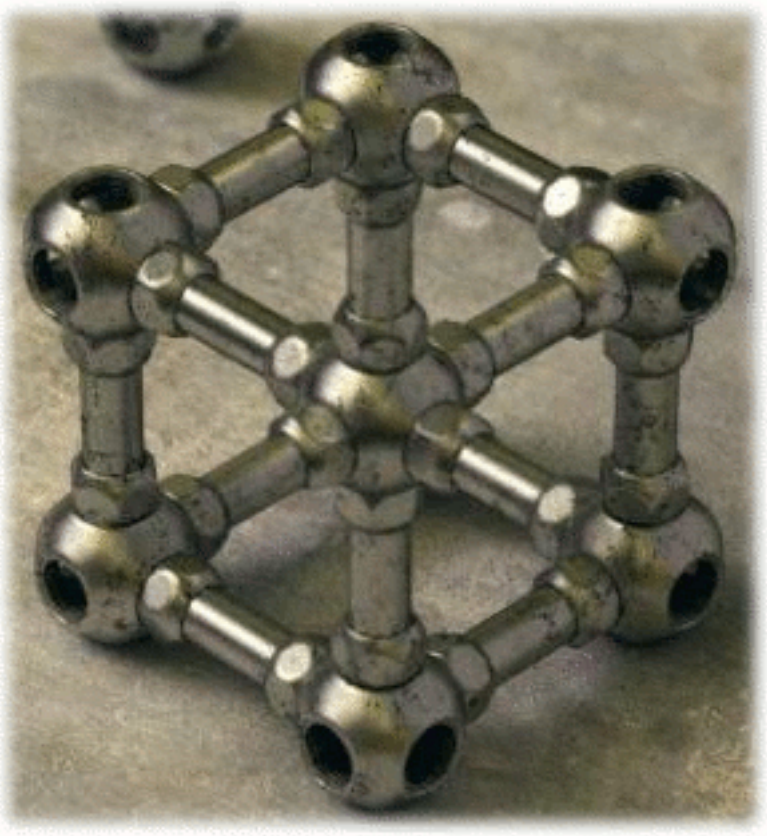


Analytic methods for the logic of proofs [1]

Author(s):
Marcelo Finger

[1]This work was supported by Fapesp Project LogProb, grant 2008/03995-5, São Paulo, Brazil.

Analytic Methods for the Logic of Proofs

Marcelo Finger

Abstract

The Logic of Proofs (LP) was proposed as Gödel's missed link between Intuitionistic and S4-proofs, but so far the tableau-based methods proposed for LP have not explored this closeness with S4 and contain rules whose analycity is not immediately evident.

We study possible formulations of analytic tableau proof methods for LP that preserve the subformula property. Two sound and complete tableau decision methods of increasing degree of analycity are proposed, KELP and preKELP. The latter is particularly inspired on S4-proofs.

The crucial role of proof constants in the structure of LP-proofs methods is analysed. In particular, a method for the abduction of proof constant specifications in strongly analytic preKELP proofs is presented; abduction heuristics and the complexity of the method are discussed.

1 Introduction

The Logic of Proofs (LP) was proposed by Artemov [Art95, Art01] as the "missed link" connecting Intuitionistic Logic to classical proofs, thus filling a gap left by Gödel's original translation from Intuitionistic Logic into S4 modal logic. LP is a propositional multi-modal language, in which modalities are *proof terms*, also called *proof polynomials*, such that if t is a proof term and A is a formula, $t : A$ is an LP-formula with the intended meaning "t is a proof of A".

In [Fit05], Melvin Fitting proposed a semantics for Artemov's LP, and presented a sound and complete tableau inference system based on that semantics. Fitting's tableau presentation has a non-analytic rule; however, its correctness was shown with respect to the analytic, cut-free sequent calculus LPG^- for LP [Art01].

As LP is capable of providing insight into proofs due to its explicit handling of proof terms, it is highly desirable that there exist automated, analytic tableau procedures for it.

This paper investigates alternative analytic tableau presentations for LP. This investigation highlights the crucial role played by proof constants in analytic proof construction. *Proof constants* are a form of atomic proof term, intended to represent evidence for formulas which are valid; they are dealt with in the semantics by a *constant specification* $\mathcal{C}$. In the semantics, one talks about $\mathcal{C}$-validity, so that constant specifications behave as a semantical parameter.

As a counterpart in terms of LP-proof theory, we introduce the concept of $\mathcal{C}$-analycity.

We investigate two possible ways to provide an analytic tableau proof system for LP. Traditional analytic tableaux [Smu68b], as well as Fitting's method for LP, are based on cut-free sequent calculus. Our approach, however, is based on a calculus using analytic cuts, which leads to tableau methods in the style of KE-tableaux [D'A99]. An advantage of using a KE-based calculus is that it is known to provide an exponential speedup in proofs for some classical formulas [D'A92].

The first KE-based method, KELP, is a $\mathcal{C}$-analytic tableau method for KE. In fact, it is a sound and complete decision procedure for $\mathcal{C}$-valid formulas, whose termination is guaranteed when $\mathcal{C}$ is finite. Admittedly, KELP-tableaux turn out to display very similar behaviour to Fittings LP-tableaux in respect to analycity. However, it has the quality of bringing to the light that the problem of computing analytic proofs is not the existence of non-analytic rules, but the way one deals with proof constants.

We then propose a second KE-based method that employs techniques of S4-theorem proving, such as prefixed modal tableaux [Fit83] and Single Step Tableaux for modal logics [Mas00], obtaining the strongly analytic preKELP tableau method for LP.

Proofs using preKELP tableaux perform a new inference step called *constant specification abduction*, which can either construct a constant specification in which the given theorem is valid, or it can verify the $\mathcal{C}$-validity of a formula according to a given constant specification $\mathcal{C}$. The abduction process uses techniques of Type Theory [BDS06] to infer a formula corresponding to a proof constant. It is shown that preKELP is sound, complete, strongly analytic and always terminates.

We study preKELP-proofs that mirror step-by-step analytic S4-tableau proofs, and show that there are cases in which this may lead to an exponential explosion on the size of terms in LP-formulas in the proof. This is considerably more complex than S4-validity, which is PSPACE-complete [Lad77]. We then provide an Abduction Heuristics, which avoids this exponential complexity, but no longer mirrors step-by-step S4-proofs. We then compare these results with respect to the related works on the literature on the complexity of the LP-validity decision and related problems.

This paper is organised as follows. The language and semantics of LP are presented in Section 2. Fitting's tableau for LP with its non-analytic set of rules is discussed in Section 3. The two KE-based tableaux for LP and their properties are presented in Section 4. Related work is discussed in Section 5. Conclusions and further work are presented in Section 6.

2 The Logic of Proofs LP

2.1 LP Syntax

The language $\mathcal{L}_{LP}$ of the logic of LP is defined over a propositional multimodal language that contains a countable set of propositional symbols $\mathcal{P} = \{p, q, r, \ldots\}$, the logic constant $\bot$ and the connective $\supset$. Besides that, there is an infinite vocabulary of modalities t, called *proof polynomials* or simply *proof terms*, such that if A is a formula so is $t : A$. The set $\mathcal{Q}$ of proof polynomials is recursively defined over a countable set of *proof constants* $\{a, b, c, \ldots\}$ and a countable set of *proof variables* $\{x, y, z, \ldots\}$ such that

- proof constants and proof variables are proof polynomials.
- if t and s are proof polynomials, then $t \cdot s$ and $t + s$ are proof polynomials;
- if t is a proof polynomial, so is $!t$.

It is assumed that the operators $\cdot$ and $+$ associate to the left so, for instance, $a \cdot x \cdot y = ((a \cdot x) \cdot y)$. The operators $\cdot$ and $+$ are neither assumed to be associative nor commutative. The semantics below clarifies these points.

An atomic LP-formula is either a propositional symbol or $\bot$. If A, B are LP-formulas, so are $A \supset B$ and $t : A$, where t is a proof polynomial. As usual, the other Boolean connectives, $\wedge$, $\vee$, $\neg$, may be defined in terms of $\supset$ and $\bot$, in a standard way.

The size of a proof term t, $|t|$, is recursively defined as follows. If t is a proof constant or variable, $|t| = 1$; $|t \cdot s| = |t + s| = |t| + |s| + 1$; $|!t| = |t| + 1$. The size of a formula A, $|A|$, is recursively defined as follows. If $A \in \mathcal{P} \cup \{\bot\}$, then $|A| = 1$; $|A \supset B| = |A| + |B| + 1$; $|t : A| = |t| + |A|$. The notion of a *subterm* is the usual.

For each LP-formula there corresponds an **S4**-modal logic formula obtained by replacing all proof polynomials modalities $t : A$ with a unary modality $\Box A$. This is called by Artemov [Art95] the *forgetful* transformation.

2.2 LP Semantics

Fitting proposed an S4-like semantics for LP formulas. An LP-model structure is a 4-tuple $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{E}, \mathcal{V} \rangle$, where $\langle \mathcal{W}, \mathcal{R} \rangle$ is an S4-frame, that is, $\mathcal{W}$ is a set of *states* or *possible worlds*, and $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{W}$ is a transitive reflexive binary accessibility relation. The *evidence function* $\mathcal{E} : \mathcal{W} \times \mathcal{Q} \to 2^{\mathcal{L}_{PL}}$ is a map that takes states and proof polynomials to sets of formulas subjected to the restrictions below; the intended meaning of $A \in \mathcal{E}(\omega, t)$ is that t is a 'possible evidence' for A in the state ω[1]. Finally, the valuation $\mathcal{V} : \mathcal{P} \to 2^{\mathcal{W}}$ takes propositional symbols to sets of states, namely the states in which that propositional symbol is true. The evidence function $\mathcal{E}$ must respect the following restrictions:

[1] It is interesting to notice that Artemov's view of proof polynomials as representing *proofs* has changed into Fitting's view as *possible evidence*

1. **Application:** if $A \supset B \in \mathcal{E}(\omega, t)$ and $A \in \mathcal{E}(\omega, s)$ then $B \in \mathcal{E}(\omega, t \cdot s)$.
2. **Monotonicity:** if $\omega_1 \mathcal{R} \omega_2$ then $\mathcal{E}(\omega_1, t) \subseteq \mathcal{E}(\omega_2, t)$.
3. **Proof Checker:** if $A \in \mathcal{E}(\omega, t)$ then $t : A \in \mathcal{E}(\omega, !t)$.
4. **Sum:** $\mathcal{E}(\omega, t) \cup \mathcal{E}(\omega, s) \subseteq \mathcal{E}(\omega, t + s)$.

