Labelled Tableau Calculi Generating Simple Models for Substructural Logics*

Kazushige Terui†
Department of Philosophy, Keio University
E-mail: terui@abelard.flet.keio.ac.jp

Abstract

In this paper we apply the methodology of Labelled Deductive Systems to the tableau method in order to obtain a deductive framework for substructural logics which incorporates the facility of *model generation*. For this special purpose, we propose new labelled tableau calculi **TL** and $\mathbf{TL_e}$ for two substructural logics **L** (essentially the Lambek calculus) and $\mathbf{L_e}$ (the multiplicative fragment of intuitionistic linear logic). The use of labels makes it possible to generate countermodels in terms of a certain very simple semantics based on monoids, which we call the *simple semantics*. We show that, given a formula C as input, every nonredundant tableau construction procedure for **TL** and $\mathbf{TL_e}$ terminates in finitely many steps, yielding either a tableau proof of C or a *finite* countermodel of C in terms of the simple semantics. It shows the finite model property for **L** and $\mathbf{L_e}$ with respect to the simple semantics.

1 Introduction

In this paper we apply the methodology of Labelled Deductive Systems (LDS, [Gab96]) to the tableau method ([Smu68], [Fit87]) in order to obtain a deductive framework for substructural logics which incorporates the facility of *model generation*. Gabbay and D'Agostino ([DG94], [Gab96]) combine LDS with the classical analytic deduction system **KE** ([DM94]) and propose a labelled tableau framework $\mathbf{LKE}_\mathcal{S}$ for substructural logics[1]. In contrast with [DG94] which aims at giving a general framework of analytic deduction for a wide range of logics, we introduce our labelled tableau framework for the special purpose of generating countermodels for some specific substructural logics (as we discuss in Section 6). We believe that our work opens another possibility of applying LDS to analytic deduction.

Model generation plays an important role in automated reasoning. Given a conjecture C, the most succinct way to refute C would be to show a counterexample, or a *countermodel*, of C. Moreover,a countermodel tells us something about what is wrong in C, and therefore helps us improve the original conjecture C to a more reasonable one. By a *model generation procedure* we mean a systematic procedure to generate a countermodel for a given conjecture C. For such a procedure to be of any use in practice, it should at least satisfy the following requirements;

*This work was partially supported by the Spinoza project 'Logic in Action' at ILLC, the University of Amsterdam.

†Research Fellow of the Japan Society for the Promotion of Science.

[1][DG94] distinguishes their framework from the tableau method, while [Gab96] calls it tableaux. We follow the latter usage.

(i) it should yield a countermodel in terms of a sufficiently simple semantics, and

(ii) it should yield a finite, or at least finitely representable, countermodel if there exists any; in particular, for logics which have the finite model property it should always yield a finite countermodel when applied to an invalid conjecture[2].

The purpose of this paper is to give a deductive framework for substructural logics with the facility of model generation, that is, a deductive framework which offers not only a proof construction procedure but also a model generation procedure which meets the above requirements.

It is well known that the tableau method directly offers such model generation procedures for a wide range of logics, including classical logic, intuitionistic logic, and some modal logics (see [Fit83] for exposition). Hence it is natural to employ the tableau method for our purpose. Several tableau calculi have been proposed for substructural logics; for example [MB79] for relevance logic and [MMB95] for linear logic. However, their calculi do not have the model generation facility.

To develop a suitable framework for our purpose, we make use of the LDS methodology. According to the LDS view, the basic units of deduction are not just formulas but labelled formulas, and the labels are used to incorporate some metalevel information such as semantics and pragmatics into syntax. In this paper we employ two specific sets of labels. Their roles are twofold.

Firstly, the labels allow us to generate countermodels in terms of a sufficiently simple semantics. Labels are often used to prove a completeness theorem with respect to a certain simple semantics to which the Lindenbaum-Tarski method cannot apply (e.g., [Bus86], [Kur94], [Pan94], [OT98]). Our labels play the same role as in those works; indeed, our sets of labels are direct descendants of Buszkowski's one ([Bus86]).

Secondly, the labels establish a tight correspondence between our labelled tableaux and bottom-up proof search trees of the sequent calculus. The idea is to introduce a certain condition on the labels and to identify a set of labelled formulas whose labels satisfy the condition with a sequent. This enables us to view a tableau as representing (a part of) a proof search tree of the sequent calculus. Since all the logics considered in this paper have the property that all proof search trees are finite, the correspondence gives us an assurance that every nonredundant tableau construction procedure terminates in finitely many steps.

In this paper we start our investigation with two basic substructural logics $\mathbf{L}$ (the Lambek calculus [Lam58] which allows the empty antecedent) and $\mathbf{L_e}$ (the multiplicative fragment of intuitionistic linear logic [Gir87][Tro92]). The reason for our choice of $\mathbf{L}$ and $\mathbf{L_e}$ is that these logics are known to be complete with respect to a certain extremely simple semantics based on monoids, which we call the *simple semantics* ([Bus86]), and they are also known to have the finite model property ([OT]). Hence, they provide a good starting point for the study of model generation for substructural logics in general. The syntax and the semantics are given in Section 2.

In Section 3 we introduce the labelled tableau calculi $\mathbf{TL}$ and $\mathbf{TL_e}$. In Section 4, we prove (i) the equivalence of $\mathbf{L}$ ($\mathbf{L_e}$, resp.) and $\mathbf{TL}$ ($\mathbf{TL_e}$, resp.), and (ii) the completeness of $\mathbf{TL}$ ($\mathbf{TL_e}$, resp.) with respect to the finite simple models (the finite commutative simple models,

[2]One might further require that a reasonable model generation procedure should yield a countermodel which is *minimal* in some sense (e.g., [Hin88], [BY96] for first order classical logic). However, the study of model generation for substructural logics has just begun, and the notion of minimality is difficult to state for the models of substructural logics. Hence, we are not concerned with the minimality requirement in this paper.

resp.). During the proof we show that given a formula C as input every nonredundant tableau construction procedure terminates in finitely many steps, yielding either a closed tableau for C or a tableau for C which contains an open completed branch; from the latter we can construct a finite countermodel of C in terms of the simple semantics. The significance of our result is, in the author's opinion, as follows;

1. From the practical point of view, the result shows that one can quite easily give a decision procedure for **TL** and $\mathbf{TL_e}$ which yields either a proof or a finite countermodel for every formula C. Indeed, *every* nonredundant tableau construction procedure serves as such a decision procedure. Thus we claim that our framework embodies the model generation facility very naturally.

2. From the theoretical point of view, the result shows that **L** ($\mathbf{L_e}$, resp.) has the finite model property with respect to the simple semantics (the commutative simple semantics, resp.). The finite model property has already been shown for these logics in [OT][3]. There, however, the property is shown with respect to the phase semantics, which properly extends the simple semantics with some closure operations. Hence the result of the present paper may be still of theoretical importance by itself (see Remark 2.1 below).

In Section 5 we discuss some extensions of **TL** and $\mathbf{TL_e}$. In particular, it is mentioned that **TL** and $\mathbf{TL_e}$ can be extended with the weakening (monotonicity) rule if we allow a slightly complicated semantics (with partial orderings on monoids).

As mentioned above, there is another labelled tableau framework $\mathbf{LKE}_\mathcal{S}$ of [DG94]. We do not claim that our framework is superior to $\mathbf{LKE}_\mathcal{S}$ for every purpose, but we do claim that our framework works much better for the purpose of generating countermodels for **L** and $\mathbf{L_e}$. To make the point clear we compare our framework with $\mathbf{LKE}_\mathcal{S}$ in Section 6.

2 Preliminaries

In Subsection 2.1 we give the syntax of two sequent calculi **L** and $\mathbf{L_e}$, and introduce the simple semantics. In Subsection 2.2 we define the signed formulas and the uniform notation for them. Then, we introduce signed sequent calculi **L'** and $\mathbf{L'_e}$, which will be used in later sections.

2.1 Syntax of L and $\mathbf{L_e}$ and the Simple Semantics

We assume that an infinite set *Atom* of atomic formulas is given. The set *Form* of formulas is defined by

$$Form ::= Atom \mid Form \otimes Form \mid Form \multimap Form \mid Form \circ\!\!- Form.$$

Lists of formulas are denoted by Γ, Δ, etc. A *sequent* is of the form $\Gamma \vdash C$.

