Course on Proof Theory - Lecture 1

Introduction to Propositional and First-order Logic

Gianluca Curzi, Marianna Girlando

University of Birmingham

Midlands Graduate School
Nottingham, 10-14 April 2022

From logic to proof theory

From logic to proof theory

Logic has to do with the formalisation of sound reasoning.

From logic to proof theory

Logic has to do with the formalisation of sound reasoning.

No humans are immortal.
Some organisms are immortal.

From logic to proof theory

Logic has to do with the formalisation of sound reasoning.

No humans are immortal.
Some organisms are immortal.

If n is divisible by 2, then n is even.
4 is divisible by 2.

From logic to proof theory

Logic has to do with the formalisation of sound reasoning.

No humans are immortal.
Some organisms are immortal.
$\therefore$ Some organisms aren't humans.

If n is divisible by 2, then n is even.
4 is divisible by 2.

From logic to proof theory

Logic has to do with the formalisation of sound reasoning.

No humans are immortal.
Some organisms are immortal.
$\therefore$ Some organisms aren't humans.

If n is divisible by 2, then n is even.
4 is divisible by 2.
$\therefore$ 4 is even.

From logic to proof theory

Logic has to do with the formalisation of sound reasoning.

No humans are immortal.
Some organisms are immortal.
$\therefore$ Some organisms aren't humans.

$$\frac{\begin{array}{ccc} P & \mathsf{e} & M \\ S & \mathsf{i} & M \end{array}}{\begin{array}{ccc} S & \mathsf{o} & P \end{array}}$$

If n is divisible by 2, then n is even.
4 is divisible by 2.
$\therefore$ 4 is even.

$$\mathsf{mp}\ \frac{A \to B \quad A}{B}$$

From logic to proof theory

Logic has to do with the formalisation of sound reasoning.

No humans are immortal.
Some organisms are immortal.
$\therefore$ Some organisms aren't humans.

$$\frac{\begin{array}{ccc} P & \mathrm{e} & M \\ S & \mathrm{i} & M \end{array}}{\begin{array}{ccc} S & \mathrm{o} & P \end{array}}$$

If n is divisible by 2, then n is even.
4 is divisible by 2.
$\therefore$ 4 is even.

$$\text{mp}\ \frac{A \to B \quad A}{B}$$

Proof theory studies mathematical proofs as formal objects.

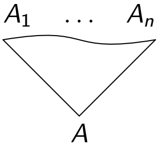
Proof systems: two fundamental results

Proof theory studies mathematical proofs as formal objects,
within proof systems.

1

Proof systems: two fundamental results

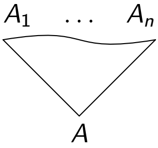
Proof theory studies mathematical proofs as formal objects,
within proof systems.



1

Proof systems: two fundamental results

Proof theory studies mathematical proofs as formal objects,
within proof systems.



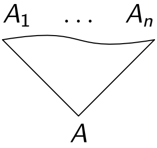
▷ Gödel's Incompleteness Theorem, 1931

We cannot prove every true sentence in certain proof systems[1].

1

Proof systems: two fundamental results

Proof theory studies mathematical proofs as formal objects, within proof systems.



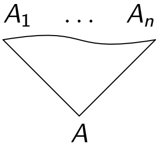
▷ Gödel's Incompleteness Theorem, 1931

We cannot prove every true sentence in certain proof systems[1].

[1] e.g. within which elementary arithmetic can be formalised.

Proof systems: two fundamental results

Proof theory studies mathematical proofs as formal objects, within proof systems.



- ▷ Gödel's Incompleteness Theorem, 1931

  We cannot prove every true sentence in certain proof systems[1].

- ▷ Gentzen's *Hauptsatz* (Cut-elimination Theorem), 1934

  Every logical theorem can be proved analytically, that is, has a proof which uses only "elements" occurring in its statement.

[1] e.g. within which elementary arithmetic can be formalised.

This course

This course

We'll introduce main results in proof theory and illustrate some relevant applications.

This course

We'll introduce main results in proof theory and illustrate some relevant applications.

1. Introduction to propositional and first-order logic
2. Soundness, completeness and other metalogical results

This course

We'll introduce main results in proof theory and illustrate some relevant applications.

1. Introduction to propositional and first-order logic
2. Soundness, completeness and other metalogical results
3. Gentzen's sequent calculus
4. A proof of the Cut-Elimination Theorem

This course

We'll introduce main results in proof theory and illustrate some relevant applications.

1. Introduction to propositional and first-order logic
2. Soundness, completeness and other metalogical results
3. Gentzen's sequent calculus
4. A proof of the Cut-Elimination Theorem
5. Beyond classical logic

Propositional logic: syntax

Propositional logic: syntax

Language

Propositional logic: syntax

Language

- Countably many propositional variables:

$$\mathsf{Var}_p = \{p, q, r, \dots\}$$

Propositional logic: syntax

Language

- Countably many propositional variables:

$$\mathsf{Var}_p = \{p, q, r, \ldots\}$$

- Propositional constants: $\bot$ (*false*), $\top$ (*true*)

Propositional logic: syntax

Language

- ▷ Countably many propositional variables:

$$\mathrm{Var}_p = \{p, q, r, \ldots\}$$

- ▷ Propositional constants: $\bot$ (*false*), $\top$ (*true*)
- ▷ Connectives: $\neg$ (*negation*), $\vee$ (*disjunction*), $\wedge$ (*conjunction*), $\rightarrow$ (*implication*)

Propositional logic: syntax

Language

- ▷ Countably many propositional variables:

$$\mathsf{Var}_p = \{p, q, r, \dots\}$$

- ▷ Propositional constants: $\bot$ (*false*), $\top$ (*true*)
- ▷ Connectives: $\neg$ (*negation*), $\vee$ (*disjunction*), $\wedge$ (*conjunction*), $\rightarrow$ (*implication*)

Formulas (Form_p) $A, B, C, \dots$ are inductively generated as follows:

Propositional logic: syntax

Language

- ▷ Countably many propositional variables:

$$\mathsf{Var}_p = \{p, q, r, \dots\}$$

- ▷ Propositional constants: $\bot$ (*false*), $\top$ (*true*)
- ▷ Connectives: $\neg$ (*negation*), $\vee$ (*disjunction*), $\wedge$ (*conjunction*), $\rightarrow$ (*implication*)

Formulas (Form_p) $A, B, C, \dots$ are inductively generated as follows:

- ▷ Propositional variables and constants are formulas

Propositional logic: syntax

Language

- ▷ Countably many propositional variables:

$$\mathrm{Var}_p = \{p, q, r, \dots\}$$

- ▷ Propositional constants: $\bot$ (*false*), $\top$ (*true*)
- ▷ Connectives: $\neg$ (*negation*), $\vee$ (*disjunction*), $\wedge$ (*conjunction*), $\rightarrow$ (*implication*)

Formulas (Form_p) $A, B, C, \dots$ are inductively generated as follows:

- ▷ Propositional variables and constants are formulas
- ▷ If A, B are formulas then $\neg A$, $A \vee B$, $A \wedge B$, $A \rightarrow B$ are formulas.

