cse357
ARTIFICIAL INTELLIGENCE

Professor Anita Wasilewska

Spring2016

Introduction to Predicate Resolution

PART 1: Introduction

PART 2: Prenex Normal Form and Skolemizatiom

PART 3: Clauses and Unification

PART 4: Resolution

PART 1: Introduction

Introduction

The resolution proof system for Predicate Logic operates, as in propositional case on sets of **clauses** 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 $\mathbf{C}_A$

Introduction

The **second goal** of this part is extend the definition of the propositional **resolution rule** to the case of predicate languages

Observe that we define, as in propositional case, a **clause** as a finite set of **literals**

We define, as before, a **literal** as an **atomic formula** or a **negation** of an **atomic** formula

The **difference** with propositional resolution is in the **language** we work with, i.e. what is a predicate atomic formula as opposed to propositional atomic formula

Definition (Reminder)

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 defined as

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

Introduction

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

We define formally the set $\mathbf{L}$ of all **literals** of $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$ as follows

$$\mathbf{L} = \{R : \ R \in A\mathcal{F}\} \cup \{\neg R : \ R \in A\mathcal{F}\}$$

Reminder:

A formula of a predicate language is an **open formula** if it does not contain any quantifiers, i.e. it is a formula build out of atomic formulas and propositional connectives only

Introduction

A transformation a formula A of a predicate language into a logically equivalent set $\mathbf{C}_A$ of clauses means that we can **represent** the formula A as a certain collection of **atomic formulas** and **negations of atomic formula**

This means any formula A of a predicate language can be that **represented** as a certain collection of **open formulas**

In order to achieve this goal we **start** with of methods that allow the transformation of any formula A into an **open formula** A^* of some **larger language** such that $A \equiv A^*$

The process is described in the following PART 2

PART 2: Prenex Normal Form and Skolemizatiom

Some Basic Notions

Let $\mathcal{L} = (\mathcal{A}, \mathbf{T}, \mathcal{F})$ be a predicate language **determined** by **P, F, C** and the set of propositional connectives $\{\neg, \cup, \cap, \Rightarrow\}$, i.e.

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

Given a formula $A(x) \in \mathcal{F}$, $t \in \mathbf{T}$, and $A(t)$ be a result of **substituting** the term t for all free occurrences of x in $A(x)$

Definition

We say that a term $t \in \mathbf{T}$ is **free for** x **in** $A(x)$, if no occurrence of a variable in t becomes a bound occurrence in the formula $A(t)$

Some Basic Notions

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

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

denotes 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 $A(t)$

This is illustrated by the following example

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 $\exists y(x \neq y)$

Examples

Example 2

Let $A(x)$ be a formula

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

and $t = f(x, z)$

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

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

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

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

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

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

$$\forall x A(x) \equiv \forall y A(y) \quad \text{and} \quad \exists x A(x) \equiv \exists y A(y)$$

Example 3

We proved in **Example 2** that the formulas $A(x) \sim A(w)$, i.e.

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

Hence by the **Fact** we get that

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

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

Renaming the Variables

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 $A \equiv A^*$

Renaming Variables Theorem

Theorem Renaming 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 the formula $\exists x A(x)$ 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 by a formula $\exists y A(y)$, then

$$B \equiv B^*$$

Naming Variables Apart

Definition 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

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

Prenex Normal Form

Definition Prenex Normal Form

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. for all $1 \leq i \leq n$, $Q_i \in \{\exists, \forall\}$,

and $x_i \neq x_j$ for $i \neq j$,

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

Theorem Prenex Normal Form Theorem

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

$$A \equiv B$$

We describe the procedure by induction on the number k of occurrences of connectives and quantifiers in A

Let's consider now few examples

Prenex Normal Form Example 1

Example 1

Given a formula A: $\forall x(P(x) \Rightarrow \exists x Q(x))$

Find its prenex normal form **PNF**

Step 1: Naming Variables Apart

We make all bound variables in A different, by **substituting** $\exists x Q(x)$ by logically equivalent formula $\exists y Q(y)$ in A as $Q(x)$ and $Q(x/y)$ **are similar**

We hence transformed A into an equivalent formula A'

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

with all its variables named apart

Prenex Normal Form Example 1

Step 2: Pull Out Quantifiers

Now, we can 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''

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

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

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

Prenex Normal Form Example 2

Example 1

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
$B(x/z, y/w) : \forall z \exists w\, R(z,w)$

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

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

Prenex Normal Form Example 2

Step 2: Pull out quantifiers

We use corresponding equational laws for quantifiers to pull out **first** quantifiers $\forall x \exists y$ and then quantifiers $\forall z \exists w$

and get the following **PNF** for A

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

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

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

Skolemization

As the next step we show how any formula A 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 language of the initial formula A

The transformation process **adds** new constants, called **Skolem constants** and new function symbols, called **Skolem function symbols** to the initial language to which the formula A belongs

The whole process is called the **Skolemization** of the initial language

Such build extension of the initial language is called the **Skolem extension**

Elimination of Quantifiers

Given a formula A be in its **Prenex Normal Forma 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 $B(x_1, x_2, \ldots x_n)$ **contains no quantifiers**

