cse352
Artificial Intelligence

Professor Anita Wasilewska

AI LOGIC LECTURE

Logic Chapter:
Introduction to Classical Logic Languages and Semantics

Logic Chapter

Introduction to Classical Logic Languages and Semantics

Logic Chapter
Introduction to Classical Logic Languages and Semantics

Part 5: Predicate Language

Predicate Language

We define a predicate language $\mathcal{L}$ following the pattern established by the definitions of symbolic and propositional language.

The predicate language **is much more complicated** in its structure.

Its alphabet $\mathcal{A}$ is **much richer**.

The definition of its set of formulas $\mathcal{F}$ is **more complicated**.

In order to define the set $\mathcal{F}$ define an **additional set T**, called a set of all **terms** of the predicate language $\mathcal{L}$.

We single out this set **T** of **terms** not only because we need it for the definition of formulas, but also because of its role in the development of other notions of **predicate logic**.

Predicate Language Definition

Definition

By a **predicate language** $\mathcal{L}$ we understand a triple

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

where $\mathcal{A}$ is a predicate **alphabet**

$\mathbf{T}$ is the set of **terms**, and $\mathcal{F}$ is a set of **formulas**

Alphabet Components

Alphabet $\mathcal{A}$

The components of $\mathcal{A}$ are as follows

1. Propositional connectives

$$\neg, \; \cap, \; \cup, \; \Rightarrow, \; \Leftrightarrow$$

2. Quantifiers $\forall$, $\exists$

$\forall$ is the universal quantifier, and $\exists$ is the existential quantifier

3. Parenthesis (and)

Alphabet Components

4. Variables

We assume that we have, as we did in the propositional case a countably infinite set VAR of variables

The variables now have a different meaning than they had in the propositional case

We hence call them variables, or individual variables

We put

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

5. Constants

The constants represent in "real life" concrete elements of sets. We assume that we have a countably. infinite set C of constants

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

Alphabet Components

6. Predicate symbols

The predicate symbols represent ”real life” relations

We denote them by P, Q, R, ..., with indices, if necessary

We use symbol **P** for the set of all predicate symbols

We assume that **P** is countably infinite and write

$$\mathbf{P} = \{P_1, P_2, P_3, ...\}$$

Alphabet Components

Logic notation

In "real life" we write symbolically $x < y$ to express that element x is smaller then element y according to the two argument order relation $<$.

In the predicate language $\mathcal{L}$ we represent the relation $<$ as a two argument predicate $P \in \mathbf{P}$.

We write $P(x, y)$ as a representation of "real life" $x < y$.

The variables x, y in $P(x, y)$ are individual variables from the set VAR.

Mathematical statements $n < 0,\ 1 < 2,\ 0 < m$ are represented in $\mathcal{L}$ by $P(x, c_1),\ P(c_2, c_3),\ P(c_1, y)$, respectively,

where c_1, c_2, c_3 are any constants and x, y any variables.

Alphabet Components

7. Function symbols

The function symbols represent ”real life” functions

We denote function symbols by $f, g, h, \ldots$, with indices, if necessary

We use symbol **F** for the set of all function symbols

We assume that **F** is countably infinite and write

$$\mathbf{F} = \{f_1, f_2, f_3, \ldots\ldots\}$$

Set **T** of Terms

Definition

Terms are expressions built out of function symbols and variables.

They describe how we build compositions of functions.

We define the set **T** of all terms recursively as follows.

1. All variables are terms;

2. All constants are terms;

3. For any function symbol $f \in \mathbf{F}$ representing a function on n variables, and any terms $t_1, t_2, ..., t_n$, the expression $f(t_1, t_2, ..., t_n)$ is a term;

4. The set **T** of all terms of the predicate language $\mathcal{L}$ is the smallest set that fulfills the conditions **1. - 3.**

Example

Example

Here are some terms of $\mathcal{L}$

$$h(c_1),\ f(g(c,x)),\ g(f(f(c)),g(x,y)),$$

$$f_1(c,g(x,f(c))),\ g(g(x,y),g(x,h(c)))\ ....$$

Observe that to obtain the predicate language representation of for example $x+y$ we can first write it as $+(x,y)$ and then replace the addition symbol $+$ by any two argument function symbol $g \in \mathbf{F}$ and get the **term** $g(x,y)$.

