CHAPTER 6

Automated Proof Systems for Classical Propositional Logic

1 Gentzen Style Proof System RS

Hilbert style systems are easy to define and admit different proofs of the Completeness Theorem but they are difficult to use. By humans, not mentioning computers. Their emphasis is on logical axioms, keeping the rules of inference, with obligatory Modus Ponens, at a minimum.

Gentzen style proof systems reverse this situation by emphasizing the importance of inference rules, reducing the role of logical axioms to an absolute minimum. They may be less intuitive then the Hilbert-style systems, but they allow us to define effective automatic procedures for proof search, what was impossible in a case of the Hilbert style systems. For this reason they are also called automated proof systems. They serve as formal models of computing systems that automate the reasoning process. Building computing systems means providing an algorithmic description to a formal proof system so that it can be implemented on a computer to prove theorems in an efficient manner.

The first proof systems of this style was invented by G. Gentzen in 1934, hence the name. His proof systems for classical and intuitionistic predicate logics introduced special expressions built of formulas called sequents. Hence the Gentzen style systems using sequents as basic expressions are often called sequent systems, or Gentzen sequent systems, or simply Gentzen formalizations.

We present here (section 4) two Gentzen systems GL and G for classical propositional logic and prove their completeness. We also present a propositional version of Gentzen original system LK and discuss a proof of Gentzen Hauptssatz for it. Hauptssatz is literally rendered as the main theorem and is known as cut-elimination theorem. We prove the equivalency of the cut-free propositional LK and the complete system G. The Gentzen original formalization for intuitionistic propositional logic LI is discussed and presented in chapter ???. The classical and intuitionistic predicate versions are discussed in chapter ???.

The other historically important automated proof system is due to Rasiowa and Sikorski (1960). Their proof systems for classical propositional and predi-
cate logic use as basic expressions sequences of formulas, less complicated then Gentzen sequents. As they were inspired Gentzen systems we call them, as we call many others similarly inspired, Gentzen style proof system, or Gentzen style formalization. The Rasiowa and Sikorski proof system is simpler and easier to understand then the Gentzen sequent systems. Hence their system RS is the first to be presented here in section 1.

Historical importance and lasting influence of Rasiowa and Sikorski work lays in the fact that they were first to use the proof searching capacity of their proof system to define a constructive method of proving the completeness theorem for both propositional and predicate classical logic. We introduce and explain in detail their method and use it prove the completeness of the RS in section 2.2. We also introduce and discuss two other RS style system RS1 and RS2 in in section 3. We also generalize the RS completeness proof method to the Gentzen sequent systems and prove the completeness of GL and G systems in section 4.1. The completeness proof for proof system RSQ for classical predicate logic is presented in chapter ??.

2 Proof System RS

We present here a propositional version of the original Rasiowa and Sikorski (1960) Gentzen style proof system for classical logic. We call it RS system for Rasiowa-Sikorski. The RS system extends naturally to predicate logic QRS system which is presented in chapter ???. Both systems admit a constructive proof of Completeness Theorem. We prove completeness of RS in section 2.2. We define components and semantics of the system RS as follows.

Components of the proof system RS

Language \( \mathcal{L} \)

Let \( \mathcal{F} \) denote a set of formulas of \( \mathcal{L} = \mathcal{L}(\neg,\Rightarrow,\cup,\cap) \). The rules of inference of our system RS operate on finite sequences of formulas, i.e. elements of \( \mathcal{F}^* \), unlike on plain formulas \( \mathcal{F} \) in Hilbert style formalizations.

Expressions \( \mathcal{E} \)

We adopt as the set of expressions \( \mathcal{E} \) of RS the set \( \mathcal{F}^* \), i.e. \( \mathcal{E} = \mathcal{F}^* \). We will denote the expressions of RS, i.e. the finite sequences of formulas by \( \Gamma, \Delta, \Sigma \), with indices if necessary.

Semantic Link
The intuitive meaning of a sequence $\Gamma \in \mathcal{F}^*$ is that the truth assignment $v$ makes it true if and only if it makes the formula of the form of the disjunction of all formulas of $\Gamma$ true. As we know, the disjunction in classical logic is associative and commutative, i.e., for any formulas $A, B, C \in \mathcal{F}$, the formulas $(A \cup (B \cup C))$, $((A \cup B) \cup C)$, $(A \cup (C \cup B))$, $((B \cup A) \cup C)$, $(C \cup (B \cup A))$, $(C \cup (A \cup B))$, etc... are logically equivalent. In particular we write

$$\delta_{\{A,B,C\}} = A \cup B \cup C$$

to denote any disjunction of formulas $A, B, C$.

In a general case, for any sequence $\Gamma \in \mathcal{F}^*$, if $\Gamma$ is of a form

$$A_1, A_2, ..., A_n$$

(1)

then by $\delta_\Gamma$ we will understand any disjunction of all formulas of $\Gamma$ and we write it

$$\delta_\Gamma = A_1 \cup A_2 \cup ... \cup A_n.$$

**Formal Semantics for RS**

Let $v : VAR \rightarrow \{T, F\}$ be a truth assignment, $v^*$ its extension to the set of formulas $\mathcal{F}$. We formally extend $v$ to the set $\mathcal{E}$ of expressions of RS, i.e., to the set $\mathcal{F}^*$ of all finite sequences of $\mathcal{F}$ as follows. For any sequence $\Gamma \in \mathcal{F}^*$, if $\Gamma$ is the sequence (1), then we define:

$$v^*(\Gamma) = v^*(\delta_\Gamma) = v^*(A_1) \cup v^*(A_2) \cup ... \cup v^*(A_n).$$

(2)

**Model**

A sequence $\Gamma$ is said to be **satisfiable** if there is a truth assignment $v : VAR \rightarrow \{T, F\}$ such that $v^*(\Gamma) = T$. Such a truth assignment is called a **model** for $\Gamma$. We denote it as

$$v \models \Gamma.$$

(3)

**Counter-Model**

A sequence $\Gamma$ is said to be **falsifiable** if there is a truth assignment $v$, such that $v^*(\Gamma) = F$. Such a truth assignment is called a **counter-model** for $\Gamma$. We write it symbolically as

$$v \not\models \Gamma.$$

(4)

**Tautology**

The sequence $\Gamma$ is said to be a **tautology** if $v^*(\Gamma) = T$ for all truth assignments $v : VAR \rightarrow \{T, F\}$. We write it as

$$\models \Gamma.$$

(5)
Exercise 1

Let \( \Gamma \) be a sequence \( a, (b \cap a), \neg b, (b \Rightarrow a) \).

1. Show that the truth assignment \( v : VAR \rightarrow \{ T, F \} \), such that \( v(a) = F \) and \( v(b) = T \) falsifies \( \Gamma \), i.e. \( v \not\models \Gamma \).

2. Let \( \Gamma \) be a sequence \( a, (\neg b \cap a), \neg b, (a \cup b) \) and let \( v \) be a truth assignment for which \( v(a) = T \). Prove that \( v \models \Gamma \).

3. Let \( \Gamma \) be a sequence \( a, (\neg b \cap a), \neg b, (a \cup b) \). Prove that \( \models \Gamma \).

Solution

1. \( \Gamma \) is the sequence \( a, (b \cap a), \neg b, (b \Rightarrow a) \). We evaluate \( v^*(\Gamma) = v^*(\delta_\Gamma) = v^*(a) \cup v^*(b \cap a) \cup v^*(\neg b) \cup v^*(b \Rightarrow a) = F \cup (F \cap T) \cup F \cup (T \Rightarrow F) = F \cup F \cup F \cup F = F \). By (4) we proved \( v \not\models \Gamma \).

2. Let \( \Gamma \) be a sequence \( a, (\neg b \cap a), \neg b, (a \cup b) \). We evaluate \( v^*(\Gamma) = v^*(\delta_\Gamma) = v^*(a) \cup v^*(\neg b \cap a) \cup v^*(\neg b) \cup v^*(a \cup b) = T \cup v^*(\neg b \cap a) \cup v^*(\neg b) \cup v^*(a \cup b) = T \). By (3) we proved \( v \models \Gamma \).

3. Assume now that \( \Gamma \) is falsifiable i.e. that we have a truth assignment \( v \) for which \( v^*(\Gamma) = v^*(\delta_\Gamma) = v^*(a) \cup v^*(\neg b \cap a) \cup v^*(\neg b) \cup v^*(a \cup b) = F \). This is possible only when (in shorthand notation)

\[
a \cup (\neg b \cap a) \cup \neg b \cup a \cup b = F,
\]

what is impossible as \((\neg b \cup b) = T\) for all \( v \). This contradiction proves that \( \Gamma \) that (5) holds and \( \Gamma \) is a tautology.

In order to define the axioms \( LA \) and the set of rules of inference of \( RS \) we need to introduce some definitions.

**Literals**

We form a special subset \( F' \subseteq F \) of formulas, called a set of all literals, which is defined as follows.

\[
LT = VAR \cup \{ \neg a : a \in VAR \},
\]

(6)

The variables are called positive literals and the elements of the second set of the above union (6) are called negative literals. I.e propositional variables are called positive literals and the negation of a variable is called a negative literal, a variable or a negation of propositional variable is called a literal.

**Indecomposable formulas and sequences**

Literals are also called the indecomposable formulas. Now we form finite sequences out of formulas (and, as a special case, out of literals). We need to
distinguish the sequences formed out of literals from the sequences formed out of other formulas, so we adopt the following notation.

We denote by
\[ \Gamma', \Delta', \Sigma' \ldots \] with indices if necessary, elements of \( LT^* \subseteq F^* \), i.e. \( \Gamma', \Delta', \Sigma' \) are finite sequences (empty included) formed out of literals. We call them indecomposable sequences.

We denote by
\[ \Gamma, \Delta, \Sigma \ldots \] with indices if necessary, the elements of \( F^* \), i.e. \( \Gamma', \Delta', \Sigma' \) are finite sequences (empty included) formed out of elements of \( F \).

**Logical Axioms** LA

As the logical axiom of RS we adopt any sequence of literals which contains any propositional variable and its negation, i.e any sequence of the form

\[ \Gamma'_1, a, \Gamma'_2, \neg a, \Gamma'_3 \] (9) or of the form

\[ \Gamma'_1, \neg a, \Gamma'_2, a, \Gamma'_3 \] (10)

for any variable \( a \in \text{VAR} \) and any sequences \( \Gamma'_1, \Gamma'_2, \Gamma'_3 \in LT^* \) of literals.

**Semantic Link**

Consider axiom (9). Directly from the extension of the notion of tautology to \( \text{bf RS} \) (5), we have that for any truth assignments \( v : \text{VAR} \rightarrow \{T,F\} \),

\[ v^*(\Gamma'_1, \neg a, \Gamma'_2, a, \Gamma'_3) = v^*(\Gamma'_1) \cup v^*(\neg a) \cup v^*(a) \cup v^*(\Gamma'_2, \Gamma'_3) = v^*(\Gamma'_1) \cup T \cup v^*(\Gamma'_2, \Gamma'_3) = T \]

The same applies to the axiom (10) We have thus proved the following.

**Fact 1**

*Logical axioms of RS are tautologies.*

**Rules of inference** \( R \)

All rules of inference of RS are of the form

\[
\begin{align*}
\frac{\Gamma_1}{\Gamma} \quad \text{or} \quad \frac{\Gamma_1; \Gamma_2}{\Gamma},
\end{align*}
\]

where \( \Gamma_1, \Gamma_2, \Gamma \in \mathcal{F}^* \), i.e. \( \Gamma_1, \Gamma_2, \Gamma \) are any finite sequences of formulas. The sequences \( \Gamma_1, \Gamma_2 \) are called *premises* and \( \Gamma \) is called a *conclusion* of the
Each rule of inference of \( RS \) introduces a new logical connective, or a negation of a logical connective. We denote a rule of inference that introduces the logical connective \( \circ \) in the conclusion sequent \( \Gamma \) by \( (\circ) \). The notation \( (\neg \circ) \) means that the negation of the logical connective \( \circ \) is introduced in the conclusion sequence \( \Gamma \). As our language contains the connectives: \( \cap, \cup, \Rightarrow \) and \( \neg \), so we are going to define the following seven inference rules:

\[
(\cup), (\neg \cup), (\cap), (\neg \cap), (\Rightarrow), (\neg \Rightarrow), \text{ and } (\neg \neg).
\]  

We define formally the inference rules of \( RS \) as follows.

**Disjunction rules**

\[
\frac{\Gamma', A, B, \Delta}{\Gamma', (A \cup B), \Delta} \quad \frac{(\neg \cup)}{\Gamma', \neg A, \Delta : \Gamma', \neg B, \Delta}
\]

\[
\frac{\Gamma', A, \Delta}{\Gamma', (A \cap B), \Delta} \quad \frac{(\neg \cap)}{\Gamma', \neg A \neg B, \Delta}
\]

**Conjunction rules**

\[
\frac{\Gamma', A, \Delta}{\Gamma', B, \Delta} \quad \frac{\Gamma', B, \Delta}{\Gamma', (A \cap B), \Delta}
\]

**Implication rules**

\[
\frac{\Gamma', \neg A, B, \Delta}{\Gamma', (A \Rightarrow B), \Delta} \quad \frac{(\neg \Rightarrow)}{\Gamma', A, \Delta : \Gamma', \neg B, \Delta}
\]

**Negation rule**

\[
\frac{\Gamma', A, \Delta}{\Gamma', \neg A, \Delta}
\]

where \( \Gamma' \in LT^*, \Delta \in F^*, A, B \in F \).

**The Proof System RS**

Formally we define the proof system \( RS \) as follows.

\[
RS = (L\{\neg, \Rightarrow, \cup, \cap\}, E, LA, R),
\]

where \( E = \{\Gamma : \Gamma \in F^*\} \), \( LA \) contains logical axioms of the system defined by the schemas (9) and (10), \( R \) is the set of rules of inference:

\[
R = \{(\cup), (\neg \cup), (\cap), (\neg \cap), (\Rightarrow), (\neg \Rightarrow), (\neg \neg)\}
\]
defined by (11).

By a formal proof of a sequence $\Gamma$ in the proof system $\textsc{RS}$ we understand any sequence

$$\Gamma_1, \Gamma_2, \ldots, \Gamma_n$$

(13)

of sequences of formulas (elements of $F^*$, such that $\Gamma_1 \in \textsc{LA}$, $\Gamma_n = \Gamma$,

and for all $i$ ($1 \leq i \leq n$) $\Gamma_i \in \textsc{AL}$, or $\Gamma_i$ is a conclusion of one of the inference rules of $\textsc{RS}$ with all its premisses placed in the sequence $\Gamma_1 \Gamma_2 \ldots \Gamma_{i-1}$.

As the proof system under consideration is fixed, we will write, as usual,

$$\vdash \Gamma$$

instead of $\vdash_{\textsc{RS}} \Gamma$ to denote that $\Gamma$ has a formal proof in $\textsc{RS}$.

As the proofs in $\textsc{RS}$ are sequences (definition of the formal proof) of sequences of formulas (definition of $\textsc{RS}$) we will not use "," to separate the steps of the proof, and write the formal proof as $\Gamma_1; \Gamma_2; \ldots; \Gamma_n$.

We write, however, the formal proofs in $\textsc{RS}$ in a form of trees rather than in a form of sequences, i.e., in a form of a tree, where leafs of the tree are axioms, nodes are sequences such that each sequence on the tree follows from the ones immediately preceding it by one of the rules. The root is a theorem. We picture, and write our tree-proofs with the node on the top, and leafs on the very bottom, instead of more common way, where the leafs are on the top and root is on the bottom of the tree. We adopt hence the following definition.

**Definition 1 (Proof Tree)**

By a proof tree, or $\textsc{RS}$-proof of $\Gamma$ we understand a tree $T_{\Gamma}$ of sequences satisfying the following conditions:

1. The topmost sequence, i.e. the root of $T_{\Gamma}$ is $\Gamma$,
2. all leafs are axioms,
3. the nodes are sequences such that each sequence on the tree follows from the ones immediately preceding it by one of the rules.

We picture, and write our proof trees with the node on the top, and leafs on the very bottom, instead of more common way, where the leafs are on the top and root is on the bottom of the tree.

In particular cases we write our proof trees indicating additionally the name of the inference rule used at each step of the proof. For example, if the proof of a given formula $A$ from three axioms was obtained by the subsequent use of
the rules \((\cap), (\cup), (\cap), (\cup), (\neg),\) and \((\Rightarrow)\), we represent it as the following proof tree:

\[
\begin{array}{c}
A \text{ (conclusion of } (\Rightarrow) ) \\
\mid (\Rightarrow) \\
\text{conclusion of } (\neg\neg) \\
\mid (\neg\neg) \\
\text{conclusion of } (\cup) \\
\mid (\cup) \\
\text{conclusion of } (\cap) \\
\Lambda(\cap) \\
\end{array}
\]

\[
\begin{array}{c}
\text{conclusion of } (\neg\neg) \\
\mid (\neg\neg) \\
\text{conclusion of } (\cup) \\
\mid (\cup) \\
\text{axiom} \\
\Lambda(\cap)
\end{array}
\]

\[
\begin{array}{c}
\text{axiom} \\
\text{axiom}
\end{array}
\]

The proof trees are often called derivation trees and we will use this notion as well. Remark that the derivation trees don’t represent a different definition of a formal proof. Trees represent a certain visualization of the proofs and any formal proof in any system can be represented in a tree form.

**Example 1**

*Here is a proof tree in RS of the de Morgan law \((\neg(a \cap b) \Rightarrow (\neg a \cup \neg b))\).*

\[
\begin{array}{c}
(\neg(a \cap b) \Rightarrow (\neg a \cup \neg b)) \\
\mid (\Rightarrow) \\
\neg\neg(a \cap b), (\neg a \cup \neg b) \\
\mid (\neg\neg) \\
(a \cap b), (\neg a \cup \neg b) \\
\Lambda(\cap)
\end{array}
\]

\[
\begin{array}{c}
a, (\neg a \cup \neg b) \\
\mid (\cup) \\
a, \neg a, \neg b
\end{array} \quad \begin{array}{c}
b, (\neg a \cup \neg b) \\
\mid (\cup) \\
b, \neg a, \neg b
\end{array}
\]

8
To obtain a "linear" formal proof of \((\neg(a \cap b) \Rightarrow (\neg a \cup \neg b))\) we just write down the tree as a sequence, starting from the leaves and going up (from left to right) to the root. The formal proof (with comments) thus obtained is:

\[
\begin{align*}
   a, \neg a, \neg b & \quad \text{(axiom)} \\
   b, \neg a, \neg b & \quad \text{(axiom)} \\
   a, (\neg a \cup \neg b) & \quad \text{(rule } \cup \text{)} \\
   b, (\neg a \cup \neg b) & \quad \text{(rule } \cup \text{)} \\
   (a \cap b), (\neg a \cup \neg b) & \quad \text{(rule } \cap \text{)} \\
   \neg\neg(a \cap b), (\neg a \cup \neg b) & \quad \text{(rule } \neg\neg \text{)} \\
   (\neg(a \cap b) \Rightarrow (\neg a \cup \neg b)) & \quad \text{(rule } \Rightarrow \text{).}
\end{align*}
\]

2.1 Search for Proofs and Decomposition Trees

The main advantage of the Gentzen style lies not in a way we generate proofs in them, but in the way we can search for proofs in them. That such proof searches happens to be deterministic and automatic. Before we describe a general proof search procedure for RS let us look at few simple examples. Consider now a formula \(A\) of the form of another de Morgan law

\[
(\neg(a \cup b) \Rightarrow (\neg a \cap \neg b)).
\]

Obviously it should have a proof in RS as we want it to be, and will prove later to be complete. The search for the proof consists of building a certain tree. We call it a decomposition tree, to be defined formally later. We proceed as follows.

