cse541
LOGIC for Computer Science

Professor Anita Wasilewska

LECTURE 10a

Chapter 10
Predicate Automated Proof Systems
Completeness of Classical Predicate Logic

Slides Set 2

PART 3: **Skolemization and Clauses**

Skolemization and Clauses : Introduction

A **resolution** based proof system for predicate logic operates on sets of **clauses** as a basic expressions and uses a resolution rule as the only rule of inference

The **first 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 its **logically equivalent** set of clauses

$$\mathbf{C}_A$$

Skolemization and Clauses: Introduction

This process of transformation is done in two stages

S1. We convert any formula A of the predicate language $\mathcal{L}$ into an **open** formula A^* of a language $\mathcal{L}^*$ by a process of **elimination of quantifiers** from the original language $\mathcal{L}$

The elimination method is due to T. Skolem (1920) and is called Skolemization

Skolem Theorem

The resulting formula A^* is equisatisfiable with A:

it is **satisfiable** if and only if the original one is **satisfiable**

Skolemization and Clauses; Introduction

The stage **S1.** is performed as the first step in a resolution based automated theorem prover

S2. We define a proof system **QRS*** based on the Skolemized language

$$\mathcal{L}^*$$

and use it transform automatically any formula A^* of $\mathcal{L}^*$ into an logically equivalent set of clauses

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

Skolemization and Clauses; Introduction

The final result of stages **S1.** and **S2.**, i.e. the set

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

of clauses of the Skolemized language $\mathcal{L}^*$ called a **clausal form** of the original formula A of the language $\mathcal{L}$

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**

Clauses: Definition

Definition

Given a formal 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** C is a **finite set** of literals

Empty clause is denoted by $\{\}$

3. We denote by **C** any finite set of all **clauses**. For any $n \geq 0$,

$$\mathbf{C} = \{C_1, C_2, \ldots C_n\}$$

Clauses: Definition

Definition

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

$$\Gamma \in LT^*$$

determined by Γ is a **set** form out of all elements of the sequence Γ

We we denote it by

$$C_\Gamma$$

Example

Example

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

$$\mathcal{C}_{\Gamma_1} = \mathcal{C}_{\Gamma_2} = \{\neg P(x_1), \neg R(x_1, y), P(x_2)\}$$

Clauses Semantics

Given a propositional or predicate language L

We use the following notations

For any **clause** C, write

$$\delta_C$$

for a disjunction of all literals in C

Let $\mathcal{M}$ denote a **structure** $[M, I]$ for a predicate language $\mathcal{L}$, or a **truth assignment** v in case when $\mathcal{L}$ is a propositional language

Clauses Semantics

Definition

$\mathcal{M}$ is called a **model** for a clause C

$$\mathcal{M} \models C, \quad \text{if and only if} \quad \mathcal{M} \models \delta_C$$

$\mathcal{M}$ is called a **model** for a **set** $\mathbf{C}$ of clauses,

$$\mathcal{M} \models \mathbf{C} \quad \text{if and only if} \quad \mathcal{M} \models C \quad \text{for all clauses } C \in \mathbf{C}$$

Clauses Semantics

Definition

A formula A is **equivalent** with a set $\mathbf{C}$ of clauses

$$(A \equiv \mathbf{C}) \quad \text{if and only if} \quad A \equiv \sigma_{\mathbf{C}}$$

where $\sigma_{\mathbf{C}}$ is a conjunction of all formulas δ_C for all clauses $C \in \mathbf{C}$

Propositional Formula-Clauses Equivalency

Theorem (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

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

Proof

Given a formula A, we first use the **RS** system (chapter 6) to build a **decomposition tree** $\mathbf{T}_A$ of A

We form clauses out of the **leaves** of the tree $\mathbf{T}_A$, i.e. for every leaf L we create a clause C_L determined by L

