

cse541  
LOGIC for COMPUTER SCIENCE

Professor Anita Wasilewska

Chapter 6  
Automated Proof Systems  
Completeness of Classical Propositional Logic

**CHAPTER 6 SLIDES**

Chapter 6  
Automated Proof Systems  
Completeness of Classical Propositional Logic

**Lecture 6**

**PART 1: Proof System RS**

Automated Search for Proofs: Decomposition Trees

**PART 2: Proof System RS**

Strong Soundness and Constructive Completeness

**PART 3: Proof Systems RS1, RS2**

Chapter 6  
Automated Proof Systems  
Completeness of Classical Propositional Logic

**Lecture 6a**

**PART 4:** Gentzen Sequent Systems **GL, G**

Strong Soundness and Constructive Completeness

**Lecture 6b**

**PART 5:** Original Gentzen Systems **LK, LI**

Classical and Intuitionistic Completeness and Hauptatz  
Theorem

Chapter 6  
Automated Proof Systems  
Completeness of Classical Propositional Logic

**Lecture 6**

**PART 1: Proof System RS**

Automated Search for Proofs: Decomposition Trees

## Gentzen Style Proof Systems

**Hilbert style** systems are easy to **define** and admit different proofs of **Completeness Theorem**

They are **difficult** to use by humans, not mentioning **computer**

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**

## Gentzen Style Proof Systems

The **Gentzen type** systems may be **less intuitive** than the **Hilbert** systems but they allow us to **define** effective **automatic** procedures for **proof search**, what was **impossible** in a case of the **Hilbert** systems

For this reason they are called **automated proof systems**  
They serve as **formal models** of **computing** systems that **automate** the **reasoning process**

## Gentzen Style Proof Systems

The **Gentzen formalizations**, as they are also called, were **invented** by **Gerald Gentzen** in **1934**, hence the name

**Gentzen** proof systems for **classical** and **intuitionistic predicate** logics introduced special **expressions** built of formulas called **sequents**

This is why the **Gentzen style** systems using **sequents** as basic expressions are often called **Gentzen sequent formalizations**



## Gentzen Style Proof Systems

We present in **Slides Set 2** our own **Gentzen sequent** systems **GL** and **G** and prove their **completeness**

We also present a **propositional** version of **Gentzen** original system **LK** and discuss the **original proof** of **Hauptsatz Theorem**

**Hauptsatz Theorem** 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** proof system **G**

## Gentzen Style Proof Systems

A propositional version of **Gentzen** historical **original** formalization for **intuitionistic** logic **LI** is presented and discussed in **Chapter 7**

The original **classical** and **intuitionistic predicate** systems **LK** and **LI** are discussed in Chapter 9

## Gentzen Style Proof Systems

The other **historically important** automated proof systems **RS** and **QRS** are due to **Rasiowa** and **Sikorski** (1960)

Their proof systems for classical **propositional** and **predicate** logic use as basic expressions **sequences** of formulas, less complicated than **Gentzen sequents**

**Rasiowa** and **Sikorski** proof systems are simpler and easier to **understand** than the **Gentzen sequent** systems

This is one of the reasons the system **RS** is the **first** to be presented here

## Gentzen Style Proof Systems

**Historical importance** and lasting **influence** of **Rasiowa** and **Sikorski** work **lays** in the fact that they were the first to **use** the **proof searching** capacity of their proof systems to **define** a **constructive method** of proving the **completeness theorem** for both **propositional** and **predicate** classical logic

We **introduce** and **explain** in detail their **constructive method** and use it **prove** the **completeness** of the **RS** system and following systems **RS1** and **RS2**

## Gentzen Style Proof Systems

We also generalize the **RS** **constructive method** to the **Gentzen sequent** systems and **prove** the **completeness** of **GL** and **G**

The **completeness proof** for classical **predicate** system **RSQ** is presented in **Chapter 9**

# RS Proof System

# RS Proof System

## Components of RS

### Language

$$\mathcal{L}_{\{\neg, \Rightarrow, \cup, \cap\}}$$

### Expressions

We adopt as the set of expressions  $\mathcal{E}$  the set  $\mathcal{F}^*$  of all **finite sequences** of formulas

$$\mathcal{E} = \mathcal{F}^*$$

### Notation

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

$$\Gamma, \Delta, \Sigma \dots$$

with **indices** if necessary.