According to the **Application** restriction, if its conditions are met, then the term $t \cdot s$ is a possible evidence for B, but nothing is said about $s \cdot t$ being a possible evidence for B or any other formula; that is, $\cdot$ does not commute; similarly, $+$ does not commute. Associativity is not forced either.

Given a model structure $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{E}, \mathcal{V} \rangle$, the forcing relation is defined for every $\omega \in \mathcal{W}$ and for every formula in $\mathcal{L}_{LP}$:

1. $\mathcal{M}, \omega \Vdash p$, for $p \in \mathcal{P}$, iff $\omega \in \mathcal{V}(p)$.
2. $\mathcal{M}, \omega \nVdash \bot$.
3. $\mathcal{M}, \omega \Vdash A \supset B$ iff $\mathcal{M}, \omega \nVdash A$ or $\mathcal{M}, \omega \Vdash B$.
4. $\mathcal{M}, \omega \Vdash t : A$ iff $A \in \mathcal{E}(\omega, t)$ and, for every $\omega' \in \mathcal{W}$ with $\omega \mathcal{R} \omega'$, $\mathcal{M}, \omega' \Vdash A$.

Finally, the semantics deals with proof constants through *constant specifications*. The idea of a proof constant is to represent an evidence for elementary truth, which requires no further analysis. A proof constant can serve as evidence for a set (possibly empty) of instances of LP-valid formulas. As a result, a constant specification is a mapping $\mathcal{C}$ from proof constants to sets of formulas. A proof constant c is *for* a formula A iff $A \in \mathcal{C}(c)$. A proof constant can be only for formulas which are true at every possible world in any model; as is discussed in detail in Section 4.1, this fact brings an extra burden to LP-theorem proving for, unlike S4-theorem proving, the validity of an $A \in \mathcal{C}(c)$ has to be "guessed" in some LP-proof methods. The proof method proposed in Section 4.2 aims at avoiding such problem.

The elementary truths contained in constant specifications are supposed to be the instances of the following axioms:

A0 All classical propositional tautologies
A1 $t : (A \supset B) \supset (s : A \supset (t \cdot s) : B)$
A2 $t : A \supset A$
A3 $t : A \supset !t : t : A$
A4 $t : A \supset (t + s) : A$
A5 $s : A \supset (t + s) : A$

A model $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{E}, \mathcal{V} \rangle$ *meets constant specification* $\mathcal{C}$ if $\mathcal{C}(c) \subseteq \mathcal{E}(\omega, c)$ for every $\omega \in \mathcal{W}$.

A formula is $\mathcal{C}$-LP *valid* iff it is true at all possible worlds in any model that meets constant specification $\mathcal{C}$.

In Fitting's presentation of LP semantics, it is mentioned that the role of proof constants is "surprisingly central". We will see that the same central role of proof constants is found in the presentation of a tableau system for LP with the subformula property.

3 Fitting's Tableau for LP

Fitting's tableaux for LP (FTLP) manipulate *signed formulas* of the form $T\ A$ and $F\ A$, where A is an LP-formula. A tableau proof of a formula A starts with a signed formula $F\ A$, which is expanded into a tree of signed formulas through a set of *expansion rules*. Rules are either linear or branching, and a branch in the tableau is *closed* if it contains either $T\ B$ and $F\ B$ for some B, or if it contains $T\ \bot$. A tableau is closed if all its branches are closed. A formula A is provable in FTLP if there is a closed tableau for $F\ A$, represented by $\vdash_{\mathrm{FTLP}} A$.

Fitting's linear expansion rules are presented in Figure 1. The branching rules are presented in Figure 2.

$$\frac{F\ A \supset B}{\begin{array}{l} T\ A \\ F\ B \end{array}}\ (F\ \supset) \qquad \frac{T\ t : A}{T\ A}\ (T\ :) \qquad \frac{F\ !t : (t : A)}{F\ t : A}\ (F\ !)$$

$$\frac{F\ (t+s) : A}{F\ t : A}\ (F\ +_1) \qquad \frac{F\ (t+s) : A}{F\ s : A}\ (F\ +_2)$$

Figure 1: Linear Expansion Rules

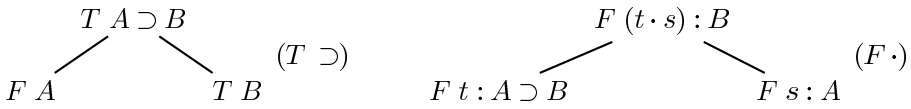


Figure 2: Branching Expansion Rules

Finally, an FTLP *using constant specification* $\mathcal{C}$ has an extra closing rule, whereby a branch closes if it contains $Fc : A$ where $A \in \mathcal{C}(c)$.

Proposition 3.1 (FTLP Soundness and Completeness [Fit05]) *A formula A has a tableau proof using constant specification $\mathcal{C}$ iff A is $\mathcal{C}$-*LP *valid.*

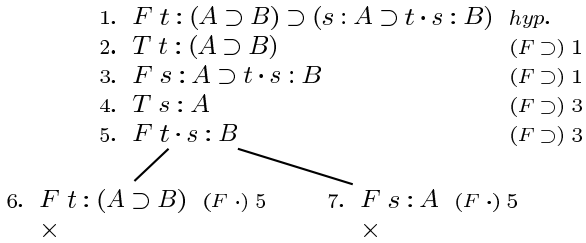
We say that a tableau method has the *subformula property* if all expanded formulas are subformulas of some previously occurring formula on the same branch; proof methods that have the subformula property, in general, allow the proof to be constructed by analysing only the internal structure of formulas,

and are called *analytic*. In case of LP, we also consider constituents of proof polynomials as forming subformulas. In this sense, the inference of $Ft : A$ from $F(t + s) : A$ respects the subformula property.

The problem with regard to the subformula property lies in the branching rule $(F\ \cdot)$. In all linear rules, as well as in rule $(F\ \supset)$, the formulas in the conclusion occur as a subformula of the premiss. Not so in rule $(F\ \cdot)$, whose premiss is $F\ (t \cdot s) : B$ but whose conclusions is either $F\ t : A \supset B$ or $F\ s : A$. Clearly, A occurs in both branched conclusions but not in the premiss.

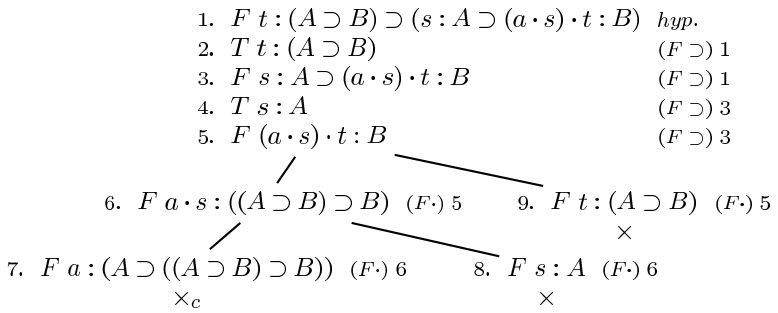
One should mention at this point that another tableau system for LP was proposed by Renne [Ren04], based not on Fitting's semantics but on an earlier semantics for LP by Mkrtychev [Mkr97]. That proposal had a branching rule β^X basically identical to rule $(F\ \cdot)$, and thus it presents the same problem of lack of analycity.

Example 3.2 The reason rule $(F\ \cdot)$ is present in the system is the proof of the LP version of the modal axiom K, namely $t : (A \supset B) \supset (s : A \supset t \cdot s : B)$, as shown in the FTLP tableau below:



We note, however, that even if rule $(F\ \cdot)$ has the "potential" for the generation of proofs without the subformula property, this is *not* the case in the proof above, for the formulas generated by the application of rule $(F\ \cdot)$ in fact generated two branches whose formulas were in fact subformulas of the initial formula. However, the potential for the generation of proofs in which the subformula property fails remains. The example below shows a real failure of the subformula property.

Example 3.3 Let $A \supset ((A \supset B) \supset B) \in \mathcal{C}(a)$. We present a tableau for a small variation of the previous example, $\vdash_{\text{FTLP}} t : (A \supset B) \supset (s : A \supset (a \cdot s) \cdot t : B)$



The leftmost branch closes due to the constant specification $A \supset ((A \supset B) \supset B) \in \mathcal{C}(a)$, which is represented by the closed branch symbol $\times_c$, as opposed to the usual branch closure, represented by $\times$. Formulas in nodes 6 and 7 are clear violations of the subformula property, for formulas $((A \supset B) \supset B)$ and $A \supset ((A \supset B) \supset B)$ are not subformulas of the original formula. We see that the presence of proof constants is central, for without them the leftmost branch would not close.

The problem with non-analytic rule $(F \cdot)$ is that there is, in principle, an infinite number of ways in which it is expandable. In fact, from $Ft \cdot s : B$ there are infinitely many formulas A such that the tableau can branch into $Ft : A \supset B$ and $Fs : A$.[2]

In the following we propose two tableau systems that try to avoid such problematic behaviour.

4 KE Tableaux for LP

KE tableaux allow for the construction of non-analytic proofs,in which the subformula property fails. This is done by having a single branching rule, namely the Principle of Bivalence, which corresponds to the Cut Rule in sequent calculus. PB branches over a formula A, creating two branches, one in which $T\ A$ holds, the other in which $F\ A$ holds. However, if the use of PB is *analytic* — that is, restricted to formulas A that are subformulas of some previously occurring formula — it is guaranteed that the tableau has the subformula property, for all tableau expansion rules are analytic and preserve this property. Furthermore, this restriction to analytic PB can be done without loosing completeness.

KE tableaux were proposed by D'Agostino and Mondadori [DM94, D'A92] as a way of incorporating the cut rule, or possibly some restricted form of it, back into a tableau calculus. As noted by Smullyan [Smu68a], all the main results one can prove using cut elimination can be obtained simply by restricting the use of the cut rule to *analytic cuts*, namely a cut over a subformula of formulas of some previously occurring formula. This restriction is usually incorporated into KE proofs; as all other tableau expansion rules are analytic — that is, the conclusion of the rule is a subformula of the premisses of the rule — KE proofs thus restricted are analytic and have the subformula property. Furthermore, it is known that analytic KE-tableaux allow for small, polynomial proofs of some propositional formulas for which Smullyan's analytic tableaux only have exponentially large proofs [D'A92].