Observe that the main connective of \(A\) is \(\Rightarrow\). So, if \(A\) had a proof in RS it would have come from the only possible rule used in its last step, namely the rule \((\Rightarrow)\) applied to its premiss, namely a sequence \(\neg\neg(a \cup b), (\neg a \cap \neg b)\). So the last step in the proof of \(A\) would look as follows.

\[
\begin{align*}
   (\neg(a \cup b) \Rightarrow (\neg a \cap \neg b)) \\
   \mid (\Rightarrow) \\
   \neg\neg(a \cup b), (\neg a \cap \neg b)
\end{align*}
\]

Now, if the sequence \(\neg\neg(a \cup b), (\neg a \cap \neg b)\) (and hence also the formula \(A\)) had a proof in RS its only step at this stage would have been the application of the rule \((\neg\neg)\) to a sequence \((a \cup b), (\neg a \cap \neg b)\). So, if \(A\) had a proof, its last two steps would have been:

\[
\begin{align*}
   (\neg(a \cup b) \Rightarrow (\neg a \cap \neg b)) \\
   \mid (\Rightarrow) \\
\end{align*}
\]
\[ \neg (a \cup b), (\neg a \land \neg b) \]
\[ \quad | (\neg \neg) \]
\[ (a \cup b), (\neg a \land \neg b) \]

Again, if the sequence \((a \cup b), (\neg a \land \neg b)\) had a proof in RS its only step at this stage would have been the application of the rule \((\cup)\) to a sequence \(a, b, (\neg a \land \neg b)\). So, if \(A\) had a proof, its last three steps would have been as follows.

\[
\begin{align*}
\neg (a \cup b) & \implies (\neg a \land \neg b) \\
| \quad (\implies) \\
\neg (a \cup b), (\neg a \land \neg b) & \quad | (\neg \neg) \\
| \quad (\cup) \\
(a \cup b), (\neg a \land \neg b) & \quad | (\cup) \\
a, b, (\neg a \land \neg b)
\end{align*}
\]

Now, if the sequence \(a, b, (\neg a \land \neg b)\) had a proof in RS its only step at this stage would have been the application of the rule \((\cap)\) to the sequences \(a, b, \neg a\) and \(a, b, \neg b\) as its left and right premisses, respectively. Both sequences are axioms and the following tree is a proof of \(A\) in RS.

\[
\begin{align*}
\neg (a \cup b) & \implies (\neg a \land \neg b) \\
| \quad (\implies) \\
\neg (a \cup b), (\neg a \land \neg b) & \quad | (\neg \neg) \\
| \quad (\cup) \\
(a \cup b), (\neg a \land \neg b) & \quad | (\cup) \\
a, b, (\neg a \land \neg b)
\end{align*}
\]

\[ a, b, \neg a \quad a, b, \neg b \]

From the above proof tree of \(A\) we construct, if we want, its formal proof, written in a vertical manner, by writing the two axioms, which form the two premisses of the rule \((\cap)\) one above the other. All other sequences remain the same. I.e. the following sequence of elements of \(F^*\) is a formal proof of \((\neg (a \cup b) \implies (\neg a \land \neg b))\) in RS.
Consider now a formula $A$ of the form

$$(((a \Rightarrow b) \cap \neg c) \cup (a \Rightarrow c)).$$

Observe that the main connective of $A$ is $\cup$. So, if $A$ had a proof in $\text{RS}$ it would have come from the only possible rule used in its last step, namely the rule ($\cup$) applied to a sequence $((a \Rightarrow b) \cap \neg c), (a \Rightarrow c)$. So the last step in the proof of $A$ would have been:

$$(((a \Rightarrow b) \cap \neg c) \cup (a \Rightarrow c)) | (\cup)$$

$$((a \Rightarrow b) \cap \neg c), (a \Rightarrow c)$$

Now, if the sequence $((a \Rightarrow b) \cap \neg c), (a \Rightarrow c)$ (and hence also had our formula) had a proof in $\text{RS}$ its only step at this stage would have been the application of the rule ($\cap$) to the sequences $(a \Rightarrow b), (a \Rightarrow c)$ and $\neg c, (a \Rightarrow c)$ as its left and right premisses, respectively. So, if $A$ had a proof, its last two steps would have been:

$$(((a \Rightarrow b) \cap \neg c) \cup (a \Rightarrow c)) | (\cup)$$

$$((a \Rightarrow b) \cap \neg c), (a \Rightarrow c)$$

$$\neg c, (a \Rightarrow c)$$

$$(a \Rightarrow b), (a \Rightarrow c)$$
Now, if the sequences \((a \Rightarrow b), (a \Rightarrow c)\) and \(\neg c, (a \Rightarrow c)\) had proofs in \(\text{RS}\), then their last, and the only steps would have been the the separate application of the rule \((\Rightarrow)\) to the sequences \(\neg a, b, (a \Rightarrow c)\) and \(\neg c, \neg a, c\), respectively. The sequence \(\neg c, \neg a, c\) is an axiom, so we stop the search on this branch. The sequence \(\neg a, b, (a \Rightarrow c)\) is not an axiom, so the search continues. In this case we can go one step further: if \(\neg a, b, (a \Rightarrow c)\) had a proof it would have been only by the application of the rule \((\Rightarrow)\) to a sequence \(\neg a, b, \neg a, c\) which is not an axiom and the search ends. The tree generated by this search is called a **decomposition tree** and is the following.

![Decomposition Tree Diagram]

The tree generated by this search contains a **non-axiom leaf**, so by definition, it is **not a proof**.

**Decomposition Rules and Trees**

The process of searching for the proof of a formula \(A\) in \(\text{RS}\) consists of building a certain tree, called a **decomposition tree** whose root is the formula \(A\), nodes correspond to sequences which are conclusions of certain rules (and those rules are well defined at each step by the way the node is built), and leafs are axioms or are sequences of a non-axiom literals. We prove that each formula \(A\) generates its **unique, finite** decomposition tree, \(T_A\) such that if all its leafs are axioms, the tree constitutes the proof of \(A\) in \(\text{RS}\). If there is a leaf of \(T_A\) that is **not an axiom**, the tree is not a proof, moreover, the proof of \(A\) does not exist.

Before we give a proper definition of the proof search procedure by building a decomposition tree we list few important observations about the structure of the rules of the system \(\text{RS}\).
Introduction of Connectives

The rules of $RS$ are defined in such a way that each of them introduces a new logical connective, or a negation of a connective to a sequence in its domain (rules $(∪), (⇒), (∩)$) or a negation of a new logical connective (rules $(¬∪), (¬∩), (¬⇒), (¬¬)$).

The rule $(∪)$ introduces a new connective $∪$ to a sequence $Γ', A, B, Δ$ and it becomes, after the application of the rule, a sequence $Γ', (A ∪ B), Δ$. Hence a name for this rule is $(∪)$.

The rule $(¬∪)$ introduces a negation of a connective, $¬∪$ by combining sequences $Γ', ¬A, Δ$ and $Γ', ¬B, Δ$ into one sequence (conclusion of the rule) $Γ', ¬(A ∪ B), Δ$. Hence a name for this rule is $(¬∪)$.

The same applies to all remaining rules of $RS$, hence their names say which connective, or the negation of which connective has been introduced by the particular rule.

Decomposition Rules

Building a proof search decomposition tree consists of using the inference rules in an inverse order; we transform them into rules that transform a conclusion into its premisses. We call such rules the decomposition rules. Here are all of $RS$ decomposition rules.

Disjunction decomposition rules

$\frac{Γ', (A ∪ B), Δ}{Γ', A, B, Δ}$,  $\frac{Γ', ¬(A ∪ B), Δ}{Γ', ¬A, Δ : Γ', ¬B, Δ}$

Conjunction decomposition rules

$\frac{Γ', (A ∩ B), Δ}{Γ', A, Δ : Γ', B, Δ'}$,  $\frac{Γ', ¬(A ∩ B), Δ}{Γ', ¬A, ¬B, Δ}$

Implication decomposition rules

$\frac{Γ', (A ⇒ B), Δ}{Γ', ¬A, B, Δ'}$,  $\frac{Γ', ¬(A ⇒ B), Δ}{Γ', A, Δ : Γ', ¬B, Δ}$

Negation decomposition rule

$\frac{Γ', ¬¬A, Δ}{Γ', A, Δ}$

where $Γ' ∈ LT^*$, $Δ ∈ F^*$, $A, B ∈ F$.

We write the decomposition rules in a visual tree form as follows.
Tree Decomposition Rules

(∪) rule

\[ \Gamma', (A \cup B), \Delta \]
\[ \mid (\cup) \]
\[ \Gamma', A, B, \Delta \]

(¬ ∪) rule

\[ \Gamma', \neg(A \cup B), \Delta \]
\[ \bigwedge(\neg \cup) \]
\[ \Gamma', \neg A, \Delta \quad \Gamma', \neg B, \Delta \]

(∩) rule:

\[ \Gamma', (A \cap B), \Delta \]
\[ \bigwedge(\cap) \]
\[ \Gamma', A, \Delta \quad \Gamma', B, \Delta \]

(¬ ∩) rule:

\[ \Gamma', \neg(A \cap B), \Delta \]
\[ \mid (\neg \cap) \]
\[ \Gamma', \neg A, \neg B, \Delta \]

(⇒) rule:

\[ \Gamma', (A \Rightarrow B), \Delta \]
\[ \mid (\cup) \]
\[ \Gamma', \neg A, B, \Delta \]
\((\neg \Rightarrow)\) rule:

\[
\begin{align*}
\Gamma', \neg(A \Rightarrow B), \Delta \\
\land (\neg \Rightarrow) \\
\Gamma', A, \Delta & \quad \Gamma', \neg B, \Delta
\end{align*}
\]

\((\neg \neg)\) rule:

\[
\begin{align*}
\Gamma', \neg
\neg\neg A, \Delta \\
\mid (\neg \neg) \\
\Gamma', A, \Delta
\end{align*}
\]

Observe that we use the same names for the inference and decomposition rules, as once the we have built the decomposition tree (with use of the decomposition rules) with all leaves being axioms, it constitutes a proof of \(A\) in \(\text{RS}\) with branches labeled by the proper inference rules.

Now we still need to introduce few useful definitions and observations.

**Definition 2**

1. A sequence \(\Gamma\) is **indecomposable** if and only if \(\Gamma \in \text{LT}^*\).

2. A formula \(A\) is **decomposable** if and only if \(A \in \mathcal{F} - \text{LT}\).

3. A sequence \(\Gamma\) is **decomposable** if and only if it contains a decomposable formula.

Directly from the definition 8 we have three simple, but important observations.

**Fact 2**

1. For any decomposable sequence \(\Gamma\), i.e. for any \(\Gamma \notin \text{LT}^*\) there is **exactly one** decomposition rule that can be applied to it. This rule is determined by the first decomposable formula in \(\Gamma\), and by the main connective of that formula.

2. If the main connective of the first decomposable formula is \(\lor, \land\), or \(\Rightarrow\), then the decomposition rule determined by it is \((\lor), (\land), \text{ or } (\Rightarrow)\), respectively.

3. If the main connective of the first decomposable formula is \(\neg\), then the decomposition rule determined by it is determined by the second connective of the formula. If the second connective is \(\lor, \land, \neg\), or \(\Rightarrow\), then corresponding decomposition rule is \((\neg \lor), (\neg \land), (\neg \neg)\) and \((\neg \Rightarrow)\).
Directly from the Fact 2 we we have the following lemma.

**Lemma 1 (Unique Decomposition)**

For any sequence $\Gamma \in F^*$,

$\Gamma \in LT^*$ or $\Gamma$ is in the domain of only one of the RS Decomposition Rules.

Now we define formally, for any formula $A \in F$ and $\Gamma \in F^*$ their decompositions trees. The decomposition tree for for the formula $A$ is a particular case (one element sequence) of the tree for a sequence $\Gamma$.

**Definition 3 (Decomposition Tree $T_A$)**

For each formula $A \in F$, its decomposition tree $T_A$ is a tree build as follows.

Step 1. The formula $A$ is the root of $T_A$ and for any node $\Gamma$ of the tree we follow the steps below.

Step 2. If $\Gamma$ is indecomposable, then $\Gamma$ becomes a leaf of the tree.

Step 3. If $\Gamma$ is decomposable, then we traverse $\Gamma$ from left to right to identify the first decomposable formula $B$ and identify the unique (Lemma 1) decomposition rule determined by the main connective of $B$. We put its left and right premisses as the left and right leaves, respectively.

Step 4. We repeat Step 2 and Step 3 until we obtain only leaves.

**Decomposition Tree $T_\Gamma$**

For each $\Gamma \in F^*$, a decomposition tree $T_\Gamma$ is a tree build as follows.

Step 1. The sequence $\Gamma$ is the root of $T_\Gamma$ and for any node $\Delta$ of the tree we follow the steps bellow.

Step 2. If $\Delta$ in indecomposable, then $\Delta$ becomes a leaf of the tree.

Step 3. If $\Delta$ is decomposable, then we traverse $\Delta$ from left to right to identify the first decomposable formula $B$ and identify the unique (Lemma 1) decomposition rule determined by the main connective of $B$. We put its left and right premisses as the left and right leaves, respectively.

Step 4. We repeat steps 2 and 3 until we obtain only leaves.

We now prove the following Decomposition Tree Theorem 1. This Theorem provides a crucial step in the proof of the Completeness Theorem for RS.

**Theorem 1 (Decomposition Tree)**

For any sequence $\Gamma \in F^*$ the following conditions hold.

1. $T_\Gamma$ is finite and unique.

2. $T_\Gamma$ is a proof of $\Gamma$ in RS if and only if all its leaves are axioms.
3. \( \not \vdash_{RS} \) if and only if \( T_{\Gamma} \) has a non-axiom leaf.

**Proof**

The tree \( T_{\Gamma} \) is unique by the Unique Decomposition Lemma 1. It is finite because there is a finite number of logical connectives in \( \Gamma \) and all decomposition rules diminish the number of connectives. If the tree has a non-axiom leaf it is not a proof by definition. By its uniqueness it also means that the proof does not exist.

**Exercise 2**

*Construct a decomposition tree \( T_A \) of the following formula \( A \).*

\[ A = ((a \cup b) \Rightarrow \neg a) \cup (\neg a \Rightarrow \neg c) \]

**Solution**

The formula \( A \) forms a one element decomposable sequence. The first decomposition rule used is determined by its main connective. We put a box around it, to make it more visible. The first and only rule applied is \((\cup)\) and we can write the first segment of our *decomposition tree* \( T_A \):

\[
T_A \\
((a \cup b) \Rightarrow \neg a) \cup (\neg a \Rightarrow \neg c) \\
| (\cup) \\
((a \cup b) \Rightarrow \neg a), (\neg a \Rightarrow \neg c)
\]

Now we decompose the sequence \( ((a \cup b) \Rightarrow \neg a), (\neg a \Rightarrow \neg c) \). It is a decomposable sequence with the first, decomposable formula \( ((a \cup b) \Rightarrow \neg a) \). The next step of the construction of our decomposition tree is determined by its main connective \( \Rightarrow \) (we put the box around it), hence the only rule determined by the sequence is \( \Rightarrow \). The second stage of the decomposition tree is now as follows.

\[
T_A \\
((a \cup b) \Rightarrow \neg a) \cup (\neg a \Rightarrow \neg c) \\
| (\cup) \\
((a \cup b) \Rightarrow \neg a), (\neg a \Rightarrow \neg c) \\
| (\Rightarrow) \\
\neg(a \cup b), \neg a, (\neg a \Rightarrow \neg c)
\]
The next sequence to decompose is the sequence \( \neg(a \cup b), \neg a, (\neg a \Rightarrow \neg c). \) 
The first decomposable formula is \( \neg(a \cup b). \) Its main connective is \( \neg, \) so determine the appropriate decomposition rule we have to examine next connective, which is \( \cup. \)

The rule determine by this stage of decomposition is \( (\neg \cup) \) and now the next stage of the decomposition tree \( T_A \) is as follows.

\[
T_A \\
((a \cup b) \Rightarrow \neg a \bigcup \neg a, (\neg a \Rightarrow \neg c)) \\
| \; (\cup) \\
((a \cup b) \Rightarrow \neg a), (\neg a \Rightarrow \neg c) \\
| \; (\Rightarrow) \\
\neg(a \cup b), \neg a, (\neg a \Rightarrow \neg c) \\
\wedge (\neg \cup) \\
\neg a, \neg a, (\neg a \Rightarrow \neg c) \quad \neg b, \neg a, (\neg a \Rightarrow \neg c)
\]

Now we have two decomposable sequences: \( \neg a, \neg a, (\neg a \Rightarrow \neg c) \) and \( \neg b, \neg a, (\neg a \Rightarrow \neg c). \) They both happen to have the same first decomposable formula \( (\neg a \Rightarrow \neg c). \) We decompose it simultenously and obtain the following:

\[
T_A \\
((a \cup b) \Rightarrow \neg a \bigcup \neg a, (\neg a \Rightarrow \neg c)) \\
| \; (\cup) \\
((a \cup b) \Rightarrow \neg a), (\neg a \Rightarrow \neg c) \\
| \; (\Rightarrow) \\
\neg(a \cup b), \neg a, (\neg a \Rightarrow \neg c) \\
\wedge (\neg \cup) \\
\neg a, \neg a, (\neg a \Rightarrow \neg c) \quad \neg b, \neg a, (\neg a \Rightarrow \neg c) \\
| \; (\Rightarrow) \\
\neg a, \neg a, \neg a, \neg c \quad \neg b, \neg a, \neg a, \neg c
\]
It is easy to see that we need only one more step to complete the process of constructing the unique decomposition tree of $T_A$, namely, by decomposing the sequences: $\neg a, \neg a, \neg a, \neg c$ and $\neg b, \neg a, \neg a, \neg c$.

The complete decomposition tree $T_A$ is:

$$
T_A
= ((a \cup b) \Rightarrow \neg a) \bigcup (\neg a \Rightarrow \neg c)
$$

$$
= ((a \cup b) \Rightarrow \neg a, (\neg a \Rightarrow \neg c))
$$

$$
= (\neg (a \cup b), \neg a, (\neg a \Rightarrow \neg c))
$$

All leaves are axioms, the tree represents a proof of $A$ in $\text{RS}$

**Exercise 3**

Prove that the formula $A = (((a \Rightarrow b) \cap \neg c) \cup (a \Rightarrow c))$ is not provable in $\text{RS}$, i.e.