Propositional Formula-Clauses Equivalency

We put

$$\mathbf{C}_A = \{C_L : \ L \text{ is a leaf of } \ \mathbf{T}_A\}$$

Directly from the **strong soundness** of rules of inference of **RS** we get

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

This ends the **proof** for the propositional case

Example

Example Consider a decomposition tree

$$\mathbf{T}_A$$

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

$$\mid (\cup)$$

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

$$\bigwedge (\cap)$$

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

$$\mid (\Rightarrow)$$

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

$$\mid (\Rightarrow)$$

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

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

$$\mid (\Rightarrow)$$

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

Example

For the formula

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

the leaves of its tree $\mathbf{T}_A$ are

$$L_1 = \neg a, b, \neg a, c \quad \text{and} \quad L_2 = \neg c, \neg a, c$$

The set of clauses determined by them is

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

By the Formula-Clauses Equivalency **Theorem**

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

Semantically it means that

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

Predicate Clausal Form

Theorem

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

$$(*) \quad A^* \equiv \mathbf{C}_{A^*}$$

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

Proof of Theorem

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** and **Skolem Theorem** described in the next section

As the next step, we define there a proof system **QRS*** based on the **quantifiers free** language $\mathcal{L}^*$

Proof of Predicate Clausal Form Theorem

The system **QRS*** is a version of the predicate system **QRS** with inference rules restricted to Propositional Rules

At this point we use the system **QRS*** to define in it a decomposition tree $\mathbf{T}_{A^*}$ for any **open** formula A^*

We form **clauses** out of its leaves and we put

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

This is the **clausal form** of the formula A of $\mathcal{L}$

To complete the proof we develop in the next section all needed **notions** and **results**

Prenex Normal Forms and Skolemization

Some Basic Notions

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

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

denote the result of replacing respectively all occurrences of the free variables x, $x_1, x_2, ..., x_n$, by the terms $t, t_1, t_2, ..., t_n$

We assume that $t, t_1, t_2, ..., t_n$ are **free for** x, $x_1, x_2, ..., x_n$, respectively, **in** A

The assumption that $t \in \mathbf{T}$ is **free for** x **in** $A(x)$ while substituting t for x, is **important** because otherwise we would distort the meaning of $A(t)$

Examples

Example 1

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

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

Example 2

Let $A(x)$ be a formula

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

and let $t = f(x, z)$

We **substitute** t on a place of x in $A(x)$ and we obtain a formula $A(t)$ of the form

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

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

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

Examples

Example 3

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

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

which contain a **bound** occurrence of the variable y of t in sub-formula $(\forall y P(f(y,z),y))$

The other occurrence of y in sub-formula $(Q(f(y,z),z))$ 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 $A(t)$

Similar Formulas

Informally, we say that formulas $A(x)$ 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** occurrence of of y

We define it formally as follows

Definition

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

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

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

Similar Formulas Examples

Example 1

The formulas

$$A(x): \ \exists z(P(x,z) \Rightarrow Q(x,y))$$

and

$$A(y): \ \exists z(P(y,z) \Rightarrow Q(y,y))$$

are **not similar**; y is **free for** x in $A(x)$ as **no occurrence** of y becomes a **bound** occurrence in the formula $A(y)$ but the formula $A(x)$ has a **free occurrence** of y

Similar Formulas Examples

Example 2

The formulas

$$A(x): \ \exists z(P(x,z) \Rightarrow Q(x,y))$$

and

$$A(w): \ \exists z(P(w,z) \Rightarrow Q(w,y))$$

are similar; w is **free** for x in $A(x)$ as **no occurrence** of w becomes a **bound** occurrence in the formula $A(w)$ and the formula $A(x)$ **has no free** occurrence of w

Renaming the Variables

Directly from the definition we get the following

Fact (Renaming the Variables)

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

if $A(x)$ and $A(y) = A(x/y)$ are similar, i.e.

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

then the following logical equivalences hold

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

and

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

Example

Example 3

We proved in **Example 2** that

$$\exists z(P(x,z) \Rightarrow Q(x,y)) \sim \exists z(P(w,z) \Rightarrow Q(w,y))$$

Hence by the **Fact** we get that

$$\forall x\exists z(P(x,z) \Rightarrow Q(x,y)) \;\equiv\; \forall w\exists z(P(w,z) \Rightarrow Q(w,y))$$

and

$$\exists x\exists z(P(x,z) \Rightarrow Q(x,y)) \;\equiv\; \exists w\exists z(P(w,z) \Rightarrow Q(w,y))$$

Replacement Theorem

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

Replacement Theorem

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

if B is a **sub-formula** of A, and A^* is the result of **replacing** zero or more occurrences of B in A by a formula C, and $B \equiv C$, then $A \equiv A^*$

Change of Bound VariablesTheorem

Theorem (Change of Bound Variables)

For any formula $A(x), A(y), B \in \mathcal{F}$,

if the formulas $A(x)$ and $A(x/y)$ are **similar**, i.e.

