LOGICS FOR COMPUTER SCIENCE: Classical and Non-Classical

Anita Wasilewska
Chapter 10
Predicate Automated Proof Systems
Completeness of Classical Predicate Logic

CHAPTER 10 SLIDES
Chapter 10
Predicate Automated Proof Systems
Completeness of Classical Predicate Logic

Slides Set 1
PART 1: QRS Proof System
PART 2: Proof of QRS Completeness

Slides Set 2
PART 3: Skolemization and Clauses
Chapter 10
Predicate Automated Proof Systems
Completeness of Classical Predicate Logic

Slides Set 1
PART 1: QRS Proof System
Predicate Automated Proof Systems

Introduction

We define and discuss here Rasiowa and Sikorski Gentzen style proof system QRS for classical predicate logic.

The propositional version of it, the RS proof system, was studied in detail in chapter 6.

These both proof systems RS and QRS admit a constructive proof of completeness theorem.
Predicate Automated Proof Systems

Introduction

We adopt Rasiowa, Sikorski (1961) technique of construction a counter model determined by a decomposition tree to prove **QRS** completeness theorem.

The proof, presented here is a generalization of the completeness proofs of **RS** and other Gentzen style propositional systems presented in details in chapter 6.

We refer the reader to the chapter 6 as it provides a good introduction to the subject.
Predicate Automated Proof Systems
Introduction

The other Gentzen type predicate proof system, including the original Gentzen proof systems $\text{LK}$, $\text{LI}$ for classical and intuitionistic predicate logics are obtained from their propositional versions discussed in detail in chapter 6 by adding the Quantifiers Rules to them.

It can be done in a similar way as a generalization of the propositional $\text{RS}$ to the predicate $\text{QRS}$ system presented here.

We leave these generalizations as an exercises for the reader.
Predicate Automated Proof Systems
Introduction

We also leave as an exercise the predicate language version of Gentzen proof of cut elimination theorem, Hauptzatz (1935).

The Hauptzatz proof for the predicate classical $\text{LK}$ and intuitionistic $\text{LI}$ systems is easily obtained from the propositional proof included in chapter 6.

There are of course other types of automated proof systems based on different methods of deduction.
There is a **Natural Deduction** mentioned by Gentzen in his Hauptzatz paper in 1935.

It was later and fully developed by Dag Prawitz (1965). It is now called Prawitz, or Gentzen-Prawitz Natural Deduction.

There is a **Semantic Tableaux** deduction method invented by Evert Beth (1955).

It was consequently simplified and further developed by Raymond Smullyan (1968). It is now often called Smullyan Semantic Tableaux.
Finally, there is **Resolution**

The resolution method can be traced back to **Davis and Putnam (1960)**
Their work is still known as **Davis-Putnam method**

The difficulties of **Davis-Putnam method** were eliminated by **John Alan Robinson (1965)**
He consequently developed it into what we call now **Robinson Resolution**, or just **Resolution**
Predicate Automated Proof Systems

Introduction

The resolution proof system for propositional or predicate logic operates on a set of clauses as a basic expressions and uses a resolution rule as the only rule of inference.

We define and prove correctness of effective procedures of converting any formula $A$ into a corresponding set of clauses in both propositional and predicate cases.
QRS Proof System
QRS Proof System

The components of the proof system QRS are as follows

**Language** $\mathcal{L}$

\[
\mathcal{L} = \mathcal{L}_{\{\cap, \cup, \Rightarrow, \neg\}}(\mathcal{P}, \mathcal{F}, \mathcal{C})
\]

for $\mathcal{P}, \mathcal{F}, \mathcal{C}$ countably infinite sets of predicate, functional, and constant symbols respectively

**Expressions** $\mathcal{E}$

Let $\mathcal{F}$ denote a set of formulas of $\mathcal{L}$. We adopt as the set of expressions the set of all finite sequences of formulas, i.e.

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

We will denote the expressions of QRS, i.e. the finite sequences of formulas by

$\Gamma, \Delta, \Sigma$, with indices if necessary
Rules of Inference of QRS

The system QRS consists of two axiom schemas and eleven rules of inference.

The rules of inference form two groups:

First group is similar to the propositional case and contains propositional connectives rules:

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

Second group deals with the quantifiers and consists of four rules:

\((\forall), (\exists), (\neg \forall), (\neg \exists)\)
Logical Axioms of RS

We adopt as *logical axioms* of QRS any sequence of formulas which contains a *formula* 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 \( A \in \mathcal{F} \) is any *formula*

We denote by LA the set of all *logical axioms* of QRS
Proof System QRS

Formally we define the system QRS as follows

\[ \text{QRS} = (\mathcal{L}\{\cap, \cup, \Rightarrow, \neg\}(P, F, C), \ F^*, \ LA, \ \mathcal{R}) \]

where the set \( \mathcal{R} \) of inference rules contains the following rule

(\cup), (\neg \cup), (\cap), (\neg \cap), (\Rightarrow), (\neg \Rightarrow), (\neg \neg), (\forall), (\exists), (\neg \forall), (\neg \exists)

and LA is the set of all logical axioms defined on previous slide
Literals in QRS

Definition
Any atomic formula, or a negation of atomic formula is called a literal.

We form, as in the propositional case, a special subset

\[ LT \subseteq F \]

of formulas, called a set of all literals defined now as follows

\[ LT = \{ A \in F : A \in AF \} \cup \{ \neg A \in F : A \in AF \} \]

The elements of the set \( \{ A \in F : A \in AF \} \) are called positive literals.