Sequent calculus **L** (essentially the Lambek calculus [Lam58]) is defined by the inference rules in Table 1 (i) [4]. Sequent calculus $\mathbf{L_e}$ (the multiplicative fragment of intuitionistic linear

[3] For the implicational fragment of $\mathbf{L_e}$, Buszkowski[Bus96] shows the finite model property result. [OT] gives more general results for a wide range of substructural logics with multiplicative/additive conjunctions and additive disjunction.

[4] One obtains the original Lambek calculus in [Lam58] if we add the restriction that Γ should not be empty in $\multimap R$ and $\circ\!\!- R$. All the results of this paper can be easily extended to this original calculus.

$$
\begin{array}{llll}
\text{(i)} & \dfrac{}{A \vdash A}\ \mathit{Identity} & \dfrac{\Delta \vdash A \quad \Gamma_1, A, \Gamma_2 \vdash C}{\Gamma_1, \Delta, \Gamma_2 \vdash C}\ \mathit{Cut} & \\[2ex]
 & \dfrac{\Gamma_1, A, B, \Gamma_2 \vdash C}{\Gamma_1, A \otimes B, \Gamma_2 \vdash C}\ \otimes L & \dfrac{A, \Gamma \vdash B}{\Gamma \vdash A \multimap B}\ \multimap R & \dfrac{\Gamma, A \vdash B}{\Gamma \vdash B \circ\!- A}\ \circ\!- R \\[2ex]
 & \dfrac{\Gamma_1 \vdash A \quad \Gamma_2 \vdash B}{\Gamma_1, \Gamma_2 \vdash A \otimes B}\ \otimes R & \dfrac{\Delta \vdash A \quad \Gamma_1, B, \Gamma_2 \vdash C}{\Gamma_1, \Delta, A \multimap B, \Gamma_2 \vdash C}\ \multimap L & \dfrac{\Delta \vdash A \quad \Gamma_1, B, \Gamma_2 \vdash C}{\Gamma_1, B \circ\!- A, \Delta, \Gamma_2 \vdash C}\ \circ\!- L \\[2ex]
\hline
\text{(ii)} & \dfrac{\Gamma_1, A, B, \Gamma_2 \vdash C}{\Gamma_1, B, A, \Gamma_2 \vdash C}\ \mathbf{e} & &
\end{array}
$$

Table 1: Inference Rules of **L** and $\mathbf{L_e}$

logic [Gir87][Tro92]) is obtained by adding the exchange (permutation) rule **e** in Table 1 (ii) to **L**. Note that in $\mathbf{L_e}$, $A \multimap B$ and $B \circ\!- A$ are mutually derivable.

Now we describe the *simple semantics*. It is called GS_1 in [Bus86], a variant of the *generalized standard (GS) semantics* for the Lambek calculus. It is also called the semantics of *powerset residuated monoids over monoids* in [Bus97]. In [OT98] the commutative simple semantics is called the *naive phase semantics*, because it is precisely the intuitionistic phase semantics ([Tro92]) without closure operations.

A *simple model* (M, v) consists of a monoid $M = (M, \cdot, \epsilon)$ with a *valuation* v which maps each atomic formula to a subset of M. A simple model (M, v) is said to be *commutative* if M is commutative. A valuation v is extended to non-atomic formulas by

- $v(A \otimes B) = \{x \cdot y | x \in v(A), y \in v(B)\}$,
- $v(A \multimap B) = \{y | \forall x \in v(A), x \cdot y \in v(B)\}$,
- $v(B \circ\!- A) = \{y | \forall x \in v(A), y \cdot x \in v(B)\}$.

We say that $B_1, \ldots, B_n \vdash A$ is *satisfied* in (M, v) if $v(B_1 \otimes \cdots \otimes B_n) \subseteq v(A)$. We also say that A is *satisfied* in (M, v) if $\epsilon \in v(A)$.

Remark 2.1 One obtains a *GS-model* if a monoid M in the above definition is replaced with a semigroup. The completeness of the Lambek calculus (**L**, resp.) with respect to the GS-models (the simple models, resp.) is due to Buszkowski ([Bus86]). The completeness of $\mathbf{L_e}$ with respect to the commutative simple semantics can be proved essentially in the same way.

A GS-model (M, v) is called a *language model (L-model)* if M is a free semigroup. Pentus ([Pen94]) refines Buszkowski's result by showing that the Lambek calculus is complete with respect to the L-models. But, of course, every free semigroup is infinite, hence one cannot hope to obtain any finite model property result. In this paper we refine Buszkowski's result in another direction, that is, we show that **L** ($\mathbf{L_e}$, resp.) is complete with respect to the *finite* simple models (simple commutative models, resp.). We can also show that the Lambek calculus is complete with respect to the *finite* GS-models in the same way.

(i)	$\dfrac{}{TA, FA}\ \textit{Identity}$	$\dfrac{\Gamma, FA \quad TA, \Delta}{\Gamma, \Delta}\ \textit{Cut}$	$\dfrac{U, \Gamma}{\Gamma, U}\ \textit{Cyclic Shift}$
	$\dfrac{\Gamma, TA, TB}{\Gamma, TA \otimes B}\ \otimes T$	$\dfrac{\Gamma, FB, TA}{\Gamma, FA \multimap B}\ \multimap F$	$\dfrac{\Gamma, TA, FB}{\Gamma, FB \multimapinv A}\ \multimapinv F$
	$\dfrac{\Gamma, FB \quad FA, \Delta}{\Gamma, FA \otimes B, \Delta}\ \otimes F$	$\dfrac{\Gamma, FA \quad TB, \Delta}{\Gamma, TA \multimap B, \Delta}\ \multimap T$	$\dfrac{\Gamma, TB \quad FA, \Delta}{\Gamma, TB \multimapinv A, \Delta}\ \multimapinv T$
(ii)	$\dfrac{\Gamma, V, U, \Delta}{\Gamma, U, V, \Delta}\ \mathbf{e}$		

Table 2: Inference rules of **L'** and **L'**$_{\mathbf{e}}$

2.2 Signed Formulas, Uniform Notation and Signed Sequent Calculi L' and L'$_{\mathbf{e}}$

A *signed formula* is an expression either of the form TA or of the form FA where $A \in Form$. Signed formulas are denoted by U, V, etc. By means of signed formulas, we may introduce alternative formulations of **L** and **L**$_{\mathbf{e}}$. A *signed sequent* is a list of signed formulas $U_1, \ldots, U_n$ where exactly one U_i is signed F. A signed sequent $TA_1, \ldots, TA_k, FB, TC_1, \ldots, TC_l$ naturally corresponds to sequent $C_1, \ldots, C_l, A_1, \ldots, A_k \vdash B$. We define two signed sequent calculi **L'** and **L'**$_{\mathbf{e}}$ in which the basic units of deduction are signed sequents. **L'** is defined by the inference rules in Table 2 (i), where Γ and Δ denote lists of signed formulas. **L'**$_{\mathbf{e}}$ is obtained by adding **e** to **L'**.

It is easily seen that every inference rule yields a signed sequent (i.e., a list which contains exactly one formula signed F) given signed sequent(s) as premise(s). The following proposition is easily shown.

Proposition 2.1
(1) A is provable in **L** *if and only if FA is provable in* **L'**.
(2) A is provable in **L**$_{\mathbf{e}}$ *if and only if FA is provable in* **L'**$_{\mathbf{e}}$.

We make use of Smullyan's uniform notation ([Smu68]) for the signed formulas. Non-atomic signed formulas fall into two groups, the α *formulas* (of the form $TA \otimes B$ or $FA \multimap B$ or $FB \multimapinv A$) and the β *formulas* (of the form $FA \otimes B$ or $TA \multimap B$ or $TB \multimapinv A$). For each α formula (β formula, resp.) two components α_1 and α_2 (β_1 and β_2, resp.) are defined as in Table 3. Note that these components are defined in such a way that the logical inference rules ($\otimes T, \otimes F, \multimap T, \multimap F, \multimapinv T, \multimap F$) in Table 2 can be summarized as

$$\frac{\Gamma, \alpha_1, \alpha_2}{\Gamma, \alpha}\ \alpha \qquad\qquad \frac{\Gamma, \beta_1 \quad \beta_2, \Delta}{\Gamma, \beta, \Delta}\ \beta.$$

α	α_1	α_2	β	β_1	β_2
$TA \otimes B$	TA	TB	$FA \otimes B$	FB	FA
$FA \multimap B$	FB	TA	$TA \multimap B$	FA	TB
$FB \circ\!\!- A$	TA	FB	$TB \circ\!\!- A$	TB	FA

Table 3: Uniform Notation

3 Labelled Tableau Calculi TL and $\mathbf{TL_e}$

In Subsection 3.1 we define two specific sets of labels. Using them, two labelled tableau calculi **TL** and $\mathbf{TL_e}$ are defined in Subsection 3.2.

