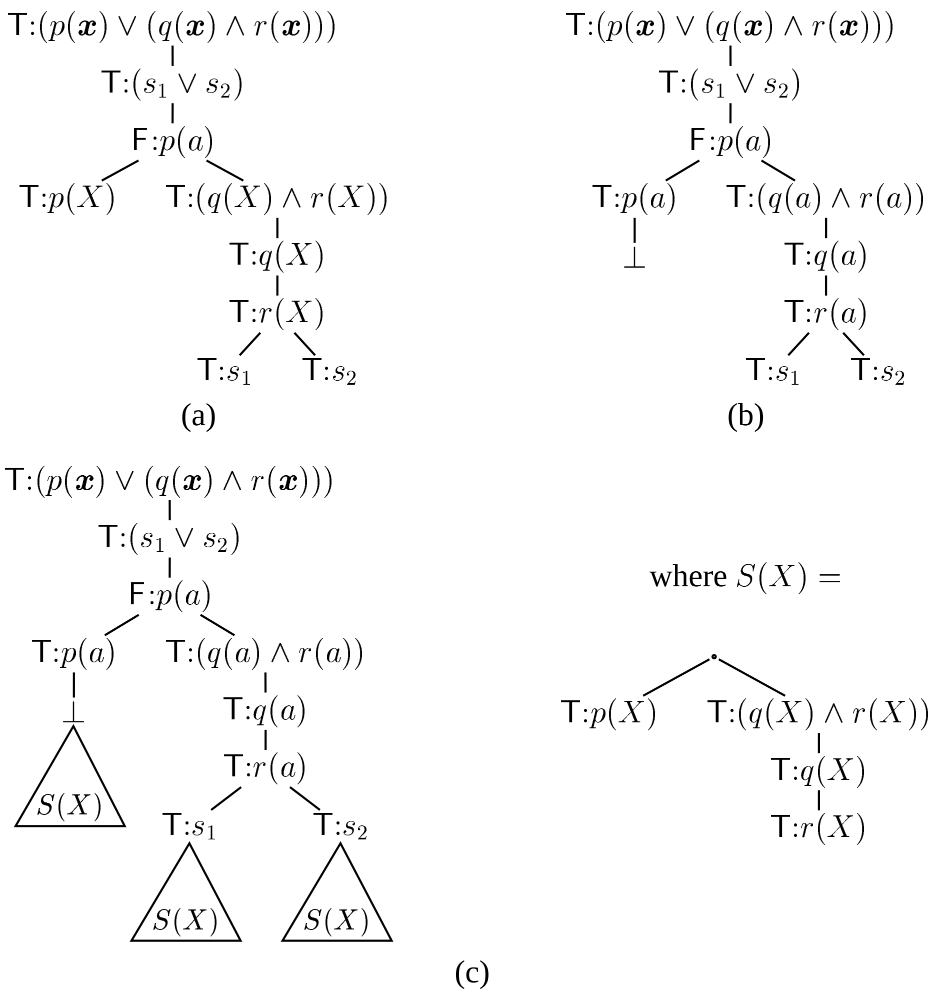


Figure 5.10: A tableau reconstruction step (Example 5.4.3).

- *T_1^+ is an initial tableau for $\mathfrak{F}$.*
- *For all $i > 1$,*
  1. *T_{i+1} is a successor tableau of T_i^+ such that (a) T_{i+1} is not k-contained in T_i^+, and (b) there is no successor tableau T'_{i+1} of T_i^+ that satisfies condition (a) and has a smaller maximal formula weight than T_{i+1} (w.r.t. the weight function w).*
  2. *Let $\langle C_i, \tau_i \rangle$ be the conclusion derivable from some premiss on T_i^+ that is used to construct T_{i+1}, and let S_i be the minimal sub-tableau of T_{i+1} that contains all occurrences of free variables instantiated by τ_i. The tableau T_{i+1}^+ is constructed from T_{i+1} by repeatedly executing all expansion rule applications needed to construct the sub-tableau S_i and thus appending S_i at the end of each branch going through the sub-tableau in T_{i+1} that results from applying τ_i to S_i.*

Proof: *Part 1:* Since $\mathcal{C}$ is ideal, if $\langle C, \sigma \rangle$ is a conclusion of some premiss Π, then $\langle C, id \rangle$ is a conclusion of $\Pi\sigma$. Therefore, the expansion rule applications needed to construct T_i^+ from T_i do not require any instantiations of rigid variables, i.e., they are non-destructive, which implies that, for all $i \geq 1$,

$$T_i \subseteq_k T_i^+ \quad .$$

By construction of T_{i+1}^+, all formulae that are destroyed by applying the substitution τ_i to T_i^+ are reintroduced to all branches from which they are missing in T_{i+1}. Thus, we have

$$T_i^+ \subseteq_k T_{i+1}^+ \quad .$$

Part 2: We show that the sequence $(T_i^+)_{i\geq 1}$ is regular (Def. 5.2.9). Assume the contrary; then the sequence contains tableaux T_i^+, T_j^+, $i > j$, such that $T_i^+ \subseteq_k T_j^+$. Using the results of Part 1, that implies

$$T_i \subseteq_k T_i^+ \subseteq_k T_j^+ \subseteq_k T_{i-1}^+ \quad ,$$

which contradicts the fact that (by definition) the tableau T_i is *not* k-contained in T_{i-1}^+.

Part 3: Let $w_{\max} \in \mathbb{N}$ be an arbitrary weight. We prove that the initial sub-sequence of $(T_i^+)_{i\geq 1}$ that only contains formulae ϕ of weight $w(\phi) \leq w_{\max}$ is finite.

Let Γ be a set of representatives of each class of tableau formulae in $(T_i^+)_{i\geq 1}$ that are identical up to variable renaming and whose weight is not bigger than $w_{\max}$; thus, if ϕ is a tableau formula with $w(\phi) \leq w_{\max}$ in $(T_i^+)_{i\geq 1}$, then there is a variant of ϕ in Γ; we assume that the representatives in Γ are chosen in such a way that no two formulae in Γ have a rigid variable in common.

The definition of weight orderings implies immediately that the set Γ must be finite. Thus, Lemma 5.2.11 applies and the initial sub-sequence of $(T_i^+)_{i\geq 1}$ in which no formula of weight bigger than $w_{\max}$ occurs is finite.

Part 4: By assumption there is a (possibly irregular) tableau proof $T_1', \dots, T_m'$ for $\mathfrak{F}$. We prove by induction on j that there are substitutions $\sigma_1, \dots, \sigma_m \in \mathit{Subst}(\Sigma^*)$ and weights $w_1, \dots, w_m \in \mathbb{N}$ such that

$$T_j' \subseteq_k T_{n_j}^+ \sigma_j$$

where T_{n_j} is the last tableau in the initial sub-sequence of $(T_i^+)_{i\geq 1}$ that contains no formula with a weight bigger than w_j (which exists according to Part 3).

$j = 1$: Let $\sigma_1 = id$ and $w_1 = 0$ such that $n_1 = 1$. The tableaux T_1' and T_1^+ are both initial tableaux for $\mathfrak{F}$ and do not contain rigid variables; thus, we have trivially $T_1' \subseteq_k T_{n_1}^+ \sigma_1$.

$j \to j+1$: Let Π' be the premiss and $\langle C', \rho \rangle$ the conclusion used in the construction of T_{j+1}' from T_j'; let B_j' be the branch in T_j' that has been expanded; and let π be the variable renaming that exists because $T_j' \subseteq_k T_{n_j}^+ \sigma_j$ according the the definition of $\subseteq_k$. Now, let the tableau $\hat{T}$ be constructed by expanding *all* branches of $T_n^+ \sigma_j \pi$ that k-contain the branch B' (they contain a variant of the premiss Π) using the conclusion $\langle C', \rho \rangle$. Obviously, the tableau $\hat{T}$ k-contains the tableau T_{j+1}'.

Let w_{j+1} be the maximal weight of the formulae in $\hat{T}$. The tableau $T_{n_{j+1}}^+ \sigma_j \pi$ k-contains $\hat{T}$. Otherwise, the expansion of $T_{n_{j+1}}^+$ using the conclusion $\langle C', \rho \circ \pi \circ \sigma_j \rangle$ is regular. That leads to a contradiction, because all formulae that are added by such an expansion rule application are variants of formulae in $\hat{T}$; they thus all have a weight that is less than or equal to w_{j+1}; but since the tableau $T_{n_{j+1}}^+$ is by definition the last tableau in the sequence $(T_i)_{i\geq 1}$ not containing a formula of weight bigger than w_{j+1}, all expansion rule applications to $T_{n_{j+1}}^+$ that do not add a formula of weight bigger than w_{j+1} must be forbidden by the regularity condition.

Therefore, we have

$$T_{j+1}' \subseteq_k \hat{T} \subseteq_k T_{n_{j+1}}^+ \sigma_j \pi$$

and thus

$$T_{j+1}' \subseteq_k T_{n_{j+1}}^+ \sigma_{j+1}$$

where $\sigma_{j+1} = \sigma_j \pi$.

Part 5: Now we can conclude the proof of the theorem as, in particular, the tableau $T_{n_m}^+ \sigma_m$ k-contains the tableau T_m'. Since T_m' is closed, every branch of T_m' contains $\bot$. Therefore, every branch of $T_{n_m}^+ \sigma_m$ and, thus, every branch of $T_{n_m}^+$ contains $\bot$, i.e., $T_{n_m}^+$ is closed. $\square$

A proof procedure satisfying the conditions of Theorem 5.4.4, which constructs a regular sequence $T_1^+, T_2^+, \dots$ of tableaux such that $T_i^+ \subseteq_k T_{i+1}^+$ for all $i \geq 1$, simulates—in a certain sense—a depth first iterative deepening search (see Section 5.1.2). The weight of the formulae that can occur in the tableaux increases stepwise. If some (possibly irregular) tableau proof exists that does not contain formulae of weight bigger than $w_{\max}$, then there is a closed tableau T_n^+ that is the last in the constructed sequence not containing formulae of weight bigger than some $w_{\max}^+ \in \mathbb{N}$. It contains all tableaux that can be constructed from formulae ϕ of weight $w(\phi) \leq w_{\max}$. The big advantage of this simulated DFID over classical DFID search based on backtracking is that the tableau T_n^+ is a very compact representation of the search space. All the information that is contained in tableaux whose formulae are of weight less than $w_{\max}$ is present in the single structure T_n^+; and all the tableaux in the search space that are identical or in some way symmetrical to each other are represented by only one sub-tableau of T_n^+. Since no backtracking occurs, no information that has been derived is ever lost. There may be parts of the tableau T_n^+ that represent redundant information and are therefore useless (i.e., non-closed subtableaux that should not have been created); but these are not harmful as they can be removed using the *pruning* technique described in Section 4.6.

The method for designing a deterministic proof procedure described in this section can only be applied if the number of formulae in minimal premisses is limited by some $k \in \mathbb{N}$. It can, however, easily be extended to arbitrary ideal calculi with a ground instance (such as, for example, the calculus for modal logics described in Section 3.7.4) by iteratively increasing the limit k during the construction of the regular sequence $(T_i^+)_{i \geq 1}$ of tableaux—such that, for all pairs $\langle k, w_{\max} \rangle$, at least k variants of all formulae of weight less than $w_{\max}$ are sooner or later added to each tableau branch. One could, for example, define that $k = k_w(w_{\max})$ where k_w is a monotonically increasing function and $w_{\max}$ is the maximal weight of the formulae in the current tableau.

5.5 Search Space Restrictions that Preserve Idealness

The method of constructing deterministic proof procedures for rigid variable calculi described in the previous section is compatible with all search space restrictions that preserve idealness of the calculus. These restrictions are usually based on semantical properties of the particular logic, such that they are difficult to formulate in a uniform way. Important examples for such restrictions are the following.

Connectedness For many calculi (and logics), it is possible to define a *connectedness* relation between tableau formulae (or between tableau formulae and sets of tableau formulae) such that, if a tableau formula ϕ is *not* connected to any other formula (or set of formulae) on the tableau branch on which it occurs, then it is known to be redundant, i.e., neither ϕ nor any formula derivable from ϕ can ever be used in a branch closure. That is, (1) no minimal premiss that contains ϕ allows the deduction of $\bot$ and, iteratively, (2) no conclusion that can be derived from a minimal premiss containing ϕ contains a formula that is connected to another formula (or sets of formulae). Of course, to be useful for restricting the search space, the connectedness relation must be easy to check.

For example, if the calculus for PL1 from Section 3.6 is used to construct tableaux, then a connectedness relation can be used where two formulae are considered to be connected if and only if they contain occurrences of the same atom with different polarity (see, for example, (Beckert & Hähnle, 1998)).

A notion of connectedness based on checking the occurrence of the same atom with different polarity can be used for most logics in the sense that the occurrence of such atoms is *sufficient* for connectedness. Often, however, formulae are connected in other ways as well (for example if they contain equalities that may be "applied" to other formulae).

