LECTURE 10
Chapter 10
CLASSICAL AUTOMATED PROOF SYSTEMS

PART 1: RS SYSTEM
PART 2: RS1, RS2, RS3 SYSTEMS
PART 3: GENTZEN SYSTEMS
CLASSICAL AUTOMATED PROOF SYSTEMS

Hilbert style systems are easy to define and admit a relatively simple proofs of the Completeness Theorem but they are difficult to use.

Automated systems are less intuitive then the Hilbert-style systems, but they will allow us to define effective automatic procedures for proof search, what is impossible in a case of the Hilbert-style systems.

The first idea of this type was presented by G. Gentzen in 1934.

We present in this chapter our version of original Gentzen system for propositional classical logic.

We present the original Gentzen systems for Intuitionistic and Classical Propositional Logics in Chapter 12.
PART 1: RS System

The automated proof system we presented here is due to Helena Rasiowa and Roman Sikorski. We present here the propositional version of the original system and call it RS system for Rasiowa-Sikorski.

The propositional RS system extends naturally to predicate logic QRS system which is presented in Chapter 14. Both systems RS and QRS admit a constructive proof of Completeness Theorem. First such constructive proofs were given, together with the formalization of the systems by H. Rasiowa and Sikorski in 1961.
AUTOMATED PROOF SYSTEMS

PART 2: RS1, RS2 Systems

We define, as an exercise 2 versions of the RS System, discuss their differences and show how the proof of Completeness Theorem for RS extends to similar proofs for all 3 systems.
PART 3: GENTZEN Systems - Lecture 13

We present our modern versions of Gentzen Sequent systems for propositional classical logic.

Both systems extend easily to predicate logic and admit a constructive proof of Completeness Theorem via Rasiowa-Sikorski method.

The original Gentzen system LK for classical propositional logic is presented in Chapter 12 together with the original Gentzen system LI for the Intuitionistic propositional logic.
PART1:
RS Proof System for Classical Propositional Logic
RS Proof System

Language of \textbf{RS} is

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

The rules of inference of our system \textbf{RS} operate on finite sequences of formulas and we adopt

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

as the set of \textbf{expressions} of \textbf{RS}

\textbf{Notation}

Elements of \textbf{\mathcal{E}} are finite sequences of formulas and we denote them by

\[ \Gamma, \Delta, \Sigma \ldots \]

with indices if necessary.
RS Proof System

The intuitive meaning of a sequence $\Gamma \in \mathcal{F}^*$ is that the truth assignment $\nu$ makes it true if and only if it makes the formula of the form of the disjunction of all formulas of $\Gamma$ true.

For any sequence $\Gamma \in \mathcal{F}^*$, $\Gamma = A_1, A_2, ..., A_n$

we denote

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

We define as the next step a formal semantics for RS.
Formal Semantics for RS

Let $v : \text{VAR} \rightarrow \{T, F\}$ be a truth assignment and $v^*$ its classical semantics extension to the set of formulas $\mathcal{F}$. We formally extend $v$ to the set $\mathcal{F}^*$ of all finite sequences of $\mathcal{F}$ as follows:

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

The sequence $\Gamma$ is said to be **satisfiable** if there is a truth assignment $v : \text{VAR} \rightarrow \{T, F\}$ such that $v^*(\Gamma) = T$. We write it as

$$v \models \Gamma$$

and call $v$ a **model** for $\Gamma$.
Formal Semantics for RS

The sequence $\Gamma$ is said to be **falsifiable** if there is a truth assignment $v$, such that $v^* (\Gamma) = F$

Such a truth assignment $v$ is called a **counter-model** for $\Gamma$

The sequence $\Gamma$ is said to be a **tautology** iff $v^* (\Gamma) = T$ for all truth assignments $v : \text{VAR} \rightarrow \{T, F\}$

We write as always,

$$\models \Gamma$$

to denote that $\Gamma$ is a **tautology**
Example

Let \( \Gamma \) be a sequence

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

The truth assignment \( v \) such that

\[ v(a) = F \quad \text{and} \quad v(b) = T \]

falsifies \( \Gamma \), i.e. is a counter-model for \( \Gamma \) as shows the following computation

\[ v^*(\Gamma) = v^*(\delta_{\Gamma}) = v^*(a) \cup v^*(b \cap a) \cup v^*(\neg b) \cup v^*(b \Rightarrow a) = F \cup (F \cap T) \cup F \cup (T \Rightarrow F) = F \cup F \cup F \cup F = F. \]
Rules of inference

Rules of inference of RS are of the form:

\[
\frac{\Gamma_1}{\Gamma} \quad \text{or} \quad \frac{\Gamma_1 ; \Gamma_2}{\Gamma}
\]

where \( \Gamma_1, \Gamma_2 \) are called \text{premisses} and \( \Gamma \) is called the \text{conclusion} of the rule.

