cse371/ mat371
LOGIC

Professor Anita Wasilewska

Fall 2015

LECTURE 13

Chapter 13
Predicate Logic Proof System **QRS**

Part 1: Predicate Languages

Part 2: Proof System **QRS**

Chapter 13
Part 1: Predicate Languages

Predicate Languages

Predicate Languages are also called First Order Languages

The same applies to the use of terms for Propositional and Predicate Logic

Propositional and **Predicate Logics** called Zero Order and First Order Logics, respectively and we will use both terms equally

We usually work with different predicate languages, depending on what applications we have in mind

All **predicate languages** have some common features, and we begin with these

Predicate Languages Components

Propositional Connectives

Predicate Languages **extend** a notion of the propositional languages so we define the set CON of their propositional connectives as follows

The set CON of **propositional connectives** is a **finite** and **non-empty** and

$$CON = C_1 \cup C_2$$

where C_1, C_2 are the sets of one and two arguments connectives, respectively

Parenthesis

As in the propositional case, we adopt the signs (and) for our parenthesis., i.e. we define a set PAR as

$$PAR = \{ \, (, \ ) \, \}$$

Predicate Languages Components

Quantifiers

We adopt two quantifiers; the **universal quantifier** denoted by $\forall$ and the **existential quantifier** denoted by $\exists$, i.e. we have the following set **Q** of quantifiers

$$\mathbf{Q} = \{\forall, \ \exists\}$$

In a case of the classical logic and the logics that **extend it**, it is possible to adopt only one quantifier and to **define the other** in terms of it and propositional connectives

Such definability is impossible in a case of some non-classical logics, for example the intuitionistic logic

But even in the case of **classical logic** the two quantifiers express better the common intuition, so we adopt the both of them

Predicate Languages Components

Variables

We assume that we always have a **countably infinite** set VAR of variables, i.e. we assume that

$$cardVAR = \aleph_0$$

We denote variables by $x, y, z, ...$, with indices, if necessary.

we often express it by writing

$$VAR = \{x_1, x_2, ...\}$$

Note

Predicate Languages Components

The set *CON* of **propositional connectives** defines a propositional part of the **predicate logic language**

Observe that what really differ one **predicate language** from the other is the choice of additional symbols added to the symbols just described

These **additional symbols** are: predicate symbols, function symbols, and constant symbols

A **particular** predicate language is determined by specifying these additional sets of symbols

They are defined as follows

Predicate Languages Components

Predicate symbols

Predicate symbols **represent relations**

Any predicate language must have **at least one** predicate symbol

Hence we assume that any predicate language contains a non empty, finite or countably infinite set

$$\mathbf{P}$$

of **predicate symbols**, i.e. we assume that

$$0 < card\mathbf{P} \leq \aleph_0$$

We denote predicate symbols by $P, Q, R, \ldots$, with indices, if necessary

Each predicate symbol $P \in \mathbf{P}$ has a positive integer $\#P$ assigned to it; when $\#P = n$ we **call** P an n-ary (n - place) **predicate (relation) symbol**

Predicate Languages Components

Function symbols

We assume that any predicate language contains a finite (may be empty) or countably infinite set $\mathbf{F}$ of **function symbols**

I.e. we assume that

$$0 \leq card\mathbf{F} \leq \aleph_0$$

When the set $\mathbf{F}$ is empty we say that we deal with a **language without functional symbols**

We denote functional symbols by $f, g, h, ...$ with indices, if necessary

Similarly, as in the case of predicate symbols, each **function symbol** $f \in \mathbf{F}$ has a positive integer $\# f$ assigned to it; if $\# f = n$ then f is called an n-ary (n - place) **function symbol**

Constant symbols

We also assume that we have a finite (may be empty) or countably infinite set

$$\mathbf{C}$$

of **constant symbols**

I.e. we assume that

$$0 \leq card\mathbf{C} \leq \aleph_0$$

The elements of $\mathbf{C}$ are **denoted** by $c, d, e...$, with indices, if necessary

We often express it by putting

$$\mathbf{C} = \{c_1, c_2, ...\}$$

When the set $\mathbf{C}$ is **empty** we say that we deal with a language without constant symbols