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

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

We assume that A is **closed**

If it is not closed we form its **closure** instead

Definition of **closure** follows

Closure of a Formula

For any formula $A \in \mathcal{F}$, a **closure** of A is a **closed** formula A' obtained from A by **prefixing** in universal quantifiers all those variables that a **free** in A; i.e. the following holds

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** i.e. $A' \equiv A$ 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

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

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

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

Elimination of Quantifiers

Case 2

All quantifiers Q_i for $1 \leq i \leq n$ are **existential**, i.e. 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 **added** to our original language $\mathcal{L}$

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

Elimination of Quantifiers; Step 1

Case 3

The quantifiers in A are **mixed**

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

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

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

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

Step 1

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

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

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

Elimination of Quantifiers; Step 1

Consider **P2**

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

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

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

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

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**

Elimination of Quantifiers; Step 2

Step 2 Elimination of $Q_2 x_2$

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

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

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

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

We **replace** A_1 by the following A_2

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

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

Elimination of Quantifiers; Step 2

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

Elimination of Quantifiers; Step 2

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

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

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

Elimination of Quantifiers; Step 2

Consider **P2**

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

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

We replace A_1 by the following A_2

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

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 i n this case

We have covered all cases and this **ends** the **Step 2**

Elimination of Quantifiers; Step 3

Step 3 Elimination of $Q_3 x_3$
Let's now consider, as an **example** a 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

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

where the formula A belongs to our original language

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

the **open** formula A^* belongs to its Skolem extension

The **Skolem extension** $S\mathcal{L}$ is obtained from $\mathcal{L}$ in the quantifiers elimination process and

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

Elimination of Quantifiers Result

Observe that in the elimination process an 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 **elimination** of an existential quantifier that **does not** follows any universal quantifiers introduces a **new** constant symbol

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

Example 1

Example 1

Let A be a closed **PNF** formula

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

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 $h(y_1)$

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

We get a formula A_2

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

Example 1

Given the 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^*

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

Example 2

Let now A be a **PNF** formula

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

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 formulas A_2, A_3; here is the formula A_3

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

Example 2

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^*

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

Open Formulas to Clauses

Definition (Reminder)

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 defined as

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

Literals and Open Formulas

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

We define formally the set $\mathbf{L}$ of all **literals** of $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$ as follows

$$\mathbf{L} = \{R : \ R \in A\mathcal{F}\} \cup \{\neg R : \ R \in A\mathcal{F}\}$$

Definition

A set $O\mathcal{F}$ of all **open formulas** of a predicate language $\mathcal{L} = \mathcal{L}_{\{\neg, \cup, \cap, \Rightarrow\}}(\mathbf{P}, \mathbf{F}, \mathbf{C})$ is the **smallest** set for which the following conditions are satisfied

(1) $A\mathcal{F} \subseteq O\mathcal{F}$ (atomic formulas are open formulas)

(2) If $A \in O\mathcal{F}$, then $\neg A \in O\mathcal{F}$

(3) If $A, B \in O\mathcal{F}$, then $(A \cup B), (A \cap B), (A \Rightarrow B) \in O\mathcal{F}$

Decomposition Rules for Open Formulas

Here are the **decomposition rules** needed to transform open formulas into logically equivalent sets of clauses

Disjunction decomposition rules

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

Conjunction decomposition rules

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

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

Decomposition Rules

Implication decomposition rules

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

Negation decomposition rule

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

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

Tree Decomposition Rules

We write the decomposition rules in a **visual tree form** as follows

Tree Decomposition Rules

$(\cup)$ **rule**

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

$$\mid (\cup)$$

$$\Gamma^{'},\ A, B,\ \Delta$$

Tree Decomposition Rules

$(\neg\cup)$ rule

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

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

$$\Gamma^{'},\ \neg A,\ \Delta \qquad\qquad \Gamma^{'},\ \neg B,\ \Delta$$

$(\cap)$ rule

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

$$\bigwedge(\cap)$$

$$\Gamma^{'},\ A,\ \Delta \qquad\qquad \Gamma^{'},\ B,\ \Delta$$

Tree Decomposition Rules

$(\neg\cup)$ **rule**

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

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

$$\Gamma^{'},\ \neg A, \neg B,\ \Delta$$

$(\Rightarrow)$ **rule**

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

$$\mid (\cup)$$

$$\Gamma^{'},\ \neg A, B,\ \Delta$$

Tree Decomposition Rules

$(\neg \Rightarrow)$ rule

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

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

$$\Gamma^{'},\ A,\ \Delta \qquad\qquad \Gamma^{'},\ \neg B,\ \Delta$$

$(\neg\neg)$ rule

$$\Gamma^{'},\ \neg\neg A,\ \Delta$$

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

$$\Gamma^{'},\ A,\ \Delta$$

Decomposable, Indecomposable

Definition: Decomposable Formula

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

Definition: Decomposable Sequence

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

Definition: Indecomposable Sequence

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

Definitions and Observations

Observation 1

Decomposition rules are functions with disjoint domains, i.e.