The elements of the set \( \{ \neg A \in F : A \in AF \} \) are called negative literals.
Sequences of 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 \( F^* \)
Connectives Inference Rules of QRS

Group 1
Disjunction rules

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

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

Conjunction rules

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

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

where \( \Gamma' \in LT^*, \Delta \in F^*, A, B \in F^* \)
Connectives Inference Rules of QRS

Group 1
Implication rules

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

Negation rule

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

where \( \Gamma' \in LT^*, \Delta \in \mathcal{F}^*, A, B \in \mathcal{F} \)
Quantifiers Inference Rules of QRS

Group 2: Universal Quantifier rules

\[ \forall \quad \frac{\Gamma', A(y), \Delta}{\Gamma', \forall x A(x), \Delta} \quad (\neg \forall) \quad \frac{\Gamma', \neg \forall x A(x), \Delta}{\Gamma', \exists x \neg A(x), \Delta} \]

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

The variable \( y \) in rule \((\forall)\) is a free individual variable which does not appear in any formula in the conclusion, i.e. in any formula in the sequence \( \Gamma', \forall x A(x), \Delta \)

The variable \( y \) in the rule \((\forall)\) is called the eigenvariable

All occurrences] of \( y \) in \( A(y) \) of the rule \((\forall)\) are fully indicated
Quantifiers Inference Rules of QRS

Group 2: Existential Quantifier rules

\[(\exists) \quad \frac{\Gamma', A(t), \Delta, \exists x A(x)}{\Gamma', \exists x A(x), \Delta} \quad \text{ and } \quad (\neg \exists) \quad \frac{\Gamma', \neg \exists x A(x), \Delta}{\Gamma', \forall x \neg A(x), \Delta}\]

where \( t \in T \) is an arbitrary term, \( \Gamma' \in LT^* \), \( \Delta \in F^* \), \( A, B \in F \)

Note that \( A(t), A(y) \) denotes a formula obtained from \( A(x) \) by writing the term \( t \) or \( y \), respectively, in place of all occurrences of \( x \) in \( A \).
Proofs and Proof Trees

By a **formal proof** of a sequence $\Gamma$ in the proof system $\text{QRS}$ we understand any sequence

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

of sequences of formulas (elements of $F^*$), such that

1. $\Gamma_1 \in LA$, $\Gamma_n = \Gamma$, and
2. for all $i$ ($1 \leq i \leq n$), $\Gamma_i \in LA$, or $\Gamma_i$ is a conclusion of one of the inference rules of $\text{QRS}$ with all its premisses placed in the sequence $\Gamma_1, \Gamma_2, \ldots, \Gamma_{i-1}$
Proofs and Proof Trees

We write, as usual,

\[ \vdash_{QRS} \Gamma \]

to denote that the sequence \( \Gamma \) has a formal proof in QRS

As the proofs in QRS are sequences (definition of the formal proof) of sequences of formulas (definition of expressions \( E \)) we will not use ”;” to separate the steps of the proof, and write the formal proof as

\[ \Gamma_1; \Gamma_2; \ldots; \Gamma_n \]
Proofs and Proof Trees

We write, however, the formal proofs in QRS as we did the propositional case (chapter 6), in a form of trees rather than in a form of sequences.

We adopt hence the following definition

**Proof Tree**

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

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

We picture, and write the proof trees with the root on the top, and leaves on the very bottom. In particular cases, as in the propositional case, we write the proof trees indicating additionally the name of the inference rule used at each step of the proof.

For example, when in a proof of a formula $A$ we use subsequently the rules

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

we represent the proof of $A$ as the following tree
Proof Trees

\[ T_A \]

Formula A

| (⇒)

conclusion of (∀)

| (∀)

conclusion of (¬¬)

| (¬¬)

conclusion of (∩)

\[ \bigwedge (\cap) \]

conclusion of (∀)

| (∀)

axiom

conclusion of (∪)

| (∪)

axiom

conclusion of (∪)

| (∪)

axiom

conclusion of (∩)

\[ \bigwedge (\cap) \]
Decomposition Trees

The main advantage of the Gentzen type proof systems lies in the way we are able to search for proofs in them.

Moreover, such proof search happens to be deterministic and automatic.

We conduct proof search by treating inference rules as decomposition rules (see chapter 6) and by building decomposition trees.

A general principle of building decomposition trees is the following.
Decomposition Trees

Decomposition Tree \( T_\Gamma \)

For each \( \Gamma \in \mathcal{F}^* \), a decomposition tree \( T_\Gamma \) is a tree build as follows:

**Step 1.** The sequence \( \Gamma \) is the **root** of \( T_\Gamma \).

For any node \( \Delta \) of the tree we follow the steps below:

**Step 2.** If \( \Delta \) is **indecomposable** or an **axiom**, then \( \Delta \) becomes a **leaf** of the tree.
Decomposition Trees

**Step 3.** If $\Delta$ is decomposable, then we traverse $\Delta$ from left to right to identify the first decomposable formula $B$ and identify inference rule treated as decomposition rule that is determined uniquely by $B$.

We put its left and right premisses as the left and right leaves, respectively.

**Step 4.** We repeat steps 2. and 3. until we obtain only leaves or an infinite branch.

In particular case when when $\Gamma$ has only one element, namely a a formula $A \in \mathcal{F}$, we call it a decomposition tree of $A$ and denote by $T_A$. 
QRS  Decomposition Trees

Given a formula  $A \in F$, we define its decomposition tree $T_A$ as follows.

Observe that the inference rules of QRS can be divided in two groups: propositional connectives rules

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

and quantifiers rules

$$(\forall), (\exists), (\neg \forall), (\neg \exists)$$

We define the decomposition tree in the case of the propositional rules and the quantifiers rules $(\neg \forall), (\neg \exists)$ in the same way as for the propositional language (chapter 6).
QRS Decomposition Trees

The case of the rules $(\forall)$ and $(\exists)$ is more complicated, as the rules contain the specific conditions under which they are applicable.

To define the way of decomposing the sequences of the form

\[ \Gamma', \forall x A(x), \Delta \quad \text{or} \quad \Gamma', \exists x A(x), \Delta, \]

i.e. to deal with the rules quantifiers rules $(\forall)$ and $(\exists)$ we assume that all terms form a one-to one sequence

\[ ST \quad t_1, t_2, \ldots, t_n, \ldots \]

Observe, that by the definition, all free variables are terms, hence all free variables appear in the sequence $ST$ of all terms.
Let $\Gamma$ be a sequence on the tree in which the first indecomposable formula has the quantifier $\forall$ as its main connective. It means that $\Gamma$ is of the form

$$\Gamma', \forall_x A(x), \Delta$$

We write a sequence

$$\Gamma', A(y), \Delta$$

below $\Gamma$ on the tree as its child, where the variable $y$ fulfills the following condition

**Condition 1:** the variable $y$ is the first free variable in the sequence $ST$ of terms such that $y$ does not appear in any formula in $\Gamma', \forall x A(x), \Delta$

Observe, that the condition the **Condition 1** corresponds to the restriction put on the application of the rule $(\forall)$
QRS Decomposition Trees

Let now the first indecomposable formula in $\Gamma$ has the quantifier $\exists$ as its main connective. It means that $\Gamma$ is of the form

$$\Gamma', \exists x A(x), \Delta$$

We write a sequence

$$\Gamma', A(t), \Delta, \exists x A(x)$$

as its child, where the term $t$ fulfills the following condition

**Condition 2:** the term $t$ is the first term in the sequence $ST$ of all terms such that the formula $A(t)$ does not appear in any sequence on the tree which is placed above

$$\Gamma', A(t), \Delta, \exists x A(x)$$
Observe that the sequence ST of all terms is one-to-one and by the **Condition 1** and **Condition 2** we always chose the first appropriate term (variable) from the sequence ST.