Set $\mathcal{F}$ of Formulas

Formulas are build out of elements of the **alphabet** $\mathcal{A}$ and the set **T** of all **terms**.

We denote the formulas by $A, B, C, ...$, with indices, if necessary.

We build them, as before in recursive steps.

The first recursive step says:

all atomic formulas are formulas.

The atomic formulas are the simplest formulas, as the propositional variables were in the case of the propositional language.

We define the atomic formulas as follows.

Atomic Formulas

Definition

An **atomic formula** is any expression of the form

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

where R is any n-argument predicate $R \in \mathbf{P}$ and $t_1, t_2, ..., t_n$ are terms, i.e. $t_1, t_2, ..., t_n \in \mathbf{T}$.

Some atomic formulas of $\mathcal{L}$ are:

$$Q(c),\ Q(x),\ Q(g(x_1, x_2)),$$

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

Set $\mathcal{F}$ of Formulas

Definition

The set $\mathcal{F}$ of formulas of predicate language $\mathcal{L}$ is the smallest set meeting the following conditions.

1. All **atomic formulas** are formulas;

2. If A, B are formulas, then $\neg A, (A \cap B), (A \cup B), (A \Rightarrow B), (A \Leftrightarrow B)$ are formulas;

3. If A is a formula, then $\forall x A$, $\exists x A$ are formulas for any variable $x \in VAR$.

Set $\mathcal{F}$ of Formulas

Example

Some formulas of $\mathcal{L}$ are:

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

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

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

Set $\mathcal{F}$ of Formulas

Let's look now closer at the following formulas.

$$R(c_1, c_2), \quad R(x, y), \quad ((R(y, d) \Rightarrow R(a, z)),$$

$$\exists x R(x, y), \quad \forall y R(x, y), \quad \exists x \forall y R(x, y).$$

Observations

1. Some formulas are without quantifiers:

$$R(c_1, c_2), \quad R(x, y), \quad (R(y, d) \Rightarrow R(a, z)).$$

A formula without quantifiers is called an **open formula**

Variables x, y in $R(x, y)$ are called **free variables**.

The variable y in R(y, d) and z in R(a,z) are also **free**.

Set $\mathcal{F}$ of Formulas

Observations

2. Quantifiers **bind variables** within formulas.

The variable x is bounded by $\exists x$ in the formula $\exists x R(x, y)$, the variable y is free.

The variable y is bounded by $\forall y$ in the formula $\forall y R(x, y)$, the variable y is free.

3. The formula $\exists x \forall y R(x, y)$ **does not** contain any free variables, **neither does** the formula $R(c_1, c_2)$.

4. A formula **without** any free variables is called a **closed formula** or a **sentence**.

Mathematical Statements

We often use logic symbols, while writing mathematical statements in a more symbolic way.
For example, mathematicians to say "all natural numbers are greater then zero and some integers are equal 1" often write

$$x \geq 0, \quad \forall_{x \in N} \text{ and } \exists_{y \in Z}, \quad y = 1.$$

Some of them who are more "logic oriented" would write it as

$$\forall_{x \in N}\, x \geq 0 \ \cap \ \exists_{y \in Z}\, y = 1,$$

or even as

$$\forall_{x \in N}\, x \geq 0 \ \cap \ \exists_{y \in Z}\, y = 1.$$

Observe that none of the above symbolic statement are formulas of the predicate language.
These are mathematical statements written with mathematical and logic symbols. They are written with different degree of "logical precision", the last being, from a logician point of view **the most precise**.

Mathematical Statements

Our goal now is to ”translate ” mathematical and natural language statement into correct formulas of the predicate language $\mathcal{L}$.

Let’s start with some **observations**.

O1 The quantifiers in $\forall_{x \in N}, \ \exists_{y \in Z}$ **are not** the one used in logic.

O2 The predicate language $\mathcal{L}$ **admits only** quantifiers $\forall x, \ \exists y$, for any variables $x, y \in VAR$.

O3 The quantifiers $\forall_{x \in N}, \ \exists_{y \in Z}$ are called **quantifiers with restricted domain**.

The **restriction** of the quantifier domain can, and often is given by more complicated statements.

