Introduction to First-Order Logic

Jean-Pierre Jouannaud
Project Formes
INRIA-LIAMA and Tsinghua University

2nd Asian-Pacific School on Formal Methods,
August 20, 2010

Outline

1. The need for a logical language
2. Syntax of FOL
3. Semantics of FOL
4. Proof theory of FOL

Outline

Outline

1. The need for a logical language
2. Syntax of FOL
3. Semantics of FOL
4. Proof theory of FOL

Outline

Outline

1. The need for a logical language
2. Syntax of FOL
3. Semantics of FOL
4. Proof theory of FOL

The need for a formal language

Consider the following C-like simple program:

```
Static integer array S[0::10];
S(0) := 0 ;
For I := 1 to 10 Do S(I) := S(I-1) + I Endo ;
Print S(10)
```

To prove that the value printed by the program is 55, we need to express the following property:

$$S(0)=0 \ \wedge \forall\ I \in [1::10] \ \ S(I) = (I \times (I+1))/2$$

This language is first-order logic.

The need for a formal language

Consider the following C-like simple program:

```
Static integer array S[0::10];
S(0) := 0 ;
For I := 1 to 10 Do S(I) := S(I-1) + I Endo ;
Print S(10)
```

To prove that the value printed by the program is 55, we need to express the following property:

$$S(0)=0 \;\; \wedge \forall \, I \in [1::10] \;\; S(I) = (I \times (I+1))/2$$

This language is first-order logic.

Outline

The first-order (unsorted) vocabulary

- a set $\mathcal{F}$ of function symbols with arities. $\mathcal{F}_n$ is the set of function symbols of arity n.
- a set $\mathcal{R}$ of predicate symbols with arities. $\mathcal{R}_n$ is the set of predicate symbols of arity n.
- a set $\mathcal{X}$ of variable symbols.
- the constants $\top, \bot$
- the connectives $\vee$, $\wedge$, $\rightarrow$ and $\neg$.
- the quantifiers $\forall, \exists$.

Remark: there are more constants and connectives than actually needed ($\bot$ and $\rightarrow$ are enough).

Syntax

- **Terms $\mathcal{T}(\mathcal{F}, \mathcal{X})$:** least set s.t. $\mathcal{F}$: set of function symbols
  - $X \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$
  - $f(t_1, ..., t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ if $f \in \mathcal{F}_n$, $t_1, .., t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$
- **Atoms $\mathcal{A}(\mathcal{R}, \mathcal{F}, \mathcal{X})$:** $\mathcal{R}$: set of predicate symbols
  $R(t_1, ..., t_n) \in \mathcal{A}(\mathcal{R}, \mathcal{F}, \mathcal{X})$ iff $R \in \mathcal{R}_n$, $t_1, .., t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$
  ($\top, \bot$ are often considered as atoms as well)
- **Literals:** atoms and their negations
- **Formulae in FOL($\mathcal{R}, \mathcal{F}, \mathcal{X}$):** least set s.t.
  $\top, \bot, A \in \mathcal{A}(\mathcal{R}, \mathcal{F}, \mathcal{X})$,
  $\neg B, B \wedge C, B \vee C, B \rightarrow C, \forall x.B, \exists x.B$,
  for formulae B, C
- **Clauses:** universally quantified disjunctions of literals
  $\forall x_1 \dots x_m.L_1 \vee \dots \vee L_n$ where all $L_i's$ are literals.
  (Convention: $\forall x_1 \dots x_n$ is usually omitted in clauses.)

Variables in the scope of a quantifier are called **bound**.
Variables not in the scope of a quantifier are called **free**.
Terms or atoms without variables are called **ground**.
Propositions without free variables are called **closed**.

Syntax

- **Terms** $\mathcal{T}(\mathcal{F}, \mathcal{X})$**:** least set s.t. $\quad$ $\mathcal{F}$: set of function symbols
  - $X \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$
  - $f(t_1, ..., t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ if $f \in \mathcal{F}_n$, $t_1, .., t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$
- **Atoms** $\mathcal{A}(\mathcal{R}, \mathcal{F}, \mathcal{X})$**:** $\quad$ $\mathcal{R}$: set of predicate symbols
  $R(t_1, ..., t_n) \in \mathcal{A}(\mathcal{R}, \mathcal{F}, \mathcal{X})$ iff $R \in \mathcal{R}_n$, $t_1, .., t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$
  ($\top, \bot$ are often considered as atoms as well)
- **Literals:** atoms and their negations
- **Formulae in FOL**$(\mathcal{R}, \mathcal{F}, \mathcal{X})$**:** least set s.t.
  $\top, \bot, A \in \mathcal{A}(\mathcal{R}, \mathcal{F}, \mathcal{X})$,
  $\neg B, B \wedge C, B \vee C, B \rightarrow C, \forall x.B, \exists x.B$,
  for formulae B, C
- **Clauses:** universally quantified disjunctions of literals
  $\forall x_1 \dots x_m . L_1 \vee \dots \vee L_n$ where all L_i's are literals.
  (Convention: $\forall x_1 \dots x_n$ is usually omitted in clauses.)

Variables in the scope of a quantifier are called **bound**.
Variables not in the scope of a quantifier are called **free**.
Terms or atoms without variables are called **ground**.
Propositions without free variables are called **closed**.