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

and the formula

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

is a **sub-formula** of B, and the formula B^* is the result of **replacing** zero or more occurrences of $A(x)$ in B by a formula $\forall y A(y)$ or by a formula $\exists y A(y)$, then

$$B \equiv B^*$$

Naming Variables Apart

Definition

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 now use the Change of Bound Variables **Theorem** to prove its more general version

Naming Variables Apart

Theorem (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 to prove, as a next step a so called a **Prenex Form Theorem**

In order to do so we first we define an important notion of prenex normal form of a formula

Closure of a Formula

Here is an important notion we need for future definition

Definition(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

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

Example

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

Prenex Normal Form

PNF Definition

Any formula 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. the following holds

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

$$Q_i \in \{\exists, \forall\} \quad \text{and} \quad x_i \neq x_j \quad \text{for} \quad i \neq j$$

and the formula 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

Prenex Normal Form Theorem

We assume that the formula A in **PNF** is always **closed**

If it is not closed we form its closure instead

PNF Theorem

There is an effective procedure for transforming any formula $A \in \mathcal{F}$ into a formula B in the prenex normal form **PNF** such that

$$A \equiv B$$

Proof

The procedure uses the Replacement and Naming Variables Apart **Theorems** and and the following Equational Laws of Quantifiers proved in chapter 2

Equational Laws of Quantifiers

For any $A(x), B \in \mathcal{F}$, where B **does not** contain any free occurrence of x the following holds

$$\forall x(A(x) \cup B) \equiv (\forall x A(x) \cup B)$$

$$\forall x(A(x) \cap B) \equiv (\forall x A(x) \cap B)$$

$$\exists x(A(x) \cup B) \equiv (\exists x A(x) \cup B)$$

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

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

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

$$\forall x(B \Rightarrow A(x)) \equiv (B \Rightarrow \forall x A(x))$$

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

PNF Procedure

The general **PNF procedure** is defined by induction on the number k of occurrences of connectives and quantifiers in A

We show here how it works in some particular cases

Exercise Find a prenex normal form **PNF** of a formula

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

Solution We find **PNF** as follows

Step 1: Naming Variables Apart

We make all **bound variables** in A different, i.e. we transform A into an equivalent formula A'

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

PNF Procedure

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

$$B: \quad (P(x) \Rightarrow \exists y Q(y))$$

of A' for $C = P(x)$, as P(x) **does not** contain the variable y

We get its equivalent formula

$$B^*: \quad \exists y (P(x) \Rightarrow Q(y))$$

We substitute B^* on place of B in A' and get the formula

$$A" \quad \forall x \exists y (P(x) \Rightarrow Q(y))$$

By the Replacement **Theorem** $A'' \equiv A' \equiv A$

The formula A'' is a required prenex normal form **PNF** for A

PNF Procedure

Example

Let's now find **PNF** for the formula A:

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

Step 1: Rename Variables Apart

Take a sub- formula $B(x,y)$: $\forall y \exists x\, R(x,y)$ of A

Rename variables in $B(x,y)$, i.e. get
$B(x/z, y/w)$: $\forall w \exists z\, R(z,w)$

Replace $B(x,y)$ by $B(x/z, y/w)$ in A and get

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

PNF Procedure

Step 2: Pull out quantifiers

We use corresponding equational laws for quantifiers to pull out **first** (one by one) quantifiers $\exists x \forall y$ and **then** pulling out one by one the quantifiers $\forall w \exists z$

We get the following **PNF** for A

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

Observe we can also perform **Step 2** by pulling out **first** (one by one) the quantifiers $\forall w \exists z$ and **then** pulling out one by one the quantifiers $\exists x \forall y$.