Quantifiers with Restricted Domain

The quantifiers $\forall_{A(x)}$ and $\exists_{A(x)}$ are called quantifiers with restricted domain, or restricted quantifiers, where $A(x) \in \mathcal{F}$ is any formula with a free variable $x \in VAR$.

Definition

$\forall_{A(x)} B(x)$ stands for a formula $\forall x(A(x) \Rightarrow B(x)) \in \mathcal{F}$.

$\exists_{A(x)} B(x)$ stands for a formula $\exists x(A(x) \cap B(x)) \in \mathcal{F}$.

We write it as the following **transformations rules** for restricted quantifiers

$$\forall_{A(x)}\, B(x) \;\equiv\; \forall x(A(x) \Rightarrow B(x))$$

$$\exists_{A(x)}\, B(x) \;\equiv\; \exists x(A(x) \cap B(x))$$

Translations to Formulas of $\mathcal{L}$

Given a mathematical statement **S** written with logical symbols.

We obtain a formula $A \in \mathcal{F}$ that is a **translation** of **S** into $\mathcal{L}$ by conducting a following sequence of steps.

Step 1 We **identify** basic statements in **S**, i.e. mathematical statements that involve only relations. They are to be translated into atomic formulas.

We **identify** the relations in the basic statements and **choose** the predicate symbols as their names.

We **identify** all functions and constants (if any) in the basic statements and **choose** the function symbols and constant symbols as their names.

Step 2 We **write** the basic statements as **atomic formulas** of $\mathcal{L}$.

Translations to Formulas of $\mathcal{L}$

Remember that in the predicate language $\mathcal{L}$ we write a function symbol in front of the function arguments not between them as we write in mathematics.

The same applies to relation symbols.

For example we re-write a basic mathematical statement $x + 2 > y$ as $>(+(x, 2), y)$, and then we write it as an **atomic formula** $P(f(x, c), y)$

$P \in \mathbf{P}$ stands for two argument relation $>$,

$f \in \mathbf{F}$ stands for two argument function $+$, and $c \in \mathbf{C}$ stands for the number 2.

Translations to Formulas of $\mathcal{L}$

Step 3 We **write** the statement **S** a formula with restricted quantifiers (if needed).

Step 4. We **apply** the **transformations rules** for restricted quantifiers to the formula from Step 3 and **obtain** a proper formula A of $\mathcal{L}$ as a result, i.e. as a **transtlation** of the given mathematical statement **S**.

In case of a translation from mathematical statement written without logical symbols **we add** a following step.

Step 0 We **identify** propositional connectives and quantifiers and use them to re-write the statement in a form that is as close to the structure of a logical formula as possible.

Translations Examples

Exercise

Given a mathematical statement **S** written with logical symbols

$$(\forall_{x \in N}\ x \geq 0\ \cap\ \exists_{y \in Z}\ y = 1)$$

1. Translate it into a proper logical formula with restricted quantifiers i.e. into a formula of $\mathcal{L}$ that **uses** the restricted domain quantifiers.

2. Translate your restricted quantifiers formula into a correct formula **without** restricted domain quantifiers, i.e. into a proper formula of $\mathcal{L}$.

A **long** and **detailed solution** is given in Chapter 2, page 28.

A short statement of the exercise and a short solution follows.

Exercise

Given a mathematical statement **S** written with logical symbols

$$(\forall_{x \in N}\ x \geq 0\ \cap\ \exists_{y \in Z}\ y = 1)$$

Translate it into a proper formula of $\mathcal{L}$.

Short Solution

The basic statements in **S** are: $x \in N,\ x \geq 0,\ y \in Z, y = 1$.

The corresponding atomic formulas of $\mathcal{L}$ are:
$N(x),\ G(x, c_1),\ Z(y),\ E(y, c_2)$, for
$n \in N,\ x \geq 0,\ y \in Z,\ y = 1$, respectively.

The statement **S** becomes restricted quantifiers formula

$$(\forall_{N(x)} G(x, c_1)\ \cap\ \exists_{Z(y)}\ E(y, c_2))$$

By the **transformation rules** we get $A \in \mathcal{F}$:

$$(\forall x(N(x) \Rightarrow G(x, c_1)) \cap \exists y(Z(y) \cap E(y, c_2)))$$

This is how you can write your solutions on Quizzes and Tests

Translations Examples

Exercise

Here is a mathematical statement **S**:

"For all real numbers x the following holds: If $x < 0$, then there is a natural number n, such that $x + n < 0$."