When one is dealing with LP-formulas, the notion of subformula property has to be clarified. For instance, we want a definition of LP-subformulas in which rules $(F +_1)$ and $(F +_2)$ are subformula-preserving. Furthermore, we

[2]Note that the complexity of the proof polynomial decreases, so if one can limit the search of A to a finite set, the termination of the FTLP method can be established.

parameterise the notion of LP-subformula to include constant specifications $\mathcal{C}$. In this sense, we define A to be an LP-*subformula of* B *under* $\mathcal{C}$ if:

(a) $A = B$;

(b) B is a boolean combination of $B_1, \ldots, B_m$, and A is an LP-subformula of B_j under $\mathcal{C}$, for some B_j, $1 \leq j \leq m$;

(c) $A = t' : A'$, where t' is a subterm of t in B and A' is an LP-subformula of B under $\mathcal{C}$;

(d) A is a subformula of $A' \in \mathcal{C}(c)$, for some proof constant c occurring in B.

Items (a) and (b) are just the usual definition of subformula. Item (c) is quite liberal and allows for $t : A$ to be a subformula of $s : B$ when t is a subterm of s and A is a subformula of B, which makes rules $(F +_1)$ and $(F +_2)$ subformula-preserving; however, it also makes $t : A$ a subformula of $(s : (A \wedge B)) \vee t : C$, which is perhaps not immediately intuitive, but if one looks at its modal version, this would simply mean that $\Box A$ is a subformula of $\Box(A \wedge B) \vee \Box C$, which is quite acceptable[3]. Item (d) makes any subformula occurring in $\mathcal{C}(c)$ to be a subformula of any formula B under $\mathcal{C}$ if c occurs in B. In particular, if all constant specifications have at most a finite number of formulas, the number of subformulas under $\mathcal{C}$ is also finite. In this case, we say that $\mathcal{C}$ is finite.

A formula A is a *simple* LP-*subformula of* B if it is an LP-subformula of B under every $\mathcal{C}$. A proof method is $\mathcal{C}$-*analytic* if every provable formula A has a proof containing only LP-subformulas of A under $\mathcal{C}$.

We develop next a $\mathcal{C}$-analytic KE tableau calculus for LP, KELP.

4.1 KELP Tableaux

KE tableaux always deal with signed formulas. KELP is an extension of propositional KE tableau that has in common with FTLP the closing conditions and the linear expansion rules in Figure 1. As usual, the proof of A starts with FA as its root node and proceeds by expanding the tableau according to the expansion rules.

The branching rules are substituted by two sets of rules. First, there are the two-premissed rules of Figure 3, one for each of the branching rules in FTLP (Figure 2). Note that the rules in Figure 3 display a symmetry not found in Figure 2. The rule $(T \supset_{KE})$ is the classical two-premissed rule for dealing the $\supset$ connective, namely the Modus Ponens rule. Rule $(T : \supset)$ is basically the same Modus Ponens rule that incorporates the labels, and its soundness is derived immediately from the version of the K axiom for LP. In these two-premissed rules, the top formula is called the *main* formula, and the second premiss is called the *auxiliary* formula in the rule.

Note that both two-premissed rules are analytic. Rule $(T \supset_{KE})$ is classically analytic. Figure 5 shows the restrictions on applications of KELPrules. The

[3]Item (c) says that, if t is a term occurring in B and A is any subformula of B or of some constant specification in the usual sense, then $t : A$ is accepted as an LP-subformula of B.

$$\frac{T\ A \supset B \quad T\ A}{T\ B}(T\ \supset_{KE}) \qquad \frac{T\ t : A \supset B \quad T\ s : A}{T\ (t \cdot s) : B}\ (T\ :\ \supset)$$

Figure 3: Two-premissed Rules for KELP

proviso of rule $(T\ :\ \supset)$ guarantees analycity. This proviso also avoids infinite derivations, as that shown by the following sequence of $(T\ :\ \supset)$-applications.

$$T\ t : A \supset A,\ \ T\ s : A,\ \ T\ t \cdot s : A,\ \ T\ t \cdot (\ t \cdot s) : A,\ \ T\ t \cdot (t \cdot (\ t \cdot s)) : A,\ \ \ldots$$

Eventually, the application of rule $(T\ :\ \supset)$ is barred, when the derived formula $T\ t' : A$ is such that t' does not occur in the formula at the root of the tableau.

Besides the linear one- and two-premissed rules, KELP contains a single branching rule, called Principle of Bivalence (PB), which is the translation of the cut rule of the sequent calculus into a tableau setting. This rule is illustrated in Figure 4.

$$\begin{array}{ccc} F\ A & & T\ A \end{array} \qquad (PB)$$

Figure 4: The Principle of Bivalence

When the proviso in Figure 5 is observed, rule PB becomes $\mathcal{C}$-analytic. A KELP proof is $\mathcal{C}$-analytic if every use of PB is $\mathcal{C}$-analytic. We assume that KELP only allows $\mathcal{C}$-analytic PB.

The tableaux system KELP is thus composed of the unary linear rules in Figure 1, the binary linear rules in Figure 3, the $\mathcal{C}$-analytic PB branching rule in Figure 4; the two closing rules of FTLP, contradiction closing $(\times)$ and proof constant closing $(\times_c)$; and the restrictions of Figure 5.

Rule	*Proviso*
$(T\ :\ \supset)$	$t \cdot s$ occurs previously on the branch
(PB)	A is an LP-subformula of the root of the tableau under $\mathcal{C}$

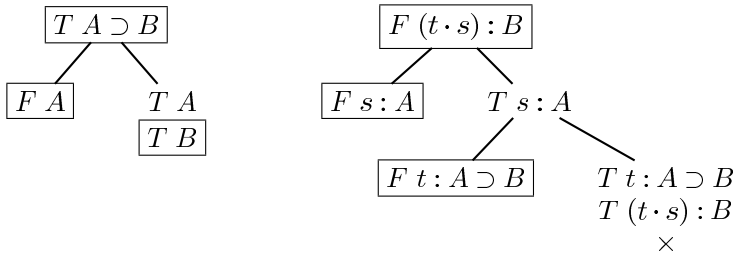
Figure 5: KELP-rule restrictions

We say that formula A_0 is provable in KELP, $\vdash_{\text{KELP}} A_0$, if there is a closed tableau whose root node is $F\ A_0$, in which every internal node is obtained by applying a KELP expansion rule, and such that every branch is closed.

Theorem 4.1 KELP *is sound and complete.*

Proof The unary linear rules inherit soundness from FTLP. Rule $(T\ \supset_{KE})$ is Modus Ponens, thus sound; rule $(T:\supset)$ is sound due to LP's version of modal axiom K. And PB is sound in any extension of propositional classical logic, such as LP. So the system is sound.

To see that this system is complete, we show that every FTLP proof can be simulated by KELP. As all FTLP linear rules are also KELP rules, it is enough to show that we can derive the branching rules of the FTLP tableau system:



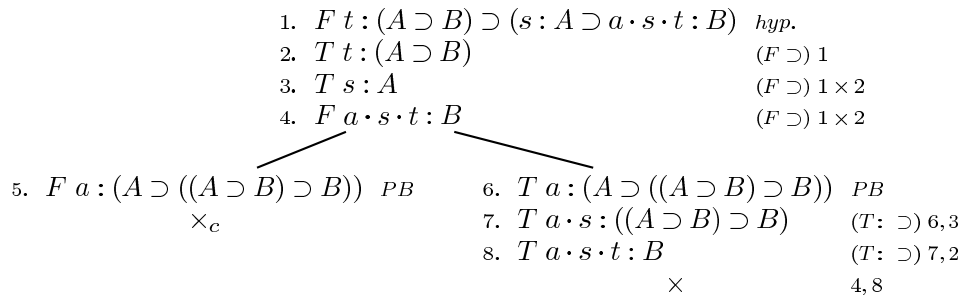
Note that the two-premissed KELP rules were used in this derivation. As FTLP proves all $\mathcal{C}$-LP valid formulas, and KELP can simulate those proofs, KELP can prove all $\mathcal{C}$-LP valid formulas.

Finally, we have to show that the use of PB is $\mathcal{C}$-analytic. In fact, the derivation of $(T\ \supset_{KE})$ is always analytic. The KELP derivation of $(T:\supset)$ is $\mathcal{C}$-analytic if the original FTLP is. As FTLP is based on analytic, cut-free sequent calculus by Artemov [Art01], for a provable LP-formula A_0 there is always a $\mathcal{C}$-analytic FTLP proof, which by the method above is transformed into a $\mathcal{C}$-analytic KELP proof, which finishes the proof. □

Rule $(T\ \supset_{KE})$ can be classically simulated by rule rule $(T\ \supset)$ in analytic tableaux, but not rule PB, since analytic tableaux are based on a cut-free calculus and PB is the tableau version of the cut rule. Rule $(T:\supset)$ *cannot* be simulated by rule $(F\cdot)$, because the two premisses of rule $(T:\supset)$ do not allow for the application of rule $(F\cdot)$.

The PB rule provides a clear way to introduce constant specifications in proofs. This is clear in the following example, that presents a KELP proof for the formula in Example 3.3.

Example 4.2 Consider a KELP proof of $t:(A\supset B)\supset(s:A\supset a\cdot s\cdot t:B)$.



The left branch closes at node 5 because $A \supset ((A \supset B) \supset B) \in \mathcal{C}(a)$. The application of PB makes it explicit that the elementary truth carried by proof constant a is brought into the proof by means of a cut-like introduction. This introduction violates the subformula property, be it an FTLP or a KELP tableau.

The branching in the example above was done over a formula in the constant specification $\mathcal{C}(a)$, so the branching was $\mathcal{C}$-analytic, and the proof is $\mathcal{C}$-analytic. This process can be repeated for any finite number of constant specifications. The F-side of the branch will immediately close.

It is clear that any $\mathcal{C}$-valid formula is also $\mathcal{C}'$ valid for some finite $\mathcal{C}'$ contained in $\mathcal{C}$. In fact, if A_0 is $\mathcal{C}$-valid, there is a KELP proof of it in which only a finite number of branches closes due to constant specifications. Let $\mathcal{C}'$ be the union of such constant specifications. Clearly, A_0 is also $\mathcal{C}'$-valid.

We consider now finite constant specifications $\mathcal{C}$.

Theorem 4.3 KELP *tableaux always terminate over finite* $\mathcal{C}$.

Proof We first note that, for a fixed size n, and a fixed number of variables m, there are only finitely many LP-formulas of size up to n.

All non-branching KELP-rules, except $(T : \supset)$, decrease the complexity of the formulas, avoiding infinite branches. Two-premissed rule $(T : \supset)$ cannot lead to an infinite loop either, for its proviso puts a bound on the complexity of formulas it can generate, so it can only be used a finite number of times on a branch. Finally, the use of $\mathcal{C}$-analytic PB has at most finitely many candidates for branching. Therefore a $\mathcal{C}$-analytic KELP proof tree has at most finitely many branches which are all finite. So the tree expansion always terminate. □

By putting together Theorems 4.1 and 4.3, we obtain the main result of this section.