Each rule of inference \text{introduces} a new \text{logical connective} or a \text{negation} of a \text{logical connective}.

We \text{name} the rule that introduces the logical connective \( \circ \) in the conclusion sequent \( \Gamma \) by \( (\circ) \).

The notation \( (\neg\circ) \) means that the \text{negation} of the logical connective \( \circ \) is introduced in the conclusion sequence \( \Gamma \).
Proof System  **RS** contains seven inference rules:

\[(\cup), (\neg\cup), (\cap), (\neg\cap), (\Rightarrow), (\neg\Rightarrow), (\neg\neg)\]

Before we **define** the **rules of inference** of **RS** we need to introduce some definitions.

**Definition**

Any propositional variable, or a negation of propositional variable is called a **literal**

The set

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

is called a set of all propositional **literals**

The **variables** are called **positive literals**

Negations of variables are called **negative literals.**
We denote by
\[ \Gamma', \Delta', \Sigma' \ldots \]
finite sequences (empty included) formed out of literals i.e
\[ \Gamma', \Delta', \Sigma' \in LT^* \]
We will denote by
\[ \Gamma, \Delta, \Sigma \ldots \]
the elements of \( \mathcal{F}^* \)
Logical Axioms of RS

We adopt as an logical axiom of RS any sequence of literals which contains a propositional variable and its negation, i.e. any sequence

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

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

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

We denote by LA the set of all logical axioms of RS.
Inference Rules of RS

Disjunction rules

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

Conjunction rules

\[ \begin{align*}
\text{(∩)} & \quad \Gamma', A, \Delta ; \quad \Gamma', B, \Delta \\
\quad & \quad \frac{}{\Gamma', (A \cap B), \Delta} \\
\text{(-∩)} & \quad \Gamma', \neg A, \neg B, \Delta \\
\quad & \quad \frac{}{\Gamma', \neg(A \cap B), \Delta} \\
\end{align*} \]
Inference Rules of RS

Implication rules

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

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

Negation rule

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

where \[ \Gamma' \in LT^*, \Delta \in \mathcal{F}^*, A, B \in \mathcal{F} \]
Proof System RS

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

\[
RS = \left( L\{\neg, \Rightarrow, \cup, \cap\}, \ F^*, \ LA, \ R \right)
\]

where the set of inference rules is

\[
R = \{(\cup), \ (\neg\cup), \ (\cap), \ (\neg\cap), \ (\Rightarrow), \ (\neg \Rightarrow), \ (\neg\neg)\}
\]

and \( LA \) is the set of all logical axioms
Proof Trees

Definition
By a **proof tree** in RS of \( \Gamma \) we understand a tree

\[
T_\Gamma
\]

built out of sequences satisfying the following conditions:
1. The topmost sequence, i.e the **root** of \( T_\Gamma \) is the sequence \( \Gamma \)
2. **All leafs** are axioms
2. **The nodes** are sequences such that each sequence on the tree follows from the ones immediately preceding it by one of the **inference rules**
Proof Trees

We picture, and write our proof trees with the root on the top, and the leafs on the very bottom,

Additionally we write our proof trees indicating the name of the inference rule used at each step of the proof.

Example
Assume that a proof of a sequence $\Gamma$ from some three axioms was obtained by the subsequent use of the rules $(\cap), (\cup), (\cup), (\cap), (\cup)$, and $(\neg\neg), (\Rightarrow)$.
We represent it as the following tree.
Proof Trees

The tree $T_\Gamma$

\[
\begin{align*}
\Gamma \\
| (\Rightarrow) \\
\text{conclusion of (\neg\neg)} \\
\neg (\neg\neg) \\
\text{conclusion of (\cup)} \\
(\cup) \\
\text{conclusion of (\cap)} \\
\wedge (\cap) \\
\text{conclusion of (\cap)} & \quad \text{conclusion of (\cup)} \\
\cup (\cup) & \quad (\cup) \\
\text{axiom} & \quad \text{conclusion of (\cap)} \\
\wedge (\cap) & \quad \text{axiom}
\end{align*}
\]
Proof Trees

The Proof Trees represent a certain visualization for the proofs.
Any formal proof in any proof system can be represented in a tree form and vice-versa.
Any proof tree can be re-written in a linear form as a previously defined formal proof.

