

Second-Order Logic

Final seminar for "Logic in Computer Science"

Mattia Zorzan

University of Verona

Last rev. June 15, 2022

Outline

1 Introduction

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

2 Second-Order Logic

- Syntax
- Semantic
- Comprehension schema
- Flattening SOL

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
 - ▶ *Semantic*: 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

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)

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)

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

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

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*

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

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

Structure and Interpretation

Definition (Structure)

A *structure* is formed by a *domain* D , $\mathbb{P} = \{P_1, \dots, P_n\}$ predicates on D , $\mathbb{F} = \{f_1, \dots, f_n\}$ **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, $(f)^{\mathfrak{D}} = f^{\mathfrak{D}} : D^k \rightarrow D \in \mathbb{F}$
- 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*.

Evaluation (1)

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

Evaluation (2)

- ① Extend \bar{a} to all terms of the language with the following rules:
 - ▶ Each variable x evaluates to $\bar{a}(x)$
 - ▶ Given $\{t_1, \dots, t_n\} \in \text{TERM}$ 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*)

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

Evaluation (3)

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

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

where v_1, \dots, 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

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 \bar{a}' s.t. it differs from \bar{a} only for the assignment of x and ϕ is evaluated true via the \bar{a}' assignment, false otherwise
- A formula $\forall x.\phi$ is evaluated true iff exists an assignment \bar{a}' s.t. it differs from \bar{a} only for the assignment of x and ϕ is evaluated true for all values in \bar{a}' , false otherwise

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

Evaluation (5)

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

$$\rho_{\mathfrak{D}} : \text{Var} \rightarrow D$$

- We can use the $\llbracket \cdot \rrbracket$ notation to express the evaluation of a *term*

Definition

Given a structure \mathfrak{D} and $\llbracket \cdot \rrbracket_{\rho_{\mathfrak{D}}} : \text{TERM} \rightarrow D$, we can define

- $\llbracket c \rrbracket_{\rho_{\mathfrak{D}}} = c^{\mathfrak{D}}$
- $\llbracket x \rrbracket_{\rho_{\mathfrak{D}}} = \rho_{\mathfrak{D}}(x)$
- $\llbracket f(t_1, \dots, t_n) \rrbracket_{\rho_{\mathfrak{D}}} = f^{\mathfrak{D}}(\llbracket t_1 \rrbracket_{\rho_{\mathfrak{D}}}, \dots, \llbracket t_n \rrbracket_{\rho_{\mathfrak{D}}})$

Satisfiability

Definition (Satisfiability relation)

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

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

Natural Deduction

- Just as reference, we introduce the *quantification rules* for the First-Order natural deduction system

$$\frac{\phi(x)}{\forall y.\phi(y)} \forall I$$

$$\frac{\phi(y)}{\exists x.\phi(x)} \exists I$$

$$\frac{\forall x.\phi(x)}{\phi(y)} \forall E$$

$$\frac{\begin{array}{c} [\phi(x)] \\ \Pi \\ \exists y.\phi(y) \end{array} \quad \psi}{\psi} \exists E$$

Second-Order Logic

Introduction

- As said before, Second-Order Logic *extends* First-Order Logic (respectively SOL and FOL from now on)
- The *terms* and *formulas* are pretty much the same of FOL
- To represent this "*extension*" we need a brief redefinitin of what we've seen in the previous section

Syntax

Alphabet

- Consists of the following symbols:
 - ▶ **Individual variables:** x_0, x_1, \dots
 - ▶ **Individual constants:** c_0, c_1, \dots
 - ▶ **Predicate variables:** X_0^n, X_1^n, \dots
 - ▶ **Predicate constants:** $\perp, P_0^n, P_1^n, \dots$
 - ▶ **Connectives:** $\wedge, \vee, \Rightarrow, \neg$
 - ▶ **Quantifiers:** \forall, \exists
 - ▶ **Auxiliary symbols:** $(;); .$ (dot); $,$ (comma)

Formulas

- The set FORM of Second-Order *formulas* is inductively defined as follows
 - ▶ $X_i^0, P_i^0, \perp \in \text{FORM}$
 - ▶ $\forall n \in \mathbb{Z}^*. X^n(t_1, \dots, t_n) \in \text{FORM}$
 - ▶ $\forall n \in \mathbb{Z}^*. P^n(t_1, \dots, t_n) \in \text{FORM}$
- FORM is *closed* under **connectives** and **quantifiers**