Corollary 4.4 KELP *tableaux are a decision procedure for* LP *over finite* $\mathcal{C}$.

The fact that we deal with $\mathcal{C}$-analytic proofs, instead of simply analytic proofs for LP, shows the central position played by constant specifications in the proof theory of LP.

Both KELP and FTLP methods deal with proof constants in such a way that one has to "guess" all the manipulations hidden by a proof constant before they are used in a proof. Furthermore, this constant specification must be finite and given a priori for the proof procedure to terminate.

In a way, the main insight brought by KELP-tableau is that the problem of analycity does not derive from the presence of non-analytic $(F\ \cdot)$, but from the fact that the closing of a branch with a formula $Fa : A$ for $a \in \mathcal{C}(a)$ makes available for consideration any formula in a constant specification, which has the potential to include any propositional classical theorem.

In the following we aim to create a tableau systems for LP that avoids that kind of behaviour. For that, we propose to do the following.

1. We want to avoid guessing the hidden associations of proof constants, letting the proof reveal the elementary validities used in it. For that, the proof construction process will compute (valid) formulas and associate them to a constant specification.

   This procedure is called the *abduction of constant specifications*.

   If a constant specification $\mathcal{C}$ is given as input, and the proof construction associates a formula A to c such that $A \notin \mathcal{C}(c)$, the proof is rejected.

2. We want to explore the proximity of LP and S4, and adapt S4-proof methods to LP. Neither FTLP nor KELP explores this path.

   The literature contains several proposals of tableaux for modal logics which employ the semantic idea of possible worlds. In particular, we base our approach on the ideas of Fitting's prefixed tableaux [Fit83] and Massacci's single step tableaux (SST) [Mas00].

4.2 Prefixed KE Tableaux for LP (preKELP)

Based on prefixed tableau for S4 modal logics, we propose here a tableau method for LP that employs prefixed formulas. It is also based on the KE method, so it is called preKELP-tableaux.

The novelty of this method is the procedure for the *abduction of constant specifications*, which makes preKELP-tableau *strongly analytic*, in the sense that if ρ is a prefix and if $F\langle\rho\rangle A$ or $T\langle\rho\rangle A$ are internal nodes in a tableau, then A is a subformula of the initial formula being proved or it is a formula computed by the abduction procedure. This abduction procedure is based on the *type inference* in Type Theory [Hin97], and explores the relationship between LP and Intuitionistic Logic, which is related to Type Theory.

In preKELP, a *prefix* ρ may be either undecorated, $\rho = \omega$, or may be decorated with a term, $\rho = \omega|t$; both forms are said to be based at ω. In any prefix, ω is a non-empty sequence $n_1.n_2 \ldots .n_m$ of non-zero natural numbers representing a possible world, with the intended meaning that $\omega.n$ is accessible from ω; it is assumed that $\omega.0 = \omega$, and $\omega.0$ is used only in connection with a reflexive element of the accessibility relation.

The nodes of a preKELP-tableau are signed prefixed formulas of the form $T\langle\rho\rangle A$ and $F\langle\rho\rangle A$. The intended meaning of $T\langle\omega.n|t\rangle A$ is that A is true at world $\omega.n$ and t is evidence for A at ω; $T\langle\omega\rangle A$ means simply that A is true at ω; similarly, $F\langle\omega.n|t\rangle A$ means that t is *not* an evidence for A at ω *or* A is false at $\omega.n$, and $F\langle\omega\rangle A$ means simply that A is false at ω.

The prefix 1 is the special designation of the initial prefix attributed to the formula we are trying to falsify. To prove A_0, the tableau starts with prefixed signed formula $F\langle 1\rangle A_0$.

The presentation of preKELP expansion rules employs the discipline of Massacci's Single Step Tableaux (SST), and can be found in Figure 6.

$$\frac{F\langle\omega\rangle A\supset B}{\begin{array}{l}T\langle\omega\rangle A\\ F\langle\omega\rangle B\end{array}}\,(F\supset)\qquad \frac{\begin{array}{l}T\langle\omega\rangle A\supset B\\ T\langle\omega\rangle A\end{array}}{T\langle\omega\rangle B}\,(T\supset)\qquad \begin{array}{ccc} & \diagup\ \diagdown & \\ F\langle\rho\rangle A & & T\langle\rho\rangle A\end{array}\,(APB)$$

$$\frac{T\langle\omega\rangle A\supset\bot}{F\langle\omega\rangle A}\,(T\neg)\qquad \frac{F\langle\omega\rangle A\supset\bot}{T\langle\omega\rangle A}\,(F\neg)$$

$$\frac{\begin{array}{l}T\langle\omega.n|t\rangle A\supset B\\ T\langle\omega.n|s\rangle A\end{array}}{T\langle\omega.n|t\cdot s\rangle B}\,(T\supset_\omega)\qquad \frac{T\langle\omega|t\rangle A}{T\langle\omega\rangle A}\,(T\downarrow)$$

$$\frac{F\langle\omega\rangle t:A}{F\langle\omega.n|t\rangle A}\,(\pi)\qquad \frac{F\langle\omega.n|t+s\rangle A}{\begin{array}{l}F\langle\omega.n|t\rangle A\\ F\langle\omega.n|s\rangle A\end{array}}\,(\pi+)$$

$$\frac{T\langle\omega\rangle t:A}{T\langle\omega.n|t\rangle A}\,(\nu K)\qquad \frac{T\langle\omega\rangle t:A}{T\langle\omega.0|t\rangle A}\,(\nu T)\qquad \frac{T\langle\omega\rangle t:A}{T\langle\omega.n|!t\rangle t:A}\,(\nu 4)$$

Figure 6: preKELP-expansion Rules

The two initial lines of Figure 6 present the preKELP version of linear and branching rules of classical KE. According to rule application restrictions in Figure 7, the branching rule PB is restricted to its analytic form, APB, that is, APB can be applied to prefixed formulas $\langle\omega|t\rangle A$ or $\langle\omega\rangle A$, provided that ω has already occurred in the tableau for some formulas, and A is a subformula of one of them. Rules $(F\supset)$ and $(T\supset)$ can only be applied to undecorated nodes. Rules $(T\neg)$ and $(F\neg)$ are derived, but are presented for convenience.

The third line of Figure 6 presents the evidence manipulation rules. The connective rule $(T\supset_\omega)$ provides evidence propagation via Modus Ponens; rules $(T\supset_\omega)$ and $(T\supset)$ would be treated by S4 as the same. Rule $(T\downarrow)$ describes that evidence decorations can always be dropped from T-marked formulas.

To deal with modalities, SST employs two types of expansion rules, π-rules and ν-rules, whose preKELP version are dealt with in the last two lines of Figure 6. The (π)-expansion rule creates a new world $\omega.n$, so that the proviso

in Figure 7 states that it must be new to the branch. On the other hand, $(\pi+)$ is simply an evidence propagation rule for terms of the form $t + s$.

The ν-expansion rules have to cope with the three basic components of an S4 system, namely, the K, T and 4-conditions. According to the restrictions in Figure 7, rules (νK) and $(\nu 4)$ can only be expanded if $\omega.n$ has been created earlier on the branch, as a result of some (π)-expansion. We say that $\omega.n$ is *active* on a branch if it occurs in the consequence of a ν-rule on that branch. The last restriction on $(\nu 4)$ in Figure 7 avoids the infinite application of $(\nu 4)$ and guarantees analycity.

Rule (νT) makes use of a world of the form $\omega.0$ instead of ω so as to state that t is an evidence for A at ω.

Rule	*Proviso*
(APB)	ω and A occur at some formula above, where $\rho \in \{\omega, \omega\|t\}$
(π)	$\omega.n$ is new
$(\nu K), (\nu 4)$	$\omega.n$ is not new
$(\nu 4)$	$!t$ occurs above

Figure 7: preKELP-rule restrictions

As in KELP, there are three branch closing conditions in preKELP-tableaux: the presence of opposite formulas $F\langle\rho\rangle A$ and $T\langle\rho\rangle A$; the presence of $T\langle\rho\rangle \bot$; and $F\langle\omega\rangle c : A$ where $A \in \mathcal{C}(c)$. If all branches of a tableau are closed, so is the tableau.

A formula A_0 is preKELP-provable, $\vdash_{\text{preKELP}} A_0$, if a tableau starting with $F\langle 1\rangle A_0$ and expanded according to the rules of Figure 6 and restrictions of Figure 7, has all branches closed.

For the time being, we relax the restriction of the branching rule (APB) and allow it to be $\mathcal{C}$-analytic, that is, one allows the branching over formulas $a : A$ such that $A \in \mathcal{C}(a)$. The correctness and completeness of this version will be shown. After that, a new closing condition will be added, and the stricter restriction on (APB) will be reinstated.

4.3 Soundness, Completeness and Termination of preKELP

First, preKELP soundness is considered. Let $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{E}, \mathcal{V}\rangle$ be an LP-model such that $\mathcal{W}$ is a set of undecorated prefixes and $\mathcal{R}$ is such that $\omega \mathcal{R} \omega.n$. We say that $\mathcal{M}$ satisfies a signed formula in the following cases:

Singned Formula	*Condition*
$T\langle\omega\rangle A$	$\mathcal{M}, \omega \Vdash A$
$T\langle\omega.n\|t\rangle A$	$\mathcal{M}, \omega.n \Vdash A$ and $A \in \mathcal{E}(\omega, t)$
$F\langle\omega\rangle A$	$\mathcal{M}, \omega \nVdash A$
$F\langle\omega.n\|t\rangle A$	$\mathcal{M}, \omega.n \nVdash A$ or $A \notin \mathcal{E}(\omega, t)$

Lemma 4.5 *If the premisses of a* preKELP*-rule are satisfied by a model $\mathcal{M}$, so are its conclusions.*

Proof By inspection all rules in Figures 6. Rules $(F\supset),(T\supset),(T\neg),(F\neg)$ and (APB) are classical logic rules, whose correctness is immediate, so details are omitted.

Consider rule $(T\supset_\omega)$. By hypothesis, $\mathcal{M},\omega.n \Vdash A\supset B$, and $\mathcal{M},\omega.n \Vdash A$, so $\mathcal{M},\omega.n \Vdash B$. From the hypothesis, it is also the case that $A\supset B\in\mathcal{E}(\omega,t)$ and $A\in\mathcal{E}(\omega,s)$, so by the **Application** rule of the evidence function, $B\in\mathcal{E}(\omega,t)$. It follows that the consequence of the rule is satisfied by $\mathcal{M}$.