Example
The proof tree in RS of the de Morgan Law

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

is the as follows
Proof Trees

The tree $T_A$

$$
\neg(a \land b) \Rightarrow (\neg a \lor \neg b)
\vert (\Rightarrow)
\neg\neg(a \land b), (\neg a \lor \neg b)
\vert (\neg\neg)
(a \land b), (\neg a \lor \neg b)
\land (\neg)
\land (\neg)

a, (\neg a \lor \neg b) \quad b, (\neg a \lor \neg b)
\vert (\lor)
\land (\lor)

a, \neg a, \neg b \quad b, \neg a, \neg b$$
Formal Proof

To obtain a formal proof (written in a vertical form) of $A$, it we just write down the tree as a sequence, starting from the leaves and going up (from left to right) to the root.

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

A search for the proof in RS of other de Morgan Law

$$A = (\neg(a \cup b) \Rightarrow (\neg a \cap \neg b))$$

consists of building a certain tree and proceeds as follows.
Example

The tree $T_A$

$$\neg(a \cup b) \implies (\neg a \cap \neg b)$$

$$\begin{align*}
\vdash (\implies) \\
\vdash (\neg) \\
\neg\neg(a \cup b), (\neg a \cap \neg b) \\
\vdash (\cup) \\
a, b, (\neg a \cap \neg b) \\
\wedge (\cap)
\end{align*}$$

$a, b, \neg a \quad a, b, \neg b$
Example

We construct its formal proof, as before, written in a vertical manner. Here it is:

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

Our GOAL in inventing proof systems like RS is to facilitate automatic proof search.

The method of such proof search is to generate what is called the decomposition trees.

The decomposition tree for

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

is built as follows.
Decomposition Trees

The tree $T_A$

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

\[
| (\cup)
\]

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

\[
\wedge (\cap)
\]

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

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

\[
| (\Rightarrow)
\]

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

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

\[
| (\Rightarrow)
\]

\[
\neg a, b, \neg a, c
\]
RS: DECOMPOSITION RULES
and
DECOMPOSITION TREES
Decomposition Trees

The process of searching for a proof of a formula \( A \in F \) in RS consists of building a certain tree \( T_A \), called a decomposition tree.

Building a decomposition tree, i.e. a proof search tree consists in the first step of transforming the RS rules into corresponding decomposition rules.
RS Decomposition Rules

Here are all of RS decomposition rules

Disjunction decomposition rules

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

Conjunction decomposition rules

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

Implication decomposition rules

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

Negation decomposition rule

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

where \( \Gamma' \in \mathcal{F}'^* \), \( \Delta \in \mathcal{F}^* \), \( A, B \in \mathcal{F} \)
Tree Decomposition Rules

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

Tree Decomposition Rules

(∪) rule

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

\[ \Gamma', A, B, \Delta \]
Tree Decomposition Rules

(¬∪) rule

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

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

(∩) rule

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

\[ \Gamma', A, \Delta \quad \Gamma', B, \Delta \]
Tree Decomposition Rules

(¬∪) rule

Γ′, ¬(A ∩ B), Δ

| (¬∩)

| (¬∪)

Γ′, ¬A, ¬B, Δ

(⇒) rule

Γ′, (A ⇒ B), Δ

| (∪)

Γ′, ¬A, B, Δ
Tree Decomposition Rules

(¬ ⇒) rule

Γ′, ¬(A ⇒ B), Δ

Γ′, ¬B, Δ

(¬¬) rule

Γ′, ¬¬A, Δ

Γ′, A, Δ
Observe that we use the same names for the inference and decomposition rules, as once the we have built the decomposition tree with all leaves being axioms, it constitutes a proof of $A$ in $RS$ with branches labeled by the proper inference rules.

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

**Definition: Indecomposable Sequence**

A sequence $\Gamma'$ built only out of literals, i.e. $\Gamma \in \mathcal{F}'^*$ is called an **indecomposable sequence**.
Definitions and Observations

Definition: Indecomposable Sequence
A sequence $\Gamma'$ built only out of literals, i.e. $\Gamma' \in \mathcal{F}^*$ is called an **indecomposable sequence**

Definition: Decomposable Formula
A formula $A$ that is not a literal, i.e. $A \in \mathcal{F} - LT$ is called a **decomposable formula**

Definition: Decomposable Sequence
A sequence $\Gamma$ that contains a **decomposable formula** is called a **decomposable sequence**
Definitions and Observations

Observation 1

For any **decomposable** sequence, i.e. for any $\Gamma \notin LT^*$ 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.
Definitions and Observations

Observation 2
If the main connective of the first decomposable formula is $\cup, \cap, \Rightarrow$,
then the decomposition rule determined by it is $(\cup), (\cap), (\Rightarrow)$, respectively