Semantic

Definition (Second-Order 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 $\mathbb{R} = \{R_1^n, \dots, R_k^n\}$ of *predicates* s.t. $R_i^n \in D_n$ 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*
- Even if the *elements* of a Second-Order structure are slightly different from the elements of a First-Order structure, we can use the same rules for **interpretation** and **evaluation**

Satisfiability

Definition (Second-Order Satisfiability relation)

Given a *Second-Order structure* \mathfrak{D} and a *language* \mathcal{L} that defines a name \bar{S} for all $S \in D$, we can define the *satisfiability relation* \models_2 as:

- ① $\rho_{\mathfrak{D}} \not\models_2 \perp$
- ② $\rho_{\mathfrak{D}} \models_2 \bar{S}^n(\bar{s}_1, \dots, \bar{s}_n) \Leftrightarrow \langle s_1, \dots, s_n \rangle \in S^{na}$
- ③ All connectives follow the same rules of First-Order Logic
- ④ Quantification over a *variable* follow the same rules of First-Order Logic
- ⑤ $\rho_{\mathfrak{D}} \models_2 \forall X_i^n. \phi(P_i^n) \Leftrightarrow \forall S^n \in D_n : \rho_{\mathfrak{D}} \models \phi(S^n)$
- ⑥ $\rho_{\mathfrak{D}} \models_2 \exists X_i^n. \phi(P_i^n) \Leftrightarrow \exists S^n \in D_n : \rho_{\mathfrak{D}} \models \phi(\bar{S}^n)$

^aBy using this notation, we can handle all types of *predicates* with one rule

Natural Deduction

- We need to add a set of *rules* that allows to validly **derive** the new "extended" *quantifications*

$$\frac{\phi}{\forall X^n.\phi} \forall^2 I$$

$$\frac{\phi^*}{\exists X^n.\phi} \exists^2 I$$

$$\frac{\forall X^n.\phi}{\phi^*} \forall^2 E$$

$$\frac{\begin{array}{c} [\phi] \\ \Pi \\ \exists X^n.\phi \end{array} \quad \psi}{\psi} \exists^2 E$$

- ϕ^* is $\phi[X^n(t_1, \dots, t_n)/\psi(t_1, \dots, t_n)]$, where ψ is a generic formula
- No t_i becomes *bounded* during the substitution above, so ψ cannot *quantify* any term t_1, \dots, t_n

Comprehension schema (1)

- States that "Any definable subclass of a set is a set", formally:

Definition (Axiom schema of comprehension)

Given a *formula* ψ with $FV[\psi] \in \{x, t_1, \dots, t_n\}$, and a set A , the following holds

$$\forall t_1, \dots, t_n. \forall A. \exists B. \forall x. (x \in B \Leftrightarrow (x \in A \wedge \psi(x, t_1, \dots, t_n, A)))$$

- Since the schema holds for all x and for all A , for the generality of ψ , it is always possible to define a set from another set
- $\exists^2 I$ gives us this schema, since we can substitute P^n with ψ , obtaining the "filtered" subset

Comprehension schema (2)

Proof idea.

Since we can derive

$$\forall t_1, \dots, t_n. (\phi(t_1, \dots, t_n) \Leftrightarrow \phi(t_1, \dots, t_n))$$

the following is correct

$$\frac{\forall t_1, \dots, t_n. (\phi(t_1, \dots, t_n) \Leftrightarrow \phi(t_1, \dots, t_n))}{\exists X^n. \forall t_1, \dots, t_n. (\phi(t_1, \dots, t_n) \Leftrightarrow P^n(t_1, \dots, t_n))} \exists^2 I$$



- **Strong result!** We can derive *Second-Order quantification* the same way our deduction system derived *First-Order quantification*
- Same proof concept can be applied for $\forall^2 E$, since we can define \forall^2 from \exists^2

Flattening SOL (1)