If an ideal calculus is restricted in such a way that a minimal premiss containing a formula must never be used for expansion, then idealness of the calculus is preserved. However, idealness and proof confluence are lost if a *strong* connectedness condition is used, where a minimal premiss may only be used for expansion if one of its formulae is connected to the leave of the branch that is expanded.

Selection Functions Another important method for restricting the search space that preserves idealness of a calculus are selection functions based in literal orderings, which are well-known from calculi for clausal predicate logic (Hähnle & Klingenbeck, 1996; Hähnle & Pape, 1997).

Note that literal orderings are a completely different concept than the weight orderings defined in Section 5.3; literal orderings are (in general) no well-orderings as there may be infinitely many formulae that are incomparable.

6 Fibring

6.1 Overview

The methodology of *fibring* is a successful framework for combining logical systems based on combining their semantics (Gabbay, 1996c; Gabbay, 1996a; Gabbay, 1998), see Section 6.2. The basic idea is to combine the models defining the semantics of two logics $\mathbf{L}_1$e and $\mathbf{L}_2$ such that the result can be used to define semantics for expressions (formulae) from the combined languages of $\mathbf{L}_1$ and $\mathbf{L}_2$. The general pre-condition for fibring is that these models have components like, for example, the worlds in Kripke structures, which is in full accordance with the assumptions we have made regarding the form of models.

To build fibred models, *fibring functions* $\mathcal{F}_{(1,2)}$ are defined that assign to each world w of an $\mathbf{L}_1$-model $\mathbf{m}_1$ an $\mathbf{L}_2$-model $\mathbf{m}_2$. When an $\mathbf{L}_2$-formula is to be evaluated in w, where its value is undefined at first, it is evaluated in $\mathbf{m}_2 = \mathcal{F}_{(1,2)}(w)$ instead. The full power of the fibring method is revealed when this process it iterated to define a semantics for the logic $\mathbf{L}_{[1,2]}$, where the operators of the component logics can occur arbitrarily nested in formulae. Fibring has been successfully used in many areas of logic to combine systems and define their semantics; for an overview see (Gabbay, 1996c; Gabbay, 1998).

In this chapter, we extend the fibring approach to *tableau calculi*. We describe how to uniformly construct a sound and complete tableau calculus for the combined logic from tableau calculi for the component logics. Since tableau calculi are readily available for most "basic" logics (including classical logics, modal logics, intuitionistic logics, temporal logics, and many more), calculi can be obtained for all "complex" logics that can be constructed by fibring basic logics, such as modal predicate logic, intuitionistic temporal logic, etc.

The main advantage of a uniform framework for fibring tableau calculi is that to construct a calculus for the combination $\mathbf{L}_{[1,2]}$ of two particular logics, no knowledge is needed about their interaction. Thus, a calculus for the combination $\mathbf{L}_{[1,2]}$ can be obtained quickly and easily. Soundness and completeness of the fibred calculus do not have to be proven but follow from soundness and completeness properties of the component calculi (Theorem 6.4.3).

It is also possible to fibre a calculus $\mathcal{C}_1$ for a logic $\mathbf{L}_1$ with a calculus $\mathcal{C}_2$ for a "sub-logic" $\mathbf{L}_2$ of $\mathbf{L}_1$ (for example, propositional logic is a sub-logic of predicate logic);

although $\mathcal{C}_1$ can handle the whole logic $\mathbf{L}_1$, the calculus $\mathcal{C}_2$ may be more efficient for formulae from $\mathbf{L}_2$ such that the fibred calculus $\mathcal{C}_{(1,2)}$ is more efficient than $\mathcal{C}_1$.

One cannot fibre just any tableau calculi in a uniform way. If nothing is known about the calculi, then it is not clear where to "plug in" the calculus for $\mathbf{L}_2$ into that for $\mathbf{L}_1$. Therefore, in this chapter, we only consider tableau calculi that are (syntactically)

– ideal,

and (semantically)

– have the strong soundness properties from Definition 3.5.8 (appropriateness of the set of tableau interpretations and soundness of expansion),
– have Strong Completeness Property 1 from Definition 3.5.10 (appropriateness of the set of tableau interpretations), and
– are strongly semantically analytic (Def. 3.5.16).

Calculi with these properties can be shown to be sufficiently "well-behaved", such that fibring is possible. Idealness of the component calculi is sufficient to define the fibred calculus (i.e., its syntax and semantics) in a uniform way. If, moreover, the component calculi have the semantical properties listed above, then the resulting fibred calculus is automatically sound and complete for the logic resulting from fibring the component logics. Moreover, it can be shown to have itself these semantical properties, which makes it possible to iterate the fibring of tableau calculi and, thus, to construct a calculus for the fully fibred logic $\mathbf{L}_{[1,2]}$.

As an example, in Section 6.5, a calculus for PL1 and a calculus for the modal logic $\widehat{\mathrm{K}}$ are fibred resulting in a calculus for modal predicate logic.

It may be only possible to use a fibred calculus for semi-deciding satisfiability of formulae in the fibred logik, i.e., a proof procedure based on the fibred calculus may only terminate for unsatisfiable input formulae—even if the component calculi can be used to decide satisfiability in the component logics. That, however, is not surprising because a fibred logic may be undecidable even if its components are decidable.

Related work includes (D'Agostino & Gabbay, 1996), where a method for fibring tableau calculi for substructural implication logics has been presented. In (**?**), a method is described for fibring tableau calculi for modal logics to construct calculi for multi-modal logics; it can be seen as an instance of the general framework presented in the following. In (Giunchiglia & Sebastiani, 1996), a deduction system for a modal logic is described that uses separate procedures for handling the classical propositional part and the modal part of formulae, respectively.

In (Pfalzgraf, 1991; Pfalzgraf & Stokkermans, 1994) the introduction of *logical fibrings* has been strongly motivated by the classical notion of fibre bundles and sheaves.

Concrete applications aim at modelling logical control of cooperating agents. Baader and Schulz (1995a; 1995b) use another variant of fibring for combining solution domains and constraint solvers for symbolic constraints.

The advantages of combining different logics are discussed in (de Rijke, 1997).

6.2 Fibring Logical Systems

In this section, the method of fibring logical systems is described in general and syntax and semantics of fibred logics are defined, based on syntax and semantics of its component logics.

Intuitively, to fibre two logics $\mathbf{L}_1$ and $\mathbf{L}_2$ means to consider a logic whose formulae are constructed from symbols and operators from both logics (Gabbay, 1996c; Gabbay, 1998). In a first step we consider a logic $\mathbf{L}_{(1,2)}$ where $\mathbf{L}_2$-formulae can occur inside $\mathbf{L}_1$-formulae but not vice versa.

The logic $\mathbf{L}_{[1,2]} \equiv \mathbf{L}_{[2,1]}$ that is the full combination of $\mathbf{L}_1$ and $\mathbf{L}_2$, where expressions from the two logics can be nested arbitrarily, can be handled by inductively repeating the construction described in the following. Similarly, it is possible to combine three or more logics.

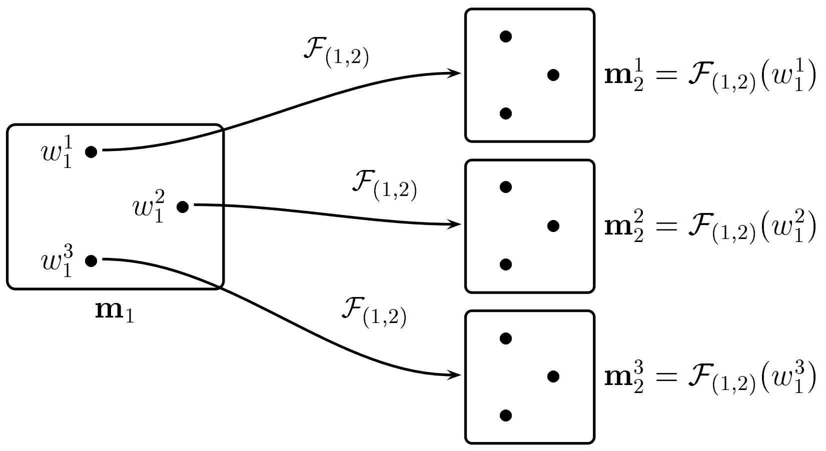
As we have defined logical systems in a very general way, there is no information on how formulae are constructed from signatures. We model the assumption that $\mathbf{L}_{(1,2)}$-formulae are (somehow) constructed using $\mathbf{L}_2$-formulae as subformulae of $\mathbf{L}_1$-formulae by considering the logic $\mathbf{L}_{(1,2)}$ to be a special version of $\mathbf{L}_1$ that contains the formulae of $\mathbf{L}_2$ as (additional) atoms. And, in each world w of the $\mathbf{L}_{(1,2)}$-models (which are enriched $\mathbf{L}_1$-models), the truth value of the additional atoms (which are $\mathbf{L}_2$-formulae) is the same as that in the initial world of an $\mathbf{L}_2$-model assigned to w. Thus, an $\mathbf{L}_{(1,2)}$-model consists of an $\mathbf{L}_1$-model $\mathbf{m}_1$ and a *fibring function* $\mathcal{F}$ that assigns to each world w in $\mathbf{m}_1$ an $\mathbf{L}_2$-model (as illustrated in Figure 6.1). Intuitively, when an $\mathbf{L}_2$-formula is to be evaluated in w, where its value is undefined, it is evaluated in $\mathbf{m}_2 = \mathcal{F}(w)$ instead.

Definition 6.2.1 Let $\mathbf{L}_1$ and $\mathbf{L}_2$ be logics; let a signature $\Sigma_{(1,2)} \in Sig_1$ be assigned to each pair $\langle \Sigma_1, \Sigma_2 \rangle \in Sig_1 \times Sig_2$ of signatures such that

$$Form_2(\Sigma_2) \subset Atom_1(\Sigma_{(1,2)}) \ .$$

The *fibred logical system* $\mathbf{L}_{(1,2)}$ that is the result of fibring the logics $\mathbf{L}_1$ and $\mathbf{L}_2$ is defined by:

- The set $Sig_{(1,2)}$ of signatures of $\mathbf{L}_{(1,2)}$ consists of all the signatures $\Sigma_{(1,2)} \in Sig_1$ assigned to a pair $\langle \Sigma_1, \Sigma_2 \rangle \in Sig_1 \times Sig_2$.



$\mathcal{F}_{(1,2)}$
$\mathcal{F}_{(1,2)}$
$\mathcal{F}_{(1,2)}$
w_1^1
w_1^2
w_1^3
$\mathbf{m}_1$
$\mathbf{m}_2^1 = \mathcal{F}_{(1,2)}(w_1^1)$
$\mathbf{m}_2^2 = \mathcal{F}_{(1,2)}(w_1^2)$
$\mathbf{m}_2^3 = \mathcal{F}_{(1,2)}(w_1^3)$

Figure 6.1: A fibred model.

- For all $\Sigma_{(1,2)} \in \mathit{Sig}_{(1,2)}$, the set $\mathit{Form}_{(1,2)}(\Sigma_{(1,2)})$ of $\mathbf{L}_{(1,2)}$-formulae over $\Sigma_{(1,2)}$ is identical to the set $\mathit{Form}_1(\Sigma_{(1,2)})$ of $\mathbf{L}_1$-formulae over $\Sigma_{(1,2)}$, and the set $\mathit{Atom}_{(1,2)}(\Sigma_{(1,2)})$ of $\mathbf{L}_{(1,2)}$-atoms over $\Sigma_{(1,2)}$ is identical to the set $\mathit{Atom}_1(\Sigma_{(1,2)})$ of $\mathbf{L}_1$-atoms over $\Sigma_{(1,2)}$.
- For all $\Sigma_{(1,2)} \in \mathit{Sig}_{(1,2)}$, the set $\mathcal{M}_{(1,2)}(\Sigma_{(1,2)})$ of $\mathbf{L}_{(1,2)}$-models consists of all pairs $\langle \mathbf{m}_1, \mathcal{F} \rangle$ where $\mathbf{m}_1 \in \mathcal{M}_1$ is an $\mathbf{L}_1$-model and $\mathcal{F}$ is a fibring function that assigns to each world w of $\mathbf{m}_1$ an $\mathbf{L}_2$-model $\mathbf{m}_2 \in \mathcal{M}_2(\Sigma_2)$ such that a formula $G \in \mathit{Form}_2(\Sigma_2)$ is true in w iff it is true in $\mathcal{F}(w)$, i.e.,

  $$w \models_1 G \text{ iff } \mathcal{F}(w) \models_2 G \ ;$$

  where $\models_1$ and $\models_2$ denote the truth relations between worlds and formulae in $\mathbf{L}_1$ and $\mathbf{L}_2$, respectively.