Observation 3
If the main connective of the first decomposable formula $A$ is negation $\neg$,
then the decomposition rule is determined by the second connective of the formula $A$.
The corresponding decomposition rules are $(\neg \cup), (\neg \cap), (\neg \neg), (\neg \Rightarrow)$
Because of the importance of the **Observation 1** we re-write it in a form of the following

**Decomposition Lemma**

For any sequence $\Gamma \in F^*$, $\Gamma \in LT^*$ or $\Gamma$ is in the domain of exactly one of RS Decomposition Rules

This rule is **determined** by the first decomposable formula in $\Gamma$ and by the **main connective** of that formula
Decomposition Tree Definition

Definition: Decomposition Tree \( T_A \)
For each \( A \in \mathcal{F} \), a decomposition tree \( T_A \) is a tree build as follows

Step 1.
The formula \( A \) is the \textbf{root} of \( T_A \)

For any other \textbf{node} \( \Gamma \) of the tree we follow the steps below

Step 2.
If \( \Gamma \) is \textbf{indecomposable} then \( \Gamma \) becomes a \textbf{leaf} of the tree
Decomposition Tree Definition

Step 3.
If $\Gamma$ is decomposable, then we traverse $\Gamma$ from left to right and identify the first decomposable formula $B$.

By the Decomposition Lemma, there is exactly one decomposition rule determined by the main connective of $B$.

We put its premiss as a node below, or its left and right premisses as the left and right nodes below, respectively.

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

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

**Decomposition Tree Theorem**

For any sequence $\Gamma \in \mathcal{F}^*$ 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 leafs are axioms
3. $\not\models_{RS} \Gamma$ if and only if $T_\Gamma$ has a non-axiom leaf
Theorem

Proof
The tree $T_\Gamma$ is unique by the Decomposition Lemma

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 $T_\Gamma$ has a non-axiom leaf it is not a proof by definition

By 1. it also means that the proof does not exist
Example

Example
Let’s construct, as an example a decomposition tree $T_A$ of the following formula $A$

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

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

$$((a \cup b) \Rightarrow \neg a) \boxed{\cup} (\neg a \Rightarrow \neg c))$$
Example

The first and only decomposition rule to be applied is \((\cup)\)

The first segment of the decomposition tree \(T_A\) is

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

| (\cup)

| (\cup)

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

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 \( \Rightarrow \) and we put the box around it

\[ ((a \cup b)\boxed{\Rightarrow} \neg a), (\neg a \Rightarrow \neg c) \]
Example

The decomposition tree becomes now

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

| (\cup)
| (\cup)

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

| (\Rightarrow)

\(\neg (a \cup b), \neg a, (\neg a \Rightarrow \neg c)\)
Example

The next sequence to decompose is

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

with the first decomposable formula

\[ \neg(a \cup b) \]

Its main connective is \( \neg \), so to find the appropriate rule we have to examine next connective, which is \( \cup \)

The decomposition rule determine by this stage of decomposition is \( (\neg \cup) \)
Example

Next stage of the construction of the decomposition tree $T_A$ is

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

| (∪)
| ((a \cup b) \Rightarrow \neg a), (\neg a \Rightarrow \neg c)
| (∴)
| \neg (a \cup b), \neg a, (\neg a \Rightarrow \neg c)
| (∧(\neg \cup))

\neg a, \neg a, (\neg a \Rightarrow \neg c) \quad \neg b, \neg a, (\neg a \Rightarrow \neg c)
Example

Finally, the complete $T_A$ is

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

| (\bigcup)

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

| (\Rightarrow)

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

| (\neg \bigcup)

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

| (\Rightarrow)

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

| (\Rightarrow)

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

| (\neg)

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

| (\neg)

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

|
Example

All leaves of $T_A$ are axioms

The tree $T_A$ is a proof of $A$ in $RS$, i.e.

$$\vdash_{RS} ((a \cup b) \Rightarrow \neg a) \cup (\neg a \Rightarrow \neg c)$$
Example

Given a formula $A$ and its decomposition tree $T_A$

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

$\setminus$ (\cup)

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

$\wedge$ (\cap)

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

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

$\setminus$ (\Rightarrow)

$\neg c, \neg a, c$

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

$\setminus$ (\Rightarrow)

$\neg a, b, \neg a, c$
Example

There is a leaf \( \neg a, b, \neg a, c \) of the tree \( T_A \) that is not an axiom. By the Decomposition Tree Theorem

\[ \forall_{RS} ((a \Rightarrow b) \cap \neg c) \cup (a \Rightarrow c) \]

It means that the proof in \( RS \) of the formula \( ((a \Rightarrow b) \cap \neg c) \cup (a \Rightarrow c) \) does not exists
Completeness Theorem