Syntax

- **Terms** $\mathcal{T}(\mathcal{F},\mathcal{X})$**:** least set s.t. $\mathcal{F}$: set of function symbols
  - $X \subseteq \mathcal{T}(\mathcal{F},\mathcal{X})$
  - $f(t_1,...,t_n) \in \mathcal{T}(\mathcal{F},\mathcal{X})$ if $f \in \mathcal{F}_n$, $t_1,..,t_n \in \mathcal{T}(\mathcal{F},\mathcal{X})$
- **Atoms** $\mathcal{A}(\mathcal{R},\mathcal{F},\mathcal{X})$**:** $\mathcal{R}$: set of predicate symbols
  $R(t_1,...,t_n) \in \mathcal{A}(\mathcal{R},\mathcal{F},\mathcal{X})$ iff $R \in \mathcal{R}_n$, $t_1,..,t_n \in \mathcal{T}(\mathcal{F},\mathcal{X})$
  ($\top,\bot$ are often considered as atoms as well)
- **Literals:** atoms and their negations
- **Formulae in FOL**$(\mathcal{R},\mathcal{F},\mathcal{X})$**:** least set s.t.
  $\top,\bot,A \in \mathcal{A}(\mathcal{R},\mathcal{F},\mathcal{X})$,
  $\neg B, B \wedge C, B \vee C, B \rightarrow C, \forall x.B, \exists x.B$,
  for formulae B, C
- **Clauses:** universally quantified disjunctions of literals
  $\forall x_1 \dots x_m.L_1 \vee \dots \vee L_n$ where all L_i's are literals.
  (Convention: $\forall x_1 \dots x_n$ is usually omitted in clauses.)

Variables in the scope of a quantifier are called **bound.**
Variables not in the scope of a quantifier are called **free.**
Terms or atoms without variables are called **ground.**
Propositions without free variables are called **closed.**

Syntax

- **Terms** $\mathcal{T}(\mathcal{F}, \mathcal{X})$**:** least set s.t. $\mathcal{F}$: set of function symbols
  - $X \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$
  - $f(t_1, ..., t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ if $f \in \mathcal{F}_n$, $t_1, .., t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$
- **Atoms** $\mathcal{A}(\mathcal{R}, \mathcal{F}, \mathcal{X})$**:** $\mathcal{R}$: set of predicate symbols
  $R(t_1, ..., t_n) \in \mathcal{A}(\mathcal{R}, \mathcal{F}, \mathcal{X})$ iff $R \in \mathcal{R}_n$, $t_1, .., t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$
  ($\top, \bot$ are often considered as atoms as well)
- **Literals:** atoms and their negations
- **Formulae in FOL(**$\mathcal{R}, \mathcal{F}, \mathcal{X}$**):** least set s.t.
  $\top, \bot, A \in \mathcal{A}(\mathcal{R}, \mathcal{F}, \mathcal{X})$,
  $\neg B, B \wedge C, B \vee C, B \rightarrow C, \forall x.B, \exists x.B$,
  for formulae B, C
- **Clauses:** universally quantified disjunctions of literals
  $\forall x_1 \dots x_m . L_1 \vee \dots \vee L_n$ where all $L_i's$ are literals.
  (Convention: $\forall x_1 \dots x_n$ is usually omitted in clauses.)

Variables in the scope of a quantifier are called **bound**.
Variables not in the scope of a quantifier are called **free**.
Terms or atoms without variables are called **ground**.
Propositions without free variables are called **closed**.

Syntax

- **Terms** $\mathcal{T}(\mathcal{F}, \mathcal{X})$**:** least set s.t. $\mathcal{F}$: set of function symbols
  - $X \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$
  - $f(t_1, ..., t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ if $f \in \mathcal{F}_n$, $t_1, .., t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$
- **Atoms** $\mathcal{A}(\mathcal{R}, \mathcal{F}, \mathcal{X})$**:** $\mathcal{R}$: set of predicate symbols
  $R(t_1, ..., t_n) \in \mathcal{A}(\mathcal{R}, \mathcal{F}, \mathcal{X})$ iff $R \in \mathcal{R}_n$, $t_1, .., t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$
  ($\top, \bot$ are often considered as atoms as well)
- **Literals:** atoms and their negations
- **Formulae in FOL(**$\mathcal{R}, \mathcal{F}, \mathcal{X}$**):** least set s.t.
  $\top, \bot, A \in \mathcal{A}(\mathcal{R}, \mathcal{F}, \mathcal{X})$,
  $\neg B, B \wedge C, B \vee C, B \rightarrow C, \forall x.B, \exists x.B$,
  for formulae B, C
- **Clauses:** universally quantified disjunctions of literals
  $\forall x_1 \dots x_m.L_1 \vee \dots \vee L_n$ where all $L_i's$ are literals.
  (Convention: $\forall x_1 \dots x_n$ is usually omitted in clauses.)

Variables in the scope of a quantifier are called **bound**.
Variables not in the scope of a quantifier are called **free**.
Terms or atoms without variables are called **ground**.
Propositions without free variables are called **closed**.

Syntax

- **Terms** $\mathcal{T}(\mathcal{F}, \mathcal{X})$**:** least set s.t. $\mathcal{F}$: set of function symbols
  - $X \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$
  - $f(t_1, ..., t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ if $f \in \mathcal{F}_n$, $t_1, .., t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$
- **Atoms** $\mathcal{A}(\mathcal{R}, \mathcal{F}, \mathcal{X})$**:** $\mathcal{R}$: set of predicate symbols
  $R(t_1, ..., t_n) \in \mathcal{A}(\mathcal{R}, \mathcal{F}, \mathcal{X})$ iff $R \in \mathcal{R}_n$, $t_1, .., t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$
  ($\top, \bot$ are often considered as atoms as well)
- **Literals:** atoms and their negations
- **Formulae in FOL(**$\mathcal{R}, \mathcal{F}, \mathcal{X}$**):** least set s.t.
  $\top, \bot, A \in \mathcal{A}(\mathcal{R}, \mathcal{F}, \mathcal{X})$,
  $\neg B, B \wedge C, B \vee C, B \rightarrow C, \forall x.B, \exists x.B$,
  for formulae B, C
- **Clauses:** universally quantified disjunctions of literals
  $\forall x_1 \ldots x_m.L_1 \vee \ldots \vee L_n$ where all $L_i's$ are literals.
  (Convention: $\forall x_1 \ldots x_n$ is usually omitted in clauses.)

