CHAPTER 10

ch10

Predicate Automated Proof Systems Completeness of Classical Predicate Logic

We define and discuss here a 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 **??**. These both proof systems admit a *constructive proof* of completeness theorem. We adopt Rasiowa, Sikorski (1961) technique of construction a counter model determined by a decomposition tree to prove **QRS** completeness theorem 4. The proof, presented in section 3, is a generalization of the completeness proofs of **RS** and other Gentzen style propositional systems presented in details in chapter **??**. We refer the reader to this chapter as it provides a good introduction to the subject.

The other Gentzen type predicate proof system, including the original Gentzen proof systems **LK, LI** for classical and intuitionistic predicate logics are obtained from their propositional versions discussed in detail in chapter **??**. It can be done in a similar way as a generalization of the propositional **RS** the predicate **QRS** system presented here. We leave these generalizations as an exercises for the reader. That includes also the predicate language version of Gentzen proof of cut elimination theorem, Hauptzatz (1935). The Hauptzatz proof for the predicate classical **LK** and intuitionistic **LI** systems is easily obtained from the propositional proof included in chapter**??**.

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 and later 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 a *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 their method were eliminated by John Alan Robinson (1965) and developed into what we call now Robinson Resolution, or just a Resolution.
There are many excellent textbooks covering each of these methods. We recommend Melvin Fitting book *First-order logic and automated theorem proving*(2nd ed.). Springer-Verlag(1996) as the one that not only covers all of them but also discusses their relationships.

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. In section 4 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. The correctness of propositional case is established by theorem 5, of predicate case by theorem 6. In the first step of the predicate procedure we define a process of *elimination of quantifiers* from the original language. It is called Skolemization of the language and is presented in section 4.1. The correctness of the Skolemization is established by Skolem theorem 11. In the second step of the procedure we show how convert a quantifiers free formula into logically equivalent set of clauses. It is presented with a proof of correctness (theorem 13) in section 4.2.

1 QRS Proof System

We define components and semantics of the proof system **QRS** as follows.
Language $\mathcal{L}$

We adopt a predicate (first order) language

$$\mathcal{L} = \mathcal{L}_{\{\cap,\cup,\Rightarrow,\neg\}}(\mathbf{P},\mathbf{F},\mathbf{C}) \tag{1}$$

Q-lang

for **P, F, C** countably infinite sets of predicate, functional, and constant symbols respectively.
Let $\mathcal{F}$ denote a set of formulas of $\mathcal{L}$. The rules of inference of our system **QRS** operate on *finite sequences of formulas*, i.e. elements of $\mathcal{F}^*$ so we define the set of expressions of of **QRS** as follows. **Expressions $\mathcal{E}$**

We adopt as the set of expressions $\mathcal{E}$ of **RS** the set $\mathcal{F}^*$, i.e.

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

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

$$\Gamma, \Delta, \Sigma, \text{with indices if necessary.}$$

In order to define the axioms LA and the set of rules of inference of **QRS** we need to bring back some notions and to introduce some definitions.

An **atomic formula** of the predicate language $\mathcal{L}$ defined by (1) is any element of $\mathcal{A}^*$ (finite strings over the alphabet of $\mathcal{L}$) of the form

$$R(t_1, t_2, ..., t_n)$$

where $R \in \mathbf{P}, \#R = n$ and $t_1, t_2, ..., t_n \in \mathbf{T}$.

The set of all **atomic formulas** is denoted by $A\mathcal{F}$ and is defined as

$$A\mathcal{F} = \{R(t_1, t_2, ..., t_n) \in \mathcal{A}^* : \quad R \in \mathbf{P},\ t_1, t_2, ..., t_n \in \mathbf{T},\ n \geq 1\}. \tag{2}$$

atomic

We use symbols $R, Q, P, ...$ with indices if necessary to denote the **atomic formulas**.

Literals
We form a special subset $LT \subseteq \mathcal{F}$ of formulas, called a set of all *literals*, which is defined as follows.

$$LT = \{R : \ R \in A\mathcal{F}\} \cup \{\neg R : \ R \in A\mathcal{F}\}. \tag{3}$$

$\boxed{\texttt{p-literal}}$

The atomic formulas (2) are called **positive literals** and the elements of the second set of the above union (3), i.e. the negations of the atomic formulas are called **negative literals**.

Indecomposable, Decomposable Formulas
A formula $A \in \mathcal{F}$ is *indecomposable* if and only if it is atomic or a negation of an atomic formula, i.e. an *literal*. Otherwise A is *decomposable*.

Now we form *finite sequences* out of formulas (and, as a special case, out of literals). We need to distinguish the sequences formed out of literals from the sequences formed out of other formulas, so we adopt the following definition and notaions.

Indecomposable, Decomposable Sequences
A sequence Γ is *indecomposable* if and only if is formed out of indecomposable formulas only. Otherwise is *decomposable*.
We denote **indecomposable sequences** by by

$$\Gamma^{'}, \ \Delta^{'}, \ \Sigma^{'}, \ldots \quad \text{with indices if necessary.} \tag{4}$$

$\boxed{\texttt{p-indec}}$

By definition, $\Gamma^{'}, \ \Delta^{'}, \ \Sigma^{'} \ldots$ are finite sequences (empty included) formed out of **literals**, i.e $\Gamma^{'}, \ \Delta^{'}, \ \Sigma^{'}\Gamma^{'}, \ \Delta^{'}, \ \Sigma^{'} \in LT^{*}$.

We denote by

$$\Gamma, \ \Delta, \ \Sigma, \ldots \quad \text{with indices if necessary,} \tag{5}$$

$\boxed{\texttt{p-seq}}$

the elements of $\mathcal{F}^{*}$, i.e. we denote $\Gamma, \ \Delta, \ \Sigma$ finite sequences (empty included) formed out of elements of $\mathcal{F}$.

Logical Axioms LA

As the *logical axiom* of **QRS** we adopt any sequence of formulas which contains a and its negation, i.e any sequence of the form

$$\Gamma_1, \ A, \ \Gamma_2, \ \neg A' \ \Gamma_3 \quad \text{or} \quad \Gamma_1, \ \neg A, \ \Gamma_2, \ A, \ \Gamma_3 \tag{6}$$

$\boxed{\texttt{qaxiom}}$

for any literal $A \in LT$ and any sequences $\Gamma_1, \Gamma_2, \Gamma_3 \in \mathcal{F}^{*}$ of formulas.

$$\textbf{Rules of inference } \ \mathcal{R} \tag{7}$$

$\boxed{\texttt{qrules}}$

Group 1: Propositional Rules

Disjunction rules

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

Conjunction rules

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

Implication rules

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

Negation rule

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

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

Group 2: Quantifiers Rules

$(\exists)$

$$\frac{\Gamma^{'}, A(t), \Delta, \exists x A(x)}{\Gamma', \exists x A(x), \Delta}$$

where t is an arbitrary term.

$(\forall)$

$$\frac{\Gamma^{'}, A(y), \Delta}{\Gamma', \forall x A(x), \Delta}$$

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

$(\neg\forall)$

$$\frac{\Gamma^{'}, \exists x \neg A(x), \Delta}{\Gamma', \neg\forall x A(x), \Delta}$$

$(\neg\exists)$

$$\frac{\Gamma^{'}, \forall x \neg A(x), \Delta}{\Gamma', \neg\exists x A(x), \Delta}$$

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

Note that $A(t), A(y)$ denotes a formula obtained from $A(x)$ by writing t, y, respectively, in place of all occurrences of x in A. The variable y in $(\forall)$ is called the *eigenvariable.* The condition: *where y is a free individual variable which does not appear in any formula in the conclusion* is called the *eigenvariable condition.*

All occurrences of y in $A(y)$ of the rule $(\forall)$ are fully indicated.

The Proof System QRS

Formally we define the proof system **QRS** as follows.

$$\mathbf{QRS} = (\mathcal{L}_{\{\neg,\Rightarrow,\cup,\cap\}},\ \mathcal{E},\ \ LA,\ \ \mathcal{R}), \tag{8}$$

def:qrs

where $\mathcal{E} = \{\Gamma : \ \ \Gamma \in \mathcal{F}^*\}$, LA contains logical axioms of the system defined by (6), $\mathcal{R}$ is the set of rules of inference:

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

defined by (7).

By a **formal proof** of a sequence Γ in the proof system **QRS** we understand any sequence

$$\Gamma_1,\ \Gamma_2, ...\ \Gamma_n \tag{9}$$

fp

of sequences of formulas (elements of $\mathcal{F}^*$), 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 Γ_i is a conclusion of one of the inference rules of **QRS** with all its premisses placed in the sequence $\Gamma_1,\ \Gamma_2,\ ....\ \Gamma_{i-1}$.

We write, as usual,

$$\vdash_{QRS} \Gamma$$

to denote that Γ 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 $\mathcal{E}$) we will not use " ; " to separate the steps of the proof, and write the formal proof as $\Gamma_1;\ \Gamma_2;\ ....\ \Gamma_n$.

We write, however, the formal proofs in **QRS** as we did the propositional case (chapter **??**), in a form of trees rather then in a form of sequences, ie. in a form of a tree, where *leafs* of the tree are axioms, *nodes* are sequences such that each sequence on the tree follows from the ones immediately preceding it by one of the rules. The *root* is a theorem. We picture, and write our tree-proofs with the node on the top, and leafs on the very bottom, instead of more common way, where the leafs are on the top and root is on the bottom of the tree. We adopt hence the following definition.

p-tree

Definition 1 (Proof Tree)

By a proof tree, or **QRS**- *tree proof of* Γ *we understand a tree* $\mathbf{T}_\Gamma$ *of sequences satisfying the following conditions:*

1. The topmost sequence, i.e the root of $\mathbf{T}_\Gamma$ *is* Γ*,*

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 (7).

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

In particular cases, as in the propositional case, we will write our proof- trees indicating additionally the name of the inference rule used at each step of the proof. For example, if in a proof of a formula A from axioms (6) we use subsequently the rules

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

we represent the proof as the following tree denoted by $\mathbf{T}_A$.

$$\mathbf{T}_A$$

$$Formula\ A$$

$$\mid (\Rightarrow)$$

$$conclusion\ of\ (\forall)$$

$$\mid (\forall)$$

$$conclusion\ of\ (\neg\neg)$$

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

$$conclusion\ of\ (\cap)$$

$$\bigwedge(\cap)$$

$$\begin{array}{ccc} conclusion\ of\ (\forall) & \qquad & conclusion\ of\ (\cup) \\ \mid (\forall) & & \mid (\cup) \\ axiom & & conclusion\ of\ (\cap) \\ & & \bigwedge(\cap) \\ & & axiom \quad axiom \end{array}$$

Remark that the derivation trees don't represent a different *definition* of a formal proof. This remains the same in the Gentzen - style systems. Trees represent a certain *visualization* for those proofs and any formal proof in any system can be represented in a tree form.

2 QRS Decomposition Trees

The main advantage of the Gentzen proof systems lies not in a way we generate proofs in them, but in the way we can *search* for proofs in them. That such proof searches happens to be deterministic and automatic. We conduct such search by treating inference rules as *decomposition rules* (see chapter **??**) and building *decomposition trees.*A general principle of building decomposition trees is the following.

Decomposition Tree $\mathbf{T}_\Gamma$
For each $\Gamma \in \mathcal{F}^*$, a decomposition tree $\mathbf{T}_\Gamma$ is a tree build as follows.
Step 1. The sequence Γ is the **root** of $\mathbf{T}_\Gamma$ and for any node Δ of the tree we follow the steps bellow.
Step 2. If Δ is indecomposable or an axiom, then Δ becomes a **leaf** of the tree.
Step 3. If Δ is decomposable, then we traverse Δ from left to right to identify the first **decomposable formula** B and identify inference rule treated as decomposition rule 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 infinite branch.

In particular case when when Γ has only one element, namely a a formula $A \in \mathcal{F}$, we define we call it a decomposition tree of A and denote by $\mathbf{T}_A$.