Our main goal is to prove the Completeness Theorem for RS. We prove first the following Completeness Theorem for formulas $A \in F$

**Completeness Theorem 1**  
For any formula $A \in F$

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

and then we generalize it to the following

**Completeness Theorem 2**  
For any $\Gamma \in F^*$,

$$\vdash_{RS} \Gamma \quad \text{if and only if} \quad \models \Gamma$$

Do so we need to introduce a new notion of a Strong Soundness and prove that the RS is strongly sound.
Strong Soundness of RS
Strong Soundness

Definition
Given a proof system

\[ S = (\mathcal{L}, \mathcal{E}, \mathcal{LA}, \mathcal{R}) \]

Definition
A rule \( r \in \mathcal{R} \) such that the conjunction of all its premisses is logically equivalent to its conclusion is called strongly sound

Definition
A proof system \( S \) is called strongly sound iff \( S \) is sound and all its rules \( r \in \mathcal{R} \) are strongly sound
Strong Soundness of RS

**Theorem**
The proof system **RS** is strongly sound

**Proof**
We prove as an example the strong soundness of two of inference rules: $(\cup)$ and $(\neg\cup)$

Proof for all other rules follows the same patterns and is left as an exercise.

By definition of strong soundness we have to show that

If $P_1, P_2$ are premisses of a given rule and $C$ is its conclusion, then for all $v$,

$v^*(P_1) = v^*(C)$

in case of one premiss rule and

$v^*(P_1) \cap v^*(P_2) = v^*(C)$

in case of the two premisses rule.
Strong Soundness of RS

Consider the rule \((\cup)\)

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

We evaluate:

\[
v^*(\Gamma', A, B, \Delta) = v^*(\delta_{\{\Gamma', A, B, \Delta\}}) = v^*(\Gamma') \cup v^*(A) \cup v^*(B) \cup v^*(\Delta) = v^*(\delta_{\{\Gamma', (A \cup B), \Delta\}}) = v^*(\Gamma', (A \cup B), \Delta)
\]
Strong Soundness of RS

Consider the rule \((-\cup)\)

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

We evaluate:

\[
\nu^*(P_1) \cap \nu^*(P_2) = \nu^*(\Gamma', \neg A, \Delta) \cap \nu^*(\Gamma', \neg B, \Delta)
\]

\[
= (\nu^*(\Gamma') \cup \nu^*(\neg A) \cup \nu^*(\Delta)) \cap (\nu^*(\Gamma') \cup \nu^*(\neg B) \cup \nu^*(\Delta))
\]

\[
= (\nu^*(\Gamma', \Delta) \cup \nu^*(\neg A)) \cap (\nu^*(\Gamma', \Delta) \cup \nu^*(\neg B))
\]

\[
= \text{distrib} \quad (\nu^*(\Gamma', \Delta) \cup (\nu^*(\neg A) \cap \nu^*(\neg B))
\]

\[
= \nu^*(\Gamma') \cup \nu^*(\Delta) \cup \nu^*(\neg A \cap \neg B) = \text{deMorgan} \quad \nu^*(\delta_{\{\Gamma', \neg (A \cup B), \Delta\}}
\]

\[
= \nu^*(\Gamma', \neg (A \cup B), \Delta) = \nu^*(C)
\]
Soundness Theorem

Observe that the strong soundness notion implies soundness (not only by name!). Obviously the LA of RS are tautologies, hence we have also proved the following

Soundness Theorem for RS

For any $\Gamma \in \mathcal{F}^*$,

$$\text{If } \vdash_{RS} \Gamma, \text{ then } |\Gamma| = \Gamma$$

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

$$\text{If } \vdash_{RS} A, \text{ then } |A| = A$$
Strong Soundness

We proved that all the rules of inference of RS are strongly sound, i.e. $C \equiv P$ and $C \equiv P_1 \cap P_2$

Strong soundness of the rules means that if at least one of premisses of a rule is false, so is its conclusion.

Given a formula $A$, such that its $T_A$ has a branch ending with a non-axiom leaf.