Hence the decomposition tree definition guarantees that the decomposition process is also unique in the case of the quantifier rules (∀) and (∃).

From all above, and we conclude the following.
Uniqueness Theorem
For any formula $A \in \mathcal{F}$,

(i) the decomposition tree $T_A$ is unique

(ii) Moreover, the following conditions hold

1. If the decomposition tree $T_A$ is finite and all its leaves are axioms, then

\[ \vdash_{QRS} A \]

2. If $T_A$ is finite and contains a non-axiom leaf, or $T_A$ is infinite, then

\[ \not\vdash_{QRS} A \]
Examples of Decomposition Trees

In all the examples below, the formulas $A(x), \ B(x)$ represent any formulas.
But as there is no indication about their particular components, they are treated as indecomposable formulas.

For example, the decomposition tree of the formula $A$ representing the de Morgan Law

$$(\neg \forall x A(x) \Rightarrow \exists x \neg A(x))$$

is constructed as follows.
Examples of Decomposition Trees

\[ T_A \]

\[ (\neg \forall x A(x) \Rightarrow \exists x \neg A(x)) \]

\[ (\Rightarrow) \]

\[ \neg \neg \forall x A(x), \exists x \neg A(x) \]

\[ (\neg \neg) \]

\[ \forall x A(x), \exists x \neg A(x) \]

\[ (\forall) \]

\[ A(x_1), \exists x \neg A(x) \]

where \( x_1 \) is a first free variable in the sequence \( ST \) such that \( x_1 \) does not appear in

\[ \forall x A(x), \exists x \neg A(x) \]

\[ (\exists) \]

\[ A(x_1), \neg A(x_1), \exists x \neg A(x) \]

where \( x_1 \) is the first term (variables are terms) in the sequence \( ST \) such that \( \neg A(x_1) \) does not appear on a tree above \( A(x_1), \neg A(x_1), \exists x \neg A(x) \)

Axiom
Examples of Decomposition Trees

The above tree $T_A$ ended with one leaf being **axiom**, so it represents a **proof** in **QRS** of the **de Morgan Law**

$$(\neg \forall x A(x) \Rightarrow \exists x \neg A(x))$$

and we have proved that

$$\vdash (\neg \forall x A(x) \Rightarrow \exists x \neg A(x))$$

The decomposition tree $T_A$ for a formula

$$(\forall x A(x) \Rightarrow \exists x A(x))$$

is constructed as follows
Examples of Decomposition Trees

\[ T_A \]

\[(\forall x A(x) \Rightarrow \exists x A(x))\]

\[
| (\Rightarrow) \\
\neg \forall x A(x), \exists x A(x) \\
| (\neg \forall) \\
\neg \forall x A(x), \exists x A(x) \\
\exists x \neg A(x), \exists x A(x) \\
| (\exists) \\
\neg A(t_1), \exists x A(x), \exists x \neg A(x)\
\]

where \( t_1 \) is the first term in the sequence \( ST \), such that \( \neg A(t_1) \) does not appear on the tree above \( \neg A(t_1), \exists x A(x), \exists x \neg A(x) \)

\[
| (\exists) \\
\neg A(t_1), A(t_1), \exists x \neg A(x), \exists x A(x)\
\]

where \( t_1 \) is the first term in the sequence \( ST \), such that \( A(t_1) \) does not appear on the tree above \( \neg A(t_1), A(t_1), \exists x \neg A(x), \exists x A(x) \)

Axiom
Examples of Decomposition Trees

The above tree also ended with the only leaf being the axiom, hence we have proved that

$$\vdash (\forall x A(x) \Rightarrow \exists x A(x))$$

We know that the the inverse implication

$$(\exists x A(x) \Rightarrow \forall x A(x))$$

is not a predicate tautology

Let’s now look at its decomposition tree $T_A$
Examples of Decomposition Trees

\[ T_A \]

\[ \exists x A(x) \]

\[ \mid (\exists) \]

\[ A(t_1), \exists x A(x) \]

where \( t_1 \) is the first term in the sequence ST, such that \( A(t_1) \) does not appear on the tree above \( A(t_1), \exists x A(x) \)

\[ \mid (\exists) \]

\[ A(t_1), A(t_2), \exists x A(x) \]

where \( t_2 \) is the first term in the sequence ST, such that \( A(t_2) \) does not appear on the tree above \( A(t_1), A(t_2), \exists x A(x) \), i.e. \( t_2 \neq t_1 \)

\[ \mid (\exists) \]

\[ A(t_1), A(t_2), A(t_3), \exists x A(x) \]

where \( t_3 \) is the first term in the sequence ST, such that \( A(t_3) \) does not appear on the tree above \( A(t_1), A(t_2), A(t_3), \exists x A(x) \), i.e. \( t_3 \neq t_2 \neq t_1 \)

\[ \mid (\exists) \]
Examples of Decomposition Trees

We continue the decomposition

\[
\begin{align*}
\vert (\exists) \\
A(t_1), A(t_2), A(t_3), A(t_4), \exists x A(x)
\end{align*}
\]

where \( t_4 \) is the first term in the sequence ST, such that \( A(t_4) \) does not appear on the tree above \( A(t_1), A(t_2), A(t_3), A(t_4), \exists x A(x) \), i.e. \( t_4 \neq t_3 \neq t_2 \neq t_1 \)

\[
\begin{align*}
\vert (\exists) \\
\vdots \\
\vert (\exists) \\
\vdots \\
\text{infinite branch}
\end{align*}
\]