How do we interpret propositional formulas?

How do we interpret propositional formulas?

▷ Propositional assignment: assigns $\{0, 1\}$ to propositional variables

$$\alpha : \mathsf{Var}_p \rightarrow \{0, 1\}$$

How do we interpret propositional formulas?

- Propositional assignment: assigns $\{0, 1\}$ to propositional variables

$$\alpha : \mathsf{Var}_p \to \{0, 1\}$$

- Extend the assignment to formulas

How do we interpret propositional formulas?

- Propositional assignment: assigns $\{0, 1\}$ to propositional variables

$$\alpha : \mathsf{Var}_p \to \{0, 1\}$$

- Extend the assignment to formulas

A	$\neg A$
1	0
0	1

A	B	$A \wedge B$
1	1	1
1	0	0
0	1	0
0	0	0

A	B	$A \vee B$
1	1	1
1	0	1
0	1	1
0	0	0

A	B	$A \to B$
1	1	1
1	0	0
0	1	1
0	0	1

How do we interpret propositional formulas?

▷ Propositional assignment: assigns $\{0,1\}$ to propositional variables

$$\alpha : \mathsf{Var}_p \to \{0,1\}$$

▷ Extend the assignment to formulas

A	$\neg A$
1	0
0	1

A	B	$A \wedge B$
1	1	1
1	0	0
0	1	0
0	0	0

A	B	$A \vee B$
1	1	1
1	0	1
0	1	1
0	0	0

A	B	$A \to B$
1	1	1
1	0	0
0	1	1
0	0	1

Equivalently: Define $\alpha \models A$ "α satisfies A"

$$
\begin{array}{lll}
\alpha \not\models \bot & & \\
\alpha \models \top & & \\
\alpha \models p & \textit{iff} & \alpha(p) = 1 \\
\alpha \models \neg A & \textit{iff} & \alpha \not\models A \\
\alpha \models A \wedge B & \textit{iff} & \alpha \models A \text{ and } \alpha \models B \\
\alpha \models A \vee B & \textit{iff} & \alpha \models A \text{ or } \alpha \models B \\
\alpha \models A \to B & \textit{iff} & \alpha \not\models A \text{ or } \alpha \models B
\end{array}
$$

Satisfiability, semantic entailment and tautologies

$\alpha \vDash A$ "α satisfies A"

Satisfiability, semantic entailment and tautologies

$$\alpha \vDash A \qquad \text{``}\alpha \text{ satisfies } A\text{''}$$

- For $\Gamma = \{A_1, \ldots, A_n\}$ set of formulas, Γ semantically entails A if, for every propositional assignment α, whenever $\alpha \vDash A_i$ for all $A_i \in \Gamma$, then $\alpha \vDash A$. We write $\Gamma \vDash A$.

Satisfiability, semantic entailment and tautologies

$$\alpha \vDash A \qquad \text{“}\alpha \text{ satisfies } A\text{”}$$

- For $\Gamma = \{A_1, \ldots, A_n\}$ set of formulas, Γ semantically entails A if, for every propositional assignment α, whenever $\alpha \vDash A_i$ for all $A_i \in \Gamma$, then $\alpha \vDash A$. We write $\Gamma \vDash A$.

- A is a propositional tautology if $\varnothing \vDash A$. We write $\vDash A$.

Predicate logic: language

We define a predicate language $\mathcal{L}$ as follows:

Predicate logic: language

We define a predicate language $\mathcal{L}$ as follows:

- Propositional constants: $\bot$, $\top$
- Connectives $\neg, \vee, \wedge, \rightarrow$.

Predicate logic: language

We define a predicate language $\mathcal{L}$ as follows:

- Countably many variables: $\mathsf{Var} = \{x, y, z, \ldots\}$

- Propositional constants: $\bot$, $\top$
- Connectives $\neg, \vee, \wedge, \rightarrow$.

Predicate logic: language

We define a predicate language $\mathcal{L}$ as follows:

- Countably many variables: $\mathsf{Var} = \{x, y, z, \ldots\}$
- A set of function symbols: $\mathsf{Fun} = \{f, g, h, \ldots\}$

- Propositional constants: $\bot$, $\top$
- Connectives $\neg, \vee, \wedge, \rightarrow$.

Predicate logic: language

We define a predicate language $\mathcal{L}$ as follows:

- Countably many variables: $\mathsf{Var} = \{x, y, z, \dots\}$
- A set of function symbols: $\mathsf{Fun} = \{f, g, h, \dots\}$
  Each function symbol has a fixed *arity* (n of arguments it takes)
- Propositional constants: $\bot$, $\top$
- Connectives $\neg, \vee, \wedge, \rightarrow$.

Predicate logic: language

We define a predicate language $\mathcal{L}$ as follows:

- Countably many variables: $\text{Var} = \{x, y, z, \ldots\}$
- A set of function symbols: $\text{Fun} = \{f, g, h, \ldots\}$
  Each function symbol has a fixed *arity* (n of arguments it takes)
  0-ary function symbols are called constants
- Propositional constants: $\bot$, $\top$
- Connectives $\neg, \vee, \wedge, \rightarrow$.

Predicate logic: language

We define a predicate language $\mathcal{L}$ as follows:

- ▷ Countably many variables: $\mathsf{Var} = \{x, y, z, \ldots\}$
- ▷ A set of function symbols: $\mathsf{Fun} = \{f, g, h, \ldots\}$
  Each function symbol has a fixed *arity* (n of arguments it takes)
  0-ary function symbols are called constants
- ▷ A set of predicate symbols: $\mathsf{Pred} = \{P, Q, R, \ldots\}$
- ▷ Propositional constants: $\bot$, $\top$
- ▷ Connectives $\neg, \vee, \wedge, \rightarrow$.