By strong soundness, 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 $v$ that falsifies a non-axiom leaf is a counter-model for $A$. 
Counter Model Theorem

We have proved the following

**Counter Model Theorem**
Let $A \in F$ be such that its decomposition tree $T_A$ contains a *non-axiom* leaf $L_A$
Any truth assignment $v$ that falsifies $L_A$ is a *counter model* for $A$

Any truth assignment that falsifies a *non-axiom leaf* is called a *counter-model* for $A$ determined by the decomposition tree $T_A$
Counter Model Example

Consider a tree $\mathcal{T}_A$

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

$\mid (\cup)$

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

$\land (\cap)$

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

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

$\mid (\Rightarrow)$

$\neg c, \neg a, c$

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

$\mid (\Rightarrow)$

$\neg a, b, \neg a, c$
Counter Model Example

The tree $T_A$ has a **non-axiom leaf** $L_A : \neg a, b, \neg a, c$

We define a truth assignment $v : \text{VAR} \rightarrow \{T, F\}$ that **falsifies** the leaf $L_A$ as follows

Observe that $v$ must be such that

$v^*(\neg a, b, \neg a, c) = v^*(\neg a) \cup v^*(b) \cup v^*(\neg a) \cup v^*(c) = \neg v(a) \cup v(b) \cup \neg v(a) \cup v(c) = F$, i.e. all components of the disjunction must be put to $F$

We hence get that $v$ must be such that

$v(a) = T, \ v(b) = F, \ v(c) = F$

By the **Counter Model Theorem**, the $v$ determined by the non-axiom leaf also **falsifies** the formula $A$, i.e. we proved that $v$ is a **counter model** for $A$ and

$\not\models (((a \Rightarrow b) \cap \neg c) \cup (a \Rightarrow c))$
Counter Model

The **Counter Model Theorem** says that $F$ determined by the non-axiom leaf "climbs" the tree $T_A$

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

$| (\cup)$

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

$\wedge (\cap)$

$(a \Rightarrow b), (a \Rightarrow c) = F$

$| (\Rightarrow)$

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

$| (\Rightarrow)$

$\neg c, \neg a, c$

$axiom$

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

$| (\Rightarrow)$

$\neg a, b, \neg a, c = F$
Counter Model

**Observe** that the same counter model construction applies to any other non-axiom leaf, if exists.

The other non-axiom leaf defines another $F$ that also "climbs the tree" picture, and hence defines another counter-model for $A$.

By **Decomposition Tree Theorem** all possible restricted counter-models for $A$ are those determined by all non-axioms leaves of the $T_A$.

In our case the formula $T_A$ has only one non-axiom leaf, and hence only one restricted counter model.
RS Completeness Theorem

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

If $\models A$, then $\vdash_{RS} A$

We prove instead the opposite implication

RS Completeness Theorem

If $\not\vdash_{RS} A$ then $\not\models A$
Proof of Completeness Theorem

Proof of Completeness Theorem

Assume that $A$ is any formula such that

$$\nu_{RS} A$$

By the **Decomposition Tree Theorem** the $T_A$ contains a non-axiom leaf $L_A$

The non-axiom leaf $L_A$ defines a truth assignment $\nu$ which falsifies it as follows:

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

Hence by **Counter Model Theorem** we have that $\nu$ also falsifies $A$, i.e.

$$\not\models A$$
PART2: Proof Systems RS1 and RS2
RS1 Proof System

Language of RS1 is the same as the language of RS, i.e.

\[ \mathcal{L} = \mathcal{L}\{-,\Rightarrow,\lor,\land\} \]

The rules of inference of our system RS1 operate as rules of RS on finite sequences of formulas and we adopt

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

as the set of expressions of RS1

Notation
Elements of \( \mathcal{E} \) are finite sequences of formulas and we denote them by

\[ \Gamma, \Delta, \Sigma \ldots \]

with indices if necessary.
Rules of inference of RS1

Proof System RS1 contains seven inference rules, denoted by the same symbols as the rules of RS:

$(\lor), \ (\neg\lor), \ (\land), \ (\neg\land), \ (\Rightarrow), \ (\neg\Rightarrow), \ (\neg\neg)$

The inference rules of RS1 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.

The set $LT = VAR \cup \{\neg a : a \in VAR\}$ is called a set of all propositional literals.

The variables are called positive literals.

Negations of variables are called negative literals.
Literals Notation

We denote, as before, by

$$\Gamma', \ \Delta', \ \Sigma' \ldots$$

finite sequences (empty included) formed out of literals i.e

$$\Gamma', \ \Delta', \ \Sigma' \in LT^*$$

We will denote by

$$\Gamma, \ \Delta, \ \Sigma \ldots$$

the elements of $$F^*$$
Logical Axioms of RS1

We adopt all logical axiom of RS as the axioms of RS1, i.e.