Obviously, the above decomposition tree is \textit{infinite}, what proves that

\[ \not\models \exists x A(x) \]
Examples of Decomposition Trees

We construct now a proof in QRS of the quantifiers distributivity law

\[(\exists x (A(x) \cap B(x)) \Rightarrow (\exists x A(x) \cap \exists x B(x)))\]

and show that the proof in QRS of the inverse implication

\[((\exists x A(x) \cap \exists x B(x)) \Rightarrow \exists x (A(x) \cap B(x)))\]

does not exist, i.e. that

\[\neg ((\exists x A(x) \cap \exists x B(x)) \Rightarrow \exists x (A(x) \cap B(x)))\]

The decomposition tree \(T_A\) of the first formula is the following
Examples of Decomposition Trees

\( T_A \)

\[(\exists x (A(x) \cap B(x)) \Rightarrow (\exists x A(x) \cap \exists x B(x)))\]

\[| (\Rightarrow)\]

\[\neg \exists x (A(x) \cap B(x)), (\exists x A(x) \cap \exists x B(x))\]

\[| (\neg \exists)\]

\[\forall x \neg (A(x) \cap B(x)), (\exists x A(x) \cap \exists x B(x))\]

\[| (\forall)\]

\[\neg (A(x_1) \cap B(x_1)), (\exists x A(x) \cap \exists x B(x))\]

where \( x_1 \) is a first free variable in the sequence ST such that \( x_1 \) does not appear in

\[\forall x \neg (A(x) \cap B(x)), (\exists x A(x) \cap \exists x B(x))\]

\[| (\neg \cap)\]

\[\neg A(x_1), \neg B(x_1), (\exists x A(x) \cap \exists x B(x))\]

\[\land (\cap)\]
Examples of Decomposition Trees

\[ \wedge (\cap) \]

\[ \neg A(x_1), \neg B(x_1), \exists x A(x) \]

\[ \mid (\exists) \]

\[ \neg A(x_1), \neg B(x_1), A(t_1), \exists x A(x) \]

where \( t_1 \) is the first term in the sequence ST, such that \( A(t_1) \) does not appear on the tree above \( \neg A(x_1), \neg B(x_1), A(t_1), \exists x A(x) \)

\[ \mid (\exists) \]

\[ \neg A(x_1), \neg B(x_1), B(t_1), \exists x B(x) \]

\[ \mid (\exists) \]

\[ \neg A(x_1), \neg B(x_1), \ldots B(x_1), \exists x B(x) \]

axiom

\[ \neg A(x_1), \neg B(x_1), \ldots A(x_1), \exists x A(x) \]

axiom
Examples of Decomposition Trees

Observe, that it is possible to choose eventually a term $t_i = x_1$, as the formula $A(x_1)$ does not appear on the tree above

$$
\neg A(x_1), \neg B(x_1), ... A(x_1), \exists x A(x)
$$

By the definition of the sequence ST, the variable $x_1$ is placed somewhere in it, i.e. $x_1 = t_i$, for certain $i \geq 1$

It means that after $i$ applications of the step $(\exists)$ in the decomposition tree, we will get an axiom leaf

$$
\neg A(x_1), \neg B(x_1), ... A(x_1), \exists x A(x)
$$
Examples of Decomposition Trees

All leaves of the above tree $T_A$ are axioms, what means that we proved

$$\vdash_{QRS} (\exists x (A(x) \cap B(x)) \Rightarrow (\exists x A(x) \cap \exists x B(x))).$$

We construct now, as the last example, a decomposition tree $T_A$ of the formula

$$(((\exists x A(x) \cap \exists x B(x)) \Rightarrow \exists x (A(x) \cap B(x))))$$
Examples of Decomposition Trees

\[ T_A \]

\[((\exists x A(x) \cap \exists x B(x)) \Rightarrow \exists x (A(x) \cap B(x)))\]

\[ | (\Rightarrow) \]

\[ \neg (\exists x A(x) \cap \exists x B(x)) \Rightarrow \exists x (A(x) \cap B(x)) \]

\[ | (\neg \cap) \]

\[ \neg \exists x A(x), \neg \exists x B(x), \exists x (A(x) \cap B(x)) \]

\[ | (\neg \exists) \]

\[ \forall x \neg A(x), \neg \exists x B(x), \exists x (A(x) \cap B(x)) \]

\[ | (\forall) \]

\[ \neg A(x_1), \neg \exists x B(x), \exists x (A(x) \cap B(x)) \]

\[ | (\neg \exists) \]

\[ \neg A(x_1), \forall x \neg B(x), \exists x (A(x) \cap B(x)) \]

\[ | (\forall) \]
Examples of Decomposition Trees

\[ \neg A(x_1), \neg B(x_2), \exists x(A(x) \cap B(x)) \]

By the reasoning similar to the reasonings in the previous examples we get that \( x_1 \neq x_2 \)

\[ \neg A(x_1), \neg B(x_2), (A(t_1) \cap B(t_1)), \exists x(A(x) \cap B(x)) \]

where \( t_1 \) is the first term in the sequence \( ST \) such that \( (A(t_1) \cap B(t_1)) \) does not appear on the tree above \( \neg A(x_1), \neg B(x_2), (A(t_1) \cap B(t_1)), \exists x(A(x) \cap B(x)) \) Observe, that it is possible that \( t_1 = x_1 \), as \( (A(x_1) \cap B(x_1)) \) does not appear on the tree above. By the definition of the sequence, \( x_1 \) is placed somewhere in it, i.e. \( x_1 = t_i \), for certain \( i \geq 1 \).