  The set $W_{(1,2)}$ of worlds of $\langle \mathbf{m}_1, \mathcal{F} \rangle$ consists of all worlds in $\mathbf{m}_1$ and all worlds in the $\mathbf{L}_2$-models assigned to worlds in $\mathbf{m}_1$ by $\mathcal{F}$. The initial world of $\langle \mathbf{m}_1, \mathcal{F} \rangle$ is the initial world of $\mathbf{m}_1$.

  The truth relation $\models_{(1,2)}$ of $\mathbf{L}_{(1,2)}$ is defined by $w \models_{(1,2)} G$ iff $w \models_1 G$ for all worlds w in $W_{(1,2)} = W_1$ and formulae G in $\mathit{Form}_{(1,2)}(\Sigma_{(1,2)}) = \mathit{Form}_1(\Sigma_{(1,2)})$. □

Example 6.2.2 To be able to fibre first-order predicate logic PL1 and a modal logic, we assume that for every PL1-signature Σ_{PL1} there is a signature $\Sigma_{\mathrm{mod}} \in \mathit{Sig}_{\mathrm{mod}}$ such that the atoms over Σ_{PL1} are the primitive propositions in Σ_{mod}. Then, $\Sigma_{(\mathrm{PL1},\mathrm{mod})}$ is a PL1-signature in which the predicate symbols are of the form $\circ_1 \cdots \circ_n p$ $(n \geq 0)$ where $\circ_i \in \{\Box, \Diamond, -\}$ and p is a predicate symbol in Σ_{PL1}.

If p and q are predicate symbols in Σ_{PL1} and a is a constant, then $p(a)$ is a primitive proposition in Σ_{mod}, $\Box p$ is a predicate symbol in $\Sigma_{(\mathrm{PL1},\mathrm{mod})}$, and $\Box p(a)$ is an element of both $\mathit{Atom}_{\mathrm{PL1}}(\Sigma_{(\mathrm{PL1},\mathrm{mod})})$ and $\mathit{Form}_{\mathrm{mod}}(\Sigma_{\mathrm{mod}})$.

Examples for formulae in $Form_{\mathrm{PL1}}(\Sigma_{(\mathrm{PL1},\mathrm{mod})})$ are

$$\begin{array}{l}\Box p(a),\\ (\forall x)(p(x)),\\ (\forall x)(\Box p(x)),\\ (\forall x)(\Box p(x)) \to (\exists x)(\Diamond q(x));\end{array}$$

but $\Box(\forall x)(p(x))$ is *not* a formula in $Form_{\mathrm{PL1}}(\Sigma_{(\mathrm{PL1},\mathrm{mod})})$ as modalities are only allowed to occur on the atomic level; and $\Box p(x)$ is neither a formula of $\mathbf{L}_{\mathrm{mod}}$ nor an atom of PL1 because free object variables are, by definition, not allowed to occur in formulae of PL1.

The fibred logic $\mathbf{L}_{(\mathrm{PL1},\mathrm{mod})}$ is a modal predicate logic, where the modal operators can only occur on the atomic level. If, however, the fibring process is iterated, then the result is a full modal predicate logic, because then the logical connectives $\vee$ and $\wedge$ of PL1 can be used inside modal formulae. $\Box$

6.3 Fibring Tableau Calculi

In this section, we describe how to construct in a uniform way a tableau calculus for a fibred logic $\mathbf{L}_{(1,2)}$ from two tableau calculi $\mathcal{C}_1$ and $\mathcal{C}_2$ for $\mathbf{L}_1$ resp. $\mathbf{L}_2$.[1]

If we take the view point that the search for a tableau proof is an attempt to construct a model for the formulae on the initial tableau, then the search for a tableau proof in the fibred calculus corresponds to a construction of a fibred model. It has therefore to be possible to represent knowledge about fibred models by tableau formulae of the fibred calculus. For that purpose, we use labels that are either of the form $\sigma_1 \in Lab_1$ denoting a world in the $\mathbf{L}_1$-model or of the form $(\sigma_1; \sigma_2)$ (where $\sigma_1 \in Lab_1$ and $\sigma_2 \in Lab_2$) denoting a world in the $\mathbf{L}_2$-model that is assigned by the fibring function to the world in the $\mathbf{L}_1$-model represented by σ_1. A tableau formula $\mathsf{T}{:}\sigma_1{:}G$ expresses that G is true in $I_1(\sigma_1)$; the meaning of a tableau formula $\mathsf{T}{:}(\sigma_1; \sigma_2){:}G$ is that G is true in the world $I_2(\sigma_2)$ of the model assigned to $I_1(\sigma_1)$ by the fibring function.

The combined calculus does not construct separate tableaux for $\mathbf{L}_1$ and $\mathbf{L}_2$ formulae, but there is only one unified structure with one unified tableau expansion rule.

The tableau expansion rule of the fibred calculus $\mathcal{C}_{(1,2)}$ constructed from $\mathcal{C}_1$ and $\mathcal{C}_2$ has three components:

1. the expansion rule of $\mathcal{C}_1$; it can be applied to $\mathbf{L}_1$-tableau formulae on a branch;

[1] Only ground calculi are considered in this chapter. To construct a rigid or universal variables calculus for a fibred logic $\mathbf{L}_{(1,2)}$, a ground calculus $\mathcal{C}^{\mathrm{gd}}_{(1,2)}$ has to be constructed first (using fibring), which then allows to construct a free variable calculus for $\mathbf{L}_{(1,2)}$ using the lifting techniques described in Chapter 4.

2. the expansion rule of $\mathcal{C}_2$; it can be applied to $\mathbf{L}_2$-tableau formulae with a label of the form $(\sigma_1; \sigma_2)$;

3. A fibring rule schema that allows to derive $\mathsf{S}{:}(\sigma_1; \sigma_2^0){:}G_2$ from $\mathsf{S}{:}\sigma_1{:}G_2$ if G_2 is an $\mathbf{L}_2$-formula and, thus, has to be handled by the $\mathcal{C}_2$-rule. This schema expresses the fact that, if an $\mathbf{L}_2$-formula G_2 is true in an $\mathbf{L}_1$-world $w = I_1(\sigma_1)$, then it is true in the initial world of the $\mathbf{L}_2$-model assigned to w.

The only additional assumption we have to make to be able to define syntax and semantics of the fibred calculus $\mathcal{C}_{(1,2)}$—besides idealness of the calculi $\mathcal{C}_1$ and $\mathcal{C}_2$ for the component logics—is that the extensions of signatures that are used in $\mathcal{C}_1$ and $\mathcal{C}_2$ are compatible.

Definition 6.3.1 Let the logic $\mathbf{L}_{(1,2)}$ be the result of fibring logics $\mathbf{L}_1$ and $\mathbf{L}_2$; and let $\mathcal{C}_1$ and $\mathcal{C}_2$ be calculi for $\mathbf{L}_1$ resp. $\mathbf{L}_2$ such that, for all signatures $\Sigma_1 \in \mathit{Sig}_1$ and $\Sigma_2 \in \mathit{Sig}_2$, there is an extension $\Sigma_2^{(*)}$ of Σ_2 with

– $\mathit{Form}_2(\Sigma_2^{(*)}) \subset \mathit{Atom}(\Sigma_{(1,2)}^*)$, and

– $\Sigma_2^* = (\Sigma_2^{(*)})^*$.

Then, the fibred calculus $\mathcal{C}_{(1,2)}$ for $\mathbf{L}_{(1,2)}$ is, for all $\Sigma_1 \in \mathit{Sig}_1, \Sigma_2 \in \mathit{Sig}_2$, defined by:

Labels: The set of labels of $\mathcal{C}_{(1,2)}$ is

$$\mathit{Lab}_{(1,2)} = \mathit{Lab}_1 \cup \{(\sigma_1; \sigma_2) \mid \sigma_1 \in \mathit{Lab}_1, \sigma_2 \in \mathit{Lab}_2\} \;;$$

the initial label $\sigma_{(1,2)}^0$ of $\mathcal{C}_{(1,2)}$ is identical to the initial label σ_1^0 of $\mathcal{C}_1$.

Expansion rule: The expansion rule $\mathcal{E}_{(1,2)}$ of $\mathcal{C}_{(1,2)}$ assigns the following possible conclusions to premisses $\Pi \subset \mathit{TabForm}_{(1,2)}(\Sigma_{(1,2)}^*)$ (where $\mathcal{E}_1$ and $\mathcal{E}_2$ are the expansion rules of $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively):

1. the conclusions in $\mathcal{E}_1(\Pi_1)$ where Π_1 consists of all tableau formulae ϕ in Π of the form $\phi = \mathsf{S}{:}\sigma_1{:}G$ *(expansion rule of $\mathcal{C}_1$),*

2. for all $\sigma_1 \in \mathit{Lab}_1$, the conclusions that can be constructed from the conclusions in $\mathcal{E}_2(\Pi_{2,\sigma_1})$ replacing σ_2 by $(\sigma_1; \sigma_2)$ where

$$\Pi_{2,\sigma_1} = \{\mathsf{S}{:}\sigma_2{:}G \mid \mathsf{S}{:}(\sigma_1; \sigma_2){:}G \in \Pi\}$$

(expansion rule of $\mathcal{C}_2$),

3. the conclusion $\{\{\mathsf{S}{:}(\sigma_1; \sigma_2^0){:}G\}\}$ for all tableau formulae ϕ in Π of the form $\phi = \mathsf{S}{:}\sigma_1{:}G$ whith $G \in \mathit{Form}_2(\Sigma_2^*)$ *(fibring rule schema).*

Tableau interpretations: The set $\mathit{TabInterp}_{(1,2)}$ of tableau interpretations of $\mathcal{C}_{(1,2)}$ consists of all pairs $\langle \mathbf{m}_{(1,2)}, I_{(1,2)} \rangle$, where $\mathbf{m}_{(1,2)} = \langle \mathbf{m}_1, \mathcal{F} \rangle$ is an $\mathbf{L}_{(1,2)}$-model, such that

1. there is a tableau interpretation $\langle \mathbf{m}_1, I_1 \rangle$ in $\mathit{TabInterp}_1(\Sigma_{(1,2)})$,
2. for all $\mathbf{L}_2$-models $\mathbf{m}_{2,w} = \mathcal{F}(w)$ that are assigned to worlds w of $\mathbf{m}_1$, there is a tableau interpretation $\langle \mathbf{m}_{2,w}, I_{2,w} \rangle$ in $\mathit{TabInterp}_2(\Sigma_2)$,
3. $I_{(1,2)}(\sigma_1) = I_1(\sigma_1)$ and $I_{(1,2)}((\sigma_1; \sigma_2)) = I_{2,I_1(\sigma_1)}(\sigma_2)$ for all $\sigma_1 \in \mathit{Lab}_1$ and $\sigma_2 \in \mathit{Lab}_2$. □

6.4 Semantical Properties of Fibred Calculi

As already mentioned in Section 6.1, we demand that a calculus that is to be used for fibring—besides being ideal—has the soundness and completeness properties from Definitions 3.5.8 and 3.5.10, and is strongly semantically analytic.

Definition 6.4.1 A calculus $\mathcal{C}$ is *suitable for fibring* if

– it is ideal,
– it has the strong soundness properties from Definition 3.5.8 (appropriateness of the set of tableau interpretations and soundness of expansion),
– it has Strong Completeness Property 1 from Definition 3.5.10 (appropriateness of the set of tableau interpretations), and
– it is strongly semantically analytic (Def. 3.5.16). □

Idealness of the component calculi $\mathcal{C}_1$ and $\mathcal{C}_2$ is sufficient for syntactically defining the fibred calculus $\mathcal{C}_{(1,2)}$. But $\mathcal{C}_{(1,2)}$ is only then *by construction* sound and complete (and suitable for fibring) if $\mathcal{C}_1$ and $\mathcal{C}_2$ have the semantical properties making them suitable for fibring (as defined above).

It is not sufficient for fibring if the component calculi only have the weak soundness and completeness properties (Def. 3.5.3 and 3.5.6). For example, if only satisfiability by *some* tableau interpretation is preserved when the expansion rule of a component calculus is applied, then the fibred calculus is not necessarily sound. Intuitively, the reason for this is the following: Suppose $T_{(1,2)}$ is a tableau for formulae of the fibred logic $\mathbf{L}_{(1,2)}$, the $\mathbf{L}_1$-tableau interpretation $\langle \mathbf{m}_1, I_1 \rangle$ satisfies the $\mathbf{L}_1$-part of $T_{(1,2)}$, and a world w in $\mathbf{m}_1$ is assigned an $\mathbf{L}_2$-model $\mathbf{m}_2$. Now, if the expansion rule of the component calculus for $\mathbf{L}_1$ only preserved satisfiability by *some* tableau interpretation, i.e., the $\mathbf{L}_1$-part of a successor tableau $T'_{(1,2)}$ of $T_{(1,2)}$ was only satisfied by some tableau interpretation $\langle \mathbf{m}'_1, I'_1 \rangle$ different from $\langle \mathbf{m}_1, I_1 \rangle$, then a problem would arise in case $\mathbf{m}_2$

and the world w' in $\langle \mathbf{m}'_1, I'_1 \rangle$ corresponding to w are not compatible, i.e., in case some $\mathbf{L}_2$-formula that is true in the initial world of $\mathbf{m}_2$ (and in w) is *not* true in w'.