Variables in the scope of a quantifier are called **bound**.
Variables not in the scope of a quantifier are called **free**.
Terms or atoms without variables are called **ground**.
Propositions without free variables are called **closed**.

Example:
$S(0) = 0 \ \wedge \ \forall I \in [1 :: 10] \ S(I) = (I \times (I+1))/2$

$\mathcal{F}_0 = \{0\}$
$\mathcal{F}_1 = \{S, +1, /2\}$
$\mathcal{F}_2 = \{\times\}$
$\mathcal{R}_1 = \{\in [1 :: 10]\}$
$\mathcal{R}_2 = \{=\}$
$\mathcal{X} = \{I\}$

The formula becomes:

$= (S(0), 0) \wedge \forall I. \in [1 :: 10](I) \rightarrow \ \ = (S(I), /2(\times(I, +1(I))))$

or, using a more liberal syntax:

$S(0) = 0 \wedge \forall I. I \in [1 :: 10] \rightarrow \ S(I) = (I \times (I+1))/2$

Alternative: $\mathcal{F}_0 = \{0, 1\}, \mathcal{F}_1 = \{S, /2\}, \mathcal{F}_2 = \{\times, +\}$, giving

$= (S(0), 0) \wedge \forall I. \in [1 :: 10](I) \rightarrow \ \ = (S(I), /2(\times(I, +(I, 1))))$

Example: $S(0) = 0 \;\wedge\; \forall I \in [1 :: 10]\; S(I) = (I \times (I + 1))/2$

$\mathcal{F}_0 = \{0\}$
$\mathcal{F}_1 = \{S, +1, /2\}$
$\mathcal{F}_2 = \{\times\}$
$\mathcal{R}_1 = \{\in [1 :: 10]\}$
$\mathcal{R}_2 = \{=\}$
$\mathcal{X} = \{I\}$

The formula becomes:

$$= (S(0), 0) \wedge \forall I. \in [1 :: 10](I) \rightarrow \;\; = (S(I), /2(\times(I, +1(I))))$$

or, using a more liberal syntax:
$S(0) = 0 \wedge \forall I. I \in [1 :: 10] \rightarrow \; S(I) = (I \times (I + 1))/2$

Alternative: $\mathcal{F}_0 = \{0, 1\}, \mathcal{F}_1 = \{S, /2\}, \mathcal{F}_2 = \{\times, +\}$, giving

$$= (S(0), 0) \wedge \forall I. \in [1 :: 10](I) \rightarrow \;\; = (S(I), /2(\times(I, +(I, 1))))$$

Example: $S(0) = 0 \ \wedge \ \forall I \in [1 :: 10] \ S(I) = (I \times (I + 1))/2$

$$\mathcal{F}_0 = \{0\}$$
$$\mathcal{F}_1 = \{S, +1, /2\}$$
$$\mathcal{F}_2 = \{\times\}$$
$$\mathcal{R}_1 = \{\in [1 :: 10]\}$$
$$\mathcal{R}_2 = \{=\}$$
$$\mathcal{X} = \{I\}$$

The formula becomes:

$$= (S(0), 0) \wedge \forall I. \in [1 :: 10](I) \rightarrow \ \ = (S(I), /2(\times(I, +1(I))))$$

or, using a more liberal syntax:

$$S(0) = 0 \wedge \forall I. I \in [1 :: 10] \rightarrow \ S(I) = (I \times (I + 1))/2$$

Alternative: $\mathcal{F}_0 = \{0, 1\}, \mathcal{F}_1 = \{S, /2\}, \mathcal{F}_2 = \{\times, +\}$, giving

$$= (S(0), 0) \wedge \forall I. \in [1 :: 10](I) \rightarrow \ \ = (S(I), /2(\times(I, +(I, 1))))$$

Example continued (with unary $+$)

$/2(\times(I, +1(I))$ is a term

$= (S(I), /2(\times(I, +1(I)))$ is an atom

$\neg(\in [1 :: 10](I))$ is a literal

$= (S(0), 0) \wedge \forall I. \in [1 :: 10](I) \rightarrow \; = (S(I), /2(\times(I, +1(I)))$ is a formula

$\forall I. \in [1 :: 10](I) \rightarrow \; = (S(I), /2(\times(I, +1(I)))$
can equivalently (explained later) be written:
$\forall I. \neg(\in [1 :: 10](I)) \; \vee \; = (S(I), /2(\times(I, +1(I)))$

which is a clause.

Example continued (with unary $+$)

$/2(\times(I, +1(I))$ is a term

$= (S(I), /2(\times(I, +1(I)))$ is an atom

$\neg(\in [1 :: 10](I))$ is a literal

$= (S(0), 0) \wedge \forall I. \in [1 :: 10](I) \rightarrow \; = (S(I), /2(\times(I, +1(I)))$
is a formula

$\forall I. \in [1 :: 10](I) \rightarrow \; = (S(I), /2(\times(I, +1(I)))$
can equivalently (explained later) be written:
$\forall I. \neg(\in [1 :: 10](I)) \;\vee\; = (S(I), /2(\times(I, +1(I)))$

which is a clause.

Example continued (with unary $+$)

$/2(\times(I, +1(I))$ is a term

$=(S(I), /2(\times(I, +1(I)))$ is an atom

$\neg(\in [1 :: 10](I))$ is a literal

$=(S(0), 0) \wedge \forall I. \in [1 :: 10](I) \rightarrow \ =(S(I), /2(\times(I, +1(I)))$ is a formula

$\forall I. \in [1 :: 10](I) \rightarrow \ =(S(I), /2(\times(I, +1(I)))$
can equivalently (explained later) be written:
$\forall I. \neg(\in [1 :: 10](I)) \ \vee \ =(S(I), /2(\times(I, +1(I)))$

which is a clause.