For simplicity, we assume that \( t_1 = x_1 \) and get the sequence:

\[ \neg A(x_1), \neg B(x_2), (A(x_1) \cap B(x_1)), \exists x(A(x) \cap B(x)) \]

\( \bigwedge (\cap) \)
Examples of Decomposition Trees

\[ \bigwedge (\cap) \]

\[ \neg A(x_1), \neg B(x_2), \]
\[ A(x_1), \exists x (A(x) \cap B(x)) \]

Axiom

\[ \neg A(x_1), \neg B(x_2), \]
\[ B(x_1), \exists x (A(x) \cap B(x)) \]
\[ \mid (\exists) \]

\[ \neg A(x_1), \neg B(x_2), B(x_1), \]
\[ (A(x_2) \cap B(x_2)), \exists x (A(x) \cap B(x)) \]

see COMMENT

\[ \bigwedge (\cap) \]
Examples of Decomposition Trees

COMMENT: where \( x_2 = t_2 \) \( (x_1 \neq x_2) \) is the first term in the sequence \( ST \), such that
\[
(A(x_2) \cap B(x_2)) \text{ does not appear on the tree above }
\]
\[
\neg A(x_1), \neg B(x_2), (B(x_1), (A(x_2) \cap B(x_2))), \exists x (A(x) \cap B(x)). \text{ We assume that } t_2 = x_2 \text{ for the reason of simplicity.}
\]

\[
\bigwedge (\cap)
\]

\[
\neg A(x_1), \neg A(x_1),
\]
\[
\neg B(x_2), \neg B(x_2),
\]
\[
B(x_1), A(x_2), B(x_1), B(x_2),
\]
\[
\exists x (A(x) \cap B(x)) \exists x (A(x) \cap B(x))
\]
\[
| (\exists) \quad Axiom
\]
\[
| (\exists)
\]
\[
\ldots
\]
\[
| (\exists)
\]

\textit{infinite branch}
Examples of Decomposition Trees

The above decomposition tree $T_A$ contains an infinite branch which means that

$$\kappa_{QRS} \ ((\exists x A(x) \cap \exists x B(x)) \Rightarrow \exists x (A(x) \cap B(x)))$$
Chapter 10
Predicate Automated Proof Systems

Slides Set 1
PART 2: Proof of QRS Completeness
Our main goal now is to prove the Completeness Theorem for the predicate proof system **QRS**.

The proof of the Completeness Theorem presented here is due to Rasiowa and Sikorski (1961), as is the proof system **QRS**.

We adopted Rasiowa - Sikorski proof of **QRS** completeness to propositional case in chapter 6.
QRS Completeness

The completeness proofs, in the propositional case and in predicate case, are constructive as they are based on a direct construction of a counter model for any unprovable formula.

The construction of the counter model for the unprovable formula $A$ uses in both cases the decomposition tree $T_A$.

Rasiowa-Sikorski type of constructive proofs by defining counter models determined by the decomposition trees relay heavily of the notion of strong soundness.
QRS Semantics

Given a first order language $\mathcal{L}$

$$\mathcal{L} = \mathcal{L}_{\{\land,\lor,\Rightarrow,\neg\}}(P, F, C)$$

with the set $VAR$ of variables and the set $F$ of formulas

We define, after chapter 8 a notion of a model and a counter-model of a formula $A \in F$

We establish the semantics for QRS by extending it to the set

$$F^*$$

of all finite sequences of formulas of $\mathcal{L}$
QRS Semantics

Model
A structure $\mathcal{M} = [M, I]$ is called a model of $A \in \mathcal{F}$ if and only if

$$(\mathcal{M}, v) \models A$$

for all assignments $v : VAR \rightarrow M$

We denote it by

$\mathcal{M} \models A$

$M$ is called the universe of the model, $I$ the interpretation
QRS Semantics

Counter - Model
A structure $\mathcal{M} = [M, I]$ is called a counter-model of $A \in \mathcal{F}$ if and only if there is $v : VAR \rightarrow M$, such that

$$(\mathcal{M}, v) \not\models A$$

We denote it by

$\mathcal{M} \not\models A$
QRS Semantics

Tautology
A formula $A \in \mathcal{F}$ is called a **predicate tautology** and denoted by $\models A$ if and only if all structures $\mathcal{M} = [M, I]$ are models of $A$, i.e.

$$\models A \text{ if and only if } \mathcal{M} \models A$$

for all structures $\mathcal{M} = [M, I]$ for $\mathcal{L}$.
QRS Semantics

For any sequence \( \Gamma \in \mathcal{F}^* \), by \( \delta_\Gamma \) we understand any disjunction of all formulas of \( \Gamma \).

A structure \( \mathcal{M} = [M, I] \) is called a model of a sequence \( \Gamma \in \mathcal{F}^* \) and denoted by

\[
\mathcal{M} \models \Gamma
\]

if and only if \( \mathcal{M} \models \delta_\Gamma \).

The sequence \( \Gamma \in \mathcal{F}^* \) is a predicate tautology if and only if the formula \( \delta_\Gamma \) is a predicate tautology, i.e.

\[
\models \Gamma \text{ if and only if } \models \delta_\Gamma
\]
Strong Soundness

Our **goal** now is to prove the **Completeness Theorem** for **QRS**

The **correctness** of the **Rasiowa-Sikorski constructive proof** depends on the **strong soundness** of the rules of inference of **QRS**

We define it (in general case) as follows
Strong Soundness

Strongly Sound Rules
Given a predicate language proof system

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

An inference rule \( r \in \mathcal{R} \) of the form

\[
(r) \quad \frac{P_1 ; P_2 ; \ldots ; P_m}{C}
\]

is strongly sound if the following condition holds for any structure \( \mathcal{M} = [M, I] \) for \( \mathcal{L} \)

\[
\mathcal{M} \models \{P_1, P_2, \ldots, P_m\} \quad \text{if and only if} \quad \mathcal{M} \models C
\]
Strong Soundness

A predicate language proof system \( S = (\mathcal{L}, \mathcal{E}, LA, R) \) is **strongly sound** if and only if all logical axioms \( LA \) are tautologies and all its rules of inference \( r \in R \) are strongly sound.

**Strong Soundness Theorem**
The proof system \( QRS \) is strongly sound.

**Proof**
We have already proved in chapter 6 strong soundness of the **propositional** rules. The **quantifiers** rules are strongly sound by straightforward verification and is left as an exercise.
Soundnesss Theorem

The strong soundness property is \textit{stronger} then soundness property, hence also the following holds

\textbf{QRS Soundness Theorem}

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

\[
\text{if } \vdash_{QRS} \Gamma, \text{ then } \models \Gamma
\]

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

\[
\text{if } \vdash_{QRS} A, \text{ then } \models A
\]
Proof of Completeness Theorem

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

$$\vdash_{QRS} \Gamma \text{ if and only if } \models \Gamma$$

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

$$\vdash_{QRS} A \text{ if and only if } \models A$$

Proof  We prove the completeness part. We need to prove the formula $A$ case only because the case of a sequence $\Gamma$ can be reduced to the formula case of $\delta_\Gamma$. I.e. we prove the implication:

if $\models A$, then $\vdash_{QRS} A$
Proof of Completeness Theorem

We do it, as in the propositional case, by proving the opposite implication

\[ \text{if } \models_{QRS} A \text{ then } \not\models A \]

This means that we want to prove that for any formula \( A \), \textbf{unprovability} of \( A \) in \( QRS \) allows us to define its \textbf{counter-model}.