We hence can obtain **another PNF** for A

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

Skolem Procedure of Elimination of Quantifiers

Skolemization

We will show now how any formula A already in its prenex normal form **PNF** can be transformed into a certain **open formula** A^*, such that

$$A \equiv A^*$$

The **open formula** A^* belongs to a **richer language** then the initial language $\mathcal{L}$ to which the formula A belongs

Skolemization

This transformation process **adds** new constants to the original language $\mathcal{L}$

They are called **Skolem constants**

The process also **adds** to $\mathcal{L}$ new functions symbols called **Skolem functions**

The whole transformation process is called **Skolemization** of the initial language $\mathcal{L}$

Such build extension of the initial language $\mathcal{L}$ is called the **Skolem extension** of and $\mathcal{L}$ and denoted

$$\mathcal{L}^*$$

Skolem Elimination of Quantifiers

Skolem Procedure of Elimination of Quantifiers

Given a formula A of the language

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

We assume that A is already in its prenex normal form **PNF**

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

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 the formula $B(x_1, x_2, \ldots x_n)$ contains **no quantifiers**

Skolem Elimination of Quantifiers

We describe now a procedure of **elimination** of all quantifiers from a **PNF** formula A

The procedure transforms **PNF** formula A into a logically equivalent **open formula** A^*

We also assume that the **PNF** formula A is **closed**

If it is not closed we form its closure instead

Closure of a Formula

For any formula A, its **closure** is a formula A' obtained from A by **prefixing** in universal quantifiers all those variables that are **free** in A

Example

Let A be a formula

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

its **closure** i.e. a formula $A' \equiv A$ is

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

Elimination of Quantifiers

Given a formula A in its closed **PNF** form

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

We considerer 3 cases

Case 1

All quantifiers Q_i for $1 \leq i \leq n$ are **universal**, i.e. the formula A is

$$A: \quad \forall x_1 \forall x_2 \dots \forall x_n B(x_1, x_2, \dots, x_n)$$

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

$$A^*: \quad B(x_1, x_2, \dots., x_n)$$

Elimination of Quantifiers

Case 2

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

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

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

$$A^*: \quad B(c_1, c_2, \dots., c_n)$$

where $c_1, c_2, \dots., c_n$ and **new** individual constants **added** to our original language $\mathcal{L}$

We call such individual **constants** added to the original language Skolem constants

Elimination of Quantifiers

Case 3

The quantifiers in A are **mixed**

We **eliminate** the mixed quantifiers one by one and step by step depending on first, and then the consecutive quantifiers in the closed **PNF** formula A

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

We have two possibilities for the **first** quantifier $Q_1 x_1$

P1 $Q_1 x_1$ is **universal**

P2 $Q_1 x_1$ is **existential**

Elimination of Quantifiers; Step 1

Step 1 Elimination of Q_1

We consider the two cases for the **first** quantifier

Case **P1**

First quantifier Q_1 is **universal**

This means that A is

$$A: \quad \forall x_1 Q_2 x_2 \ldots Q_n x_n B(x_1, x_2, \ldots x_n)$$

We **replace** A by the following formula A_1

$$A_1: \quad Q_2 x_2 Q_3 x_3 \ldots Q_n x_n B(x_1, x_2, x_3, \ldots x_n)$$

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

Elimination of Quantifiers; Step 1

Case **P2**

First quantifier Q_1 is **existential**. This means that A is

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

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

$$A_1 \quad Q_2 x_2 \dots Q_n x_n B(b_1, x_2, \dots x_n)$$

where b_1 is a new constant symbol **added** to our original language $\mathcal{L}$

We call such constant symbol **added** to the language a Skolem constant

We have **eliminated** the quantifier Q_1 in both cases and this ends the **Step 1**