1. Re-write **S** as a symbolic mathematical statement SF that only uses mathematical and ogical symbols.

2. Translate the symbolic statement SF into to a corresponding formula $A \in \mathcal{F}$ of the predicate language $\mathcal{L}$

Translations Examples

Solution

The statement **S** is:

"For all real numbers x the following holds: If $x < 0$, then there is a natural number n, such that $x + n < 0$."

S becomes a symbolic mathematical statement SF

$$\forall_{x \in R}(x < 0 \Rightarrow \exists_{n \in N}\, x + n < 0)$$

We write R(x) for $x \in R$, N(y) for $n \in N$, a constant c for the number 0. We use $L \in \mathbf{P}$ to denote the relation $<$. We use $f \in \mathbf{F}$ to denote the function $+$.

The statement $x < 0$ becomes an **atomic formula** L(x, c).
The statement $x + n < 0$ becomes L(f(x,y), c).

Translations Examples

Solution c.d.

The symbolic mathematical statement SF

$$\forall_{x\in R}(x < 0 \Rightarrow \exists_{n\in N}\ x + n < 0)$$

becomes a restricted quantifiers formula

$$\forall_{R(x)}(L(x, c) \Rightarrow \exists_{N(y)} L(f(x, y), c))$$

We apply now the **transformation rules** and get a corresponding formula $A \in \mathcal{F}$:

$$\forall x(N(x) \Rightarrow (L(x, c) \Rightarrow \exists y(N(y) \cap L(f(x, y), c)))$$

Translations from Natural Language

Exercise

Translate a natural language statement

S: "Any friend of Mary is a friend of John and Peter is not John's friend. Hence Peter is not May's friend."

into a formula $A \in \mathcal{F}$ of the predicate language $\mathcal{L}$.

Solution

1. We identify the basic relations and functions (if any) and **translate** them into atomic formulas.

We have only one relation of "being a friend".

We **translate** it into an **atomic formula** F(x, y),

where F(x, y) stands for "x is a friend of y".

Translations from Natural Language

S: ”Any friend of Mary is a friend of John and Peter is not John’s friend. Hence Peter is not May’s friend.”

We use **constants** m, j, p for Mary, John, and Peter, respectively.

We hence have the following **atomic formulas**:

F(x, m), F(x, j), F(p, j), where

F(x, m) stands for ”x is a friend of Mary”,

F(x, j) stands for ”x is a friend of John”, and

F(p, j) stands for ”Peter is a friend of John”.

Translations from Natural Language

2. Statement "Any friend of Mary is a friend of John" **translates** into a restricted quantifier formula $\forall_{F(x,m)} F(x,j)$.

"Peter is not John's friend" **translates** into $\neg F(p,j)$, and

"Peter is not May's friend" **translates** into $\neg F(p,m)$.

3. Restricted quantifiers formula for **S** is

$$((\forall_{F(x,m)} F(x,j) \cap \neg F(p,j)) \Rightarrow \neg F(p,m))$$

and the formula $A \in \mathcal{F}$ of $\mathcal{L}$ is

$$((\forall x(F(x,m) \Rightarrow F(x,j)) \cap \neg F(p,j)) \Rightarrow \neg F(p,m)).$$

Rules of Translations

Rules of translation from natural language to the predicate language $\mathcal{L}$

1. Identify the basic relations and functions (if any) and **translate** them into **atomic formulas**.

2. Identify propositional connectives and use symbols $\neg, \cup, \cap, \Rightarrow, \Leftrightarrow$ for them.

3. Identify quantifiers: restricted $\forall_{A(x)}$, $\exists_{A(x)}$, and non-restricted $\forall x$, $\exists x$.

4. Use the symbols from **1.** - **3.** and restricted quantifiers **transformation rules** to write $A \in \mathcal{F}$ of the predicate language $\mathcal{L}$.

Translation Example

Exercise

Given a natural language statement

S: "For any bird one can find some birds that white."

Show that the **translation** of **S** into a formula of the predicate language $\mathcal{L}$ is $\forall x(B(x) \Rightarrow \exists x(B(x) \cap W(x)))$

Solution

We follow the rules of translation to **verify** the correctness of the translation.