3.1 Labels

In this subsection we define two sets $\mathcal{L}$ and $\mathcal{L}_{\mathbf{com}}$ of labels. $\mathcal{L}$ is a monoid and serves as the labels for **TL** tableau calculus, while $\mathcal{L}_{\mathbf{com}}$ is a commutative monoid and serves for $\mathbf{TL_e}$. $\mathcal{L}$ and $\mathcal{L}_{\mathbf{com}}$ have their origin in Buszkowski's ND-terms ([Bus86]) which are used to set up a certain labelled natural deduction calculus for the Lambek calculus (see also [Bus97] for the explanation). Pankrat'ev[Pan94] exploits Buszkowski's labels to show the completeness of **L** with respect to a certain subclass of relational models. Two fundamental lemmas stated below (Lemma 3.1 and Lemma 3.2) are essentially due to [Bus86] and [Pan94], respectively.

Definition 3.1 Let **N** be the set of natural numbers. The set $\mathcal{B}$ is defined as follows;
• $\epsilon, \star \in \mathcal{B}$;
• $a \in \mathcal{B}$ and $i \in \mathbf{N}$, then $(a,i)_1, (a,i)_2 \in \mathcal{B}$ (i is called an *index*);
• if $a \in \mathcal{B}$ and $b \in \mathcal{B}$ then $(ab) \in \mathcal{B}$.

Let $\equiv$ be the least congruence relation on $\mathcal{B}$ satisfying the following;
(**assoc**) $a(bc) \equiv (ab)c$;
(**unit**) $\epsilon a \equiv a \equiv a\epsilon$,
and let $\equiv_{\mathbf{com}}$ be the least congruence relation on $\mathcal{B}$ satisfying (**assoc**), (**unit**) above and (**com**) $(ab) \equiv (ba)$.
These relations induce two quotient structures on $\mathcal{B}$, $\mathcal{B}/\equiv$ and $\mathcal{B}/\equiv_{\mathbf{com}}$, respectively. In the sequel, elements of $\mathcal{B}/\equiv$ and $\mathcal{B}/\equiv_{\mathbf{com}}$ are also denoted by a, $(a,i)_1$, ab, etc., and unneccessary parentheses are sometimes omitted.

Let $\mapsto$ be the least binary relation on $\mathcal{B}$ satisfying the following;
• $(b,i)_1(b,i)_2 \mapsto b$;
• If $b \mapsto b'$ then $(b,i)_j \mapsto (b',i)_j$ for $j \in \{1,2\}$, $(ab) \mapsto (ab')$ and $(ba) \mapsto (b'a)$.
The reflexive transitive closure of $\equiv \circ \mapsto \circ \equiv$ is denoted by $\mapsto^*$ (here $\circ$ means the composition of two relations), and similarly, the reflexive transitive closure of $\equiv_{\mathbf{com}} \circ \mapsto \circ \equiv_{\mathbf{com}}$ is denoted by $\mapsto^*_{\mathbf{com}}$. $\mapsto^*$ ($\mapsto^*_{\mathbf{com}}$, resp.) may be regarded as relation on $\mathcal{B}/\equiv$ ($\mathcal{B}/\equiv_{\mathbf{com}}$, resp.).

Lemma 3.1 *(cf. [Bus86])*
(1) $\mapsto^*$ is confluent and terminating on $\mathcal{B}/\equiv$;
(2) $\mapsto^*_{\mathbf{com}}$ is confluent and terminating on $\mathcal{B}/\equiv_{\mathbf{com}}$.

α-expansion rule	β-expansion rule	Closure condition
$\dfrac{\alpha : a}{\begin{array}{c}\alpha_1 : (a,i)_1 \\ \alpha_2 : (a,i)_2\end{array}}$	$\dfrac{\beta : b}{\beta_1 : b \bullet c \mid \beta_2 : a \bullet b}$	$\dfrac{\begin{array}{c}TA : a \\ FA : b\end{array}}{\times}$
where i is a fresh index.	where $a \bullet b \bullet c$ is total.	where $a \bullet b$ is total.

Table 4: Tableau Expansion Rules and Closure Condition

Proof. (1) is essentially equivalent to Lemma 2 and Lemma 3 of [Bus86]. (2) can be shown similarly. ∎

As a corollary, every $a \in \mathcal{B}/\equiv$ has a unique *normal form* $a^\bullet \in \mathcal{B}/\equiv$. Similarly, every $a \in \mathcal{B}/\equiv_{\mathbf{com}}$ has a unique normal form $a^\bullet \in \mathcal{B}/\equiv_{\mathbf{com}}$. Let us denote the set of terms in $\mathcal{B}/\equiv$ (resp. $\mathcal{B}/\equiv_{\mathbf{com}}$) which are in normal form by $\mathcal{L}$ (resp. $\mathcal{L}_{\mathbf{com}}$). Write $a \bullet b$ to denote $(ab)^\bullet$. Then it holds that $\epsilon \bullet a = a \bullet \epsilon = a$, $(a \bullet b) \bullet c = a \bullet (b \bullet c)$ both for $\mathcal{L}$ and for $\mathcal{L}_{\mathbf{com}}$, and that $a \bullet b = b \bullet a$ for $\mathcal{L}_{\mathbf{com}}$. Therefore $(\mathcal{L}, \bullet, \epsilon)$ (resp. $(\mathcal{L}_{\mathbf{com}}, \bullet, \epsilon)$) forms a monoid (resp. a commutative monoid). Elements of $\mathcal{L}$ and $\mathcal{L}_{\mathbf{com}}$ are called **TL**-*labels* and $\mathbf{TL_e}$-*labels*, respectively.

Definition 3.2 A **TL**-label ($\mathbf{TL_e}$-label, resp.) a is said to be *total* if for some **TL**-labels ($\mathbf{TL_e}$-labels, resp.) b and c, $b_1 \bullet b_2 = \star$ and $a = b_2 \bullet b_1$.

It holds by definition that a $\mathbf{TL_e}$-label a is total only if $a = \star$.

The following lemma is useful when proving the soundness of our labelled tableau calculi (Theorem 4.1).

Lemma 3.2 *(cf. [Pan94]) For $a_1, a_2, b, c \in \mathcal{L}$, $a_1 \bullet b \bullet a_2 = a_1 \bullet c \bullet a_2$ implies $b = c$. The same property holds for $\mathcal{L}_{\mathbf{com}}$.*

Proof. See Lemma 3 of [Pan94]. ∎

3.2 Labelled Tableau Calculi TL and $\mathbf{TL_e}$

A **TL**-*labelled signed formula* ($\mathbf{TL_e}$-*labelled signed formula*, resp.) is of the form $U : a$ where a is a **TL**-label ($\mathbf{TL_e}$-label, resp.) and U is a signed formula. A list of labelled signed formulas $U_1 : b_1, \ldots, U_n : b_n (n \geq 0)$ is abbreviated as $\Gamma : b$, where $\Gamma = U_1, \ldots, U_n$ and $b = b_1 \bullet \cdots \bullet b_n$. We assume that the empty list $\emptyset$ is labelled as $\emptyset : \epsilon$.

A **TL**-tableau (a $\mathbf{TL_e}$-tableau, resp.) is a rooted tree whose nodes are labelled with **TL**-labelled signed formulas ($\mathbf{TL_e}$-labelled signed formulas, resp.) which is constructed according to the tableau expansion rules in Table 4. The tableau expansion rules and the closure condition are common in **TL** and $\mathbf{TL_e}$. The difference of **TL**-tableau and $\mathbf{TL_e}$-tableau is reduced to that of $\mathcal{L}$ and $\mathcal{L}_{\mathbf{com}}$. The formal definition follows.

Definition 3.3 Let C be a formula. The set of **TL**-*tableaux for* C is defined as follows;

1. The tree with a single node labelled with $FC : \star$ is a tableau for C (called the *initial tableau* for C);

2. Suppose that T is a tableau for C and θ is a branch of T in which a **TL**-labelled signed formula $\alpha : a$ occurs. If T' results from T by adding a node to the end of θ labelled with $\alpha_1 : (a, i)_1$ and another node after that labelled with $\alpha_2 : (a, i)_2$ where i is an index which does not occur on θ, then T' is a tableau for C.

3. Suppose that T is a tableau for C and θ is a branch of T in which a **TL**-labelled signed formula $\beta : b$ occurs. Suppose also that a, c are labels such that $a \bullet b \bullet c$ is **TL**-total. If T' results from T by adding left and right children to the final node of θ which are labelled with $\beta_1 : b \bullet c$ and $\beta_2 : a \bullet b$ respectively, then T' is a tableau for C.