Consider rule $(T\downarrow)$. The result is immediate.

Consider rule (π). By hypothesis, $\mathcal{M},\omega \nVdash t:A$. So according to the semantics, $A\notin\mathcal{E}(\omega,t)$ or there exists ω' such that $\omega\mathcal{R}\omega'$ and $\mathcal{M},\omega' \nVdash A$. According to the restrictions in Figure 7, $\omega.n$ is new, and as $\omega\mathcal{R}\omega.n$, make $\omega'=\omega.n$, such that $\mathcal{M}$ satisfies $F\,\langle\omega.n|t\rangle\,A$.

Consider rule $(\pi+)$. By hypothesis, $A\notin\mathcal{E}(\omega,t+s)$ or $\mathcal{M},\omega.n \nVdash A$. From the **Sum** property of evidence functions, it follows that $A\notin\mathcal{E}(\omega,t)$ and $A\notin\mathcal{E}(\omega,s)$, so $\mathcal{M}$ satisfies both $F\,\langle\omega.n|t\rangle\,A$ and $F\,\langle\omega.n|s\rangle\,A$.

Consider rule (νK). By hypothesis, $\mathcal{M},\omega \Vdash t:A$, so $A\in\mathcal{E}(\omega,t)$ and for every ω' such that $\omega\mathcal{R}\omega'$, $\mathcal{M},\omega' \Vdash A$. From the restrictions in Figure 7, $\omega.n$ already exists and by construction $\omega\mathcal{R}\omega.n$, so $\mathcal{M}$ satisfies $T\,\langle\omega.n|t\rangle\,A$.

Consider rule (νT). By hypothesis, $\mathcal{M},\omega \Vdash t:A$, so $A\in\mathcal{E}(\omega,t)$ and for every ω' such that $\omega\mathcal{R}\omega'$, $\mathcal{M},\omega' \Vdash A$. As $\mathcal{R}$ is reflexive and $\omega=\omega.0$, it follows that $\mathcal{M},\omega.0 \Vdash A$, so $\mathcal{M}$ satisfies $T\,\langle\omega.0|t\rangle\,A$.

Consider rule $(\nu 4)$. By hypothesis, $\mathcal{M},\omega \Vdash t:A$, so $A\in\mathcal{E}(\omega,t)$ and for every ω' such that $\omega\mathcal{R}\omega'$, $\mathcal{M},\omega' \Vdash A$. From the **Proof Checker** property of evidence functions, $t:A\in\mathcal{E}(\omega,!t)$. Furthermore, from the restrictions in Figure 7, $\omega.n$ already exists, $\omega\mathcal{R}\omega.n$, and by the **Monotonicity** property, $A\in\mathcal{E}(\omega.n,t)$. Consider any ω'' such that $\omega.n\mathcal{R}\omega''$; from the transitivity of $\mathcal{R}$ one obtains $\omega\mathcal{R}\omega''$, so $\mathcal{M},\omega'' \Vdash A$. From $A\in\mathcal{E}(\omega.n,t)$, this yields $\mathcal{M},\omega.n \Vdash t:A$ and as $t:A\in\mathcal{E}(\omega,!t)$, $\mathcal{M}$ satisfies $T\,\langle\omega.n|!t\rangle\,t:A$.

This finishes the proof. □

Define a *KE-saturated set* of signed formulas Θ (an analog of Hintikka's *downward saturated sets*) as follows.

- Let $\alpha\in\Theta$, where α is a premiss of a one-premissed rule, i.e. $(F\supset)$, (π), $(\pi+)$, $(T\downarrow)$, (νK), (νT) and $(\nu 4)$. Then all the conclusions of this rule are in Θ.
- Let $\beta\in\Theta$, where β is the main formula of a two-premissed rule, $(T\supset)$ or $(T\supset_\omega)$; let β_1 be the auxiliary formula and β_2 the conclusion. Let $\bar{\beta}_1$ be the opposite of β_1. Then $\beta_1\in\Theta$ or $\bar{\beta}_1\in\Theta$; if $\beta,\beta_1\in\Theta$, then $\beta_2\in\Theta$.

A KE-saturated set is *consistent* if it does not contain a pair of opposite formulas, nor a formula of the form $T\,\langle\omega\rangle\,\bot$, nor $F\,\langle\omega\rangle\,a:A$, where $A\in\mathcal{C}(a)$. A set of formulas has a model if there is $\mathcal{M}$ that satisfies every formula in it.

Lemma 4.6 *If a branch of a* PREKELP *tableau has a model, it can be expanded into a consistent KE-saturated set.*

Proof By Lemma 4.5, every expansion remains satisfied by the model. If rule APB is applied to saturate the branch, one of the branches is satisfied by the model, so when a saturated branch is obtained, it is consistent. □

Theorem 4.7 (Soundness) *Let $\mathcal{C}$ be a constant specification. If the formula A has a closed* preKELP*-tableau validated for $\mathcal{C}$, then A is $\mathcal{C}$-valid.*

Proof Assume that A is not $\mathcal{C}$-valid. Then FA is a consistent formula. By Lemma 4.6 it will be expanded into a consistent KE saturated set, contradicting the fact that the tableau closes. □

For completeness, the following result is shown.

Lemma 4.8 *Every consistent KE-saturated set has a model.*

Proof Let Θ be a consistent KE-saturated set. We construct a model $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{E}, \mathcal{V} \rangle$ in the following way.

Let $\mathcal{W}$ be the set of all prefixes occurring in Θ. Let $\mathcal{R}_0$ be such that $\omega \mathcal{R}_0 \omega.n$ for every $\omega, \omega.n \in \mathcal{W}$. Let $\mathcal{R}$ be the reflexive-transitive closure of $\mathcal{R}_0$.

Let $\mathcal{E}$ be such that $A \in \mathcal{E}(\omega, t)$ iff adding $F \langle \omega \rangle t : A$ to Θ closes it.[4] We show that $\mathcal{E}$ respects the restrictions of evidence functions; due to $\mathcal{C}$-analycity, only terms occurring in Θ have to be considered.

- **Application:** Assume $B \supset A \in \mathcal{E}(\omega, t)$ and $B \in \mathcal{E}(\omega, s)$. Then Θ has as elements (1) $T \langle \omega \rangle t : B \supset A$ and (2) $T \langle \omega \rangle s : B$. Now add (3) $F \langle \omega \rangle t \cdot s : A$ to Θ so by (π) on (3), (4) $F \langle \omega.n | t \cdot s \rangle A$; by (νK) on (1) and (2), derive (5) $T \langle \omega.n | t \rangle B \supset A$ and (6) $T \langle \omega.n | s \rangle B$. From $(T \supset_\omega)$ on (5) and (6) obtain $T \langle \omega.n | t \cdot s \rangle A$, which closes with (4). So $A \in \mathcal{E}(\omega, t \cdot s)$.

- **Monotonicity:** Assume $A \in \mathcal{E}(\omega, t)$, so (1) $T \langle \omega \rangle t : A \in \Theta$. Add to Θ (2) $F \langle \omega.n \rangle t : A$ so by (π) on (2), (3) $F \langle \omega.n.m | t \rangle A$. By $(\nu 4)$ on (1), (4) $T \langle \omega.n | !t \rangle t : A$ and by $(T \downarrow)$, (5) $T \langle \omega.n \rangle t : A$, which yields (6) $T \langle \omega.n.m | t \rangle A$, closing Θ. So $A \in \mathcal{E}(\omega.n, t)$.

- **Type Checking:** Assume $A \in \mathcal{E}(\omega, t)$, so (1) $T \langle \omega \rangle t : A \in \Theta$. Add to Θ (2) $F \langle \omega \rangle !t : t : A$ so by (π) on (2), (3) $F \langle \omega.n | !t \rangle t : A$. By $(\nu 4)$ on (1), (4) $T \langle \omega.n | !t \rangle t : A$, closing Θ. So $t : A \in \mathcal{E}(\omega, !t)$.

- **Sum:** Assume $A \in \mathcal{E}(\omega, t)$, so (1) $T \langle \omega \rangle t : A \in \Theta$. Add to Θ (2) $F \langle \omega \rangle t + s : A$ so by (π) on (2), (3) $F \langle \omega.n | t + s \rangle A$ and by $(\pi+)$ on (3), (4) $F \langle \omega.n | t \rangle A$. By (νK) on (1), (5) $T \langle \omega.n | t \rangle A$, closing Θ. So $A \in \mathcal{E}(\omega, t+s)$.

The valuation $\mathcal{V}$ is built such that $\omega \in \mathcal{V}(p)$ iff $T \langle \omega \rangle p \in \Theta$. As Θ is consistent, $\mathcal{V}$ is well-defined and $\mathcal{M}$ is an LP-model. The model $\mathcal{M}$ thus constructed is usually called a *canonical model*.

By structural induction one shows that $\mathcal{M}$ satisfies every signed prefixed formula in Θ. Note that for a formula of the form $T \langle \omega.n | t \rangle A$ one only has to show that $\mathcal{M}, \omega.n \Vdash A$, for the construction of the canonical model guarantees

[4] This condition has a subtle difference from requiring $T \langle \omega \rangle A \in \Theta$.

that $A \in \mathcal{E}(\omega, t)$; furthermore, as Θ is saturated, $T\langle\omega.n\rangle A$. It is also worth noting that formulas of the form $F\langle\omega.n|t\rangle A$ are blocked, that is, no preKELPrule applies to it, and they are always satisfied by $\mathcal{M}$; in fact, a formula of the form $F\langle\omega.n|t\rangle A$ is satisfied by $\mathcal{M}$ if $A \notin \mathcal{E}(\omega, t)$; but the only possible way that $A \in \mathcal{E}(\omega, t)$ is closing Θ with the addition of $F\langle\omega\rangle t : A$, in which case $T\langle\omega\rangle t : A \in \Theta$, which implies that $T\langle\omega.n|t\rangle A$, a contradiction. So $A \notin \mathcal{E}(\omega, t)$ and $F\langle\omega.n|t\rangle A$ is satisfied by $\mathcal{M}$.

The basic case of the structural induction is given by the definition of $\mathcal{V}$, the classical cases are all straightforward and omitted. We show only the modal cases.

Suppose $F\langle\omega\rangle t : A \in \Theta$. As Θ is saturated, it follows that $F\langle\omega.n|t\rangle A \in \Theta$ for some new $\omega.n$. By induction, we have that $A \in \mathcal{E}(\omega, t)$ or $\mathcal{M}, \omega.n \nVdash A$ which implies that $\mathcal{M}, \omega \nVdash t : A$. So $F\langle\omega\rangle t : A$ is satisfied by $\mathcal{M}$.