Here is a detailed definition of the decomposition tree for **QRS**.

QRS Decomposition Tree Definition (10) d:dtree

Given a formula $A \in \mathcal{F}$, we define its decomposition tree $\mathbf{T}_A$ as follows.

Observe that the inference rules of **QRS** are divided in two groups: propositional connectives rules and quantifiers rules. The propositional connectives rules are: $(\cup), (\neg\cup), (\cap), (\neg\cap), (\Rightarrow), (\neg \Rightarrow)$, and $(\neg\neg)$. The quantifiers rules are: $(\forall), (\exists), (\neg\forall)$ and $(\neg\exists)$.

We define the decomposition tree in the case of the propositional rules and the rules $(\neg\forall)$, $(\neg\exists)$ in the same way as for the propositional language (chapter **??**).

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 $\Gamma^{'}, \forall x A(x), \Delta$ or $\Gamma^{'}, \exists x A(x), \Delta$, i.e. to deal with the rules $(\forall)$ and $(\exists)$ in a way which would preserve the property of the *uniqueness* of the decomposition tree, we assume that all terms form a one-to one sequence

$$t_1, t_2, ..., t_n, ... \qquad (11)$$ terms

Observe, that by the definition, all free variables are terms, hence all free variables appear in the sequence 11. Let Γ be a sequence on the tree in which the first indecomposable formula has the quantifier $\forall$ as a main connective, i.e. Γ is of the form $\Gamma^{'}, \forall x A(x), \Delta$. We write a sequence $\Gamma^{'}, A(x), \Delta$ below it on the tree, as its child, where the variable x has to fulfill the following condition.

call **Condition 1** $(\forall)$

x is the first free variable in the sequence 11 such that x does not appear in any formula in $\Gamma^{'}, \forall x A(x), \Delta$.

Observe, that the condition 1 corresponds to the restriction put on the application of the rule $(\forall)$.

If the main connective of Γ, i.e. the main connective of the first formula in Γ which is not an literal, is $(\exists)$. In this case Γ 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 has to fulfill the following condition.

cexists **Condition 2** $(\exists)$

t is the first term in the sequence 11 such that the formula A(t) does not appear in any sequence which is placed above $\Gamma^{'}, A(t), \Delta, \exists x A(x)$ on the tree.

The fact that the sequence 11 is one- to - one and the fact that, by the conditions 1 and 2, we always chose the first appropriate term (variable) from this sequence, guarantee that the decomposition process is also unique in the case of the quantifiers rules (∀) and (∃).

From all above, and we conclude the following.

thm:tree **Theorem 1**

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

(i) *the decomposition tree* $\mathbf{T}_A$ *is unique.*

(ii) *Moreover, the following conditions hold.*

1. If $\mathbf{T}_A$ *is* **finite** *and all its leaves are axioms, then*

$$\vdash_{QRS} \; A$$

and $\mathbf{T}_A$ *is a tree-proof of* A *in* **QRS**.

2. If $\mathbf{T}_A$ *is* **finite** *and contains a non-axiom leaf, or* $\mathbf{T}_A$ *is* **infinite**, *then*

$$\nvdash_{QRS} A.$$

2.1 Examples of Decomposition Trees

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

Example 1

The decomposition tree $\mathcal{T}_A$ *of the de Morgan Law*

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

is the following.

$$\mathbf{T}_A$$

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

$$\mid (\Rightarrow)$$

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

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

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

$$\mid (\forall)$$

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

where x_1 is a first free variable in the sequence 11 such that x_1 does not appear in $\forall x A(x), \exists x \neg A(x)$

$$\mid (\exists)$$

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

where x_1 is the first term (variables are terms) in the sequence 11 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

The above tree $\mathbf{T}_A$ ended with an axiom, so it represents a proof of

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

in **QRS**, i.e. we proved that

$$\vdash_{QRS} (\neg \forall x A(x) \Rightarrow \exists x \neg A(x)).$$

Example 2

The decomposition tree $\mathbf{T}_A$ *of*

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

is the following.

$$\mathbf{T}_A$$

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

$$\mid (\Rightarrow)$$

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

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

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

$$\exists x \neg A(x), \exists x A(x)$$

$$\mid (\exists)$$

$$\neg A(t_1), \exists x A(x), \exists x \neg A(x)$$

where t_1 is the first term in the sequence 11, such that $\neg A(t_1)$ does not appear on the tree above $\neg A(t_1), \exists x A(x), \exists x \neg A(x)$

$$\mid (\exists)$$

$$\neg A(t_1), A(t_1), \exists x \neg A(x), \exists x A(x)$$

where t_1 is the first term in the sequence 11, 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

The above tree also ended with the axiom, hence we proved that

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

Example 3

The decomposition tree $\mathbf{T}_A$ *of*

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

is the following.

$$\mathbf{T}_A$$

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

$$\mid (\Rightarrow)$$

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

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

$$\forall x \neg A(x), \forall x A(x)$$

$$\mid (\forall)$$

$$\neg A(x_1), \forall x A(x)$$

where x_1 is a first free variable in 11 such that x_1 does not appear in $\forall x \neg A(x), \forall x A(x)$

$$\mid (\forall)$$

$$\neg A(x_1), A(x_2)$$

where x_2 is a first free variable in 11 such that x_2 does not appear in $\neg A(x_1), \forall x A(x)$, the sequence 11 is one-to- one, hence $x_1 \neq x_2$

Non - axiom

The decomposition tree, for any formula A is *unique*, so we conclude from the fact that the above tree has a non-axiom branch that

$$\nvdash_{QRS} (\exists x A(x) \Rightarrow \forall x A(x)).$$

Remark when constructing the following tree $\mathbf{T}_A$ for the formula $\exists x A(x)$ in example 4 below we adopt on the right branch of the a tree in the the shorthand notation instead of the repeating a similar reasoning performed on the left branch.

e-tree

Example 4

The decomposition tree $\mathbf{T}_A$ *of the formula* $\exists x A(x)$ *is the following.*

$$\mathbf{T}_A$$

$$\exists x A(x)$$

$$\mid (\exists)$$

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

where t_1 is the first term in the sequence 11, 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 11, 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 11, 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. $t_3 \neq t_2 \neq t_1$

$$\mid (\exists)$$

$$A(t_1), A(t_2), A(t_3), A(t_4), \exists x A(x)$$

$$\mid (\exists)$$

.....

$$\mid (\exists)$$

.....

infinite branch

Obviously, the above decomposition tree is infinite, what proves that

$$\nvdash_{QRS} \exists x A(x).$$

We will find now the proof of the distributivity law

$$(\exists x(A(x) \cap B(x)) \Rightarrow (\exists x A(x) \cap \exists x B(x)))$$

and show that we can't prove in **QRS** the inverse implication

$$((\exists x A(x) \cap \exists x B(x)) \Rightarrow \exists x(A(x) \cap B(x))).$$

Remark when constructing the following trees $\mathbf{T}_A$ in examples 5, 6 adopt, as we did in the previous example 4, the shorthand notation when the reasoning is similar to the one presented in the example 4.

e-tree1 **Example 5**

The decomposition tree $_A$ of the first formula

$$(\exists x(A(x)\cap B(x))\Rightarrow(\exists xA(x)\cap\exists xB(x)))$$

is the following.

$$\mathbf{T}_A$$

$$(\exists x(A(x)\cap B(x))\Rightarrow(\exists xA(x)\cap\exists xB(x)))$$

$$\mid(\Rightarrow)$$

$$\neg\exists x(A(x)\cap B(x)),(\exists xA(x)\cap\exists xB(x))$$

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

$$\forall x\neg(A(x)\cap B(x)),(\exists xA(x)\cap\exists xB(x))$$

$$\mid(\forall)$$

$$\neg(A(x_1)\cap B(x_1)),(\exists xA(x)\cap\exists xB(x))$$

where x_1 is a first free variable in the sequence 11 such that x_1 does not appear in $\forall x\neg(A(x)\cap B(x)),(\exists xA(x)\cap\exists xB(x))$

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

$$\neg A(x_1),\neg B(x_1),(\exists xA(x)\cap\exists xB(x))$$

$$\bigwedge(\cap)$$

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

$$\mid(\exists)$$

$$\neg A(x_1),\neg B(x_1),A(t_1),\exists xA(x)$$

where t_1 is the first term in the sequence 11, such that $A(t_1)$ does not appear on the tree above $\neg A(x_1),\neg B(x_1),A(t_1),\exists xA(x)$ Observe, that it is possible that $t_1=x_1$, as $A(x_1)$ does not appear on the tree above. By the definition of the sequence 11, 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 a step:

$$\mid(\exists)$$

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

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

$$\mid(\exists)$$

$$\neg A(x_1),\neg B(x_1),B(t_1),\exists xB(x)$$

$$\mid(\exists)$$

$$...$$

$$\mid(\exists)$$

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

All leaves of the above tree $\mathbf{T}_A$ 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 $\mathbf{T}_A$ of the formula $((\exists x A(x) \cap \exists x B(x)) \Rightarrow \exists x(A(x) \cap B(x)))$.

e-tree2

Example 6

The decomposition tree of the formula

$$((\exists x A(x) \cap \exists x B(x)) \Rightarrow \exists x(A(x) \cap B(x)))$$

is the following.

$$\mathbf{T}_A$$

$$((\exists x A(x) \cap \exists x B(x)) \Rightarrow \exists x(A(x) \cap B(x)))$$

$$\mid (\Rightarrow)$$

$$\neg(\exists x A(x) \cap \exists x B(x)) \exists x(A(x) \cap B(x))$$

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

$$\neg\exists x A(x), \neg\exists x B(x), \exists x(A(x) \cap B(x))$$

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

$$\forall x \neg A(x), \neg\exists x B(x), \exists x(A(x) \cap B(x))$$

$$\mid (\forall)$$

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

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

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

$$\mid (\forall)$$

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

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

$$\mid (\exists)$$

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

where t_1 is the first term in the sequence 11, 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 11, 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:*

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

$$\bigwedge(\cap)$$

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

Axiom

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

$$\mid (\exists)$$

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

where $x_2 = t_2$ ($x_1 \neq x_2$) is the first term in the sequence 11, such that $(A(x_2) \cap B(x_2))$ does not appear on the tree above $\neg A(x_1), \neg B(x_2), (B(x_1), (A(x_2) \cap B(x_2)), \exists x(A(x) \cap B(x))$. We assume that $t_2 = x_2$ for the reason of simplicity.

$$\bigwedge(\cap)$$

$$\neg A(x_1),$$
$$\neg B(x_2),$$
$$B(x_1), A(x_2),$$
$$\exists x(A(x) \cap B(x))$$
$$\mid (\exists)$$
$$...$$
$$\bigwedge(\cap)$$
$$...$$
$$\mid (\exists)$$
$$...$$
$$\mid (\exists)$$

infinite branch

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

Axiom

The above decomposition tree $\mathbf{T}_A$ contains an infinite branch what means that

$$\not\vdash_{QRS} \ ((\exists x A(x) \cap \exists x B(x)) \Rightarrow \exists x(A(x) \cap B(x))).$$

3 Proof of QRS Completeness

sec:cproof

Our main goal is to prove the Completeness Theorem for **QRS**. The proof of completeness theorem presented here is due to Rasiowa and Sikorski (1961), as is the proof system **QRS**. We adopted their proof to propositional case in chapter **??**.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 the decomposition tree $\mathbf{T}_A$. We call such constructed counter model a counter model determined by the tree $\mathbf{T}_A$. Rasiowa-Sikorski type of constructive proofs of counter models determined by the tree decomposition trees relay heavily of the notion of a *strong soundness*. We define it here (definition 8), adopting chapter **??** general definition to our case.

Given a first order language $\mathcal{L}$ (1) with the set VAR of variables and the set $\mathcal{F}$ of formulas. We define, after chapter **??** a notion of a model and a counter-model of a formula A of $\mathcal{L}$ and then extend it to the the set $\mathcal{F}^*$ establishing the **semantics** for **QRS**.