Example continued (with unary $+$)

$/2(\times(I, +1(I))$ is a term

$= (S(I), /2(\times(I, +1(I)))$ is an atom

$\neg(\in [1 :: 10](I))$ is a literal

$= (S(0), 0) \wedge \forall I. \in [1 :: 10](I) \rightarrow \; = (S(I), /2(\times(I, +1(I)))$
is a formula

$\forall I. \in [1 :: 10](I) \rightarrow \; = (S(I), /2(\times(I, +1(I)))$
can equivalently (explained later) be written:
$\forall I. \neg(\in [1 :: 10](I)) \; \vee \; = (S(I), /2(\times(I, +1(I)))$

which is a clause.

Example continued (with unary $+$)

$/2(\times(I, +1(I))$ is a term

$= (S(I), /2(\times(I, +1(I)))$ is an atom

$\neg(\in [1 :: 10](I))$ is a literal

$= (S(0), 0) \wedge \forall I. \in [1 :: 10](I) \rightarrow \ = (S(I), /2(\times(I, +1(I)))$
is a formula

$\forall I. \in [1 :: 10](I) \rightarrow \ = (S(I), /2(\times(I, +1(I)))$
can equivalently (explained later) be written:
$\forall I. \neg(\in [1 :: 10](I)) \ \vee \ = (S(I), /2(\times(I, +1(I)))$

which is a clause.

Important sublogics of FOL

Propositional Logic:

$\mathcal{F} = \emptyset$,
$\mathcal{R} = \mathcal{R}_0$ (propositional symbols $p, q, r \ldots$),
$\mathcal{X} = \emptyset$

Equational Logic:

$\mathcal{R} = \mathcal{R}_2 = \{=\}$

Membership equational Logic:

$\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ with $\mathcal{R}_2 = \{=\}$

Datalog (for querying data bases):

$\mathcal{F} = \mathcal{F}_0$

Outline

Syntax has no *meaning per se*, that is

In general, there are infinitely many different ways to give meaning to a given formula.

but there are two principles:

1. *compositionality:*

   The meaning of a formula can be obtained from the meaning of its parts.

2. Given a *domain of interpretation* $\mathcal{D}$, interpretations are functions from
   $\mathcal{D}^n$ to $\mathcal{D}$ for open terms with n free variables
   $\mathcal{D}^n$ to the set $\mathcal{B} := \{T, F\}$ of *truth values* for propositions with n free variables

   These functions become values in $\mathcal{D}$ or $\mathcal{B}$ when applied to *valuations*, that is n-tuples of values in $\mathcal{D}$

CAVEAT

Syntax has no *meaning per se*, that is

In general, there are infinitely many different ways to give meaning to a given formula.

but there are two principles:

1. *compositionality:*

   The meaning of a formula can be obtained from the meaning of its parts.

2. Given a *domain of interpretation* $\mathcal{D}$, interpretations are functions from
   $\mathcal{D}^n$ to $\mathcal{D}$ for open terms with n free variables
   $\mathcal{D}^n$ to the set $\mathcal{B} := \{T, F\}$ of *truth values* for propositions with n free variables

   These functions become values in $\mathcal{D}$ or $\mathcal{B}$ when applied to *valuations*, that is n-tuples of values in $\mathcal{D}$

Interpretation in a *structure* $I = (\mathcal{D}, \mathcal{F}_I, \mathcal{R}_I)$

We use $[\![A]\!]_I$ or simply $[\![A]\!]$ for the interpretation of A in I.

Interpretation of terms and atoms:

1. $[\![x]\!] =$ identity function
2. $[\![f(t_1, \ldots, t_n)]\!] = f_I([\![t_1]\!], \ldots, [\![t_n]\!])$
3. $[\![R(t_1, \ldots, t_n)]\!] = R_I([\![t_1]\!], \ldots, [\![t_n]\!])$
4. $[\![A \wedge B]\!] = [\![A]\!] \wedge_I [\![B]\!]$
   where $\wedge_I$ between interpretations is defined as:
   $([\![A]\!] \wedge_I [\![B]\!])(v) = [\![A]\!](v) \wedge_{\mathcal{B}} [\![B]\!](v)$
   reducing this case to the truth table of conjunction
5. same for $\vee$, $\rightarrow$, $\perp$, $\top$
6. $[\![\forall x.A(x_1, \ldots, x_n, x)]\!](v_n) = T$ if
   $[\![A(x_1, \ldots, x_n, x)]\!](v_n, d) = T$ for all $d \in \mathcal{D}$
7. $[\![\exists x.A(x_1, \ldots, x_n, x)]\!](v_n) = T$ if
   $[\![A(x_1, \ldots, x_n, x)]\!](v_n, d) = T$ for some $d \in \mathcal{D}$

Interpretation in a *structure* $I = (\mathcal{D}, \mathcal{F}_I, \mathcal{R}_I)$

We use $[\![A]\!]_I$ or simply $[\![A]\!]$ for the interpretation of A in I.