The following example shows that a calculus has to be semantically analytic to be indeed suitable for fibring, as otherwise the fibred calculus may not be complete.

Example 6.4.2 Completeness of a calculus $\mathcal{C}_1$ for PL1 is preserved if a formula of the form $\mathsf{T}{:}(p \vee q)$ is not used for expansion of a branch B in case the atom q does not occur anywhere else on B (i.e., in case the atom q is *pure*). A calculus using this search space restriction is, however, *not* semantically analytic.

When $\mathcal{C}_1$ is fibred with a calculus $\mathcal{C}_2$ for a modal logic, then an PL1-atom may be an unsatisfiable modal formula. Thus, the expansion of formulae containing pure atoms may not be redundant; and a calculus for PL1 that is to be suitable for fibring must allow to expand a branch containing, for example, $\mathsf{T}{:}\sigma{:}[\Diamond(r \wedge \neg r) \vee q]$ by sub-branches containing $\mathsf{T}{:}\sigma{:}\Diamond(r \wedge \neg r)$ and $\mathsf{T}{:}\sigma{:}q$, respectively, such that $\mathsf{T}{:}\sigma{:}\Diamond(r \wedge \neg r)$ can be passed on to the modal component of the fibred calculus, and its unsatisfiability can be detected. $\square$

The following is the main theorem of this chapter.

Theorem 6.4.3 *Let the logic $\mathbf{L}_{(1,2)}$ be the result of fibring logics $\mathbf{L}_1$ and $\mathbf{L}_2$; let $\mathcal{C}_1$ and $\mathcal{C}_2$ be calculi for $\mathbf{L}_1$ resp. $\mathbf{L}_2$ that are suitable for fibring (Def. 6.4.1), and let the calculus $\mathcal{C}_{(1,2)}$ for $\mathbf{L}_{(1,2)}$ be the result of fibring $\mathcal{C}_1$ and $\mathcal{C}_2$ according to Definition 6.3.1.*

Then, the calculus $\mathcal{C}_{(1,2)}$ is suitable for fibring.

Proof: *Idealness:* By construction, $\mathcal{C}_{(1,2)}$ is a calculus with expansion rule, and it is easy to check that its expansion rule is monotonic and non-structural. Therefore, $\mathcal{C}_{(1,2)}$ is ideal.

Strong Soundness Property 1: Let a set $\mathfrak{F} \subset \mathit{Form}_{(1,2)}(\Sigma_{(1,2)})$ of formulae be satisfied by a model $\langle \mathbf{m}, \mathcal{F} \rangle \in \mathcal{M}_{(1,2)}(\Sigma_{(1,2)})$.

Since the calculus $\mathcal{C}_1$ has Strong Soundness Property 1, there is a tableau interpretation $\langle \mathbf{m}^*_1, I_1 \rangle \in \mathit{TabForm}_1(\Sigma^*_{(1,2)})$ that satisfies the initial tableau for $\mathfrak{F}$ such that $\mathbf{m}^*_1$ is an extension of $\mathbf{m}_1$. Now, for all worlds w of $\mathbf{m}^*_1$, let $\mathfrak{F}_w$ be the set of all formulae in $\mathit{Form}_2(\Sigma^*_2)$ that are true in w; $\mathfrak{F}_w$ is satisfied by the model $\mathbf{m}_{2,w} = \mathcal{F}(w)$. Because the calculus $\mathcal{C}_2$ has Strong Soundness Property 1, there is a tableau interpretation $\langle \mathbf{m}^*_{2,w}, I_{2,w} \rangle \in \mathit{TabForm}_2(\Sigma^*_{(1,2)})$ that satisfies the initial tableau for $\mathfrak{F}_w$ such that $\mathbf{m}^*_{2,w}$ is an extension of $\mathbf{m}_{2,w}$.

The tableau interpretation $\langle \langle \mathbf{m}^*_1, \mathcal{F}^* \rangle, I_{(1,2)} \rangle \in \mathit{TabInterp}_{(1,2)}(\Sigma^*_{(1,2)})$ where

– $\mathcal{F}^*(w) = \mathbf{m}^*_{2,w}$,

– $I_{(1,2)}(\sigma_1) = I_1(\sigma_1)$ and $I_{(1,2)}((\sigma_1;\sigma_2)) = I_{2,I_1(\sigma_1)}(\sigma_2)$ for all labels $\sigma_1 \in \mathit{Lab}_1$ and $\sigma_2 \in \mathit{Lab}_2$,

satisfies the initial tableau for $\mathfrak{F}$, and it is an extension of $\langle \mathbf{m}, \mathcal{F}\rangle$.

Strong Soundness Property 2: As the calculus $\mathcal{C}_{(1,2)}$ is ideal, we can use Lemma 3.5.9. Let $\Pi \subset \mathit{TabForm}_{(1,2)}(\Sigma^*_{(1,2)})$ be a minimal premiss of a conclusion C; and assume that Π is satisfied by a tableau interpretation $\langle\langle \mathbf{m}_1, \mathcal{F}\rangle, I_{(1,2)}\rangle \in \mathit{TabInterp}_{(1,2)}(\Sigma^*_{(1,2)})$. We show that $\langle\langle \mathbf{m}_1, \mathcal{F}\rangle, I_{(1,2)}\rangle$ satisfies one of the extensions in C by cases according to the form of Π.

If Π only contains formulae of the form $\mathsf{S}{:}\sigma_1{:}G$, i.e., C has been derived from Π using the expansion rule of $\mathcal{C}_1$, then one of the extensions in C is satisfied by $\langle\langle \mathbf{m}_1, \mathcal{F}\rangle, I_{(1,2)}\rangle$ because $\mathcal{C}_1$ has Strong Soundness Property 2.

If Π only contains formulae of the form $\mathsf{S}{:}(\sigma_1;\sigma_2){:}G$, i.e., C has been derived from Π using the expansion rule of $\mathcal{C}_2$, then one of the extensions in C is satisfied by the tableau interpretation $\langle\langle \mathbf{m}_1, \mathcal{F}\rangle, I_{(1,2)}\rangle$ because $\mathcal{C}_2$ has Strong Soundness Property 2.

If $\Pi = \{\mathsf{S}{:}\sigma_1{:}G\}$ where $G \in \mathit{Form}_2(\Sigma^*_2)$ and $C = \{\{\mathsf{S}{:}(\sigma_1;\sigma^0_2){:}G\}\}$, then the truth value of G is the same in the world $w_1 = I_{(1,2)}(\sigma_1)$ and the world $I_{(1,2)}(\sigma_1;\sigma^0_2)$, which is the initial world of $\mathcal{F}(w_1)$, and therefore $\langle\langle \mathbf{m}_1, \mathcal{F}\rangle, I_{(1,2)}\rangle$ satisfies $\mathsf{S}{:}(\sigma_1;\sigma^0_2){:}G$.

Strong Completeness Property 1: Let $\langle\langle \mathbf{m}^*_1, \mathcal{F}^*\rangle, I_{(1,2)}\rangle$ be a tableau interpretation satisfying an initial tableau T_1 for a set $\mathfrak{F} \subset \mathit{Form}_{(1,2)}(\Sigma_{(1,2)})$ of formulae.

The $\mathcal{C}_1$-tableau interpretation $\langle \mathbf{m}^*_1, I_1\rangle$ satisfies T_1 as well, where I_1 is the restriction of $I_{(1,2)}$ to labels in Lab_1. Since the calculus $\mathcal{C}_1$ has Strong Completeness Property 1, there is a model $\mathbf{m}_1 \in \mathcal{M}_1(\Sigma_{(1,2)})$ that is a restriction of $\mathbf{m}^*_1$ and satisfies $\mathfrak{F}$.

Now, for all worlds w of $\mathbf{m}^*_1$, let $\mathfrak{F}_w$ be the set of all tableau formulae $\sigma^0_2{:}\mathsf{T}{:}G$ such that $G \in \mathit{Form}_2(\Sigma^*_{(1,2)})$ is true in w; and let σ_1 be a label in Lab_1 such that $I_1(\sigma_1) = w$. The $\mathcal{C}_2$-tableau interpretation $\langle \mathcal{F}^*(w), I_2\rangle$ where $I_2(\sigma_2) = I_{(1,2)}((\sigma_1;\sigma_2))$ satisfies $\mathfrak{F}_w$. Since the calculus $\mathcal{C}_2$ has Strong Completeness Property 1, there is a model $\mathbf{m}_{2,w}$ in $\mathcal{M}_2(\Sigma_2)$ that is a restriction of $\mathcal{F}^*(w)$ and satisfies $\mathfrak{F}_w$. By construction, the model $\mathbf{m}_{2,w}$ satisfies all formulae G that are true in w.

Therefore, the model $\langle \mathbf{m}_1, \mathcal{F}\rangle$ of $\mathbf{L}_{(1,2)}$ satisfies $\mathfrak{F}$ and is a restriction of $\langle\langle \mathbf{m}^*_1, \mathcal{F}^*\rangle$, where $\mathcal{F}(w) = \mathbf{m}_{2,f(w)}$ for all worlds w in $\mathbf{m}_1$ (here $f(w)$ is the world in $\mathbf{m}^*_1$ corresponding to w).

The calculus is strongly semantically analytic: Let B be a fully expanded branch that is not closed; and let $\Phi_{(1,2)} \subset \mathit{TabForm}_{(1,2)}(\Sigma^*)$ be a set of atomic tableau formulae such that, for *no* ϕ in $\Phi_{(1,2)}$, both ϕ and $\overline{\phi}$ are in $\mathit{Form}(B) \cup \Phi_{(1,2)}$. We show that there is a tableau interpretation in $\mathit{TabInterp}_{(1,2)}(\Sigma_{(1,2)})$ satisfying B and $\Phi_{(1,2)}$.

For all $\sigma_1 \in \mathit{Lab}_1$, let B_{2,σ_1} be the sub-branch of B that consists of formulae of the form $\mathsf{S}{:}(\sigma_1;\sigma_2){:}G$. Since B_{2,σ_1} is a fully expanded non-closed $\mathcal{C}_2$-branch and the calculus $\mathcal{C}_2$ is strongly semantically analytic, there is a tableau interpretation $\langle \mathbf{m}_{2,\sigma_1}, I_{2,\sigma_1}\rangle$

in $\mathit{TabInterp}_{(1,2)}(\Sigma_{(1,2)})$ satisfying both B_{2,σ_1} and the set

$$\{\mathsf{S}{:}\sigma_2{:}G \mid \mathsf{S}{:}(\sigma_1;\sigma_2){:}G \in \Phi_{(1,2)}\}$$

of atomic tableau formulae.

Let B_1 be the sub-branch of B that consists of formulae of the form $S{:}\sigma_1{:}G$ where $\sigma_1 \in \mathit{Lab}_1$. Since B_1 is a fully expanded non-closed $\mathcal{C}_1$-branch and $\mathcal{C}_1$ is strongly semantically analytic, there is a tableau interpretation $\langle \mathbf{m}_1, I_1\rangle \in \mathit{TabInterp}_1(\Sigma_{(1,2)})$ satisfying both B_1 and the set of atomic tableau formulae of the form $\mathsf{S}{:}\sigma_1{:}G$ in $\Phi_{(1,2)}$.

Let the set $\Phi_1 \subset \mathit{TabForm}_1(\Sigma_{(1,2)})$ of atomic tableau formulae be defined by:

$$\Phi_1 = \{\mathsf{S}{:}\sigma_1{:}G \mid G \in \mathit{Form}_2(\Sigma_2^*), \text{ and } \mathsf{S}{:}\sigma_2{:}G \text{ is satisfied by } \langle \mathbf{m}_{2,\sigma_1}, I_{2,\sigma_1}\rangle\} \ .$$

By construction, the set Φ_1 cannot contain an atom ϕ and its complement $\overline{\phi}$; and the complement of $\phi = \mathsf{S}{:}\sigma_1{:}G \in \Phi$ cannot occur in $\mathit{Form}(B_1)$ because otherwise B_{2,σ_1} would contain the complement of $\mathsf{S}{:}\sigma_2{:}G$ (since B is fully expanded), which however is not possible as $\langle \mathbf{m}_{2,\sigma_1}, I_{2,\sigma_1}\rangle$ satisfies both $\mathsf{S}{:}\sigma_2{:}G$ and B_{2,σ_1}.