## RS Proof System

### Semantic Link

The 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**

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

$$\Gamma = A_1, A_2, \dots, A_n$$

we **denote**

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

We define as the next step a **formal semantics** for **RS**



## Formal Semantics for RS

### Formal Semantics

Let  $v : VAR \rightarrow \{T, F\}$  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

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

## Formal Semantics for RS

### Model

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

We write it as

$$v \models \Gamma$$

and call  $v$  a **model** for  $\Gamma$

### Counter- Model

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  $\Gamma$

## Formal Semantics for **RS**

### **Tautology**

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

We write

$$\models \Gamma$$

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

## Example

### Example

Let  $\Gamma$  be a sequence

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

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

$$v^*(\Gamma) = v^*(\delta_\Gamma) = v^*(a) \cup v^*(b \wedge a) \cup v^*(\neg b) \cup v^*(b \Rightarrow a) = F \cup (F \wedge T) \cup F \cup (T \Rightarrow F) = F \cup F \cup F \cup F = F$$

## Exercise

### Exercise

1. 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

$$v \models \Gamma$$

2. Let  $\Gamma$  be a sequence

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

Prove that

$$\models \Gamma$$

## Exercise

### Solution

1.  $\Gamma$  is a sequence

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

We evaluate

$$\begin{aligned} v^*(\Gamma) &= v^*(\delta_\Gamma) = v^*(a) \cup v^*(\neg b \wedge a) \cup v^*(\neg b) \cup v^*(a \cup b) = \\ &T \cup v^*(\neg b \wedge a) \cup v^*(\neg b) \cup v^*(a \cup b) = T \end{aligned}$$

We proved

$$v \models \Gamma$$

## Exercise

### Solution

2. 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 short-hand 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  $\Gamma$  is a **tautology**

## Rules of inference

**Rules of inference** 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 **premisses** and  $\Gamma$  is called the **conclusion** of the rule

Each rule of inference **introduces** a new logical **connective** or a **negation** of a logical **connective**

We **name** the rule 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  $\Gamma$



## Rules of inference of **RS**

### Rules of Inference

**RS** contains seven inference rules:

$(\cup)$ ,  $(\neg\cup)$ ,  $(\cap)$ ,  $(\neg\cap)$ ,  $(\Rightarrow)$ ,  $(\neg\Rightarrow)$ ,  $(\neg\neg)$

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

## Literals

### 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**

## Literals

We denote by

$$\Gamma', \Delta', \Sigma' \dots$$

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

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

We will denote by

$$\Gamma, \Delta, \Sigma \dots$$

the elements of  $\mathcal{F}^*$

## Logical Axioms of **RS**

### Logical Axioms

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  $a \in \mathit{VAR}$  is any **propositional variable**

We denote by **LA** the set of all **logical axioms** of **RS**

## Logical Axioms of **RS**

### Semantic Link

Consider axiom

$$\Gamma'_1, a, \Gamma'_2, \neg a, \Gamma'_3$$

Directly from the extension of the notion of tautology to **RS** we have that for any truth assignment  $v : VAR \rightarrow \{T, F\}$

$$\begin{aligned} v^*(\Gamma'_1, \neg a, \Gamma'_2, a, \Gamma'_3) &= v^*(\Gamma'_1) \cup v^*(\neg a) \cup v^*(a) \cup v^*(\Gamma'_2, \Gamma'_3) = \\ v^*(\Gamma'_1) \cup T \cup v^*(\Gamma'_2, \Gamma'_3) &= T \end{aligned}$$

The same applies to the axiom

$$\Gamma'_1, \neg a, \Gamma'_2, a, \Gamma'_3$$

We have thus proved the following.