Alphabet of Predicate Languages

Sometimes the **constant symbols** are defined as **0-ary function symbols**, i.e. we have that

$$\mathbf{C} \subseteq \mathbf{F}$$

We single them out as a separate set for our convenience

We assume that all of the above sets of symbols are **disjoint**

Alphabet

The union of all of above disjoint sets of symbols is called the **alphabet** $\mathcal{A}$ of the **predicate language**, i.e. we **define**

$$\mathcal{A} = VAR \cup CON \cup PAR \cup \mathbf{Q} \cup \mathbf{P} \cup \mathbf{F} \cup \mathbf{C}$$

Predicate Languages Notation

Observe, that once the set of propositional connectives is fixed, the **predicate language** is determined by the sets **P, F** and **C**

We use the **notation**

$$\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$$

for the **predicate language** $\mathcal{L}$ **determined** by **P, F, C**

If there is no danger of confusion, we may **abbreviate** $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$ to just $\mathcal{L}$

If the set of propositional connectives involved is not fixed, we also use the notation

$$\mathcal{L}_{CON}(\mathbf{P}, \mathbf{F}, \mathbf{C})$$

to denote the **predicate language** $\mathcal{L}$ **determined** by **P, F, C** and the set of propositional connectives *CON*

Predicate Languages Notation

We sometimes allow the same symbol to be used as an n-place relation symbol, and also as an m-place one; no confusion should arise because the different uses can be told apart easily

Example

If we write $P(x, y)$, the symbol P denotes **2-argument** predicate symbol

If we write $P(x, y, z)$, the symbol P denotes **3-argument** predicate symbol

Similarly for **function symbols**

Two more Predicate Language Components

Having defined the alphabet we now complete the formal **definition of the predicate language** by defining two more components:

the set $\mathbf{T}$ of all **terms** and

the set $\mathcal{F}$ of all **well formed formulas**

of the **language** $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$

Set of Terms

Terms

The set $\mathbf{T}$ of **terms** of the **predicate language** $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$ is the **smallest** set

$$\mathbf{T} \subseteq \mathcal{A}^*$$

meeting the conditions:

1. any variable is a **term**, i.e. $VAR \subseteq \mathbf{T}$
2. any constant symbol is a **term,** i.e. $\mathbf{C} \subseteq \mathbf{T}$
3. if f is an n-place **function symbol**, i.e. $f \in \mathbf{F}$ and $\#f = n$ and $t_1, t_2, ..., t_n \in T$, then $f(t_1, t_2, ..., t_n) \in \mathbf{T}$

Terms Examples

Example 1

Let $f \in \mathbf{F}, \# f = 1$, i.e. f is a 1-place **function symbol**

Let x, y be **variables**, c, d be **constants**, i.e.
$x, y \in VAR, c, d \in \mathbf{C}$

Then the following expressions are **terms**:

$$x,\ y,\ f(x),\ f(y),\ f(c), f(d),\ f(f(x)),\ f(f(y)),\ f(f(c)),\ f(f(d)), \ldots$$

Example 2

Let $\mathbf{F} = \emptyset, \mathbf{C} = \emptyset$

In this case terms consists of **variables only**, i.e.

$$\mathbf{T} = VAR = \{x_1, x_2, \ldots.\ \}$$

Terms Examples

Directly from the **Example 2** we get the following

REMARK

For any predicate language $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$, the set $\mathbf{T}$ of its **terms** is always **non-empty**

Example 3

Let $f \in \mathbf{F}, \# f = 1$, $g \in \mathbf{F}, \# g = 2$, $x, y \in VAR, c, d \in \mathbf{C}$

Some of the **terms** are the following:

$$f(g(x,y)),\ f(g(c,x)),\ g(f(f(c)), g(x,y)),$$

$$g(c, g(x, f(c))), g(f(g(x,y)), g(x, f(c))) \ ....$$

Terms Notation

From time to time, the logicians are and we may be informal about how we write terms

Example

If we **denote** a 2- place function symbol g by $+$, we **may write** $x + y$ instead $+(x, y)$

Because in this case we can think of $x + y$ as an unofficial way of designating the **"real" term** $g(x, y)$

Atomic Formulas

Before we define the **set of formulas**, we need to define one more set; the set of **atomic**, or **elementary** formulas

Atomic formulas are the simplest formulas as the propositional variables were in the case of **propositional languages**

Atomic Formulas

Definition

An **atomic formula** of a **predicate language** $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$ is any element of $\mathcal{A}^*$ of the form