Predicate logic: language

We define a predicate language $\mathcal{L}$ as follows:

- Countably many variables: $\mathsf{Var} = \{x, y, z, \ldots\}$
- A set of function symbols: $\mathsf{Fun} = \{f, g, h, \ldots\}$
  Each function symbol has a fixed *arity* (n of arguments it takes)
  0-ary function symbols are called constants
- A set of predicate symbols: $\mathsf{Pred} = \{P, Q, R, \ldots\}$
  Each predicate symbol has a fixed *arity* (n of arguments it takes)

- Propositional constants: $\bot$, $\top$
- Connectives $\neg, \vee, \wedge, \rightarrow$.

Predicate logic: language

We define a predicate language $\mathcal{L}$ as follows:

- ▷ Countably many variables: $\mathrm{Var} = \{x, y, z, \ldots\}$
- ▷ A set of function symbols: $\mathrm{Fun} = \{f, g, h, \ldots\}$
  Each function symbol has a fixed *arity* (n of arguments it takes)
  0-ary function symbols are called constants
- ▷ A set of predicate symbols: $\mathrm{Pred} = \{P, Q, R, \ldots\}$
  Each predicate symbol has a fixed *arity* (n of arguments it takes)
  Propositional variables are 0-ary predicates
- ▷ Propositional constants: $\bot$, $\top$
- ▷ Connectives $\neg, \vee, \wedge, \rightarrow$.

Predicate logic: language

We define a predicate language $\mathcal{L}$ as follows:

- ▷ Countably many variables: $\mathsf{Var} = \{x, y, z, \dots\}$
- ▷ A set of function symbols: $\mathsf{Fun} = \{f, g, h, \dots\}$
  Each function symbol has a fixed *arity* (n of arguments it takes)
  0-ary function symbols are called constants
- ▷ A set of predicate symbols: $\mathsf{Pred} = \{P, Q, R, \dots\}$
  Each predicate symbol has a fixed *arity* (n of arguments it takes)
  Propositional variables are 0-ary predicates
- ▷ The equality symbol $=$ (2-ary predicate)
- ▷ Propositional constants: $\bot$, $\top$
- ▷ Connectives $\neg, \vee, \wedge, \to$.

Predicate logic: language

We define a predicate language $\mathcal{L}$ as follows:

- ▷ Countably many variables: $\mathrm{Var} = \{x, y, z, \dots\}$
- ▷ A set of function symbols: $\mathrm{Fun} = \{f, g, h, \dots\}$
  Each function symbol has a fixed *arity* (n of arguments it takes)
  0-ary function symbols are called constants
- ▷ A set of predicate symbols: $\mathrm{Pred} = \{P, Q, R, \dots\}$
  Each predicate symbol has a fixed *arity* (n of arguments it takes)
  Propositional variables are 0-ary predicates
- ▷ The equality symbol $=$ (2-ary predicate)
- ▷ Propositional constants: $\bot$, $\top$
- ▷ Connectives $\neg, \vee, \wedge, \rightarrow$.
- ▷ Quantifiers: $\exists$ (*existential*) and $\forall$ (*universal*)

Predicate logic: terms

Predicate logic: terms

Terms (Ter) $s, t, u, \ldots$ are inductively generated as follows:

Predicate logic: terms

Terms (Ter) $s, t, u, \ldots$ are inductively generated as follows:

- ▷ Variables are terms

Predicate logic: terms

Terms (Ter) $s, t, u, \ldots$ are inductively generated as follows:

- ▷ Variables are terms
- ▷ If $f \in \mathsf{Fun}$ is a k-ary function symbol and $t_1, \ldots, t_k$ are terms, then the following is a term:

$$f(t_1, \ldots, t_k)$$

Predicate logic: terms

Terms (Ter) $s, t, u, \ldots$ are inductively generated as follows:

- ▷ Variables are terms
- ▷ If $f \in \text{Fun}$ is a k-ary function symbol and $t_1, \ldots, t_k$ are terms, then the following is a term:
$$f(t_1, \ldots, t_k)$$

Any constant is a term.

Predicate logic: terms

Terms (Ter) $s, t, u, \ldots$ are inductively generated as follows:

- ▷ Variables are terms
- ▷ If $f \in \mathrm{Fun}$ is a k-ary function symbol and $t_1, \ldots, t_k$ are terms, then the following is a term:

$$f(t_1, \ldots, t_k)$$

Any constant is a term.

Informally, terms denote individual entities.

Predicate logic: formulas

Predicate logic: formulas

Atomic formulas $P(t_1, \ldots, t_k)$ are inductively generated as follows:

Predicate logic: formulas

Atomic formulas $P(t_1, \ldots, t_k)$ are inductively generated as follows:

- ▷ If s, t are terms, then $s = t$ is an atomic formula.

Predicate logic: formulas

Atomic formulas $P(t_1, \ldots, t_k)$ are inductively generated as follows:

- If s, t are terms, then $s = t$ is an atomic formula.
- If P is a predicate symbol or arity k and $t_1, \ldots, t_k$ are terms, then the following is an atomic formula:

$$P(t_1, \ldots, t_k)$$

Predicate logic: formulas

Atomic formulas $P(t_1, \ldots, t_k)$ are inductively generated as follows:

- If s, t are terms, then $s = t$ is an atomic formula.
- If P is a predicate symbol or arity k and $t_1, \ldots, t_k$ are terms, then the following is an atomic formula:

$$P(t_1, \ldots, t_k)$$

Formulas (Form) $P, Q, R, \ldots$ are inductively generated as follows:

Predicate logic: formulas

Atomic formulas $P(t_1, \ldots, t_k)$ are inductively generated as follows:

- If s, t are terms, then $s = t$ is an atomic formula.
- If P is a predicate symbol or arity k and $t_1, \ldots, t_k$ are terms, then the following is an atomic formula:

$$P(t_1, \ldots, t_k)$$

Formulas (Form) $P, Q, R, \ldots$ are inductively generated as follows:

- Atomic formulas are formulas

Predicate logic: formulas

Atomic formulas $P(t_1, \dots, t_k)$ are inductively generated as follows:

- If s, t are terms, then $s = t$ is an atomic formula.
- If P is a predicate symbol or arity k and $t_1, \dots, t_k$ are terms, then the following is an atomic formula:

$$P(t_1, \dots, t_k)$$

Formulas (Form) $P, Q, R, \dots$ are inductively generated as follows:

- Atomic formulas are formulas
- $\perp$ and $\top$ are formulas

Predicate logic: formulas

Atomic formulas $P(t_1, \ldots, t_k)$ are inductively generated as follows:

- If s, t are terms, then $s = t$ is an atomic formula.
- If P is a predicate symbol or arity k and $t_1, \ldots, t_k$ are terms, then the following is an atomic formula:

$$P(t_1, \ldots, t_k)$$

Formulas (Form) $P, Q, R, \ldots$ are inductively generated as follows:

- Atomic formulas are formulas
- $\perp$ and $\top$ are formulas
- If A, B are formulas then $\neg A$, $A \vee B$, $A \wedge B$ and $A \to B$ are formulas

Predicate logic: formulas

Atomic formulas $P(t_1, \ldots, t_k)$ are inductively generated as follows:

- If s, t are terms, then $s = t$ is an atomic formula.
- If P is a predicate symbol or arity k and $t_1, \ldots, t_k$ are terms, then the following is an atomic formula:

$$P(t_1, \ldots, t_k)$$

Formulas (Form) $P, Q, R, \ldots$ are inductively generated as follows:

- Atomic formulas are formulas
- $\bot$ and $\top$ are formulas
- If A, B are formulas then $\neg A$, $A \vee B$, $A \wedge B$ and $A \to B$ are formulas
- If A is a formula then $\exists x A$ and $\forall x A$ are formulas.

Free and bound variables

$$\forall y. P(x, y)$$

Free and bound variables

$$\forall y.P(x, y)$$

x is free in P

Free and bound variables

$$\forall y. P(x, y)$$

x is free in P $\qquad$ y is bound in P

Free and bound variables

$$\forall y.P(x, y)$$

x is free in P $\qquad$ y is bound in P

Formally, we define $\mathsf{FV}(A)$ by induction over the structure of a formula A:

$$\begin{aligned}
\mathsf{FV}(x) &:= \{x\} \\
\mathsf{FV}(f(t_1, \ldots, t_k)) = \mathsf{FV}(P(t_1, \ldots, t_k)) &:= \bigcup_{i=1}^{k} \mathsf{FV}(t_i) \\
\mathsf{FV}(s = t) &:= \mathsf{FV}(s) \cup \mathsf{FV}(t) \\
\mathsf{FV}(\neg A) &:= \mathsf{FV}(A) \\
\mathsf{FV}(A \star B) &:= \mathsf{FV}(A) \cup \mathsf{FV}(B) \qquad \text{for } \star \in \{\vee, \wedge, \rightarrow\} \\
\mathsf{FV}(\forall x.A) = \mathsf{FV}(\exists x.A) &:= \mathsf{FV}(A) \backslash \{x\}
\end{aligned}$$

If $x \in \mathsf{FV}(A)$ then x is free in A. Otherwise it is bound in A.

Free and bound variables

$$\forall y.P(x, y)$$

x is free in P $\qquad$ y is bound in P

Formally, we define $\mathsf{FV}(A)$ by induction over the structure of a formula A:

$$\begin{aligned}
\mathsf{FV}(x) &:= \{x\} \\
\mathsf{FV}(f(t_1, \ldots, t_k)) = \mathsf{FV}(P(t_1, \ldots, t_k)) &:= \bigcup_{i=1}^{k} \mathsf{FV}(t_i) \\
\mathsf{FV}(s = t) &:= \mathsf{FV}(s) \cup \mathsf{FV}(t) \\
\mathsf{FV}(\neg A) &:= \mathsf{FV}(A) \\
\mathsf{FV}(A \star B) &:= \mathsf{FV}(A) \cup \mathsf{FV}(B) \qquad \text{for } \star \in \{\vee, \wedge, \rightarrow\} \\
\mathsf{FV}(\forall x.A) = \mathsf{FV}(\exists x.A) &:= \mathsf{FV}(A) \backslash \{x\}
\end{aligned}$$

If $x \in \mathsf{FV}(A)$ then x is free in A. Otherwise it is bound in A.

We write $A[t/x]$ for the operation of substituting all free occurrences of x with term t.

How do we interpret a predicate formula?

How do we interpret a predicate formula?

Propositional logic

$$\alpha \vDash A$$

How do we interpret a predicate formula?

Propositional logic

$$\alpha \vDash A$$

Predicate logic

$$\mathcal{M}, \sigma \vDash A$$

How do we interpret a predicate formula?

Propositional logic

$$\alpha \vDash A$$

Predicate logic

$$\mathcal{M}, \sigma \vDash A$$

A structure $\mathcal{M}$ for a first order language consists of:

How do we interpret a predicate formula?

Propositional logic

$$\alpha \models A$$

Predicate logic

$$\mathcal{M}, \sigma \models A$$

A structure $\mathcal{M}$ for a first order language consists of:

- ▷ A non-empty set D, called *domain*

How do we interpret a predicate formula?

Propositional logic

$$\alpha \vDash A$$

Predicate logic

$$\mathcal{M}, \sigma \vDash A$$

A structure $\mathcal{M}$ for a first order language consists of:

- A non-empty set D, called *domain*
- For each k-ary function symbol f, a *function* $f^{\mathcal{M}} : D^k \to D$

How do we interpret a predicate formula?

Propositional logic

$$\alpha \vDash A$$

Predicate logic

$$\mathcal{M}, \sigma \vDash A$$

A structure $\mathcal{M}$ for a first order language consists of:

- A non-empty set D, called *domain*
- For each k-ary function symbol f, a *function* $f^{\mathcal{M}} : D^k \to D$
  (For each constant c, an element $c^{\mathcal{M}} \in D$)

How do we interpret a predicate formula?

Propositional logic

$$\alpha \models A$$

Predicate logic

$$\mathcal{M}, \sigma \models A$$

A structure $\mathcal{M}$ for a first order language consists of:

- A non-empty set D, called *domain*
- For each k-ary function symbol f, a *function* $f^{\mathcal{M}} : D^k \to D$
  (For each constant c, an element $c^{\mathcal{M}} \in D$)
- For each k-ary predicate symbol P, a *relation* $P^{\mathcal{M}} \subseteq D^k$

How do we interpret a predicate formula?

Propositional logic

$$\alpha \vDash A$$

Predicate logic

$$\mathcal{M}, \sigma \vDash A$$

A structure $\mathcal{M}$ for a first order language consists of:

- A non-empty set D, called *domain*
- For each k-ary function symbol f, a *function* $f^{\mathcal{M}} : D^k \to D$
  (For each constant c, an element $c^{\mathcal{M}} \in D$)
- For each k-ary predicate symbol P, a *relation* $P^{\mathcal{M}} \subseteq D^k$

A variable assignment $\sigma : \mathrm{Var} \to D$.