1. Atomic formulas: B(x), W(x).

B(x) stands for " x is a bird" and W(x) stands for " x is white".

2. There is no propositional connectives in **S**.

Translation Example

3. Restricted quantifiers:

$\forall_{B(x)}$ for "any bird " and

$\exists_{B(x)}$ for "one can find some birds".

Restricted quantifiers formula for **S** is

$$\forall_{B(x)}\exists_{B(x)}\ W(x)$$

4. By the **transformation rules** we get a required formula of the predicate language $\mathcal{L}$:

$$\forall x(B(x) \Rightarrow \exists x(B(x) \cap W(x)))$$

Translation Example

Exercise

Translate into $\mathcal{L}$ a natural language statement
S: " Some patients like all doctors."

Solution

1. Atomic formulas: P(x), D(x), L(x, y).

P(x) stands for " x is a patient",

D(x) stands for " x is a doctor", and

L(x,y) stands for " x likes y".

2. There is no propositional connectives in **S**.

Translation Example

3. Restricted quantifiers:

$\exists_{P(x)}$ for "some patients " and $\forall_{D(x)}$ for "all doctors".

Observe that we **can't** write L(x, D(y)) for "x likes doctor y".

D(y) is a predicate, not a term, and hence L(x, D(y)) **is not a formula**.

We have to express the statement " x likes all doctors y" in terms of restricted quantifiers and the predicate L(x,y) only.

Translation Example

Observe that the statement " x likes all doctors y" means also " all doctors y are liked by x".

We can **re- write** it as "for all doctors y, x likes y" what translates to a formula $\forall_{D(y)} L(x, y)$.

Hence the statement **S** translates to

$$\exists_{P(x)} \forall_{D(x)} L(x, y).$$

4. By the transformation rules we get the following **translation** of **S** into $\mathcal{L}$.

$$\exists x (P(x) \cap \forall y (D(y) \Rightarrow L(x, y))).$$

Translation in Artificial Intelligence

In AI we usually deal with what is called an **intended interpretation**

It means we use logic symbols to describe, similarly as we do in mathematics, a concrete, specific universes with specific relations, functions or constants

In logic we use **general symbols** without any meaning

Logic is created to describe statements (formulas) and methods of reasoning that are universally applicable (tautologically true) and hence **independent** of any particular domain

Translation in Artificial Intelligence

In AI we use **intended names** for relations, functions and constants

The symbolic language we use is still a symbolic language, even if the **intended names** are used.

In the AI we write, for example

$$\text{Like(John, Mary)}$$

instead of a formula $L(c_1, c_2)$ in logic.

We write

$$\textit{greater(x, y)} \quad \text{or} \quad >(x, y)$$

instead of $R(x, y)$ in logic.

Example

AI formulas corresponding to a statement

S: "For every student there is a student that is an elephant"

are as follows.

Restricted quantifiers AI formula:

$$\forall_{Student(x)}\exists_{Student(x)}\ Elephant(x)$$

Non-restricted quantifiers AI formula:

$$\forall x(Student(x) \Rightarrow \exists x(Student(x) \cap Elephant(x)))$$

Translation in Artificial Intelligence

Observe that a proper formulas of the LOGIC language corresponding the statement

"For every student there is a student that is an elephant" are the same as the formulas corresponding to the natural language statement

For any bird one can find some birds that white", namely

Restricted quantifiers logic formula:

$$\forall_{P(x)} \exists_{P(x)} \; R(x)$$

Non-restricted quantifiers logic formula:

$$\forall x(P(x) \Rightarrow \exists x(P(x) \cap Rx)))$$

Translation in Artificial Intelligence

Statement "Any friend of Mary is a friend of John" translates in AI as follows.

Restricted quantifier AI formula:

$$((\forall_{Friend(x,Mary)}\ Friend(x, John)\ \cap \neg Friend(Peter, John))$$

$$(\Rightarrow \neg Friend(Peter, Mary))$$

Non-restricted AI formula:

$$((\forall x(Friend(x, Mary) \Rightarrow Friend(x, John))\ \cap \neg Friend(Peter, John))$$

$$\Rightarrow \neg Friend(Peter, Mary))$$