Elimination of Quantifiers; Step 2

Step 2 Elimination of Q_2

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

$$A_1 \qquad Q_2 x_2 \ldots Q_n x_n B(x_1, x_2, \ldots x_n)$$

Remark that the formula A_1 might **not be** closed

We have again two cases for elimination of the quantifier Q_2

P1 Q_2 is **universal**

P2 Q_2 is **existential**

Elimination of Quantifiers; Step 2

Case **P1**

First quantifier in A_1 is **universal**

The formula A_1 is

$$A_1 \quad \forall x_2 Q_3 x_3 \ldots Q_n x_n B(x_1, x_2, x_3, \ldots x_n)$$

We **replace** A_1 by the following A_2

$$A_2 \quad Q_3 x_3 \ldots Q_n x_n B(x_1, x_2, x_3, \ldots x_n)$$

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

Elimination of Quantifiers; Step 2

Case **P2**

First quantifier in A_1 is **existential**

The formula A_1 is

$$A_1 \quad \exists x_2 Q_3 x_3 \ldots Q_n x_n B(x_1, x_2, x_3, \ldots x_n)$$

Observe that now the variable x_1 is a **free** variable in

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

and hence x_1 is a **free** variable in in the formula A_1

Elimination of Quantifiers; Step 2

The variable x_1 is **free** in A_1

$$A_1 \quad \exists x_2 Q_3 x_3 \ldots Q_n x_n B(x_1, x_2, x_3, \ldots x_n)$$

We **replace** A_1 by the following A_2

$$A_2 \quad Q_3 x_3 \ldots Q_n x_n B(x_1, f(x_1), x_3, \ldots x_n)$$

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

Elimination of Quantifiers; Step 2

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

$$A_1 \quad Q_2 x_2 Q_3 x_3 \ldots Q_n x_n B(b_1, x_2, \ldots x_n)$$

Again we have two cases for the quantifier Q_2

Case **P1**

First quantifier Q_2 in A_1 is **universal**

The formula A_1 is

$$A_1 \quad \forall x_2 Q_3 x_3 \ldots Q_n x_n B(b_1, x_2, x_3, \ldots x_n)$$

We **replace** A_1 by the following A_2

$$A_2 \quad Q_3 x_3 \ldots Q_n x_n B(b_1, x_2, x_3, \ldots x_n)$$

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

Elimination of Quantifiers; Step 2

Case **P2**

First quantifier in A_1 is **existential**

The formula A_1 is

$$A_1 \quad \exists x_2 Q_3 x_3 \ldots Q_n x_n B(b_1, x_2, x_3, \ldots x_n)$$

We **replace** A_1 by the following A_2

$$A_2 \quad Q_3 x_3 \ldots Q_n x_n B(b_1, b_2, x_3, \ldots x_n)$$

where $b_2 \neq b_1$ is a **new** Skolem constant **added** to the 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**

Elimination of Quantifiers; Step 3

Step 3 Elimination of Q_3

Let's now consider, as an **example** a formula A_2 from **Step 2** - case **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 two cases but we describe only the following

P2 First quantifier in A_2 is **existential**

The formula A_2 is

$$A_2 \quad \exists x_2 Q_4 x_4 \dots Q_n x_n B(x_1, x_2, x_3, x_4, \dots x_n)$$

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

Elimination of Quantifiers; Step 2

The the variables x_1, x_2 are **free** in A_2

$$A_2 \qquad \exists x_2 Q_4 x_4 \ldots Q_n x_n B(x_1, x_2, x_3, x_4, \ldots x_n)$$

We replace A_2 by the following A_3

$$A_3 \qquad Q_4 x_3 \ldots Q_n x_n B(x_1, x_2, g(x_1, x_2), x_4 \ldots x_n)$$

where g is a **new** two argument functional symbol **added** to the original language $\mathcal{L}$

We have **eliminated** the quantifier Q_3 in this case

Elimination of Quantifiers

At each **Step i** for $1 \leq i \leq n$ we build a **binary tree** of cases

P1 Q_i is universal or **P2** Q_i is existential

The result in each case is a formula A_i with one less quantifier

The **elimination** of the proper quantifier **adds** new Skolem constant or Skolem function symbol to the original language $\mathcal{L}$

Elimination of Quantifiers

The **elimination of quantifiers** process builds a sequence of formulas

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

where the formula A belongs to our original language

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

and the **open** formula A^* belongs to its Skolem extension defined as follows

Skolem Extension

Definition

The **Skolem extension** $\mathcal{L}^*$ of a language

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

is the language

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

where the sets $\mathbf{SF}$ and $\mathbf{SC}$ are respectively the sets of Skolem functions and Skolem constants