The definition of $\mathbf{TL_e}$-tableaux for C is just the same except that the **TL**-labels are replaced with $\mathbf{TL_e}$-labels.

Definition 3.4 A branch θ of a tableau is *closed* if both $TA : a$ and $FA : b$ occur on θ for some formula A and labels a, b such that $a \bullet b$ is total. A tableau T is *closed* if every branch is closed. A **TL**-*tableau proof* ($\mathbf{TL_e}$-*tableau proof*, resp.) of C is a closed **TL**-tableau ($\mathbf{TL_e}$-tableau, resp.) for C.

Remark 3.1 One way of understanding the labelling in Table 4 is to assign labels to signed sequents of **L'**(or $\mathbf{L'_e}$). Suppose that in an α-inference

$$\frac{\Gamma, \alpha_1, \alpha_2}{\Gamma, \alpha}$$

the lower signed sequent is labelled as $\Gamma : b, \alpha : a$ where $b \bullet a$ is **TL**-total. Try to label the upper sequent so that the sum of its labels becomes total as well. The most natural labelling is as follows;

$$\frac{\Gamma : b, \alpha_1 : (a, i)_1, \alpha_2 : (a, i)_2}{\Gamma : b, \alpha : a}.$$

By this labelling we have $b \bullet (a, i)_1 \bullet (a, i)_2 = b \bullet a$ which is total as required. (The need of index i is explained by Example 2 below.) Similarly, we can label a β-inference as

$$\frac{\Gamma : a, \beta_1 : b \bullet c \quad \beta_2 : a \bullet b, \Delta : c}{\Gamma : a, \beta : b, \Delta : c},$$

where $a \bullet b \bullet c$ is total. That is the intuition behind the labelling of the tableau expansion rules in Table 4. Note that the notion of totality is defined in such a way that the *Cyclic Shift* rule of **L'**, labelled as

$$\frac{U : b_2, \Gamma : b_1}{\Gamma : b_1, U : b_2},$$

preserves the totality of the labels. In this way, the tableau expansion rules are related to the inference rules of **L'** and $\mathbf{L'_e}$ via the labels and the totality condition. This relation is extensively used when proving the soundness (Theorem 4.1) and the termination of every nonredundant tableau construction procedure (Theorem 4.2). See Remark 4.1.

Example 1. Figure 1 is a $\mathbf{TL_e}$-tableau proof of $P \otimes Q \multimap Q \otimes P$. At the last step, β-expansion rule is applied to $FQ \otimes P : (\star, 7)_1$ with respect to $((\star, 7)_2, 9)_1$ and $((\star, 7)_2, 9)_2$. Note

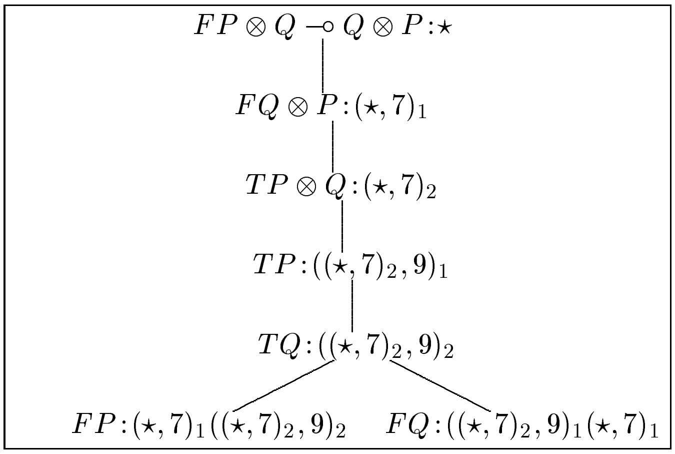


Figure 1: $\mathbf{TL_e}$-tableau proof of $P \otimes Q \multimap Q \otimes P$

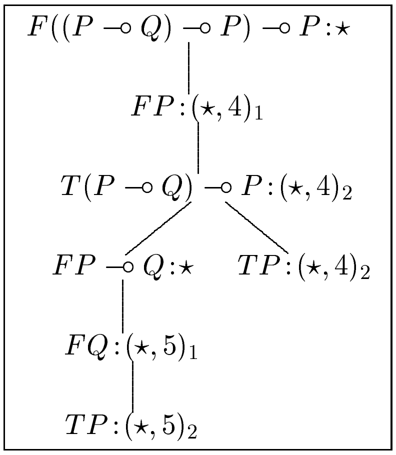


Figure 2

that $((\star,7)_2,9)_1 \bullet (\star,7)_1 \bullet ((\star,7)_2,9)_2$ is $\mathbf{TL_e}$-total, but is not $\mathbf{TL}$-total. The left branch is closed because it contains $\{TP : ((\star,7)_2,9)_1, FP : (\star,7)_1((\star,7)_2,9)_2\}$. The right branch is closed because it contains $\{TQ : ((\star,7)_2,9)_2, FQ : ((\star,7)_2,9)_1(\star,7)_1\}$.

Example 2. Figure 2 is a failure of proving the *Pierce's Law* $((P \multimap Q) \multimap P) \multimap P$. At the second step, β-expansion rule is applied to $T(P \multimap Q) \multimap P : (\star,4)_2$ with respect to ϵ and $(\star,4)_1$. The right branch is closed because it contains $\{FP : (\star,4)_1, TP : (\star,4)_2\}$, while the left branch is open. It should be noted that if we did not employ indices, then the left branch would (incorrectly) become closed, because it contains $\{FP : (\star,4)_1, TP : (\star,5)_2\}$. Therefore, indices cannot be omitted in general.

4 Formal Properties of Labelled Tableau Calculi TL and $\mathbf{TL_e}$

In this section we show (i) the equivalence of $\mathbf{L}$ ($\mathbf{L_e}$, resp.) and $\mathbf{TL}$ ($\mathbf{TL_e}$, resp.), (ii) the completeness of $\mathbf{TL}$ ($\mathbf{TL_e}$, resp.) with respect to the finite simple models (the finite commutative simple models, resp.) (Corollary 4.1, Corollary 4.2), and (iii) every nonredundant tableau construction procedure for $\mathbf{TL}$ and $\mathbf{TL_e}$ terminates in finitely many steps (Theorem 4.2, Theorem 4.4). Since the proofs for $\mathbf{L}$ involve some intricacies due to non-commutativity, first we show the above for $\mathbf{L_e}$ in Subsections 4.1, 4.2 and 4.3. Then, we point out the modifications for $\mathbf{L}$ in Subsection 4.4. Subection 4.5 gives an example of a countermodel which can be generated in our framework.

4.1 Soundness of $\mathbf{TL_e}$

Let $X = \{U_1 : a_1, \ldots, U_n : a_n\}$ be a set of $\mathbf{TL_e}$-labelled signed formulas. We say that X *occurs* on a tableau branch θ if each $U_i : a_i \in X$ occurs on θ. X is said to be *total* if $a_1 \bullet \cdots \bullet a_n$ is total, i.e., $a_1 \bullet \cdots \bullet a_n = \star$. X is said to be *regular* if exactly one U_i is signed F. The set of total sets and the set of regular total sets which occur on θ are denoted by $Total(\theta)$ and $RegTotal(\theta)$, respectively. If X is a regular total set $\{TA_1 : a_1, \ldots, TA_n : a_n, FB : b\}$, we write $Seq(X)$ to denote signed sequent $TA_1, \ldots, TA_n, FB$. It makes sense because X determines $Seq(X)$ uniquely modulo permutation and the derivability of a signed sequent in $\mathbf{L'_e}$ is unchanged by an application of the exchange (permutation) rule.

For a tableau branch θ, the set $Label(\theta)$ is defined by

> $a \in Label(\theta)$ if $a = a_1 \bullet \cdots \bullet a_n$ for some subset $X' = \{U_1 : a_1, \ldots, U_n : a_n\}$ of a total set X on θ.

In particular, $\epsilon \in Label(\theta)$ for every θ. Note that $a, b \in Label(\theta)$ does not necessarily imply $a \bullet b \in Label(\theta)$. Note also that $Label(\theta)$ is finite whenever θ is.

Now we show the soundness of $\mathbf{TL_e}$ with respect to $\mathbf{L_e}$.

Theorem 4.1 *If A is provable in $\mathbf{TL_e}$, then it is also provable in $\mathbf{L_e}$.*

The theorem follows from the following lemmas.