**Fact** Logical axioms of **RS** are **tautologies**

## Inference Rules of **RS**

### Disjunction rules

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

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

### Conjunction rules

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

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

## Inference Rules of **RS**

### Implication rules

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

### Negation rule

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

where  $\Gamma' \in LT^*$ ,  $\Delta \in \mathcal{F}^*$ ,  $A, B \in \mathcal{F}$

## Proof System **RS**

Formally we define the system **RS** as follows

$$\mathbf{RS} = (\mathcal{L}_{\{\neg, \Rightarrow, \cup, \cap\}}, \mathcal{F}^*, \mathbf{LA}, \mathcal{R})$$

where the set of inference rules is

$$\mathcal{R} = \{(\cup), (\neg\cup), (\cap), (\neg\cap), (\Rightarrow), (\neg\Rightarrow), (\neg\neg)\}$$

and **LA** is the set of logical axioms



## Formal Proofs

### Definition

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

$$\Gamma_1, \Gamma_2, \dots, \Gamma_n$$

of sequences of formulas (elements of  $\mathcal{F}^*$ , such that

$$\Gamma_1 \in LA \quad \text{and} \quad \Gamma_n = \Gamma$$

and for all  $1 \leq i \leq n$

$\Gamma_i \in AL$ , or  $\Gamma_i$  is a **conclusion** of one of the inference rules of **RS** with all its **premisses** placed in the sequence

$$\Gamma_1 \Gamma_2, \dots, \Gamma_{i-1}$$

## Formal Proofs

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

$$\vdash \Gamma$$

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

As the proofs in **RS** are sequences (definition of the formal proof) of sequences of formulas (definition of **RS**) we will not use “;” to separate the steps of the proof, and write the **formal proof** as

$$\Gamma_1; \Gamma_2; \dots; \Gamma_n$$

## Formal Proofs

We write, however, the **formal proofs** in **RS** in a form of **trees** rather than in a form of **sequences**

We write them in form of a **tree**, where

all **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 the **tree proofs** with the **node** on the **top**, and **leafs** on the very **bottom**

We adopt hence the following definition

## Proof Trees

### Definition

By a **proof tree** in **RS** of  $\Gamma$  we understand a tree

$T_{\Gamma}$

built out of  $\Gamma \in \mathcal{E}$  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

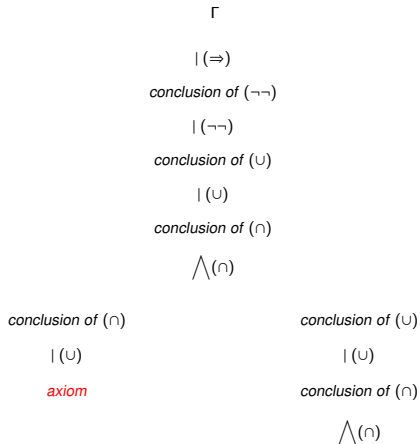
### Example

Assume that a **proof** of a sequence  $\Gamma$  from **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  $T_{\Gamma}$



## 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 proof tree  $T_A$

$$(\neg(a \wedge b) \Rightarrow (\neg a \vee \neg b))$$

$$| (\Rightarrow)$$

$$\neg\neg(a \wedge b), (\neg a \vee \neg b)$$

$$| (\neg\neg)$$

$$(a \wedge b), (\neg a \vee \neg b)$$

$$\wedge (\wedge)$$

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

$$| (\vee)$$

$$a, \neg a, \neg b$$

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

$$| (\vee)$$

$$b, \neg a, \neg b$$



## Formal Proof

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

$$a, \neg a, \neg b$$

$$b, \neg a, \neg b$$

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

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

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

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

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

## Example

### 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  $T_A$

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

$$| (\Rightarrow)$$

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

$$| (\neg\neg)$$

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

$$| (\cup)$$

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

$$\bigwedge (\cap)$$

$a, b, \neg a$

$a, b, \neg b$

## Example

We construct its **formal proof**, as before, written in a vertical manner as follows

