Introduction to Logic Programming

Foundations, First-Order Language

Temur Kutsia

Research Institute for Symbolic Computation
Johannes Kepler University Linz, Austria
kutsia@risc.jku.at

What is a Logic Program

- Logic program is a set of certain formulas of a first-order language.
- In this lecture: syntax and semantics of a first-order language.

Introductory Examples

- Representing "John loves Mary": $loves(John, Mary)$.
- $loves$: a binary predicate (relation) symbol.
- Intended meaning: The object in the first argument of $loves$ loves the object in its second argument.
- $John$, $Mary$: constants.
- Intended meaning: To denote persons John and Mary, respectively.

Introductory Examples

- $father$: A unary function symbol.
- Intended meaning: The father of the object in its argument.
- John's father loves John: $loves(father(John), John)$.

First-Order Language

- Syntax
- Semantics

Syntax

- Alphabet
- Terms
- Formulas

Alphabet

A first-order alphabet consists of the following disjoint sets of symbols:

- A countable set of variables $\mathcal{V}$.
- For each $n \geq 0$, a set of n-ary function symbols $\mathcal{F}^n$. Elements of $\mathcal{F}^0$ are called constants.
- For each $n \geq 0$, a set of n-ary predicate symbols $\mathcal{P}^n$.
- Logical connectives $\neg$, $\vee$, $\wedge$, $\Rightarrow$, $\Leftrightarrow$.
- Quantifiers $\exists$, $\forall$.
- Parenthesis '(', ')', and comma ','.

Notation:

- x, y, z for variables.
- f, g for function symbols.
- a, b, c for constants.
- p, q for predicate symbols.

Terms

Definition

- A variable is a term.
- If $t_1, \ldots, t_n$ are terms and $f \in \mathcal{F}^n$, then $f(t_1, \ldots, t_n)$ is a term.
- Nothing else is a term.

Notation:

- s, t, r for terms.

Example

- $plus(plus(x, 1), x)$ is a term, where $plus$ is a binary function symbol, 1 is a constant, x is a variable.
- $father(father(John))$ is a term, where $father$ is a unary function symbol and $John$ is a constant.

Formulas

Definition

- If $t_1, \ldots, t_n$ are terms and $p \in \mathcal{P}^n$, then $p(t_1, \ldots, t_n)$ is a formula. It is called an atomic formula.
- If A is a formula, $(\neg A)$ is a formula.
- If A and B are formulas, then $(A \vee B)$, $(A \wedge B)$, $(A \Rightarrow B)$, and $(A \Leftrightarrow B)$ are formulas.
- If A is a formula, then $(\exists x.A)$ and $(\forall x.A)$ are formulas.
- Nothing else is a formula.

Notation:

- A, B for formulas.

Eliminating Parentheses

- Excessive use of parentheses often can be avoided by introducing binding order.
- $\neg, \forall, \exists$ bind stronger than $\vee$.
- $\vee$ binds stronger than $\wedge$.
- $\wedge$ binds stronger than $\Rightarrow$ and $\Leftrightarrow$.
- Furthermore, omit the outer parentheses and associate $\vee, \wedge, \Rightarrow, \Leftrightarrow$ to the right.

Eliminating Parentheses

Example

The formula

$$(\forall y.(\forall x.((p(x)) \wedge (\neg r(y))) \Rightarrow ((\neg q(x)) \vee (A \vee B)))))$$

due to binding order can be rewritten into

$$(\forall y.(\forall x.(p(x) \wedge \neg r(y) \Rightarrow \neg q(x) \vee (A \vee B))))$$

which thanks to the convention of the association to the right and omitting the outer parentheses further simplifies to

$$\forall y.\forall x.(p(x) \wedge \neg r(y) \Rightarrow \neg q(x) \vee A \vee B).$$

Example

Translating English sentences into first-order logic formulas:

1. Every rational number is a real number.
   $$\forall x.(rational(x) \Rightarrow real(x))$$
2. There exists a number that is prime.
   $$\exists x.\, prime(x)$$
3. For every number x, there exists a number y such that $x < y$.
   $$\forall x.\, \exists y.\, x < y$$

Assume:

- $rational, real, prime$: unary predicate symbols.
- $<$: binary predicate symbol.

Example

Translating English sentences into first-order logic formulas:

1. There is no natural number whose immediate successor is 0.
$$\neg\exists x.zero \doteq succ(x)$$
2. For each natural number there exists exactly one immediate successor natural number.
$$\forall x.\exists y.(y \doteq succ(x) \land \forall z.(z \doteq succ(x) \Rightarrow y \doteq z))$$
3. For each nonzero natural number there exists exactly one immediate predecessor natural number.
$$\forall x.(\neg(x \doteq zero) \Rightarrow \exists y.(y \doteq pred(x) \land \forall z.(z \doteq pred(x) \Rightarrow y \doteq z)))$$

Assume:

- $zero$: constant
- $succ, pred$: unary function symbols.
- $\doteq$: binary predicate symbol.