Valuation of terms and formulas

Valuation of terms and formulas

Valuation of terms The map $[-]^{\sigma}_{\mathcal{M}} : \mathsf{Ter} \to D$ is defined as follows:

Valuation of terms and formulas

Valuation of terms The map $[-]^{\sigma}_{\mathcal{M}}$: Ter $\to D$ is defined as follows:

$$[x]^{\sigma}_{\mathcal{M}} := \sigma(x)$$

Valuation of terms and formulas

Valuation of terms The map $[-]^{\sigma}_{\mathcal{M}} : \mathrm{Ter} \to D$ is defined as follows:

$$[x]^{\sigma}_{\mathcal{M}} := \sigma(x)$$
$$[f(t_1, \ldots, t_k)]^{\sigma}_{\mathcal{M}} := f^{\mathcal{M}}([t_1]^{\sigma}_{\mathcal{M}}, \ldots, [t_k]^{\sigma}_{\mathcal{M}})$$

Valuation of terms and formulas

Valuation of terms The map $[-]^{\sigma}_{\mathcal{M}} : \mathrm{Ter} \to D$ is defined as follows:

$$
\begin{aligned}
[x]^{\sigma}_{\mathcal{M}} &:= \sigma(x) \\
[f(t_1, \ldots, t_k)]^{\sigma}_{\mathcal{M}} &:= f^{\mathcal{M}}([t_1]^{\sigma}_{\mathcal{M}}, \ldots, [t_k]^{\sigma}_{\mathcal{M}}) \\
[c]^{\sigma}_{\mathcal{M}} &:= c^{\mathcal{M}}
\end{aligned}
$$

Valuation of terms and formulas

Valuation of terms The map $[-]^{\sigma}_{\mathcal{M}} : \mathrm{Ter} \to D$ is defined as follows:

$$\begin{aligned} [x]^{\sigma}_{\mathcal{M}} &:= \sigma(x) \\ [f(t_1, \ldots, t_k)]^{\sigma}_{\mathcal{M}} &:= f^{\mathcal{M}}([t_1]^{\sigma}_{\mathcal{M}}, \ldots, [t_k]^{\sigma}_{\mathcal{M}}) \\ [c]^{\sigma}_{\mathcal{M}} &:= c^{\mathcal{M}} \end{aligned}$$

Satisfiability relation for formulas $\mathcal{M}, \sigma \vDash A$ "$\mathcal{M}, \sigma$ satisfies A"

Valuation of terms and formulas

Valuation of terms The map $[-]^{\sigma}_{\mathcal{M}}$: Ter $\to D$ is defined as follows:

$$\begin{aligned} [x]^{\sigma}_{\mathcal{M}} &:= \sigma(x) \\ [f(t_1, \ldots, t_k)]^{\sigma}_{\mathcal{M}} &:= f^{\mathcal{M}}([t_1]^{\sigma}_{\mathcal{M}}, \ldots, [t_k]^{\sigma}_{\mathcal{M}}) \\ [c]^{\sigma}_{\mathcal{M}} &:= c^{\mathcal{M}} \end{aligned}$$

Satisfiability relation for formulas $\mathcal{M}, \sigma \vDash A$ "$\mathcal{M}, \sigma$ satisfies A"

$$\mathcal{M}, \sigma \vDash P(t_1, \ldots, t_k) \quad \textit{iff} \quad \langle [t_1]^{\sigma}_{\mathcal{M}}, \ldots, [t_k]^{\sigma}_{\mathcal{M}} \rangle \in P^{\mathcal{M}}$$

Valuation of terms and formulas

Valuation of terms The map $[-]^{\sigma}_{\mathcal{M}} : \mathsf{Ter} \to D$ is defined as follows:

$$
\begin{aligned}
[x]^{\sigma}_{\mathcal{M}} &:= \sigma(x) \\
[f(t_1, \ldots, t_k)]^{\sigma}_{\mathcal{M}} &:= f^{\mathcal{M}}([t_1]^{\sigma}_{\mathcal{M}}, \ldots, [t_k]^{\sigma}_{\mathcal{M}}) \\
[c]^{\sigma}_{\mathcal{M}} &:= c^{\mathcal{M}}
\end{aligned}
$$

Satisfiability relation for formulas $\mathcal{M}, \sigma \models A$ "$\mathcal{M}, \sigma$ satisfies A"

$$
\begin{array}{rcl}
\mathcal{M}, \sigma \models P(t_1, \ldots, t_k) & \textit{iff} & \langle [t_1]^{\sigma}_{\mathcal{M}}, \ldots, [t_k]^{\sigma}_{\mathcal{M}} \rangle \in P^{\mathcal{M}} \\
\mathcal{M}, \sigma \models \neg A & \textit{iff} & \mathcal{M}, \sigma \not\models A \\
\mathcal{M}, \sigma \models A \wedge B & \textit{iff} & \mathcal{M}, \sigma \models A \text{ and } \mathcal{M}, \sigma \models B \\
\mathcal{M}, \sigma \models A \vee B & \textit{iff} & \mathcal{M}, \sigma \models A \text{ or } \mathcal{M}, \sigma \models B \\
\mathcal{M}, \sigma \models A \to B & \textit{iff} & \mathcal{M}, \sigma \not\models A \text{ or } \mathcal{M}, \sigma \models B
\end{array}
$$

Valuation of terms and formulas

Valuation of terms The map $[-]^{\sigma}_{\mathcal{M}} : \mathsf{Ter} \to D$ is defined as follows:

$$
\begin{aligned}
[x]^{\sigma}_{\mathcal{M}} &:= \sigma(x) \\
[f(t_1, \ldots, t_k)]^{\sigma}_{\mathcal{M}} &:= f^{\mathcal{M}}([t_1]^{\sigma}_{\mathcal{M}}, \ldots, [t_k]^{\sigma}_{\mathcal{M}}) \\
[c]^{\sigma}_{\mathcal{M}} &:= c^{\mathcal{M}}
\end{aligned}
$$

