Second-Order Logic

Final seminar for "Logic in Computer Science"

Mattia Zorzan

University of Verona

May 27, 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**: $x, y, z, \ldots, x_0, x_1, \ldots$ (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**

$$P_0^n, P_1^n, \dots \qquad \text{(Uppercase letters)}$$

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

$$f_0^n, f_1^n, \dots \qquad \text{(Lowercase letters)}$$

Syntax - Formation rules (1)

Definition (Terms formation)

The set $\mathtt{TERM}$ of *terms* can be inductively defined by the following rules:

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

$$f(t_1, \ldots, t_n) \in \mathtt{TERM}$$

Syntax - Formation rules (2)

Definition (Formulas formation)

The set $\mathtt{FORM}$ of *formulas* can be inductively defined by the following rules:

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

[a] The 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*, than any variable $x \in Var(\phi)$ 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