They are obtained by the **quantifiers elimination procedure**

Elimination of Quantifiers Result

Given a formula A in its closed **PNF** form

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

Observe that the **elimination** of an universal quantifier Q_i introduces a **free** variable x_i in the formula

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

Elimination of Quantifiers Result

The **elimination** of an existential quantifier Q_i that follows universal quantifiers introduces a **new** functional symbol with number of arguments equal the number of universal quantifiers preceding it

The **elimination** of an existential quantifier Q_i that **does not** follows any universal quantifiers introduces a **new** constant symbol

The resulting **open** formula A^* is logically equivalent to the **PNF** formula A

Skolemization

Definition

Given a formula A of $\mathcal{L}$

A formula

$$A^*$$

of the Skolem extension language $\mathcal{L}^*$ obtained from A

by the **elimination of quantifiers** process is called a

Skolem form of the formula A

The elimination of quantifiers process obtaining it is called **Skolemization**

Example

Example 1

Let A be a closed **PNF** formula

$$A: \quad \forall y_1 \exists y_2 \forall y_3 \exists y_4\, B(y_1, y_2, y_3, y_4, y_4)$$

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

$$A_1: \quad \exists y_2 \forall y_3 \exists y_4\, B(y_1, y_2, y_3, y_4)$$

We **eliminate** $\exists y_2$ by **replacing** the variable y_2 by $h(y_1)$

The symbol h is a **new** one argument functional symbol **added** to the language $\mathcal{L}$

We get a formula A_2

$$A_2: \quad \forall y_3 \exists y_4\, B(y_1, h(y_1), y_3, y_4)$$

Example 1

Given the formula A_2

$$A_2: \quad \forall y_3 \exists y_4\, B(y_1, h(y_1), y_3, y_4)$$

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

$$A_3: \quad \exists y_4\, B(y_1, h(y_1), y_3, y_4)$$

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 $\mathcal{L}$

We get a formula A_4 that is our resulting **open** formula A^*

$$A^*: \quad B(y_1, h(y_1), y_3, f(y_1, y_3))$$

Example 2

Let A be a closed **PNF** formula

$$A: \quad \exists y_1 \forall y_2 \forall y_3 \exists y_4 \exists y_5 \forall y_6 \, B(y_1, y_2, y_3, y_4, y_4, y_5, y_6)$$

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

$$A_1: \quad \forall y_2 \forall y_3 \exists y_4 \exists y_5 \forall y_6 \, B(b_1, y_2, y_3, y_4, y_4, y_5, y_6)$$

where b_1 is a **new** constant **added** to the language $\mathcal{L}$

We **eliminate** $\forall y_2, \forall y_3$ and get formulas A_2, A_3

$$A_2: \quad \forall y_3 \exists y_4 \exists y_5 \forall y_6 \, B(b_1, y_2, y_3, y_4, y_4, y_5, y_6)$$

$$A_3: \quad \exists y_4 \exists y_5 \forall y_6 \, B(b_1, y_2, y_3, y_4, y_4, y_5, y_6)$$

Example 2

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

$$A_4 : \quad \exists y_5 \forall y_6\, B(b_1, y_2, y_3, g(y_2, y_3), y_5, y_6)$$

where g is a **new** two argument functional symbol **added** to the original language $\mathcal{L}$

We **eliminate** $\exists y_5$ and get a formula A_5

$$A_5 : \quad \forall y_6\, B(b_1, y_2, y_3, g(y_2, y_3), h(y_2, y_3), y_6)$$

where h is a **new** two argument functional symbol **added** to the language $\mathcal{L}$

We **eliminate** $\forall y_6$ and get a formula A_6 that is the resulting **open** formula A^*

$$A^* : \quad B(b_1, y_2, y_3, g(y_2, y_3), h(y_2, y_3), y_6)$$

Skolem Theorem

The **correctness** of the Skolemization process is established by the **Skolem Theorem**

It states informally that the formula A^* obtained from a formula A via the Skolemization process is **satisfiable** if and only if the original formula A is **satisfiable**