$\not\vdash_{\text{RS}} (((a \Rightarrow b) \cap \neg c) \cup (a \Rightarrow c))$.

**Solution**

We construct the formula $A$ decomposition tree as follows.

$$
T_A
= (((a \Rightarrow b) \cap \neg c) \cup (a \Rightarrow c))
$$

$$
= ((a \Rightarrow b) \cap \neg c, (a \Rightarrow c))
$$

$\bigwedge (\cap)$

19
The above tree $T_A$ is unique by the Theorem 1 and represents the only possible search for proof of the formula $A = ((a \Rightarrow b) \cap \lnot c) \cup (a \Rightarrow c)$ in RS. It has a non-axiom leaf, hence by Theorem 1 the proof of $A$ in RS does not exists.

2.2 Strong Soundness and Completeness

Our main goal is to prove the Completeness Theorem for RS. The proof of completeness presented here is due to Rasiowa and Sikorski, as is the proof system RS. Their proof, and the proof system was inverted for Classical Predicate Logic and was published in 1961. We present their Predicate Logic proof system QRS together with its completeness proof in Chapter ???. Both completeness proofs, for propositional RS and predicate QRS proof systems, are constructive as they are based on direct construction of of counter model for any unprovable formula. The construction of a counter model for a formula $A$ uses directly its decomposition tree $T_A$. We call it a counter model determined by the tree $T_A$. Both proofs relay heavily of the notion of a strong soundness. We define it now, adopting Chapter ?? general definition to our semantics.

**Definition 4 (Strongly Sound Rules)**

Given a proof system $S = \langle \mathcal{L}, \mathcal{E}, LA, \mathcal{R} \rangle$ An inference rule $r \in \mathcal{R}$ of the form

\[
\begin{array}{c}
\hline
P_1 : P_2 : ... : P_m \\
\hline
\hline
C
\end{array}
\]

is **strongly sound** (undef classical semantics) if the following condition holds for all $v : \text{VAR} \rightarrow \{T, F\}$

\[ v \models \{P_1, P_2, \ldots, P_m\} \quad \text{if and only if} \quad v \models C. \]  \hfill (14)

We say it less formally that a rule $(r)$ is **strongly sound** if the conjunction of its premisses is logically equivalent with the conclusion, i.e.

\[ P_1 \cap P_2 \cap \ldots \cap P_m \equiv C. \]  \hfill (15)

**Definition 5 (Strongly Sound S)**

A proof system $S = \langle \mathcal{L}, \mathcal{E}, LA, \mathcal{R} \rangle$ is **strongly sound** (undef classical semantics) if and only if all logical axioms $LA$ are tautologies and all its rules of inference $r \in \mathcal{R}$ are **strongly sound**.
Theorem 2 (Strong Soundness)

The proof system RS (18) is strongly sound.

Proof

The logical axioms (9), (10) are tautologies by Fact ???. We prove as an example the strong soundness of two of inference rules: \((\cup)\) and \((\neg\cup)\). Proofs for all other rules follow the same patterns and are left as an exercise. By definition 4 of strong soundness we have to show the condition (15). Written formally it says that we have to show that if \(P_1, P_2\) are premisses of a given rule and \(C\) is its conclusion, then for all truth assignments \(v : VAR \rightarrow \{T, F\}\), \(v^*(P_1) = v^*(C)\) in case of one premiss rule, and \(v^*(P_1) \cap v^*(P_2) = v^*(C)\), in case of a two premisses rule. Consider the rule \((\cup)\).

\[
(\cup) \quad \Gamma', A, B, \Delta \quad \Gamma', (A \cup B), \Delta.
\]

By the definition:

\[
v^*(\Gamma', A, B, \Delta) = v^*(\delta_{\{\Gamma', A, B, \Delta\}}) = v^*(\Gamma') \cup v^*(A) \cup v^*(B) \cup v^*(\Delta) = v^*(\Gamma') \cup v^*(A \cup B) \cup v^*(\Delta) = v^*(\delta_{\{\Gamma', A \cup B, \Delta\}}) = v^*(\Gamma', (A \cup B), \Delta).
\]

Consider the rule \((\neg\cup)\).

\[
(\neg\cup) \quad \Gamma', \neg A, \Delta \quad \Gamma', \neg B, \Delta \quad \Gamma', \neg (A \cup B), \Delta.
\]

By the definition:

\[
v^*(\Gamma', \neg A, \Delta) \cap v^*(\Gamma', \neg B, \Delta) = (v^*(\Gamma') \cup v^*(\neg A) \cup v^*(\Delta)) \cap (v^*(\Gamma') \cup v^*(\neg B) \cup v^*(\Delta)) = (v^*(\Gamma') \cup v^*(\neg A) \cup v^*(\Delta)) \cap (v^*(\Gamma') \cup v^*(\neg B) \cup v^*(\Delta)) = (v^*(\Gamma') \cup v^*(\neg A) \cap v^*(\neg B)) = v^*(\delta_{\{\Gamma', \neg (A \cup B), \Delta\}}) = v^*(\Gamma', \neg (A \cup B), \Delta).
\]

Observe that the strong soundness implies soundness (not only by name!), hence we have also proved the following

Theorem 3 (Soundness for RS)

For any \(\Gamma \in F^*\),

\(\Gamma \vdash_{RS} \Gamma\), then \(\models \Gamma\). In particular, for any \(A \in F\), \(A \vdash_{RS} A\), then \(\models A\).

We have just proved (Theorem 2) that all the rules of inference of RS of are strongly sound, i.e. \(C \equiv P\) and \(C \equiv P_1 \cap P_2\). The strong soundness of the rules means that if at least one of premisses of a rule is false, so is its conclusion. Hence
given a formula \(A\), such that its \(T_A\) has a branch ending with a non-axiom leaf. By Strong Soundness Theorem 2, any \(v\) that make this non-axiom leaf false also falsifies all sequences on that branch, and hence falsifies the formula \(A\). This means that any \(v\), such that it falsifies a non-axiom leaf is a **counter-model** for \(A\). We have hence proved the following.

**Theorem 4 (Counter Model)**

*Given a formula \(A \in \mathcal{F}\) such that its decomposition tree \(T_A\) contains a non-axiom leaf \(L_A\). Any truth assignment \(v\) that falsifies the non-axiom leaf \(L_A\) is a counter-model for \(A\). We call it a counter-model for \(A\) determined by the decomposition tree \(T_A\).*

Here is a simple example explaining how the construction of a counter-model determined by the decomposition tree of a works. Consider a tree

\[
T_A
\]

\[
(((a \Rightarrow b) \cap \neg c) \cup (a \Rightarrow c))
\]

\[
| (\cup)
\]

\[
((a \Rightarrow b) \cap \neg c), (a \Rightarrow c)
\]

\[
\bigwedge (\cap)
\]

\[
(a \Rightarrow b), (a \Rightarrow c)
\]

\[
\neg c, (a \Rightarrow c)
\]

\[
| (\Rightarrow)
\]

\[
\neg a, b, (a \Rightarrow c)
\]

\[
\neg c, \neg a, c
\]

\[
| (\Rightarrow)
\]

\[
\neg a, b, \neg a, c
\]

The tree \(T_A\) has a non-axiom leaf \(L_A : \neg a, b, \neg a, c\). The truth assignment \(v : \text{VAR} \rightarrow \{T, F\}\) that falsifies the leaf \(\neg a, b, \neg a, c\) must be such that

\[
v^*(-a, b, \neg a, c) = v^*(-a) \cup v^*(b) \cup v^*(\neg a) \cup v^*(c) = v(a) \cup v(b) \cup \neg v(a) \cup v(c) = F,
\]

i.e. \(v\) must be such that \(\neg v(a) \cup v(b) \cup \neg v(a) \cup v(c) = F\). We hence get that \(v(a) = T, v(b) = F, v(c) = F\). By the Counter Model Theorem 4, the truth assignment \(v\) determined by the non-axiom leaf also falsifies the formula \(A\), i.e. we proved that \(v\) is a counter model for \(A\) and

\[
\not\models ((a \Rightarrow b) \cap \neg c) \cup (a \Rightarrow c).
\]

The Counter Model Theorem 4, says that the logical value \(F\) determined by the evaluation a non-axiom leaf "climbs" the decomposition tree. We picture it as follows.
\[ T_A \]
\[ (((a \Rightarrow b) \cap \neg c) \cup (a \Rightarrow c)) = F \]
\[ | (\cup) \]
\[ ((a \Rightarrow b) \cap \neg c), (a \Rightarrow c) = F \]
\[ \bigwedge (\cap) \]
\[ (a \Rightarrow b), (a \Rightarrow c) = F \]
\[ \neg c, (a \Rightarrow c) \]
\[ | (\Rightarrow) \]
\[ \neg a, b, (a \Rightarrow c) = F \]
\[ \neg c, \neg a, c \]
\[ | (\Rightarrow) \]
\[ \neg a, \neg a, c = F \]

**Observe** that the same counter model construction applies to any other non-axiom leaf of \( T_A \), if exists. The other non-axiom leaf of \( T_A \) defines another evaluation of the non-axiom leaf to \( F \) that also "climbs the tree" and hence defines another counter-model for a formula \( A \). By Counter Model 4 all possible restricted counter-models for \( A \) are those determined by its all non-axioms leaves.

In our case the tree \( T_A \) has only one non-axiom leaf, and hence the formula \(((a \Rightarrow b) \cap \neg c) \cup (a \Rightarrow c))\) only only one restricted counter model.

Our main goal is to prove the Completeness Theorem for \( \text{RS} \). We prove first the Completeness Theorem for formulas \( A \in \mathcal{F} \) and then we generalize it to any sequences \( \Gamma \in \mathcal{F}^* \).

**Theorem 5 (Completeness Theorem)**

*For any formula \( A \in \mathcal{F} \),
1. \( \vdash_{\text{RS}} A \) if and only if \( \models A \), and for any \( \Gamma \in \mathcal{F}^* \),
2. \( \vdash_{\text{RS}} \Gamma \) if and only if \( \models \Gamma \).*

**Proof**

Case 1. We have already proved the Soundness Theorem 3, so we need to prove only the completeness part of it, namely to prove the implication:

\[ \text{if} \quad \models A, \quad \text{then} \quad \vdash_{\text{RS}} A. \quad (16) \]

We prove instead of the opposite implication:

\[ \text{if} \quad \not\vdash_{\text{RS}} A \quad \text{then} \quad \not\models A. \quad (17) \]

Assume that \( A \) is any formula is such that \( \not\vdash_{\text{RS}} A \). By the Decomposition Tree Theorem 1 the tree \( T_A \) contains a non-axiom leaf \( L_A \). We use the non-axiom
leaf \( L_A \) to define a truth assignment \( v : VAR \rightarrow \{ T, F \} \) which falsifies it as follows:

\[
v(a) = \begin{cases} 
  F & \text{if } a \text{ appears in } L_A \\
  T & \text{if } \neg a \text{ appears in } L_A \\
  \text{any value} & \text{if } a \text{ does not appear in } L_A 
\end{cases}
\]

By the Counter Model Theorem 4 we have that \( v \) also falsifies the formula \( A \). We proved that

\[ \not\models A \]

and it ends the proof of the case 1.

Case 2. Assume that \( \Gamma \in F^* \) is any sequence such that \( \not\models_{RS} \Gamma \). But obviously, \( \models_{RS} \Gamma \) if and only if \( \models_{RS} \delta_\Gamma \), where \( \delta_\Gamma \) is any disjunction of all formulas of \( \Gamma \). So \( \not\models_{RS} \Gamma \) if and only if \( \not\models_{RS} \delta_\Gamma \) and by already proven Case 1, \( \not\models \delta_\Gamma \) what is obviously equivalent to \( \not\models \Gamma \). This ends the proof of Case 2 and Completeness Theorem.

### 3 Proof Systems RS1 and RS2

We present here a two modifications of the system RS as an exercise of importance of paying close attention to the syntax. Proof systems might be, as all presented here RS type systems are, semantically identical, nevertheless they are very different proof systems.

Language of \( RS_1 \) is the same as the language of RS, i.e.

\[ \mathcal{L} = \mathcal{L}_{\{\neg, \cup, \cap\}}. \]

Rules of inference of \( RS_1 \) operate as rules of RS on finite sequences of formulas and we adopt

\[ \mathcal{E} = F^* \]

as the set of expressions of \( RS_1 \). We denote them, as before, by \( \Gamma', \Delta', \Sigma', \ldots \), with indices if necessary.

The proof system \( RS_1 \) contains seven inference rules, denoted by the same symbols as the rules of RS, namely \( \cup, \neg\cup, \cap, \neg\cap, \Rightarrow, \neg\Rightarrow \).

The inference rules of \( RS_1 \) are quite similar to the rules of RS Look at them carefully to see where lies the difference.

**Reminder** Any propositional variable, or a negation of propositional variable is called a literal. \( LT = VAR \cup \{ \neg a : a \in VAR \} \) is called a set of all propositional literals. The variables are called positive literals. Negations of variables are called negative literals. We denote, as before, by \( \Gamma', \Delta', \Sigma', \ldots \)
finite sequences (empty included) formed out of literals. We adopt all logical axiom of RS as the axioms of RS1, i.e. logical axioms LA of RS1 are:

\[ \Gamma_1', a, \Gamma_2', \neg a, \Gamma_3', \Gamma_1', \neg a, \Gamma_2', a, \Gamma_3' \]

where \( a \in VAR \) is any propositional variable.

We define the inference rules of RS1 as follows.

**Disjunction rules**

\[
\begin{align*}
(\cup) & \quad \frac{\Gamma, A, B, \Delta'}{\Gamma, (A \cup B), \Delta'}, & (\neg \cup) & \quad \frac{\neg A, \Delta'}{\Gamma, \neg (A \cup B), \Delta'},
\end{align*}
\]

**Conjunction rules**

\[
\begin{align*}
(\cap) & \quad \frac{\Gamma, A, \Delta'}{\Gamma, (A \cap B), \Delta'}, & (\neg \cap) & \quad \frac{\neg A, \neg B, \Delta'}{\Gamma, \neg (A \cap B), \Delta'},
\end{align*}
\]

**Implication rules**

\[
\begin{align*}
(\Rightarrow) & \quad \frac{\Gamma, A, B, \Delta'}{\Gamma, (A \Rightarrow B), \Delta'}, & (\neg \Rightarrow) & \quad \frac{\Gamma, A, \Delta'}{\Gamma, \neg (A \Rightarrow B), \Delta'},
\end{align*}
\]

**Negation rule**

\[
\begin{align*}
(\neg \neg) & \quad \frac{\Gamma, A, \Delta'}{\Gamma, \neg \neg A, \Delta'}
\end{align*}
\]

where \( \Gamma \in F^* \), \( \Delta' \in LT^* \), \( A, B \in F \).

**Proof System RS1**

Formally we define the proof system RS1 as follows.

\[
RS1 = (L_{\neg, \Rightarrow, \cup, \cap}, E, LA, R),
\]

(18)

where \( E = \{ \Gamma : \Gamma \in F^* \} \), LA is the set logical axioms and \( R \) is the set of rules of inference defined above.

**Exercise 4**

*Construct a proof in RS1 of a formula*

\[ A = (\neg(a \cap b) \Rightarrow (\neg a \cup \neg b)). \]
Solution
The decomposition tree below is a proof of \( A \) in RS1 as all its leaves are axioms.

\[
T_A
\]

\[
(- (a \land b) \Rightarrow (-a \lor -b))
\]

\[
| (\Rightarrow)
\]

\[
(- (- (a \land b)) , (-a \lor -b))
\]

\[
| (\lor)
\]

\[
(- (a \land b) , -a , -b)
\]

\[
| (\neg)
\]

\[
(a \land b) , -a , -b
\]

\[
\land (\land)
\]

\[
a , -a , -b
\]

\[
b , -a , -b
\]

Exercise 5
Prove that RS1 is strongly sound.

Solution
Observe that the system RS1 is obtained from RS by changing the sequence \( \Gamma' \) into \( \Gamma \) and the sequence \( \Delta \) into \( \Delta' \) in all of the rules of inference of RS. These changes do not influence the essence of proof of strong soundness of the rules of RS. One has just to replace the sequence \( \Gamma' \) by \( \Gamma \) and the sequence \( \Delta \) by \( \Delta' \) in the proof of strong soundness of each rule of RS to obtain a corresponding proof of strong soundness of corresponding rule of RS1. We do it, for example for the rule \( (\lor) \) of RS1:

\[
(\lor) \quad \Gamma , A , B , \Delta' \\
\Gamma , (A \lor B) , \Delta'.
\]

We evaluate:

\[
v^*(\Gamma , A , B , \Delta') = v^*(\delta(\Gamma , A , B , \Delta')) = v^*(\Gamma) \cup v^*(A) \cup v^*(B) \cup v^*(\Delta')
\]

\[
= v^*(\Gamma) \cup v^*(A \cup B) \cup v^*(\Delta') = v^*(\delta(\Gamma , (A \cup B) , \Delta')) = v^*(\Gamma , (A \cup B) , \Delta').
\]

Exercise 6
Define in your own words, for any formula \( A \in \mathcal{F} \) the decomposition tree \( T_A \) in RS1.
Solution
The definition of the decomposition tree $T_A$ is again, it its essence similar to the one for RS except for the changes which reflect the differences in the corresponding rules of inference. We follow now the following steps.

**Step 1** Decompose $A$ using a rule defined by its main connective.

**Step 2** Traverse resulting sequence $\Gamma$ on the new node of the tree from right to left and find the first decomposable formula.

**Step 3** Repeat **Step 1** and **Step 2** until there is no more decomposable formulas. **End** of tree construction.

Exercise 7

*Prove the following Completeness Theorem for RS1.*

**Theorem 6**

For any formula $A \in \mathcal{F}$,

1. $\vdash_{\text{RS1}} A$ if and only if $\models A$, and for any $\Gamma \in \mathcal{F}^*$,
2. $\vdash_{\text{RS1}} \Gamma$ if and only if $\models \Gamma$.

**Solution** Part 1.

Observe that directly from the definition of the uniqueness of the decomposition tree $T_A$ we have that the following holds.

**Fact 3**

The decomposition tree $T_A$ is a proof if and only if all leaves are axioms and the proof does not exist otherwise, i.e. we have that $\not\vdash_{\text{RS1}} A$ if and only if there is a non-axiom leaf on $T_A$.

The Fact 3 together with strong soundness of the rules of inference of RS1 justify the correctness of construction of a counter-model generated by a tree with a non-axiom leaf and hence the correctness of the following proof of the Completeness Theorem.