Lemma 4.1 *For every closed branch θ there is some $X \in RegTotal(\theta)$ such that $Seq(X)$ is provable in $\mathbf{L'_e}$.*

Proof. Take X as $\{TA:a, FA:b\}$ which closes θ. ■

Lemma 4.2 *Suppose that branch θ' result from θ by an application of α-expansion rule to $\alpha:a$. If there is an $X' \in RegTotal(\theta')$ such that $Seq(X')$ is provable in $\mathbf{L'_e}$, then there is an $X \in RegTotal(\theta)$ such that $Seq(X)$ is provable in $\mathbf{L'_e}$.*

Proof. If X' also occurs on θ, there is nothing to prove. Otherwise, X' should contain either $\alpha_1 : (a,i)_1$ or $\alpha_2 : (a,i)_2$. Without loss of generality we may assume that X' contains the former. First we show that X' also contains the latter. Since X' is total, X' is of the form $\{\alpha_1 : (a,i)_1, U_1 : b_1, \ldots, U_n : b_n\}$ and $(a,i)_1 \bullet b_1 \bullet \cdots \bullet b_n = \star$. The equation holds only in case $b_j = (a,i)_2$ for some $1 \leq j \leq n$ by the freshness of index i. Hence, X' contains $\alpha_2 : (a,i)_2$. Therefore we may assume that X' is of the form $\{\alpha_1 : (a,i)_1, \alpha_2 : (a,i)_2, \Gamma : b\}$ for some $\Gamma : b$. Let X be $\{\alpha : a, \Gamma : b\}$. It is easy to see $X \in RegTotal(\theta)$ and that $Seq(X)$ is derivable from $Seq(X')$ in $\mathbf{L'_e}$. ■

Lemma 4.3 *Suppose that θ_1 and θ_2 result from θ by an application of β-expansion rule to $\beta : b$ with respect to (a,c). If there is $X_i \in RegTotal(\theta_i)$ such that $Seq(X_i)$ is provable in $\mathbf{L'_e}$ for $i \in \{1,2\}$, then there is an $X \in RegTotal(\theta)$ such that $Seq(X)$ is provable in $\mathbf{L'_e}$.*

Proof. If either X_1 or X_2 occurs on θ, then there is nothing to prove. Otherwise, X_1 should be of the form $\{\Gamma : a', \beta_1 : b \bullet c\}$ and X_2 should be of the form $\{\beta_2 : a \bullet b, \Delta : c'\}$. Let X be $\{\Gamma : a', \beta : b, \Delta : c'\}$. $a \bullet b \bullet c = \star = a' \bullet b \bullet c$ implies that $a = a'$ (by Lemma 3.2). Similarly $c = c'$. Therefore, X is total. It is easy to see that X is regular and that $Seq(X)$ is derivable from $Seq(X_1)$ and $Seq(X_2)$. ■

Proof of Theorem 4.1. Suppose that A has a closed tableau T_f. It means that there is a sequence $T_0, \ldots, T_n$ of tableaux for A, where T_0 is the initial tableau for A, T_{i+1} results from T_i by an application of a tableau expansion rule, and $T_n = T_f$. Using Lemma 4.1, Lemma 4.2 and Lemma 4.3, it can be shown by induction on $0 \leq i \leq n$ that every branch of T_{n-i} contains a regular total set X such that $Seq(X)$ is provable in $\mathbf{L'_e}$. Since the only regular total set of T_0 is $\{FA : \star\}$, we conclude that FA is provable in $\mathbf{L'_e}$, that is, A is provable in $\mathbf{L_e}$ by Proposition 2.1. ■

4.2 Tableau Construction Procedures for $\mathbf{TL_e}$

In this subsection we show an important property of $\mathbf{TL_e}$, that is, every nonredundant tableau construction procedure for $\mathbf{TL_e}$ is terminating. First we define some notions.

Definition 4.1

1. A labelled signed formula $\alpha : a$ is *fulfilled* in branch θ if there are labels a_1 and a_2 such that $a = a_1 \bullet a_2$ and that both $\alpha_1 : a_1$ and $\alpha_2 : a_2$ occur on θ.
2. Let $(a, c) \in \mathcal{L}_{\mathbf{com}} \times \mathcal{L}_{\mathbf{com}}$. A labelled signed formula $\beta : b$ is (a, c)*-fulfilled* in branch θ if either $\beta_1 : b \bullet c$ or $\beta_2 : a \bullet b$ occurs on θ.
3. A labelled signed formula $\beta : b$ is *fulfilled* in branch θ if $\beta : b$ is (a, c)-fulfilled in θ for every $(a, c) \in \mathit{Label}(\theta) \times \mathit{Label}(\theta)$ such that $a \bullet b \bullet c$ is total.
4. A branch θ is *completed* if every labelled signed formula occurring in θ is fulfilled.

By a *tableau sequence* for C we mean a finite or infinite sequence $T_0, T_1, \ldots$ such that T_0 is the initial tableau for C and each T_{i+1} is obtained from T_i by applying a tableau expansion rule.

Definition 4.2 A tableau sequence $T_0, T_1, \ldots$ is said to be *nonredundant* when the following conditions are satisfied;

(i) if T_{i+1} is obtained from T_i by applying a tableau expansion rule to $U : a$ in a branch θ of T_i, then $U : a$ is not fulfilled in θ;

(ii) if T_{i+1} is obtained from T_i by applying β-expansion rule to $\beta : b$ with respect to (a, c) in a branch θ of T_i, then $a, c \in \mathit{Label}(\theta)$.

By a *tableau construction procedure* we mean an algorithm which, given a formula C as input, yields a tableau sequence $T_0, T_1, \ldots$ until either a closed tableau or a tableau which contains an open completed branch is obtained; otherwise it does not terminate. A tableau construction procedure is *nonredundant* if it yields only nonredundant tableau sequences.

It is routine to give a nonredundant tableau construction procedure. If such a procedure terminates for a given input C, it outputs either a closed tableau for C or an open completed branch, and from the latter we can construct a countermodel of C, as shown in the next subsection. In the rest of this subsection we show the following.

Theorem 4.2 *Every nonredundant tableau construction procedure for* $\mathbf{TL_e}$ *terminates in finitely many steps.*

To show this, we need to define some notions. Let $\sqsubset$ be the binary relation on the total sets of labelled signed formulas defined by
- $\{\Gamma : b, \alpha_1 : (a, i)_1, \alpha_2 : (a, i)_2\} \sqsubset \{\Gamma : b, \alpha : a\}$ for $i \in \mathbf{N}$;
- $\{\Gamma : a, \beta_1 : b \bullet c\} \sqsubset \{\Gamma : a, \beta : b, \Delta : c\}$;
- $\{\beta_2 : a \bullet b, \Delta : c\} \sqsubset \{\Gamma : a, \beta : b, \Delta : c\}$.

Let $\sqsubseteq$ be the reflexive transitive closure of $\sqsubset$. For a formula C, we define $\downarrow(C) = \{X \mid X \sqsubseteq \{FC : \star\}\ \}$.

Let $(\cdot)^0$ be the mapping from $\mathcal{L}_{\mathbf{com}}$ to $\mathcal{L}_{\mathbf{com}}$ defined by
- $\epsilon^0 = \epsilon$, $\star^0 = \star$;

- $(a,i)^0_1 = (a^0,0)_1$, $(a,i)^0_2 = (a^0,0)_2$;
- $(ab)^0 = (a^0b^0)$.

The above $(\cdot)^0$ is extended to labelled signed formulas, sets of labelled singed formulas, sets of sets of labelled singed formulas, etc., by $(U\!:\!a)^0 = U\!:\!a^0$ and $W^0 = \{w^0 | w \in W\}$.

To prove Theorem 4.2, it suffices to show that, given C, every nonredundant tableau sequence $T_0, T_1, \ldots$ for C is finite. The following three lemmas suffice to show this.