The \textbf{counter-model} is determined, as in the propositional case, by the decomposition tree \( T_A \).

We have proved the following

**Tree Theorem**

Each formula \( A \), generates its unique decomposition tree \( T_A \) and \( A \) has a proof only if this tree is \textbf{finite} and all its end sequences (leaves) are \textbf{axioms}.
Proof of Completeness Theorem

The **Tree Theorem** says that we have two cases to consider:

(C1) the tree $T_A$ is **finite** and contains a leaf which is not axiom, or

(C2) the tree $T_A$ is **infinite**

We will show how to construct a counter-model for $A$ in both cases:

a counter-model determined by a **non-axiom leaf** of the decomposition tree $T_A$,

or a counter-model determined by an **infinite branch** of $T_A$
Proof of Completeness Theorem

Proof in case (C1)
The tree $T_A$ is finite and contains a non-axiom leaf

Before describing a general method of constructing the counter-model determined by the decomposition tree $T_A$ we describe it, as an example, for a case of a general formula

$$(\exists x A(x) \Rightarrow \forall x A(x)),$$

and its particular case

$$(\exists x (P(x) \cap R(x, y)) \Rightarrow \forall x (P(x) \cap R(x, y)))$$

where $P, R$ are one and two argument predicate symbols, respectively
Proof of Completeness Theorem

First we build its decomposition tree:

\[ T_A \]

\[
(\exists x(P(x) \cap R(x, y)) \Rightarrow \forall x(P(x) \cap R(x, y)))
\]

| (⇒)

\[ \neg\exists x(P(x) \cap R(x, y)), \forall x(P(x) \cap R(x, y)) \]

| (¬∃)

\[ \forall x(\neg(P(x) \cap R(x, y)), \forall x(P(x) \cap R(x, y)) \]

| (∀)

\[ \neg(P(x_1) \cap R(x_1, y)), \forall x(P(x) \cap R(x, y)) \]

where \( x_1 \) is a first free variable in the sequence of term ST such that \( x_1 \) does not appear in \( \forall x(\neg(P(x) \cap R(x, y)), \forall x(P(x) \cap R(x, y)) \)

| (¬∩)

\[ \neg P(x_1), \neg R(x_1, y), \forall x(P(x) \cap R(x, y)) \]

| (∀)
Proof of Completeness Theorem

\[ | (\forall) \]

\[ \neg P(x_1), \neg R(x_1, y), (P(x_2) \cap R(x_2, y)) \]

where \( x_2 \) is a first free variable in the sequence of term ST such that \( x_2 \) does not appear in \( \neg P(x_1), \neg R(x_1, y), \forall x (P(x) \cap R(x, y)) \), the sequence ST is one-to-one, hence \( x_1 \neq x_2 \)

\[ \bigwedge (\cap) \]

\[ \neg P(x_1), \neg R(x_1, y), P(x_2) \]

\[ \neg P(x_1), \neg R(x_1, y), R(x_2, y) \]

\( x_1 \neq x_2 \), Non-axiom

\( x_1 \neq x_2 \), Non-axiom
Proof of Completeness Theorem

There are two non-axiom leaves
In order to define a counter-model determined by the tree $T_A$ we need to chose only one of them
Let’s choose the leaf

$$L_A = \neg P(x_1), \neg R(x_1, y), P(x_2)$$

We use the non-axiom leaf $L_A$ to define a structure $\mathcal{M} = [M, I]$ and an assignment $\nu$, such that

$$(\mathcal{M}, \nu) \not\models A$$

Such defined $\mathcal{M}$ is called a counter-model determined by the tree $T_A$
Proof of Completeness Theorem

We take a the **universe** of $\mathcal{M}$ the set $T$ of all terms of the language $L$, i.e. we put $M = T$.

We define the **interpretation** $I$ as follows.

For any **predicate** symbol $Q \in P$, $\#Q = n$ we put that $Q_I(t_1, \ldots t_n)$ is true (holds) for terms $t_1, \ldots t_n$ if and only if

the negation $\neg Q_I(t_1, \ldots t_n)$ of the formula $Q(t_1, \ldots t_n)$ appears on the leaf $L_A$

and $Q_I(t_1, \ldots t_n)$ is false (does not hold) for terms $t_1, \ldots t_n$, otherwise

For any **functional** symbol $f \in F$, $\#f = n$ we put

$$f_I(t_1, \ldots t_n) = f(t_1, \ldots t_n)$$
Proof of Completeness Theorem

It is easy to see that in particular case of our non-axiom leaf

\[ L_A = \neg P(x_1), \neg R(x_1, y), P(x_2) \]

\( P_i(x_1) \) is true (holds) for \( x_1 \), and not true for \( x_2 \)

\( R_i(x_1, y) \) is true (holds) for \( x_1 \) and for any \( y \in VAR \)
Proof of Completeness Theorem

We define the assignment $v : \text{VAR} \rightarrow T$ as identity, i.e., we put $v(x) = x$ for any $x \in \text{VAR}$.

Obviously, for such defined structure $[M, I]$ and the assignment $v$ we have that

$([T, I], v) \models P(x_1), \quad ([T, I], v) \models R(x_1, y),$ \quad $([T, I], v) \not\models P(x_2)$

We hence obtain that

$([T, I], v) \not\models \neg P(x_1), \neg R(x_1, y), P(x_2)$

This proves that such defined structure $[T, I]$ is a **counter model** for a non-axiom leaf $L_A$ and by the **Strong Soundness** we proved that

$\not\models (\exists x (P(x) \cap R(x, y)) \Rightarrow \forall x (P(x) \cap R(x, y)))$
C1: Proof of Completeness Theorem

C1: General Method

Let $A$ be any formula such that

$$\kappa_{QRS} A$$

Let $T_A$ be a decomposition tree of $A$

By the fact that $\kappa_{QRS}$ and C1, the tree $T_A$ is finite and has a non axiom leaf

$$L_A \subseteq LT^*$$

By definition, the leaf $L_A$ contains only atomic formulas and negations of atomic formulas
C1: Counter Model Definition