Definition 2 (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 $v : VAR \longrightarrow M$.
We denote it by

$$\mathcal{M} \models A.$$

M is called the universe of the model, I the interpretation.

Definition 3 (Counter - Model)

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

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

We denote it by

$$\mathcal{M} \not\models A.$$

The definition of the first order logic tautology is the following.

Definition 4 (Tautology)

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

$$\models A \quad \textit{if and only if} \quad \mathcal{M} \models A$$

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

Directly from the above definition we get the following, simple fact.

Fact 1 (Counter Model)

For any $A \in \mathcal{F}$, A is not a tautology ($\not\models A$) if and only if there is a counter - model $\mathcal{M} = [M, I]$ of A, i.e. we can define M, I, and v such that $([M, I], v) \not\models A$.

d:G **Definition 5**

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

$$\delta_\Gamma$$

we understand any disjunction of all formulas of Γ.

d:qsem **Definition 6 (QRS Semantics)**

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

$$\mathcal{M} \models \ \Gamma$$

if and only if

$$\mathcal{M} \models \delta_\Gamma.$$

The sequence Γ is a predicate tautology if and only if the formula δ_Γ is a predicate tautology, i.e.

$$\models \Gamma \quad \textit{if and only if} \quad \models \delta_\Gamma.$$

Our goal now is to prove the completeness theorem for **QRS**.The correctness of the proof we present depends on the strong soundness of the rules of inference of rules of inference defined as follows.

qr-ss **Definition 7 (Strongly Sound Rules)**

Given a predicate language (1) proof system $S = (\mathcal{L}, \mathcal{E}, LA, \mathcal{R})$ An inference rule $r \in \mathcal{R}$ of the form

$$(r) \quad \frac{P_1 \ \ ; \ \ P_2 \ \ ; \ \ .... \ \ ; \ \ P_m}{C}$$

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

$$\mathcal{M} \ \models \{P_1, P_2, .P_m\} \quad \textit{if and only if} \quad \mathcal{M} \ \models \ C. \tag{12}$$ q-rss

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

$$P_1 \cap P_2 \cap \ \dots \ \cap P_m \equiv C. \qquad (13)$$ sr-e

q-strong **Definition 8 (Strongly Sound System)**

A predicate language (1) proof system $S = (\mathcal{L}, \mathcal{E}, LA, \mathcal{R})$ *is* **strongly sound** *if and only if all logical axioms* LA *are tautologies and all its rules of inference* $r \in \mathcal{R}$ *are* **strongly sound**.

thm:qss **Theorem 2 (Strong Soundness)**

The proof system **QRS** *(8) is* **strongly sound**.

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

The strong soundness property is stronger then soundness property, hence also the following holds.

thm:qsound **Theorem 3 (Soundness Theorem)**

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

$$\textit{if} \quad \vdash_{QRS} \Gamma, \quad \textit{then} \quad \models \Gamma.$$

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

$$\textit{if} \quad \vdash_{QRS} A, \quad \textit{then} \quad \models A.$$

Q-compl **Theorem 4 (Completeness Theorem)**

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

$$\vdash_{QRS} \Gamma \quad \textit{if and only if} \quad \models \Gamma.$$

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

$$\vdash_{QRS} A \quad \textit{if and only if} \quad \models A.$$

Proof
We have to prove the inverse implication to the soundness theorem 3. We need to prove the formula A case only because the case of a sequence Γ can be reduced to the formula case. Namely, the disjunction of all formulas in Γ. I.e. we prove the implication:

$$\text{if} \quad \models A, \quad \text{then} \quad \vdash_{QRS} A.$$

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

$$\text{if} \quad \nvdash_{QRS} \; A \quad \text{then} \quad \not\models \; A.$$

This means that we want prove that for any formula A, unprovability of A in **QRS** ($\nvdash_{QRS} \; A$), allows us to define its counter- model. The counter- model is determined, as in the propositional case, by the decomposition tree $\mathbf{T}_A$. By theorem 1 each formula A, generates its unique decomposition tree $\mathcal{T}_A$ and A has a proof only if this tree is finite and all its end sequences (leaves) are axioms. Moreover, it says that we have two cases to consider:

(C1) the tree $\mathbf{T}_A$ is **finite** and contains a leaf which is not axiom, or

(C2) the tree $\mathbf{T}_A$ 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 $\mathbf{T}_A$, or a counter- model determined by an infinite branch of $\mathbf{T}_A$.

Proof in case **(C1)**: $\mathbf{T}_A$ is **finite** and contains a non- axiom leaf.

Before describing a general method of constructing the counter-model determined by the decomposition tree $\mathcal{T}_A$ we describe it, as an example, for a case of a formula

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

and its **particular case**

$$(\exists x(P(x) \cap R(x,y)) \Rightarrow \forall x(P(x) \cap R(x,y))) \tag{14}$$

$\boxed{\texttt{ex}}$

for P, R one and two argument predicate symbols, respectively.

We construct the counter model for the formula (14) as follows.

First we build its decomposition tree:

$$\mathbf{T}_A$$

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

$$\mid (\Rightarrow)$$

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

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

$$\forall x \neg(P(x) \cap R(x,y)), \forall x(P(x) \cap R(x,y))$$

$$\mid (\forall)$$

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

where x_1 is a first free variable in 11 such that x_1 does not appear in
$\forall x \neg(P(x) \cap R(x,y)), \forall x(P(x) \cap R(x,y))$

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

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

$$\mid (\forall)$$

$$\neg P(x_1), \neg R(x_1,y), (P(x_2) \cap R(x_2,y))$$

where x_2 is a first free variable in the sequence 11 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 11 is one-to- one, hence $x_1 \neq x_2$

$$\bigwedge(\cap)$$

$$\neg P(x_1), \neg R(x_1,y), P(x_2) \qquad\qquad \neg P(x_1), \neg R(x_1,y), R(x_2,y)$$

$x_1 \neq x_2$, Non-axiom $\qquad\qquad$ $x_1 \neq x_2$, Non-axiom

There are two non-axiom leaves. In order to define a counter-model for (14) determined by the tree $\mathbf{T}_A$ we need to chose only one of them. Let's choose the leaf

$$L_A = \neg P(x_1), \neg R(x_1,y), P(x_2). \tag{15}$$

leaf

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

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

Such defined $\mathcal{M}$ is called a **counter - model** determined by the tree $\mathbf{T}_A$.

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

$$M = \mathbf{T}.$$

We define the interpretation I as follows. For any predicate symbol $Q \in \mathbf{P}, \#Q = n$ 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 $f \in \mathbf{F}, \#f = n$ we put $f_I(t_1, \ldots t_n) = f(t_1, \ldots t_n)$.

It is easy to see that in particular case of our non-axiom leaf (15)

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

$P_I(x_1)$ is true (holds) for x_1, and not true for x_2. $R_I(x_1,y)$ is true (holds) for x_1 any for any $y \in VAR$.

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

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

$$([\mathbf{T}, I],\ v) \models P(x_1), \quad ([\mathbf{T}, I],\ v) \models R(x_1, y), \quad \text{and} \quad ([\mathbf{T}, I],\ v) \not\models P(x_2).$$

We hence obtain that

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

This proves that such defined structure $[\mathbf{T}, I]$ is a counter model for a non-axiom leaf (15). By the strong soundness of QRS) (theorem 2) the structure $\mathcal{M} = [\mathbf{T}, I]$ is also a counter- model for the formula (14), i.e. we proved that

$$\not\models (\exists x(P(x) \cap R(x, y)) \Rightarrow \forall x(P(x) \cap R(x, y))).$$

C1: General Method
Let A be any formula such that $\not\vdash_{QRS}\ A$.

Let $\mathbf{T}_A$ be a decomposition tree of A. By the fact that $\not\vdash_{QRS}$ and **C1**, the tree $\mathbf{T}_A$ is finite and has a *non axiom leaf*

$$L_A \subseteq LT^*. \qquad (16) \quad \boxed{\texttt{T-leaf}}$$

By definition, the leaf L_A contains only atomic formulas and negations of atomic formulas.

We use the **non-axiom leaf** L_A (16) to define a structure $\mathcal{M} = [M, I]$ an assignment $v : VAR \longrightarrow M$, such that $(\mathcal{M}, v) \not\models A$. Such defined structure $\mathcal{M}$ is called a **counter - model** determined by the tree $\mathbf{T}_A$.

Structure $\mathcal{M}$ Definition (17) $\boxed{\texttt{def:cmod}}$

Given L_A. We define a structure

$$\mathcal{M} = [M, I] \qquad (18) \quad \boxed{\texttt{c-mod}}$$

and an assignment $v : VAR \longrightarrow M$ as follows.

1. We take a the universe of $\mathcal{M}$ the set $\mathbf{T}$ of all terms of our language $\mathcal{L}$' i.e. we put

$$M = \mathbf{T}.$$

2. For any predicate symbol $Q \in \mathbf{P}, \#Q = n$,

$$Q_I \subseteq \mathbf{T}^n$$

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.

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

$$f_I : \mathbf{T}^n \longrightarrow \mathbf{T}$$

is identity function, i.e. we put

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

for all $t_1, \dots t_n \in \mathbf{T}$.

4. We define the assignment $v : VAR \longrightarrow \mathbf{T}$ as *identity*, i.e. we put for all $x \in VAR$ $v(x) = x$. Obviously, for such defined structure $[\mathbf{T}, I]$ and the assignment v we have that

$$([\mathbf{T}, I],\ v) \not\models\ P \ \text{ if formula } P \text{ appears in } \quad L_A,$$

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

This proves that the structure $\mathcal{M} = [\mathbf{T}, I]$ and assignment v defined by (18) are such that

$$([\mathbf{T}, I], v) \not\models\ L_A.$$

By the **strong soundness** (theorem 2) of **QRS**

$$(([\mathbf{T}, I], v) \not\models\ A.$$

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

$$\not\models\ A.$$

This ends the proof of the case **C1**.

Proof in case **(C2)**: $\mathbf{T}_A$ is **infinite**.

The case of the **infinite tree** is similar, even if a little bit more complicated. Observe first 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 formula

$$\exists x P(x)$$

where P is an one argument relational symbol. All other cases are similar to this one. The infinite branch $\mathcal{B}_A$ in this case consists of all elements of the decomposition tree:

$$\mathbf{T}_A$$

$$\exists x P(x)$$

$$\mid (\exists)$$

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

where t_1 is the first term in the sequence 11, 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 11, such that $P(t_2)$ does not appear on the tree above $P(t_1), P(t_2), \exists x P(x)$, i.e. $t_2 \neq t_1$

$$\mid (\exists)$$

$$P(t_1), P(t_2), P(t_3), \exists x P(x)$$

where t_3 is the first term in the sequence 11, such that $P(t_3)$ does not appear on the tree above $P(t_1), P(_2), P(t_3), \exists x P(x)$, i.e. $t_3 \neq t_2 \neq t_1$

$$\mid (\exists)$$

$$P(t_1), P(t_2), P(t_3), P(t_4), \exists x P(x)$$

$$\mid (\exists)$$

$$.....$$

$$\mid (\exists)$$

$$.....$$

The infinite branch of $\mathbf{T}_A$, written from the top, in oder of appearance of formulas is

$$\mathcal{B}_A = \{\exists x P(x),\ P(t_1),\ A(t_2),\ P(t_2),\ P(t_4), ...\}$$

where $t_1, t_2, ...$ is a one - to one sequence (11) of all elements of the set $\mathbf{T}$ of all terms.

This means that the infinite branch $\mathcal{B}$ contains with the formula $\exists x P(x)$ all its instances $P(t)$, for all terms $t \in \mathbf{T}$.

We define the structure $\mathcal{M} = [M, I]$ and valuation v following the definition 17. We take as the universe M the set $\mathbf{T}$ of all terms, and now in our case we 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 $P(t) \in \mathcal{B}_A$.