We prove, as we did in case of RS the implication

$$\text{if } \not\vdash_{\text{RS1}} A \text{ then } \not\models A.$$ 

Assume that $A$ is any formula such that $\not\vdash_{\text{RS1}} A$. By the Fact 3 the decomposition tree $T_A$ contains a non-axiom leaf $L_A$. We use the non-axiom leaf $L_A$ and define a truth assignment $v$ which falsifies $A$, as follows:

$$v(a) = \begin{cases} 
F & \text{if } a \text{ appears in } L_A \\
T & \text{if } \neg a \text{ appears in } L_A \\
\text{any value} & \text{if } a \text{ does not appear in } L_A.
\end{cases}$$
This proves, by the strong soundness of \textbf{RS1}, that \( \not\models A \).

The proof of Part 2. is identical to the proof in \textbf{RS} case.

**Proof System RS2**

System \textbf{RS2} is a proof system obtained from \textbf{RS} by changing the sequences \( \Gamma' \) into \( \Gamma \) in all of the rules of inference of \textbf{RS}. The logical axioms \( \text{LA} \) remain the same. Observe that now the decomposition tree may not be unique.

**Exercise 8**

*Construct two* decomposition trees in \textbf{RS2} *of the formula*

\[
A = (\neg(\neg a \Rightarrow (a \land \neg b)) \Rightarrow (\neg (\neg a \cup \neg b))).
\]

**Solution**

Here are two out of many more decomposition trees.

\[
\begin{align*}
\text{T1} &_A \\
& (\neg(\neg a \Rightarrow (a \land \neg b)) \Rightarrow (\neg (\neg a \cup \neg b'))) \\
& \quad | (\Rightarrow) \\
& \quad \neg(\neg a \Rightarrow (a \land \neg b)), (\neg (\neg a \cup \neg b')) \\
& \quad \quad | (\neg\neg) \\
& \quad (\neg a \Rightarrow (a \land \neg b)), (\neg (\neg a \cup \neg b)) \\
& \quad \quad \quad | (\Rightarrow) \\
& \quad \neg\neg a, (a \land \neg b), (\neg a \land (\neg a \lor \neg b)) \\
& \quad \quad \quad \quad | (\neg\neg) \\
& \quad a, (a \land \neg b), (\neg a \land (\neg a \lor \neg b)) \\
& \quad \quad \quad \quad \quad \land (\cap) \\
& a, a, (\neg a \land (\neg a \lor \neg b)) \quad a, a, (\neg a \land (\neg a \lor \neg b)) \quad a, \neg b, (\neg a \land (\neg a \lor \neg b)) \quad a, \neg b, (\neg a \land (\neg a \lor \neg b)) \\
& \quad \quad \quad \quad \quad \quad \quad \quad \land (\cap) \\
& a, a, \neg a, (\neg a \lor \neg b) \quad a, a, (\neg a \lor \neg b) \quad a, \neg b, \neg a \quad a, \neg b, (\neg a \lor \neg b) \\
& \quad \quad | (\lor) \quad \quad | (\lor) \quad \quad \text{axiom} \quad \quad | (\lor) \\
& a, a, \neg a, \neg a, \neg b \quad a, a, \neg a, \neg b \quad a, \neg b, \neg a \quad a, \neg b, \neg a, \neg b \\
& \quad \quad \text{axiom} \quad \quad \text{axiom} \quad \quad \text{axiom}
\end{align*}
\]

The other tree is:
Exercise 9

Explain why the system RS₂ is strongly sound. You can use the strong soundness of the system RS.

Solution

The only one difference between RS and RS₂ is that in RS₂ each inference rule has at the beginning a sequence of any formulas, not only of literals, as in RS. So there are many ways to apply the decomposition rules while constructing the decomposition tree, but it does not affect strong soundness, since for all rules of RS₂ premisses and conclusions are still logically equivalent as they were in RS.

Consider, for example, RS₂ rule

\[
\text{(∪)} \quad \Gamma, A, B, \Delta \\
\Gamma, (A ∪ B), \Delta.
\]

We evaluate \(v^*(\Gamma, A, B, \Delta) = v^*(\Gamma) ∪ v^*(A) ∪ v^*(B) \cup v^*(\Delta) = v^*(\Gamma) ∪ v^*(A ∪ B) \cup v^*(\Delta) = v^*(\Gamma, (A ∪ B), \Delta).\) Similarly, as in RS, we show all other rules of RS₂ to be strongly sound, thus RS₂ is sound.
Exercise 10

Define shortly, in your own words, for any formula $A$, its decomposition tree $T_A$ in $RS2$. Justify why your definition is correct. Show that in $RS2$ the decomposition tree for some formula $A$ may not be unique.

Solution

Given a formula $A$. The decomposition tree $T_A$ can be defined as follows. It has $A$ as a root. For each node, if there is a rule of $RS2$ which conclusion has the same form as node sequence, i.e. there is a decomposition rule to be applied, then the node has children that are premises of the rule. If the node consists only of literals (i.e. there is no decomposition rule to be applied), then it does not have any children. The last statement define a termination condition for the tree $T_A$.

This definition defines correctly the decomposition tree $T_A$ as it identifies and uses appropriate decomposition rules. Since all rules of inference of $RS2$ have a sequence $\Gamma$ instead of $\Gamma'$ as it was in $RS$, the choice of the decomposition rule for a node may not unique. For example consider a node $(a = b), (b \cup a)$, $\Gamma$ in the $RS2$ rules may be a sequence of formulas, not only literals, so for the node $(a = b), (b \cup a)$ we can choose as a decomposition rule either $(=)$ or $(\cup)$. This leads to a non-unique tree.

Exercise 11

Prove the following Completeness Theorem for $RS2$.

Theorem 7

For any formula $A \in F$,
1. $\vdash_{RS2} A$ if and only if $\models A$, and for any $\Gamma \in F^*$,
2. $\vdash_{RS2} \Gamma$ if and only if $\models \Gamma$.

Solution

We need to prove the completeness part only, as the Soundness has been already proved, i.e. we have to prove the implication (Part 1): for any formula $A$,

$$\text{if } \nvdash_{RS2} A \text{ then } \nmodels A.$$ 

Assume $\nvdash_{RS2} A$. Then every decomposition tree of $A$ has at least one non-axiom leaf. Otherwise, there would exist a tree with all axiom leaves and it would be a proof for $A$. Let $T_A$ be a set of all decomposition trees of $A$. We choose an arbitrary $T_A \in T_A$ with at least one non-axiom leaf $L_A$. We use the non-axiom leaf $L_A$ to define a truth assignment $v$ which falsifies $A$, as follows:

$$v(a) = \begin{cases} 
F & \text{if } a \text{ appears in } L_A \\
T & \text{if } \neg a \text{ appears in } L_A \\
\text{any value} & \text{if } a \text{ does not appear in } L_A
\end{cases}$$
The value for a sequence that corresponds to the leaf in is F. Since, because of the strong soundness F "climbs" the tree, we found a counter-model for A. This proves that $\not\models A$. Part 2. proof is identical to the proof in RS case.

**Exercise 12**

Write a procedure $TREE_A$ such that for any formula $A$ of RS2 it produces its unique decomposition tree.

**Solution**

Here is the procedure.

Procedure $TREE_A$ (Formula A, Tree T)

\{ 
    \begin{align*}
    B &= \text{ChooseLeftMostFormula}(A) \quad \text{// Choose the left most formula that is not a literal} \\
    c &= \text{MainConnective}(B) \quad \text{// Find the main connective of B} \\
    R &= \text{FindRule}(c) \quad \text{// Find the rule which conclusion that has this connective} \\
    P &= \text{Premises}(R) \quad \text{// Get the premises for this rule} \\
    \text{AddToTree}(A, P) &\quad \text{// add premises as children of A to the tree} \\
    \text{For all } p \text{ in } P &\quad \text{// go through all premises} \\
    &\quad \text{TREE}_A(p, T) \quad \text{// build subtrees for each premiss}
    \end{align*}
\}

**Exercise 13**

Prove completeness of your Procedure $TREE_A$.

**Solution**

Procedure $TREE_A$ provides a unique tree, since it always chooses the most left indecomposable formula for a choice of a decomposition rule and there is only one such rule. This procedure is equivalent to RS system, since with the decomposition rules of RS the most left decomposable formula is always chosen. The proof RS system is complete, thus this Procedure $TREE_A$ is complete.

### 4 Gentzen Sequent Systems GL, G, LK

Gentzen proof systems GL and G for the classical propositional logic presented here are inspired by and all are versions of the original (1934) Gentzen system LK. Their axioms, the rules of inference of the proof system considered here operate, as the original Gentzen system LK, on expressions called by Gentzen sequents, hence the name Gentzen sequent proof systems, called also Gentzen sequent calculus, or sequents calculus. The original system LK is presented and discussed in detail in section 6.
4.1 Gentzen Sequent Systems GL and G

The system GL presented here is the most similar in its structure to the system RS (18) and hence is the first to be considered. It admits a constructive proof of the Completeness Theorem that is very similar to the proof of the Completeness Theorem for the system RS.

Language of GL

We adopt a propositional language \( \mathcal{L} = \mathcal{L}_{\{\cup, \cap, \Rightarrow, \neg\}} \) with the set of formulas denoted by \( \mathcal{F} \) and we add a new symbol \( \rightarrow \) called a Gentzen arrow, to it. As the next step we build expressions called sequents. The sequents are built out of finite sequences (empty included) of formulas, i.e. elements of \( \mathcal{F}^* \), and the additional sign \( \rightarrow \).

We denote, as in the RS system, the finite sequences of formulas by Greek capital letters \( \Gamma, \Delta, \Sigma \), with indices if necessary. We define a sequent as follows.

**Definition 6 (Sequent)**

For any \( \Gamma, \Delta \in \mathcal{F}^* \), the expression

\[ \Gamma \rightarrow \Delta \]

is called a sequent. \( \Gamma \) is called the antecedent of the sequent, \( \Delta \) is called the succedent, and each formula in \( \Gamma \) and \( \Delta \) is called a sequent-formula.

Intuitively, a sequent \( A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m \) (where \( n, m \geq 1 \)) means: if \( A_1 \cap \ldots \cap A_n \) then \( B_1 \cup \ldots \cup B_m \). The sequent \( A_1, \ldots, A_n \rightarrow \) (where \( n \geq 1 \)) means that \( A_1 \cap \ldots \cap A_n \) yields a contradiction. The sequent \( \rightarrow B_1, \ldots, B_m \) (where \( m \geq 1 \)) means that \( B_1 \cup \ldots \cup B_m \) is true. The empty sequent \( \rightarrow \) means a contradiction.

Given non empty sequences \( \Gamma, \Delta \), we denote by \( \sigma_\Gamma \) any conjunction of all formulas of \( \Gamma \), and by \( \delta_\Delta \) any disjunction of all formulas of \( \Delta \). The intuitive semantics for a sequent \( \Gamma \rightarrow \Delta \) (where \( \Gamma, \Delta \) are nonempty) is hence that it is logically equivalent to the formula \( \sigma_\Gamma \Rightarrow \delta_\Delta \), i.e.

\[ \Gamma \rightarrow \Delta \equiv (\sigma_\Gamma \Rightarrow \delta_\Delta). \]

Formal semantics

Formally, let \( v : VAR \rightarrow \{T, F\} \) be a truth assignment, \( v^* \) its extension to the set of formulas \( \mathcal{F} \). We extend \( v^* \) to the set

\[ SQ = \{ \; \Gamma \rightarrow \Delta : \; \Gamma, \Delta \in \mathcal{F}^* \} \]

of all sequents as follows.
Definition 7 For any sequent \( \Gamma \rightarrow \Delta \in SQ \),

\[ v^*(\Gamma \rightarrow \Delta) = v^*(\sigma_\Gamma) \Rightarrow v^*(\delta_\Delta). \]

In the case when \( \Gamma = \emptyset \) we define: \( v^*(\rightarrow \Delta) = T \Rightarrow v^*(\delta_\Delta) \). In the case \( \Delta = \emptyset \) we define \( v^*(\Gamma \rightarrow ) = v^*(\sigma_\Gamma) \Rightarrow F \).

Model

The sequent \( \Gamma \rightarrow \Delta \) is **satisfiable** if there is a truth assignment \( v : VAR \rightarrow \{T,F\} \) such that \( v^*(\Gamma \rightarrow \Delta) = T \). Such a truth assignment is called a **model** for \( \Gamma \rightarrow \Delta \). We write

\[ v \models \Gamma \rightarrow \Delta. \]

Counter-model

The sequent \( \Gamma \rightarrow \Delta \) is **falsifiable** if there is a truth assignment \( v \), such that \( v^*(\Gamma \rightarrow \Delta) = F \). In this case \( v \) is called a **counter-model** for \( \Gamma \rightarrow \Delta \) and we write it as

\[ v \not\models \Gamma \rightarrow \Delta. \]

Tautology

The sequent \( \Gamma \rightarrow \Delta \) is a **tautology** if \( v^*(\Gamma \rightarrow \Delta) = T \) for all truth assignments \( v : VAR \rightarrow \{T,F\} \) and we write

\[ \models \Gamma \rightarrow \Delta. \]

Example 2

Let \( \Gamma \rightarrow \Delta \) be a sequent

\[ a, (b \land a) \rightarrow \neg b, (b \Rightarrow a). \]

Any truth assignment \( v \), such that \( v(a) = T \) and \( v(b) = T \) is a model for \( \Gamma \rightarrow \Delta \), i.e.

\[ \models a, (b \land a) \rightarrow \neg b, (b \Rightarrow a). \]

We verify it by performing the following computation.

\[ v^*(a, (b \land a) \rightarrow \neg b, (b \Rightarrow a)) = v^*(\sigma_{\{a,(b\land a)\}}) \Rightarrow v^*(\delta_{\neg b, (b \Rightarrow a)}) = v(a) \cap (v(b) \cap v(a)) \Rightarrow \neg v(b) \cup (v(b) \Rightarrow v(a)) = T \cap T \cap T \Rightarrow \neg T \cup (T \Rightarrow T) = T \Rightarrow (F \cup T) = T \Rightarrow T = T. \]

Observe that the only \( v \) for which \( v^*(\Gamma) = v^*(a, (b\land a) = T \) is the above \( v(a) = T \) and \( v(b) = T \) that is a model for \( \Gamma \rightarrow \Delta \). Hence it is impossible to find \( v \) which would falsify it, what proves that \( \Gamma \rightarrow \Delta \) is a tautology, i.e.

\[ \models a, (b \land a) \rightarrow \neg b, (b \Rightarrow a). \]
The Proof System GL

The rules of inference of GL are of the form:

\[
\frac{P_1 \quad P_2}{C} \quad \text{or} \quad \frac{P_1 : P_2}{C},
\]

where \( P_1, P_2 \) and \( C \) are sequents. \( P_1, P_2 \) are called premisses and \( C \) is called the conclusion of the rule of inference. Each rule of inference introduces a new logical connective to the antecedent or to the succedent of the conclusion sequent. We denote the rule that introduces the logical connective \( \circ \) to the antecedent of the conclusion sequent \( P \) by \((\circ \rightarrow)\). The notation \((\rightarrow \circ)\) means that the logical connective is introduced to the succedent of the conclusion sequent \( P \).

As our language contains the connectives: \( \cap, \cup, \Rightarrow \) and \( \neg \), we are going to adopt the following inference rules: \((\cap \rightarrow)\) and \((\rightarrow \cap)\), \((\cup \rightarrow)\) and \((\rightarrow \cup)\), \((\Rightarrow \rightarrow)\) and \((\rightarrow \Rightarrow)\), and finally, \((\neg \rightarrow)\) and \((\rightarrow \neg)\).

Definition 8

Finite sequences formed out of positive literals i.e. out of propositional variables are called indecomposable. We denote them as before by \( \Gamma', \Delta' \), with indices, if necessary.

A sequent is indecomposable if it is formed out of indecomposable sequences, i.e. is of the form

\[
\Gamma' \rightarrow \Delta'
\]

for any \( \Gamma', \Delta' \in VAR^* \).

Axioms of GL

As the axioms of GL we adopt any indecomposable sequent sequent which contains a positive literal \( a \) (variable) that appears on both sides of the sequent arrow \( \rightarrow \), i.e any sequent of the form

\[
\Gamma'_1, a, \Gamma'_2 \rightarrow \Delta'_1, a, \Delta'_2,
\]

for any \( a \in VAR \) and any sequences \( \Gamma'_1, \Gamma'_2, \Delta'_1, \Delta'_2 \in VAR^* \).

Semantic Link

Consider axiom (21). Directly from the Definition 7 of semantics for bf GL we evaluate (in shorthand notation), for any truth assignments \( v : VAR \rightarrow \{T, F\} \), the following (in shorthand notation).

\[
v^*(\Gamma'_1, a, \Gamma'_2 \rightarrow \Delta'_1, a, \Delta'_2) =
(\sigma_{\Gamma'_1} \cap a \cap \sigma_{\Gamma'_2}) \Rightarrow (\delta_{\Delta'_1} \cup a \cup \delta_{\Delta'_2}) = T.
\]
The evaluation is correct because $\models (((A \cap a) \cap B) \Rightarrow (C \cup a) \cup D))$. We have thus proved the following.

**Fact 4**

*Logical axioms of GL are tautologies.*

### Inference Rules of GL (22)

We adopt the following rules of inference.

**Conjunction rules**

$$(\cap \rightarrow) \quad \frac{\Gamma', A, B, \Gamma \rightarrow \Delta'}{\Gamma', (A \cap B), \Gamma \rightarrow \Delta'}.$$  

$$(\rightarrow \cap) \quad \frac{\Gamma \rightarrow \Delta, A, \Delta'}{\Gamma \rightarrow \Delta, (A \cap B), \Delta'}.$$  

**Disjunction rules**

$$(\rightarrow \cup) \quad \frac{\Gamma \rightarrow \Delta, A, B, \Delta'}{\Gamma \rightarrow \Delta, (A \cup B), \Delta'},$$  

$$(\cup \rightarrow) \quad \frac{\Gamma, A, \Gamma \rightarrow \Delta'; \Gamma', B, \Gamma \rightarrow \Delta'}{\Gamma, (A \cup B), \Gamma \rightarrow \Delta'}.$$  

**Implication rules**

$$(\rightarrow \Rightarrow) \quad \frac{\Gamma', A, \Gamma \rightarrow \Delta, B, \Delta'}{\Gamma, \Gamma \rightarrow \Delta, (A \Rightarrow B), \Delta'},$$  

$$(\Rightarrow \rightarrow) \quad \frac{\Gamma', \Gamma \rightarrow \Delta, A, \Delta'; \Gamma', B, \Gamma \rightarrow \Delta, \Delta'}{\Gamma, (A \Rightarrow B), \Gamma \rightarrow \Delta, \Delta'}.$$  

**Negation rules**

$$(\neg \rightarrow) \quad \frac{\Gamma', \Gamma \rightarrow \Delta, A, \Delta'}{\Gamma', \neg A, \Gamma \rightarrow \Delta, \Delta'},$$  

$$(\rightarrow \neg) \quad \frac{\Gamma', A, \Gamma \rightarrow \Delta, \Delta'}{\Gamma, \Gamma \rightarrow \Delta, \neg A, \Delta'}.$$  

Formally we define:

$$\text{GL} = (\mathcal{L}, SQ, LA, (\cup), (\neg \cup), (\cap), (\neg \cap), (\Rightarrow), (\neg \Rightarrow), (\neg \neg)), \quad (23)$$

where $SQ = \{ \Gamma \rightarrow \Delta : \Gamma, \Delta \in \mathcal{F}^* \}$, $(\cup), (\neg \cup), (\cap), (\neg \cap), (\Rightarrow), (\neg \Rightarrow), (\neg \neg)$ are the inference rules defined above and $AL$ are the logical axioms of the system defined by the schema (21).

