Second-Order Logic

Final seminar for "Logic in Computer Science"

Mattia Zorzan

University of Verona

Last rev. June 13, 2022

Outline

1. Introduction
   - A recall on first-order logic
     - Syntax
     - Semantic
2. Second-Order Logic

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

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**

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

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

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

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

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

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 $\phi \,\square\, \psi \in \mathtt{FORM}$ (with $\square \in \{\wedge, \vee, \Rightarrow\}$)
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 $\phi \,\Box\, \psi$ iff x is *free/bound* in either ϕ or ψ (with $\Box \in \{\wedge, \vee, \Rightarrow\}$)
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**

Semantic

Semantic - Structure and Interpretation

Definition (Structure)

A *structure* is formed by a *domain* D, $\mathbb{P} = \{P_1, \ldots, P_n\}$ predicates on D, $\mathbb{F} = \{f_1, \ldots, f_n\}$ **total** functions on D and a set $\mathbb{C} \subseteq D$ of constants

Definition (Interpretation)

Given a *structure* $\mathfrak{D}$ and a *map* $(\cdot)^{\mathfrak{D}}$ s.t.

- for all c in my language, $(c)^{\mathfrak{D}} = c^{\mathfrak{D}} \in \mathbb{C}$
- for all *k-ary* function f in my language, $(f)^{\mathfrak{D}} = f^{\mathfrak{D}} : D^k \to D \in \mathbb{F}$
- for all *n-ary* predicate P in my language, $(P)^{\mathfrak{D}} = P^{\mathfrak{D}} \subseteq D^k \in \mathbb{P}$

we call $\langle \mathfrak{D}, (\cdot)^{\mathfrak{D}} \rangle$ an *interpretation*.

Semantic - Evaluation (1)

- Given an *interpretation* and an **assignment** $\overline{a}$, it is possible to evaluate a formula
- The **evaluation** process *maps* the whole formula to a **truth value**
- The *assignment* $\overline{a}$ associates each *free variable* with a *truth value*
- If the formula is a *sentence*, $\overline{a}$ does not affect the *truth value* of the formula
- Next slides shows the evaluation steps

Semantic - Evaluation (2)

1. Extend $\overline{a}$ to all terms of the language with the following rules:
   - Each variable x evaluates to $\overline{a}(x)$
   - Given $\{t_1, \dots, t_n\} \in \mathtt{TERM}$ evaluated to $\{d_1, \dots, d_n\}$, a function $f(t_1, \dots, t_n)$ evaluates to $(f)^{\mathfrak{D}}$[1]$(d_1, \dots, d_n)$
2. Assign each formula to a *truth value* with the following (inductive) rules:
   - (*Continues in next slides*)

[1] Supposing we're evaluating f in a structure $\mathfrak{D}$

Semantic - Evaluation (3)

- An *atomic formula* $P(t_1, \ldots, t_n)$ is associated with a *truth value*, depending on the truth of the following:

$$\langle v_1, \ldots, v_n \rangle \in (P)^{\mathfrak{D}}$$

  where $v_1, \ldots, v_n$ represents the evaluation of the predicate terms
- An *atomic formula* $t_1 = t_2$ evaluates to a *truth value* depending if $v_1 = v_2$ in D, where v_1, v_2 represents the evaluation of the terms

Semantic - Evaluation (4)

- A formula containing *logical connectives* (e.g. $\phi \,\Box\, \psi$[2], $\neg\phi$) is evaluated according to the *truth table* of the connective
- A formula $\exists x.\phi$ is evaluated `true` iff exists an assignment $\overline{a}'$ s.t. it differs from $\overline{a}$ only for the assignment of x and ϕ is evaluated `true` via the $\overline{a}'$ assignment, `false` otherwise
- A formula $\forall x.\phi$ is evaluated `true` iff exists an assignment $\overline{a}'$ s.t. it differs from $\overline{a}$ only for the assignment of x and ϕ is evaluated `true` for all values in $\overline{a}'$, `false` otherwise

[2] with $\Box \in \{\wedge, \vee, \Rightarrow\}$

Semantic - Evaluation (5)

- Given a *structure* $\mathfrak{D}$, evaluation can be seen as a *map*

$$\rho_{\mathfrak{D}} : Var \to D$$

- We can use the $[\![\cdot]\!]$ notation to express the evaluation of a *term*

Definition

Given a structure $\mathfrak{D}$ and $[\![\cdot]\!]_{\rho_{\mathfrak{D}}} : \mathtt{TERM} \to D$, we can define

1. $[\![c]\!]_{\rho_{\mathfrak{D}}} = c^{\mathfrak{D}}$
2. $[\![x]\!]_{\rho_{\mathfrak{D}}} = \rho_{\mathfrak{D}}(x)$
3. $[\![f(t_1, \ldots, t_n)]\!]_{\rho_{\mathfrak{D}}} = f^{\mathfrak{D}}([\![t_1]\!]_{\mathfrak{D}}, \ldots, [\![t_n]\!]_{\mathfrak{D}})$

Semantic - Satisfiability

Definition (Satisfiability)

Given a *structure* $\mathfrak{D}$ we can recursively define the *satisfiability relation* $\models$ as:

1. $\rho_{\mathfrak{D}} \not\models \perp$
2. $\rho_{\mathfrak{D}} \models P(t_1, \ldots, t_n) \Leftrightarrow \langle [\![t_1]\!]_{\rho_{\mathfrak{D}}}, \ldots, [\![t_n]\!]_{\rho_{\mathfrak{D}}} \rangle \in \mathbb{R}$
3. $\rho_{\mathfrak{D}} \models t_1 = t_2 \Leftrightarrow [\![t_1]\!]_{\rho_{\mathfrak{D}}} = [\![t_2]\!]_{\rho_{\mathfrak{D}}}$
4. $\rho_{\mathfrak{D}} \models (\phi \Rightarrow \psi) \Leftrightarrow \rho_{\mathfrak{D}} \not\models \phi$ o $\rho_{\mathfrak{D}} \models \psi$
5. $\rho_{\mathfrak{D}} \models (\phi \wedge \psi) \Leftrightarrow \rho_{\mathfrak{D}} \models \phi$ e $\rho_{\mathfrak{D}} \models \psi$
6. $\rho_{\mathfrak{D}} \models (\phi \vee \psi) \Leftrightarrow \rho_{\mathfrak{D}} \models \phi$ o $\rho_{\mathfrak{D}} \models \psi$
7. $\rho_{\mathfrak{D}} \models \forall x.\, \phi \Leftrightarrow \forall a \in D : \rho_{\mathfrak{D}}[x/a] \models \phi$
8. $\rho_{\mathfrak{D}} \models \exists x.\, \phi \Leftrightarrow \exists a \in D : \rho_{\mathfrak{D}}[x/a] \models \phi$

Second-Order Logic

Syntax and Semantic

- As said before, second-order logic *extends* first-order logic
- The *terms* and *formulas* are unchanged...
- ...but we need to redefine *structures* to represent this "*extension*"

Structure

Definition (Structure)

A *structure* is formed by a *domain* D, a set $D^* = \langle D_n \mid n \in \mathbb{N} \rangle$ with $D_n \subseteq \mathcal{P}(A^n)$, a set $\mathbb{P} = \{P_1^n, \ldots, P_k^n\}$ of *predicates* s.t. $P_i^n \in D_n$ and a set of *constants* $\mathbb{C} \subseteq D$

- If D_n contains **all** n-ary predicates ($D_n = \mathcal{P}(D^n)$) we call the structure *full*