Therefore, since the calculus $\mathcal{C}_1$ is strongly semantically analytic, there is a tableau interpretation $\langle \mathbf{m}_1', I_1'\rangle \in \mathit{TabInterp}_1(\Sigma^*_{(1,2)})$ that satisfies both B_1 and Φ_1.

Now, we have that the tableau interpretation $\langle\langle \mathbf{m}_1', \mathcal{F}\rangle, I_{(1,2)}\rangle \in \mathit{TabInterp}_{(1,2)}(\Sigma^*_{(1,2)})$ with

- for all worlds w' of $\mathbf{m}_1'$, the $\mathbf{L}_2$-model assigned to w' is $\mathcal{F}(w') = \mathbf{m}_{2,\sigma_1}$ where σ_1 is an arbitrary label in Lab_1 such that $I_1'(\sigma_1) = w'$,
- $I_{(1,2)}(\sigma_1) = I_1'(\sigma_1)$ and $I_{(1,2)}((\sigma_1;\sigma_2)) = I_{2,\sigma_1}(\sigma_2)$ for all $\sigma_1 \in \mathit{Lab}_1$ and $\sigma_2 \in \mathit{Lab}_2$,

satisfies B. □

A calculus that is suitable for fibring has by definition properties that ensure its soundness and completeness (Theorems 3.5.4 and 3.5.7). Therefore, Theorem 6.4.3 implies the following corollary.

Corollary 6.4.4 *The calculus $\mathcal{C}_{(1,2)}$ is sound and complete.*

6.5 Example: Fibring Calculi for PL1 and a Modal Logic

As an example, we fibre the calculus $\mathcal{C}_{\mathrm{PL1}}$ for first-order predicate logic PL1 introduced in Section 3.6 and a calculus $\mathcal{C}_{\widehat{\mathrm{K}}}$ for the logic $\widehat{\mathrm{K}}$ of modalities (without binary connectives) defined in Section 2.5.[2] The result is a calculus $\mathcal{C}_{(1,2)}$ for first-order modal logic where the modal operators can only occur on the atomic level (Example 6.2.2).

[2] To distinguish negations in PL1 and in $\widehat{\mathrm{K}}$, the symbol $-$ is used in modal formulae instead of $\neg$.

The tableau expansion rule of the fibred calculus can easily be constructed by instantiating the calculi $\mathcal{C}_1$ and $\mathcal{C}_2$ in Definition 6.3.1 with $\mathcal{C}_{\mathrm{PL1}}$ and $\mathcal{C}_{\widehat{\mathrm{K}}}$, respectively. As an example we use a tableau proof for the validity of the formula

$$G \;=\; (\forall x)(\Box p(x)) \rightarrow [\neg(\exists y)(\Diamond{-p(y)}) \wedge \neg(\exists z)(\Diamond{-p(z)})]$$

in the logic $\mathbf{L}_{(1,2)} = \mathbf{L}_{(\mathrm{PL1},\widehat{\mathrm{K}})}$, using the calculus $\mathcal{C}_{(1,2)} = \mathcal{C}_{(\mathrm{PL1},\widehat{\mathrm{K}})}$ to construct a closed tableau for $\neg G$. The tableau is shown in Figure 6.2; it is constructed as follows: The tableau formula 1 is put on the tableau initially; then formulae 2–7 are added using the rule schemata of $\mathcal{C}_{\mathrm{PL1}}$ for α- and β-formulae. The schema of $\mathcal{C}_{\mathrm{PL1}}$ for δ-formulae is applied to derive 8 from 7, introducing the Skolem constant $c_1 = sko((\exists y)(\Diamond{-p(y)}))$. Now, since 8 contains an $\widehat{\mathrm{K}}$-formula, the fibring rule schema is applied to add 9 to the branch, which then allows to apply the $\mathcal{C}_{\widehat{\mathrm{K}}}$-expansion rule to derive 10 from 9 (we assume that $\lceil \Diamond{-p(c_1)} \rceil = 1$) and to derive 11 from 10. At this point, the rule schema of $\mathcal{C}_{\mathrm{PL1}}$ for γ-formulae is applied to 3 to derive 12, replacing the universally quantified object variable x with the ground term c_1 (which shows that $\mathcal{C}_1$- and $\mathcal{C}_2$-rule schemata can be applied in an arbitrary order). Finally, the fibring rule schema is applied to 12 to derive 13, and the rule schema of $\mathcal{C}_{\widehat{\mathrm{K}}}$ for ν-formulae is applied to derive 14. At this point, the left branch of the tableau is closed by the expansion rule schema for closing branches of $\mathcal{C}_{\widehat{\mathrm{K}}}$, because it contains the complementary atoms 11 and 14. The right branch is expanded and closed in the same way.

The full power of the fibring method is revealed when the fibring process is iterated to construct from $\mathcal{C}_{\mathrm{PL1}}$ and $\mathcal{C}_{\widehat{\mathrm{K}}}$ a calculus $\mathcal{C}_{[\mathrm{PL1},\widehat{\mathrm{K}}]}$ for the full modal predicate logic $\mathbf{L}_{[\mathrm{PL1},\widehat{\mathrm{K}}]}$; this is possible because the calculi $\mathcal{C}_{(1,2)}, \mathcal{C}_{(1,(2,1))}, \ldots$ are all suitable for fibring. As an example, we use $\mathcal{C}_{[\mathrm{PL1},\widehat{\mathrm{K}}]}$ to prove the validity of the formula $G' =$

$$(\forall x)(\Box(r(x) \wedge s(x))) \rightarrow [\neg(\exists y)(\Diamond({-r(y)} \vee {-s(y)})) \wedge \neg(\exists z)(\Diamond({-r(z)} \vee {-s(z)}))]$$

in $\mathbf{L}_{[\mathrm{PL1},\widehat{\mathrm{K}}]}$; it is constructed from G replacing the atom $p(x)$ by $r(x) \wedge s(x)$ and replacing $-p(y)$ and $-p(z)$ by $-r(y) \vee -s(y)$ resp. $-r(z) \vee -s(z)$. The construction of the tableau starts as above for G (Figure 6.2). We only consider the left branch (the right branch can be closed in the same way). Instead of the atoms 10 and 14, the branch now contains $10' = \mathsf{T}{:}(*;1.1){:}({-r(c_1)} \vee {-s(c_1)})$ and $14' = \mathsf{T}{:}(*;1.1){:}(r(c_1) \wedge s(c_1))$. The expansion of the branch continues as shown in Figure 6.3 (to simplify notation, we write $(*;1;*)$ instead of $(*;(1;*))$, etc.). The tableau formula $14'$ contains an PL1-formula. Therefore, the fibring rule schema is applied, and 23 is derived from $14'$; this is the fibring rule schema of the calculus $\mathcal{C}_{(\widehat{\mathrm{K}},\mathrm{PL1})}$ that, during the iteration process, has been fibred with $\mathcal{C}_{\mathrm{PL1}}$ to construct $\mathcal{C}_{(\mathrm{PL1},(\widehat{\mathrm{K}},\mathrm{PL1}))}$. The rule schema of $\mathcal{C}_{\mathrm{PL1}}$ for α-formulae is used to derive 24 and 25 from 23; then, 26 is derived from $10'$ by again applying the fibring rule schema, and the rule schema for β-formulae is applied to derive 27 and 31 from 26. The atom $-p(c_1)$ in 27 contains the modal and not the first-oder negation sign. Thus, the fibring rule schema has to be applied again to derive 28, which then allows to derive 29 by applying the schema for modal negation.

1 $\mathsf{T}{:}{*}{:}\neg((\forall x)(\Box p(x)) \to [\neg(\exists y)(\Diamond - p(y)) \wedge \neg(\exists z)(\Diamond - p(z))])$

2 $\mathsf{F}{:}{*}{:}(\forall x)(\Box p(x)) \to [\neg(\exists y)(\Diamond - p(y)) \wedge \neg(\exists z)(\Diamond - p(z))]$

3 $\mathsf{T}{:}{*}{:}(\forall x)(\Box p(x))$

4 $\mathsf{F}{:}{*}{:}\neg(\exists y)(\Diamond - p(y)) \wedge \neg(\exists z)(\Diamond - p(z))$

5 $\mathsf{F}{:}{*}{:}\neg(\exists y)(\Diamond - p(y))$	**6** $\mathsf{F}{:}{*}{:}\neg(\exists y)(\Diamond - p(y))$
7 $\mathsf{T}{:}{*}{:}(\exists y)(\Diamond - p(y))$	**15** $\mathsf{T}{:}{*}{:}(\exists y)(\Diamond - p(y))$
8 $\mathsf{T}{:}{*}{:}\Diamond - p(c_1)$	**16** $\mathsf{T}{:}{*}{:}\Diamond - p(c_1)$
9 $\mathsf{T}{:}({*};1){:}\Diamond - p(c_1)$	**17** $\mathsf{T}{:}({*};1){:}\Diamond - p(c_1)$
10 $\mathsf{T}{:}({*};1.1){:}-p(c_1)$	**18** $\mathsf{T}{:}({*};1.1){:}-p(c_1)$
11 $\mathsf{F}{:}({*};1.1){:}p(c_1)$	**19** $\mathsf{F}{:}({*};1.1){:}p(c_1)$
12 $\mathsf{T}{:}{*}{:}\Box p(c_1)$	**20** $\mathsf{T}{:}{*}{:}\Box p(c_1)$
13 $\mathsf{T}{:}({*};1){:}\Box p(c_1)$	**21** $\mathsf{T}{:}({*};1){:}\Box p(c_1)$
14 $\mathsf{T}{:}({*};1.1){:}p(c_1)$	**22** $\mathsf{T}{:}({*};1.1){:}p(c_1)$
$\perp$	$\perp$

Figure 6.2: A tableau proof for the validity of the formula G.

$$\begin{tikzcd}[column sep=small]
 & \textbf{10}' \ \mathsf{T}{:}(*;1.1){:}-r(c_1) \vee -s(c_1) \arrow[d, no head] & \\
 & \textbf{14}' \ \mathsf{T}{:}(*;1.1){:}r(c_1) \wedge s(c_1) \arrow[d, no head] & \\
 & \textbf{23} \ \mathsf{T}{:}(*;1.1;*){:}r(c_1) \wedge s(c_1) \arrow[d, no head] & \\
 & \textbf{24} \ \mathsf{T}{:}(*;1.1;*){:}r(c_1) \arrow[d, no head] & \\
 & \textbf{25} \ \mathsf{T}{:}(*;1.1;*){:}s(c_1) \arrow[d, no head] & \\
 & \textbf{26} \ \mathsf{T}{:}(*;1.1;*){:}-r(c_1) \vee -s(c_1) \arrow[dl, no head] \arrow[dr, no head] & \\
\textbf{27} \ \mathsf{T}{:}(*;1.1;*){:}-r(c_1) \arrow[d, no head] & & \textbf{31} \ \mathsf{T}{:}(*;1.1;*){:}-s(c_1) \arrow[d, no head] \\
\textbf{28} \ \mathsf{T}{:}(*;1.1;*;1){:}-r(c_1) \arrow[d, no head] & & \textbf{32} \ \mathsf{T}{:}(*;1.1;*;1){:}-s(c_1) \arrow[d, no head] \\
\textbf{29} \ \mathsf{F}{:}(*;1.1;*;1){:}r(c_1) \arrow[d, no head] & & \textbf{33} \ \mathsf{F}{:}(*;1.1;*;1){:}s(c_1) \arrow[d, no head] \\
\textbf{30} \ \mathsf{T}{:}(*;1.1;*;1){:}r(c_1) \arrow[d, no head] & & \textbf{34} \ \mathsf{T}{:}(*;1.1;*;1){:}s(c_1) \arrow[d, no head] \\
\perp & & \perp
\end{tikzcd}$$

Figure 6.3: The continuation of the tableau for G'.

The atomic tableau formulae 24 and 29 cannot be used to close the branch, because their labels are different. Thus, the fibring rule schema is applied a last time to derive 30 from 24. Then, the branch is closed applying the schema for closing branches to 29 and 30.

7 Theory Reasoning

7.1 Overview

Theory reasoning is an important technique for increasing the efficiency of automated deduction systems, that is well-known from theorem proving in first-order predicate logic. The specific knowledge from a given domain (or theory) is made use of by applying efficient methods for reasoning in that domain.

Theory reasoning is very important for automated deduction in real world domains. Equality theory, for example, is frequently used, but most specifications of real world problems use other theories as well: algebraic theories in mathematical problems and specifications of abstract data types in software verification to name a few.

Following the pioneering work of Stickel (1985), theory reasoning methods have been described for various types of calculi for first-order predicate logic; e.g. resolution (Stickel, 1985; Policriti & Schwartz, 1995), path resolution (Murray & Rosenthal, 1987b), the connection method (Petermann, 1992; Petermann, 1993), model elimination (Baumgartner, 1992), connection tableaux (Baumgartner *et al.*, 1992; Furbach, 1994), the matrix method (Murray & Rosenthal, 1987a).