Logical Axioms LA of RS1 are as follows

$$\Gamma', a, \Gamma_2, \neg a, \Gamma_3$$

$$\Gamma', \neg a, \Gamma_2, a, \Gamma_3$$

where $a \in VAR$ is any propositional variable
Inference Rules of RS1

Disjunction rules

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

$\forall \Gamma, \neg A, \Delta' \frac{\Gamma, \neg A, \Delta'}{\Gamma, \neg(A \cup B), \Delta'}$

Conjunction rules

$\forall \Gamma, A, \Delta' \frac{\Gamma, A, \Delta'}{\Gamma, (A \cap B), \Delta'}$

$\forall \Gamma, \neg A, \neg B, \Delta' \frac{\Gamma, \neg A, \neg B, \Delta'}{\Gamma, \neg(A \cap B), \Delta'}$
Inference Rules of RS1

Implication rules

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

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

Negation rule

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

where \( \Gamma' \in LT^* \), \( \Delta \in F^* \), \( A, B \in F \)
Proof System RS1

Formally we define the system $\mathbf{RS1}$ as follows

$$\mathbf{RS1} = (\mathcal{L}_{\neg, \Rightarrow, \lor, \land}, \mathcal{E}, \mathcal{LA}, \mathcal{R})$$

where

$$\mathcal{R} = \{(\lor), (\neg \lor), (\land), (\neg \land), (\Rightarrow), (\neg \Rightarrow), (\neg \neg)\}$$

for the inference rules is defined above and $\mathcal{LA}$ is the set of all logical axioms (the same as for $\mathbf{RS}$)
System RS1

Exercise

E1. Construct a proof in RS1 of a formula

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

E2. Prove that RS1 is strongly sound

E3. Define in your own words, for any formula \( A \), the decomposition tree \( T_A \) in RS1

E4. Prove Completeness Theorem for RS1
System RS1

The decomposition tree $T_A$ in RS1 is a proof of $A$ in RS1 as all leaves are axioms

$$T_A$$

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

$$\vdash (\Rightarrow)$$

$$(\neg\neg(a \cap b), (\neg a \cup \neg b)$$

$$\vdash (\cup)$$

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

$$\vdash (\neg)$$

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

$$\bigwedge (\cap)$$

$$a, \neg a, \neg b \quad \quad b, \neg a, \neg b$$
Strong Soundness of RS1

E2. 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 $\Delta$ by $\Delta'$ in the proof of strong soundness of each rule of RS to obtain the corresponding proof of strong soundness of corresponding rule of RS1.

We do it, for example for the rule $(\cup)$ of RS1 as follows.
Strong Soundness of RS1

Consider the rule \((\cup)\) of RS1

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

We evaluate:

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

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

\[
= v^*(\Gamma, (A \cup B), \Delta')
\]
Decomposition Trees in RS1

**E3.** The definition of the decomposition tree $T_A$ is again, 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 $Γ$ 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 no more decomposable formulas.

**End of Tree Construction**
Decomposition Trees in RS1

**E4.** Observe that directly from the definition of the decomposition tree $T_A$ we have that the following holds:

**Fact 1:** The decomposition tree $T_A$ is a proof iff all leaves are axioms.

**Fact 2:** The proof does not exist otherwise, i.e. $\not\models_{RS1} A$ iff there is a non-axiom leaf on $T_A$.

Fact 2 holds because the tree $T_A$ is unique.

Observe that we need Facts 1, 2 in order to prove Completeness Theorem by construction of a counter-model generated by a non-axiom leaf.
Proof of Completeness Theorem for RS1

Proof of Completeness Theorem
Assume that $A$ is any formula such that

$\not\models_{RS1} A$

By Fact 2 the decomposition tree $T_A$ contains a non-axiom leaf

The non-axiom leaf $L_A$ defines 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 that

$\not\models A$
System RS2 Definition

Definition
System RS2 is a proof system obtained from RS by changing the sequences $\Gamma'$ into $\Gamma$ in all of the rules of inference of RS. The logical axioms LA remind the same.

Observe that now the decomposition tree may not be unique.

Exercise 1
Construct two decomposition trees in RS2 of the formula

$$(\neg(\neg a \Rightarrow (a \cap \neg b)) \Rightarrow (\neg a \cap (\neg a \cup \neg b)))$$
RS2 Exercises

T1_A
\[-(\neg a \Rightarrow (a \cap \neg b)) \Rightarrow (\neg a \cap (\neg a \cup \neg b))\]
\[\vdash (\Rightarrow)\]
\[-\neg (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cap (\neg a \cup \neg b))\]
\[\vdash (\neg\neg)\]
\[-(\neg a \Rightarrow (a \cap \neg b)), (\neg a \cap (\neg a \cup \neg b))\]
\[\vdash (\Rightarrow)\]
\[-\neg a, (a \cap \neg b), (\neg a \cap (\neg a \cup \neg b))\]
\[\vdash (\neg\neg)\]
\[a, (a \cap \neg b), (\neg a \cap (\neg a \cup \neg b))\]
\[\land (\cap)\]

\[a, a, (\neg a \cap (\neg a \cup \neg b))\]
\[\land (\cap)\]

\[a, a, \neg a, (\neg a \cup \neg b)\]
\[\land (\cup)\]