Interpretation of terms and atoms:

1. $[\![x]\!] =$ identity function
2. $[\![f(t_1, \ldots, t_n)]\!] = f_I([\![t_1]\!], \ldots, [\![t_n]\!])$
3. $[\![R(t_1, \ldots, t_n)]\!] = R_I([\![t_1]\!], \ldots, [\![t_n]\!])$
4. $[\![A \wedge B]\!] = [\![A]\!] \wedge_I [\![B]\!]$
   where $\wedge_I$ between interpretations is defined as:
   $([\![A]\!] \wedge_I [\![B]\!])(v) = [\![A]\!](v) \wedge_{\mathcal{B}} [\![B]\!](v)$
   reducing this case to the truth table of conjunction
5. same for $\vee$, $\rightarrow$, $\perp$, $\top$
6. $[\![\forall x.A(x_1, \ldots, x_n, x)]\!](v_n) = T$ if
   $[\![A(x_1, \ldots, x_n, x)]\!](v_n, d) = T$ for all $d \in \mathcal{D}$
7. $[\![\exists x.A(x_1, \ldots, x_n, x)]\!](v_n) = T$ if
   $[\![A(x_1, \ldots, x_n, x)]\!](v_n, d) = T$ for some $d \in \mathcal{D}$

Interpretation in a *structure* $I = (\mathcal{D}, \mathcal{F}_I, \mathcal{R}_I)$

We use $[\![A]\!]_I$ or simply $[\![A]\!]$ for the interpretation of A in I.

Interpretation of terms and atoms:

1. $[\![x]\!]$ = identity function
2. $[\![f(t_1, \ldots, t_n)]\!] = f_I([\![t_1]\!], \ldots, [\![t_n]\!])$
3. $[\![R(t_1, \ldots, t_n)]\!] = R_I([\![t_1]\!], \ldots, [\![t_n]\!])$
4. $[\![A \wedge B]\!] = [\![A]\!] \wedge_I [\![B]\!]$
   where $\wedge_I$ between interpretations is defined as:
   $([\![A]\!] \wedge_I [\![B]\!])(v) = [\![A]\!](v) \wedge_{\mathcal{B}} [\![B]\!](v)$
   reducing this case to the truth table of conjunction
5. same for $\vee$, $\rightarrow$, $\perp$, $\top$
6. $[\![\forall x.A(x_1, \ldots, x_n, x)]\!](v_n) = T$ if
   $[\![A(x_1, \ldots, x_n, x)]\!](v_n, d) = T$ for all $d \in \mathcal{D}$
7. $[\![\exists x.A(x_1, \ldots, x_n, x)]\!](v_n) = T$ if
   $[\![A(x_1, \ldots, x_n, x)]\!](v_n, d) = T$ for some $d \in \mathcal{D}$

Interpretation in a *structure* $I = (\mathcal{D}, \mathcal{F}_I, \mathcal{R}_I)$

We use $[\![A]\!]_I$ or simply $[\![A]\!]$ for the interpretation of A in I.

Interpretation of terms and atoms:

1. $[\![x]\!] =$ identity function
2. $[\![f(t_1, \ldots, t_n)]\!] = f_I([\![t_1]\!], \ldots, [\![t_n]\!])$
3. $[\![R(t_1, \ldots, t_n)]\!] = R_I([\![t_1]\!], \ldots, [\![t_n]\!])$
4. $[\![A \wedge B]\!] = [\![A]\!] \wedge_I [\![B]\!]$
   where $\wedge_I$ between interpretations is defined as:
   $([\![A]\!] \wedge_I [\![B]\!])(v) = [\![A]\!](v) \wedge_{\mathcal{B}} [\![B]\!](v)$
   reducing this case to the truth table of conjunction
5. same for $\vee$, $\rightarrow$, $\perp$, $\top$
6. $[\![\forall x.A(x_1, \ldots, x_n, x)]\!](v_n) = T$ if
   $[\![A(x_1, \ldots, x_n, x)]\!](v_n, d) = T$ for all $d \in \mathcal{D}$
7. $[\![\exists x.A(x_1, \ldots, x_n, x)]\!](v_n) = T$ if
   $[\![A(x_1, \ldots, x_n, x)]\!](v_n, d) = T$ for some $d \in \mathcal{D}$

Example: an *intended* interpretation

Let $S(0) = 0 \wedge \forall I.I \in [1 :: 10] \rightarrow \; S(I) = (I \times (I+1))/2$

$\mathcal{D} = \mathbb{N}$,
0 is interpreted by the constant 0 of $\mathbb{N}$.
S is interpreted by the function on $\mathbb{N}$ which returns the sum of all natural numbers smaller-equal than argument.
$+1$ is interpreted by the successor function.
$/2$ is interpreted by integer division by 2.
$=$ is interpreted by equality of natural numbers.
$\in [1 :: 10]$ is interpreted by membership to $[1 :: 10]$.

$[\![S(0) = 0]\!] = T$
$[\![\forall I.I \in [0 :: 10] \rightarrow \; S(I) = (I \times (I+1))/2]\!] = T$ iff
$[\![S(I)]\!](n) = [\![(I \times (I+1))/2]\!](n)$ for all values $n \in \mathbb{N}$ such that $[\![I \in [1 :: 10]]\!](n) = T$.

Therefore

$[\![S(1) = 0 \wedge \forall I.I \in [1 :: 10] \rightarrow \; S(I) = (I \times (I+1))/2]\!] = T$

Example: an *intended* interpretation

Let $S(0) = 0 \wedge \forall I.I \in [1 :: 10] \rightarrow S(I) = (I \times (I + 1))/2$

$\mathcal{D} = \mathbb{N}$,
0 is interpreted by the constant 0 of $\mathbb{N}$.
S is interpreted by the function on $\mathbb{N}$ which returns the sum of all natural numbers smaller-equal than argument.
$+1$ is interpreted by the successor function.
$/2$ is interpreted by integer division by 2.
$=$ is interpreted by equality of natural numbers.
$\in [1 :: 10]$ is interpreted by membership to $[1 :: 10]$.

$[\![S(0) = 0]\!] = T$
$[\![\forall I.I \in [0 :: 10] \rightarrow S(I) = (I \times (I + 1))/2]\!] = T$ iff $[\![S(I)]\!](n) = [\![(I \times (I + 1))/2]\!](n)$ for all values $n \in \mathbb{N}$ such that $[\![I \in [1 :: 10]]\!](n) = T$.