Satisfiability relation for formulas $\mathcal{M}, \sigma \vDash A$ "$\mathcal{M}, \sigma$ satisfies A"

$$
\begin{array}{rll}
\mathcal{M}, \sigma \vDash P(t_1, \ldots, t_k) & \textit{iff} & \langle [t_1]^{\sigma}_{\mathcal{M}}, \ldots, [t_k]^{\sigma}_{\mathcal{M}} \rangle \in P^{\mathcal{M}} \\
\mathcal{M}, \sigma \vDash \neg A & \textit{iff} & \mathcal{M}, \sigma \nvDash A \\
\mathcal{M}, \sigma \vDash A \wedge B & \textit{iff} & \mathcal{M}, \sigma \vDash A \text{ and } \mathcal{M}, \sigma \vDash B \\
\mathcal{M}, \sigma \vDash A \vee B & \textit{iff} & \mathcal{M}, \sigma \vDash A \text{ or } \mathcal{M}, \sigma \vDash B \\
\mathcal{M}, \sigma \vDash A \to B & \textit{iff} & \mathcal{M}, \sigma \nvDash A \text{ or } \mathcal{M}, \sigma \vDash B \\
\mathcal{M}, \sigma \vDash \exists x.A & \textit{iff} & \mathcal{M}, \sigma[x \mapsto d] \vDash A, \ \textbf{for some } d \in D \\
\mathcal{M}, \sigma \vDash \forall x.A & \textit{iff} & \mathcal{M}, \sigma[x \mapsto d] \vDash A, \ \textbf{for all } d \in D
\end{array}
$$

Valuation of terms and formulas

Valuation of terms The map $[-]^\sigma_\mathcal{M} : \mathrm{Ter} \to D$ is defined as follows:

$$
\begin{aligned}
[x]^\sigma_\mathcal{M} &:= \sigma(x) \\
[f(t_1, \ldots, t_k)]^\sigma_\mathcal{M} &:= f^\mathcal{M}([t_1]^\sigma_\mathcal{M}, \ldots, [t_k]^\sigma_\mathcal{M}) \\
[c]^\sigma_\mathcal{M} &:= c^\mathcal{M}
\end{aligned}
$$

Satisfiability relation for formulas $\mathcal{M}, \sigma \vDash A$ "$\mathcal{M}, \sigma$ satisfies A"

$$
\begin{aligned}
\mathcal{M}, \sigma \vDash P(t_1, \ldots, t_k) \quad & \textit{iff} \quad \langle [t_1]^\sigma_\mathcal{M}, \ldots, [t_k]^\sigma_\mathcal{M} \rangle \in P^\mathcal{M} \\
\mathcal{M}, \sigma \vDash \neg A \quad & \textit{iff} \quad \mathcal{M}, \sigma \nvDash A \\
\mathcal{M}, \sigma \vDash A \wedge B \quad & \textit{iff} \quad \mathcal{M}, \sigma \vDash A \text{ and } \mathcal{M}, \sigma \vDash B \\
\mathcal{M}, \sigma \vDash A \vee B \quad & \textit{iff} \quad \mathcal{M}, \sigma \vDash A \text{ or } \mathcal{M}, \sigma \vDash B \\
\mathcal{M}, \sigma \vDash A \to B \quad & \textit{iff} \quad \mathcal{M}, \sigma \nvDash A \text{ or } \mathcal{M}, \sigma \vDash B \\
\mathcal{M}, \sigma \vDash \exists x.A \quad & \textit{iff} \quad \mathcal{M}, \sigma[x \mapsto d] \vDash A, \ \textbf{for some } d \in D \\
\mathcal{M}, \sigma \vDash \forall x.A \quad & \textit{iff} \quad \mathcal{M}, \sigma[x \mapsto d] \vDash A, \ \textbf{for all } d \in D
\end{aligned}
$$

$\sigma[x \mapsto d]$ denotes the assignment which agrees with σ for all variables which are different from x, and maps x to d.

Satisfiability, semantic entailment, validity

$\mathcal{M}, \sigma \vDash A$ "$\mathcal{M}, \sigma$ satisfies A"

Satisfiability, semantic entailment, validity

$$\mathcal{M}, \sigma \vDash A \qquad \text{“}\mathcal{M}, \sigma \text{ satisfies } A\text{”}$$

- A structure $\mathcal{M}$ models a formula A if $\mathcal{M}, \sigma \vDash A$ for **every** variable assignment σ. We write $\mathcal{M} \vDash A$.

Satisfiability, semantic entailment, validity

$$\mathcal{M}, \sigma \vDash A \qquad \text{“}\mathcal{M}, \sigma \text{ satisfies } A\text{”}$$

- ▷ A structure $\mathcal{M}$ models a formula A if $\mathcal{M}, \sigma \vDash A$ for **every** variable assignment σ. We write $\mathcal{M} \vDash A$.
- ▷ For $\Gamma = \{A_1, \ldots, A_n\}$ is a set of formulas, Γ semantically entails A if, for any structure $\mathcal{M}$ such that $\mathcal{M} \vDash A_i$ for all $A_i \in \Gamma$, then $\mathcal{M} \vDash A$. We write $\Gamma \vDash A$.

Satisfiability, semantic entailment, validity

$$\mathcal{M}, \sigma \vDash A \qquad \text{“}\mathcal{M}, \sigma \text{ satisfies } A\text{”}$$

- A structure $\mathcal{M}$ models a formula A if $\mathcal{M}, \sigma \vDash A$ for **every** variable assignment σ. We write $\mathcal{M} \vDash A$.
- For $\Gamma = \{A_1, \ldots, A_n\}$ is a set of formulas, Γ semantically entails A if, for any structure $\mathcal{M}$ such that $\mathcal{M} \vDash A_i$ for all $A_i \in \Gamma$, then $\mathcal{M} \vDash A$. We write $\Gamma \vDash A$.
- A sentence A is valid if $\varnothing \vDash A$. We write $\vDash A$.

What are proof systems?

- Deduction (or derivation) of a statement A from a (possibly empty) set of hypothesis (or assumptions) $\Gamma = \{A_1, \ldots, A_n\}$ can be seen as an argumentation construed by applying a series of "elementary" inference steps.
- Deduction gives us evidence that A follows "logically" from Γ.

What are proof systems?

- Deduction (or derivation) of a statement A from a (possibly empty) set of hypothesis (or assumptions) $\Gamma = \{A_1, \dots, A_n\}$ can be seen as an argumentation construed by applying a series of "elementary" inference steps.
- Deduction gives us evidence that A follows "logically" from Γ.
- Proof systems: formal (syntactic) notion of deduction $\Gamma \vdash A$ to model logical (and mathematical) reasoning.

What are proof systems?