The abstract model usually used to characterise an automated deduction system that employs theory reasoning techniques is that the general purpose *foreground reasoner* calls a special purpose *background reasoner* to handle problems from a certain theory. In the case of tableau-based theorem proving, the foreground reasoner is (an implementation of) a tableau calculus, whereas the background reasoner is an arbitrary algorithm that, when provided with a premiss Π, computes a conclusion or a set of conclusions of Π.

This model fits perfectly within our framework. There is no need to introduce new notions for theory reasoning or to define conditions that the conclusions computed by a background reasoner must satisfy to establish soundness and completeness of the resulting calculus for a theory $\mathcal{T}$ and a logic $\mathbf{L}$ (as is usually done in the literature on tableau-based theory reasoning, see (Beckert, 1998a)). Instead, we take the viewpoint that a theory $\mathcal{T}$ defines a new logical system $\mathbf{L}_{\mathcal{T}}$. This logic $\mathbf{L}_{\mathcal{T}}$ has the same syntax as the original logic $\mathbf{L}$; but the models of $\mathbf{L}_{\mathcal{T}}$ are only those models of $\mathbf{L}$ that satisfy $\mathcal{T}$. Then, all the notions and methods from the previous chapters can be applied without the need to define special theory reasoning versions. A background reasoner is assumed to be a procedure or algorithm for computing conclusions such that the resulting calculus is sound and complete for the logic $\mathbf{L}_{\mathcal{T}}$. No special soundness and completeness criteria have to be defined for theory reasoning, but the techniques for proving soundness and completeness described in Chapter 3 can be used. In addition, all methods for improving the efficiency of a tableau-based proof procedure from Chapters 4 and 5 can be applied, including the rigid and universal variable techniques.

Background reasoners have been designed for various theories, in particular for equality reasoning; an overview can be found in (Baumgartner *et al.*, 1992; Furbach, 1994); for set theory in (Cantone *et al.*, 1989). Reasoning in single models, e.g. natural numbers, is discussed in (Bürckert, 1990).

7.2 Theories

We define any satisfiable set of formulae to be a theory.

Definition 7.2.1 Let $\mathbf{L}$ be a logic; and let $\Sigma \in \mathit{Sig}$ be a signature. Then, a satisfiable set $\mathcal{T}(\Sigma) \subset \mathit{Form}(\Sigma)$ of formulae is a *theory*.

The logical system $\mathbf{L}_{\mathcal{T}}$ is identical to $\mathbf{L}$ except that, for all signatures $\Sigma \in \mathit{Sig}$, its set $\mathcal{M}_{\mathcal{T}}(\Sigma)$ of models only contains those models of $\mathbf{L}(\Sigma)$ that satisfy $\mathcal{T}(\Sigma)$. $\square$

In the literature, often the additional condition (besides satisfiability) is imposed on theories that they are closed under the logical consequence relation. Without that restriction we do not have to distinguish between a theory and its defining set of axioms.

Definition 7.2.2 A theory $\mathcal{T}(\Sigma)$ is *(finitely) axiomatisable* if there is a (finite) decidable set $\Psi \subset \mathit{Form}(\Sigma)$ of formulae (the axioms) such that: $\phi \in \mathit{Form}(\Sigma)$ is satisfied by all models of $\mathbf{L}_{\mathcal{T}}(\Sigma)$ if and only if ϕ is satisfied by all models of $\mathbf{L}(\Sigma)$ that satisfy Ψ. $\square$

Most theories that are of practical interest are axiomatisable. An example for a theory that is not axiomatisable is the set $\mathcal{T}$ of all satisfiable formulae in PL1. If a theory $\mathcal{T}$ is axiomatisable, then the set of unsatisfiable formulae of $\mathbf{L}_{\mathcal{T}}$ can be enumerated using a tableau calculus for $\mathbf{L}$.

Example 7.2.3 The most important PL1-theory in practice is the equality theory $\mathcal{T}_{\approx}$. It consists of the following axioms:

(1) $(\forall x)(x \approx x)$ (reflexivity),

(2) for all function symbols $f \in F(\Sigma)$:

$$(\forall x_1)\cdots(\forall x_n)(\forall y_1)\cdots(\forall y_n)((x_1 \approx y_1 \wedge \ldots \wedge x_n \approx y_n) \to f(x_1,\ldots,x_n) \approx f(y_1,\ldots,y_n))$$

where $n = \alpha_\Sigma(f)$ (monotonicity for function symbols),

Premiss	Conclusions
$\{\mathsf{F}{:}(a \approx a)\}$	$\langle\{\{\bot\}\}\quad, id\rangle$
$\{\mathsf{F}{:}(X \approx a)\}$	$\langle\{\{\bot\}\}\quad, \{X \mapsto a\}\rangle$
$\{\mathsf{F}{:}(\boldsymbol{x} \approx a)\}$	$\langle\{\{\bot\}\}\quad, id\rangle$
$\{\mathsf{T}{:}p(a), \mathsf{F}{:}p(b)\}$	$\langle\{\{\mathsf{F}{:}(a \approx b)\}\}\quad, id\rangle$
$\{\mathsf{T}{:}p(f(a), f(b)), \mathsf{T}{:}(f(X) \approx X)\}$	$\langle\{\{\mathsf{T}{:}p(a, f(b))\}\}, \{X \mapsto a\}\rangle$ $\langle\{\{\mathsf{T}{:}p(f(a), b)\}\}, \{X \mapsto b\}\rangle$
$\{p(f(a), f(b)), f(\boldsymbol{x}) \approx \boldsymbol{x}\}$	$\langle\{\{\mathsf{T}{:}p(a, b)\}\}\quad, id\rangle$

Table 7.1: Examples for premisses and their conclusions using equality theory.

(3) for all predicate symbols $p \in P(\Sigma)$:

$$(\forall x_1) \cdots (\forall x_n)(\forall y_1) \cdots (\forall y_n)((x_1 \approx y_1 \wedge \ldots \wedge x_n \approx y_n) \rightarrow \\ (p(x_1, \ldots, x_n) \rightarrow p(y_1, \ldots, y_n)))$$

where $n = \alpha_\Sigma(p)$ (monotonicity for predicate symbols),

Symmetry and transitivity of $\approx$ are implied by reflexivity (1) and monotonicity for predicate symbols (3) (observe that $\approx \in P(\Sigma)$). □

Example 7.2.4 The PL1-theory $\mathcal{T}_<$ of partial orderings consists of the axioms

(1) $(\forall x)\neg(x < x)$ (anti-reflexivity),

(2) $(\forall x)(\forall y)(\forall z)((x < y) \wedge (y < z) \rightarrow (x < z))$ (transitivity).

The theory $\mathcal{T}_<$ is finite; contrary to the equality theory, it does not contain monotonicity axioms. □

7.3 Examples for Background Reasoners

Table 7.1 shows some examples for conclusions a sound and complete background reasoner for the equality theory $\mathcal{T}_\approx$ may compute; the premisses and conclusions contain both rigid and universal variables.

Using the universal variable technique is of great importance for theory reasoning as the following example illustrates:

Example 7.3.1 Suppose a tableau branch contains the equality $\mathsf{T}{:}(f(\boldsymbol{x}) \approx \boldsymbol{x})$ and the atoms $\mathsf{T}{:}p(f(a), f(b))$ and $\mathsf{F}{:}p(a, b)$. In that case, the conclusion $\langle\{\{\bot\}\}, id\rangle$ can be derived, and the branch can be closed immediately.

$$\frac{\begin{array}{c} t < t' \\ t' < t'' \end{array}}{t < t''} \qquad \frac{t < t}{\bot}$$

for all $t, t', t'' \in \mathit{Term}^0_{\mathrm{PL1}}(\Sigma^*)$ — for all $t \in \mathit{Term}^0_{\mathrm{PL1}}(\Sigma^*)$

Table 7.2: Additional expansion rule schemata for the theory of partial orderings (Example 7.3.2).

In a rigid variable tableau where a similar branch contains the equality $\mathsf{T}{:}(f(X) \approx X)$ instead of $\mathsf{T}{:}(f(\boldsymbol{x}) \approx \boldsymbol{x})$, which allows only to derive the possible conclusions

$$\langle\{\{\mathsf{T}{:}(f(b) \approx b)\}\}, \{X \mapsto a\}\rangle \text{ and } \langle\{\{\mathsf{T}{:}(f(a) \approx a)\}\}, \{X \mapsto b\}\rangle \ ,$$

the branch cannot be closed (immediately). □

Example 7.3.2 A sound and complete (ground) calculus for the logic $\mathrm{PL1}_{\mathcal{T}_>}$, i.e., for the logical system that results from adding the theory of partial orderings (Example 7.2.4) to first-order predicate logic, can be constructed by extending the expansion rule of the ground calculus $\mathcal{C}_{\mathrm{PL1}}$ from Section 3.6 by the two expansion rule schemata shown in Table 7.2. That is, for all premisses $\Pi \in \mathit{TabForm}_{\mathrm{PL1}}(\Sigma^*)$, the set $\mathcal{E}_{\mathcal{T}_>}(\Sigma)(\Pi)$ of possible conclusions is the smallest set containing

– all conclusions in $\mathcal{E}_{\mathrm{PL1}}(\Sigma)(\Pi)$,
– the conclusion $\{\{\mathsf{T}{:}(t < t'')\}\}$ for all terms $t, t'' \in \mathit{Term}^0_{\mathrm{PL1}}(\Sigma^*)$ such that there are formulae $\mathsf{T}{:}(t < t')$ and $\mathsf{T}{:}(t' < t'')$ in Π for some $t' \in \mathit{Term}^0_{\mathrm{PL1}}(\Sigma^*)$,
– the conclusion $\{\{\bot\}\}$ if $\mathsf{T}{:}(t < t) \in \Pi$ for some term $t \in \mathit{Term}^0_{\mathrm{PL1}}(\Sigma^*)$. □

Bibliography

AHRENDT, WOLFGANG, BECKERT, BERNHARD, & STENZ, GERNOT. 1997. Search-oriented vs. Representation-oriented Calculi. *In: Proceedings, International Workshop on Proof Transformation and Presentation*.

ANDREWS, PETER B. 1981. Theorem Proving through General Matings. *Journal of the ACM*, **28**, 193–214.

BAADER, F., & SCHULZ, K. 1995a. Combination of Constraints Solving Techniques, and Algebraic Point of View. *In: Proceedings, RTA-95*. LNCS 914. Springer.

BAADER, F., & SCHULZ, K. 1995b. On the Combination of Symbolic Constraints, Solution Domains and Constraint Solvers. *In: Proceedings, CP-95*. LNCS 976.

BAUMGARTNER, PETER. 1992. A Model Elimination Calculus with Built-in Theories. *Pages 30–42 of:* OHLBACH, H.-J. (ed), *Proceedings, German Workshop on Artificial Intelligence (GWAI)*. LNCS 671. Springer.

BAUMGARTNER, PETER. 1998. *Fairness Strategies in Hyper Tableaux*. Talk given as part of the Tele Seminar *Automated Theorem Proving and Applications* jointly held at the University of Karlsruhe and the University of Koblenz.

BAUMGARTNER, PETER, FURBACH, ULRICH, & PETERMANN, UWE. 1992. *A Unified Approach to Theory Reasoning*. Forschungsbericht 15/92. University of Koblenz.

BECKERT, BERNHARD. 1998a. Equality and Other Theories. *In:* D'AGOSTINO, M., GABBAY, D., HÄHNLE, R., & POSEGGA, J. (eds), *Handbook of Tableau Methods*. Kluwer. To appear.

BECKERT, BERNHARD. 1998b. Rigid E-Unification. *In:* BIBEL, WOLFGANG, & SCHMITT, PETER H. (eds), *Automated Deduction – A Basis for Applications*, vol. I. Kluwer, Dordrecht. To appear.

BECKERT, BERNHARD, & GABBAY, DOV. 1998. Fibring Semantic Tableaux. *Pages 77–92 of: Proceedings, International Conference on Theorem Proving with Analytic Tableaux and Related Methods, Oisterwijk, The Netherlands*. LNCS 1397. Springer.

BECKERT, BERNHARD, & GORÉ, RAJEEV. 1997. Free Variable Tableaux for Propositional Modal Logics. *Pages 91–106 of: Proceedings, International Conference on Theorem Proving with Analytic Tableaux and Related Methods, Pont-a-Mousson, France.* LNCS 1227. Springer.

BECKERT, BERNHARD, & HÄHNLE, REINER. 1992. An Improved Method for Adding Equality to Free Variable Semantic Tableaux. *Pages 507–521 of:* KAPUR, DEPAK (ed), *Proceedings, 11th International Conference on Automated Deduction (CADE), Saratoga Springs, NY, USA.* LNCS 607. Springer.