We define the notion of a bf formal proof in $\text{GL}$ as in any proof system, i.e., by a formal proof of a sequent $\Gamma \rightarrow \Delta$ in the proof system $\text{GL}$ we understand any sequence

$$\Gamma_1 \rightarrow \Delta_1, \Gamma_2 \rightarrow \Delta_2, \ldots, \Gamma_n \rightarrow \Delta_n.$$
of sequents, such that $\Gamma_1 \rightarrow \Delta_1 \in AL$, $\Gamma_n \rightarrow \Delta_n = \Gamma \rightarrow \Delta$, and for all $i$ $(1 < i \leq n)$ $\Gamma_i \rightarrow \Delta_i \in AL$, or $\Gamma_i \rightarrow \Delta_i$ is a conclusion of one of the inference rules of $GL$ with all its premisses placed in the sequence $\Gamma_1 \rightarrow \Delta_1$, $\ldots \Gamma_{i-1} \rightarrow \Delta_{i-1}$.

We write, as usual, $\vdash_{GL} \Gamma \rightarrow \Delta$ to denote that $\Gamma \rightarrow \Delta$ has a formal proof in $GL$, or we write simply $\vdash \Gamma \rightarrow \Delta$ when the system $GL$ is fixed.

We say that a formula $A \in \mathcal{F}$, has a proof in $GL$ and denote it by $\vdash_{GL} A$ if the sequent $\rightarrow A$ has a proof in $GL$, i.e. we define:

$$\vdash_{GL} A \text{ if and only if } \vdash_{GL} \rightarrow A. \quad (24)$$

We write, however, the formal proofs in $GL$ in a form of proof trees rather than in a form of sequences of sequents.

**Proof trees**

A proof tree $\Gamma \rightarrow \Delta$ is a tree $T_{\Gamma \rightarrow \Delta}$ satisfying the following conditions:

1. The topmost sequent, i.e. the root of $T_{\Gamma \rightarrow \Delta}$ is $\Gamma \rightarrow \Delta$.
2. All leaves are axioms.
3. The nodes are sequents such that each sequent on the tree follows from the ones immediately preceding it by one of the rules.

We picture, and write our proof-trees, as we did in case of $RS$ type systems, with the node on the top, and leafs on the very bottom, instead of more common way, where the leaves are on the top and root is on the bottom of the tree. We also write the proof- trees indicating additionally the name of the inference rule used at each step of the proof.

Here is a tree- proof of the de Morgan law $(\neg(a \cap b) \Rightarrow (\neg a \cup \neg b))$.

\[
\rightarrow (\neg(a \cap b) \Rightarrow (\neg a \cup \neg b)) \\
\quad | (\rightarrow \Rightarrow) \\
\neg(a \cap b) \rightarrow (\neg a \cup \neg b) \\
\quad | (\rightarrow \cup) \\
\neg(a \cap b) \rightarrow \neg a, \neg b \\
\quad | (\rightarrow \neg) \\
b, \neg(a \cap b) \rightarrow \neg a
\]
Remark 1

The proof search in $\text{GL}$ (to be defined by the decomposition tree) results are not always unique; one formula (sequent) can have many proofs.

Here is another proof in $\text{GL}$ of the de Morgan Law.

\[
\begin{array}{c}
\vdash (\to \neg) \\
b, a, \neg(a \cap b) \to \\
\ | (\neg \to) \\
b, a \to (a \cap b) \\
\bigwedge (\to \cap) \\
\end{array}
\]

\[
\begin{array}{c}
b, a \to a \\
\end{array}
\]

\[
\begin{array}{c}
b, a \to b \\
\end{array}
\]
The process of searching for proofs of a formula $A$ in $\text{GL}$ consists, as in the $\text{RS}$ type systems, of building decomposition trees. Their construction is similar to the one defined for $\text{RS}$ type systems and is described intuitively as follows.

We take a root of a decomposition tree $T_A$ a sequent $\rightarrow A$. For each node, if there is a rule of $\text{GL}$ which conclusion has the same form as the node sequent, then the node has children that are premises of the rule. If the node consists only of an indecomposable sequent (built out of variables only), then it does not have any children. This is a termination condition for the decomposition tree.

We prove that each formula $A$ generates a finite set $T_A$ of decomposition trees, such that the following holds. If there exist a tree $T_A \in T_A$ whose all leaves are axioms, then tree $T_A$ constitutes a proof of $A$ in $\text{GL}$. If all trees in $T_A$ have at least one non-axiom leaf, the proof of $A$ does not exist.

The first step in formally defining a notion of a decomposition tree consists of transforming the inference rules of $\text{GL}$, as we did in the case of the $\text{RS}$ type systems, into corresponding decomposition rules.

**Decomposition rules of GL**

Building a proof search decomposition tree consists of using the inference rules in an inverse order; we transform the inference rules into decomposition rules by reversing the role of conclusion and its premisses. We call such rules the decomposition rules. Here are all of $\text{GL}$ decomposition rules.

**Conjunction decomposition rules**

\[
\begin{array}{c}
\Gamma, (A \land B), \Gamma \rightarrow \Delta' \\
\Gamma, A, B, \Gamma \rightarrow \Delta \\
\end{array}
\]

**Disjunction decomposition rules**

\[
\begin{array}{c}
\Gamma \rightarrow \Delta, (A \lor B), \Delta' \\
\Gamma, A, B, \Gamma \rightarrow \Delta \\
\end{array}
\]

**Implication decomposition rules**

\[
\begin{array}{c}
\Gamma', \Gamma \rightarrow \Delta, (A \Rightarrow B), \Delta' \\
\Gamma', A, \Gamma \rightarrow \Delta, B, \Delta' \\
\end{array}
\]

**Negation decomposition rules**

\[
\begin{array}{c}
\Gamma', (A \Rightarrow B), \Gamma \rightarrow \Delta, \Delta' \\
\Gamma', A, \Gamma \rightarrow \Delta, B, \Delta' \\
\end{array}
\]
We write the decomposition rules in a visual tree form as follows.

\((\to \cup)\) rule

\[
\frac{\Gamma', \neg A, \Gamma \to \Delta, \Delta'}{\Gamma, \Gamma \to \Delta, \Delta'} \quad \text{ (\to \neg)} \quad \frac{\Gamma', \Gamma \to \Delta, \neg A, \Delta'}{\Gamma', A, \Gamma \to \Delta, \Delta'}
\]

\((\to \cup)\) rule

\[
\Gamma \to \Delta, (A \cup B), \Delta' \\
\mid (\to \cup) \\
\Gamma \to \Delta, A, B, \Delta'
\]

\((\cup \to)\) rule

\[
\frac{\Gamma', (A \cup B), \Gamma \to \Delta'}{\bigwedge (\cup \to)}
\]

\[
\Gamma', A, \Gamma \to \Delta' \\
\Gamma', B, \Gamma \to \Delta'
\]

\((\to \cap)\) rule

\[
\frac{\Gamma \to \Delta, (A \cap B), \Delta'}{\bigwedge (\to \cap)}
\]

\[
\Gamma \to \Delta, A, \Delta' \\
\Gamma \to \Delta, B, \Delta'
\]

\((\cap \to)\) rule

\[
\frac{\Gamma', (A \cap B), \Gamma \to \Delta'}{\bigwedge (\cap \to)}
\]

\[
\Gamma', A, B, \Gamma \to \Delta'
\]

\((\to \Rightarrow)\) rule

\[
\frac{\Gamma', \Gamma \to \Delta, (A \Rightarrow B), \Delta'}{\bigwedge (\to \Rightarrow)}
\]

\[
\Gamma', A, \Gamma \to \Delta, B, \Delta'
\]

\((\Rightarrow \rightarrow)\) rule

39
\[ \Gamma', (A \Rightarrow B), \Gamma \rightarrow \Delta, \Delta' \]

\[ \wedge(\Rightarrow\rightarrow) \]

\[ \Gamma', \Gamma \rightarrow \Delta, A, \Delta' \quad \Gamma', B, \Gamma \rightarrow \Delta, \Delta' \]

(\(\neg \rightarrow\)) rule

\[ \Gamma', \neg A, \Gamma \rightarrow \Delta, \Delta' \]

\[ | (\neg \rightarrow) \]

\[ \Gamma', \Gamma \rightarrow \Delta, A, \Delta' \]

(\(\rightarrow \neg\)) rule

\[ \Gamma', \Gamma \rightarrow \Delta, \neg A, \Delta' \]

\[ | (\neg \rightarrow) \]

\[ \Gamma', A, \Gamma \rightarrow \Delta, \Delta' \]

Observe that we use the same names for the inference and decomposition rules, as once the we have built a decomposition tree (with use of the decomposition rules) with all leaves being axioms, it constitutes a proof of \(A\) in \(GL\) with branches labeled by the proper inference rules.

We have already defined (definition 8) indecomposable sequence as any sequence \(\Gamma' \rightarrow \Delta\) when \(\Gamma', \Delta \in VAR^\ast\). In particular, a formula that is not a positive literal (propositional variable) is called a decomposable formula, and a sequent \(\Gamma \rightarrow \Delta\) where either \(\Gamma\) or \(\Delta\) contains a decomposable formula is called a decomposable sequent.

By inspecting the domain of the rules we can see that at most two rules could apply for any given decomposable sequent \(\Gamma \rightarrow \Delta\).

For any decomposable sequent, at most two decomposition rules can be applied to it. This rule is determined by the first decomposable formula in \(\Gamma\) when we traverse it from left to right, and by the main connective of that formula, or by the first decomposable formula in \(\Delta\) when we traverse it from the right to left, and by the main connective of that formula. We hence are now ready to define a decomposition tree.
**Decomposition Tree** $T_{\rightarrow A}$

For each formula $A \in F$, a decomposition tree $T_{\rightarrow A}$ is a tree built as follows.

**Step 1.** The sequent $\rightarrow A$ is the root of $T_{\rightarrow A}$ and for any node $\Gamma \rightarrow \Delta$ of the tree we follow the steps below.

**Step 2.** If $\Gamma \rightarrow \Delta$ is indecomposable, then $\Gamma \rightarrow \Delta$ becomes a leaf of the tree.

**Step 3.** If $\Gamma \rightarrow \Delta$ is decomposable, then we pick a decomposition rule that applies by matching the sequent of the current node with the domain of the decomposition rule. To do so we proceed as follows.

1. We traverse $\Gamma$ from left to right to find the first decomposable formula. Its main connective $\circ$ identifies a possible decomposition rule ($\circ \rightarrow$). Then we check if this decomposition rule applies. If it does we put its conclusions (conclusion) as leaves (leaf).

2. We traverse $\Delta$ from right to left to find the first decomposable formula. Its main connective $\circ$ identifies a possible decomposition rule ($\rightarrow \circ$). Then we check if this decomposition rule applies. If it does we put its conclusions (conclusion) as leaves (leaf).

3. If 1. and 2. apply we choose one of the rules.

**Step 4.** We repeat steps 2 and 3 until we obtain only leaves.

**Observation 1**

The decomposable $\Gamma \rightarrow \Delta$ is always in the domain in one of the decomposition rules ($\circ \rightarrow$), ($\rightarrow \circ$), or in the domain of both. Hence the tree $T_{\rightarrow A}$ may not be unique and all possible choices of 3. give all possible decomposition trees.

We generalize the definition of $T_{\rightarrow A}$ to the decomposition tree $T_{\Sigma \rightarrow \Lambda}$ of any sequent $\Sigma \rightarrow \Lambda \in SQ$ as follows.

**Decomposition Tree** $T_{\Sigma \rightarrow \Lambda}$

**Step 1.** The sequent $\Sigma \rightarrow \Lambda$ is the root of $T_{\Sigma \rightarrow \Lambda}$, and for any node $\Gamma \rightarrow \Delta$ of the tree we follow the steps below.

**Step 2.** If $\Gamma \rightarrow \Delta$ is indecomposable, then $\Gamma \rightarrow \Delta$ becomes a leaf of the tree.

**Step 3.** and **Step 4.** are the same as in the above definition of the tree $T_{\rightarrow A}$.

**Exercise 14**

Prove, by constructing a proper decomposition tree that

\[ \vdash_{GL} (\neg a \Rightarrow b) \Rightarrow (\neg b \Rightarrow a) \].

41
Solution
By definition, we have that

\( \vdash_{\text{GL}} (\neg a \Rightarrow b) \Rightarrow (\neg b \Rightarrow a) \) if and only if \( \vdash_{\text{GL}} \rightarrow ((\neg a \Rightarrow b) \Rightarrow (\neg b \Rightarrow a)) \).

We construct a decomposition tree as follows.

\[
\begin{array}{c}
T_{\rightarrow A} \\
\rightarrow ((\neg a \Rightarrow b) \Rightarrow (\neg b \Rightarrow a)) \\
\mid (\Rightarrow) \\
(\neg a \Rightarrow b) \rightarrow (\neg b \Rightarrow a) \\
\mid (\Rightarrow) \\
\neg b, (\neg a \Rightarrow b) \rightarrow a \\
\mid (\neg) \\
(\neg a \Rightarrow b) \rightarrow b, a \\
\bigwedge (\Rightarrow) \\
\rightarrow \neg b, b, a \\
\mid (\neg) \\
b \rightarrow b, a \\
\text{axiom} \\
b \rightarrow b, a \\
\text{axiom}
\end{array}
\]

All leaves of the tree are axioms, hence it constitutes a proof in GL.

Exercise 15
Prove, by constructing proper decomposition trees that

\( \not\vdash_{\text{GL}} ((a \Rightarrow b) \Rightarrow (\neg b \Rightarrow a)) \).

Solution
Observe that for some formulas \( A \), their decomposition tree \( T_{\rightarrow A} \) in GL may not be unique. Hence we have to construct all possible decomposition trees to see that none of them is a proof, i.e. to see that each of them has a non axiom leaf. We construct the decomposition trees for \( \rightarrow A \) as follows.

\[
\begin{array}{c}
T_{1 \rightarrow A} \\
\rightarrow ((a \Rightarrow b) \Rightarrow (\neg b \Rightarrow a)) \\
\text{(one choice)}
\end{array}
\]
$$(a \Rightarrow b) \implies (\neg b \Rightarrow a)$$

(first of two choices)

$$(\neg b, (a \Rightarrow b)) \implies a$$

(one choice)

$$(a \Rightarrow b) \implies b, a$$

(one choice)

\[ \land(\Rightarrow \Rightarrow) \]

\[ \rightarrow a, b, a \]

non-axiom

\[ b \rightarrow b, a \]

axiom

The tree contains a non-axiom leaf, hence it is **not a proof**. We have one more tree to construct.

**T2**

\[ \rightarrow ((a \Rightarrow b) \Rightarrow (\neg b \Rightarrow a)) \]

\[ \land(\Rightarrow \Rightarrow) \]

\[ \rightarrow (\neg b \Rightarrow a), a \]

non-axiom

\[ b \rightarrow (\neg b \Rightarrow a) \]

axiom

All possible trees end with a non-axiom leaf. It proves that

$$\not{\vdash}_{G\text{L}}((a \Rightarrow b) \Rightarrow (\neg b \Rightarrow a)).$$

**Exercise 16**

*Does the tree below constitute a proof in GL?*
Solution
The tree above is not a proof in GL because a decomposition rule used in the decomposition step below does not exists in GL

\((-a \Rightarrow b), \neg b \Rightarrow a\)

| (\(\neg \rightarrow\))

\((-a \Rightarrow b) \Rightarrow (\neg b \Rightarrow a)\) ---

\(\neg ((\neg a \Rightarrow b) \Rightarrow (\neg b \Rightarrow a))\)

| (\(\neg \rightarrow\))

\((-a \Rightarrow b) \Rightarrow (\neg b \Rightarrow a)\)

| (\(\neg \rightarrow\))

\((-a \Rightarrow b), \neg b \Rightarrow a\)

| (\(\neg \rightarrow\))

\((-a \Rightarrow b) \Rightarrow b, a\)

\(\wedge(\Rightarrow \rightarrow)\)

\(-a, b, a\) \hspace{1cm} \(b \Rightarrow b, a\)

| (\(\rightarrow \neg\))

\(a \Rightarrow b, a\)

axiom

It is a proof is some system GL1 that has all the rules of GL except its rule (\(\neg \rightarrow\))

\((-a \Rightarrow b), \neg b \Rightarrow a\)

| (\(\neg \rightarrow\))

\((-a \Rightarrow b) \Rightarrow b, a.\)

This rule has to be replaced in by the rule:

\((- \rightarrow)\)

\(\Gamma, \Gamma \rightarrow \Delta, A, \Delta'\)

\(\Gamma', \neg A, \Gamma \rightarrow \Delta, \Delta'\)
5 GL Soundness and Completeness

The system GL admits a constructive proof of the Completeness Theorem, similar to completeness proofs for RS type proof systems (Theorems 11, 6, 7). It also relies on strong soundness property of its inference rules. We are going to prove that the following holds.

**Theorem 8 (GL Strong Soundness)**

The proof system GL is strongly sound.

**Proof** We have already proved (Fact 4) that logical axioms of GL are tautologies, so we have to prove now that its rules of inference are strongly sound (definition 4). Proofs of strong soundness of rules of inference of GL are more involved than the proofs for the RS type rules. We prove as an example the strong soundness of four of inference rules. Proofs for all other rules follows the same patterns and is left as an exercise.