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 $\exists x P(x)$, hence

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

C2: General Method
Let A be any formula such that $\nvdash_{QRS} A$.

Let $\mathcal{T}_A$ be an **infinite** decomposition tree of a formula A. Let $\mathcal{B}_A$ the infinite branch of $\mathbf{T}_A$, written from the top, in order of appearance of sequences $\Gamma \in \mathcal{F}^*$ on it, where $\Gamma_0 = A$.

$$\mathcal{B}_A = \{\Gamma_0, \ \ \Gamma_1, \ \ \Gamma_2, \ \ \dots \ \ \Gamma_i, \ \ \Gamma_{i+1}, \ \ \dots\} \tag{19}$$ `branch`

We define a set $L\mathcal{F} \subseteq \mathcal{F}$ of all *indecomposable* formulas appearing in at least one $\Gamma_i, \ i \leq j$, i.e.

$$L\mathcal{F} = \{B \in LT : \ \text{there is } \ \Gamma_i \in \mathcal{B}_A, \ \text{such that } B \text{ is in } \Gamma_i\}. \tag{20}$$ `b-indec`

Note, that the following holds.
(1) If $i \leq i'$ and an indecomposable formula appears in Γ_i, then it also appears in $\Gamma_{i'}$.
(2) Since none of Γ_i, is an axiom (6), for every atomic formula (2) $P \in A\mathcal{F}$, at most one of the formulas P and $\neg P$ is in $L\mathcal{F}$ (20).

Counter Model Definition
Let $\mathbf{T}$ be the set of all terms. We define the structure $\mathcal{M} = [\mathbf{T}, I]$ and valuation v in the set $\mathbf{T}$ as in the definition 17, with the interpretation of predicates $Q \in \mathbf{P}$ defined as follows.

For any predicate symbol $Q \in \mathbf{P}, \#Q = n$, $Q_I \subseteq \mathbf{T}^n$ is such that:

(1) $Q_I(t_1, \dots t_n)$ does not hold (is false) for terms $t_1, \dots t_n$ if and only if

$$Q_I(t_1, \dots t_n) \in L\mathcal{F}$$

and
(2) $Q_I(t_1, \dots t_n)$ does holds (is true) for terms $t_1, \dots t_n$ if and only if

$$Q_I(t_1, \dots t_n) \notin L\mathcal{F}.$$

This proves that the structure $\mathcal{M} = [\mathbf{T}, I]$ is such that

$$\mathcal{M} \not\models \ L\mathcal{F}. \tag{21}$$ `c-ind`

To prove that $\not\models \ A$ it suffices that

$$\mathcal{M} \not\models \ A. \tag{22}$$ `c-A`

For this purpose we first introduce, for any formula $A \in \mathcal{F}$, an inductive definition of the order *ord* A of the formula A.
(1) If $A \in A\mathcal{F}$, then *ord* $A = 1$.
(2) If *ord* $A = n$, then *ord* $\neg A = n + 1$. (3) If *ord* $A \leq n$ and *ord* $B \leq n$, then *ord* $(A \cup B) =$ *ord* $(A \cap B) =$ *ord* $(A \Rightarrow B) = n + 1$.
(4) If *ord* $A(x) = n$, then *ord* $\exists x A(x) =$ *ord* $\forall x A(x) = n + 1$.

We conduct the proof of (23) by contradiction. Suppose that (23) does not hold, i.e. assume that

$$\mathcal{M} \models A. \tag{23}$$ m-A

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

$$\mathcal{M} \models B. \tag{24}$$ m-B

We write the the set $M\mathcal{F}$ formally as follows.

$$M\mathcal{F} = \{B \in \mathcal{F} : \text{ for some } \Gamma_i \in \mathcal{B}_A,\ B \text{ is in } \Gamma_i \text{ and } \mathcal{M} \models B\}. \tag{25}$$ M-set

Observe that by assumption (23) and the definition (25), the formula A is in $M\mathcal{F}$ and hence $M\mathcal{F} \neq \emptyset$.

Let B' be a formula in $M\mathcal{F}$ such that *ord* $B' \leq$ *ord* B for every $B \in M\mathcal{F}$. There exists $\Gamma_i \in\in \mathcal{B}_A$ that is of the form Γ', B', Δ with an indecomposable Γ'.
We have that B' can not be of the form

$$\neg\exists x A(x) \quad \text{or} \quad \neg\forall x A(x) \tag{26}$$ n-Q

for if (26) is in $M\mathcal{F}$, then also formula $\forall x \neg A(x)$ or $\exists x \neg A(x)$ is in $M\mathcal{F}$ and the orders of the two formulas are equal.

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

$$(A \cup B), \neg(A \cup B), (A \cap B), \neg(A \cap B), (A \Rightarrow B), \neg(A \Rightarrow B), \neg\neg A, \forall x A(x). \tag{27}$$ n-r

The formula B' can't be of the form

$$\exists x B(x) \tag{28}$$ n-ex

since then there exists term t and j such that $i \leq j$, $B'(t)$ appears in Γ_j and the formula $B(t)$ satisfies (24). Thus $B(t) \in M\mathcal{F}$ and $ordB(t) < ordB'$. This contradicts the definition of B'.

Since B' is not of the form (26), (27), (28), B' is indecomposable. Thus $B' \in L\mathcal{F}$ (20), and consequently by (21),

$$\mathcal{M} \not\models B'.$$

On the other hand B' by definition is in the set $M\mathcal{F}$ and hence is one o the formulas satisfying (24), i.e.

$$\mathcal{M} \not\models B'.$$

This contradiction proves that (23) $\mathcal{M} \not\models A$ and hence we proved

$$\not\models A.$$

This **ends** the proof of the Completeness Theorem 4 for **QRS**.

sec:skolem

4 Skolemization and Clauses

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

The goal of this part is to define an effective *process of transformation* of any formula A of a predicate language $\mathcal{L} = \mathcal{L}_{\{\neg,\cup,\cap,\Rightarrow\}}(\mathbf{P}, \mathbf{F}, \mathbf{C})$ into a certain corresponding set of clauses $\mathbf{C}_A$. This is done in two stages.
S1. We convert any formula A of $\mathcal{L}$ into an *open* formula A^* of a language $\mathcal{L}^*$ by a process of *elimination of quantifiers* from the original $\mathcal{L}$. The method is due to T. Skolem (1920) and is called Skolemization. The resulting formula A^* is equisatisfiable with A: it is satisfiable if and only if the original one is satisfiable (Skolem Theorem 11).

The stage S1. is performed as the first step in a Resolution based automated theorem prover and is described in section 4.1.

S2. We define a proof system $\mathbf{QRS}^*$ based on the language $\mathcal{L}^*$ and use it transform any formula A^* of $\mathcal{L}^*$ into an logically equivalent set of clauses $\mathbf{C}_{A^*}$ (theorem 13).

The final result of stages S1. and S1 is the set $\mathbf{C}_A$ of clauses corresponding to the formula A, called a *clausal form* of A (theorem 6.

The *transformation process* for any propositional formula A into its *logically equivalent* set $\mathbf{C}_A$ of clauses follows directly from the use of the propositional system **RS** (theorem 5).

.

def:clause

Definition 9 (Clauses)

Given a language $\mathcal{L}$, propositional or predicate.

1. A **literal** *as an atomic, or a negation of an atomic formula of $\mathcal{L}$. We denote by LT the set of all literals of $\mathcal{L}$.*

2. A **clause** *$\mathcal{C}$ is a* **finite set** *of literals.Empty clause is denoted by $\{\}$.*

3. We denote by $\mathbf{C}$ any finite set of all clauses.

$$\mathbf{C} = \{\mathcal{C}_1,\ \mathcal{C}_2,\ \ldots\ \mathcal{C}_n\},$$

for any $n \geq 0$.

G-clause

Definition 10

Given a propositional or predicate language L*, and a sequence* $\Gamma \in LT^*$*. A* **clause determined** *by* Γ *is a set form out of all elements of the sequence* Γ *We we denote it by* $\mathcal{C}_\Gamma$.

Example 7

In particular,
1. if $\Gamma_1 = a, a, \neg b, c, \neg b, c$ *and* $\Gamma_2 = \neg b, c, a$*, then* $\mathcal{C}_{\Gamma_1} = \mathcal{C}_{\Gamma_2} = \{a, c, \neg b\}$.

2. If $\Gamma_1 = \neg P(x_1), \neg R(x_1, y), P(x_2), \neg P(x_1), \neg R(x_1, y), P(x_2)$ *and*
$\Gamma_2 = \neg P(x_1), \neg R(x_1, y), P(x_2)$*, then* $\mathcal{C}_{\Gamma_1} = \mathcal{C}_{\Gamma_2} = \{\neg P(x_1), \neg R(x_1, y), P(x_2)\}$.

The semantics for clauses is basically the same as for the sequences. We define it as follows.

d:C

Definition 11 *Given a propositional or predicate language* L*. For any clause* $\mathcal{C}$*, write* $\delta_{\mathcal{C}}$ *for a* disjunction *of all literals in* $\mathcal{C}$.

d:qsem

Definition 12 (Clauses Semantics)

Let $\mathcal{M} = [M, I]$ *be a structure for a predicate language* $\mathcal{L}$*, or a truth assignment* v *in case of* $\mathcal{L}$ *propositional.*

$\mathcal{M}$ *is called a* **model** *for a clause* $\mathcal{C}$ *(predicate or propositional), (*$\mathcal{M} \models \mathcal{C}$*) if and only if*

$$\mathcal{M} \models \delta_{\mathcal{C}}.$$

$\mathcal{M}$ *is called a* **model** *for a set* $\mathbf{C}$ *of clauses (*$\mathcal{M} \models \mathbf{C}$*) if and only if*

$$\mathcal{M} \models \delta_{\mathcal{C}} \qquad \textit{for all clauses } \mathcal{C} \in \mathbf{C}.$$

A-eq

Definition 13 (Equivalence)

A formula A *of a language* $\mathcal{L}$ *is* **equivalent** *with a set set* $\mathbf{C}$ *of clauses* $(A \equiv \mathbf{C})$ *if and only if* $A \equiv \sigma_{\mathbf{C}}$*, where* $\sigma_{\mathbf{C}}$ *is a* conjunction *of all formulas* $\delta_{\mathcal{C}}$ *for all clauses* $\mathcal{C} \in \mathbf{C}$.

thm:ceq1

Theorem 5 (Formula-Clauses Equivalency)

For any formula A *of a propositional language* $\mathcal{L}$*, there is an effective procedure of generating a corresponding set* $\mathbf{C}_A$ *of clauses such that*

$$A \equiv \mathbf{C}_A \tag{29}$$

Proof

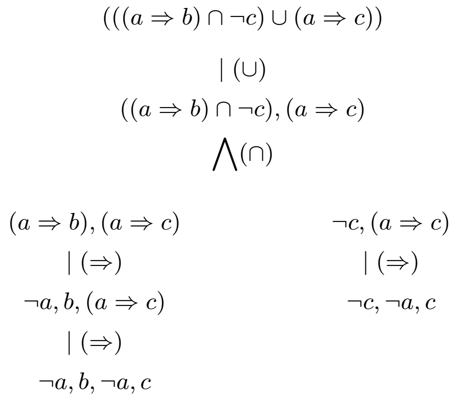
Let $\mathcal{L} = \mathcal{L}_{\{\neg,\cup,\cap,\Rightarrow\}}$. Given $A \in \mathcal{F}$, we use the **RS** system (chapter **??**) to build the decomposition tree $\mathbf{T}_A$. We form clauses out of the leaves of the tree $\mathbf{T}_A$, i.e. for every leaf L we create a clause $\mathcal{C}_L$ determined by L (definition 10). We put

$$\mathbf{C}_A = \{\mathcal{C}_L : \quad L \text{ is a leaf of } \quad \mathbf{T}_A\}.$$

Directly from the strong soundness (13) of rules of inference of **RS** and the definition 13 we get $A \equiv \mathbf{C}_A$. This ends the **proof** for the propositional case.