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

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

I.e. R is n-ary **relational symbol** and $t_1, t_2, ..., t_n$ are **any terms**

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

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

Atomic Formulas Examples

Example 1

Consider a language $\mathcal{L}(\{P\}, \emptyset, \emptyset)$, for #P = 1

Our language

$$\mathcal{L} = \mathcal{L}(\{P\}, \emptyset, \ \emptyset)$$

is a language **without** neither **functional**, nor **constant** symbols, and with one, 1-place **predicate** symbol P

The set of **atomic formulas** contains all formulas of the form $P(x)$, for x any variable, i.e.

$$A\mathcal{F} = \{P(x) : x \in VAR\}$$

Atomic Formulas Examples

Example 2

Let now consider a **predicate language**

$$\mathcal{L} = \mathcal{L}(\{R\}, \{f, g\}, \ \{c, d\})$$

for $\#f = 1, \#g = 2, \#R = 2$

The language $\mathcal{L}$ has **two functional symbols:** 1-place symbol f and 2-place symbol g, one 2-place **predicate symbol** R, and two **constants:** c,d

Some of the **atomic formulas** in this case are the following.

$$R(c, d), \ \ R(x, f(c)), \ \ R((g(x, y)), f(g(c, x))),$$

$$R(y, \ g(c, g(x, f(d)))) \ .....$$

Set of Formulas Definition

Now we are ready to define the set $\mathcal{F}$ of all **well formed formulas** of any **predicate language** $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$

Definition

The set $\mathcal{F}$ of all **well formed formulas**, called shortly **set of formulas**, of the language $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$ is the smallest set meeting the following **four conditions:**

1. Any **atomic formula** of $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$ is a **formula** , i.e.

$$A\mathcal{F} \subseteq \mathcal{F}$$

2. If A is a formula of $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$, $\triangledown$ is an one argument **propositional connective**, then $\triangledown A$ is a formula of $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$, i.e. the following **recursive condition** holds

$$\text{if } A \in \mathcal{F}, \triangledown \in C_1 \text{ then } \triangledown A \in \mathcal{F}$$

Set of Formulas Definition

3. If A, B are **formulas** of $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$ and $\circ$ is a two argument **propositional connective**, then $(A \circ B)$ is a **formula** of $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$, i.e. the following **recursive condition** holds

   If $A \in \mathcal{F}, \nabla \in C_2$, then $(A \circ B) \in \mathcal{F}$

4. If A is a **formula** of $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$ and x is a **variable**, $\forall, \exists \in \mathbf{Q}$, then $\forall xA$, $\exists xA$ are **formulas** of $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$, i.e. the following recursive condition holds

   If $A \in \mathcal{F}$, $x \in VAR$, $\forall, \exists \in \mathbf{Q}$, then $\forall xA$, $\exists xA \in \mathcal{F}$

Scope of the Quantifier

Another important notion of the **predicate language** is the notion of **scope of a quantifier**

It is defined as follows

Definition

Given formulas $\forall xA$, $\exists xA$, the formula A is said to be in the **scope of the quantifier** $\forall$, $\exists$, respectively.

Example 3

Let $\mathcal{L}$ be a language of the previous **Example 2** with the set of connectives $\{\cap, \cup, \Rightarrow, \neg\}$, i.e. let's consider

$$\mathcal{L} = \mathcal{L}_{\{\cap,\cup,\Rightarrow,\neg\}}(\{f, g\}, \{R\}, \{c, d\})$$

for $\#f = 1$, $\#g = 2$, $\#R = 2$

Some of the formulas of $\mathcal{L}$ are the following.

$$R(c, d), \quad \exists yR(y, f(c)), \quad \neg R(x, y),$$

$$(\exists xR(x, f(c)) \Rightarrow \neg R(x, y)), \quad (R(c, d) \cap \forall zR(z, f(c))),$$

$$\forall yR(y, \; g(c, g(x, f(c)))), \quad \forall y\neg\exists xR(x, y)$$

Scope of Quantifiers

The formula $R(x, f(c))$ is in **scope of the quantifier** $\exists$ in the formula

$$\exists x R(x, f(c))$$

The formula $(\exists_x R(x, f(c)) \Rightarrow \neg R(x, y))$ **is not in scope of any quantifier**

The formula $(\exists x R(x, f(c)) \Rightarrow \neg R(x, y))$ is in **scope** of quantifier $\forall$ in the formula