Suppose $T\langle\omega\rangle t : A \in \Theta$. Assume there is no $\omega.n \in \mathcal{W}$, then the only accessible world from ω is ω itself, so by an application of (νT) one obtains $T\langle\omega.0|t\rangle A \in \Theta$, so the inductive hypothesis gives us that $A \in \mathcal{E}(\omega, t)$ and $\mathcal{M}, \omega \Vdash A$, in which case $\mathcal{M}$ satisfies $T\langle\omega\rangle t : A$. Now assume there is $\omega.n \in \mathcal{W}$ and let $W' = \{\omega.m \in \mathcal{W} |$ for some $m\}$; clearly, W' is the set of all ω-accessible worlds in $\mathcal{W}$. By an application of (νK), one obtains $T\langle\omega.n|t\rangle A \in \Theta$ for each $\omega.n \in W'$. By the induction hypothesis, $A \in \mathcal{E}(\omega, t)$ and $T\langle\omega.n|t\rangle A \in \Theta$ for every ω-accessible $\omega.n$. So $\mathcal{M}, \omega \Vdash A$, that is, $\mathcal{M}$ satisfies $T\langle\omega\rangle t : A$. This finishes the proof. □

Theorem 4.9 (Completeness) *Let $\mathcal{C}$ be a constant specification, such that A is $\mathcal{C}$-valid. Then A has a closed* preKELP*-tableau.*

Proof Suppose that A is $\mathcal{C}$-valid, but there is a preKELP tableau for $F\langle 1\rangle A$ with a KE saturated open branch. By Lemma 4.8, there exists a model that falsifies A, which violates its validity. □

Just note that preKELP, as defined in this section, is $\mathcal{C}$-analytic. That is, all formulas in the conclusion of linear expansion rules are subformulas of one of the premisses; and (APB) is $\mathcal{C}$-analytic.

Finally, a note on termination. It is known that SST tableaux for S4 may have infinite branches due to the interplay of π- and ν-rules. To make SST-tableaux into a decision procedure, a bound is imposed on the length of the prefixes generated.

Fact 4.10 ([Mas00]) *There is a bound b on the length of prefixes on a branch after which either there are no more modal operators or formulae just repeat themselves with a longer prefix. This bound is computed from the initial tableau formula.*

So we can use exactly the same bound, and compute it by transforming the LP-formula into its S4-correspondent. When this bound is reached and the tableau is not closed, an application of (π) at ω, instead of generating a new prefix, would reflexively re-generate ω, so as to create a countermodel as described by the result above.

So preKELP is also a decision procedure for LP.

4.4 Abductive Closing in preKELP

A new closing condition is added to preKELP tableaux. The *abductive closing condition* accepts a weakened closing pair, represented by $\times_{\text{abd}}$, namely

$$F\langle\omega.n|t\rangle A \qquad T\langle\omega.n|t'\rangle A$$

This form of closing has to be validated, which may occur in two distinct contexts:

- If a constant specification $\mathcal{C}$ is provided, validation consists of adding to the branch some formulas of the form $Ta : A$ for $A \in \mathcal{C}$, so as to close the branch.
- If no constant specification is given a priori, one may compute (abduce) a constant specification so as to close the branch.

We call it the validation/abduction of the abductive branch closing. Of course, we do not want to actually introduce the constant specification formulas into the tableau, as that would simply be the previous method, licensed by a $\mathcal{C}$-analytic application of (APB) over $a : A$. A method for performing validation/abduction without the insertion of formulas of the form $a : A$ is described below.

The reason to introduce abductive closing condition is that the tableau becomes *strongly analytic*, that is, all nodes of the tableau are subformulas of its initial formula or are formulas computed during the proof. The rule (APB) is then constrained to be applicable only to formulas that are subformulas of the initial formula A_0.

Note that abductive closing only makes sense if some proof constant occurs in the term t in $F\langle\omega.n|t\rangle A$; otherwise, due to analycity, the addition of formulas of the form $a : A$ would not validate the closing.

Note also that not all forms of terms t have to be taken in consideration. For $t = t' + s'$, the formula $F\langle\omega.n|t' + s'\rangle A$ is a $(\pi+)$-premiss, which yields $F\langle\omega.n|t'\rangle A$ and $F\langle\omega.n|s'\rangle A$, which can be used instead as the F-side of the abductive closing. For $t =!s$, only formulas of the form $!s : s : A$ have to be taken into account, so one of the closing formulas is of the form $F\langle\omega.n|!s\rangle\, s : A$; by inspection, we see that such a formula must have been derived from premiss $F\langle\omega\rangle !s : s : A$, which, by an application of (APB) over $s : A$, yields $F\langle\omega\rangle\, s : A$ (this inference corresponds to rule $(F!)$ in FTLP). By a (π)-expansion one gets $F\langle\omega.n|s\rangle A$, which can be used instead as the F-side of the closing.

For this reason, we deal only with abductive closing when t in $F\langle\omega.n|t\rangle A$ is of the form $t_1 \cdot t_2 \cdots t_n$, $n \geq 2$. Instead of adding formulas of the form $a : A$ to the tableau, one can perform *type checking inference*, which is based on Basic Simple Type Theory [Hin97], to verify if t is typable. In this case, the term t' in $T\langle\omega.n|t'\rangle A$ has to be built from the terms t_i and $\cdot$, for the type inference to be allowed. For example, if $\text{type}(t) = A \supset B$ and $\text{type}(s) = A$, then the expression $t \cdot s$ is typable and receives type B, but the term $s \cdot t$ is not typable; if no type is attributed to t, neither $t \cdot s$ nor $s \cdot t$ is typable. According to this procedure, t is only typable if each t_i in t is either a variable or a constant and,

as mentioned, at least one such t_i must be a constant. If the formats of t or t' are not as described, abductive closing is not applicable.

4.4.1 Branch Closure Validation via Type Checking

The basic idea of branch closure validation is the following. At the creation of a prefix $\langle \omega.n|t\rangle$ by a π-expansion, a type (ie, an LP-formula) is attributed to term t. Suppose $t = t_1 \cdots t_n$ is a product of terms. During subsequent ν-expansions, a type is attributed to each non-constant component t_i; the type of a constant a is A if $A \in \mathcal{C}(a)$. The closure is validated if every term t associated to an active prefix is typable in the sense of Type Theory. Otherwise the closure is rejected.

Type attribution by π- and ν-rules is shown in Figure 8, according to the presentation of π- and ν-expansions in Figure 6.

Rule	*Expression*	*Type attribution*
(π)	t	$\text{type}(t) = A$
$(\pi+)$	t	$\text{type}(t) = A$
	s	$\text{type}(s) = A$
(νK)	x	$\text{type}(x) = A$
(νT)	x	$\text{type}(x) = A$
$(\nu 4)$	$!x$	$\text{type}(!x) = x : A$

Figure 8: Type attribution by π- and ν-rules

Branch closure validation follows the procedure below.

1. Every time a prefix $\langle \omega.n|t\rangle$ is created by a π-rule, a type is attributed to t.

2. Every time a prefix $\langle \omega.n|t'\rangle$ is activated by a ν-rule, a type is attributed to t'.

3. At branch closure, every term defined at 1 has its type checked according to the types attributed in 2 for the same $\omega.n$.

Let $t = t_1 \cdots t_n$ be a product term created in 1. There are a few notable points. First, if all component expressions t_i have a type, type checking is decidable. Second, if some component t_i that is not a proof constant is not attributed a type, then the expression is not typable. Third, if t_i occurs n times in t, there must be n attributions of $\text{type}(t_i)$, then the typing of t must be checked as if each t_i is a distinct expression; this is due to the fact that in LP a term is allowed to serve as evidence to more than one fact; if there are less that n attributions of $\text{type}(t_i)$, t is not typable.

If $n = 1$, t_1 is either a variable or a constant. In both cases, it is immediately typable with its creation type A. In case $t_1 = a$, we verify that $A \in \mathcal{C}(a)$ to

validate its type. However, if the constant specification is not given, we can construct it, by forcing $A \in \mathcal{C}(a)$.

This is the basis of the abduction of constant specifications. We note that the same applies if the type of a proof constant is computed at steps 1 and 2 above.

4.4.2 Abduction of Constant Specifications

Assume no constant specification is given, such that the abductive closing was employed over $F\langle\omega.n|t\rangle A$ and $T\langle\omega.n|t'\rangle A$. Let $t = t_1 \cdots t_n$, $n \geq 1$. The goal of the abductive procedure is to attribute a type to every proof constant occurring in t such that, according to the type attributions of Figure 8, the following tableau closes employing only $(T\supset)$:

$$T\langle 1\rangle \operatorname{type}(t_1), \ldots, T\langle 1\rangle \operatorname{type}(t_n), F\langle 1\rangle \operatorname{type}(t)$$

The idea is to deal with $\cdot$ as function application, which is in fact the behaviour of $\cdot$ in LP, so that the problem above becomes equivalent to inferring the type of a λ-calculus term $\lambda t_1, \ldots, t_n(t)$.

Let $a_1, \ldots, a_k$ be proof constants for which the method above computed a type. The *abduction of a constant specification* is the creation of $\mathcal{C}$ such that $\operatorname{type}(a_i) \in \mathcal{C}(a_i)$, $1 \leq i \leq k$.

Theorem 4.11 (Correctness of Abduction) *Let $\mathcal{C}$ be a constant specification constructed as above for a branch closed with abductive closing. Then it is possible to close the branch with the usual closing condition.*

Proof A proof is constructed that shows that the branch closes. Initially the branch contains $F\langle\omega.n|t\rangle A$, such that $t = t_1 \cdots t_n$, so $\operatorname{type}(t) = A$. The branch also contains, for each non-constant term t_i a node $T\langle\omega.n|t_i\rangle \operatorname{type}(t_i)$. For all constant terms $t_i = a_i$ that the type inference algorithm produced $\operatorname{type}(a_i) \in \mathcal{C}(a_1)$, apply (APB) at ω over $a_i : \operatorname{type}(a_i)$; the F-side closes immediately, so the branch now contains $T\langle\omega\rangle\, a_i : \operatorname{type}(a_i)$ which yields $T\langle\omega.n|a_i\rangle \operatorname{type}(a_i)$. It follows that the branch contains $T\langle\omega.n|t_i\rangle \operatorname{type}(t_i)$ for every t_i, $1 \leq i \leq n$.

