

Second-Order Logic

Final seminar for "Logic in Computer Science"

Mattia Zorzan

University of Verona

May 28, 2022

Outline

1 Introduction

- A recall on first-order logic
 - Syntax
 - Semantics

Introduction

- First-order logic allows "iteration" over the *elements* of a structure
- Happens thanks to **quantifiers**: \forall, \exists
 - ▶ $\forall x. \phi(x) \rightarrow$ "For each x , x satisfies the formula ϕ "
 - ▶ $\exists x. \phi(x) \rightarrow$ "There exists x s.t. the formula ϕ is satisfied"
- Limiting since we may only need to range over *subsets* or "*combinations*" (e.g. *Cartesian product*)

A brief recall

- Second-order logic "extends" first-order logic
- Since that, let's recall the basics of first-order logic
- Two key parts:
 - ▶ *Syntax*: Which sequences constitute **well-formed** expressions
 - ▶ *Semantics*: The **meaning** behind this expressions

Syntax - Introduction

- Two base types:
 - ▶ **Terms:** Represents *objects*
 - ▶ **Formulas:** Represents *predicates*
- Both formed by *symbol* concatenation
- All symbols together form the **alphabet** of the language
- Can divide symbols in two categories
 - ▶ *Logical* symbols
 - ▶ *Non-logical* symbols

Syntax - Logical symbols

- Infinite set of **variables**: $x, y, z, \dots, x_0, x_1, \dots$ (Lowercase letters)
- **Connectives**: $\wedge, \vee, \Rightarrow, \neg$
- **Quantifiers**: \forall, \exists
- **Equality** (or *Identity*): $=$
- **Auxiliary symbols**: $(;); .$ (dot); $,$ (comma)

Syntax - Non-logical symbols

- Represents *predicates* (or *relations*), *functions* and *constants*
- $\forall n \in \mathbb{Z}^*$ we have a set of n -ary **predicate symbols**

P_0^n, P_1^n, \dots (Uppercase letters)

- $\forall n \in \mathbb{Z}^*$ there exist infinite n -ary **function symbols**

f_0^n, f_1^n, \dots (Lowercase letters)

Syntax - Formation rules (1)

Definition (Terms formation)

The set **TERM** of *terms* can be inductively defined by the following rules:

- 1 If x is a variable, then $x \in \text{TERM}$
- 2 Any expression $f(t_1, \dots, t_n)$, with $t_1, \dots, t_n \in \text{TERM}$, is a term.
Since that, the following statement holds

$$f(t_1, \dots, t_n) \in \text{TERM}$$

Syntax - Formation rules (2)

Definition (Formulas formation)

The set FORM of *formulas* can be inductively defined by the following rules:

- 1 If $P \in \text{PRED}^a$ and $t_1, \dots, t_n \in \text{TERM}$, then $P(t_1, \dots, t_n) \in \text{FORM}$
- 2 If $t_1, t_2 \in \text{TERM}$, then $t_1 = t_2 \in \text{FORM}$
- 3 If $\phi \in \text{FORM}$, then $\neg\phi \in \text{FORM}$
- 4 If $\phi, \psi \in \text{FORM}$, then $\phi \square \psi \in \text{FORM}$ (with $\square \in \{\wedge, \vee, \Rightarrow\}$)
- 5 If $\phi \in \text{FORM}$ and x is a variable, then $Qx.\phi \in \text{FORM}$ (with $Q \in \{\forall, \exists\}$)

^aThe set of *predicate symbols*

Syntax - Variables (1)

Definition (Free and Bound variables)

The *free* and *bound* variable occurrences in a formula are defined inductively by the following rules:

- 1 If ϕ is *atomic*, then any variable $x \in \text{Var}(\phi)$ is *free*
- 2 x is *free/bound* in $\neg\phi$ iff x is *free/bound* in ϕ
- 3 x is *free/bound* in $\phi \square \psi$ iff x is *free/bound* in either ϕ or ψ (with $\square \in \{\wedge, \vee, \Rightarrow\}$)
- 4 x is *free* in $Qy.\phi$ iff x is *free* in ϕ and $y \neq x$
- 5 x is *bound* in $Qy.\phi$ iff x is *bound* in ϕ

Syntax - Variables (2)

- More easily, a variable x is *bounded* if it occurs in a quantification, x is *free* otherwise
- A variable can be both *free* and *bounded* in the same formula, e.g.

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

- 1 In the **LHS** x is *free*
 - 2 In the **RHS** x is *bounded*
 - 3 Even so, the formula is still *well-formed*
- A formula with no *free* variables is called a **sentence**

Semantics - Introduction

- Interpretation defines the **domain** D , a non-empty set of "*objects*"
- Interpretation *maps* functions, predicates and constants symbols into elements of D

Definition (Structure)

A *structure* is formed by a *domain* D and an *interpretation function* I s.t.

- 1 Each function symbol f of arity n is assigned to a function s.t.

$$I(f) : D^n \rightarrow D$$

- 2 Each predicate symbol f of arity n is assigned to a function s.t.

$$I(P) : D^n \rightarrow \{\text{true}, \text{false}\}$$