$$a, b, \neg b$$

$$a, b, \neg a$$

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

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

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

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

## Decomposition Trees

The **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**

A **decomposition tree**  $T_A$  for the formula

$$A = (((a \Rightarrow b) \wedge \neg c) \cup (a \Rightarrow c))$$

is built as follows

## Decomposition Trees

$T_A$

$$(((a \Rightarrow b) \wedge \neg c) \vee (a \Rightarrow c))$$

$\vee (\vee)$

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

$\wedge (\wedge)$

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

$\vee (\Rightarrow)$

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

$\vee (\Rightarrow)$

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

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

$\vee (\Rightarrow)$

$$\neg c, \neg a, c$$

# RS Decomposition Rules and Decomposition Trees

## Decomposition Trees

The process of **searching for a proof** of a formula  $A \in \mathcal{F}$  in **RS** consists of building a certain tree  $T_A$ , called a **decomposition tree**

Building a **decomposition tree** what really is a **proof search tree** consists in the **first step** of **transforming** the **RS rules** into corresponding **decomposition rules**



## Decomposition Rules

### RS Decomposition Rules

#### Disjunction

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

#### Conjunction

$$(\cap) \frac{\Gamma', (A \cap B), \Delta}{\Gamma', A, \Delta ; \Gamma', B, \Delta}, \quad (\neg \cap) \frac{\Gamma', \neg(A \cap B), \Delta}{\Gamma', \neg A, \neg B, \Delta}$$

## Decomposition Rules

### Implication

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

### Negation

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

where  $\Gamma' \in \mathcal{F}'^*$ ,  $\Delta \in \mathcal{F}^*$ ,  $A, B \in \mathcal{F}$

## Tree Rules

We write the **Decomposition Rules** in a **visual tree** form as follows

### Tree Rules

( $\cup$ ) rule

$$\Gamma', (A \cup B), \Delta$$
$$| (\cup)$$
$$\Gamma', A, B, \Delta$$

## Tree Rules

$(\neg\cup)$  rule

$$\Gamma', \neg(A \cup B), \Delta$$

$$\bigwedge(\neg\cup)$$

$(\cap)$  rule

$$\Gamma', \neg A, \Delta \quad \Gamma', \neg B, \Delta$$

$$\Gamma', (A \cap B), \Delta$$

$$\bigwedge(\cap)$$

$$\Gamma', A, \Delta \quad \Gamma', B, \Delta$$

## Tree Rules

$(\neg\cup)$  rule

$$\Gamma', \neg(A \cap B), \Delta$$
$$| (\neg\cap)$$
$$\Gamma', \neg A, \neg B, \Delta$$

$(\Rightarrow)$  rule

$$\Gamma', (A \Rightarrow B), \Delta$$
$$| (\Rightarrow)$$
$$\Gamma', \neg A, B, \Delta$$

## Tree Rules

$(\neg \Rightarrow)$  rule

$$\Gamma', \neg(A \Rightarrow B), \Delta$$

$$\bigwedge (\neg \Rightarrow)$$

$$\Gamma', A, \Delta$$

$$\Gamma', \neg B, \Delta$$

$(\neg \neg)$  rule

$$\Gamma', \neg \neg A, \Delta$$

$$\mid (\neg \neg)$$

$$\Gamma', A, \Delta$$

## Definitions and Observations

**Observe** that we use the same **names** for the **inference** and **decomposition** rules

We do so because 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.