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

\[\text{axiom}\]

\[a, a, (\neg a \cup \neg b)\]
\[\land (\cup)\]

\[\text{axiom}\]

\[\neg a, (\neg a \cup \neg b)\]
\[\land (\cup)\]

\[\text{axiom}\]

\[a, \neg b, (\neg a \cup \neg b)\]
RS2 Exercises

**T1**

\((\neg(\neg a \Rightarrow (a \cap \neg b)) \Rightarrow (\neg a \cap (\neg a \cup \neg b)))\)

| (\Rightarrow) |
| (\neg) |
| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cap (\neg a \cup \neg b)) |

\((\neg a \Rightarrow (a \cap \neg b)), (\neg a \cap (\neg a \cup \neg b))\)

**T1**

\((\neg a \Rightarrow (a \cap \neg b)), (\neg a \cap (\neg a \cup \neg b))\)

| (\neg a \Rightarrow (a \cap \neg b)) |
| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |

| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |
| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |

| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |
| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cap (\neg a \cup \neg b)) |

| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cap (\neg a \cup \neg b)) |
| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |

| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |
| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |

| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |
| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |

| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |
| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |

| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |
| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |

| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |
| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |

| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |
| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |

| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |
| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |

| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |
| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |

| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |
| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |

| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |
| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |

| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |
| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |

| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |
| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |

| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |
| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |

| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |
| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |

| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |
| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |

| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |
| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |

| (\neg a \Rightarrow (a \cap \neg b)), \neg a, \neg b |
| (\neg a \Rightarrow (a \cap \neg b)), (\neg a \cup \neg b) |
Exercise 2  Explain why the system RS2 is strongly sound. You can use the 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 rules as the decomposition rules while constructing the decomposition tree, but it does not affect strong soundness, since for all rules in RS2 premisses and conclusions are still logically equivalent as they were in RS
RS2 Exercises

Consider, for example, RS2 rule

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

We evaluate

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

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

Similarly, as in RS, we show all other rules of RS2 to be strongly sound, thus RS2 is sound.
Exercise 3

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 a given formula $A$ may not be unique

Solution

For a formula $A$ decomposition tree $T_A$ can be defined as following

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
RS2 Exercises

If the node consists only of literals (i.e. no decomposition rules to be applied), then it does not have any children. The last statement define a termination condition for the tree.

This definition correctly defines a decomposition tree for a formula as it identifies and uses appropriate the decomposition rules.
RS2 Exercises

Since in **RS2 all rules** of inference have a sequence $\Gamma$ instead of $\Gamma'$ as it was defined for in RS, the **choice** of the decomposition rule for a node is **not unique**

**For example** consider a **node** $(a \implies b), (b \cup a)$

$\Gamma$ in the RS2 rules is a sequence of formulas, **not literals**, so for this **node** we can choose as a **decomposition rule** either $(\implies)$ or $(\cup)$

This leads to a **non-unique tree**
RS2 Exercises

Exercise 4
Prove the Completeness Theorem for RS2

Solution
We need to prove the completeness part only, as the Soundness has been already proved, i.e. we have to prove the implication:

For any formula $A$, if $\not\vdash_{RS2} A$ then $\not|= A$

Assume $\not\vdash_{RS2} A$,

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 $A$
RS2 Exercises

Let $\mathcal{T}_A$ be a set of all decomposition trees of $A$. We choose an arbitrary $T_A \in \mathcal{T}_A$.

We choose an arbitrary $T_A \in \mathcal{T}_A$ with at least one non-axiom leaf $L_A$.

The non-axiom leaf $L_A$ defines a truth assignment $\nu$ which falsifies $A$, as follows:

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

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 $A$, i.e.

$$
\not\models A
$$
RS2 Exercises

Exercise 5  Write a procedure $TREE_A$ such that for any formula $A$ of RS2 it produces its unique decomposition tree

Procedure  $TREE_A$(Formula A, Tree T)
{
    $B = \text{ChoseLeftMostFormula}(A)$ // Choose the left most formula that is not a literal
    $c = \text{MainConnective}(B)$ // Find the main connective of B
    $R = \text{FindRule}(c)$ // Find the rule which conclusion that has this connective
    $P = \text{Premises}(R)$ // Get the premises for this rule
    $\text{AddToTree}(A, P)$ // add premises as children of A to the tree
    For all p in P // go through all premises
        $TREE_A(p, T)$ // build subtrees for each premiss
}
Exercise 6
Prove **completeness** of your **Procedure** $TREE_A$

**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.
**RS** system is **complete**, thus this **Procedure** is **complete**.