We use the non-axiom leaf $L_A$ to define a structure $\mathcal{M} = [M, I]$, an assignment $\nu : VAR \rightarrow M$, such that $(\mathcal{M}, \nu) \not\models A$

Such defined structure $\mathcal{M}$ is called a counter-model determined by the tree $T_A$
C1: Counter Model Definition

**Structure $\mathcal{M}$ Definition**

Given a formula $A$ and a non-axiom leaf $L_A$

We define a structure

\[ \mathcal{M} = [M, I] \]

and an assignment $v : VAR \rightarrow M$ as follows

1. We take a the universe of $\mathcal{M}$ the set $T$ of all terms of the language $L$, i.e. we put

\[ M = T \]
C1: Counter Model Definition

2. For any predicate symbol \( Q \in P \), \( \#Q = n \),

\[
Q_I \subseteq T^n
\]

is such that \( Q_I(t_1, \ldots t_n) \) holds (is true) for terms \( t_1, \ldots t_n \)

if and only if

the negation \( \neg Q(t_1, \ldots t_n) \) of the formula \( Q(t_1, \ldots t_n) \)
appears on the leaf \( L_A \) and

\( Q_I(t_1, \ldots t_n) \) does not hold (is false) for terms \( t_1, \ldots, t_n \)
otherwise
C1: Counter Model Definition

3. For any constant $c \in C$, we put $c_I = c$
   For any variable $x$, we put $x_I = x$
   For any functional symbol $f \in F$, $\#f = n$

   $$f_I : T^n \rightarrow T$$

   is identity function, i.e. we put

   $$f_I(t_1, \ldots t_n) = f(t_1, \ldots t_n)$$

   for all $t_1, \ldots t_n \in T$

4. We define the assignment $\nu : \text{VAR} \rightarrow T$ as identity, i.e. we put for all $x \in \text{VAR}$

   $$\nu(x) = x$$
C1: Counter Model Definition

Obviously, for such defined structure \([T, I]\) and the assignment \(v\) we have that

\[(T, I, v) \notmodels P \text{ if formula } P \text{ appears in } L_A,\]

\[(T, I, v) \models P \text{ if formula } \neg P \text{ appears in } L_A\]

This proves that the structure \(\mathcal{M} = [T, I]\) and assignment \(v\) are such that

\[(T, I, v) \notmodels L_A\]
C1: Counter Model Definition

By the **Strong Soundness Theorem** we have that

$$ \not\models A $$

This proves $ M \not\models A $ and we proved that

$$ \not\models A $$

This **ends** the proof of the case **C1**
**C2: Counter Model Definition**

**Proof of case C2:** $T_A$ is infinite

The case of the infinite tree is similar to the C1 case, even if a little bit more complicated.

Observe that the rule $$(\exists)$$ is the only rule of inference (decomposition) which can "produces" an infinite branch.

We first show how to construct the counter-model in the case of the simplest application of this rule, i.e. in the case of the atomic formula $$\exists x P(x)$$

for $P$ one argument relational symbol. All other cases are similar to this one.
C2: Particular Case n

The infinite branch $B_A$ in the following

$B_A$

$\exists x P(x)$

$\mid (\exists)$

$P(t_1), \exists x P(x)$

where $t_1$ is the first term in the sequence of terms, such that $P(t_1)$ does not appear on the tree above $P(t_1), \exists x P(x)$

$\mid (\exists)$

$P(t_1), P(t_2), \exists x P(x)$

where $t_2$ is the first term in the sequence of terms, such that $P(t_2)$ does not appear on the tree above $P(t_1), P(t_2), \exists x P(x)$, i.e. $t_2 \neq t_1$

$\mid (\exists)$
C2: Particular Case

\[ | (\exists) \]
\[ P(t_1), P(t_2), P(t_3), \exists x P(x) \]

where \( t_3 \) is the first term in the sequence of terms, such that \( P(t_3) \) does not appear on the tree above \( P(t_1), P(t_2), P(t_3), \exists x P(x) \), i.e. \( t_3 \neq t_2 \neq t_1 \)

\[ | (\exists) \]
\[ P(t_1), P(t_2), P(t_3), P(t_4), \exists x P(x) \]

\[ | (\exists) \]
\[ \ldots \]

\[ | (\exists) \]
\[ \ldots \]

The infinite branch \( B_A \), written from the top, in order of appearance of formulas is

\[ B_A = \{ \exists x P(x), P(t_1), A(t_2), P(t_2), P(t_4), \ldots \} \]

where \( t_1, t_2, \ldots \) is a one-to-one sequence of all terms
**C2:** Particular Case n

The **infinite** branch

\[ \mathcal{B}_A = \{ \exists x P(x), \ P(t_1), \ A(t_2), \ P(t_2), \ P(t_4), \ldots \} \]

contains with the formula \( \exists x P(x) \) all its instances \( P(t) \), for all terms \( t \in T \).

We define the structure \( \mathcal{M} = [M, I] \) and the assignment \( v \) as we did previously, i.e.

we take as the universe \( M \) the set \( T \) of all terms, and define \( P_I \) as follows:

\( P_I(t) \) holds if \( \neg P(t) \in \mathcal{B}_A \), and

\( P_I(t) \) does not hold if \( P(t) \in \mathcal{B}_A \).
**C2: Particular Case**

For any constant \( c \in \mathbb{C} \), we put \( c_I = c \), for any variable \( x \), we put \( x_I = x \).