Consider a decomposition tree of a formula $(((a \Rightarrow b) \cap \neg c) \cup (a \Rightarrow c))$

$\mathbf{T}_A$


$(((a \Rightarrow b) \cap \neg c) \cup (a \Rightarrow c))$
$\mid (\cup)$
$((a \Rightarrow b) \cap \neg c), (a \Rightarrow c)$
$\bigwedge(\cap)$
$(a \Rightarrow b), (a \Rightarrow c)$
$\mid (\Rightarrow)$
$\neg a, b, (a \Rightarrow c)$
$\mid (\Rightarrow)$
$\neg a, b, \neg a, c$
$\neg c, (a \Rightarrow c)$
$\mid (\Rightarrow)$
$\neg c, \neg a, c$

Example 8

For the formula $(((a \Rightarrow b) \cap \neg c) \cup (a \Rightarrow c))$ and the tree $\mathbf{T}_A$, the leaves are
$L_1 = \neg a, b, \neg a, c$ and $\mathcal{C}_{L_1} = \{\neg a, b, c\}$ and
$L_2 = \neg c, \neg a, c$ and $\mathcal{C}_{L_2} = \{\neg c, \neg a, c\}$. The set of clauses is

$$\mathbf{C}_A = \{\{\neg a, b, c\}, \ \{\neg c, \neg a, c\}\}.$$

By theorem 5, $A \equiv \mathbf{C}_A$. Semantically it means, by definition 13,

$$A \equiv (((\neg a \cup b) \cup c) \cap ((\neg c \cup \neg a) \cup c)).$$

thm:ceq2 **Theorem 6 (Clausal Form)**

For any formula A of a predicate language $\mathcal{L}$, there is an effective procedure of generating an open formula A^* of a quantifiers free *language $\mathcal{L}^*$ and a set $\mathbf{C}_{A^*}$ of clauses such that*

$$A^* \ \equiv \ \mathbf{C}_{A^*}. \qquad (30)$$ C-prop

The set $\mathbf{C}_{A^*}$ of clauses of $\mathcal{L}^*$ with the property (30) is called a **clausal form** *of the formula A of $\mathcal{L}$.*

Proof
Given a formula A of a language $\mathcal{L}$. The open formula A^* of the quantifiers free language $\mathcal{L}^*$ is obtained by the Skolemization process. The effectiveness and correctness of the process follows from **PNF** theorem 10 and **Skolem** theorem 11 described in section 4.1.

As the next step, we define (section 4.2) a proof system $\mathbf{QRS}^*$(43) based on the quantifiers free language $\mathcal{L}^*$. The system $\mathbf{QRS}^*$ is a version of the system **QRS** (8) restricted to its Propositional Rules. At this point we carry the proof in a similar way to the proof in the propositional case (theorem 5). Namely, for any formula A^* of $\mathcal{L}^*$ obtained from A of $\mathcal{L}$ we construct its the decomposition tree $\mathbf{T}_{A^*}$. We form clauses out of the leaves of the tree $\mathbf{T}_{A^*}$, i.e. for every leaf L we create a clause $\mathcal{C}_L$ determined by L and we put

$$\mathbf{C}_{A^*} = \{\mathcal{C}_L : \quad L \text{ is a leaf of } \quad \mathbf{T}_{A^*}\}.$$

This is the **clausal form** of the formula A of $\mathcal{L}$ by theorem 13 proved in section 4.2. To complete the proof we need now to develop results of the section 4.1 and the section 4.2.

4.1 Prenex Normal Forms and Skolemization

sec:prenex

We remind the following important notion.

Term t is free for x in $A(x)$. Let $A(x) \in \mathcal{F}$ and t be a term, $A(t)$ be a result of substituting t for all free occurrences of x in $A(x)$.

We say that t **is free for** x **in** $A(x)$, if no occurrence of a variable in t becomes a bound occurrence in $A(t)$.

In particular, if $A(x), A(x_1, x_2, ..., x_n) \in \mathcal{F}$ and $t, t_1, t_2, ..., t_n \in \mathbf{T}$, then

$$A(x/t), \quad A(x_1/t_1, x_2/t_2, ..., x_n/t_n)$$

or, more simply just

$$A(t), \quad A(t_1, t_2, ..., t_n)$$

denotes the result of replacing all occurrences of the free variables $x, x_1, x_2, ..., x_n$, by the terms $t, t_1, t_2, ..., t_n$, respectively, assuming that $t, t_1, t_2, ..., t_n$ are free for x, $x_1, x_2, ..., x_n$, respectively, in A.

The assumption that t **is free for** x **in** $A(x)$ while substituting t for x, is important because otherwise we would distort the meaning of $A(t)$. This is illustrated by the following example.

Example 9

Let $t = y$ and $A(x)$ be

$$\exists y(x \neq y).$$

Obviously t is not free for y in A. The substitution of t for x produces a formula $A(t)$ of the form

$$\exists y(y \neq y),$$

which has a different meaning than $\exists y(x \neq y)$.

Here are more examples illustrating the notion: *t is free for x in $A(x)$.*

Example 10

Let $A(x)$ be a formula

$$(\forall y P(x, y) \cap Q(x, z))$$

and t be a term $f(x, z)$, i.e. $t = f(x, z)$.

None of the occurrences of the variables x, z of t is bound in $A(t)$, hence we say that $t = f(x, z)$ is **free for** *x in $(\forall y P(x, y) \cap Q(x, z))$.*

Substituting t on a place of x in $A(x)$ we obtain a formula $A(t)$ of the form

$$(\forall y P(f(x, z), y) \cap Q(f(x, z), z)).$$

Example 11

Let $A(x)$ be a formula

$$(\forall y P(x, y) \cap Q(x, z))$$

The term $t = f(y, z)$ is **not free** *for x in $A(x)$ because substituting $t = f(y, z)$ on a place of x in $A(x)$ we obtain now a formula $A(t)$ of the form*

$$(\forall y P(f(y, z), y) \cap Q(f(y, z), z))$$

which contain a bound occurrence of the variable y of t $(\forall y P(f(y, z), y))$.

The other occurrence $(Q(f(y, z), z))$ of y is free, but it is not sufficient, as for term to be free for x, all occurrences of its variables has to be free in $A(t)$.

Another important notion we will use here is the following notion of *similarity of formulas.*

Intuitively, we say that $A(x)$ and $A(y)$ are similar if and only if $A(x)$ and $A(y)$ are the same except that $A(x)$ has free occurrences of x in exactly those places where $A(y)$ has free occurrences of y.

Example 12

The formulas $\exists z(P(x,z) \Rightarrow Q(x))$ *and* $\exists z(P(y,z) \Rightarrow Q(y))$ *are similar.*

The formal definition of this notion follows.

Definition 14 (Similarity)

Let x *and* y *be two different variables. We say that the formulas* $A(x)$ *and* $A(x/y)$ *are* **similar** *and denote it by*

$$A(x) \sim A(x/y)$$

if and only if y *is free for* x *in* $A(x)$ *and* $A(x)$ **has no** *free occurrences of* y.

Example 13

The formulas $A(x)$*:* $\exists z(P(x,z) \Rightarrow Q(x,y))$ *and* $A(x/y)$*:* $\exists z(P(y,z) \Rightarrow Q(y,y))$ *are* **not similar***;* y *is free for* x *in* $A(x)$*, but the formula* $A(x/y)$ **has** *a free occurrence of* y.

Example 14

The formulas $A(x)$*:* $\exists z(P(x,z) \Rightarrow Q(x,y))$ *and* $A(x/w)$*:* $\exists z(P(w,z) \Rightarrow Q(w,y)$ **are similar***;* w *is free for* x *in* $A(x)$ *and the formula* $A(x/w)$ *has no free occurrence of* w.

Directly from the definition we get the following.

`lemm` **Lemma 1**

For any formula $A(x) \in \mathcal{F}$*, if* $A(x)$ *and* $A(x/y)$ *are similar* $A(x) \sim A(y)$*, then*

$$\forall x A(x) \equiv \forall y A(y),$$

$$\exists x A(x) \equiv \exists y A(y).$$

We prove, by the induction on the number of connectives and quantifiers in a formula A the following.

`th:rep` **Theorem 7 (Replacement Theorem)**

For any formulas $A, B \in \mathcal{F}$*, if* B *is a sub-formula of* A*, if* A^* *is the result of replacing zero or more occurrences of* B *in* A *by a formula* C*, and* $B \equiv C$*, then* $A \equiv A^*$.

Directly from lemma 1 and replacement theorem 7 we get that the following theorem holds.

bvar **Theorem 8 (Change of Bound Variables)**

For any formula $A(x), A(y), B \in \mathcal{F}$, if $A(x)$ and $A(x/y)$ are similar, i.e. $A(x) \sim A(y)$, and the formula $\forall x A(x)$ or $\exists x A(x)$ is a sub-formula of B, and B^* is the result of replacing zero or more occurrences of $A(x)$ in B by a formula $\forall y A(y)$ or $\exists y A(y)$, then $B \equiv B^*$.

defapart **Definition 15 (Naming Variables Apart)**

We say that a formula B has its variables **named apart** *if no two quantifiers in B bind the same variable and no bound variable is also free.*

We can now use theorem 8 to prove its more general version.

apart **Theorem 9 (Naming Variables Apart)** *Every formula $A \in \mathcal{F}$ is logically equivalent to one in which all variables are named apart.*

We use the above theorems plus the equational laws for quantifiers (31) to prove, as a next step a so called a Prenex Form Theorem 10.
In order to do so we first we define an important notion of **prenex normal form** of a formula.

closure **Definition 16 (Closure of a Formula)**

By a **closure** *of a formula A we mean a* **closed** *formula A' obtained from A prefixing in universal quantifiers all those variables that a free in A; i.e. if $A(x_1, \ldots, x_n)$ then $A' \equiv A$ is*

$$\forall x_1 \forall x_2 .... \forall x_n A(x_1, x_2, ..., x_n)$$

Example 15

Let A be a formula $(P(x,y) \Rightarrow \neg\exists z\ R(x,y,z))$, its **closure** *$A' \equiv A$ is $\forall x \forall y (P(x,y) \Rightarrow \neg\exists z\ R(x,y,z))$.*

defPNF **Definition 17 (Prenex Normal Form)**

Any formula A of the form

$$Q_1 x_1 Q_2 x_2 .... Q_n x_n B$$

where each Q_i is a universal or existential quantifier, i.e. for all $1 \leq i \leq n$, $Q_i \in \{\exists, \forall\}$, $x_i \neq x_j$ for $i \neq j$, and B **contains no quantifiers***, is said to be in prenex normal form* **(PNF)***.*
We include the case $n = 0$ when there are no quantifiers at all.

We assume that the formula A in **PNF** is always **closed**. If it is not closed we form its **closure** (definition 16) instead. We prove that, for every formula A, we can effectively construct a formula B that is in the prenex normal form **PNF** and $A \equiv B$.

`thmPNF` **Theorem 10 (PNF Theorem)**

There is an effective procedure for transforming any formula $A \in \mathcal{F}$ into a logically equivalent formula A' in the prenex normal form **PNF**.

Proof
We use theorems 7, 8, 9, theorem 15, and the following logical equivalences (31) proved in chapter **??**.