- Deduction (or derivation) of a statement A from a (possibly empty) set of hypothesis (or assumptions) $\Gamma = \{A_1, \ldots, A_n\}$ can be seen as an argumentation construed by applying a series of "elementary" inference steps.
- Deduction gives us evidence that A follows "logically" from Γ.
- Proof systems: formal (syntactic) notion of deduction $\Gamma \vdash A$ to model logical (and mathematical) reasoning.
- The formal deducibility relation $\vdash$ should satisfy two desiderata:
  - Soundness: $\Gamma \vdash A \implies \Gamma \vDash A$
  - Completeness: $\Gamma \vDash A \implies \Gamma \vdash A$
- Soundness prevents fallacious reasoning, completeness guarantees that $\vdash$ is sufficiently powerful to model any valid reasoning.

Two approaches to proof systems

Hilbert-Frege proof systems:

- *Axiomatic approach*: logical laws are expressed as axioms, a few inference rules
- Makes easier to establish meta-properties (easy to prove things about the system) but ad hoc and artificial notion of deduction (hard to prove things within the system).

Two approaches to proof systems

Hilbert-Frege proof systems:

- *Axiomatic approach*: logical laws are expressed as axioms, a few inference rules
- Makes easier to establish meta-properties (easy to prove things about the system) but ad hoc and artificial notion of deduction (hard to prove things within the system).

Gentzen proof systems:

- *Inferential approach*: only inference rules (e.g. natural deduction, sequent calculus)
- Models logical and mathematical reasoning in a natural way (easy to prove things within the system) but less suitable for establishing meta-properties (hard to prove things about the system)

A simpler language

Semantic equivalences:

- $\mathcal{M}, \sigma \models \neg A \iff \mathcal{M}, \sigma \models A \to \bot$
- $\mathcal{M}, \sigma \models A \lor B \iff \mathcal{M}, \sigma \models \neg A \to B$
- $\mathcal{M}, \sigma \models A \land B \iff \mathcal{M}, \sigma \models \neg(\neg A \lor \neg B)$
- $\mathcal{M}, \sigma \models \top \iff \mathcal{M}, \sigma \models \neg\bot$
- $\mathcal{M}, \sigma \models \exists x.A \iff \mathcal{M}, \sigma \models \neg\forall x \neg A$ (Exercise)

Example:

$$
\begin{aligned}
\mathcal{M}, \sigma \models \neg(\neg A \lor \neg B) &\iff \mathcal{M}, \sigma \not\models \neg A \lor \neg B \\
&\iff \mathcal{M}, \sigma \not\models \neg A \quad \textbf{and} \quad \mathcal{M}, \sigma \not\models \neg B \\
&\iff \mathcal{M}, \sigma \models A \quad \textbf{and} \quad \mathcal{M}, \sigma \models B
\end{aligned}
$$

Language of predicate logic over the logical basis $\{\bot, \to, \forall\}$, using the abbreviation $\neg A := A \to \bot$

Hilbert-Frege proof system for predicate logic

Axioms of HF:

- HF1. $A \to (B \to A)$
- HF2. $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$
- HF3. $\neg\neg A \to A$
- HF4. $\forall x.A \to A[t/x]$
- HF5. $(\forall x.(A \to B)) \to (A \to \forall x.B)$ where $x \notin \mathrm{FV}(A)$
- HF6. $\forall x.(x = x)$
- HF7. $\forall x.\forall y.(x = y \to (A[x/z] \to A[y/z]))$

Inference rules of HF:

$$\text{mp}\ \frac{A \quad A \to B}{B} \qquad\qquad \text{gen}\ \frac{A}{\forall x.A}$$

Condition on gen: x **not free** in the assumptions A depends on.

Proofs and derivations

A derivation of A from Γ is a list of formulae $(A_1, \ldots, A_n)$ with $A_n = A$ (called the conclusion) such that each member of the sequence is either:

- an axiom;
- a hypothesis i.e. an element of Γ;
- obtained from previous formulae by an inference rule.

We write $\Gamma \vdash A$ if there exists a derivation of A from Γ.

A proof is a derivation from an empty set of premises. We write $\vdash A$ if there is a proof of A ("A is a theorem").

Proofs and derivations

A derivation of A from Γ is a list of formulae $(A_1, \ldots, A_n)$ with $A_n = A$ (called the conclusion) such that each member of the sequence is either:

- an axiom;
- a hypothesis i.e. an element of Γ;
- obtained from previous formulae by an inference rule.

We write $\Gamma \vdash A$ if there exists a derivation of A from Γ.

A proof is a derivation from an empty set of premises. We write $\vdash A$ if there is a proof of A ("A is a theorem").

Deduction theorem (Herbrand): $\vdash$ and $\rightarrow$ are "equivalent"

$$\Gamma \cup \{A\} \vdash B \iff \Gamma \vdash A \rightarrow B$$

NB: It allows us to move from derivations to proofs and vice versa

Propositional examples

Example: A proof of $\vdash A \to A$

1.	$A \to (B \to A) \to A$	HF1
2.	$(A \to (B \to A) \to A) \to (A \to (B \to A)) \to A \to A$	HF2
3.	$(A \to B \to A) \to A \to A$	mp(1, 2)
4.	$A \to B \to A$	HF1
5.	$A \to A$	mp(4, 3)

Propositional examples

Example: A proof of $\vdash A \to A$

1.	$A \to (B \to A) \to A$	HF1
2.	$(A \to (B \to A) \to A) \to (A \to (B \to A)) \to A \to A$	HF2
3.	$(A \to B \to A) \to A \to A$	mp$(1, 2)$
4.	$A \to B \to A$	HF1
5.	$A \to A$	mp$(4, 3)$

Example: A derivation of $\{A \to B, B \to C\} \vdash A \to C$

1.	$B \to C$	hyp
2.	$(B \to C) \to A \to (B \to C)$	HF1
3.	$A \to (B \to C)$	mp$(1, 2)$
4.	$(A \to (B \to C)) \to (A \to B) \to (A \to C)$	HF2
5.	$(A \to B) \to (A \to C)$	mp(3,4)
6.	$A \to B$	hyp
7.	$A \to C$	mp$(6, 5)$

An example with quantifiers

Example: A derivation of $\{\forall x.(A \rightarrow B), \forall x.A\} \vdash \forall x.B$

1.	$\forall x.(A \rightarrow B)$	hyp
2.	$\forall x.(A \rightarrow B) \rightarrow A \rightarrow B$	HF4
3.	$A \rightarrow B$	$\text{mp}(1,2)$
4.	$\forall x.A$	hyp
5.	$\forall x.A \rightarrow A$	HF4
6.	A	$\text{mp}(4,5)$
7.	B	$\text{mp}(3,6)$
8.	$\forall x.B$	$\text{gen}(7)$