Therefore

$[\![S(1) = 0 \wedge \forall I.I \in [1 :: 10] \rightarrow S(I) = (I \times (I + 1))/2]\!] = T$.

Example: a *non-intended* interpretation

Let $S(0) = 0 \wedge \forall I.I \in [1 :: 10] \rightarrow\ S(I) = (I \times (I + 1))/2$

$\mathcal{D} = \mathbb{N}$,
0 is interpreted by the constants 0 of $\mathbb{N}$.
S is interpreted by the function which returns the sum of squares of natural numbers smaller-equal than argument.
$+1$ is interpreted by the successor function.
$/2$ is interpreted by the function $3n(2n + 3)/2$.
$=$ is interpreted by equality of natural numbers.
$\in [1 :: 10]$ is interpreted by membership to $[1 :: 10]$.
$[\![S(0) = 0]\!] = T$
$[\![\forall I.I \in [1 :: 10] \rightarrow\ S(I) = (I \times (I + 1))/2]\!] = T$ iff $[\![S(I)]\!](n) = [\![(I \times (I + 1))/2]\!](n)$ for all values $n \in \mathbb{N}$ such that $[\![I \in [1 :: 10]]\!](n) = T$.

Since $[\![S(1)]\!] = 1$ and $(3 \times 5)/2 = 7$, then

$[\![S(0) = 0 \wedge \forall I.I \in [1 :: 10] \rightarrow\ S(I) = (I \times (I + 1))/2]\!] = F$.

Example: a *non-intended* interpretation

Let $S(0) = 0 \land \forall I . I \in [1 :: 10] \rightarrow \; S(I) = (I \times (I + 1))/2$

$\mathcal{D} = \mathbf{N}$,
0 is interpreted by the constants 0 of $\mathbf{N}$.
S is interpreted by the function which returns the sum of squares of natural numbers smaller-equal than argument.
$+1$ is interpreted by the successor function.
$/2$ is interpreted by the function $3n(2n + 3)/2$.
$=$ is interpreted by equality of natural numbers.
$\in [1 :: 10]$ is interpreted by membership to $[1 :: 10]$.

$[\![S(0) = 0]\!] = T$
$[\![\forall I . I \in [1 :: 10] \rightarrow \; S(I) = (I \times (I + 1))/2]\!] = T$ iff $[\![S(I)]\!](n) = [\![(I \times (I + 1))/2]\!](n)$ for all values $n \in \mathbf{N}$ such that $[\![I \in [1 :: 10]]\!](n) = T$.

Since $[\![S(1)]\!] = 1$ and $(3 \times 5)/2 = 7$, then

$[\![S(0) = 0 \land \forall I . I \in [1 :: 10] \rightarrow \; S(I) = (I \times (I + 1))/2]\!] = F$.

Example: a *non-intended* interpretation continued

It must be noted, however, that
$[\![S(10)]\!] = 1 + 4 + \ldots 100 = 345$
and
$3n(2n + 3)/2 = 345$!!.

This shows that the simpler property
$S(I) = 3I(2I + 3)/2$ is satisfied by both interpretations for
$I = 10$, that is, when exiting the loop of the program !

In other word, the first formula is a better description of the program properties than the second: the choice of a particular formula for characterizing the properties of a program is a difficult question. In practice, this may follow a trial-and-errors process.

A canonical structure: the Herbrand Structure

- The Herbrand Universe is the set $\mathcal{T}(\mathcal{F})$ of ground terms, used as the domain of a Herbrand Interpretation H;
- $\mathcal{T}(\mathcal{F})$ is made into an $\mathcal{F}$-algebra by canonically associating to each symbol $f \in \mathcal{F}_n$ an operation from $\mathcal{T}(\mathcal{F})^n$ to $\mathcal{T}(\mathcal{F})$ defined as $f_{\mathrm{H}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$
- Valuations in H map variables to ground terms. They are called ground substitutions.
- The Herbrand base is the set of ground atoms.
- $\mathcal{T}(\mathcal{F})$ is made into a Herbrand structure $\mathrm{H} = (\mathcal{T}(\mathcal{F}), \mathcal{F}_{\mathrm{H}} = \{f_{\mathrm{H}} \mid f \in \mathcal{F}\}, \mathcal{R}_{\mathrm{H}} = \{R_{\mathrm{H}} \mid R \in \mathcal{R}\})$ by associating an n-ary relational symbol R_{H} to each predicate symbol $R \in \mathcal{R}_n$
- A particular Herbrand interpretation is defined by
  - interpreting $=_{\mathrm{H}}$ as syntactic equality ;
  - choosing for each relation $R_{\mathrm{H}} \neq =_{\mathrm{H}}$ a set $\{R(t_1, \ldots, t_n) \mid (t_1, \ldots, t_n) \in R_{\mathrm{H}}\}$.

Examples of important structures

Domain	Operations	Relations	Name
N	$0, S, +$	$=, \leq$	Presburger arithmetic
N	$+, *, 0, S$	$=, \leq$	Peano arithmetic
R	$+, *, 0, S$	$=, \leq$	Tarski's real arithmetic
$\{T, F\}$	$\emptyset$	$\mathcal{R}_{n \neq 0} = \emptyset$	Propositional structure
$\mathcal{T}(\mathcal{F})$	$\mathcal{F}_{\mathrm{H}}$	$\mathcal{R}_{\mathrm{H}}$	Herbrand structure
$\mathcal{T}(\mathcal{F})$	$\mathcal{F}_{\mathrm{H}}$	$\{=\}$	algebraic structure

Satisfiability and validity

- A propositional formula is *satisfiable* if there exists an interpretation (also called *assignement*) of propositional symbols (to T, F) which makes it true.