## Definitions and Observations

### Definition

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

### Definition

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

### Definition

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  $(\neg\cup), (\neg\cap), (\neg\neg), (\neg\Rightarrow)$

## Decomposition Lemma

Because of the importance of the **Observation 1** we re-write it in a form of the following

### Decomposition Lemma

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

$\Gamma \in LT^*$  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 **root** of  $T_A$

For any other **node**  $\Gamma$  of the tree we follow the steps below

**Step 2.**

If  $\Gamma$  is **indecomposable** then  $\Gamma$  becomes a **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.  $\not\vdash_{\text{RS}} \Gamma$  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

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

$$| (\cup)$$

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

## 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

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

$$| (\cup)$$

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

$$| (\Rightarrow)$$

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

## 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

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

$$| (\cup)$$

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

$$| (\Rightarrow)$$

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

$$\bigwedge (\neg \cup)$$

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

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

## Example

Finally, the complete  $T_A$  is

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

$$| (\cup)$$

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

$$| (\Rightarrow)$$

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

$$\bigwedge (\neg \cup)$$

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

$$| (\Rightarrow)$$

$$\neg a, \neg a, \neg\neg a, \neg c$$

$$| (\neg\neg)$$

$$\neg a, \neg a, a, \neg c$$

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

$$| (\Rightarrow)$$

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

$$| (\neg\neg)$$

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

## Example

All leaves of  $T_A$  are **axioms**

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

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

## Example

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

$$(((a \Rightarrow b) \wedge \neg c) \vee (a \Rightarrow c))$$

$$| (\vee)$$

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

$$\wedge (\wedge)$$

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

$$| (\Rightarrow)$$

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

$$| (\Rightarrow)$$

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

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

$$| (\Rightarrow)$$

$$\neg c, \neg a, c$$



## 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**

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

It means that the proof in **RS** of the formula  $((a \Rightarrow b) \wedge \neg c) \cup (a \Rightarrow c)$  **does not exist**

## Completeness Theorem

Our main goal is to prove the **Completeness Theorem** for **RS**

We **prove** first the following **Completeness Theorem** for formulas  $A \in \mathcal{F}$

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

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

and then we generalize it to the following

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

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

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

## Part 2: Strong Soundness and Constructive Completeness

## Strong Soundness

### Definition

Given a proof system

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

### 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** if and only if  $S$  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 (U)

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

We evaluate:

$$\begin{aligned} v^*(\Gamma', A, B, \Delta) &= v^*(\delta_{\{\Gamma', A, B, \Delta\}}) = v^*(\Gamma') \cup v^*(A) \cup v^*(B) \cup v^*(\Delta) \\ &= v^*(\Gamma') \cup v^*(A \cup B) \cup v^*(\Delta) = v^*(\delta_{\{\Gamma', (A \cup B), \Delta\}}) \\ &= v^*(\Gamma', (A \cup B), \Delta) \end{aligned}$$

## Strong Soundness of RS

Consider the rule  $(\neg\cup)$

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

We evaluate:

$$\begin{aligned} v^*(P_1) \cap v^*(P_2) &= v^*(\Gamma', \neg A, \Delta) \cap v^*(\Gamma', \neg B, \Delta) \\ &= (v^*(\Gamma') \cup v^*(\neg A) \cup v^*(\Delta)) \cap (v^*(\Gamma') \cup v^*(\neg B) \cup v^*(\Delta)) \\ &= (v^*(\Gamma', \Delta) \cup v^*(\neg A)) \cap (v^*(\Gamma', \Delta) \cup v^*(\neg B)) \\ &=^{\text{distrib}} (v^*(\Gamma', \Delta) \cup (v^*(\neg A) \cap v^*(\neg B))) \\ &= v^*(\Gamma') \cup v^*(\Delta) \cup v^*(\neg A \cap \neg B) =^{\text{deMorgan}} v^*(\delta_{\{\Gamma', \neg(A \cup B), \Delta\}}) \\ &= v^*(\Gamma', \neg(A \cup B), \Delta) = v^*(C) \end{aligned}$$