By definition 4 of strong soundness we have to show the condition (15). Written formally it says that we have to show that if \( P_1, P_2 \) are premisses of a given rule and \( C \) is its conclusion, then for all truth assignments \( v : VAR \rightarrow \{T,F\} \),

\[
v^*(P_1) = v^*(C) \quad \text{in case of one premiss rule, and}
\]

\[
v^*(P_1) \cap v^*(P_2) = v^*(C), \text{in case of a two premisses rule.}
\]

In order to prove it we need additional classical equivalencies listed below. You can find a list of most basic classical equivalences in Chapter 3.

\[
((A \Rightarrow B) \cap (A \Rightarrow C)) \equiv (A \Rightarrow (B \cap C))
\]

\[
((A \Rightarrow C) \cap (B \Rightarrow C)) \equiv ((A \cup B) \Rightarrow C)
\]

\[
((A \cap B) \Rightarrow C) \equiv (A \Rightarrow (\neg B \cup C))
\]

\[
\left(\neg \cap \rightarrow\right) \Gamma', A, B, \Gamma \rightarrow \Delta' \quad \frac{\Gamma', (A \cap B), \Gamma \rightarrow \Delta'}{\Gamma \rightarrow \Delta', (A \cap B), \Delta'}
\]

\[
v^*(\Gamma', A, B, \Gamma \rightarrow \Delta') = (v^*(\Gamma') \cap v^*(A) \cap v^*(B) \cap v^*(\Gamma)) \Rightarrow v^*(\Delta') = (v^*(\Gamma') \cap v^*(A \cap B) \cap v^*(\Gamma)) \Rightarrow v^*(\Delta') = v^*(\Gamma', (A \cap B), \Gamma \rightarrow \Delta')
\]

\[
\left(\neg \cap \rightarrow\right) \Gamma \rightarrow \Delta, A, \Delta' ; \Gamma \rightarrow \Delta, B, \Delta' \quad \frac{\Gamma \rightarrow \Delta, (A \cap B), \Delta'}{\Gamma \rightarrow \Delta, A, \Delta', B, \Delta'}
\]

\[
v^*(\Gamma \rightarrow \Delta, A, \Delta') \cap v^*(\Gamma \rightarrow \Delta, B, \Delta')
\]

\[
= (v^*(\Gamma) \Rightarrow v^*(\Delta) \cup v^*(A) \cup v^*(\Delta')) \cap (v^*(\Gamma) \Rightarrow v^*(\Delta) \cup v^*(B) \cup v^*(\Delta'))
\]
For any sequent \( \Gamma \rightarrow \Delta \in SQ \),
if \( \vdash_{GL} \Gamma \rightarrow \Delta \), then \( \models \Gamma \rightarrow \Delta \). In particular, for any \( A \in F \),
if \( \vdash_{GL} A \), then \( \models A \).

We know by theorem 8 that all the rules of inference of GL of are strongly sound. The strong soundness of the rules means that if at least one of premisses of a rule is false, so is its conclusion. Hence given a sequent \( \Gamma \rightarrow \Delta \in SQ \), such
that its decomposition tree $T_{\Gamma \rightarrow \Delta}$ has a branch ending with a non-axiom leaf. It means that any truth assignment $\nu$ that make this non-axiom leaf false also falsifies all sequences on that branch, and hence falsifies the sequent $\Gamma \rightarrow \Delta$. In particular, given a sequent $\rightarrow A$ and its tree $T_{\rightarrow A}$, any $\nu$, such that falsifies its a non-axiom leaf is a counter-model for $A$. We have hence proved the following.

**Theorem 10 (GL Counter Model)**

*Given a sequent $\Gamma \rightarrow \Delta$, such that its decomposition tree $T_{\Gamma \rightarrow \Delta}$ contains a non-axiom leaf $L_A$. Any truth assignment $\nu$ that falsifies the non-axiom leaf $L_A$ is a counter model for $\Gamma \rightarrow \Delta$. In particular, given a formula $A \in F$, and its decomposition tree $T_A$ with a non-axiom leaf, this leaf let us define a counter-model for $A$ determined by the decomposition tree $T_A$.*

Here is a simple exercise explaining how the construction of a counter-model determined by the decomposition tree of a works.

**Exercise 17**

*Prove, by constructing a counter-model determined by decomposition tree that $\not\models ((b \Rightarrow a) \Rightarrow (\neg b \Rightarrow a))$.*

**Solution**

We construct the decomposition tree for the formula $A : ((b \Rightarrow a) \Rightarrow (\neg b \Rightarrow a))$ as follows.

$$
\begin{align*}
T_{\rightarrow A} \\
\rightarrow ((b \Rightarrow a) \Rightarrow (\neg b \Rightarrow a)) \\
\mid (\rightarrow \Rightarrow) \\
(b \Rightarrow a) \rightarrow (\neg b \Rightarrow a) \\
\mid (\rightarrow \Rightarrow) \\
\neg b, (b \Rightarrow a) \rightarrow a \\
\mid (\neg \rightarrow) \\
(b \Rightarrow a) \rightarrow b, a \\
\wedge (\Rightarrow \rightarrow) \\
\rightarrow b, b, a \\
\rightarrow a \rightarrow b, a \\
\text{non-axiom} \\
\text{axiom}
\end{align*}
$$
The non-axiom leaf $L_A$ we want to falsify is $\rightarrow b, b, a$. Let $v : VAR \rightarrow \{T, F\}$ be a truth assignment. By definition 7 of semantic for $\text{GL}$ we have that $v^*(L_A) = v^*(\rightarrow b, b, a) = (T \Rightarrow v(b) \cup v(b) \cup v(a))$. Hence $v^*(\rightarrow b, b, a) = F$ if and only if $(T \Rightarrow v(b) \cup v(b) \cup v(a)) = F$ if and only if $v(b) = v(a) = F$.

The Theorem 10, says that the logical value $F$ determined by the evaluation a non-axiom leaf $L_A$ "climbs" the decomposition tree. We picture it as follows.

$$
\begin{array}{c}
T \rightarrow A \\
\rightarrow ((b \Rightarrow a) \Rightarrow (\neg b \Rightarrow a)) \ F \\
| (\Rightarrow \Rightarrow) \\
(b \Rightarrow a) \rightarrow (\neg b \Rightarrow a) \ F \\
| (\neg \rightarrow) \\
\neg b, (b \Rightarrow a) \rightarrow a \ F \\
| (\neg \Rightarrow) \\
(b \Rightarrow a) \rightarrow b, a \ F \\
\bigwedge (\Rightarrow \rightarrow)
\end{array}
$$

So, by theorem 10, any truth assignment $v : VAR \rightarrow \{T, F\}$, such that $v(b) = v(a) = F$ falsifies the sequence $\rightarrow A$, i.e. $v^*(\rightarrow A) = T \Rightarrow v^*(A) = F$. This is possible only if $v^*(A) = F$. This proves that $v$ is a counter model for $A$ and we proved that $\not\models A$.

Our main goal is to prove the Completeness Theorem for $\text{RS}$. We prove first the Completeness Theorem for formulas $A \in F$ and then we generalize it to any sequences $\Gamma \in F^*$.

**Theorem 11 (Completeness Theorem)**

For any formula $A \in F$,

$$
\vdash_{\text{GL}} A \quad \text{if and only if} \quad \models A.
$$

For any sequent $\Gamma \rightarrow \Delta \in SQ$,

$$
\vdash_{\text{GL}} \Gamma \rightarrow \Delta \quad \text{if and only if} \quad \models \Gamma \rightarrow \Delta.
$$
Proof

We have already proved the Soundness Theorem 9, so we need to prove only
the completeness part of it, namely to prove the implication:

\[ \text{if } \models A, \text{ then } \vdash_{\text{GL}} A. \]  \hfill (25)

We prove instead of the logically equivalent opposite implication:

\[ \text{if } \neg \vdash_{\text{GL}} A \text{ then } \neg \models A. \]  \hfill (26)

Assume \( \neg \vdash_{\text{GL}} A \). By (24) it means that \( \neg \vdash_{\text{GL}} \Gamma \rightarrow A \). Let \( T_\rightarrow \) be a set of all
decomposition trees of \( \rightarrow A \). As \( \neg \vdash_{\text{GL}} \rightarrow A \), each tree \( T_\rightarrow A \) in the set \( T_\rightarrow \) has a
non-axiom leaf. We choose an arbitrary \( T_\rightarrow A \) from \( T_\rightarrow \). Let \( L_A = \Gamma \rightarrow \Delta' \), be
a non-axiom leaf of \( T_\rightarrow A \). We define a truth assignment \( v : \text{VAR} \rightarrow \{T,F\} \)
which falsifies \( \Gamma \rightarrow \Delta' \) as follows.

\[
v(a) = \begin{cases} 
T & \text{if } a \text{ appears in } \Gamma' \\
F & \text{if } a \text{ appears in } \Delta' \\
\text{any value} & \text{if } a \text{ does not appear in } \Gamma' \rightarrow \Delta'
\end{cases}
\]

By the strong soundness of the rules of inference of \( \text{GL} \) and Theorem 10 it
proves that \( v^* (\rightarrow A) = F \), i.e. that \( \neg \models A \) and hence \( \neg \models A \).

Assume that \( \Gamma \rightarrow \Delta \) is any sequence such that \( \neg \vdash_{\text{GL}} \Gamma \rightarrow \Delta \). But \( \vdash_{\text{GL}} \Gamma \rightarrow \Delta \) if and only if \( \vdash_{\text{GL}} (\sigma_\Gamma \Rightarrow \delta_\Delta) \). So \( \neg \vdash_{\text{GL}} \Gamma \rightarrow \Delta \) if and only if \( \neg \vdash_{\text{GL}} (\sigma_\Gamma \Rightarrow \delta_\Delta) \). By already proven Case 1, \( \neg \models (\sigma_\Gamma \Rightarrow \delta_\Delta) \), what is obviously equivalent to
\( \neg \models \Gamma \rightarrow \Delta \). This ends the proof of Case 2 and Completeness Theorem.

Gentzen Sequent Proof System G

The proof system \( \text{G} \) is in its structure the most similar to the proof system \( \text{RS2} \)
defined by (19).

It is obtained from in the same way is a proof system obtained from \( \text{GL} \) by
changing the indecomposable sequences \( \Gamma', \Delta' \) into any sequences \( \Sigma, \Lambda \in \mathcal{F}^* \)
in all of the rules of inference of \( \text{GL} \).

The logical axioms \( \text{LA} \) remain the same; i.e. the components of \( \text{G} \) are as
follows.

Axioms of G

As the axioms of \( \text{GL} \) we adopt any indecomposable sequent which contains a
positive literal \( a \) (variable) that appears on both sides of the sequent arrow \( \rightarrow \),
i.e any sequent of the form

\[ \Gamma'_1, a, \Gamma'_2 \rightarrow \Delta'_1, a, \Delta'_2. \]  \hfill (28)

for any \( a \in \text{VAR} \) and any sequences \( \Gamma'_1, \Gamma'_2, \Delta'_1, \Delta'_2 \in \text{VAR}^* \).
We adopt the following rules of inference.

**Conjunction rules**

\[(\land \rightarrow) \quad \Sigma, A, B, \Gamma \rightarrow \Lambda \quad \therefore \quad \Sigma, (A \land B), \Gamma \rightarrow \Lambda,\]

\[(\rightarrow \land) \quad \Gamma \rightarrow \Delta, A, \Lambda ; \Gamma \rightarrow \Delta, B, \Lambda \quad \therefore \quad \Gamma \rightarrow \Delta, (A \land B), \Lambda,\]

**Disjunction rules**

\[(\rightarrow \lor) \quad \Gamma \rightarrow \Delta, A, B, \Lambda \quad \therefore \quad \Gamma \rightarrow \Delta, (A \lor B), \Lambda,\]

\[(\lor \rightarrow) \quad \Sigma, A, \Gamma \rightarrow \Lambda ; \Sigma, B, \Gamma \rightarrow \Lambda \quad \therefore \quad \Sigma, (A \lor B), \Gamma \rightarrow \Lambda,\]

**Implication rules**

\[(\rightarrow \Rightarrow) \quad \Sigma, A, \Gamma \rightarrow \Delta, B, \Lambda \quad \therefore \quad \Sigma, \Gamma \rightarrow \Delta, (A \Rightarrow B), \Lambda,\]

\[(\Rightarrow \rightarrow) \quad \Sigma, \Gamma \rightarrow \Delta, A, \Lambda ; \Sigma, B, \Gamma \rightarrow \Delta, \Lambda \quad \therefore \quad \Sigma, (A \Rightarrow B), \Gamma \rightarrow \Delta, \Lambda,\]

**Negation rules**

\[(\neg \rightarrow) \quad \Sigma, \Gamma \rightarrow \Delta, A, \Lambda \quad \therefore \quad \Sigma, \neg A, \Gamma \rightarrow \Delta, \Lambda,\]

\[(\rightarrow \neg) \quad \Sigma, A, \Gamma \rightarrow \Delta, \Lambda \quad \therefore \quad \Sigma, \Gamma \rightarrow \Delta, \neg A, \Lambda,\]

where \(\Gamma, \Delta, \Sigma, \Lambda \in \mathcal{F}^*\).

**Exercise 18**  Follow the example of the \(\text{GL} \) system and adopt all needed definitions and proofs to prove the completeness of the system \(G\).

**Solution**

We leave it to the reader to fill in details. In particular, one has to accomplish the steps below.

1. Explain why the system \(G\) is strongly sound. You can use the strong soundness of the system \(\text{GL}\).
2. Prove, as an example, a strong soundness of 4 rules of \(G\).
3. Prove the following Strong Soundness Theorem for \(G\).
Theorem 12

*The proof system $G$ is strongly sound.*

4. Define shortly, in your own words, for any formula $A \in \mathcal{F}$, its decomposition tree $T_{A}$ in $G$.

5. Extend your definition to a decomposition tree $T_{\Gamma \rightarrow \Delta}$.

6. Prove that for any $\Gamma \rightarrow \Delta \in \text{SQ}$, the decomposition tree $T_{\Gamma \rightarrow \Delta}$ are finite.

7. Give an example of formulas $A, B \in \mathcal{F}$ such that that $T_{\rightarrow A}$ is unique and $T_{\rightarrow B}$ is not.

8. Prove the following Counter Model Theorem for $G$.

**Theorem 13**

*Given a sequent $\Gamma \rightarrow \Delta$, such that its decomposition tree $T_{\Gamma \rightarrow \Delta}$ contains a non-axiom leaf $L_{A}$. Any truth assignment $v$ that falsifies the non-axiom leaf $L_{A}$ is a counter model for $\Gamma \rightarrow \Delta$.*

10. Prove the following Completeness Theorem for $G$.

**Theorem 14**

*For any formula $A \in \mathcal{F}$,
1. $\vdash_{G} A$ if and only if $\models A$, and for any sequent $\Gamma \rightarrow \Delta \in \text{SQ},$
2. $\vdash_{G} \Gamma \rightarrow \Delta$ if and only if $\models \Gamma \rightarrow \Delta$."

6. **Original Gentzen System LK**

   **Proofs of Completeness and Hauptzatz Theorems**

The original systems LK and LI were created by Gentzen in 1935 for classical and intuitionistic predicate logics, respectively. We present here only classical propositional version and call it also LK. The proof system LI for intuitionistic propositional logic is a particular case of the proof system LK and is presented and discussed in chapter ???. The full predicate logics versions of of both LK and LI follow in chapter ???.

Both systems LK and LI have two groups of inference rules and a special rule called a *cut rule*. One group consists of a set of rules similar to the rules of systems GL and G. We call them Logical Rules. The other group contains a new type of rules, called Structural Rules. The cut rule in Gentzen sequent systems corresponds to the Modus Ponens rule in Hilbert proof systems as
Modus Ponens is a particular case of the cut rule. The cut rule is needed to carry
the original Gentzen proof of the completeness of the system LK and proving
the adequacy of LI system for intuitionistic logic. We talk about adequacy, not
a completeness of LI, as at that time intuitionistic logic was defined as a proof
system only with its semantics to be invented 10 years later. Gentzen proof of
completeness of LK was not direct. He used the completeness of already known
Hilbert proof system H and proved that any formula provable in the system
H is also provable in LK (hence the need of the cut rule). By completeness
of H, any tautology is provable in H. Hence any tautology is also provable in
LK. The soundness is LK holds by direct evaluation. Observe that by presence
of the cut rule, Gentzen LK is also a Hilbert system. What distinguishes it
from all other known Hilbert proof systems is the fact that the cut rule could
be eliminated from it. This is Gentzen famous Hauptzatz Theorem, also called
Cut Elimination Theorem. The elimination of the cut rule and the structure of
other rules makes it possible to define effective automatic procedures for proof
search, what is impossible in a case of the Hilbert style systems.

Gentzen, in his proof of Hauptzatz Theorem, developed a powerful technique
adaptable to other logics. We present it here in classical propositional case and
show how to adapt it to the intuitionistic case in chapter ???. The predicate case
is a straightforward extension of the propositional and will be discussed follows
in chapter ???. Gentzen proof is purely syntactical. It defines a constructive
method of transformation of any formal proof (derivation) of a sequent \( \Gamma \rightarrow \Delta \)
that uses a cut rule (and other rules) into its proof without use of the cut rule.
Hence the English name Cut Elimination Theorem.

Rasiowa and Sikorski method of proving completeness theorem by constructing
counter-models on the decomposition trees is a semantical equivalence to purely
syntactical Gentzen proof of cut elimination. It is relatively simple, powerful
and easy to understand. It was the reason it was first to be presented here. But
it is more difficult and sometimes impossible to apply (generalize) to many non-
classical logics then Gentzen cut elimination method. Moreover the Gentzen
method is more flexible and in this sense more general and powerful. This is
why we preset it here.

Components of LK

Language \( \mathcal{L} \)
The language is the same as the in case of GL, namely
\[
\mathcal{L} = \mathcal{L}_{\{\neg,\land,\lor,\Rightarrow\}}.
\]

Expressions
The set of all expressions \( \mathcal{E} \) is, as before, the set
\[
SQ = \{\Gamma \rightarrow \Delta : \Gamma, \Delta \in \mathcal{F}^*\}
\]
of all sequents.
**Logical Axioms**
There is only one logical axiom, namely

\[ A \rightarrow A, \]

where \( A \) is any formula of \( L \).

**Rules of Inference**
There are two groups of rules of inference and they are defined as follows.

GROUP ONE: STRUCTURAL RULES.