Equational Laws of Quantifiers (31) `eq-law`

$$\forall x(A(x) \cup B) \equiv (\forall x A(x) \cup B) \qquad (32) \qquad \boxed{\texttt{1}}$$

$$\forall x(A(x) \cap B) \equiv (\forall x A(x) \cap B) \qquad (33) \qquad \boxed{\texttt{2}}$$

$$\exists x(A(x) \cup B) \equiv (\exists x A(x) \cup B) \qquad (34) \qquad \boxed{\texttt{3}}$$

$$\exists x(A(x) \cap B) \equiv (\exists x A(x) \cap B) \qquad (35) \qquad \boxed{\texttt{4}}$$

$$\forall x(A(x) \Rightarrow B) \equiv (\exists x A(x) \Rightarrow B) \qquad (36) \qquad \boxed{\texttt{5}}$$

$$\exists x(A(x) \Rightarrow B) \equiv (\forall x A(x) \Rightarrow B) \qquad (37) \qquad \boxed{\texttt{6}}$$

$$\forall x(B \Rightarrow A(x)) \equiv (B \Rightarrow \forall x A(x)) \qquad (38) \qquad \boxed{\texttt{7}}$$

$$\exists x(B \Rightarrow A(x)) \equiv (B \Rightarrow \exists x A(x)) \qquad (39) \qquad \boxed{\texttt{8}}$$

where B is a formula such that B *does not contain any free occurrence* of x.

The formal procedure is defined by induction on the number k of occurrences of connectives and quantifiers in A. We show now how it works in some particular cases.

Exercise 1

Find a prenex normal form **PNF** *of a formula A:* $(\forall x(P(x) \Rightarrow \exists x Q(x))$.

Solution

We find **PNF** in the following steps.

Step 1: Rename Variables Apart

By the theorem 8 we can make all **bound variables** in A different, i.e. we transform A into an equivalent formula A'

$$\forall x(P(x) \Rightarrow \exists y Q(y)).$$

Step 2: Pull out Quantifiers

We apply the equational law

$$(C \Rightarrow \exists y Q(y)) \equiv \exists y \ (C \Rightarrow Q(y))$$

to the sub-formula $B : (P(x) \Rightarrow \exists y Q(y))$ of A' for $C = P(x)$, as P(x) does not contain the variable y. We get its equivalent formula $B^* : \exists y(P(x) \Rightarrow Q(y))$. We substitute now B^* on place of B in A' and get a formula

$$A'' : \ \ \forall x \exists y(P(x) \Rightarrow Q(y))$$

such that $A'' \equiv A' \equiv A$.

A'' is a required prenex normal form **PNF** for A.

e:pnf **Exercise 2** *Find a prenex normal form* **PNF** *formula A' for the formula A:*

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

Solution
Step 1: Rename Variables Apart
Take a sub- formula $B(x, y) : \ \forall y \exists x \ R(x, y)$ of A, get $B(x/z, y/w) : \forall z \exists w \ R(z, w)$ and replace B(x,y) by $B(x/z, y/w)$ in A and get

$$(\exists x \forall y \ R(x, y) \Rightarrow \forall z \exists w \ R(z, w))$$

Step 2: Pull out quantifiers
We use corresponding equational laws of quantifiers (36), 37) to pull out first quantifiers $\exists x \forall y$ and get the following

$$A' : \ \ \forall x \exists y((R(x, y) \Rightarrow \ \forall z \exists w \ R(z, w))),$$

such that $A' \equiv A$. Now we pull quantifiers $\forall z \exists w$ in $(R(x,y) \Rightarrow\ \forall z \exists w\ R(z,w))$ and get the prenex normal form **PNF** formula

$$A'':\quad \forall x \exists y \forall z \exists w\ ((R(x,y) \Rightarrow\ R(z,w))),$$

such that $A'' \equiv A' \equiv A$.

Observe we can also perform a different **Step 2** by pulling first the quantifiers $\forall z \exists w$ and then quantifiers $\forall x \exists y$ and obtain **another PNF** A''' for A:

$$A''':\quad \forall z \exists w \forall x \exists y\ (R(x,y) \Rightarrow\ R(z,w)).$$

We will show now how any formula A in its prenex normal form **PNF** we can transformed it into a corresponding **open formula** A^*.

The open formula A^* belongs to a richer language then the initial language to which the formula A belongs. The transformation process **adds** new constants, called **Skolem constants**, and new function symbols, called **Skolem function symbols** to the initial language.

The whole process is called the skolemisation of the initial language $\mathcal{L}$, the such build extension of the initial language is called a **Skolem extension** of $\mathcal{L}$,.

Skolem Procedure of Elimination of Quantifiers (40) q-proc

Given a formula A of the language $\mathcal{L} = \mathcal{L}_{\{\neg,\cup,\cap,\Rightarrow\}}(\mathbf{P},\mathbf{F},\mathbf{C})$ in its prenex normal form **PNF**, i.e.

$$A = Q_1 x_1 Q_2 x_2 \dots Q_n x_n B(x_1, x_2, \dots x_n) \qquad (41)$$ pnf

where each Q_i is a universal or existential quantifier, i.e. for all $1 \leq i \leq n$, $Q_i \in \{\exists, \forall\}$, $x_i \neq x_j$ for $i \neq j$, and $B(x_1, x_2, \dots x_n)$ contains no quantifiers.

We describe now a **procedure** of elimination of all quantifiers from the formula A (41) and hence transforming it into a corresponding **open formula** A^*.

We assume that the formula $A = Q_1 x_1 Q_2 x_2 \dots Q_n x_n B(x_1, x_2, \dots x_n)$ is **closed**. If it is not closed we form its **closure** instead.

We considerer 3 cases.
Case 1
All quantifiers Q_i for $1 \leq i \leq n$ are **universal**, i.e. the **closed formula** A is

$$\forall x_1 \forall x_2 \dots \forall x_n B(x_1, x_2, \dots, x_n)$$

We replace the formula A by the **open formula** A^*:

$$B(x_1, x_2, \dots., x_n).$$

Case 2
All quantifiers Q_i for $1 \leq i \leq n$ are **existential**, i.e. the **closed formula** A is

$$\exists x_1 \exists x_2 .... \exists x_n B(x_1, x_2, \dots x_n)$$

We replace the formula A by the **open formula** A^*:

$$B(c_1, c_2, \dots., c_n)$$

where $c_1, c_2, \dots., c_n$ and **new** individual constants, all different, **added** to our original language $\mathcal{L}$. We call such constants added to the language **Skolem constants**

Case 3
The quantifiers are **mixed** . We assume that A is **closed**. If it is not closed we form its **closure** instead. We eliminate quantifiers one by one and step by step depending on first, and consecutive quantifiers.

Given a **closed PNF** formula A

$$Q_1 x_1 Q_2 x_2 \dots Q_n x_n B(x_1, x_2, \dots x_n)$$

Step 1 Elimination of $Q_1 x_1$
We have two possibilities for the first quantifier $Q_1 x_1$, namely **P1** $Q_1 x_1$ is **universal** or **P2** $Q_1 x_1$ is **existential**.

Consider **P1**
First quantifier in A is universal, i. e. A is

$$\forall x_1 Q_2 x_2 \dots Q_n x_n B(x_1, x_2, \dots x_n)$$

We **replace** A by a formula A_1 :

$$Q_2 x_2 \dots Q_n x_n B(x_1, x_2, \dots x_n)$$

We have **eliminated** the quantifier Q_1 in this case.

Consider **P2**
First quantifier in A is **existential**, i. e. A is

$$\exists x_1 Q_2 x_2 \dots Q_n x_n B(x_1, x_2, \dots x_n)$$

We **replace** A by a formula A_1 :

$$Q_2 x_2 \dots Q_n x_n B(b_1, x_2, \dots x_n)$$

where b_1 is a **new** constant symbol **added** to our original language $\mathcal{L}$. We call such constant symbol added to the language **Skolem constant** symbol.

We have **eliminated** the quantifier Q_1 in this case. We have covered all cases and this **ends** the **Step 1**.

Step 2 Elimination of $Q_2 x_2$.

Consider now the **PNF** formula A_1 from **Step1- P1**

$$Q_2 x_2 \ldots Q_n x_n B(x_1, x_2, \ldots x_n)$$

Remark that the formula A_1 might not be closed.

We have again two possibilities for elimination of the quantifier $Q_2 x_2$, namely **P1** $Q_2 x_2$ is **universal** or **P2** $Q_2 x_2$ is **existential**.

Consider **P1**
First quantifier in A_1 is **universal**, i.e. A_1 is

$$\forall x_2 Q_3 x_3 \ldots Q_n x_n B(x_1, x_2, x_3, \ldots x_n)$$

We replace A_1 by the following A_2

$$Q_3 x_3 \ldots Q_n x_n B(x_1, x_2, x_3, \ldots x_n)$$

We have **eliminated** the quantifier Q_2 in this case.

Consider **P2**
First quantifier in A_1 is **existential**, i.e. A_1 is

$$\exists x_2 Q_3 x_3 \ldots Q_n x_n B(x_1, x_2, x_3, \ldots x_n)$$

Observe that now the variable x_1 is a **free** variable in $B(x_1, x_2, x_3, \ldots x_n)$ and hence in A_1.
We replace A_1 by the following A_2

$$Q_3 x_3 \ldots Q_n x_n B(x_1, f(x_1), x_3, \ldots x_n)$$

where f is a **new** one argument functional symbol **added** to our original language $\mathcal{L}$. We call such functional symbols added to the original language **Skolem** functional symbols.
We have **eliminated** the quantifier Q_2 in this case.

Consider now the **PNF** formula A_1 from **Step1 - P2**

$$Q_2 x_2 Q_3 x_3 \ldots Q_n x_n B(b_1, x_2, \ldots x_n)$$

Again we have two cases.

Consider **P1**
First quantifier in A_1 is **universal**, i.e. A_1 is

$$\forall x_2 Q_3 x_3 \dots Q_n x_n B(b_1, x_2, x_3, \dots x_n)$$

We replace A_1 by the following A_2

$$Q_3 x_3 \dots Q_n x_n B(b_1, x_2, x_3, \dots x_n)$$

We have **eliminated the quantifier** Q_2 in this case.
Consider **P2**
First quantifier in A_1 is **existential**, i.e. A_1 is

$$\exists x_2 Q_3 x_3 \dots Q_n x_n B(b_1, x_2, x_3, \dots x_n)$$

We replace A_1 by A_2

$$Q_3 x_3 \dots Q_n x_n B(b_1, b_2, x_3, \dots x_n)$$

where $b_2 \neq b_1$ is a new Skolem constant symbol **added** to our original language $\mathcal{L}$.
We have **eliminated** the quantifier Q_2 in this case. We have covered all cases and this **ends** the **Step 2**. **Step 3** Elimination of $Q_3 x_3$

Let's now consider, as an **example** formula A_2 from **Step 2; P1** i.e. the formula

$$Q_3 x_3 \dots Q_n x_n B(x_1, x_2, x_3, \dots x_n)$$

We have again 2 choices to consider, but will describe only the following.

P2 First quantifier in A_2 is **existential**, i. e. A_2 is

$$\exists x_2 Q_4 x_4 \dots Q_n x_n B(x_1, x_2, x_3, x_4, \dots x_n)$$

Observe that now the variables x_1, x_2 are **free** variables in $B(x_1, x_2, x_3, \dots x_n)$ and hence in A_2.

We replace A_2 by the following A_3

$$Q_4 x_3 \dots Q_n x_n B(x_1, x_2, g(x_1, x_2), x_4 \dots x_n)$$

where g is a **new** two argument functional symbol **added** to our original language $\mathcal{L}$.
We have **eliminated** the quantifier Q_3 in this case.

Step i
At each **Step i**, for $1 \leq i \leq n$), we build a **binary tree** of possibilities:

P1 $Q_i x_i$ is **universal** or **P2** $Q_i x_i$ is **existential** and as result we obtain a formula A_i with one less quantifier. The elimination process builds a sequence of formulas

$$A,\ A_1,\ A_2,\ \ldots,\ A_n = A^*$$

where the formula A belongs to our original language

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

the formula A^* belongs to its **Skolem extension** language (42) defined as follows.