BECKERT, BERNHARD, & HÄHNLE, REINER. 1998. Analytic Tableaux. *In:* BIBEL, WOLFGANG, & SCHMITT, PETER H. (eds), *Automated Deduction — A Basis for Applications*, vol. I: Foundations. Kluwer, Dordrecht. To appear.

BECKERT, BERNHARD, & HARTMER, ULRIKE. 1998. A Tableau Calculus for Quantifier-free Set Theoretic Formulae. *Pages 93–107 of: Proceedings, International Conference on Theorem Proving with Analytic Tableaux and Related Methods, Oisterwijk, The Netherlands.* LNCS 1397. Springer.

BECKERT, BERNHARD, & POSEGGA, JOACHIM. 1995. leanT^AP: Lean Tableau-based Deduction. *Journal of Automated Reasoning*, **15**(3), 339–358.

BECKERT, BERNHARD, HÄHNLE, REINER, & SCHMITT, PETER H. 1993. The Even More Liberalized δ-Rule in Free Variable Semantic Tableaux. *Pages 108–119 of:* GOTTLOB, G., LEITSCH, A., & MUNDICI, D. (eds), *Proceedings, 3rd Kurt Gödel Colloquium (KGC), Brno, Czech Republic.* LNCS 713. Springer.

BETH, EVERT W. 1955. Semantic entailment and formal derivability. *Mededelingen van de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde, N.R.*, **18**(13), 309–342. Reprinted as pages 262–266 of: Karel Berka and Lothar Kreiser, editors. *Logik-Texte. Kommentierte Auswahl zur Geschichte der modernen Logik.* Akademie-Verlag, Berlin, 1986.

BETH, EVERT W. 1959. *The Foundations of Mathematics.* Amsterdam: North-Holland.

BIBEL, WOLFGANG. 1982. *Automated Theorem Proving.* Vieweg, Braunschweig.

BOWEN, K. A. 1982. programming with Full First-order Logic. *Machine Intelligence*, **10**, 421–440. *First published as:* Technical Report 6-80, Syracuse University, Syracuse, NY, USA, 1980.

BRODA, KRYSIA. 1980. The Relationship between Semantic Tableaux and Resolution Theorem Proving. *In: Proceedings, Workshop on Logic, Debrecen, Hungary.* Also as technical report, Imperial College, Department of Computing, London, UK.

BROWN, FRANK MALLOY. 1978. Towards the Automation of Set Theory and its Logic. *Artificial Intelligence*, **10**, 281–316.

BÜRCKERT, H. 1990. A Resolution Principle for Clauses with Constraints. *Pages 178–192 of: Proceedings, 10th International Conference on Automated Deduction (CADE)*. LNCS 449. Springer.

CANTONE, DOMENICO. 1991. Decision Procedures for Elementary Sublanguages of Set Theory. X. *Journal of Automated Reasoning*, **7**, 193–230.

CANTONE, DOMENICO. 1997. A Fast Saturation Strategy for Set-theoretic Tableaux. *Pages 122–137 of: Proceedings, TABLEAUX, Pont-a-Mousson, France*. LNCS 1227. Springer.

CANTONE, DOMENICO, & FERRO, ALFREDO. 1995. Techniques of Computable Set Theory with Applications to Proof Verification. *Comm. on Pure and Applied Mathematics*, **48**, 901–946.

CANTONE, DOMENICO, & SCHWARTZ, T. J. 1991. Decision Procedures for Elementary Sublanguages of Set Theory. XI. *Journal of Automated Reasoning*, **7**, 231–256.

CANTONE, DOMENICO, FERRO, ALFREDO, & SCHWARTZ, T. J. 1985. Decision Procedures for Elementary sublanguages of Set Theory. VI. *Comm. on Pure and Applied Mathematics*, **38**, 549–571.

CANTONE, DOMENICO, FERRO, ALFREDO, & SCHWARTZ, T. J. 1987. Decision Procedures for Elementary Sublanguages of Set Theory. V. *J. of Computer and Syst. Sciences*, **34**, 1–18.

CANTONE, DOMENICO, FERRO, ALFREDO, & OMODEO, EUGENIO. 1989. *Computable Set Theory*. International Series of Monographs on Computer Science, vol. 6. Oxford University Press.

CARNIELLI, WALTER. 1987. Systematization of Finite Many-valued Logics through the Method of Tableaux. *Journal of Symbolic Logic*, **52**(2), 473–493.

COHEN, J., TRILLING, L., & WEGNER, P. 1974. A Nucleus of a Theorem Prover Described in ALGOL-68. *International Journal of Computer and Information Sciences*, **3**(1), 1–31.

D'AGOSTINO, MARCELLO, & GABBAY, DOV. 1996. Fibred Tableaux for Multi-implication Logics. *Pages 16–35 of:* MIGLIOLI, P., MOSCATO, U., MUNDICI, D., & ORNAGHI, M. (eds), *Proceedings, 5th International Workshop on Theorem Proving with Analytic Tableaux and Related Methods (TABLEAUX), Terrasini, Palermo, Italy*. LNCS 1071. Springer.

D'AGOSTINO, MARCELLO, GABBAY, DOV, HÄHNLE, REINER, & POSEGGA, JOACHIM (eds). 1998. *Handbook of Tableau Methods*. Kluwer, Dordrecht. To appear.

DE NIVELLE, HANS. 1997 (Feb.). *Implementation of Sequent Calculus and Set Theory*. Draft.

DE RIJKE, MAARTEN. 1997. Why Combine Logics? *Studia Logica*, **59**(1), 5–27.

DROSDOWSKI, GÜNTHER, MÜLLER, WOLFGANG, SCHOLZE-STUBENRECHT, WERNER, & WERMKE, MATTHIAS (eds). 1991. *Duden. Rechtschreibung der Deutschen Sprache*. 20th edn. Mannheim: Dudenverlag.

EGLY, UWE. 1998. Cuts in Tableaux. *In:* BIBEL, WOLFGANG, & SCHMITT, PETER H. (eds), *Automated Deduction – A Basis for Applications*, vol. I. Kluwer, Dordrecht. To appear.

FERRO, ALFREDO, & OMODEO, EUGENIO. 1987. Decision procedures for Elementary Sublanguages of Set Theory. VII. *Communications on Pure and Applied Mathematics*, **40**, 265–280.

FITTING, MELVIN. 1998. Introduction. *Chap. 1 of:* M., GABBAY, D., HÄHNLE, R., & POSEGGA, J. (eds), *Handbook of Tableau Methods*. Kluwer, Dordrecht. To appear.

FITTING, MELVIN C. 1969. *Intuitionistic Logic Model Theory and Forcing*. Amsterdam: North-Holland.

FITTING, MELVIN C. 1983. *Proof Methods for Modal and Intuitionistic Logics*. Synthese Library, vol. 169. Dordrecht: Reidel.

FITTING, MELVIN C. 1990. *First-Order Logic and Automated Theorem Proving*. New York: Springer. Second edition published in 1996.

FURBACH, ULRICH. 1994. Theory Reasoning in First Order Calculi. *Pages 139–156 of:* V. LUCK, K., & MARBURGER, H. (eds), *Proceedings, Third Workshop on Information Systems and Artificial Intelligence, Hamburg, Germany*. LNCS 777. Springer.

GABBAY, DOV. 1996a. Fibred Semantics and the Weaving of Logics. Part 1: Modal and Intuitionistic Logics. *Journal of Symbolic Logic*, **61**, 1057–1120.

GABBAY, DOV. 1996b. *Labelled Deductive Systems*. Oxford Logic Guides, no. 33. Oxford University Press.

GABBAY, DOV. 1996c. An Overview of Fibred Semantics and the Combination of Logics. *Pages 1–55 of:* BAADER, F., & SCHULZ, K. (eds), *Proceedings, Frontiers of Combining Systems*. Kluwer, Dordrecht.

GABBAY, DOV. 1998. *Fibring Logic*. Oxford University Press. Forthcoming.

GABBAY, DOV, & GOVERNATORI, GUIDO. 1997. *Fibred Modal Tableaux*. Submitted to TABLEAUX'98.

GENTZEN, GERHARD. 1935. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, **39**, 176–210, 405–431. English translation in: Szabo, M., (ed.), *The Collected Papers of Gerhard Gentzen*, pages 68–131. North-Holland, Amsterdam, 1969.

GIUNCHIGLIA, FAUSTO, & SEBASTIANI, ROBERTO. 1996. A SAT-based Decision Procedure for $\mathcal{ALC}$. *Pages 304–314 of:* AIELLO, L., DOYLE, J., & SHAPIRO, S. (eds), *Proceedings, International Conference on Principals of Knowledge Representation and Reasoning (KR), Boston, USA*. Morgan Kaufmann.

GORÉ, RAJEEV. 1998. Tableau Methods for Modal and Temporal Logics. *Chap. 7 of:* M., GABBAY, D., HÄHNLE, R., & POSEGGA, J. (eds), *Handbook of Tableau Methods*. Kluwer, Dordrecht. To appear.

HÄHNLE, REINER. 1993. *Automated Deduction in Multiple-Valued Logics*. International Series of Monographs on Computer Science, vol. 10. Oxford University Press.

HÄHNLE, REINER, & KLINGENBECK, STEFAN. 1996. A-Ordered Tableaux. *Journal of Logic and Computation*, **6**(6), 819–834.

HÄHNLE, REINER, & PAPE, CHRISTIAN. 1997. Ordered Tableaux: Extensions and Applications. *Pages 173–187 of: Proceedings, International Conference on Theorem Proving with Analytic Tableaux and Related Methods, Pont-a-Mousson, France*. LNCS 1227. Springer.

HÄHNLE, REINER, & SCHMITT, PETER H. 1994. The Liberalized δ-rule in Free Variable Semantic Tableaux. *Journal of Automated Reasoning*, **13**(2), 211–222.

HÄHNLE, REINER, MURRAY, NEIL, & ROSENTHAL, ERIK. 1997. Completeness for Linear Regular Negation Normal Form Inference Systems. *In:* RAS, Z. (ed), *Proceedings, International Symposium on Methodologies for Intelligent Systems (ISMIS), Charlotte, NC, USA*. LNCS. Springer.

HARTMER, ULRIKE. 1997. *Erweiterung des Tableaukalküls mit freien Variablen um die Behandlung von Mengentheorie*. Diplomarbeit, Universität Karlsruhe.

HAYWARD, ARTHUR L., & SPARKES, JOHN J. 1968. *The Concise English Dictionary*. Fourth edn. Munich: Orbis Verlag.

HILBERT, DAVID, & BERNAYS, PAUL. 1939. *Grundlagen der Mathematik II*. Die Grundlehren der mathematischen Wissenschaften in Einzeldarstellungen mit besonderer Berücksichtigung der Anwendungsgebiete, vol. 50. Springer.

HINTIKKA, JAAKKO. 1955. Form and Content in Quantification Theory. *Acta Philosophica Fennica*, **8**, 7–55.

JECH, THOMAS. 1978. *Set Theory*. New York: Academic Press.

KANGER, STIG. 1957. *Provability in Logic*. Acta Universitatis Stockholmiensis, vol. 1. Almqvist & Wiksell, Stockholm.

KANGER, STIG. 1963. A Simplified Proof Method for Elementary Logic. *Pages 87–94 of:* BRAFFORT, P., & HIRSCHBERG, D. (eds), *Computer Programming and Formal Systems*. North Holland. Reprinted in (Siekmann & Wrightson, 1983, vol. 1, pages 364–371).

KORF, RICHARD E. 1985. Depth-First Iterative Deepening: An Optimal Admissible Tree Search. *Artificial Intelligence*, **27**, 97–109.

KRIPKE, SAUL. 1959. A Completeness Theorem in Modal Logic. *Journal of Symbolic Logic*, **24**(1), 1–14.

LIS, Z. 1960. Wynikanie semantyczne a wynikanie formalne (logical consequence, semantic and formal). *Studia Logica*, **10**, 39–60. Polish, with Russian and English abstracts.

MATSUMOTO, K., & OHNISHI, M. 1957. Gentzen Method in Modal Calculi I. *Osaka Mathematical Journal*, **9**, 113–130.

MATSUMOTO, K., & OHNISHI, M. 1959. Gentzen Method in Modal Calculi II. *Osaka Mathematical Journal*, **11**, 115–120.

MEYER VIOL, WILFRIED. 1995. *Instantiational Logic: An Investigation into Reasoning with Instances. ILLC Dissertation Series 1995-11*. Ph.D. thesis, University of Utrecht.

MURRAY, NEIL V., & ROSENTHAL, ERIC. 1987a. Inference with path resolution and semantic graphs. *Journal of the ACM*, **34**(2), 225–254.

MURRAY, NEIL V., & ROSENTHAL, ERIC. 1987b. Theory Links: Applications to Automated Theorem Proving. *Journal of Symbolic Computation*, **4**, 173–190.