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

Predicate Language Definition

Now we are ready to define formally a **predicate language**

Let $\mathcal{A}, T, \mathcal{F}$ be the **alphabet** , the set of **terms** and the set of **formulas** as already defined

Definition

A **predicate language** $\mathcal{L}$ is a triple

$$\mathcal{L} = (\mathcal{A}, \mathbf{T}, \mathcal{F})$$

As we have said before, the language $\mathcal{L}$ is determined by the **choice** of the symbols of its **alphabet**, namely of the **choice** of **connectives**, **predicates**, **functions** , and **constants** symbols

If we want specifically mention these **choices**, we write

$$\mathcal{L} = \mathcal{L}_{CON}(\mathbf{P}, \mathbf{F}, \mathbf{C}) \text{ or } \mathcal{L} = \mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$$

Free and Bound Variables

Given a **predicate language** $\mathcal{L} = (\mathcal{A}, T, \mathcal{F})$, we must distinguish between formulas like

$$P(x, y), \quad \forall x P(x, y) \quad \text{and} \quad \forall_x \exists_y P(x, y)$$

This is done by introducing the notion of **free** and **bound** variables, and **open** and **closed** formulas

Closed formulas are also called **sentences**

Informally, in the formula

$$P(x, y)$$

both variables x and y are called **free** variables

They **are not** in the **scope** of any quantifier

The formula of that type, i.e. formula **without quantifiers** is an **open formula**

Free and Bound Variables

In the formula

$$\forall y P(x, y)$$

the variable x is **free**, the variable y is **bounded** by the the quantifier $\forall$

In the formula

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

both x and y are **free**

In the formulas

$$\forall z P(z, y), \quad \forall x P(x, y)$$

only the variable y is **free**

Free and Bound Variables

In the formula

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

there is **no free variables**

In the formula

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

the variable x (in $Q(x, y)$) is **free**

Sometimes in order to distinguish more easily **which** variable is **free** and which is **bound** in the formula we might use the bold face type for the quantifier bound variables, i.e. to write the last formulas as

$$(\forall \mathbf{x} P(\mathbf{x}) \Rightarrow \exists \mathbf{y} Q(x, \mathbf{y}))$$

Free and Bound Variables Formal Definition

Definition

The set $FV(A)$ of **free variables** of a formula A is defined by the **induction** of the degree of the formula as follows.

1. If A is an **atomic** formula, i.e. $A \in A\mathcal{F}$, then $FV(A)$ is just the set of variables appearing in A;
2. for any **unary** propositional connective, i.e. for any $\triangledown \in C_1$
$$FV(\triangledown A) = FV(A)$$
i.e. the **free** variables of $\triangledown A$ are the **free** variables of A;
3. for any **binary** propositional connective, i.e, for any $\circ \in C_2$
$$FV(A \circ B) = FV(A) \cup FV(B)$$
i.e. the **free** variables of $(A \circ B)$ are the **free** variables of A together with the **free** variables of B;
4. $FV(\forall xA) = FV(\exists xA) = FV(A) - \{x\}$ i.e. the **free** variables of $\forall xA$ and $\exists xA$ are the **free** variables of A, **except** for x

Bound Variables, Sentence, Open Formula

Bound variables: a variable is called bound if it is not free

Sentence: a formula with **no free variables** is called a sentence

Open formula: a formula with **no bound variables** is called an open formula

Example

The formulas

$$\exists x Q(c, g(x, d)), \qquad \neg \forall x (P(x) \Rightarrow \exists y (R(f(x), y) \cap \neg P(c)))$$

are **sentences**

The formulas

$$Q(c, g(x, d)), \qquad \neg (P(x) \Rightarrow (R(f(x), y) \cap \neg P(c)))$$

are **open formulas**

Examples

Example

The formulas

$$\exists x Q(c, g(x, y)), \qquad \neg(P(x) \Rightarrow \exists y(R(f(x), y) \cap \neg P(c)))$$

are **neither** sentences **nor** open formulas

They contain **some free** and **some bound** variables;

the variable y is **free** in $\exists \mathbf{x} Q(c, g(\mathbf{x}, y))$

the variable x is **free** in $\neg(P(x) \Rightarrow \exists \mathbf{y}(R(f(x), \mathbf{y}) \cap \neg P(c)))$