Definition 18

The language $\mathcal{L}^*$ obtained from $\mathcal{L}$ by the quantifiers elimination procedure (40) is is called a **Skolem extension** *of $\mathcal{L}$.*

$$\mathcal{L}^* = \mathcal{L}_{\{\neg,\cup,\cap,\Rightarrow\}}(\mathbf{P}, \mathbf{F} \cup S\mathbf{F},\ \mathbf{C} \cup S\mathbf{C}). \qquad (42)$$ Slang

Observe that in the elimination process (40) a **universal quantifier** introduces free variables in the formula $B(x_1, x_2, \ldots x_n)$. The **elimination** of an existential quantifier that follows universal quantifiers introduces a **new** functional symbol with number of arguments equal the number of universal quantifiers preceding it.

The resulting is an **open** formula A^* of **Skolem extension** language $\mathcal{L}^*$. By **PNF** theorem 10, for any formula A of $\mathcal{L}$ its **PNF** formula (41) exists and is logically equivalent with A. We hence introduce the following definition.

skolemiz **Definition 19 (Skolemization)**

Given a formula A of $\mathcal{L}$.
A formula A^* of the Skolem extension language $\mathcal{L}^*$ (42) obtained from a **PNF** *form of A by the Skolem Procedure (40) is called a* **Skolem form** *of the formula A and the process obtaining it is called a* **Skolemization** *of A.*

Sex **Exercise 3** *Let A be a* **PNF** *formula*

$$\forall y_1 \exists y_2 \forall y_3 \exists y_4\ B(y_1, y_2, y_3, y_4, y_4).$$

Find the **Skolem form** *of A (the formula $B(y_1, y_2, y_3, y_4, y_4)$ is quantifiers free).*

Solution
We eliminate $\forall y_1$ and get a formula A_1

$$\exists y_2 \forall y_3 \exists y_4\ B(y_1, y_2, y_3, y_4).$$

We eliminate $\exists y_2$ by replacing y_2 by $h(y_1)$ where h is a **new** one argument functional symbol **added** to our original language $\mathcal{L}$.
We get a formula A_2

$$\forall y_3 \exists y_4 \; B(y_1, h(y_1), y_3, y_4).$$

We eliminate $\forall y_3$ and get a formula A_3

$$\exists y_4 \; B(y_1, h(y_1), y_3, y_4).$$

We eliminate $\exists y_4$ by replacing y_4 by $f(y_1, y_3)$, where f is a **new** two argument functional symbol **added** to our original language $\mathcal{L}$.
We get a formula A_4 that is our resulting **open** formula A^*

$$B(y_1, h(y_1), y_3, f(y_1, y_3)).$$

Exercise 4

Let now A be a **PNF** *formula*

$$\exists y_1 \forall y_2 \forall y_3 \exists y_4 \exists y_5 \forall y_6 \; B(y_1, y_2, y_3, y_4, y_4, y_5, y_6)$$

Find the **Skolem form** *of A (the formula $B(y_1, y_2, y_3, y_4, y_4, y_5)$ is quantifiers free).*

Solution
We eliminate $\exists y_1$ and get a formula A_1

$$\forall y_2 \forall y_3 \exists y_4 \exists y_5 \forall y_6 \; B(b_1, y_2, y_3, y_4, y_4, y_5, y_6)$$

where b_1 is a **new** constant symbol **added** to our original language $\mathcal{L}$.
We eliminate $\forall y_2, forall y_3$ and get a formulas A_2, A_3; here is the formula A_3

$$\exists y_4 \exists y_5 \forall y_6 \; B(b_1, y_2, y_3, y_4, y_4, y_5, y_6)$$

We eliminate $\exists y_4$ and get a formula A_4

$$\exists y_5 \forall y_6 \; B(b_1, y_2, y_3, g(y_2, y_3), y_5, y_6)$$

where g is a **new** two argument functional symbol **added** to our original language $\mathcal{L}$.
We eliminate $\exists y_5$ and get a formula A_5

$$\forall y_6 \; B(b_1, y_2, y_3, g(y_2, y_3), h(y_2, y_3), y_6)$$

where h is a **new** two argument functional symbol **added** to our original language $\mathcal{L}$.
We eliminate $\forall y_6$ and get a formula A_6 that is the resulting **open** formula A^*

$$B(b_1, y_2, y_3, g(y_2, y_3), h(y_2, y_3), y_6).$$

The **correctness** of the Skolemization process is established by the Skolem theorem 11. It states informally that the formula A^* obtained from a formula A via the Skolemization is satisfiable if and only if the original one is satisfiable. We define this notion formally as follows.

.

eqsat

Definition 20 (Equisatisfiable)

For any formulas A of $\mathcal{L}$ and B of the Skolem extension $\mathcal{L}^*$ (42) of $\mathcal{L}$, we say that A and B are **equisatisfiable** *if and only if the following conditions are satisfied.*

1. *Any structure $\mathcal{M}$ of $\mathcal{L}$ can be extended to a structure $\mathcal{M}^*$ of $\mathcal{L}^*$ and following implication holds.*

$$\textit{If } \ \mathcal{M} \models A, \ \textit{then } \mathcal{M}^* \models B.$$

2. *Any structure $\mathcal{M}^*$ of $\mathcal{L}^*$ can be restricted to a structure $\mathcal{M}$ of $\mathcal{L}$ and following implication holds.*

$$\textit{If } \ \mathcal{M}^* \models B, \ \textit{then } \mathcal{M} \models A.$$

thm:Sk

Theorem 11 (Skolem Theorem)

Let $\mathcal{L}^*$ be the Skolem extension (42) of a language $\mathcal{L}$. Any formula A of $\mathcal{L}$ and its Skolem form *A^* of $\mathcal{L}^*$ are* **equisatisfiable**.

Q-C

4.2 Clausal Form of Formulas

Let $\mathcal{L}^*$ be the Skolem extension of $\mathcal{L}$, i.e. $\mathcal{L}^*$ does not contain quantifiers. We define a proof system $\mathbf{QRS}^*$ as an open formulas language version of **QRS** that includes only its Group 1: Propositional Rules of (7).

We denote the set of formulas of $\mathcal{L}^*$ by $O\mathcal{F}$ to stress the fact that all its formulas are *open* and define $\mathbf{QRS}^*$ formally as follows.

$$\mathbf{QRS}^* = (\mathcal{L}^*, \ \mathcal{E}, \ \ LA, \ \ \mathcal{R}), \tag{43}$$

Qrs

where $\mathcal{E} = \{\Gamma : \quad \Gamma \in O\mathcal{F}^*\}$, LA is defined by (6), and $\mathcal{R}$ contains Group 1: Propositional Rules (7):

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

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

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

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

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

For any formula $A \in O\mathcal{F}$ we define, as we did in chapter **??** its decomposition tree $\mathbf{T}_A$ as follows.

Decomposition tree $\mathbf{T}_A$
Step 1. The formula A is the **root** of $\mathbf{T}_A$ and for any node Δ of the tree we follow the steps bellow.
Step 2. If Δ in *indecomposable*, then Δ becomes a *leaf* of the tree.
Step 3. If Δ is decomposable, then we traverse Δ from left to right to identify the first *decomposable formula B*. In case of a one premisses rule we put is premise as a *leaf;* in case of a two premisses rule we put its left and right premisses as the left and right *leaves*, respectively.

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

We adopt the definition 12 to $\mathbf{QRS}^{*}$ and the language $\mathcal{L}^{*}$.

d:Qsem

Definition 21 (Semantics)

For any sequence Γ of formulas of $\mathcal{L}^{*}$, any structure $\mathcal{M} = [M, I]$ for $\mathcal{L}^{*}$,

$$\mathcal{M} \models\ \Gamma \quad \textit{if and only if} \quad \mathcal{M} \models \delta_{\Gamma},$$

where δ_{Γ} denotes a disjunction *of all formulas in Γ.*

The semantics for clauses is basically the same as for the sequences. We define it, after definition 5, as follows.

Q-Csem

Definition 22 (Clauses Semantics)

For any clause $\mathcal{C}$ of the language $\mathcal{L}^{*}$ (definition 9), $\delta_{\mathcal{C}}$ denotes a disjunction *of all literals in $\mathcal{C}$.*
For any finite set of clauses $\mathbf{C}$ of $\mathcal{L}^{*}$, any structure $\mathcal{M} = [M, I]$ for $\mathcal{L}^{*}$, and any $\mathcal{C} \in \mathbf{C}$,

1. $\mathcal{M} \models\ \mathcal{C}$ *if and only if* $\mathcal{M} \models \delta_{\mathcal{C}}$.

2. $\mathcal{M} \models\ \mathbf{C}$ *if and only if* $\mathcal{M} \models \delta_{\mathcal{C}}$ *for all* $\mathcal{C} \in \mathbf{C}$.

3. $(A \equiv \mathbf{C})$ *if and only if* $A \equiv \sigma_{\mathbf{C}}$,
where $\sigma_{\mathbf{C}}$ is a conjunction *of all formulas $\delta_{\mathcal{C}}$ for all clauses $\mathcal{C} \in \mathbf{C}$.*

Obviously, all rules of $\mathbf{QRS}^*$ are **strongly sound** (definition 7) and theorem 2 holds for $\mathbf{QRS}^*$, i.e. we have the following.

thm:Qss **Theorem 12 (Strong Soundness)**

The proof system $\mathbf{QRS}^*$ *is* **strongly sound**.

We are going to prove now that any formula A of $\mathcal{L}^*$ can be transformed into in logically equivalent set of clauses.

thm:Ceq **Theorem 13 (Formula-Clauses Equivalency)**

For any formula A of $\mathcal{L}^*$, there is an effective procedure of generating a set of clauses $\mathbf{C}_A$ of $\mathcal{L}^*$ such that

$$A \equiv \mathbf{C}_A \tag{44}$$ C-eq

Proof
Given $A \in O\mathcal{F}$. Here is the two steps procedure. S1. We construct (finite and unique) decomposition tree $\mathbf{T}_A$. S2. We form clauses out of the leaves of the tree $\mathbf{T}_A$, i.e. for every leaf L we create a clause $\mathcal{C}_L$ determined by L (definition 10) and we put

$$\mathbf{C}_A = \{\mathcal{C}_L : \quad L \text{ is a leaf of } \quad \mathbf{T}_A\}.$$

Directly from the strong soundness of rules of inference of $\mathbf{QRS}^*$ (theorem 12) and the semantics for clauses definition 22 we get that

$$A \equiv \mathbf{C}_A.$$

Exercise 5

Find he set $\mathbf{C}_A$ of clauses for the following formula A.

$$(((P(b, f(x)) \Rightarrow Q(x)) \cup \neg R(z)) \cup (P(b, f(x)) \cap R(z))))$$

Solution
S1. We construct the decomposition tree for A as follows

$$\mathbf{T}_A$$

$$(((P(b, f(x)) \Rightarrow Q(x)) \cup \neg R(z)) \cup (P(b, f(x)) \cap R(z)))$$

$$\mid (\cup)$$

$$(((P(b, f(x)) \Rightarrow Q(x)) \cup \neg R(z)), (P(b, f(x)) \cap R(z))$$

$$\mid (\cup)$$

$$(P(b, f(x)) \Rightarrow Q(x)), \neg R(z), (P(b, f(x)) \cap R(z))$$

$$\mid (\Rightarrow)$$

$$\neg P(b,f(x)), Q(x), \neg R(z), (P(b,f(x)) \cap R(z))$$

$$\bigwedge(\cap)$$

$$\neg P(b,f(x)), Q(x), \neg R(z), P(b,f(x)) \qquad \neg P(b,f(x)), Q(x), \neg R(z), R(z)$$

S2. The leaves of $\mathbf{T}_A$ are

$L_1 = \neg P(b,f(x)),\ Q(x),\ \neg R(z),\ P(b,f(x))$ and
$L_2 = \neg P(b,f(x)),\ Q(x),\ \neg R(z),\ R(z)$.

The corresponding clauses are
$\mathcal{C}_1 = \{\neg P(b,f(x)), Q(x), \neg R(z), P(b,f(x))\}$ and
$\mathcal{C}_2 = \{\neg P(b,f(x)), Q(x), \neg R(z), R(z)\}$.