OPPACHER, F., & SUEN, E. 1988. HARP: A Tableau-Based Theorem Prover. *Journal of Automated Reasoning*, **4**, 69–100.

PASTRE, D. 1978. Automatic Theorem Proving in set theory. *J. of AI*, **10**, 1–27.

PETERMANN, UWE. 1992. How to Build-in an Open Theory into Connection Calculi. *Journal on Computer and Artificial Intelligence*, **11**(2), 105–142.

PETERMANN, UWE. 1993. *Building-in a Theory into a Connection Calculus with Positive Refinement*. Preprint Nr. 25. Naturwissenschaftlich-Theoretisches Zentrum, Universität Leipzig.

PFALZGRAF, J. 1991. Logical Fiberings and Polycontextural Systems. *In:* JORRAND, PH., & KELEMEN, J. (eds), *Fundamentals of Artificial Intelligence Resarch*. LNCS 535. Springer.

PFALZGRAF, J., & STOKKERMANS, K. 1994. On Robotics Scenarios and Modeling with Fibered Structures. *In:* PFALZGRAF, J., & WANG, D. (eds), *Automated Practical Reasoning: Algebraic Approaches*. Texts and Monographs in Symbolic Computation. Springer.

POLICRITI, ALBERTO, & SCHWARTZ, JACOB T. 1995. T-Theorem Proving I. *Journal of Symbolic Computation*, **20**, 315–342.

POPPLESTONE, R. J. 1967. Beth-Tree Methods in Automatic Theorem Proving. *Pages 31–46 of:* COLLINS, N., & MICHIE, D. (eds), *Machine Intelligence*, vol. 1. Oliver and Boyd.

PRAWITZ, DAG. 1960. An Improved Proof Procedure. *Theoria*, **26**, 102–139. Reprinted in (Siekmann & Wrightson, 1983, vol. 1, pages 162–199).

PRAWITZ, DAG, PRAWITZ, HÅKAN, & VOGHERA, NERI. 1960. A Mechanical Proof Procedure and Its Realization in an Electronic Computer. *Journal of the ACM*, **7**(1–2), 102–128.

REEVES, STEVE V. 1987. Semantic Tableaux as a Framework for Automated Theorem-Proving. *Pages 125–139 of:* MELLISH, C., & HALLAM, J. (eds), *Advances in Artificial Intelligence (Proceedings of AISB-87)*. Wiley.

RESCHER, N., & URQUHART, A. 1971. *Temporal Logic*. Heidelberg: Springer.

ROUSSEAU, G. 1967. Sequents in Many-valued Logic I. *Fundamenta Methematica*, **60**, 23–33.

SCHMITT, PETER H. 1987. *The THOT Theorem Prover*. Tech. rept. 87.9.7. IBM Germany, Scientific Center, Heidelberg, Germany.

SCHÜTTE, KURT. 1956. Ein System des verknüpfenden Schließens. *Archiv für mathematische Logik und Grundlagenforschung*, **2**(2–4), 55–67.

SHULTS, BENJAMIN. 1997 (May). *Comprehension and Description in Tableaux*. Draft.

SIEKMANN, JÖRG, & WRIGHTSON, GRAHAM (eds). 1983. *Automation of Reasoning: Classical Papers in Computational Logic*. Springer.

SMULLYAN, RAYMOND M. 1968. *First-Order Logic*. Springer, Heidelberg. Second corrected edition published in 1995 by Dover Publications, New York.

STENZ, GERNOT. 1997. *Beweistransformation in Gentzenkalkülen*. Diplomarbeit, Universität Karlsruhe.

STICKEL, MARK E. 1985. Automated Deduction by Theory Resolution. *Journal of Automated Reasoning*, **1**, 333–355.

SUCHON, W. 1974. La méthode de Smullyan de construire le calcul n-valent des propositions de Łukasiewicz avec implication et négation. *Reports on Mathematical Logic*, **2**, 37–42.

SURMA, S. J. 1984. An Algortihm for Axiomatizing Every Finite Logic. *Pages 143–149 of:* RINE, D. C. (ed), *Computer Science and Multiple-valued Logic*, revised edn. Amsterdam: North Holland.

WANG, HAO. 1960. Toward Mechanical Mathematics. *IBM Journal of Research and Development*, **4**(1). Reprinted in (Siekmann & Wrightson, 1983, vol. 1, pages 244–264).

WEIDENBACH, CHRISTOPH. 1995. First-Order Tableaux with Sorts. *Journal of the IGPL*, **3**(6), 887–906.

WRIGHTSON, GRAHAM. 1984. *Semantic Tableaux, Unification and Links*. Technical Report CSD-ANZARP-84-001. Victoria University, Wellington, New Zealand.

List of Symbols

. . . but we need notions,
not notation.
— A. TARSKI, 1943

$\lceil\cdot\rceil$	Bijection from the set of modal formulae to $\mathbb{N}$, S. 51
Γ	set of labels, S. 49
Π	premiss (finite set of tableau formulae), S. 30
Σ^*_{fv}	extension of a signature Σ that introduces free variables, Def. 4.2.3, S. 84
Σ^*_{gd}	extension of a signature Σ that is ground, i.e., does not introduce free variables, Def. 4.2.3, S. 84
Σ^*	extension of a signature Σ, S. 25
Σ	signature, S. 9
α	α-formula, S. 42
β	β-formula, S. 42
δ	δ-formula, S. 42
γ	γ-formula, S. 42
ν	ν-formula, S. 51
$\overline{\phi}$	complement of the tableau formula ϕ, Def. 3.2.1, S. 25
ϕ, ψ	tableau formula, S. 25
π	π-formula, S. 51
$\sigma_{\|V}$	restriction of a substitution σ to a set V of variables, Def. 2.2.4, S. 12
$\sigma \circ \tau$	composition of substitutions σ and τ, Def. 2.2.4, S. 12
$[\sigma]$	equivalence class of labels identical to σ up to parentheses, Def. 3.7.1, S. 48
σ, τ	substitution, S. 12
$\Box$	box operator in modal logics, S. 17
$\Diamond$	diamond operator in modal logics, S. 17
$\wedge$	conjunction, S. 15
$\to$	implication, S. 15
$\neg$	negation, S. 15
$\vee$	disjunction, S. 15
$\exists$	existential quantification, S. 15

$\forall$	universal quantification, S. 15
$\lvert\sigma\rvert$	length of a label σ, Def. 3.7.1, S. 48
$\leq_w$	weight ordering, Def. 5.3.1, S. 142
$\rhd$	accessibility relation on the set of labels, Def. 3.7.3, S. 49
$\leq^W$	subsumption relation on rigid variable conclusions, Def. 4.2.11, S. 87
$\leq^W$	subsumption relation on substitutions, Def. 2.2.6, S. 14
$\models$	relation between worlds and formulae, Def. 2.1.1, S. 9
$\subseteq_k$	k-contains relation, Def. 5.2.4, S. 138
$\cap$	meta level set intersection, S. 21
$\sqcap$	object level set intersection, S. 21
$\cup$	meta level set union, S. 21
$\sqcup$	object level set union, S. 21
$\setminus$	set difference, S. 21
$=$	meta level equality, S. 21
$\approx$	object level equality, S. 21
$\in$	meta level membership, S. 21
⋿	object level membership, S. 21
$\{\cdot\}_n$	object level set constructor, S. 21
$\sqsubseteq$	object level set inclusion, S. 21
$\subset$	meta level set inclusion, S. 21
$\emptyset$	meta level empty set, S. 21
$\emptyset$	object level empty set, S. 21
$Atom$	set of all atomic formulae (atoms) of a logic, Def. 2.1.1, S. 9
B	tableau branch, S. 25
$CondLab(\mathbb{N})$	the set of conditional labels consisting of natural numbers, Def. 3.7.1, S. 48
C	conclusion, S. 28
$\mathcal{C}$	tableau calculus, S. 25
$dom(\sigma)$	the domain of the substitution σ, Def. 2.2.4, S. 12
E	extension (finite set of tableau formulae), S. 28
$\mathcal{E}$	expansion rule, S. 28
F, G	formula, S. 9
$\mathfrak{F}, \mathfrak{G}$	set of formulae, S. 9
$Form(B)$	formulae on a tableau branch B, Def. 3.2.1, S. 25
$Form$	set of all formulae of a logic, Def. 2.1.1, S. 9
$Form^0_{\mathrm{PL1}}$	set of all (ground) closed PL1-formulae (sentences), Def. 2.3.1, S. 15
$Form^{nc}$	set of all (ground) PL1-formulae possibly containing object variables not bound by a quantifier, Def. 2.3.1, S. 15
F	sign (falsehood), Def. 3.2.1, S. 25

$\mathcal{F}$	fibring function, Def. 6.2.1, S. 157
$\mathcal{G}$	the set of all set terms over Σ in a set Π of MLSS-tableau formulae over Σ^*, Def. 3.8.8, S. 68
$Lab(\mathbb{N})$	the set of labels consisting of natural numbers, Def. 3.7.1, S. 48
L	formal language, S. 11
$\mathbf{L}$	logical system (logic), S. 9
$\widehat{\mathbf{L}}$	modal logic **L** without binary connectives, S. 19
MLSSF	the logical system MLSSF, S. 21
MLSS	the logical system MLSS, S. 21
$\mathcal{M}$	set of all models of a logic, Def. 2.1.1, S. 9
$\mathbb{N}$	the set of natural numbers, S. 48
Ord	the class of all ordinals, Def. 2.6.4, S. 22
$\mathrm{PL1}$	first-order predicate logic, S. 15
$ran(\sigma)$	the range of the substitution σ, Def. 2.2.4, S. 12
R	reachability relation in Kripke frames, Def. 2.4.1, S. 17
$\mathcal{R}$	realisation of a set of MLSS tableau formulae, S. 69
$\mathcal{R}$	tableau rule, S. 25
Sig	set of all signatures of a logic, Def. 2.1.1, S. 9
$Subst$	set of all idempotent substitutions of a logic, Def. 2.2.4, S. 12
S	set of sorts, S. 11
$TabInterp$	set of tableau interpretations of a logic, Def. 3.4.2, S. 33
$TabTerm$	set of terms in a tableau formula, Def. 4.2.3, S. 84
$TabForm$	set of tableau formulae of a logic, Def. 3.2.1, S. 25
$Term^{\mathrm{fv}}$	set of all (non-ground) terms of a language, Def. 2.2.2, S. 11
$Term^{0}_{\mathrm{PL1}}$	set of all (ground) PL1-terms not containing object variables, Def. 2.3.1, S. 15
$Term^{nc}_{\mathrm{PL1}}$	set of all (ground) PL1-terms possibly containing object variables, Def. 2.3.1, S. 15
$Term$	set of all (ground) terms of a language, Def. 2.2.1, S. 11
$Term$	set of all (ground) terms of a logic, Def. 2.2.3, S. 12
T	sign (truth), Def. 3.2.1, S. 25
$\mathcal{T}$	certain set of constants, Def. 3.8.8, S. 68
T	tableau, S. 25
$\mathcal{T}$	theory, S. 168
$\mathcal{T}_{<}$	theory of partial orderings, S. 169
$\mathcal{T}_{\approx}$	equality theory, S. 168
V, W	set of free variables, S. 12
Var	set of all free variables, Def. 2.2.2, S. 11
$\mathcal{V}$	certain set of set terms, Def. 3.8.8, S. 68

$\mathfrak{V}$	von Neumann hierarchy of sets, Def. 2.6.4, S. 22
W	set of all worlds in a model, Def. 2.1.1, S. 9
X, Y, Z	free (rigid) variable, S. 12
fv	indicates the free variable version of a calculus, a free variable signature, etc., S. 84
id	the empty substitution, Def. 2.2.4, S. 12
ipr	set of initial prefixes of a label, Def. 3.7.1, S. 48
mv	indicates the mixed variable version of a calculus, an expansion rule, etc., S. 84
$\mathbf{m}$	model, S. 9
n, m	natural number, S. 48
$[n]$	conditional or unconditional position in label, S. 49
(n)	conditional position in a label, S. 48
rv	indicates the rigid variable version of a calculus, an expansion rule, etc., S. 84
sko_{fv}	function assigning Skolem terms to free variable formulae, Def. 4.2.28, S. 96
sko	function assigning Skolem terms to formulae, Def. 3.6.2, S. 42
$sort$	function assigning sorts to terms, Def. 2.2.1, S. 11
uv	indicates the universal variable version of a calculus, an expansion rule, etc., S. 84
val	valuation function of first-order predicate logic, Def. 2.3.2, S. 17
w^0	initial world, Def. 2.1.1, S. 9
w	weight function, S. 142
$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$	free (universal) variable, S. 12
x, y, z	object variable, S. 12

Index

Numbers of pages on which notions are defined are typeset in bold face; if a whole section is dedicated to discussing a notion or concept, the page numbers of that section are typeset in italics.