Semantics

- Meaning of a first-order language consists of an universe and an appropriate meaning of each symbol.
- This pair is called structure.
- Structure fixes interpretation of function and predicate symbols.
- Meaning of variables is determined by a variable assignment.
- Interpretation of terms and formulas.

Structure

- Structure: a pair (D, I).
- D is a nonempty universe, the domain of interpretation.
- I is an interpretation function defined on D that fixes the meaning of each symbol associating
  - to each $f \in \mathcal{F}^n$ an n-ary function $f_I : D^n \to D$, (in particular, $c_I \in D$ for each constant c)
  - to each $p \in \mathcal{P}^n$ different from $\doteq$, an n-ary relation p_I on D.

Variable Assignment

- A structure $\mathcal{S} = (D, I)$ is given.
- Variable assignment $\sigma_{\mathcal{S}}$ maps each $x \in \mathcal{V}$ into an element of D: $\sigma_{\mathcal{S}}(x) \in D$.
- Given a variable x, an assignment $\vartheta_{\mathcal{S}}$ is called an x-variant of $\sigma_{\mathcal{S}}$ iff $\vartheta_{\mathcal{S}}(y) = \sigma_{\mathcal{S}}(y)$ for all $y \neq x$.

Interpretation of Terms

- A structure $\mathcal{S} = (D, I)$ and a variable assignment $\sigma_\mathcal{S}$ are given.
- Value of a term t under $\mathcal{S}$ and $\sigma_\mathcal{S}$, $Val_{\mathcal{S},\sigma_\mathcal{S}}(t)$:
  - $Val_{\mathcal{S},\sigma_\mathcal{S}}(x) = \sigma_\mathcal{S}(x)$.
  - $Val_{\mathcal{S},\sigma_\mathcal{S}}(f(t_1, \ldots, t_n)) = f_I(Val_{\mathcal{S},\sigma_\mathcal{S}}(t_1), \ldots, Val_{\mathcal{S},\sigma_\mathcal{S}}(t_n))$.

Interpretation of Formulas

- A structure $\mathcal{S} = (D, I)$ and a variable assignment $\sigma_\mathcal{S}$ are given.
- Value of an atomic formula under $\mathcal{S}$ and $\sigma_\mathcal{S}$ is one of *true*, *false*:
  - $Val_{\mathcal{S},\sigma_\mathcal{S}}(s \doteq t) = true$ iff $Val_{\mathcal{S},\sigma_\mathcal{S}}(s) = Val_{\mathcal{S},\sigma_\mathcal{S}}(t)$.
  - $Val_{\mathcal{S},\sigma_\mathcal{S}}(p(t_1, \ldots, t_n)) = true$ iff $(Val_{\mathcal{S},\sigma_\mathcal{S}}(t_1), \ldots, Val_{\mathcal{S},\sigma_\mathcal{S}}(t_n)) \in p_I$.

Interpretation of Formulas

- A structure $\mathcal{S} = (D, I)$ and a variable assignment $\sigma_\mathcal{S}$ are given.
- Values of compound formulas under $\mathcal{S}$ and $\sigma_\mathcal{S}$ are also either *true* or *false*:
  - $Val_{\mathcal{S},\sigma_\mathcal{S}}(\neg A) = true$ iff $Val_{\mathcal{S},\sigma_\mathcal{S}}(A) = false$.
  - $Val_{\mathcal{S},\sigma_\mathcal{S}}(A \vee B) = true$ iff $Val_{\mathcal{S},\sigma_\mathcal{S}}(A) = true$ or $Val_{\mathcal{S},\sigma_\mathcal{S}}(B) = true$.
  - $Val_{\mathcal{S},\sigma_\mathcal{S}}(A \wedge B) = true$ iff $Val_{\mathcal{S},\sigma_\mathcal{S}}(A) = true$ and $Val_{\mathcal{S},\sigma_\mathcal{S}}(B) = true$.
  - $Val_{\mathcal{S},\sigma_\mathcal{S}}(A \Rightarrow B) = true$ iff $Val_{\mathcal{S},\sigma_\mathcal{S}}(A) = false$ or $Val_{\mathcal{S},\sigma_\mathcal{S}}(B) = true$.
  - $Val_{\mathcal{S},\sigma_\mathcal{S}}(A \Leftrightarrow B) = true$ iff $Val_{\mathcal{S},\sigma_\mathcal{S}}(A) = Val_{\mathcal{S},\sigma_\mathcal{S}}(B)$.
  - $Val_{\mathcal{S},\sigma_\mathcal{S}}(\exists x.A) = true$ iff $Val_{\mathcal{S},\vartheta_\mathcal{S}}(A) = true$ for some x-variant $\vartheta_\mathcal{S}$ of $\sigma_\mathcal{S}$.
  - $Val_{\mathcal{S},\sigma_\mathcal{S}}(\forall x.A) = true$ iff $Val_{\mathcal{S},\vartheta_\mathcal{S}}(A) = true$ for all x-variants $\vartheta_\mathcal{S}$ of $\sigma_\mathcal{S}$.