Lemma 4.4 $(\downarrow(C))^0$ *is finite.*

Proof. Firstly, every descendent sequence $X_0 \sqsupset X_1 \sqsupset X_2, \ldots$ is finite because $X \sqsupset Z$ implies $size(X) > size(Z)$, where $size(X)$ denotes the number of logical connectives $\{\otimes, \multimap\}$ occurring in X. Secondly, $\{Z \mid Z \sqsubset X\}^0$ is finite for any total set X. Thirdly, $X^0 = Y^0$ implies $\{Z \mid Z \sqsubset X\}^0 = \{Z \mid Z \sqsubset Y\}^0$. These facts suffice to prove the lemma. ■

Lemma 4.5 *For every T_i and every branch θ of T_i, $(Total(\theta))^0 \subseteq (\downarrow(C))^0$.*

Proof. It suffices to show $Total(\theta) \subseteq \downarrow(C)$. This is proved by induction on i, using condition (ii) in Definition 4.2. ■

Lemma 4.6 *Suppose that a branch θ' of T_{i+1} is obtained from a branch θ of T_i by an application of a tableau expansion rule. Then $(Total(\theta))^0$ is a proper subset of $(Total(\theta'))^0$.*

Proof. By condition (i) in Definition 4.2. ■

Proof of Theorem 4.2. Lemma 4.4, Lemma 4.5 and Lemma 4.6 together show that in the sequence $T_0, T_1, \ldots$ no branch can be expanded infinitely. In fact, the lengths of the branches are bounded by $|\downarrow(C)^0|$. Since every tableau is finitely branching, the sequence cannot be infinite. ■

Remark 4.1 Note that $(\downarrow(C))^0)$ essentially corresponds to the (completed) bottom-up proof search tree for C in $\mathbf{L_e}$, which is obviously finite. Lemma 4.5 and Lemma 4.6 state that every nonredundant tableau construction procedure simulates the bottom-up proof construction of the sequent calculus, hence the termination of such a procedure is intuitively clear.

4.3 Countermodel Construction from Open Completed Branches

In this subsection we describe how to construct a finite countermodel of C from an open completed branch in a tableau for C. This proves the completeness of $\mathbf{TL_e}$ with respect to the finite commutative simple models.

Theorem 4.3 *If there is a tableau for C which contains an open completed branch θ, then there is a commutative simple model (M, v) in which C is not satisfied. Moreover, if θ is finite, then M is also finite.*

Recall that $Label(\theta)$ is defined as

> $a \in Label(\theta)$ if $a = a_1 \bullet \cdots \bullet a_n$ for some subset $X' = \{U_1 : a_1, \ldots, U_n : a_n\}$ of a total set X on θ.

Here we define a subset $PosLabel(\theta)$ of $Label(\theta)$ as follows;

> $a \in PosLabel(\theta)$ if $a = a_1 \bullet \cdots \bullet a_n$ for some subset $X' = \{TA_1 : a_1, \ldots, TA_n : a_n\}$ of a total set X on θ such that all labelled signed formulas in X' is signed T.

Suppose that an open completed branch θ be given. We define M and a valuation v as follows;

- $M = PosLabel(\theta) \cup \{\surd\}$. Note that $\epsilon \in PosLabel(\theta)$.
- For $a, b \in M$, $a \cdot b = \begin{cases} a \bullet b & \text{if } a \bullet b \in PosLabel(\theta); \\ \surd & \text{otherwise.} \end{cases}$
  In particular, $a \cdot \surd = \surd$ for any $a \in M$.
- For each atomic formula P, $v(P) = \{b \mid TP : b \text{ occurs on } \theta\} \cup \{\surd\}$.

Lemma 4.7 *$(M, \cdot, \epsilon)$ is a commutative monoid.*

Proof. Only nontrivial is the associativity $(a \cdot b) \cdot c = a \cdot (b \cdot c)$. If $a, b, c, a \bullet b \bullet c \in PosLabel(\theta)$, then $a \bullet b, b \bullet c \in PosLabel(\theta)$ by definition. Hence $(a \cdot b) \cdot c = (a \bullet b) \cdot c = (a \bullet b) \bullet c = a \bullet (b \bullet c) = a \cdot (b \cdot c)$. If $a \bullet b \bullet c \notin PosLabel(\theta)$, then $(a \cdot b) \cdot c = \surd = a \cdot (b \cdot c)$. ■

Lemma 4.8 *For every formula A,*
(i) if $TA : b$ occurs on θ, then $b \in v(A)$;
(ii) if $FA : c$ occurs on θ, $b \in PosLabel(\theta)$ and $b \bullet c$ is total, then $b \notin v(A)$[5];
(iii) $\surd \in v(A)$.

Proof. (iii) can be easily shown. Here we prove (i) and (ii) by induction on the complexity of A.
(Case 1) A is an atomic formula. (i) holds by definition. As for (ii), since θ is open, $TA : b$ never occurs on θ. Hence $b \notin v(A)$.
(Case 2) A is of the form $B \otimes D$. To show (i), assume that $TB \otimes D : b$ occurs on θ. Then $TB : b_1$ and $TD : b_2$ occurs on θ for some b_1, b_2 such that $b_1 \bullet b_2 = b$. By the induction hypothesis $b_1 \in v(B)$ and $b_2 \in v(D)$. Hence $b = b_1 \bullet b_2 \in v(B \otimes D)$.

To show (ii), assume that $FB \otimes D : c$ occurs on θ, $b \in PosLabel(\theta)$ and $b \bullet c$ is total. It suffices to show that if $b = d \cdot e$ for some $d, e \in M$, then either $d \notin v(B)$ or $e \notin v(D)$. Suppose that $b = d \cdot e$. Then $d \neq \surd, e \neq \surd$, hence $d, e \in PosLabel(\theta)$. Since $d \bullet e \bullet c = e \bullet c \bullet d$ is total, either $FC : c \bullet d$ or $FB : e \bullet c$ occurs on θ. Therefore, by the induction hypothesis, either $d \notin v(B)$ or $e \notin v(D)$.
(Case 3) A is of the form $B \multimap D$. To show (i), assume that $FB \multimap D : b$ occurs on θ. It suffices to show that for any $d \in v(B)$, $d \cdot b \in v(D)$. If $d \cdot b = \surd$, then by (iii), $\surd \in v(D)$. Otherwise, $d, b \in PosLabel(\theta)$ and $d \cdot b \equiv d \bullet b \in PosLabel(\theta)$. It means that for some $e \in Label(\theta)$, $d \bullet b \bullet e$ is total. Therefore, either $FB : b \bullet e$ or $TD : d \bullet b$ occurs on θ. By the induction hypothesis (ii) the former would imply $d \notin v(B)$, hence is impossible. From the latter and the induction hypothesis (i) it follows that $d \bullet b \in v(D)$.

To show (ii), assume that $FB \multimap D : c$ occurs on θ, $b \in PosLabel(\theta)$ and $b \bullet c$ is total. Then it follows that $TB : c_1$ and $FD : c_2$ also occur on θ for some $c_1 \in PosLabel(\theta)$ and $c_2 \in Label(\theta)$ such that $c_1 \bullet c_2 = c$. By the induction hypothesis (ii) we have $c_1 \in v(B)$, and

[5] Note that c does not have to be in $PosLabel(\theta)$.

by the induction hypothesis (i) we have $c_1 \bullet b \notin v(D)$. This proves that $b \notin v(B \multimap D)$.
(Case 4) A is of the form $D \circ\!\!- B$. Similar to (Case 3). ■

Proof of Theorem 4.3. From an open completed branch θ we can construct a commutative simple model (M, v) as above. It is obvious from the construction that (M, v) is finite whenever θ is. Since $\epsilon \in PosLabel(\theta)$, $FC : \star$ occurs on θ and $\epsilon \bullet \star = \star$ is total, it holds by Lemma 4.8 that $\epsilon \notin v(C)$. ■

Corollary 4.1 *For every formula A, the following are equivalent;*
(1) A is provable in $\mathbf{L_e}$.
(2) A is satisfied in every commutative simple model.
(3) A is satisfied in every finite commutative simple model.
(4) A is provable in $\mathbf{TL_e}$.

Proof. (1) implies (2) by the standard soundness argument. (2) implies (3) trivially. (4) implies (1) by Theorem 4.1. To show that (3) implies (4), suppose that A is not provable in $\mathbf{TL_e}$. Take an arbitrary nonredundant tableau construction procedure. It yields a tableau for A which contains a finite open completed branch θ (by Theorem 4.2), from which we can construct a finite commutative simple model in which A is not satisfied (by Theorem 4.3). ■

4.4 Modifications for TL

To adapt the proofs in the former subsections to **TL**, we need to modify some notions. Let $\approx$ be the equivalence relation on the finite lists of **TL**-labelled signed formulas defined by

$$[U_1 : a_1, \ldots, U_n : a_n] \approx [U_i : a_i, \ldots, U_n : a_n, U_1 : a_i, \ldots, U_{i-1} : a_{i-1}] \text{ for } 1 \leq i \leq n.$$

A *cycle* is an equivalence class of finite lists of **TL**-labelled signed formulas induced by $\approx$.