For any functional symbol \( f \in F \), \( \#f = n \)

\[
f_I : T^n \rightarrow T
\]

is **identity** function, i.e. we put

\[
f_I(t_1, \ldots t_n) = f(t_1, \ldots t_n)
\]

for all \( t_1, \ldots t_n \in T \).
**C2: Particular Case**

We define the assignment \( v : \text{VAR} \rightarrow T \) as identity, i.e. we put for all \( x \in \text{VAR} \)

\[ v(x) = x \]

It is easy to see that for any formula \( P(t) \in \mathcal{B} \),

\[ ([T, I], v) \not\models P(t) \]

But the \( P(t) \in \mathcal{B} \) are all instances of the formula \( \exists x P(x) \), hence

\[ ([T, I], v) \not\models \exists x P(x) \]

and we proved

\[ \not\models \exists x P(x) \]
C2: General Method
C2: General Method

Let \( A \) be any formula such that

\[ \forall_{QRS} \ A \]

Let \( \mathcal{T}_A \) be an infinite decomposition tree of the formula \( A \)

Let \( \mathcal{B}_A \) be the infinite branch of \( \mathcal{T}_A \), written from the top, in order of appearance of sequences \( \Gamma \in \mathcal{F}^* \) on it, where \( \Gamma_0 = A \), i.e.

\[ \mathcal{B}_A = \{ \Gamma_0, \ \Gamma_1, \ \Gamma_2, \ ... \ \Gamma_i, \ \Gamma_{i+1}, \ ... \} \]
**C2: General Method**

Given the infinite branch

\[ \mathcal{B}_A = \{ \Gamma_0, \Gamma_1, \Gamma_2, \ldots \Gamma_i, \Gamma_{i+1}, \ldots \} \]

We define a set

\[ \mathcal{L}_F \subseteq \mathcal{F} \]

of all **indecomposable** formulas appearing in at least one sequence \( \Gamma_i, \ i \leq j \), i.e. we put

\[ \mathcal{L}_F = \{ B \in \mathcal{L}T : \text{there is } \Gamma_i \in \mathcal{B}_A, \text{ such that } B \text{ iappiears } \Gamma_i \} \]
C2: General Method

Note, that the following holds

(1) If \( i \leq i' \) and an **indecomposable** formula appears in \( \Gamma_i \), then it also appears in \( \Gamma_{i'} \).

(2) Since **none** of \( \Gamma_i \) is an axiom, for every atomic formula \( P \in \mathcal{A} \), at most one of the formulas \( P \) and \( \neg P \) is in \( \mathcal{L} \).
Counter Model Definition

Counter Model Definition

Let $T$ be the set of all terms. We define the structure $M = [T, I]$, the interpretation $I$ of constants and functional symbols, and the assignment $v$ in the set $T$, as in previous cases.

We define the interpretation $I$ of predicates $Q \in P$ as follows.

For any predicate symbol $Q \in P$, $\#Q = n$, we put

1. $Q_I(t_1, \ldots t_n)$ does not hold (is false) for terms $t_1, \ldots t_n$ if and only if
   $$Q_I(t_1, \ldots t_n) \in LF$$

2. $Q_I(t_1, \ldots t_n)$ does holds (is true) for terms $t_1, \ldots t_n$ if and only if
   $$[Q_I(t_1, \ldots t_n) \notin LF$$
Counter Model Definition

Directly from the definition we we have that \( M \not\models LF \)

Our goal now is to prove that

\[ M \not\models A \]

For this purpose we first introduce, for any formula \( A \in \mathcal{F} \), an inductive definition of the order \( \text{ord} A \) of the formula \( A \)

(1) If \( A \in \mathcal{AF} \), then \( \text{ord} A = 1 \)

(2) If \( \text{ord} A = n \), then \( \text{ord} \neg A = n + 1 \)

(3) If \( \text{ord} A \leq n \) and \( \text{ord} B \leq n \), then

\[ \text{ord}(A \cup B) = \text{ord}(A \cap B) = \text{ord}(A \Rightarrow B) = n + 1 \]

(4) If \( \text{ord} A(x) = n \), then \( \text{ord} \exists x A(x) = \text{ord} \forall x A(x) = n + 1 \)
Proof of Completeness Theorem

We conduct the proof of $\mathcal{M} \not\models A$ by contradiction.

Assume that $\mathcal{M} \models A$

Consider now a set $\mathcal{MF}$ of all formulas $B$ appearing in one of the sequences $\Gamma_i$ of the branch $\mathcal{B}_A$, such that $\mathcal{M} \models B$

We write the the set $\mathcal{MF}$ formally as follows

$$\mathcal{MF} = \{ B \in \mathcal{F} : \text{for some } \Gamma_i \in \mathcal{B}_A, \ B \text{ is in } \Gamma_i \text{ and } \mathcal{M} \models B \}$$
Proof of Completeness Theorem

Observe that the formula $A$ is in $\mathcal{M}_F$ so

$$\mathcal{M}_F \neq \emptyset$$

Let $B'$ be a formula in $\mathcal{M}_F$ such that

$$ordB' \leq ordB$$ for every $B \in \mathcal{M}_F$

There exists $\Gamma_i \in \mathcal{B}_A$ that is of the form $\Gamma', B', \Delta$ with an indecomposable $\Gamma'$

We have that $B'$ can not be of the form

$$(*) \quad \neg \exists x A(x) \text{ or } \neg \forall x A(x)$$

for if $B'$ of the $(*)$ form is in $\mathcal{M}_F$, then also formula $\forall x \neg A(x)$ or $\exists x \neg A(x)$ is in $\mathcal{M}_F$ and the orders of the two formulas are equal
Proof of Completeness Theorem

We carry the same order argument and show that $B'$ can not be of the form

$$\text{(**) } (A \cup B), \neg(A \cup B), (A \cap B), \neg(A \cap B), (A \Rightarrow B), \neg(A \Rightarrow B), \neg\neg A, \forall x A(x)$$

The formula $B'$ can not be of the form

$$\text{(***) } \exists x B(x)$$

since then there exists term $t$ and $j$ such that $i \leq j$, and $B'(t)$ appears in $\Gamma_j$ and the formula $B(t)$ is such that $\mathcal{M} \models B$.
Proof of Completeness Theorem

Thus \( B(t) \in \mathcal{MF} \) and \( \text{ord}B(t) < \text{ord}B' \)
This \textit{contradicts} the definition of \( B' \)
Since \( B' \) \textbf{is not} of the forms (\( \ast \)), (\( \ast\ast \)), (\( \ast\ast\ast \)), \( B' \) \textit{is indecomposable}. Thus \( B' \in \mathcal{LF} \) and consequently

\[ \mathcal{M} \not\models B' \]

On the other hand \( B' \) is in the set \( \mathcal{MF} \) and hence is one of the formulas satisfying

\[ \mathcal{M} \models B' \]

This \textit{contradiction} proves that \( \mathcal{M} \not\models A \) and hence we proved that

\[ \not\models A \]

This \textit{ends} the proof of the \textbf{Completeness Theorem} for \textbf{QRS}