**Weakening** in the antecedent

\[
(\text{weak} \rightarrow) \quad \frac{\Gamma \rightarrow \Delta}{A, \Gamma \rightarrow \Delta'},
\]

**Weakening** in the succedent

\[
(\rightarrow \text{weak}) \quad \frac{\Gamma \rightarrow \Delta}{\Gamma \rightarrow \Delta', A},
\]

**Contraction** in the antecedent

\[
(\text{contr} \rightarrow) \quad \frac{A, A, \Gamma \rightarrow \Delta}{A, \Gamma \rightarrow \Delta},
\]

**Contraction** in the succedent

\[
(\rightarrow \text{contr}) \quad \frac{\Gamma \rightarrow \Delta, A, A}{\Gamma \rightarrow \Delta, A'},
\]

**Exchange** in the antecedent

\[
(\text{exch} \rightarrow) \quad \frac{\Gamma_1, A, B, \Gamma_2 \rightarrow \Delta}{\Gamma_1, B, A, \Gamma_2 \rightarrow \Delta},
\]

**Exchange** in the succedent

\[
(\rightarrow \text{exch}) \quad \frac{\Delta \rightarrow \Gamma_1, A, B, \Gamma_2}{\Delta \rightarrow \Gamma_1, B, A, \Gamma_2},
\]

**Cut Rule**

\[
(\text{cut}) \quad \frac{\Gamma \rightarrow \Delta, A ; A, \Sigma \rightarrow \Theta}{\Gamma, \Sigma \rightarrow \Delta, \Theta}.
\]

The formula \( A \) is called a *cut formula.*
GROUP TWO: LOGICAL RULES

Conjunction

\[
\begin{align*}
(\cap \rightarrow)_1 & : A, \Gamma \rightarrow \Delta \quad (A \cap B), \Gamma \rightarrow \Delta' \\
(\cap \rightarrow)_2 & : B, \Gamma \rightarrow \Delta \quad (A \cap B), \Gamma \rightarrow \Delta'
\end{align*}
\]

\[\rightarrow \cap\]
\[
\frac{\Gamma \rightarrow \Delta, A ; \Gamma \rightarrow \Delta, B}{\Gamma \rightarrow \Delta, (A \cap B)}.
\]

Disjunction

\[
\begin{align*}
(\rightarrow \cup)_1 & : \Gamma \rightarrow \Delta, A \\
\Gamma \rightarrow \Delta, (A \cup B)
\end{align*}
\]

\[\rightarrow \cup_1\]
\[
\frac{\Gamma \rightarrow \Delta, B}{\Gamma \rightarrow \Delta, (A \cup B)}.
\]

\[\cup \rightarrow\]
\[
\frac{A, \Gamma \rightarrow \Delta ; B, \Gamma \rightarrow \Delta}{(A \cup B), \Gamma \rightarrow \Delta}.
\]

Implication

\[
\begin{align*}
(\rightarrow \Rightarrow) & : A, \Gamma \rightarrow \Delta, B \\
\Gamma \rightarrow \Delta, (A \Rightarrow B)
\end{align*}
\]

\[\rightarrow \Rightarrow\]
\[
\frac{\Gamma \rightarrow \Delta, A ; B, \Gamma \rightarrow \Delta}{(A \Rightarrow B), \Gamma \rightarrow \Delta}.
\]

Negation

\[
\begin{align*}
(\rightarrow \neg) & : \Gamma \rightarrow \Delta, A \\
\neg A, \Gamma \rightarrow \Delta
\end{align*}
\]

\[\rightarrow \neg\]
\[
\frac{A, \Gamma \rightarrow \Delta}{\Gamma \rightarrow \Delta, \neg A}.
\]

Definition 9

The Gentzen system \( LK \) as

\[
LK = (L, SQ, AL, \text{ Structural Rules, Cut Rule, Logical Rules}),
\]

where all the components are defined by (30) above.

We say that a formula \( A \in \mathcal{F} \), has a proof in \( LK \) and denote it by \( \vdash_{LK} A \) if the sequent \( \rightarrow A \) has a proof in \( GL \), i.e. we define:

\[\vdash_{LK} A \text{ if and only if } \vdash_{LK} \rightarrow A.\] (31)

Proof Trees

We write formal proofs in \( LK \), as we did for other Gentzen style proof systems in a form of trees in an "upside - down" form.
By a **proof tree** of a sequent $\Gamma \rightarrow \Delta$ in $\textbf{LK}$ we understand a tree
\[
D_{\Gamma \rightarrow \Delta}
\]
satisfying the following conditions:

1. The topmost sequent, i.e the **root** of $D_{\Gamma \rightarrow \Delta}$ is $\Gamma \rightarrow \Delta$.
2. All **leaves** are axioms.
3. The **nodes** are sequents such that each sequent on the tree follows from the ones immediately preceding it by one of the rules.

The proofs are often called **derivations**. In particular, Gentzen, in his work used the term derivation we will use this notion as well. This is why we denote the proof trees by $D$ (for derivation).

Finding derivations $D$ in $\textbf{LK}$ are is a more complex process, as the logical rules are different, then in $\textbf{GL}$ and $\textbf{G}$. Proofs rely strongly on use of the Structural Rules. Even if we find a derivation that does not involve the Cut rule, the Structural rules are usually present. For example, a **derivation** of Excluded Middle ($A \cup \neg A$) formula $B$ in $\textbf{LK}$ is as follows.

\[
\begin{align*}
D \rightarrow & B \\
\rightarrow & (A \cup \neg A) \\
\mid & (\rightarrow \text{contr}) \\
\rightarrow & (A \cup \neg A), (A \cup \neg A) \\
\mid & (\rightarrow \cup)_1 \\
\rightarrow & (A \cup \neg A), A \\
\mid & (\rightarrow \text{exch}) \\
\rightarrow & A, (A \cup \neg A) \\
\mid & (\rightarrow \cup)_1 \\
\rightarrow & A, \neg A \\
\mid & (\rightarrow \neg) \\
A \rightarrow & A \\
\text{axiom}
\end{align*}
\]

Here is as yet another example a proof (also cut free) of the de Morgan Law ($\neg(A \cap B) \Rightarrow (\neg A \cup \neg B)$).
\[
\begin{align*}
P \rightarrow A \\
\rightarrow (\neg (A \cap B) \Rightarrow (\neg A \cup \neg B)) \\
\mid (\rightarrow \Rightarrow) \\
(\neg (A \cap B) \rightarrow (\neg A \cup \neg B)) \\
\mid (\rightarrow \neg) \\
\rightarrow (\neg A \cup \neg B), (A \cap B) \\
\wedge (\Rightarrow \rightarrow)
\end{align*}
\]

\[
\begin{align*}
\rightarrow (\neg A \cup \neg B), A & \quad \rightarrow (\neg A \cup \neg B), B \\
| (\rightarrow \text{exch}) & \quad | (\rightarrow \text{exch}) \\
\rightarrow A, (\neg A \cup \neg B) & \quad \rightarrow B, (\neg A \cup \neg B) \\
| (\rightarrow \cup)_1 & \quad | (\rightarrow \cup)_1 \\
\rightarrow A, \neg A & \quad \rightarrow B, \neg B \\
| (\rightarrow \neg) & \quad B \rightarrow B \\
A \rightarrow A & \quad \text{axiom} \\
\text{axiom}
\end{align*}
\]

Observe that the Logical Rules are similar in their structure to the rules of the system \(G\) and hence admit the same proof of their soundness.

The rules \((\rightarrow \cup)_1, (\rightarrow \cup)_2\) and \((\rightarrow \cup)_1, (\rightarrow \cup)_2\) are not strongly sound as \(A \not\equiv (A \cap B), B \not\equiv (A \cap B)\) and \(A \not\equiv (A \cap B), B \not\equiv (A \cap B)\).

All other Logical Rules are strongly sound.

The Contraction and Exchange structural are also strongly sound as for any formulas \(A, B \in \mathcal{F}, A \equiv (A \cap A), A \equiv (A \cup A)\) and \((A \cap B) \equiv (B \cap A), (A \cap B) \equiv (B \cap A)\). The Weakening rule is sound because (we use shorthand notation) if a formula \((\Gamma \Rightarrow \Delta) = T\) then also \(((A \cap \Gamma) \Rightarrow \Delta) = T\) for any logical value of the formula \(A\). But obviously \((\Gamma \Rightarrow \Delta) \not\equiv ((A \cap \Gamma) \Rightarrow \Delta)\), i.e. the Weakening rule is not strongly sound.

The Cut rule is sound as the fact \((\Gamma \Rightarrow (\Delta \cup A)) = T\) and \(((A \cap \Sigma) \Rightarrow \Lambda) = T\) implies that \((\Gamma, \Sigma \rightarrow \Delta, \Lambda)\). It is not strongly sound. Any truth assignment such that \(\Gamma = T, \Delta = \Sigma = \Lambda = A = F\) proves that \((\Gamma \rightarrow \Delta, A) \cap (A, \Sigma \rightarrow \Lambda) \not\equiv (\Gamma, \Sigma \rightarrow \Delta, \Lambda)\). Obviously, \(\models A \rightarrow A\).

We have proved that \(\text{LK}\) is sound and hence the following theorem holds.
Theorem 15 (Soundness for LK)

For any sequent \( \Gamma \rightarrow \Delta \),

\[
\text{if } \vdash_{\text{LK}} \Gamma \rightarrow \Delta, \text{ then } \models \Gamma \rightarrow \Delta.
\]

In particular, for any \( A \in \mathcal{F} \),

\[
\text{if } \vdash_{\text{LK}} A, \text{ then } \models A.
\]

We follow now Gentzen way of proving completeness of LK. We choose any complete Hilbert proof system for the LK language \( L = L_{\{\neg, \cap, \cup, \Rightarrow\}} \) and prove its equivalency with LK.

Gentzen referred to the Hilbert-Ackerman (1920) system (axiomatization) included in chapter ???. We choose here the Rasiowa-Sikorski (1952) formalization RS also included in chapter ???. We do it for two reasons. First, it reflexes a connection between classical and intuitionistic logics very much in a spirit Gentzen relationship between LK and LI. We obtain a complete proof system RSI from RS by just removing the last axiom A10. Second, both sets of axioms reflect the best what set of provable formulas is needed to conduct algebraic proofs of completeness of \( R \) and \( RI \), respectively.

Axioms of RS

The set of logical axioms of the Hilbert style proof system RS for classical propositional logic all formulas of the forms

A1 \( (A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C)) \),

A2 \( A \Rightarrow (A \cup B) \),

A3 \( B \Rightarrow (A \cup B) \),

A4 \( ((A \Rightarrow C) \Rightarrow ((B \Rightarrow C) \Rightarrow ((A \cup B) \Rightarrow C))) \),

A5 \( ((A \cap B) \Rightarrow A) \),

A6 \( ((A \cap B) \Rightarrow B) \),

A7 \( ((C \Rightarrow A) \Rightarrow ((C \Rightarrow B) \Rightarrow (C \Rightarrow (A \cap B)))) \),

A8 \( ((A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \cap B) \Rightarrow C)) \),

A9 \( (((A \cap B) \Rightarrow C) \Rightarrow (A \Rightarrow (B \Rightarrow C))) \),

A10 \( (A \cap \neg A) \Rightarrow B \),

A11 \( (A \Rightarrow (A \cap \neg A)) \Rightarrow \neg A \),

A12 \( (A \cup \neg A) \),
where $A, B, C \in \mathcal{F}$ are any formulas in $\mathcal{L} = \mathcal{L}_{\{\neg, \cap, \cup, \Rightarrow\}}$.

We adopt a Modus Ponens

\[(MP) \quad \frac{A; (A \Rightarrow B)}{B}\]

as the only inference rule. We define Hilbert System $R$ as

\[R = ( \mathcal{L}_{\{\neg, \cap, \cup, \Rightarrow\}}, \mathcal{F}, A1 - A12, (MP) ), \quad (33)\]

where $A1 - A12$ are defined by $(32)$.

The system $RS$ is complete, i.e. we have the following.

**Theorem 16**

For any formula $A \in \mathcal{F}$,

\[\vdash_{RS} A \quad if \ and \ only \ if \quad \models A.\]

We leave it as an exercise for the reader to show that all axioms $A1 - A12$ of the system $RS$ are provable in $LK$. Moreover, the Modus Ponens is a particular case of the cut rule, for $\Gamma, \Delta, \Sigma$ empty sequences and $\Theta$ containing only one element, a formula $B$. We call it also MP rule.

\[(MP) \quad \frac{\rightarrow A; A \rightarrow B}{\rightarrow B}.\]

This proves the following.

**Theorem 17**

For any formula $A \in \mathcal{F}$,

\[\vdash_{RS} A \quad if \ then \quad \vdash_{LK} A.\]

Directly from the above theorem 17, soundness of $LK$ (theorem 15) and completeness of $RS$ (theorem 16) we get the completeness of $LK$.

**Theorem 18 (LK Completeness)**

For any formula $A \in \mathcal{F}$,

\[\vdash_{LK} A \quad if \ and \ only \ if \quad \models A.\]

Here is Gentzen original formulation of the Hauptzatz Theorem, which we call also the Cut Elimination Theorem.
Theorem 19 (Hauptzatz)

Every derivation in LK can be transformed into another LK derivation of the same sequent, in which no cuts occur.

The proof is quite long and involved. We present here its main and most important steps. To facilitate the proof we introduce a more general form of the cut rule, called a mix rule defined as follows.

\[
\text{(mix)} \quad \frac{\Gamma \rightarrow \Delta ; \Sigma 
\rightarrow \Theta}{\Gamma, \Sigma^* \rightarrow \Delta^*, \Theta},
\]

where \(\Sigma^*, \Delta^*\) are obtained from \(\Sigma, \Delta\) by removing all occurrences of a common formula \(A\). The formula \(A\) is now called a mix formula.

Example 3

Here are some examples of an applications of the mix rule. Observe that the mix rule applies, as the cut does, to only one mix formula at the time.

\[
\text{(mix)} \quad a \rightarrow b, \neg a ; (b \cup c), b, b, D, b \rightarrow a, (b \cup c), D \rightarrow \neg a
\]

\(b\) is the mix formula.

\[
\text{(mix)} \quad A \rightarrow B, B, \neg A ; (b \cup c), B, B, D, B \rightarrow \neg B
\]

\(B\) is the mix formula.

\[
\text{(mix)} \quad A \rightarrow B, \neg A, \neg A ; \neg A, B, B \rightarrow \neg A, B, B, \neg A, B \rightarrow \neg B
\]

\(\neg A\) is the mix formula.

Notice, that every derivation with cut may be transformed into a derivation with mix by means of a number of weakenings (multiple application of the weakening rules) and interchanges (multiple application of the exchange rules). Conversely, every mix may be transformed into a cut derivation by means of a certain number of preceding exchanges and contractions, though we do not use this fact in the proof. Observe that cut is a particular case of mix.

Proof of Hauptzatz Theorem

We conduct the proof in three main steps.

Step 1: we consider only derivations in which only mix rule is used.
Step 2: we consider first derivation with a certain Property H (definition 10) and prove lemma 2 for them. This lemma is the most crucial for the proof of the Hauptzatz.
Definition 10

We say that a derivation $D_{Γ → Δ}$ of a sequent $Γ → Δ$ has a Property H if it satisfies the following conditions.

1. The root $Γ → Δ$ of the derivation $D_{Γ → Δ}$ is obtained by direct use of the mix rule, i.e. the mix rule is the last rule of inference used in the proof (derivation) of $Γ → Δ$.

2. The derivation $D_{Γ → Δ}$ does not contain any other application of the mix rule, i.e. the proof (derivation) of $Γ → Δ$ does not contain any other application of the mix rule.

Lemma 2 (H lemma)

Any derivation that fulfills the Property H (definition 10) may be transformed into a derivation of the same sequent in which no mix occurs.

Step 3: we use the H lemma 2 and to prove the the Hauptzatz as follows.

Hauptzatz proof from H lemma

Let $D$ be any derivation (tree proof). Let $Γ → Δ$ be any node on $D$ such that its sub-tree $D_{Γ → Δ}$ has the Property H (definition 10). By H lemma 2 the sub-tree $D_{Γ → Δ}$ can be replaced by a tree $D^*_{Γ → Δ}$ in which no mix occurs. The rest of $D$ remains unchanged. We repeat this procedure for each node $N$, such that the sub-tree $D_N$ has the Property H until every application of mix rule has systematically been eliminated. This ends the proof of Hauptzatz provided the H lemma 2 has already been proved.

Step 2: proof of H lemma.

We now consider derivation tree $D$ with the Property H, i.e. such that the mix rule is the last rule of inference used, and $D$ does not contain any other application of the mix rule.

We define now two important notions: degree $n$ and rank $r$ of the derivation $D$. Observe that $D$ contains only one application of mix rule, and the mix rule, contains only one mix formula $A$. Mix rule used may contain many copies of $A$, but there always is only one mix formula. We call is a mix formula of $D$.

Definition 11

Given a derivation tree $D$ with the Property H.
Let $A ∈ F$ be the mix formula of $D$. The degree $n ≥ 0$ of $A$ is called the degree of the derivation $D$. We write it as $degD = degA = n$. 
Definition 12

Given a derivation tree $D$ with the Property $H$. We define the rank $r$ of $D$ as a sum of its left rank $L_r$ and right rank $R_r$ of $D$, i.e.

$$r = L_r + R_r,$$

where:

1. the left rank $L_r$ of $D$ in the largest number of consecutive nodes on the branch of $D$ staring with the node containing the left premiss of the mix rule, such that each sequent on these nodes contains the mix formula in the succedent;

2. the right rank $R_r$ of $D$ in the largest number of consecutive nodes on the branch of $D$ staring with the node containing the right premiss of the mix rule, such that each sequent on these nodes contains the mix formula in the antecedent.

The lowest possible rank is evidently 2.
To prove the lemma we carry out two complete inductions, one on the degree $n$, the other on the rank $r$, of the derivation $D$.

It means we prove the lemma for a derivation of the degree $n$, assuming it to hold for derivations of a lower degree (in so far as there are such derivations, i.e., as long as $n \neq 0$), supposing, therefore, that derivations of lower degree can be already transformed into derivations without mix.

Furthermore, we shall begin by considering the case 1 when the rank $r = 2$, and after that the case 2 when the rank $r > 2$, where we assume that the lemma already holds for derivations of the same degree, but a lower rank.

Case 1. Rank of $r = 2$.

We present some cases and leave similar others to the reader as an exercise. Observe that first group contains cases that are especially simple in that they allow the mix to be immediately eliminated. The second group contains the most important cases since their consideration brings out the basic idea behind the whole proof. Here we use the induction hypothesis with respect do the degree of the derivation. We reduce each one of the cases to transformed derivations of a lower degree.

GROUP 1. Axioms and Structural Rules.

1. The left premiss of the mix rule is an axiom $A \rightarrow A$.
Then the sub-tree of $D$ containing mix is as follows.
We transform it, and replace it in \( D \) by

\[
A, \Sigma^* \rightarrow \Delta \\
\wedge^{(\text{mix})} \\
A \rightarrow A \quad \Sigma \rightarrow \Delta
\]

\[
A, \Sigma^* \rightarrow \Delta \\
\text{possibly several exchanges and contractions} \\
\Sigma \rightarrow \Delta
\]

Such obtained \( D^* \) proves the same sequent and contains no mix.

2. The right premiss of the mix rule is an axiom \( A \rightarrow A \).

This is a case dual to 1. We show here the dial transformation, but will leave the dual cases to the reader in the future.

Then the sub-tree of \( D \) containing mix is as follows.

\[
\Sigma \rightarrow \Delta^*, A \\
\wedge^{(\text{mix})} \\
\Sigma \rightarrow \Delta \quad A \rightarrow A
\]

We transform it, and replace it in \( D \) by

\[
\Sigma \rightarrow \Delta^*, A \\
\text{possibly several exchanges and contractions} \\
\Sigma \rightarrow \Delta
\]
Such obtained $D^*$ proves the same sequent and contains no mix.

Suppose that neither of premisses of mix is an axiom. As the rank $r=2$, the right and left ranks are equal one. This means that in the sequents on the nodes directly below left premiss of the mix, the mix formula $A$ does not occur in the succedent; in the sequents on the nodes directly below right premiss of the mix, the mix formula $A$ does not occur in the antecedent.

In general, if a formula occurs in the antecedent (succedent) of a conclusion of a rule of inference, it is either obtained by a logical rule or by a contraction rule.