## 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  $\Gamma \in \mathcal{F}^*$ ,

If  $\vdash_{\mathbf{RS}} \Gamma$ , then  $\models \Gamma$

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

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



## 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 hence 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<sub>A</sub>** 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** 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  $A \in \mathcal{F}$  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  $T_A$

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

$$| (\cup)$$

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

$$\wedge (\cap)$$

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

$$| (\Rightarrow)$$

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

$$| (\Rightarrow)$$

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

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

$$| (\Rightarrow)$$

$$\neg c, \neg a, c$$

## Counter Model Example

The tree  $T_A$  has a **non-axiom leaf**

$$L_A : \neg a, b, \neg a, c$$

We want to define a truth assignment  $v : VAR \rightarrow \{T, F\}$   
**falsifies** this leaf  $L_A$

Observe that  $v$  must be such that

$$\begin{aligned} v^*(\neg a, b, \neg a, c) &= v^*(\neg a) \cup v^*(b) \cup v^*(\neg a) \cup v^*(c) = \\ &\neg v(a) \cup v(b) \cup \neg v(a) \cup v(c) = F \end{aligned}$$

It means that **all components** of the **disjunction** must be put to **F**

## Counter Model Example

We hence get that  $v$  must be such that

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

By the **Counter Model Theorem**, the  $v$  **determined** by the **non-axiom leaf** also **falsifies** the formula **A**

IT proves that  $v$  is a **counter model** for **A** and

$$\not\models (((a \Rightarrow b) \wedge \neg c) \vee (a \Rightarrow c))$$

## Counter Model

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

$$(((a \Rightarrow b) \wedge \neg c) \vee (a \Rightarrow c)) = \mathbf{F}$$

| ( $\vee$ )

$$((a \Rightarrow b) \wedge \neg c), (a \Rightarrow c) = \mathbf{F}$$

$\wedge$  ( $\wedge$ )

$$(a \Rightarrow b), (a \Rightarrow c) = \mathbf{F}$$

| ( $\Rightarrow$ )

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

| ( $\Rightarrow$ )

$$\neg a, b, \neg a, c = \mathbf{F}$$

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

| ( $\Rightarrow$ )

$$\neg c, \neg a, c$$

*axiom*

## 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

### 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 is such that

$$\not\models_{RS} A$$

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

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

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

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

$$\not\models A$$

PART3:  
Proof Systems **RS1** and **RS2**

## RS1 Proof System

Proof System **RS1**

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

$$\mathcal{L} = \mathcal{L}_{\{\neg, \Rightarrow, \cup, \cap\}}$$

**Expressions**

$$\mathcal{E} = \mathcal{F}^*$$

is the set of **expressions** of **RS1**

**Notation**

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

$$\Gamma, \Delta, \Sigma \dots$$

with indices if necessary.

## Rules of inference of RS1

### Rules of inference

**RS1** contains **seven inference rules**, denoted by the same symbols as the rules of **RS**

$(\cup)$ ,  $(\neg\cup)$ ,  $(\cap)$ ,  $(\neg\cap)$ ,  $(\Rightarrow)$ ,  $(\neg\Rightarrow)$ ,  $(\neg\neg)$

The inference rules of **RS1** are quite **similar** to the rules of **RS**  
Observe them **carefully** to see where lies the **difference**

### Reminder

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

The set

$$LT = VAR \cup \{\neg a : a \in VAR\}$$

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

## Literals Notation

We denote, as before, by

$$\Gamma', \Delta', \Sigma' \dots$$

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

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

We will denote by

$$\Gamma, \Delta, \Sigma \dots$$

the elements of  $\mathcal{F}^*$

## Logical Axioms

### Logical Axioms

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

$$\Gamma'_1, a, \Gamma'_2, \neg a, \Gamma'_3$$

$$\Gamma'_1, \neg a, \Gamma'_2, a, \Gamma'_3$$

where  $a \in VAR$  is any **propositional variable**

## Inference Rules of RS1

### Disjunction rules

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

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

### Conjunction rules

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

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

## Inference Rules of **RS1**

### Implication rules

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

$$(\neg \Rightarrow) \frac{\Gamma, A, \Delta' : \Gamma, \neg B, \Delta'}{\Gamma, \neg(A \Rightarrow B), \Delta'}$$