A cycle $X = [U_1 : a_1, \ldots, U_n : a_n]$ is said to be *total* if $a_1 \bullet \cdots \bullet a_n$ is total. X is said to be *regular* if exactly one U_i is signed F. If $X = [TA_1 : a_1, \ldots, TA_n : a_n, FB : b]$ is a regular total cycle, then we write $Seq(X)$ to denote signed sequent $TA_1, \ldots, TA_n, FB$. It makes sense because X determines $Seq(X)$ uniquely modulo cyclic shift and the derivability of a signed sequent in **L'** is unchanged by an application of the cyclic shift rule.

For a **TL**-tableau branch θ, the set $Label(\theta)$ is defined by

> $a \in Label(\theta)$ if there is a total cycle $X = [U_1 : a_1, \ldots, U_n : a_n]$ on θ and $a = a_j \bullet \cdots \bullet a_k$ for some $0 \leq j \leq k \leq n$.

According to this change, all the proofs in Subsections 4.1, 4.2 and 4.3 can be modified for **L**. Therefore, we have the following.

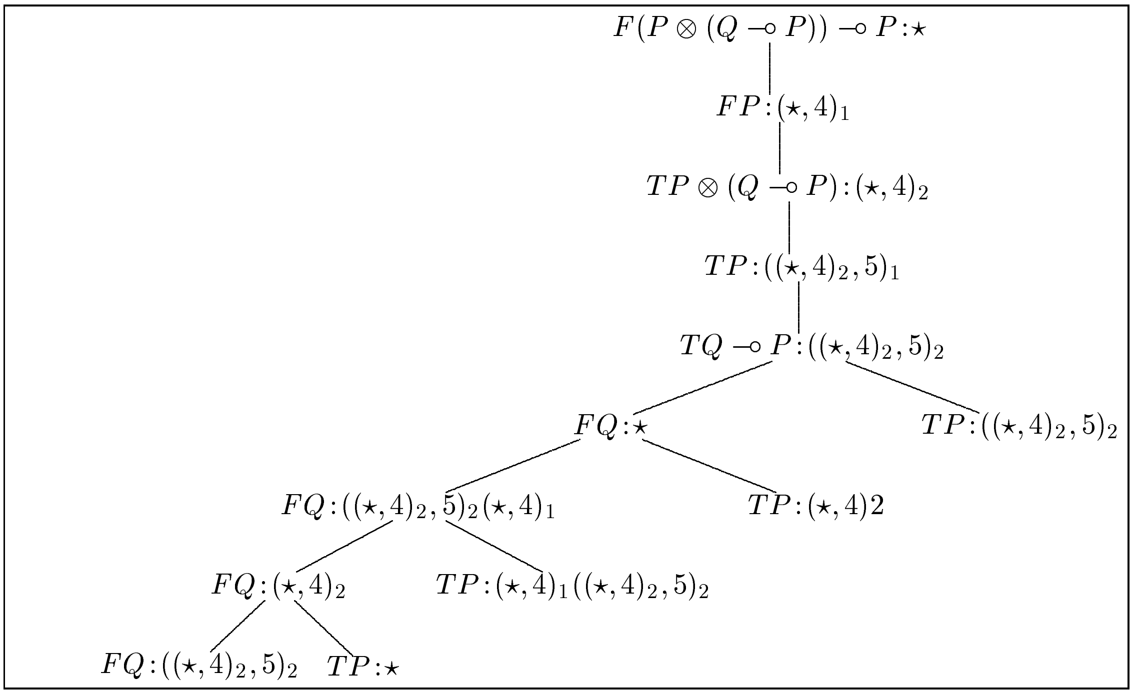
Theorem 4.4 *Every nonredundant tableau construction procedure for* **TL** *terminates in finitely many steps.*

Corollary 4.2 *For every formula A, the following are equivalent;*
(1) A is provable in **L**.
(2) A is satisfied in every simple model.
(3) A is satisfied in every finite simple model.
(4) A is provable in **TL**.

4.5 An Example of a Countermodel

Let us fix a nonredundant tableau construction procedure σ. Given a formula C, we can effectively obtain either a proof or a countermodel of C as follows;
(1) Apply σ to C. Then σ terminates in finitely many steps.
(2) If σ outputs a closed tableau T, then T is a tableau proof of C.
(3) Otherwise, σ outputs a tableau T which has a finite open completed branch θ.
(4) From θ, a countermodel of C is obtained by the construction described in Subsection 4.3.
Let us give an example for the last step (4).

Example. The following is a $\mathbf{TL_e}$-tableau for $(P \otimes (Q \multimap P)) \multimap P$, where the leftmost branch θ is open and completed;



For readability, we omit indices and denote $(\star, 4)_j$ by $\star_j$ and $((\star, 4)_j, 5)_k$ by $\star_{jk}$ $((j, k \in \{1, 2\})$. Then, we have $PosLabel(\theta) = \{\epsilon, \star_2, \star_{21}, \star_{22}\}$. By the construction described in Subsection 4.3, we obtain the following commutative simple model (M, v);

- $M = \{\epsilon, \star_2, \star_{21}, \star_{22}, \surd\}$ (monoid operation $a \cdot b$ is defined as in Subsection 4.3);
- $v(P) = \{\star_{21}, \surd\}$; $v(Q) = \{\surd\}$.

In this model, $v(Q \multimap P) = \{\epsilon, \star_2, \star_{21}, \star_{22}, \surd\}$, $v(P \otimes (Q \multimap P)) = \{\star_{21}, \star_2, \surd\}$, and $v((P \otimes (Q \multimap P)) \multimap P) = \{\surd\}$. Therefore we see that (M, v) is a countermodel of $(P \otimes (Q \multimap P)) \multimap P$, as expected.

5 Notes on Extensions

In this section we discuss possible extensions of $\mathbf{TL}$ and $\mathbf{TL_e}$.

As mentioned before, all results of this paper can be easily extended to the original Lambek calculus in [Lam58], i.e., **L** with the restriction that the antecedent of each sequent should not be empty.

It is surely difficult to extend our framework with additional logical connectives such as additive (boolean) connectives. But see [OT98] for some attempts, where the model generation for a certain fragment (including the additive disjunction) of intuitionistic linear logic is described. [OT98] also describes the model generation method for full linear logic in the framework of the sequent calculus based on the phase semantics.

As for extensions with structural rules, we can obtain similar results for the extensions of **L** and $\mathbf{L_e}$ with the weakening (monotonicity) rule

$$\frac{\Gamma, \Delta \vdash C}{\Gamma, A, \Delta \vdash C}\ \mathbf{w}$$

if we admit a more complicated semantics than the simple semantics. Let us denote **L** with **w** by $\mathbf{L_w}$, and $\mathbf{L_e}$ with **w** by $\mathbf{L_{ew}}$. In the sequel, we just mention the results without proofs.

By a *partially ordered monoid model (PO-monoid model)*, we mean a triple $(M, \leq, v)$ where $\leq$ is a partial order on M which satisfies $\forall x_1, x_2, y \in M(x_1 \cdot x_2 \leq x_1 \cdot y \cdot x_2)$, and valuation v assigns a $\leq$-upward closed subset of M to each atomic formula. $v(A \multimap B)$ and $v(B \multimap A)$ are defined just as before, and we define $v(A \otimes B) = \{z | \exists x \in v(A), y \in v(B)(x \cdot y \leq z)\}$. Then, every formula is interpreted by a $\leq$-upward closed subset of M.

Let $\preceq$ be the smallest preordering on $\mathcal{L}$ which satisfies $\forall a_1, a_2, b \in \mathcal{L}(a_1 \bullet a_2 \preceq a_1 \bullet b \bullet a_2)$. Preordering $\preceq_{\mathbf{com}}$ on $\mathcal{L}_{\mathbf{com}}$ is similarly defined. An $\mathbf{TL_w}$-label is an element of $\mathcal{L}$, and an $\mathbf{TL_{ew}}$-label is an element of $\mathcal{L}_{\mathbf{com}}$. A ($\mathbf{TL_w}$- or $\mathbf{TL_{ew}}$-) label a is said to be *total* if $\exists b_1, b_2(b_1 \bullet b_2 = \star$ and $a \preceq b_2 \bullet b_1)$. $\mathbf{TL_w}$-tableaux and $\mathbf{TL_{ew}}$-tableaux are defined just in the same way as **TL**-tableaux and $\mathbf{TL_e}$-tableaux. Then we can show that the following are equivalent;
(1) A is provable in $\mathbf{L_w}$ ($\mathbf{L_{ew}}$, resp.);
(2) A is satisfied in every PO-monoid model (commutative PO-monoid model, resp.);
(3) A is satisfied in every finite PO-monoid model (commutative PO-monoid model, resp.);
(4) A is provable in $\mathbf{TL_w}$ ($\mathbf{TL_{ew}}$, resp.).
Moreover, we can show as before that
• every nonredundant tableau construction procedure for $\mathbf{TL_w}$ and $\mathbf{TL_{ew}}$ terminates in finitely many steps.