Recall that the tableau with $T\langle 1\rangle \operatorname{type}(t_i)$, $1 \leq i \leq n$, and $F\langle 1\rangle A$ closes. As this is a basic simple deduction, only $(T\supset)$ is applicable. At the current branch, each $(T\supset)$ application is replaced by $(T\supset_\omega)$, so as to produce $T\langle\omega|t_1 \cdots t_n\rangle A$, which closes the branch. □

The procedures of abduction and verification is unified, so that the types of all constant specification are computed. If $\mathcal{C}$ is given, the typing is verified if $\operatorname{type}(a) \in \mathcal{C}$. If no $\mathcal{C}$ is given a priori, a constant specification is constructed by making $\operatorname{type}(a) \in \mathcal{C}(a)$. Note that this abduction process always guarantees a finite constant specification.

In accordance with the semantic of proof terms, the abducted formula $\operatorname{type}(a)$ has to be LP-valid, possibly with a PREKELP-tableau with root $F\ \operatorname{type}(a)$. If this is not the case, the validation must be rejected.

A simplified way to compute branch abduction occurs when there is only one constant a at the head of t, that is, $t = a \cdot t_1 \cdots t_n$. In this case, $\operatorname{type}(a) =$

$\text{type}(t_1) \supset (\ldots(\text{type}(t_n) \supset \text{type}(t))\ldots)$. This situation is very frequent in practice and guarantees that $\text{type}(a)$ is provable.

Theorem 4.12 *Let $t = a{\cdot}t_1 \cdots t_n$. Then abducted formula* $\text{type}(a)$ *is* preKELP*-provable.*

Proof As the original tableau closes, a branch containing

$$F\,\langle\omega, t\rangle\,\text{type}(t), T\,\langle\omega, t_1\rangle\,\text{type}(t_1), \ldots, T\,\langle\omega, t_n\rangle\,\text{type}(t_n)$$

closes, which clearly implies that a branch containing

$$F\text{type}(t), T\text{type}(t_1), \ldots, T\text{type}(t_n)$$

also closes. But that is a tableau for $\text{type}(a)$, which is therefore preKELP-provable. □

Next some examples of preKELP tableaux are presented.

4.4.3 Examples and the Abduction Heuristics

Example 4.13 We revisit once more Example 3.3, namely the proof of $t : (A \supset B) \supset (s : A \supset a \cdot s \cdot t : B)$, this time using preKELP-expansion rules and abductive closure.

1.	$F\,\langle 1\rangle\, t : (A \supset B) \supset (s : A \supset a \cdot s \cdot t : B)$	
2.	$T\,\langle 1\rangle\, t : (A \supset B)$	$(F\supset)$ in 1
3.	$F\,\langle 1\rangle\, s : A \supset a \cdot s \cdot t : B$	$(F\supset)$ in 1
4.	$T\,\langle 1\rangle\, s : A$	$(F\supset)$ in 3
5.	$F\,\langle 1\rangle\, a \cdot s \cdot t : B$	$(F\supset)$ in 3
6.	$F\,\langle 1.1 \vert a \cdot s \cdot t\rangle\, B$	(π) in 5 $[\text{type}(a \cdot s \cdot t) = B]$
7.	$T\,\langle 1.1 \vert t\rangle\, A \supset B$	(νK) in 2 $[\text{type}(t) = A \supset B]$
8.	$T\,\langle 1.1 \vert s\rangle\, A$	(νK) in 4 $[\text{type}(s) = A]$
9.	$T\,\langle 1.1 \vert t \cdot s\rangle\, B$	$(T\supset_\omega)$ in 7,8
10.	$\times_{\text{abd}}$	abductive closure 6,9

The abductive closure is applied to close the tableau. Line 6 assigns $\text{type}(a{\cdot}s{\cdot}t)$; lines 7 and 8 assigns $\text{type}(t)$ and $\text{type}(s)$, respectively.

The abduction procedure computes $\text{type}(a) = A \supset ((A \supset B) \supset B)$. So the proof is validated by making $A \supset ((A \supset B) \supset B) \in \mathcal{C}(a)$, as was known from Examples 3.3 and 4.2.

Next, we show two examples in which no validation is required, the tableau either closes or not. It also shows that the presence of the proof constant is crucial for a formula to become LP-valid.

Example 4.14 Consider the two tableaux below, the left one corresponding to $t : (A \supset B) \supset (s : A \supset t \cdot s : B)$ and the right one corresponding to

$t : (A \supset B) \supset (s : A \supset s \cdot t : B)$

1.	$T\langle 1\rangle\, t : (A \supset B)$	
2.	$T\langle 1\rangle\, s : A$	
3.	$F\langle 1\rangle\, t \cdot s : B$	
4.	$F\langle 1.1 \vert t \cdot s\rangle\, B$	(π) in 3
5.	$T\langle 1.1 \vert t\rangle\, A \supset B$	(νK) in 1
6.	$T\langle 1.1 \vert s\rangle\, A$	(νK) in 2
7.	$T\langle 1.1 \vert t \cdot s\rangle\, B$	$(T \supset_\omega)$ in 6,5
	$\times$	

1.	$Tt : (A \supset B)$	
2.	$Ts : A$	
3.	$Fs \cdot t : B$	
4.	$F\langle 1.1 \vert s \cdot t\rangle\, B$	(π) in 3
5.	$T\langle 1.1 \vert t\rangle\, A \supset B$	(νK) in 1
6.	$T\langle 1.1 \vert s\rangle\, A$	(νK) in 2
7.	$T\langle 1.1 \vert t \cdot s\rangle\, B$	$(T \supset_\omega)$ in 6,5

We see that the left tableau closed with the conflict condition, no abduction is needed. The right tableau does not close; abductive closure is not applicable, as no proof constant is found in the decoration of the prefix in $F\langle 1.1 \vert s \cdot t\rangle\, B$.

Both formulas at the root of the tableaux yield the same S4-theorem if proof polynomials are replaced with $\Box$; but only one of them is LP-valid.

One interesting consequence of this process is that once an abduction step yields $A \in \mathcal{C}(a)$ on a branch, the formula $Ta : A$ becomes available to all other branches as a consequence of the closure condition on $F\langle \omega\rangle\, a : A$ for $A \in \mathcal{C}(a)$. This motivates the creation of an Abduction Heuristics.

> **Abduction Heuristics.** Consider a branch containing the signed prefixed formula $F\langle \omega.n \vert a \cdot t_1 \cdots t_n\rangle\, A$. Expand this branch to a point where every t_i has been assigned a type. Then apply (APB) and branch over $T\langle \omega.n \vert t_1 \cdots t_n\rangle\, A$ and $F\langle \omega.n \vert t_1 \cdots t_n\rangle\, A$. The T-side closes immediately with an abduction closure, producing $C \in \mathcal{C}(a)$. The formula $T\langle \omega\rangle\, a : C$ becomes available for F-side (and the rest of the tableau).

The Abduction Heuristics is used in the following example.

Example 4.15 Consider the the LP-translation of S4 theorem $\Box A \supset \Box(\Box A \vee \Box B)$ as $x : A \supset a{\cdot}!x : ((x : A \supset \bot) \supset y : B)$.

			1.	$F\langle 1\rangle\, (x : A \supset a{\cdot}!x : ((x : A \supset \bot) \supset y : B)$	
			2.	$T\langle 1\rangle\, x : A$	$(F \supset)$ in 1
			3.	$F\langle 1\rangle\, a{\cdot}!x : ((x : A \supset \bot) \supset y : B)$	$(F \supset)$ in 1
			4.	$F\langle 1.1 \vert a{\cdot}!x\rangle\, (x : A \supset \bot) \supset y : B$	(π) in 3
			5.	$T\langle 1.1 \vert !x\rangle\, x : A$	$(\nu 4)$ in 2
6.	$T\langle 1.1 \vert !x\rangle\, ((x : A \supset \bot) \supset y : B)$	APB	7.	$F\langle 1.1 \vert !x\rangle\, ((x : A \supset \bot) \supset y : B)$	APB
			8.	$T\langle 1\rangle\, a : (x : A \supset ((x : A \supset \bot) \supset y : B)))$	abduction
	$\times_{abd}$	4,6	9.	$T\langle 1.1 \vert a\rangle\, x : A \supset ((x : A \supset \bot) \supset y : B))$	(νK) in 8
	$x : A \supset ((x : A \supset \bot) \supset y : B)$ $\in \mathcal{C}(a)$		10.	$T\langle 1.1 \vert a.!x\rangle\, (x : A \supset \bot) \supset y : B)$	
			11.	$\times$	6,10

The Abduction Heuristics is applied after line 5. The formula $x : A \supset ((x : A \supset \bot) \supset y : B) \in \mathcal{C}(a)$ was computed on the T-side and then applied on the F-side, which closed normally.

This heuristic strategy actually cheats on analycity. However, the main claim for analycity (in terms of proof construction) is that non-analytic proofs rely on "guessing", which cannot be automated. This criticism does not apply to this strategy, as the abduction step is totally algorithmic. The proof search method only employs its own "resources", in the form of subformulas or of some other derived/abduced/learned formulas, which result from the analysis and inspection of the proof itself. In the sense that only subformulas and internally computed formulas are used, strong analycity is observed.

Another possible criticism to the use of the Abduction Heuristics is that preKELP-proofs applying it no longer correspond directly to SST proofs. However, as will be shown next, the Abduction Heuristics may reduce considerably the complexity of a proof.

4.4.4 Strong Analycity and Complexity

Theorem 4.16 preKELP *tableaux with constant specification abduction is strongly analytic.*

Proof Just note that all preKELP linear and branching inferences are strongly analytic. If the result of constant specification abduction is used in the proof, it remains strongly analytic. Note that, in this case, the closing $\times_c$ is never used. □

Let the size of a signed formula $T/F\,\langle\omega, t\rangle\,A$ be $|t| + |A|$. A preKELP expansion can lead to terms of exponential size terms even in a single branch tableau, as shown by the following result.

Lemma 4.17 *There is an* LP*-valid formula A which has* preKELP*-tableau nodes of size at least $\sqrt{|A|}^{\sqrt{|A|}}$ if no Abduction Heuristics is applied.*

Proof Consider a formula $A = x : A_0 \supset \alpha_1 \supset \cdots \supset \alpha_n \supset a \cdot x_0 \cdots x_n : A_n$, where $\alpha_{i+1} = \underbrace{A_i \supset \ldots \supset A_i}_{n \text{ times}} \supset A_{i+1}$. So $|A| = \Omega(n^2)$. The expansion of $F\,\langle 1\rangle\,A$ leads, with no branching, to the following formulas:

$$\begin{array}{l} F\,\langle 1.1|a \cdot x_0 \cdots x_n\rangle\,A_n \\ T\,\langle 1.1|x_0\rangle\,A_0 \\ T\,\langle 1.1|x_1\rangle\,A_0 \supset \ldots \supset A_0 \supset A_1 \\ \qquad\qquad \vdots \\ T\,\langle 1.1|x_n\rangle\,A_{n-1} \supset \ldots \supset A_{n-1} \supset A_n \end{array}$$

By n successive applications of $(T \supset_\omega)$ one obtains $T\,\langle 1.1|t_1\rangle\,A_1$, such that x_0 occurs n times in t_1. Next, by n successive applications of $(T \supset_\omega)$ one obtains $T\,\langle 1.1|t_2\rangle\,A_2$, such that x_0 occurs n^2 times in t_2. Inductively, one obtains $T\,\langle 1.1|t_n\rangle\,A_n$, such that x_0 occurs n^n times in ω_n, which closes the tableau. This last node is of size $\Omega(\sqrt{|A|}^{\sqrt{|A|}})$.