For any **decomposable** sequence, i.e. for any $\Gamma \notin \mathbf{L}^*$

there is **exactly one** decomposition rule that can be applied to it

This rule is **determined** by the first decomposable formula in Γ and by the main connective of that formula

Observation 2

If the main connective of the **first** decomposable formula is $\cup, \cap, \Rightarrow$,

then the **decomposition rule** determined by it is $(\cup), (\cap), (\Rightarrow)$, respectively

Definitions and Observations

Observation 3

If the main connective of the **first** decomposable formula A is negation $\neg$

then the **decomposition rule** is determined by the **second connective** of the formula A

The corresponding **decomposition rules** are $(\neg\cup), (\neg\cap), (\neg\neg),\ (\neg \Rightarrow)$

Observation 4

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

$\Gamma \in \mathbf{L}^*$ or Γ is in the domain of **exactly one** of Decomposition Rules

Decomposition Tree Definition

Definition: **Decomposition Tree** $\mathbf{T}_A$

For each $A \in \mathcal{OF}$, a **decomposition tree** $\mathbf{T}_A$ is a tree build as follows

Step 1.

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

For any other **node** Γ of the tree we follow the steps below

Step 2.

If Γ is **indecomposable** then Γ becomes a **leaf** of the tree

Decomposition Tree Definition

Step 3.

If Γ is **decomposable**, then we **traverse** Γ from **left** to **right** and identify the first **decomposable formula** B

We put its premiss as a node below, or its left and right premisses as the left and right nodes below, respectively

Step 4.

We repeat steps 2 and 3 until we obtain only leaves

Decomposition Tree and Clauses

Directly from **Observations 1 - 4** and the fact that premisses and conclusion in all decomposition rules are logically equivalent we get the following

Theorem

For any $A \in \mathcal{OF}$, its decomposition tree $\mathbf{T}_A$ is unique and its leaves form a set of clauses that is logically equivalent to the formula A

More precisely, let $\Gamma_1, \Gamma_2, \ldots \Gamma_n \in \mathbf{L}^*$ be all leaves of $\mathbf{T}_A$

The set of all clauses corresponding to the formula A is

$$\mathbf{C}_A = \{\{\Gamma_1\},\ \{\Gamma_2\},\ \ldots\ \{\Gamma_n\}\}$$

and

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

Example

The tree $\mathbf{T}_A$

$$(((P(x) \Rightarrow Q(y)) \cap \neg R(x)) \cup (P(x) \Rightarrow R(x))$$

$$\mid (\cup)$$

$$((P(x) \Rightarrow Q(y)) \cap \neg R(x)), (P(x) \Rightarrow R(x))$$

$$\bigwedge (\cap)$$

$$(P(x) \Rightarrow Q(y)), (P(x) \Rightarrow R(x)) \qquad \neg R(x), (P(x) \Rightarrow R(x))$$

$$\mid (\Rightarrow) \qquad \mid (\Rightarrow)$$

$$\neg P(x), Q(y), (P(x) \Rightarrow R(x)) \qquad \neg R(x), \neg P(x), R(x)$$

$$\mid (\Rightarrow)$$

$$\neg P(x), Q(y), \neg P(x), R(x)$$

Example

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

$$\neg P(x), Q(y), \neg P(x), R(x) \quad \text{and} \quad \neg R(x), \neg P(x), R(x)$$

The clauses corresponding to the leaves are

$$C_1 = \{\neg P(x), Q(y), R(x)\} \quad \text{and} \quad C_2 = \{\neg R(x), \neg P(x), R(x)\}$$

The set of all clauses corresponding to the formula A is

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

$$\mathbf{C}_A = \{\{\neg P(x), Q(y), R(x)\},\ \{\neg R(x), \neg P(x), R(x)\}\}$$

Unification

Unification is the process of determining whether **two** atomic formulas, i.e. **two** positive literals can be made identical by appropriate **substitution** for their variables

Unification is an essential part of resolution

Unification is defined in terms of a notion of a **substitution**

Sunstitution

Intuitively, a **substitution** is is a set of associations between **variables** and **terms** in which

1.each variable is associated with **at most one** term, and

2. no **variable** with an associated **term** occurs within any of the associated **terms**

Substitution

Example

The following is a well defined **substitution**

$$\{x/c,\ y/f(b),\ z/w\}$$

and the following is **not a substitution**

$$\{x/g(y),\ y/f(x)\}$$

as the variable x which is **associated** with term $g(y)$, occurs in the term $f(x)$ **associated** withy;

the variable y occurs in term $g(y)$ **associated** with variable x

Unification

Given two positive literals $P_1 = P(x, y, z)$ and $P_2 = P(c, f(b), w)$

The substitution $\{x/c,\ y/f(b),\ z/w\}$ **unifies** P_1 and P_2, as when **applied** to P_1 produces P_2, i.e.

$$P_1\{x/c,\ y/f(b),\ z/w\} = P(x, y, z)\{x/c,\ y/f(b),\ z/w\}$$

$$= P(c, f(b), w) = P_2$$