- Given the *First-Order predicates* Ap_0, Ap_1, \dots s.t. Ap_n is $(n + 1)$ -ary
- $Ap_n(X, t_1, \dots, t_n)$ can be seen as $X^n(t_1, \dots, t_n)$, so Ap_0 is the First-Order version of X^0
- We also use some *unary predicates*
 - ▶ $V \rightarrow$ "is an element"
 - ▶ $U_0 \rightarrow$ "is an 0-ary predicate"
 - ▶ $U_1 \rightarrow$ "is an 1-ary predicate"
 - ▶ *and so on...*

Flattening SOL (2)

- ① $\forall x, y, z. (U_i(x) \wedge U_j(y) \wedge V(z) \Rightarrow x \neq y \wedge y \neq z \wedge z \neq x)$ for all $i \neq j$
- ② $\forall X, y_1, \dots, y_n. (Ap_n(X, y_1, \dots, y_n) \Rightarrow U_n(P) \wedge \bigwedge_i V(y_i))$ for $n \geq 1$
- ③ $U_0(C_0, V(C_{2^{i+1}}))$ for $i \geq 0$ and $U_n(C_{3 \cdot 5^n})$ for $i, n \geq 0$
 $\forall z_1, \dots, z_m. \exists P. ($
 - ④ $U_n(X) \wedge \forall y_1, \dots, y_n. \left(\bigwedge V(y_i) \Rightarrow (\phi^* \Leftrightarrow Ap_n(X, y_1, \dots, y_n)) \right)$
 $)$ where $X \notin FV[\phi^*], FV[\phi] \in \{z_1, \dots, z_m, y_1, \dots, y_n\}$
- ⑤ $\neg Ap_0(C_0)$

Flattening SOL (3)

- 1 The i -ary and j -ary predicates are pairwise disjoint and disjoint from the element
- 2 If $\langle X, y_1, \dots, y_n \rangle \in Ap_n$ then X is a *predicate* and y_1, \dots, y_n are its *elements*
- 3 We can have both *element* and *predicate* constants
- 4 First-Order equivalent of the *comprehension schema*
- 5 The 0-ary predicate for "false", equivalent to \perp

Flattening SOL (4)

- SOL can be translated preserving **derivability**
- We can assign symbols to the ones in SOL alphabet, in order to convert strings inductively

- ▶ $(x_i)^* \leftarrow x_{2^{i+1}}$
- ▶ $(c_i)^* \leftarrow c_{2^{i+1}}$
- ▶ $(X_i^n)^* \leftarrow x_{3^i 5^n}$
- ▶ $(P_i^n)^* \leftarrow c_{3^i 5^n}$
- ▶ $(X_i^0)^* \leftarrow Ap_0(x_{3^i})$
- ▶ $(P_i^0)^* \leftarrow Ap_0(c_{3^i})$
- ▶ $(\perp)^* \leftarrow Ap_0(c_0)$

with $i, n \geq 0$

Flattening SOL (5)

- Doing this, we can "*translate*" Second-Order formulas as follows:

- ▶ $(\phi \Box \psi)^* \leftarrow \phi^* \Box \psi^*$
- ▶ $(\neg \phi)^* \leftarrow \neg \phi^*$
- ▶ $(\forall x_i. \phi(x_i))^* \leftarrow \forall x_i^*. (V(x_i^*) \Rightarrow \phi^*(x_i^*))$
- ▶ $(\exists x_i. \phi(x_i))^* \leftarrow \exists x_i^*. (V(x_i^*) \wedge \phi^*(x_i^*))$
- ▶ $(\forall X_i^n. \phi(x_i^n))^* \leftarrow \forall (x_i^n)^*. (U_n((X_i^n)^*) \Rightarrow \phi^*((X_i^n)^*))$
- ▶ $(\exists X_i^n. \phi(x_i^n))^* \leftarrow \exists (x_i^n)^*. (U_n((X_i^n)^*) \wedge \phi^*((X_i^n)^*))$

Model

Definition (Second-Order model)

Given a *structure* \mathfrak{D} , it is called a **model** of SOL if it admits (a.k.a is *valid*) the *comprehension schema*.

If \mathfrak{D} is *full*, then it is called a **principal** (or *standard*) **model**

- From the definition above we get two distinct notions of "validity" in SOL
 - ▶ true in all models
 - ▶ true in all *principal* models
- We'll use the first one as default