Second-Order Logic

Final seminar for "Logic in Computer Science"

Mattia Zorzan

University of Verona

Last rev. June 24, 2022

Outline

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

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

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

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

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

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

Formation rules (2)

Definition (Formulas formation)

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

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

[a] The 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*, 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

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, $\mathbb{P} = \{P_1, \ldots, P_n\}$ 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*.

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

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 $\{t_1, \ldots, t_n\} \in \mathtt{TERM}$ evaluated to $\{d_1, \ldots, d_n\}$, a function $f(t_1, \ldots, t_n)$ evaluates to $(f)^{\mathfrak{D}}$[1]$(d_1, \ldots, d_n)$
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 $\mathfrak{D}$

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. $\phi \,\square\, \psi$[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 $\square \in \{\wedge, \vee, \Rightarrow\}$

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

Satisfiability

Definition (Satisfiability relation)

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

1. $\rho_{\mathfrak{D}} \not\models \bot$
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. $\rho_{\mathfrak{D}} \models \forall x.\, \phi(x) \Leftrightarrow \forall a \in D : \rho_{\mathfrak{D}}[x/a] \models \phi$
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

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

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

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 redefinition of what we've seen in the previous section

Syntax

Alphabet

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

Formulas

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

Semantic

Structure

Definition (Second-Order Structure)

A *Second-Order 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 $\mathbb{R} = \{R_1^n, \ldots, R_k^n\}$ 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* $\mathfrak{D}_2$ and a *language* $\mathcal{L}$ that defines a name $\overline{S}$ for all $S \in D$, we can define the *satisfiability relation* $\vDash_2$ as:

1. $\rho_{\mathfrak{D}_2} \nvDash_2 \perp$
2. $\rho_{\mathfrak{D}_2} \vDash_2 \overline{S}^n(\overline{s}_1, \ldots, \overline{s}_n) \Leftrightarrow \langle s_1, \ldots, s_n \rangle \in S^n$[a]
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. $\rho_{\mathfrak{D}_2} \vDash_2 \forall X_i^n . \phi(P_i^n) \Leftrightarrow \forall S^n \in D_n : \rho_{\mathfrak{D}_2} \vDash_2 \phi(S^n)$
6. $\rho_{\mathfrak{D}_2} \vDash_2 \exists X_i^n . \phi(P_i^n) \Leftrightarrow \exists S^n \in D_n : \rho_{\mathfrak{D}_2} \vDash_2 \phi(\overline{S}^n)$

[a] By 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*

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

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

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

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

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

Derivability

- We need to introduce another relation that states when a *formula* is *derivable*

Definition (Second-Order Derivability relation)

Given a *deduction system* and a set of *formulas* Γ, we can say $\Gamma \vdash_2 \phi$ (read as "*Gamma derives phi*") iff starting from Γ we can derive ϕ using the *deduction system*.
Formally, if exists a derivation

$$\frac{\Pi}{\phi}$$

s.t. $Hp[\Pi] \subseteq \Gamma$

- Same as in *First-Order Logic*

Comprehension schema (1)

- States that "Any <u>definable</u> subclass of a set is a set", formally:

Definition (Axiom schema of comprehension)

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

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

- 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

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

the following is correct

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

$\square$

- **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* $Ap_0, Ap_1, \ldots$ s.t. Ap_n is $(n+1)$-ary, since it comprehends the *symbol* of the predicate and it's *arguments*
- $Ap_n(X, t_1, \ldots, t_n)$ can be seen as $X^n(t_1, \ldots, t_n)$, 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. $\forall x, y, z.(U_i(x) \wedge U_j(y) \wedge V(z) \Rightarrow x \neq u \wedge y \neq z \wedge z \neq x)$ for all $i \neq j$
2. $\forall X, y_1, \dots, y_n.(Ap_n(X, y_1, \dots, y_n) \Rightarrow U_n(P) \wedge \bigwedge\!\!\!\bigwedge_i V(y_i))$ for $n \geq 1$
3. $U_0(C_0, V(C_{2^{i+1}}))$ for $i \geq 0$ and $U_n(C_{3^i 5^n})$ for $i, n \geq 0$
4. $\forall z_1, \dots, z_m.\exists P.($
   $$U_n(X) \wedge \forall y_1, \dots, y_n.\Big(\bigwedge\!\!\!\bigwedge V(y_i) \Rightarrow (\phi^* \Leftrightarrow Ap_n(X, y_1, \dots, y_n))\Big)$$
   $)$ 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 $\langle X, y_1, \ldots, y_n \rangle \in Ap_n$ then X is a *predicate* and $y_1, \ldots, y_n$ are it's *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:
  - $(\phi \,\Box\, \psi)^* \leftarrow \phi^* \,\Box\, \psi^*$
  - $(\neg\phi)^* \leftarrow \neg\phi^*$
  - $(\forall x_i.\phi(x_i))^* \leftarrow \forall x_i^*.(V(x_i^*) \Rightarrow \phi^*(x_i^*))$
  - $(\exists x_i.\phi(x_i))^* \leftarrow \exists x_i^*.(V(x_i^*) \wedge \phi^*(x_i^*))$
  - $(\forall X_i^n.\phi(X_i^n))^* \leftarrow \forall (X_i^n)^*.(U_n((X_i^n)^*) \Rightarrow \phi^*((X_i^n)^*))$
  - $(\exists X_i^n.\phi(X_i^n))^* \leftarrow \exists (X_i^n)^*.(U_n((X_i^n)^*) \wedge \phi^*((X_i^n)^*))$
- This further strengthens the result of *derivability*, since we can state that $\vdash \Rightarrow \vdash_2$

Model

Definition (Model)

Given a *structure* $\mathfrak{D}_2$, it is called a **model** of SOL if it admits (a.k.a is *valid*) the *comprehension schema*.
If $\mathfrak{D}_2$ 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

Soundness (1)

- Given the *derivability relation* $\vdash_2$, it is easy to prove the **Soundness** result
- Just recall that by *flattening* we just proved that we can translate each *Second-Order Formula* into a *First-Order formula*
- Since the result of derivability *holds* for SOL the same way it holds for FOL, we can state the following

Corollary

Given a formula ϕ, $\vdash \phi \Rightarrow \vdash_2 \phi$ since we never added a derivation

$$\frac{\Pi}{\bot}$$

in extending FOL Natural Deduction, so $\nvdash_2 \bot$

Soundness (2)

Theorem (Soundness)

$\Gamma \vdash_2 \phi \Rightarrow \Gamma \vDash_2 \phi$

Proof idea.

From the *corollary* seen in tre previous slide $\not\vdash_2 \bot$, so $\Gamma \vdash_2 \phi \Rightarrow \phi \neq \bot$. If Γ is formed by any *Second-Order formula*, we can say that $\Gamma \vDash_2 \phi$ since we can find a derivation

$$\frac{\Pi}{\phi}$$

s.t. $Hp[\Pi] \subseteq \Gamma$ and $Hp[\Pi] \vDash_2 \phi$. □

Completeness

- It's not as easy to prove **Completeness**