### Negation rule

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

where  $\Gamma' \in LT^*$ ,  $\Delta \in \mathcal{F}^*$ ,  $A, B \in \mathcal{F}$



## Proof System **RS1**

Formally we define the system **RS1** as follows

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

where

$$\mathcal{R} = \{(\cup), (\neg \cup), (\cap), (\neg \cap), (\Rightarrow), (\neg \Rightarrow), (\neg \neg)\}$$

for the **inference rules** is defined above and **LA** is the set of all logical **axioms** is the same as for **RS**

## System **RS1**

### Exercises

**E1.** **Construct** a proof in **RS1** of a formula

$$A = (\neg(a \wedge b) \Rightarrow (\neg a \vee \neg b))$$

**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**

## Exercises Solutions

**E1.** The decomposition tree  $\mathbf{T}_A$  is a **proof** of **A** in **RS1** as **all leaves are axioms**

$\mathbf{T}_A$

$$(\neg(a \wedge b) \Rightarrow (\neg a \vee \neg b))$$

| ( $\Rightarrow$ )

$$(\neg\neg(a \wedge b), (\neg a \vee \neg b))$$

| ( $\vee$ )

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

| ( $\neg\neg$ )

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

$\wedge$  ( $\wedge$ )

$a, \neg a, \neg b$

$b, \neg a, \neg b$

## Exercises Solutions

**E2.** Prove that **RS1** is **strongly sound**

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 the **proof** of **strong soundness** of each rule of **RS** to obtain the **corresponding proof** of **strong soundness** of corresponding rule of **RS1**

## Strong Soundness of **RS1**

We do it, for example for the rule **(U)** as follows

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

We evaluate:

$$\begin{aligned} v^*(\Gamma, A, B, \Delta') &= v^*(\delta_{\{\Gamma, A, B, \Delta'\}}) = v^*(\Gamma) \cup v^*(A) \cup v^*(B) \cup v^*(\Delta') \\ &= v^*(\Gamma) \cup v^*(A \cup B) \cup v^*(\Delta') = v^*(\delta_{\{\Gamma, (A \cup B), \Delta'\}}) \\ &= v^*(\Gamma, (A \cup B), \Delta') \end{aligned}$$

## Decomposition Trees in RS1

**E3.** Define in your own words, for any formula  $A$ , the decomposition tree  $T_A$  in RS1

The **definition** of the decomposition tree  $T_A$  is in its **essence** similar to the one for **RS** except for the **changes** which reflect the **differences** in the corresponding **rules** of inference

## Decomposition Trees in RS1

### Definition

To construct the decomposition tree  $T_A$  we follow the steps below

### Step 1

**Decompose** formula  $A$  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 the decomposition tree **construction**

## Completeness Theorem for **RS1**

**E4. Prove** the following **Completeness Theorem**

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

If  $\models A$ , then  $\vdash_{\text{RS1}} A$

We prove instead the **opposite implication**

**Completeness Theorem**

If  $\not\vdash_{\text{RS1}} A$  then  $\not\models A$



## Completeness Theorem for **RS1**

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

**Fact 1:** The decomposition tree  $T_A$  is a **proof** if and only if **all leaves are axioms**

**Fact 2:** The **proof does not exist** otherwise, i.e.

$\not\models_{RS1} A$  if and only if **there is a non- axiom leaf** on  $T_A$

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

## Proof of Completeness Theorem for **RS1**

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

### **Proof**

Assume that **A** is any formula such that

$$\not\models_{\text{RS1}} A$$

By **Fact 2** the decomposition tree **T<sub>A</sub>** contains a non-axiom leaf **L<sub>A</sub>**