Interpretation of Formulas

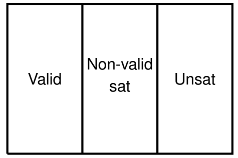
- A structure $\mathcal{S} = (D, I)$ is given.
- The value of a formula A under $\mathcal{S}$ is either *true* or *false*:
  - $Val_\mathcal{S}(A) = true$ iff $Val_\mathcal{S}, \sigma_\mathcal{S}(A) = true$ for all $\sigma_\mathcal{S}$.
- $\mathcal{S}$ is called a model of A iff $Val_\mathcal{S}(A) = true$.
- Written $\models_\mathcal{S} A$.

Example

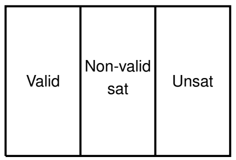
- Formula: $\forall x.(p(x) \Rightarrow q(f(x), a))$
- Define $\mathcal{S} = (D, I)$ as
  - $D = \{1, 2\}$,
  - $a_I = 1$,
  - $f_I(1) = 2, f_I(2) = 1$,
  - $p_I = \{2\}$,
  - $q_I = \{(1,1), (1,2), (2,2)\}$.
- If $\sigma_{\mathcal{S}}(x) = 1$, then $Val_{\mathcal{S},\sigma_{\mathcal{S}}}(\forall x.(p(x) \Rightarrow q(f(x), a))) = true$.
- If $\sigma_{\mathcal{S}}(x) = 2$, then $Val_{\mathcal{S},\sigma_{\mathcal{S}}}(\forall x.(p(x) \Rightarrow q(f(x), a))) = true$.
- Hence, $\models_{\mathcal{S}} A$.

Validity, Unsatisfiability

- A formula A is valid, if $\models_{\mathcal{S}} A$ for all $\mathcal{S}$.
- Written $\models A$.
- A formula A is unsatisfiable, if $\models_{\mathcal{S}} A$ for no $\mathcal{S}$.
- If A is valid, then $\neg A$ is unsatisfiable and vice versa.
- The notions extend to (multi)sets of formulas.
- For $\{A_1, \ldots, A_n\}$, just formulate them for $A_1 \wedge \cdots \wedge A_n$.



Validity, Unsatisfiability



- $\forall x.p(x) \Rightarrow \exists y.p(y)$ is valid.
- $p(a) \Rightarrow \neg\exists x.p(x)$ is satisfiable non-valid.
- $\forall x.p(x) \wedge \exists y.\neg p(y)$ is unsatisfiable.

Logical Consequence

Definition
A formula A is a logical consequence of the formulas $B_1, \ldots, B_n$, if every model of $B_1 \wedge \cdots \wedge B_n$ is a model of A.

Example
- $mortal(socrates)$ is a logical consequence of $\forall x.(person(x) \Rightarrow mortal(x))$ and $person(socrates)$.
- $cooked(apple)$ is a logical consequence of $\forall x.(\neg cooked(x) \Rightarrow tasty(x))$ and $\neg tasty(apple)$.
- $genius(einstein)$ is not a logical consequence of $\exists x.person(x) \wedge genius(x)$ and $person(einstein)$.

Logic Programs

- Logic programs: finite non-empty sets of formulas of a special form, called program clauses.
- Program clause:

$$\forall x_1 . \dots \forall x_k . B_1 \wedge \dots \wedge B_n \Rightarrow A,$$

  where
  - $k, n \geq 0$,
  - A and the B's are atomic formulas,
  - $x_1, \dots, x_k$ are all the variables which occur in $A, B_1, \dots, B_n$.
- Usually written in the inverse implication form without quantifiers and conjunctions:

$$A \Leftarrow B_1, \dots, B_n$$

Goal

- Goals or queries of logic programs: formulas of the form

$$\exists x_1 . \dots \exists x_k . B_1 \wedge \dots \wedge B_n,$$

  where
  - $k, n \geq 0$,
  - the B's are atomic formulas,
  - $x_1, \dots, x_k$ are all the variables which occur in $B_1, \dots, B_n$.
- Usually written without quantifiers and conjunction:

$$B_1, \dots, B_n$$

- The problem is to find out whether a goal is a logical consequence of the given logic program or not.

The Problem and the Idea

- Let P be a program and G be a goal.
- Problem: Is G a logical consequence of P?
- Idea: Try to show that the set of formulas $P \cup \{\neg G\}$ is inconsistent.
- How? This we will learn in this course.

Example

Let P consist of the two clauses:

- $\forall x . mortal(x) \Leftarrow person(x)$.
- $person(socrates)$.

Goal: $G = \exists x . mortal(x)$.

$\neg G$ is equivalent to $\forall x . \neg mortal(x)$.

The set

$$\{\forall x . mortal(x) \Leftarrow person(x),\ person(socrates),\ \forall x . \neg mortal(x)\}$$

is inconsistent.

Hence, G is a logical consequence of P.

We can even compute the witness term for the goal: $x = socrates$.

How? This we will learn in this lecture.