We define this notion formally as follows

Skolem Theorem

Definition Equisatisfiable formulas

Given any formulas A of $\mathcal{L}$ and B of the **Skolem extension** $\mathcal{L}^*$ 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

$$\text{If } \mathcal{M} \models A, \quad \text{then } \mathcal{M}^* \models B$$

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

$$\text{If } \mathcal{M}^* \models B, \quad \text{then } \mathcal{M} \models A$$

Skolem Theorem

Skolem Theorem

Let $\mathcal{L}^*$ be the **Skolem extension** of a language $\mathcal{L}$

Any formula A of $\mathcal{L}$ and its **Skolem form** A^* of $\mathcal{L}^*$ are **equisatisfiable**

Clausal Form of Formulas

Poof System **QRS***

Let $\mathcal{L}^*$ be the **Skolem extension** of $\mathcal{L}$

By definition, the language $\mathcal{L}^*$ does not contain quantifiers and all its formulas and **open**

We define a proof system **QRS*** as an **open formulas** version of the proof system **QRS** based on the language $\mathcal{L}$

We denote the set of **formulas** of $\mathcal{L}^*$ by $O\mathcal{F}$ to stress the fact that all its formulas are **open**

Let

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

be the set of all **atomic** formulas of $\mathcal{L}^*$ and the set

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

the set of all **literals** of $\mathcal{L}^*$

Poof System **QRS***

We denote by

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

finite sequences (empty included) formed out of **literals**, i.e of the elements of LT^*

We will denote by

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

finite sequences (empty included) formed out of **formulas**, i.e of the elements of $\mathcal{OF}^*$

Poof System **QRS***

We define the proof system **QRS*** formally as follows

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

where $\mathcal{E} = \{\Gamma : \ \Gamma \in \mathcal{OF}^*\}$

The set LA of logical axioms contains any sequence $\Gamma' \in LT^*$

which contains an atomic formula and its negation

$\mathcal{R}$ is the set inference rules

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

defined as follows

Poof System **QRS***

Disjunction rules

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

Conjunction rules

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

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

Poof System **QRS***

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

$\mathbf{QRS}^*$ Semantics

Definition

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

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

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 as follows

Clauses Semantics

Definition

For any finite set of clauses $\mathbf{C}$ of $\mathcal{L}^*$, any structure $\mathcal{M} = [M, I]$ for $\mathcal{L}^*$, and any clause $C \in \mathbf{C}$,

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

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

3. $(A \equiv \mathbf{C})$ if and only if $A \equiv \sigma_{\mathbf{C}}$

where δ_C denotes a disjunction of all literals in C and $\sigma_{\mathbf{C}}$ is a conjunction of all formulas δ_C for all clauses $C \in \mathbf{C}$

Obviously, the rules of inference of $\mathbf{QRS}^*$ are strongly sound and the following holds

Strong Soundness Theorem

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

Formula to Clauses Transformation

We use the **QRS*** system to define an effective procedure that **transforms** any formula A of $\mathcal{L}^*$ into set of clauses and prove correctness of this transformation

We treat the rules of inference of **QRS*** as decomposition rules and use them to **generate** needed set $\mathbf{C}_A$ of **clauses** corresponding to a given formula A

Decomposable, Indecomposable

Definition

A formula that is not a literal, i.e. any formula $A \in \mathcal{OF} - \mathbf{L}$

is called a **decomposable**

Otherwise A is called **indecomposable**

Definition

A sequence Γ that contains a decomposable formula is called a **decomposable** sequence

Definition

A sequence Γ' built only out of literals, i.e. $\Gamma' \in \mathbf{L}^*$

is called an **indecomposable** sequence

Decomposition Tree $\mathbf{T}_A$

Definition

Given a formula $A \in \mathcal{OF}$

We build the **decomposition tree** $\mathbf{T}_A$ of A as follows

Step 1.

The formula A is the **root** of $\mathbf{T}_A$

For any node Δ of the tree $\mathbf{T}_A$ we follow the steps bellow

Step 2.