  Propositional satisfiability is NP-complete.
- A propositional formula is *valid* (also called a *tautology*) if all *assignements* of propositional symbols (to T, F) make it true.
- A closed first-order proposition A is *satisfiable* if there exists an interpretation I such that $[\![A]\!]_I = T$. We write $I \models A$ or $\models_I A$.
- A closed first-order proposition A is *valid*, written $\models A$, if $[\![A]\!]_I = T$ for all interpretation I.

  Validity in FOL is undecidable (Gödel).

The *theory* of a structure I is the set of formulae $TH(I) = \{A \text{ closed } \mid I \models A\}$.

Satisfiability and validity

- A propositional formula is *satisfiable* if there exists an interpretation (also called *assignement*) of propositional symbols (to T, F) which makes it true.

  Propositional satisfiability is NP-complete.
- A propositional formula is *valid* (also called a *tautology*) if all *assignements* of propositional symbols (to T, F) make it true.
- A closed first-order proposition A is *satisfiable* if there exists an interpretation I such that $[\![A]\!]_I = T$. We write $I \models A$ or $\models_I A$.
- A closed first-order proposition A is *valid*, written $\models A$, if $[\![A]\!]_I = T$ for all interpretation I.

  Validity in FOL is undecidable (Gödel).

The *theory* of a structure I is the set of formulae $TH(I) = \{A \text{ closed } \mid I \models A\}$.

Satisfiability and validity

- A propositional formula is *satisfiable* if there exists an interpretation (also called *assignement*) of propositional symbols (to T, F) which makes it true.

  Propositional satisfiability is NP-complete.
- A propositional formula is *valid* (also called a *tautology*) if all *assignements* of propositional symbols (to T, F) make it true.
- A closed first-order proposition A is *satisfiable* if there exists an interpretation I such that $[\![A]\!]_I = T$. We write $I \models A$ or $\models_I A$.
- A closed first-order proposition A is *valid*, written $\models A$, if $[\![A]\!]_I = T$ for all interpretation I.

  Validity in FOL is undecidable (Gödel).

  The *theory* of a structure I is the set of formulae $TH(I) = \{A \text{ closed } \mid I \models A\}$.

Examples

The propositional formula $p \rightarrow (\neg p \rightarrow q)$ is both satisfiable and valid.

The previous first-order formula
$S(0) = 0 \wedge \forall I . I \in [1 :: 10] \rightarrow \; S(I) = (I \times (I + 1))/2$
is satisfiable since we have exhibited one interpretation which makes it true, but is not valid since we have have exhibited another which falsifies it.

Exercise: is the following formula valid :
$S(0) = 0 \wedge \forall I . I \in [10 :: 10] \rightarrow \; S(I) = (I \times (I + 1))/2$?

Examples

The propositional formula $p \to (\neg p \to q)$ is both satisfiable and valid.

The previous first-order formula
$S(0) = 0 \wedge \forall I. I \in [1 :: 10] \to \ S(I) = (I \times (I+1))/2$
is satisfiable since we have exhibited one interpretation which makes it true, but is not valid since we have have exhibited another which falsifies it.

Exercise: is the following formula valid :
$S(0) = 0 \wedge \forall I. I \in [10 :: 10] \to \ S(I) = (I \times (I + 1))/2$?

Decidability

A theory is *decidable* is there is an algorithm for deciding whether a given formula is valid or not.

Presburger arithmetic is decidable
Tarski's real arithmetic is decidable
Peano arithmetic is undecidable

A theory $\mathcal{T}$ is *complete* if, given an arbitrary closed formula A, either A or $\neg A$ is valid in T.
Peano's arithmetic is incomplete [Gödel].

Problem: Given a set $\mathcal{F}$ of function symbols, describe an algorithm for deciding whether a formula of the form $s = t$, where s, t are terms, is satisfiable in the algebraic structure $\mathcal{T}(\mathcal{F})$.

Equivalence of formulae

A formula B is *entailed* by a formula A, written $A \models B$, if every model I of A is a model of B
(in other words, $[\![A]\!]_I = T$ implies $[\![B]\!]_I = T$.).

Two formulae A and B are *equivalent*, written $A \equiv B$ if they entail each other
(in other words, they have the same models).

Exercise: show that for all integer n, the formula $(A_1 \wedge \ldots \wedge A_n) \rightarrow B$, where $A_1, \ldots A_n, B$ are arbitrary atoms is equivalent to the clause $B \vee (\neg A_1) \vee \ldots \vee (\neg A_n)$.

Does the statement hold (be careful; if not, correct the statement) for arbitrary formulae $A_1, \ldots A_n, B$?

Boolean Algebra

The following equalities describe propositional equivalences:

$$
\begin{aligned}
A \vee (B \vee C) &\equiv (A \vee B) \vee C \\
A \wedge (B \wedge C) &\equiv (A \wedge B) \wedge C \\
A \vee (B \wedge C) &\equiv (A \vee B) \wedge (A \vee C) \\
A \wedge (B \vee C) &\equiv (A \wedge B) \vee (A \wedge C)
\end{aligned}
$$

$$
\begin{array}{rcl|rcl}
A \vee B & \equiv & B \vee A & A \wedge B & \equiv & B \wedge A \\
A \vee 0 & \equiv & A & A \wedge 0 & \equiv & 0 \\
A \vee 1 & \equiv & 1 & A \wedge 1 & \equiv & A \\
A \vee A & \equiv & A & A \wedge A & \equiv & A \\
\neg(A \vee B) & \equiv & (\neg A) \wedge (\neg B) & \neg(A \wedge B) & \equiv & (\neg A) \vee (\neg B) \\
\neg 1 & \equiv & 0 & \neg 0 & \equiv & 1 \\
A \vee \neg A & \equiv & 1 & A \wedge \neg A & \equiv & 0
\end{array}
$$