We use the non-axiom leaf **L<sub>A</sub>** 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 that

$$\not\models A$$

## System **RS2** Definition

### **RS2** 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

$$(\neg(\neg a \Rightarrow (a \wedge \neg b)) \Rightarrow (\neg a \wedge (\neg a \vee \neg b)))$$

# RS2 Exercises

**T1<sub>A</sub>**

$$(\neg(\neg a \Rightarrow (a \wedge \neg b)) \Rightarrow (\neg a \wedge (\neg a \vee \neg b)))$$

| ( $\Rightarrow$ )

$$\neg(\neg a \Rightarrow (a \wedge \neg b)), (\neg a \wedge (\neg a \vee \neg b))$$

| ( $\neg\neg$ )

$$(\neg a \Rightarrow (a \wedge \neg b)), (\neg a \wedge (\neg a \vee \neg b))$$

| ( $\Rightarrow$ )

$$\neg\neg a, (a \wedge \neg b), (\neg a \wedge (\neg a \vee \neg b))$$

| ( $\neg\neg$ )

$$a, (a \wedge \neg b), (\neg a \wedge (\neg a \vee \neg b))$$

$\wedge$ ( $\wedge$ )

$$a, a, (\neg a \wedge (\neg a \vee \neg b))$$

$\wedge$ ( $\wedge$ )

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

| ( $\vee$ )

$$a, a, \neg a, \neg a, \neg b$$

*axiom*

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

| ( $\vee$ )

$$a, a, \neg a, \neg b$$

*axiom*

$$a, \neg b, (\neg a \wedge (\neg a \vee \neg b))$$

$\wedge$ ( $\wedge$ )

$$a, \neg b, \neg a$$

*axiom*

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

| ( $\vee$ )

$$a, \neg b, \neg a, \neg b$$

*axiom*

## RS2 Exercises

$T2_A$

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

| ( $\Rightarrow$ )

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

| ( $\neg\neg$ )

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

$\bigwedge$ ( $\cap$ )

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

| ( $\Rightarrow$ )

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

| ( $\neg\neg$ )

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

$\bigwedge$ ( $\cap$ )

$a, a, \neg a$

*axiom*

$a, \neg b, \neg a$

*axiom*

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

| ( $\cup$ )

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

| ( $\Rightarrow$ )

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

| ( $\neg\neg$ )

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

$\bigwedge$ ( $\cap$ )

$a, a, \neg a, \neg b$

*axiom*

$a, \neg b, \neg a, \neg b$

*axiom*

## System **RS2**

### Exercise 2

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

### Solution

The **only** 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 of **RS2** premisses and conclusions are still **logically equivalent** as they were in **RS**

## RS2 Exercises

Consider, for example, **RS2** rule

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

We evaluate

$$\begin{aligned} v^*(\Gamma, A, B, \Delta) &= v^*(\Gamma) \cup v^*(A) \cup v^*(B) \cup v^*(\Delta) = \\ v^*(\Gamma) \cup v^*(A \cup B) \cup v^*(\Delta) &= v^*(\Gamma, (A \cup B), \Delta) \end{aligned}$$

Similarly, as in **RS**, we show all other rules of **RS2** to be **strongly sound**, thus **RS2** is also **strongly sound**

## RS2 Exercises

### 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 some formula  $A$  may **not be unique**



## RS2 Exercises

### 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**

## 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 defines a **termination condition** for the **tree**

This definition **correctly** defines a **decomposition tree** 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 may be **not unique**

Fore**example** consider a **node**

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

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

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  $\not\models A$

**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  $T_A \in \mathcal{T}_A$  with at least one non-axiom leaf  $L_A$

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

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

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**  $TREE_A(\text{Formula } A, \text{Tree } T)$

```
{  
     $B = \text{ChoseLeftMostFormula}(A)$  // 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  
}
```

## RS2 Exercises

### 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**