Notations

Notation: It is common practice to use the notation

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

to indicate that

$$FV(A) \subseteq \{x_1, x_2, ..., x_n\}$$

without implying that **all of** $x_1, x_2, ..., x_n$ are actually **free** in A

This is similar to the practice in **algebra** of writing $p(x_1, x_2, ..., x_n)$ for a polynomial p in the variables $x_1, x_2, ..., x_n$ without implying that **all of** them have nonzero coefficients

Notations

Replacing x by $t \in \mathbf{T}$ in A

If $A(x)$ is a formula, and t is a term then

$$A(t/x)$$

or, more simply,

$$A(t)$$

denotes the result of replacing **all** occurrences of the free variable x by the term t throughout

Notation

When using the notation

$$A(t)$$

we always **assume** that **none** of the variables in t occur as **bound** variables in A

Notations

Rememeber

When **replacing** x by $t \in \mathbf{T}$ in a formula A, we **denote** the result as

$$A(t)$$

and do it under the **assumption** that **none** of the variables in t occur as **bound** variables in A

The assumption that **none** of the variables in t **occur as bound** variables in $A(t)$ is **essential** because **otherwise** by **substituting** t on the place of x we **would distort** the meaning of $A(t)$

Example

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

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

i.e. the variable y in t **is bound** 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 $\exists y(x \neq y)$

But if $t = z$, i.e. the variable z in t is **not bound** in A, then $A(t/x) = A(t)$ is

$$\exists y(z \neq y)$$

and express the **same meaning** as $A(x)$

Remark that if for example $t = f(z, x)$ we obtain $\exists y(f(z, x) \neq y)$ as a result of substitution of $t = f(z, x)$ for x in $\exists y(x \neq y)$

Chapter 13

Part 2: Gentzen Style Proof System for Classical Predicate Logic

The System **QRS**

The System **QRS**

Let $\mathcal{F}$ be a set of formulas of a **predicate language**

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

for **P, F, C** countably infinite sets of predicate, functional, and constant symbols, respectively

The **rules of inference** of the system **QRS** operate, as in the propositional case, on **finite sequences of formulas**, i.e. on elements of $\mathcal{F}^*$

We will denote, as previously the 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:

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

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

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

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

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

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

$$LT \subseteq \mathcal{F}$$

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

$$LT = \{A \in \mathcal{F} : A \in A\mathcal{F}\} \cup \{\neg A \in \mathcal{F} : A \in A\mathcal{F}\}$$

The elements of the set $\{A \in \mathcal{F} : A \in A\mathcal{F}\}$ are called **positive literals**

The elements of the set $\{\neg A \in \mathcal{F} : A \in A\mathcal{F}\}$ are called **negative literals**

Sequences of Literals

We denote by

$$\Gamma^{'}, \quad \Delta^{'}, \quad \Sigma^{'} \ldots$$

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

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

We will denote by

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

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

Connectives Inference Rules of **QRS**

Group 1

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

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

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

Connectives Inference Rules of **QRS**

Group 1

Implication rules

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

Negation rule

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

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

Quantifiers Inference Rules of **QRS**

Group 2: Universal Quantifier rules

$$(\forall)\ \frac{\Gamma',\ A(y),\ \Delta}{\Gamma',\ \forall_x A(x),\ \Delta} \qquad (\neg\forall)\ \frac{\Gamma',\ \neg\forall_x A(x),\ \Delta}{\Gamma',\ \exists_x \neg A(x),\ \Delta}$$

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

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

The condition: the variable y **does not appear** in **any formula** in the conclusion of $(\forall)$ is called the eigenvariable condition

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

Quantifiers Inference Rules of **QRS**

Group 2: Existential Quantifier rules

$$(\exists)\ \frac{\Gamma',\ A(t),\ \Delta, \exists_x A(x)}{\Gamma',\ \exists_x A(x),\ \Delta} \qquad (\neg\exists)\ \frac{\Gamma',\ \neg\exists_x A(x),\ \Delta}{\Gamma',\ \forall_x \neg A(x),\ \Delta}$$

where $t \in T$ is an arbitrary term, $\Gamma' \in LT^*$, $\Delta \in \mathcal{F}^*$, $A, B \in \mathcal{F}$

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

QRS Decomposition Trees

Given a formula $A \in \mathcal{F}$, we define its **decomposition tree** $\mathcal{T}_A$ in a similar way as in the propositional case