3. The left premiss of the mix rule is the conclusion of a contraction rule ($\rightarrow contr$). The sub-tree of $D$ containing mix is:

$$
\begin{align*}
\Gamma, \Sigma^* & \to \Delta, \Theta \\
\Lambda^{(mix)} & \\
\Gamma \to \Delta, A & \Sigma \to \Theta \\
| (\rightarrow contr) & \\
\Gamma \to \Delta
\end{align*}
$$

We transform it, and replace it in $D$ by

$$
\begin{align*}
\Gamma, \Sigma^* & \to \Delta, \Theta \\
\text{possibly several weakenings and exchanges} & \\
\Gamma \to \Delta
\end{align*}
$$

Observe that the whole branch of $D$ that starts with the node $\Sigma \to \Theta$ disappears. Such obtained $D^*$ proves the same sequent and contains no mix.

4. The right premiss of the mix rule is the conclusion of a contraction rule ($\rightarrow contr$). It is a dual case to 3. and is left to the reader.

GROUP 2. Logical Rules.

1. The main connective of the mix formula is $\cap$, i.e. the mix formula is $(A \cap B)$. The left premiss of the mix rule is the conclusion of a rule ($\rightarrow \cap$). The right premiss of the mix rule is the conclusion of a rule ($\cap \to$). The right premiss of the mix rule is the conclusion of a rule $(\cap \to)_1$. The sub-tree $T$ of $D$ containing mix is:

63
Γ, Σ → Δ, Θ

\( \land (\text{mix}) \)

Γ → Δ, (A ∩ B)

\( \land (\neg \land) \)

(A ∩ B), Σ → Θ

| (\neg \lor) \)

A, Σ → Θ

Γ → Δ, A

Γ → Δ, B

We transform T into T* as follows.

Γ, Σ → Δ, Θ

possibly several weakenings and exchanges

Γ, Σ* → Δ*, Θ

\( \land (\text{mix}) \)

Γ → Δ, A

A, Σ → Θ

We replace T by T* in D and obtain D*. Now we can apply induction hypothesis with respect to the degree of the mix formula. The mix formula A in D* has a lower degree then the mix formula (A ∩ B) and by the inductive assumption the derivation D*, and hence D may be transformed into one without mix.

2. The case when the left premiss of the mix rule is the conclusion of a rule (→ ∩) and right premiss of the mix rule is the conclusion of a rule (∩ →)\textsubscript{2}

3. The main connective of the mix formula is ∪, i.e. the mix formula is (A ∪ B). The left premiss of the mix rule is the conclusion of a rule (→ ∪)\textsubscript{1} or (→ ∪)\textsubscript{2}. The right premiss of the mix rule is the conclusion of a rule (∪ →)\textsubscript{1}. This is to be dealt with symmetrically to the ∩ cases.

4. The main connective of the mix formula is ¬, i.e. the mix formula is ¬A. The left premiss of the mix rule is the conclusion of a rule (→ ¬). The right premiss of the mix rule is the conclusion of a rule (¬ →).

Here is the sub-tree T of D containing the application of the mix rule.

Γ, Σ → Δ, Θ
\[
\land^{(\text{mix})}
\]
\[
\begin{array}{ll}
\Gamma & \Delta, \neg A \\
\mid (\rightarrow \neg) & \neg A, \Sigma \rightarrow \Theta \\
A, \Gamma & \Delta \\
\Sigma & \Theta, A
\end{array}
\]

We transform $T$ into $T^*$ as follows.

\[
\begin{array}{ll}
\Gamma, \Sigma & \Delta, \Theta \\
\text{possibly several weakenings and exchanges} & \\
\Sigma, \Gamma^* & \Theta^*, \Delta \\
\land^{(\text{mix})} & \\
\Sigma & \Theta, A \\
A, \Gamma & \Delta
\end{array}
\]

We replace $T$ by $T^*$ in $D$ and obtain $D^*$. The new mix in $D^*$ may be eliminated by virtue of inductive assumption, and so from the derivation $D$.

5. The main connective of the mix formula is $\Rightarrow$, i.e. the mix formula is $(A \Rightarrow B)$. The left premiss of the mix rule is the conclusion of a rule $(\rightarrow \Rightarrow)$. The right premiss of the mix rule is the conclusion of a rule $(\Rightarrow \rightarrow)$.

Here is the sub-tree $T$ of $D$ containing the application of the mix rule.

\[
\begin{array}{ll}
\Gamma, \Sigma & \Delta, \Theta \\
\land^{(\text{mix})} & \\
\Gamma & \Delta, (A \Rightarrow B) \\
\mid (\rightarrow \Rightarrow) & (A \Rightarrow B), \Sigma \rightarrow \Theta \\
A, \Gamma & \Delta, B \\
\Sigma & \Theta, A \\
B, \Sigma & \Theta
\end{array}
\]

We transform $T$ into $T^*$ as follows.
\( \Gamma, \Sigma \rightarrow \Delta, \Theta \)

possibly several weakenings and exchanges

\( \Sigma, \Gamma^*, \Sigma^{**} \rightarrow \Theta^*, \Delta^*, \Theta \)

\( \land (\text{mix}) \)

\( \Sigma \rightarrow \Theta, A \quad A, \Gamma, \Sigma^* \rightarrow \Delta^*, \Theta \)

\( \land (\text{mix}) \)

\( A, \Gamma \rightarrow \Delta, B \quad B, \Sigma \rightarrow \Theta, \)

The asterisks are, of course, intended as follows: \( \Sigma^*, \Delta^* \) results from \( \Sigma, \Delta \) by the omission of all formulas \( B \); \( \Gamma^*, \Sigma^{**}, \Theta^* \) results from \( \Gamma, \Sigma^*, \Theta \) by the omission of all formulas \( A \).

We replace \( T \) by \( T^* \) in \( D \) and obtain \( D^* \). Now we have two mixes, but both mix formulas are of a lower degree than \( n \). We first apply the inductive to the assumption to the lower mix. Thus it can be eliminated. We can then also eliminate the upper mix. This ends the proof of the case of rank \( r=2 \).

Case \( r > 2 \).
In the case \( r = 2 \), we generally reduced the derivation to one of lower degree. Now we shall proceed to reduce the derivation to one of the same degree, but of a lower rank. This allows us to to be able to carry the induction with respect to the rank \( r \) of the derivation.

We use the inductive assumption in all cases except, as before, a case of an axiom or structural rules. In these cases the mix can be eliminated immediately, as it was eliminated in the previous case of rank \( r = 2 \).

In a case of logical rules we obtain the reduction of the mix of the lemma to derivations with mix of a lower ranks which consequently can be eliminated by the inductive assumption. We carry now proofs for two logical rules: \((\rightarrow \cap)\) and \((\cup \rightarrow)\). The proof for all other rules is similar and is left to the reader.

Also, we consider a case of left rank \( \text{Lr} = 1 \) and the right rank \( \text{Rr} = r \neq 1 \). The symmetrical case left rank \( \text{Lr} = r \neq 1 \) and the right rank \( \text{Rr} = 1 \) is left to the reader as an exercise.

Case: \( \text{Lr} = 1, \text{Rr} = r > 1 \). The right premiss of the mix is a conclusion of the rule \((\rightarrow \cap)\), i.e. it is of a form \( \Gamma \rightarrow \Delta, (A \cap B) \) and \( \Gamma \) contains the mix formula \( M \). The left premiss of the mix is a sequent \( \Theta \rightarrow \Sigma \) and \( \Theta \) contains the mix formula \( M \). The end of the derivation \( D \), i.e. the sub-tree \( T \) of \( D \) containing mix is:
We transform $T$ into $T^*$ as follows.

\[
\Theta, \Gamma^* \rightarrow \Sigma^*, \Delta, (A \cap B) \\
\land^{(\text{mix})}
\]

\[
\Theta \rightarrow \Sigma \\
\Gamma \rightarrow \Delta, (A \cap B) \\
\land^{(\rightarrow \cap)}
\]

\[
\Gamma \rightarrow \Delta, A \\
\Gamma \rightarrow \Delta, B
\]

We replace $T$ by $T^*$ in $D$ and obtain $D^*$. Now we have two mixes, but both have the right rank $Rr = r - 1$ and both of them can be eliminated by the inductive assumption.

Case: $Lr = 1, Rr = r > 1$. The right premiss of the mix is a conclusion of the rule $(\cup \rightarrow$, i.e. it is of a form $(A \cup B), \Gamma \rightarrow \Delta$ and $\Gamma$ contains the mix formula $M$. The left premiss of the mix is a sequent $\Theta \rightarrow \Sigma$ and $\Theta$ contains the mix formula $M$. The end of the derivation $D$, i.e. the sub-tree $T$ of $D$ containing mix is:

\[
\Theta, (A \cup B)^*, \Gamma^* \rightarrow \Sigma^*, \Delta, (A \cap B) \\
\land^{(\text{mix})}
\]

\[
\Theta \rightarrow \Sigma \\
(A \cup B) \Gamma \rightarrow \Delta \\
\land^{(\cup \rightarrow)}
\]

\[
A, \Gamma \rightarrow \Delta \\
B, \Gamma \rightarrow \Delta
\]
\((A \cup B)^*\) stands either for or for nothing according as \((A \cup B)\) is unequal or equal to the mix formula \(M\). The mix formula \(M\) certainly occurs in \(\Gamma\). For otherwise \(M\) would be equal to \((A \cup B)\) and the right rank \(R_r\) would be equal to 1 contrary to the assumption.

We transform \(T\) into \(T^*\) as follows.

\[
\begin{align*}
\Theta, (A \cup B), \Gamma^* & \rightarrow \Sigma^*, \Delta, (A \cap B) \\
\land (\land \rightarrow) \\
A, \Theta, \Gamma^* & \rightarrow \Sigma^*, \Delta \\
\text{some weakenings, exchanges} & \\
\Theta, A^*, \Gamma^* & \rightarrow \Sigma^*, \Delta \\
\land (\text{mix}) & \\
B, \Theta, \Gamma^* & \rightarrow \Sigma^*, \Delta \\
\text{some weakenings, exchanges} & \\
\Theta, B^*, \Gamma^* & \rightarrow \Sigma^*, \Delta \\
\land (\text{mix}) & \\
\Theta & \rightarrow \Sigma \\
A, \Gamma & \rightarrow \Delta \\
\Theta & \rightarrow \Sigma \\
B, \Gamma & \rightarrow \Delta \\
\end{align*}
\]

Now we have two mixes, but both have the right rank \(R_r = r-1\) and both of them can be eliminated by the inductive assumption. We replace \(T\) by \(T^*\) in \(D\) and obtain \(D^*\). This ends the proof of the Hauptzatz lemma and hence the proof of the Hauptzatz Theorem 19.

Let’s denote by \(L^K_c\) the system \(L^K\) without the cut rule, i.e. we put

\[
L^K - \text{c} = L^K - \{\text{cut}\}.
\]

Directly from the completeness theorem 18 18 and the Hauptzatz Theorem 19 we get that the following.

**Theorem 20 (L^K-c Completeness)**

*For any sequent \(\Gamma \rightarrow \Delta,\)

\[
\vdash_{L^K-c} \Gamma \rightarrow \Delta \quad \text{if and only if} \quad \models \Gamma \rightarrow \Delta.
\]

Let \(G\) be the Gentzen sequents proof system defined by (27). We replace the logical axiom of \(G\)

\[
\Gamma_1', a, \Gamma_2' \rightarrow \Delta_1', a, \Delta_2',
\]

where \(a \in \text{VAR}\) is any propositional variable and \(\Gamma_1', \Gamma_2', \Delta_1', \Delta_2' \in \text{VAR}^*\) are any indecomposable sequences, by a new logical axiom

\[
\Gamma_1, A, \Gamma_2 \rightarrow \Delta_1, A, \Delta_2
\]

(36)
for any $A \in \mathcal{F}$ and any sequences $\Gamma_1, \Gamma_2, \Delta_1, \Delta_2 \in SQ$. We call a resulting proof system $\textsf{GK}$, i.e. we have that

$$\textsf{GK} = (\mathcal{L}_{\cup, \cap, \Rightarrow, \neg}, \ SQ, \ LA, \ \mathcal{R})$$

where $LA$ is the axiom (36) and $\mathcal{R}$ is the set (29) of rules of $G$.

Observe that the only difference between the systems $\textsf{GK}$ and $G$ is the form of their logical axioms, both being tautologies. Hence hence get the proof completeness of $\textsf{GK}$ in the same way as we proved it for $G$, i.e. we have the following.

**Theorem 21**

*For any formula $A \in \mathcal{F}$,*

$$\vdash_{\textsf{GK}} A \quad \text{if and only if} \quad \models A.$$

*For any sequent $\Gamma \rightarrow \Delta \in SQ$,*

$$\vdash_{\textsf{GK}} \Gamma \rightarrow \Delta \quad \text{if and only if} \quad \models \Gamma \rightarrow \Delta.$$

By the $\textsf{GK}$ the completeness theorem 21, $\textsf{LK-c}$ completeness theorem 20 we get the equivalency of $\textsf{GK}$ and the cut free $\textsf{LK-c}$.

**Theorem 22 (Equivalency)**

*The proof systems $\textsf{GK}$ and $\textsf{LK-c}$ are equivalent, i.e for any sequent $\Gamma \rightarrow \Delta$,*

$$\vdash_{\textsf{LK-c}} \Gamma \rightarrow \Delta \quad \text{if and only if} \quad \vdash_{\textsf{GK}} \Gamma \rightarrow \Delta.$$  

### 6.1 Homework Problems

1. Write all proofs in $\textsf{GL}$ of $(-\neg(a \cap b) \Rightarrow (-\neg a \cup \neg b)))$.
2. Find a formula which has a unique decomposition tree in $\textsf{GL}$.
3. Define shortly, in your own words, for any formula $A \in \mathcal{F}$, its decomposition tree $T_{\rightarrow A}$ in $G$.
4. Extend your definition $T_{\rightarrow A}$ in $G$ to a decomposition tree $T_{\Gamma \rightarrow \Delta}$.
5. Prove that for any $\Gamma \rightarrow \Delta \in SQ$, the decomposition tree $T_{\Gamma \rightarrow \Delta}$ in $G$ are finite.
6. Write all proofs in $\textsf{G}$ of $(-\neg(a \cap b) \Rightarrow (-\neg a \cup \neg b)))$.
7. Find a formula $A$ which has a unique decomposition tree in $\textsf{G}$.
8. Prove strong soundness of rules \((\rightarrow \cup), (\rightarrow \Rightarrow)\) in \(\text{GL}\). List all logical equivalences used in the proofs.

9. Prove strong soundness of rules \((\Rightarrow \rightarrow), (\neg \rightarrow)\) in \(\text{GL}\). List all logical equivalences used in the proofs.

10. Prove strong soundness of rules \((\cup \rightarrow), (\rightarrow \neg), (\cap \rightarrow)\) in \(\text{G}\). List all logical equivalences used in the proofs.

11. Prove strong soundness of rules \((\Rightarrow \rightarrow), (\neg \rightarrow \cup), (\Rightarrow \rightarrow)\) in \(\text{G}\). List all logical equivalences used in the proofs.

12. Explain why the system \(\text{G}\) is strongly sound.

13. Prove the following.
   
   For any sequent \(\Gamma \rightarrow \Delta \in \mathit{SQ}\), if \(\vdash_{\text{G}} \Gamma \rightarrow \Delta\), then \(\models \Gamma \rightarrow \Delta\).

14. Given a formula \(A = ((b \Rightarrow (a \cap c)) \Rightarrow (\neg(a \cup c) \Rightarrow (\neg b \cup a)))\).
   
   (i) Find all counter models determined by the decomposition trees of \(A\) in \(\text{GL}\). Explain why the definition of a counter model determined by the decomposition tree is correct.

   (ii) Find all counter models determined by the decomposition trees of \(A\) in \(\text{G}\). Explain why the definition of a counter model determined by the decomposition tree is correct.

15. Prove the following.
   
   Given a sequent \(\Gamma \rightarrow \Delta\), such that its decomposition tree \(T_{\Gamma \rightarrow \Delta}\) in \(\text{G}\) contains a non-axiom leaf \(L_A\). Any truth assignment \(v\) that falsifies the non-axiom leaf \(L_A\) is a counter model for \(\Gamma \rightarrow \Delta\).

16. Prove the following.
   
   For any sequent \(\Gamma \rightarrow \Delta \in \mathit{SQ}\), if \(\vdash_{\text{G}} \Gamma \rightarrow \Delta\), then \(\models \Gamma \rightarrow \Delta\).

17. Let \(\text{LK-c} = \text{LK} - \{(\text{cut})\}\) and \(\text{GK}\) be proof systems defined as defined by (35) and (37), respectively
   
   (i) We know that \(\text{GK}\) is strongly sound. Prove that \(\text{LK-c}\) is sound but not strongly sound.

   (ii) Find proofs of axioms A3, A7, and A11 of the R system (32) in \(\text{LK-c}\) and in \(\text{GK}\), i.e. proofs of formulas \((B \Rightarrow (A \cup B)), ((C \Rightarrow A) \Rightarrow ((C \Rightarrow B) \Rightarrow (C \Rightarrow (A \cap B)))), and ((A \Rightarrow (A \cap \neg A)) \Rightarrow \neg A)\). Compare your results.

   (iii) Find proofs of axioms A1, A8, and A9 of the R system (32) in \(\text{LK-c}\) and in \(\text{GK}\), i.e. proofs of formulas \(((A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C))), ((A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \cap B) \Rightarrow C)), and (((A \cap B) \Rightarrow C) \Rightarrow (A \Rightarrow (B \Rightarrow C)))\). Compare your results.
(iv) Find proofs of axioms A1, A5, and A12 of the R system (32) in LK-c and in GK, i.e. proofs of formulas \((A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C)))\), \(((A \cap B) \Rightarrow A)\), and \((A \cup \neg A)\). Compare your results.

18. Define shortly, in your own words, for any formula \(A \in \mathcal{F}\), its decomposition tree \(T_A\) in LK-c. Is the tree \(T_A\) always finite?

19. Given a formula \(A = (\neg(a \cap b) \Rightarrow (\neg a \cup \neg b))\). Construct one infinite and one infinite decomposition tree for \(A\).

20. Describe major differences in the decomposition trees in LK-c and GK.

21. We have proved that LK-c and GK are equivalent, i.e. that for any sequent \(\Gamma \rightarrow \Delta\),

\[ \vdash_{\text{LK-c}} \Gamma \rightarrow \Delta \text{ if and only if } \vdash_{\text{GK}} \Gamma \rightarrow \Delta. \]

The proof was not constructive; it was obtained from that fact that both systems are complete.

(ii) Describe a constructive procedure of transforming any proof in GK into a proof in LK-c.

(i) Transform a proof of a formula \((A \Rightarrow (A \cup B))\) in GK into a proof in LK-c.

(ii) Describe a constructive procedure of transforming any proof in GK into a proof in LK-c.

(iii) Show that the procedure of elimination of structural rules of LK-c leads to the rules inference of GK.