□

There are several interesting points to note in the proof above. First, as only linear rules have been applied, this is in fact the LP-proof associated to the minimal S4-proof of the corresponding S4-formula. And this LP-proof is in EXPSPACE.

Second, the abduction process computes a formula that is of size linear with the initial formula. So the size of the constant specification is not exponential.

Third, we can apply the Abduction Heuristics before the first application of $(T \supset_\omega)$, computing exactly the same formula. This formula then becomes available and in a linear number of applications of $(T \supset_\omega)$, the tableau closes.

This actually shows that LP-proofs that correspond directly to S4-proofs may be deal with exponential terms, but some preKELP-only manipulation may restore the size correspondence.

Finally, if we are constructing a preKELPtableau for an invalid formula, say, one obtained by replacing A_n by B, the saturation of a branch will lead to an exponential term in exactly the same way as in Lemma 4.17. So the preKELP-decision problem is in EXPSPACE.

5 Related Work

The aim of this section is to position the results obtained in the previous section in the context of recent work described in the literature. In particular, there seems to be a few clarifications in need due to apparent contradiction between the EXPSPACE complexity for abduction and some recent complexity results in the literature, namely:

- A complexity upper bound for the decision problem for LP in the class Π_2^p in the polynomial hierarchy, obtained by Kuznets [Kuz00]. This is considered a much better complexity bound in respect to logic S4, which is known to be PSPACE-complete [Lad77].
- An NP-complete decision procedure proposed by Krupski [Kru06] for a fragment of LP, known as the *Reflected Logic of Proofs*; this fragment has the same expressive power as LP itself.
- A result by Brezhenv and Kuznets, in which a relation between LP theorems can be computed from S4 proofs in polynomial time [BK06].

To understand how all these results fit in, one has to recall the definition of the *realization* of an S4-formula A into an LP-formula A^r, that is, to substitute proof polynomials for all occurrences of $\Box$ in A, which yields A^r. A central result for this discussion is the following.

Fact 5.1 (Realization Theorem) *The modal formula A is* S4*-provable iff A^r is* LP*-provable for some realization r.*

This result was originally proved by Artemov [Art95, Art01] with two distinct algorithms, both of which, however, generated a realization A^r whose

LP-derivation was exponential with the size of the S4-derivation of A. This result was greatly improved by Brezhnev and Kuznets [BK06], by presenting a realization algorithm in which the size of the LP-derivation of A^r, and of every term occurring in it, is polynomial on the size of the S4-derivation of A. However, the authors of [BK06] note that, since S4-decision is PSPACE-complete, it may be the case that the S4-derivation of A is exponential, in which case the LP-derivation of A^r, or some LP-formula in it (including A^r itself), may be exponential with respect to $|A|$; this is consistent with Lemma 4.17.

For recent analyses on the structure of realizations, see [Fit07, Fit08].

With respect to the Π_2^p upper bound for LP-derivability in [Kuz00], note that the exponential lower bound of Lemma 4.17 only applies to constant specification abduction using strongly analytic preKELP-tableaux. In fact, the construction of preKELP-proofs with constant specification abduction can be seen as an intermediate problem between deciding an LP-formula and computing the realization of an S4-formula. In terms of complexity, that is where it lies.

The exponential lower bound of Lemma 4.17 cannot be applied to the general LP-decision problem, which is a different problem, even if it is a related one. In the derivation presented in Lemma 4.17, there are two possibilities of closing the tableau without any abduction. In both cases, no contradiction is obtained in relation to the Π_2^p upper bound for LP-derivability.

First, if we substitute the exponentially sized term computed in the derivation for $a \cdot x_0 \cdots a_n$ in the input, the preKELP-tableau closes without any abduction and the size of the input itself becomes exponential with respect to the size of the corresponding S4-formula. However, the size of the LP-derivation and the sizes of all formulas in it are polynomial with respect to the size of the input LP-formula. Second, note that the constant specification computed by abduction in Lemma 4.17 *is not* exponential. If the beginning of the derivation branches over it, following the construction of Theorem 4.11, the tableau closes without generating any exponential terms and without any abduction.

The conclusion is that the price for not having *a priori knowledge* of the constant specification is the potential of an exponential explosion on the sizes of terms in an analytic derivation.

Finally, we note that in [Kuz00] formulas are stored as direct acyclic graphs, and employs algorithms that manipulate this data structure to avoid the combinatorial explosion. Analogously, the particular exponential term computed in Lemma 4.17 can be encoded in polynomial space. In fact, let $x \circ^1 y = x \cdot y$ and $x \circ^{i+1} y = x \cdot (x \circ^i y)$; then the exponential term ω_n is represented as $x_0 \circ^n x_1 \circ^n \ldots \circ^n x_n$, where $\circ^n$ associates to the left. So the exponentiality of a derivation may depend on how one computes the size of terms.

6 Conclusion and Further Work

In this paper we presented two analytic tableau methods for the Logic of Proofs. Instead of basing them on a cut-free calculus, the methods presented were based

on a calculus with analytic cuts and extend the KE tableau method. We have shown that both $\mathcal{C}$-analytic (KELP) and strongly analytic (preKELP) methods for LP exist. The analysis of the KELP method showed that to solve the problem of LP-analycity, one should focus on the role of proof constants in an LP-proof; apart from that, KELP has very similar properties to Fitting's tableau method.

The preKELP-tableau method performs constant specification abduction. Furthermore, it tries to mirror on a step-by-step analytic S4-SST proofs. We have shown that in such cases it is possible that the preKELP-proof may suffer an exponential explosion on the size of terms. The Abduction Heuristics helps avoid this exponential growth, but eliminates the strong relation with S4-proofs. It is worth noting that such heuristics relies on the use of analytic cuts. Analycity is preserved by the use of the Abduction Heuristics, but that requires one to accept analycity in the sense that a proof has to be constructed on its own "resources".

Future work includes the use of a modification of preKELP to compute realizations. With regards to complexity, it would be interesting to see if some modification of LP can lead to less complex decision procedures in which the proof polynomials actually guide the process, instead of needing extra abduction processing.

One may also investigate if constant specification abduction can be performed in a method based on cut-free proofs. A different strategy than that of the Abduction Heuristics proposed here would be necessary.

Acknowledgements

The author would like to thank an anonymous referee for very detailed comments which helped a lot to improve the quality of the paper. The author would also like to thank Renata Wassermann, who helped checking grammar and content of an earlier version of the paper.

References

[Art95] Sergei Artemov. Operational modal logic. Technical Report MSI 95-29, Cornell University, December 1995.

[Art01] Sergei Artemov. Explicit provability and constructive semantics. *Bulletin of Symbolic Logic*, 7(1):1–36, 2001.

[BDS06] Henk Barendregt, Wil Dekkers, and Richard Statman. Typed lambda calculus, volume I. Preliminary Version, 2006. Available at `ftp://ftp.cs.ru.nl/pub/CompMath.Found/`.

[BK06] Vladimir Brezhnev and Roman Kuznets. Making knowledge explicit: How hard it is. *Theor. Comput. Sci.*, 357(1-3):23–34, 2006.

[D'A92] Marcello D'Agostino. Are tableaux an improvement on truth-tables? — Cut-free proofs and bivalence. *Journal of Logic, Language and Information*, 1:235–252, 1992.

[D'A99] Marcello D'Agostino. Tableau methods for classical propositional logic. In Marcello D'Agostino, Dov Gabbay, Rainer Haehnle, and Joachim Posegga, editors, *Handbook of Tableau Methods*, pages 45–124. Kluwer, 1999.

[DM94] Marcello D'Agostino and Marco Mondadori. The taming of the cut. Classical refutations with analytic cut. *Journal of Logic and Computation*, 4(285–319), 1994.

[Fit83] Melvin Fitting. *Proof Methods for Modal and Intuitionistic Logic*. Reidel, 1983.

[Fit05] Melvin Fitting. The logic of proofs, semantically. *Annals of Pure and Applied Logic*, 132:1–25, 2005.

[Fit07] Melvin Fitting. Realizations and LP. In *Logical Foundations of Computer Science, (LFCS 2007), New York, NY*, volume 4514 of *Lecture Notes in Computer Science*, pages 212–223. Springer, 2007.

[Fit08] Melvin Fitting. S4LP and local realizability. In *Third International Computer Science Symposium in Russia (CSR 2008)*, volume 5010 of *Lecture Notes in Computer Science*, pages 168–179. Springer, 2008.

[Hin97] J. Roger Hindley. *Basic simple type theory*. Cambridge University Press, New York, NY, USA, 1997.

[Kru06] Nikolai V. Krupski. On the complexity of the reflected logic of proofs. *Theor. Comput. Sci.*, 357(1-3):136–142, 2006.

[Kuz00] Roman Kuznets. On the complexity of explicit modal logics. In Peter Clote and Helmut Schwichtenberg, editors, *Computer Science Logic (CSL)*, volume 1862 of *Lecture Notes in Computer Science*, pages 371–383. Springer, 2000.

[Lad77] Richard E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM J. Comput.*, 6(3):467–480, 1977.

[Mas00] Fabio Massacci. Single step tableaux for modal logics: Methodology, computations, algorithms. *Journal of Automated Reasoning*, 24(3):319–364, 2000.

[Mkr97] A. Mkrtychev. Models for the logic of proofs. In *4th International Symposium on Logical Foundations of Computer Science*, volume 1234 of *LNCS*, pages 266–275. Springer, 1997.

[Ren04] Bryan Renne. Tableaux for the logic of proofs. Technical Report TR-2004001, CUNY Graduate Center PhD Program in Computer Science, 2004.

[Smu68a] Raymond M. Smullyan. Analytic Cut. *Journal of Symbolic Logic*, 33:560–564, 1968.

[Smu68b] Raymond M. Smullyan. *First-Order Logic*. Springer-Verlag, 1968.