The set of clauses is

$$\mathbf{C}_A = \{\{\neg P(b,f(x)), Q(x), \neg R(z), P(b,f(x)\}, \{\neg P(b,f(x)), Q(x), \neg R(z), R(z)\}.$$

Definition 23 *Clausal Form Given a formula A of the language $\mathcal{L}$ and its Skolem form A^* of $\mathcal{L}^*$. The set $\mathbf{C}_{A^*}$ of clauses such that*

$$A^* \ \equiv\ \mathbf{C}_{A^*}$$

s called a **clausal form** *of the formula A of $\mathcal{L}$.*

Exercise 6 *Find the clausal form of a formula A:*

$$(\exists x \forall y\ (R(x,y) \cup \neg P(x)) \Rightarrow \forall y \exists x\ \neg R(x,y)).$$

Solution
Step 1: We rename variables apart in A and get a formula A':

$$(\exists x \forall y\ (R(x,y) \cup \neg P(x)) \Rightarrow \forall z \exists w\ \neg R(z,w)).$$

Step 2: We use Equational Laws of Quantifiers (36), (37)t o pull out $\exists x$ and $\forall y$ and get a formula A'':

$$(\forall x \exists y\ ((R(x,y) \cup \neg P(x)) \Rightarrow \forall z \exists w\ \neg R(z,w)).$$

Step 3: We use Equational Laws of Quantifiers (36), (37)t o pull out $\exists x$ and $\forall y$ and get a formula A''':

$$(\forall x \exists y\ ((R(x,y) \cup \neg P(x)) \Rightarrow \forall z \exists w\ \neg R(z,w)).$$

Step 4: We use Equational Laws of Quantifiers (38), (39)t o pull out $\exists z$ and $\forall w$ from the sub formula $((R(x,y) \cup \neg P(x)) \Rightarrow \forall z \exists w\ \neg R(z,w))$ and get a formula A'''' This is the prenex normal form **PNF** of A.

$$(\forall x \exists y \forall z \exists w\ ((R(x,y) \cup \neg P(x)) \Rightarrow\ \neg R(z,w)). \qquad (45) \quad \boxed{\texttt{Apnf}}$$

Step 5: We perform the Skolemization Procedure (40) to (45). Observe (45) that the formula is of the form of the formula of exercise 3. We follow the exercise and eliminate $\forall x$ and get a formula A_1

$$\exists y \forall z \exists w \ ((R(x,y) \cup \neg P(x)) \Rightarrow \ \neg R(z,w)).$$

We eliminate $\exists y$ by replacing y by $h(x)$ where h is a **new** one argument functional symbol **added** to our original language $\mathcal{L}$.
We get a formula A_2

$$\forall z \exists w \ ((R(x,h(x)) \cup \neg P(x)) \Rightarrow \ \neg R(z,w)).$$

We eliminate $\forall z$ and get a formula A_3

$$\exists w \ ((R(x,h(x)) \cup \neg P(x)) \Rightarrow \ \neg R(z,w)).$$

We eliminate $\exists w$ by replacing w by $f(x,z)$, where f is a **new** two argument functional symbol **added** to our original language $\mathcal{L}$.
We get a formula A_4 that is our resulting **open** formula A^*

$$A^* : \ \ ((R(x,h(x)) \cup \neg P(x)) \Rightarrow \ \neg R(z,(x,z))). \tag{46}$$

Ask

Step 6: We build the decomposition tree $\mathbf{T}_{A^*}$ for (46).

$$\begin{array}{c}
\mathbf{T}_{A^*} \\
((R(x,h(x)) \cup \neg P(x)) \Rightarrow \ \neg R(z,f(x,z))) \\
\mid (\Rightarrow) \\
\neg(R(x,h(x)) \cup \neg P(x)), \ \neg R(z,f(x,z)) \\
\bigwedge(\neg\cup)
\end{array}$$

$$\begin{array}{cc}
\neg R(x,h(x)), \neg R(z,f(x,z) & \begin{array}{c} \neg\neg P(x), \ \neg R(z,f(x,z)) \\ \mid (\neg\neg) \\ P(x), \ \neg R(z,f(x,z)) \end{array}
\end{array}$$

Step 7: The leaves of $\mathbf{T}_{A^*}$ are

$L_1 = \neg R(x,h(x)), \neg R(z,f(x,z)$ and $L_2 = P(x), \ \neg R(z,f(x,z))$.

The corresponding clauses are
$\mathcal{C}_1 = \{\neg R(x,h(x)), \neg R(z,f(x,z)\}$ and
$\mathcal{C}_2 = \{P(x), \ \neg R(z,f(x,z))\}$.

Step 8: The **clausal form** of the formula A

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

is the set of clauses

$$\mathbf{C}_{A^*} = \{ \ \{\neg R(x,h(x)), \neg R(z,f(x,z)\}, \ \ \{P(x), \ \neg R(z,f(x,z))\} \ \}.$$

5 Homework Problems

Hch10

1. Given a predicate (first order) language (1), i.e. $\mathcal{L} = \mathcal{L}_{\{\cap,\cup,\Rightarrow,\neg\}}(\mathbf{P},\mathbf{F},\mathbf{C})$. Let **QRS** de a proof system (8). For any formulas A, B of $\mathcal{L}$, we define:

   $$\vdash_{QRS} A \equiv B \quad \text{it and only if} \quad \vdash_{QRS} (A \Rightarrow B) \;\; \text{and} \;\; \vdash_{QRS} (B \Rightarrow A).$$

   Show that for any formulas $A(x), B(x)$ with a free variable x the following holds.
   Remider: 1. you treat $A(x), B(x)$ as atomic foprmulas, 2.you must transform formulas with restricted domain quantifiers into proper formulas of $\mathcal{L}$.

   (i) $\vdash_{QRS} \forall x\ (A(x) \cap B(x)) \ \equiv\ (\forall x A(x) \cap \forall x B(x))$

   (ii) $\vdash_{QRS} \exists x\ (A(x) \cup B(x)) \ \equiv\ (\exists x A(x) \cup \exists x B(x))$.

   (iii) $\vdash_{QRS} \neg\forall_{B(x)}\ A(x) \equiv \exists_{B(x)}$.

   (iv) $\vdash_{QRS} \neg\exists_{B(x)}\ A(x) \equiv \forall_{B(x)}\neg A(x)$.

   (v) $\vdash_{QRS} \neg\forall x A(x) \equiv \exists x \neg A(x)$.

   (vi) $\vdash_{QRS} \neg\exists x A(x) \equiv \forall x \neg A(x)$.

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

2. Show that for any formulas $A(x), B\ B$ where B does not contain any free occurrence of x the following holds.

   (i) $\vdash_{QRS} \forall x(A(x) \cap B) \equiv (\forall x A(x) \cap B)$.

   (ii) $\vdash_{QRS} \forall x(A(x) \cup B) \equiv (\forall x A(x) \cup B)$.

   (iii) $\vdash_{QRS} \exists x(A(x) \Rightarrow B) \equiv (\forall x A(x)$.

   (iv) $\vdash_{QRS} \exists x(A(x) \Rightarrow B) \equiv (\forall x A(x)$.

3. Prove that following formulas are not provable in **QRS**.
   Remider: you must transform formulas with restricted domain quantifiers into proper formulas of $\mathcal{L}$.

   (i) $\exists_{C(x)}(A(x) \cup B) \not\equiv (\exists_{C(x)} A(x) \cup B)$.

   (ii) $\forall_{C(x)}(A(x) \cap B) \not\equiv (\forall_{C(x)} A(x) \cap B)$.

   (iii) $\exists_{C(x)}(A(x) \Rightarrow B) \not\equiv (\forall_{C(x)} A(x) \Rightarrow B)$.

   (iv) $\exists_{C(x)}(B \Rightarrow A(x)) \not\equiv (B \Rightarrow \exists x A(x))$.

4. Prove that following formulas are not provable in **QRS**.

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

   (ii) $(\forall x \exists y\ A(x,y) \Rightarrow \exists x \forall y\ A(x,y))$.

   (iii) $(\exists x \exists y\ A(x,y) \Rightarrow \exists y\ A(y,y))$.

   (iv) $(\forall x \exists y\ A(x,y) \Rightarrow \exists y\ A(y,y))$.

   (v) $(\forall x\ (A(x) \Rightarrow\ B(x)) \Rightarrow (\forall x\ A(x) \Rightarrow \exists x\ B(x)))$.

5. Prove that following formulas are not provable in **QRS**.

   (i) $A_1: \ \forall x \neg \exists y(P(x, g(y, y)) \cup P(x, g(g(y, y), d)))$.

   (ii) $A_2: \ (\neg \forall y P(f(x, y), c) \Rightarrow (P(x, c) \cup P(y, c)))$

   (iii) $A_3: \ \forall x(P(x) \Rightarrow \exists y Q(x, y))$.

   (iv)$A_4: \ \forall x \neg \exists y(P(x) \cap \neg Q(x, y))$.

6. Find counter-models determined by the decomposition trees $\mathbf{T}_{A_i}$ for the following formulas A_i, $i = 1, 2, 3, 4$.

   (i) $A_1: \ \forall x \neg \exists y(Q(x, g(y)) \cup R(x, f(x, y), c)))$.

   (ii) $A_2: \ (\neg \forall y R(f(x, y), c) \Rightarrow (Q(x, c) \cup Q(y, c)))$

   (iii) $A_3: \ \forall x(P(x) \Rightarrow \exists y Q(x, y))$.

   (iv)$A_4: \ \forall x \neg \exists y(P(x) \cap \neg Q(f(x, y)))$.

7. Find prenex normal form **PNF** of the following formulas.
   Reminder: We assume that the formula A in **PNF** is always **closed**. If it is not closed we form its **closure** (definition 16) instead.

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

   (ii) $((\forall x Q(x) \Rightarrow (\exists x R(x) \cup \neg \forall x Q(x))) \Rightarrow (\neg \exists x Q(x) \cap R(x)))$.

   (iii) $(\forall x \ R(f(x, y), c) \Rightarrow \ (\exists x R(f(x, y), c)) \cap \neg R(f(x, y), c)) \Rightarrow (\neg \forall x \ R(f(x, y), c) \Rightarrow \exists x \ R(f(x, y), c)))$.

   (iv) $((\exists_{R(y)} P(y) \Rightarrow Q(x)) \Rightarrow (P(y) \Rightarrow \exists x Q(x)))$

8. Find a **Skolem form** of the following formulas (the formula $B(y_1, y_2, y_3, y_4, y_4)$ is quantifiers free).

   (i) $\forall y_1 \forall y_2 \forall y_3 \exists y_4 \ B(y_1, y_2, y_3, y_4, y_4)$.

   (ii) $\exists y_1 \exists y_2 \forall y_3 \exists y_4 \ B(y_1, y_2, y_3, y_4, y_4)$.

   (iii) $\exists y_1 \forall y_2 \exists y_3 \exists y_4 \ B(y_1, y_2, y_3, y_4, y_4)$.

   (iv) $\forall y_1 \forall y_2 \exists y_3 \exists y_4 \ B(y_1, y_2, y_3, y_4, y_4)$.

9. Find the clausal form of the following formulas.

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

   (ii) $((\forall x Q(x) \Rightarrow (\exists x R(x) \cup \neg \forall x Q(x))) \Rightarrow (\neg \exists x Q(x) \cap R(x)))$.

   (iii) $(\forall x \ R(f(x, y), c) \Rightarrow \ (\exists x R(f(x, y), c)) \cap \neg R(f(x, y), c)) \Rightarrow (\neg \forall x \ R(f(x, y), c) \Rightarrow \exists x \ R(f(x, y), c)))$.

   (iv) $((\exists_{R(y)} P(y) \Rightarrow Q(x)) \Rightarrow (P(y) \Rightarrow \exists x Q(x)))$

10. Find the set of clauses logically equivalent to clausal form of the following formulas.

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

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

(iii) $(\neg(\neg a => (a \cap \neg b)) => (\neg a \cap (\neg a \cup \neg b)))$.

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

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