If Δ is **indecomposable**, then Δ becomes a **leaf** of the tree

Decomposition Tree $\mathbf{T}_A$

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 premiss 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**

Formula-Clauses Equivalency Theorem

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

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

Proof

Given $A \in \mathcal{OF}$. 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 C_L determined by L and we put

$$\mathbf{C}_A = \{C_L : \ L \text{ is a leaf of } \ \mathbf{T}_A\}$$

Directly from the **QRS*** **Strong Soundness Theorem** and the semantics for clauses definition we get that

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

Exercise

Exercise

Find the 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

Step **S1.** We construct the decomposition tree $\mathbf{T}_A$ for A

Step **S2.** We form **clauses** out of the leaves of the tree $\mathbf{T}_A$

We put

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

Exercise

Step **S1.** The decomposition tree is

$$\mathbf{T}_A$$

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

$$\mid (\cup)$$

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

$$\mid (\cup)$$

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

$$\mid (\Rightarrow)$$

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

$$\bigwedge (\cap)$$

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

$$L_1 \qquad\qquad L_2$$

Exercise

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

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

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

The corresponding clauses are

$$C_1 = \{\neg P(b, f(x)), Q(x), \neg R(z), P(b, f(x))\}$$

$$C_2 = \{\neg P(b, f(x)), Q(x), \neg R(z), R(z)\}$$

The set of clauses is

$$\mathbf{C}_A = \{\ C_1,\ \ C_2\ \}$$

Clausal Form of Formulas of $\mathcal{L}$

Definition

Given a formula A of the original language $\mathcal{L}$

Let A^* of $\mathcal{L}^*$ be the **Skolem form** A obtained by the Skolemization process

A a set $\mathbf{C}_{A^*}$ of clauses of $\mathcal{L}^*$ such that

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

is called a **clausal form** of the formula A of the language $\mathcal{L}$

Exercise

Exercise Find the clausal form of a formula A

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

Solution We first find the Skolem form A^* of A

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

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

Step 2: We use **Equational Laws** of Quantifiers to pull out quantifiers $\exists x$ and $\forall y$ and get a formula A''

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

Exercise

Step 3 : We use **Equational Laws** of Quantifiers to pull out the quantifiers $\exists z$ 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'''

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

This is the Prenex Normal Form **PNF** of A

Exercise

Step 4: We perform the Skolemization Procedure

Observe that the formula

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

is of the form of the formulas of the **Examples 1, 2**

We follow them and eliminate $\forall x$ and get a formula A_1

$$A_1 : \quad \exists y \forall z \exists w \, ((R(x,y) \cup \neg P(x)) \Rightarrow \neg R(z,w))$$

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

We get a formula A_2

$$A_2 : \quad \forall z \exists w \, ((R(x,h(x)) \cup \neg P(x)) \Rightarrow \neg R(z,w))$$

Exercise

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

$$A_3 : \quad \exists w\, ((R(x, h(x)) \cup \neg P(x)) \Rightarrow \neg R(z, w))$$

We eliminate $\exists w$ by replacing w by $f(x, z)$, where f is a **new** two argument functional symbol **added** to the original language $\mathcal{L}$

We get a formula A_4 that is the resulting **open** formula A^* of $\mathcal{L}^*$

$$A^* : \quad ((R(x, h(x)) \cup \neg P(x)) \Rightarrow \neg R(z, (x, z)))$$

Exercise

Step 5: We build the decomposition tree of A^* as follows

$$\mathbf{T}_{A^*}$$

$$((R(x,h(x)) \cup \neg P(x)) \Rightarrow \neg R(z,f(x,z)))$$

$$\mid (\Rightarrow)$$

$$\neg(R(x,h(x)) \cup \neg P(x)), \ \neg R(z,f(x,z))$$

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

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

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

$$P(x), \ \neg R(z,f(x,z))$$

Exercise

Step 6: The leaves of $\mathbf{T}_{A^*}$ are

$L_1 = \neg R(x, h(x)), \neg R(z, f(x, z)$

$L_2 = P(x),\ \neg R(z, f(x, z))$

The corresponding clauses are

$C_1 = \{\neg R(x, h(x)), \neg R(z, f(x, z)\}$

$C_2 = \{P(x),\ \neg R(z, f(x, z))\}$

Step 7: The **clausal form** of the formula A

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

is the **set of clauses**

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