$$\neg\neg A \equiv A$$

$$A \rightarrow B \equiv B \vee (\neg A)$$

Properties of quantifiers

The following equalities describe the additional properties of quantifiers:

$$
\begin{array}{rcll}
\neg\forall x A & \equiv & \exists x \neg A & \\
\neg\exists x A & \equiv & \forall x \neg A & \\
(\forall x A) \wedge B & \equiv & \forall x (A \wedge B) & \text{if } x \notin \mathcal{V}ar(B) \\
(\forall x A) \vee B & \equiv & \forall x (A \vee B) & \text{if } x \notin \mathcal{V}ar(B) \\
(A \rightarrow \forall x B) & \equiv & \forall x (A \rightarrow B) & \text{if } x \notin \mathcal{V}ar(A) \\
(A \rightarrow \exists x B) & \equiv & \exists x (A \rightarrow B) & \text{if } x \notin \mathcal{V}ar(A) \\
(\forall x A \rightarrow B) & \equiv & \exists x (A \rightarrow B) & \text{if } x \notin \mathcal{V}ar(B) \\
(\exists x A \rightarrow B) & \equiv & \forall x (A \rightarrow B) & \text{if } x \notin \mathcal{V}ar(B)
\end{array}
$$

Outline

Validity in Propositional Logic

- Tautologies can be proved by *deduction* rules.
- Each deduction rule is a pair made of an *antecedent* written above a horizontal bar and a *conclusion* written below the bar.
- Antecedent and conclusion are of the form $\Gamma \vdash A$ meaning that the formula A is *provable* under assumptions in Γ by using the deduction rules.
- Γ is a set, often called *environment*.
- *Proofs* are naturally organized as a tree of deductions.
- There are many possible deduction systems for FOL and even for propositional logic. We chose one which relationship to Coq will be explained in the very last course (descriding the Curry-Howard isomorphism).

Natural deduction rules

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$$

$$\frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma \vdash C}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash A \rightarrow B}{\Gamma \vdash B}$$

$$\frac{\bot \in \Gamma}{\Gamma \vdash A} \qquad \frac{\Gamma, A \rightarrow \bot \vdash \bot}{\Gamma \vdash A}$$

$$\frac{A \in \Gamma}{\Gamma \vdash A} \qquad \frac{}{\Gamma \vdash A \vee \neg A}$$

The last inference rule is called *excluded middle law*.

Removing this rule yields *intuitionistic FOL*.

Natural deduction rules

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$$

$$\frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma \vdash C}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B} \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash A \to B}{\Gamma \vdash B}$$

$$\frac{\bot \in \Gamma}{\Gamma \vdash A} \qquad \frac{\Gamma, A \to \bot \vdash \bot}{\Gamma \vdash A}$$

$$\frac{A \in \Gamma}{\Gamma \vdash A} \qquad \frac{}{\Gamma \vdash A \vee \neg A}$$

The last inference rule is called *excluded middle law*.

Removing this rule yields *intuitionistic FOL*.

Validity in First-Order Logic

Valid propositions need the following additional set of rules for handling quantifiers:

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall x.A} \qquad \frac{\Gamma \vdash \forall x.A}{\Gamma \vdash A\{x \mapsto t\}}\ x \text{ free in } \Gamma$$

$$\frac{\Gamma \vdash A\{x \mapsto t\}}{\Gamma \vdash \exists x.A} \qquad \frac{\Gamma \vdash \exists x.A \qquad \Gamma, A \vdash B}{\Gamma \vdash B}$$

In both rules above, t is an arbitrary term

Remark the analogy between:
the rules for $\wedge$ and $\forall$

the rules for $\vee$ and $\exists$

$$\dfrac{\dfrac{\dfrac{\dfrac{A \in A, A \to B}{A, A \to B \vdash A} \qquad \dfrac{A \to B \in A, A \to B}{A, A \to B \vdash A \to B}}{A, A \to B \vdash B}}{A \vdash (A \to B) \to B}}{\vdash A \to (A \to B) \to B}$$

Adequacy of the rules with the semantics

The above rules enjoy two important properties:

Soundness: if a proposition has a natural deduction proof, then it is valid.

Completeness: if a proposition is valid, then it has a natural deduction proof.

Cut elimination

Redundancy in proofs occurs when an introduction rule is immediately followed by the corresponding elimination rule, as in:

$$\dfrac{\dfrac{\cdots}{\Gamma \vdash A} \qquad \dfrac{\dfrac{\dfrac{\dfrac{A \in \Gamma, A}{\Gamma, A \vdash A}}{\cdots}}{\Gamma, A \vdash B}}{\Gamma \vdash A \to B}}{\Gamma \vdash B}$$

replaced by

$$\dfrac{\dfrac{\cdots}{\Gamma \vdash A}}{\dfrac{\cdots}{\Gamma \vdash B}}$$

- The process of cutting out these redundancies is called *cut elimination*.
- Proving termination of cut elimination is quite difficult.

replaced by

$$\dfrac{\dfrac{\dfrac{\cdots}{\Gamma \vdash A}}{\cdots}}{\Gamma \vdash B}$$

- The process of cutting out these redundancies is called *cut elimination*.
- Proving termination of cut elimination is quite difficult.

Searching for a cut-free proof is feasible, but modern provers are usually not directly based on this paradigm.

In the propositional case, algorithms try to explore the possible models in the most efficient way.

In the first-order case, algorithms are usually based on some restriction of *resolution*, introduced by Herbrand and rediscovered by Alan Robinson, which is related to another set of deduction rules called *sequent calculus*. The study of resolution could be the subject an entire course.

End marker

LOF