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**

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

We define the **decomposition tree** in the case of the **propositional rules** and the rules $(\neg\forall)$, $(\neg\exists)$ in the exactly the same way as in the **propositional case**

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 $\Gamma', \forall x A(x), \Delta$ or $\Gamma', \exists x A(x), \Delta$, i.e. to deal with the rules $(\forall)$ and $(\exists)$

we assume that **all terms** form a one-to one sequence

$$ST \qquad t_1, t_2, ..., t_n, ...$$

Observe, that by the definition, all free variables are terms, hence all free variables **appear** in the sequence ST of all terms

QRS Decomposition Trees

Let Γ be a sequence on the tree in which the first **indecomposable formula** has $\forall$ as its **main connective**

It means that Γ is of the form

$$\Gamma^{'},\ \forall_x A(x),\ \Delta$$

We write a sequence

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

below it on the tree, i.e. **as its child**,

where the variable y fulfills the following condition

C1: 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 **C1** corresponds to the **restriction** put on the application of the rule $(\forall)$

QRS Decomposition Trees

Let now first **indecomposable formula** in Γ has $\exists$ as its **main connective**

It means that Γ is of the form

$$\Gamma^{'},\ \exists_x A(x),\ \Delta$$

We e write a sequence

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

as **its child**,

where the term t fulfills the following conditions

C2: 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** $\Gamma^{'}, A(t), \Delta$

QRS Decomposition Trees

Observe that the sequence ST of all terms is one- to - one and by the conditions **C1** and **C1** 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** $(\forall)$ and $(\exists)$

From all above, and we conclude the following.

Uniqueness Theorem

For any formula $A \in \mathcal{F}$, its decomposition tree $\mathcal{T}_A$ is **unique**

Moreover, by definition we have that

If $\mathcal{T}_A$ is **finite** and all its leaves are axioms, then $\mathcal{T}_A$ is a proof of A in **QRS**, i.e. $\vdash A$

If $\mathcal{T}_A$ is **finite** and contains a non-axiom leaf or is **infinite**, then $\nvdash A$

Examples of Decomposition Trees

In all the examples below, the formulas $A(x), B(x)$ represent **any formuls**

But as there is no indication about their particular components, so they are treated as **indecomposable formulas**

The decomposition tree of the formula A reprezenting the **de Morgan Law**

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

is constructed as follows

Examples of Decomposition Trees

Here is the $\mathcal{T}_A$

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

$$\mid (\Rightarrow)$$

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

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

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

$$\mid (\forall)$$

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

where x_1 is a first free variable in the sequence ST 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 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 $\mathcal{T}_A$ ended with one leaf being axiom, so it represents a proof in **QRS** of the **de Morgan Law**

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

i.e. we have proved that

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

The decomposition tree $\mathcal{T}_A$ for a formula

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

is constructed as follows

Examples of Decomposition Trees

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

$$\mid (\Rightarrow)$$

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

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

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

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

$$\mid (\exists)$$

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

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

$$\mid (\exists)$$

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

where t_1 is the first term in the sequence 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

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

in **not a tautology** of predicate language (with formal semantics yet to come!)

Let's now look at its **decomposition tree**

Examples of Decomposition Trees

$$\exists x A(x)$$

$$\mid (\exists)$$

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

where t_1 is the first term in the sequence **??**, 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 repeat the procedure

$$\mid (\exists)$$

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

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. $t_4 \neq t_3 \neq t_2 \neq t_1$

$$\mid (\exists)$$

.....

$$\mid (\exists)$$

.....

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

$$\nvdash \ \exists x A(x)$$

Examples of Decomposition Trees

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

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

and show that the proof in **QRS** of the inverse implication

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

does not exist, i.e. that

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

The decomposition tree of the first formula is the following

Examples of Decomposition Trees

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

$$\mid (\Rightarrow)$$

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

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

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

$$\mid (\forall)$$

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

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

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

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

$$\bigwedge (\cap)$$

Examples of Decomposition Trees

$$\bigwedge(\cap)$$

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

$$\mid (\exists)$$

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

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

$$\mid (\exists)$$

....

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

axiom

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

$$\mid (\exists)$$

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

$$\mid (\exists)$$

...

$$\mid (\exists)$$

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

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

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

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 a leaf

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

which is an **axiom**