It may be expected that similar results are obtained for extensions with the contraction rule and the expansion rule. For the present, however, these extensions remain open problems.

6 Conclusion

In this paper we have proposed a new framework of the labelled tableau method for substructural logics. It has the following advantage over the existing tableau calculi for substructural logics such as [MB79] and [MMB95]: it yields not only proofs for theorems but also countermodels for non-theorems. The resulting countermodels are finite and sufficiently simple as required for practical applications. From the theoretical point of view, we have shown the finite model property for **L** and $\mathbf{L_e}$ with respect to the simple semantics and the commutative simple semantics, respectively. It may be regarded as an refinement of the completeness results given by [Bus86] (See Remark 2.1).

As mentioned in the introduction, there is another labelled tableau framework for substructural logics, i.e., $\mathbf{LKE}_{\mathcal{S}}$ of Gabbay and D'Agostino ([DG94], [Gab96]). Here we compare our framework with $\mathbf{LKE}_{\mathcal{S}}$. First of all, it should be mentioned that their motivation is completely different from ours. $\mathbf{LKE}_{\mathcal{S}}$ is not intended as a system oriented to a special purpose such as model generation, but just as a *uniform* and *transparent* system for a wide range of substructural logics. We would like to remark the following points.

(1) We use two fixed sets $\mathcal{L}$ and $\mathcal{L}_{\mathbf{com}}$ of labels, and formulas are labelled with elements of these sets directly. On the other hand, the use of labels in $\mathbf{LKE}_{\mathcal{S}}$ is more flexible than ours in that each formula is labelled with an element of a labelling algebra which identifies a class of *information frames*. Information frames (or *unital quantales*, see [Yet90], [Ono94]) have more complicated algebraic structures than our $\mathcal{L}$ and $\mathcal{L}_{\mathbf{com}}$ which are just a monoid and a commutative monoid. It is this flexibility that allows $\mathbf{LKE}_{\mathcal{S}}$ to cover a considerably wide range of substructural logics. However, as a consequence of this flexibility, countermodels which may be obtained in their framework are inevitably complicated ones based on information frames (see Proposition 3 of [DG94]). Hence, for the purpose of generating countermodels for $\mathbf{L}$ and $\mathbf{L_e}$, our tableau calculi works much better than $\mathbf{LKE}_{\mathcal{S}}$.

(2) Our tableau calculi retain a tight connection with the sequent calculus via the totality condition on labels (see Remark 3.1 and Remark 4.1). This connection makes it very easy to give a decision procedure for $\mathbf{L}$ and $\mathbf{L_e}$; indeed, every nonredundant tableau construction procedure is terminating. On the other hand, $\mathbf{LKE}_{\mathcal{S}}$ loses such a connection, and it might be a nontrivial task to give such a decision procedure for $\mathbf{L}$ and $\mathbf{L_e}$ ([DG94] announces that the decision problems will be discussed in Part II of their paper).

(3) $\mathbf{LKE}_{\mathcal{S}}$ puts the "cut" rule (the PB-rule) into the tableau expansion rules. The idea stems from the precursor of $\mathbf{LKE}_{\mathcal{S}}$, called $\mathbf{KE}$ ([DM94]), which is an analytic deduction system for classical logic. The PB-rule allows $\mathbf{KE}$ to produce essentially shorter proofs than the standard tableau method for classical logic. One might expect that a similar statement holds for $\mathbf{L}$ and $\mathbf{L_e}$, i.e., that $\mathbf{LKE}_{\mathcal{S}}$ produces essentially shorter proofs than our tableau calculi which lack the "cut" rule. However, it is not the case, because in substructural logics without the contraction rule such as $\mathbf{L}$ and $\mathbf{L_e}$, shortest proofs are always cut-free.

To sum up, (i) $\mathbf{LKE}_{\mathcal{S}}$ covers much wider class of substructural logics than our framework; (ii) as for proof construction for $\mathbf{L}$ and $\mathbf{L_e}$, our tableau calculi produce tableau proofs essentially of the same size as $\mathbf{LKE}_{\mathcal{S}}$; (iii) as for model generation for $\mathbf{L}$ and $\mathbf{L_e}$, our tableau calculi generate much simpler countermodels than $\mathbf{LKE}_{\mathcal{S}}$.

Our labels have two uses; one is to generate simple models, and the other is to relate tableaux to proof search trees of the sequent calculus. The latter use seems to be original to our work, and we hope that our technique will be found useful in other applications of the LDS methodology.

Acknowledgments
The author would like to thank Prof. Maarten de Rijke and Prof. Mitsuhiro Okada for their helpful advice.

References

[Bus86] W. Buszkowski. Completeness results for Lambek syntactic calculus. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 32:13–28, 1986.

[Bus96] W. Buszkowski. The finite model property for BCI and related systems. *Studia Logica*, 57:303–323, 1996.

[Bus97] W. Buszkowski. Mathematical linguistics and proof theory. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, chapter 12. Elsevier Science B. V., 1997.

[BY96] F. Bry and A. Yahya. Minimal model generation with positive unit resolution tableaux. In *Proc. 5th Workshop on Theorem Proving with Analytic Tableaux and Related Methods*. Springer-Verlag LNCS, 1996.

[DG94] M. D'Agostino and D. M. Gabbay. A generalization of analytic deduction via labelled deductive systems. Part I: basic substructural logics. *Journal of Automated Reasoning*, 13:243–281, 1994.

[DM94] M. D'Agostino and M. Mondadori. The taming of the cut. *Journal of Logic and Computation*, 4(3):285–319, 1994.

[Fit83] M. Fitting. *Proof Methods for Modal and Intuitionistic Logics*. Reidel, Dordrecht, 1983.

[Fit87] M. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, 1987.

[Gab96] Dov. M. Gabbay. *Labelled Deductive Systems*. Oxford: Oxford University Press, 1996.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Hin88] J. Hintikka. Model minimization – an alternative to circumscription. *Journal of Automated Reasoning*, 4:1–13, 1988.

[Kur94] Natasha Kurtonina. The Lambek calculus: Relational semantics and the method of labelling. ILLC research report and technical notes series LP–94–05, Institute for Logic, Language and Computation, University of Amsterdam, 1994.

[Lam58] J. Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65(3):154–170, 1958.

[MB79] M. A. McRobbie and N. D. Belnap Jr. Relevant analytic tableaux. *Studia Logica*, 38:187–200, 1979.

[MMB95] R. K. Meyer, M. A. McRobbie, and N. Belnap. Linear analytic tableaux. In P. Baumgartner, R. Hahnle, and J. Posegga, editors, *Theorem proving with analytic tableaux and related methods: Proceedings TABLEAUX'95*, pages 278–293. Springer-Verlag LNCS 918, 1995.

[Ono94] Hiroakira Ono. Semantics for substructural logics. In K. Došen and P. Schröder-Heister, editors, *Substructural logics*, pages 259–291. Oxford University Press, 1994.

[OT] Mitsuhiro Okada and Kazushige Terui. The finite model property for various fragments of intuitionsitic linear logic. *Journal of Symbolic Logic*. to appear.

[OT98] Mitsuhiro Okada and Kazushige Terui. Completeness proofs for linear logic based on the proof search method (preliminary report). In J. Garrigue, editor, *Type theory and its applications to computer systems*, pages 57–75. Research Institute for Mathematical Sciences, Kyoto University, 1998. Available by ftp at ftp://abelard.flet.mita.keio.ac.jp/pub/Papers/Terui.

[Pan94] Nikolai Pankrat'ev. On the completeness of the Lambek calculus with respect to relativized relational semantics. *Journal of Logic, Language, and Information*, 3:233–246, 1994.

[Pen94] M. Pentus. Language completeness of the Lambek calculus. In *proc. of LICS 94*, pages 487–496. IEEE, 1994.

[Smu68] R. Smullyan. *First-Order Logic*. Springer-Verlag, 1968.

[Tro92] Anne S. Troelstra. *Lectures on Linear Logic.* CSLI Lecture Notes 29, Center for the Study of Language and Information, Stanford, California, 1992.

[Yet90] David N. Yetter. Quantales and (noncommutative) linear logic. *Journal of Symbolic Logic*, 55(1):41–64, March 1990.