An example with quantifiers

Example: A derivation of $\{\forall x.(A \to B), \forall x.A\} \vdash \forall x.B$

1.	$\forall x.(A \to B)$	hyp
2.	$\forall x.(A \to B) \to A \to B$	HF4
3.	$A \to B$	mp(1, 2)
4.	$\forall x.A$	hyp
5.	$\forall x.A \to A$	HF4
6.	A	mp(4, 5)
7.	B	mp(3, 6)
8.	$\forall x.B$	gen(7)

NB: $\{\forall x.(A \to B), A\} \vdash \forall x.B$ does **not** hold

1.	$\forall x.(A \to B)$	hyp
2.	$\forall x.(A \to B) \to A \to B$	HF4
3.	$A \to B$	mp(1, 2)
4.	A	hyp
5.	B	mp(3, 6)
6.	$\forall x.B$	???

Theories and models

A (first-order) theory T consists of:

- A language of predicate logic $\mathcal{L}_\mathsf{T}$
- A set of formulas Γ_T in the language $\mathcal{L}_\mathsf{T}$ called (non-logical) axioms.

Theories and models

A (first-order) theory T consists of:

- $\triangleright$ A language of predicate logic $\mathcal{L}_\mathsf{T}$
- $\triangleright$ A set of formulas Γ_T in the language $\mathcal{L}_\mathsf{T}$ called (non-logical) axioms.

A model of a theory T is a structure (for $\mathcal{L}_\mathsf{T}$) $\mathcal{M}$ such that $\mathcal{M} \vDash A$ for all $A \in \Gamma_\mathsf{T}$. We write $\mathcal{M} \vDash \Gamma_\mathsf{T}$ in this case.

Theories and models

A (first-order) theory T consists of:

- A language of predicate logic $\mathcal{L}_\mathsf{T}$
- A set of formulas Γ_T in the language $\mathcal{L}_\mathsf{T}$ called (non-logical) axioms.

A model of a theory T is a structure (for $\mathcal{L}_\mathsf{T}$) $\mathcal{M}$ such that $\mathcal{M} \vDash A$ for all $A \in \Gamma_\mathsf{T}$. We write $\mathcal{M} \vDash \Gamma_\mathsf{T}$ in this case.

Example The theory G

- $\mathcal{L}_\mathsf{G}$ = constant e, unary function symbol i, binary function symbol $\circ$.
- The axioms are the following:

$$\forall x.(x \circ e = x)$$

$$\forall x.(x \circ i(x) = e)$$

$$\forall x.\forall y.\forall z.(x \circ (y \circ z) = (x \circ y) \circ z)$$

Theories and models

A (first-order) theory T consists of:

- A language of predicate logic $\mathcal{L}_\mathsf{T}$
- A set of formulas Γ_T in the language $\mathcal{L}_\mathsf{T}$ called (non-logical) axioms.

A model of a theory T is a structure (for $\mathcal{L}_\mathsf{T}$) $\mathcal{M}$ such that $\mathcal{M} \vDash A$ for all $A \in \Gamma_\mathsf{T}$. We write $\mathcal{M} \vDash \Gamma_\mathsf{T}$ in this case.

Example The theory G

- $\mathcal{L}_\mathsf{G}$ = constant e, unary function symbol i, binary function symbol $\circ$.
- The axioms are the following:

$$\forall x.(x \circ e = x)$$

$$\forall x.(x \circ i(x) = e)$$

$$\forall x.\forall y.\forall z.(x \circ (y \circ z) = (x \circ y) \circ z)$$

NB: $\mathcal{M}$ is a model of G precisely when $\mathcal{M}$ is a group!

The theory of Peano arithmetic

Peano Arithmetic (PA) = first-order theory of natural numbers and arithmetical operations.

The theory PA. $\mathcal{L}_{\mathrm{PA}}$ = constant 0 (zero), unary function symbol s (successor), binary function symbols $+$ (addition) and $\cdot$ (multiplication).

The theory of Peano arithmetic

Peano Arithmetic (PA) = first-order theory of natural numbers and arithmetical operations.

The theory PA. $\mathcal{L}_{\mathrm{PA}}$ = constant 0 (zero), unary function symbol s (successor), binary function symbols $+$ (addition) and $\cdot$ (multiplication).

- PA1. $\forall x.\neg(0 = s(x))$
- PA2. $\forall x.\forall y.(s(x) = s(y) \to x = y)$
- PA3. $\forall x.(x + 0 = x)$
- PA4. $\forall x.\forall y.(x + s(y) = s(x + y))$
- PA5. $\forall x.(x \cdot 0 = 0)$
- PA6. $\forall x.\forall y.(x \cdot s(y) = (x \cdot y) + x)$
- Induction scheme: $A[0/x] \to \forall x.(A \to A[s(x)/x]) \to \forall x.A$

NB: Induction scheme = infinite axioms, one for each $A \in \mathcal{L}_{\mathrm{PA}}$.

Proving $\forall x.(0 + x = x)$

Induction scheme with $A :\equiv 0 + x = x$:

$$\underbrace{0 + 0 = 0}_{A[0/x]} \rightarrow \forall x(\underbrace{0 + x = x}_{A} \rightarrow \underbrace{0 + s(x) = s(x)}_{A[s(x)/x]}) \rightarrow \forall x(\underbrace{0 + x = x}_{A})$$

Goal: show $\vdash A[0/x]$ and $\vdash \forall x.(A \rightarrow A[s(x)/x])$.

1.	$\forall x.(x + 0 = x)$	PA3
2.	$\forall x.(x + 0 = x) \rightarrow 0 + 0 = 0$	HF4
3.	$\underbrace{0 + 0 = 0}_{A[0/x]}$	mp$(1, 2)$

Proving $\forall x.(0 + x = x)$ continued

1.	$\underbrace{B[x/z]}_{s(x)=s(x)} \to (0 + x = x) \to B[0 + x/z]$	HF7, $B := s(z) = s(x)$
2.	$s(x) = s(x)$	HF4, HF7
3.	$(0 + x = x) \to \underbrace{B[0 + x/z]}_{s(0+x)=s(x)}$	mp(1, 2)
4.	$\underbrace{C[0 + x/z]}_{0+s(x)=s(0+x)} \to (s(0 + x) = s(x)) \to C[x/z]$	HF6, $C := 0 + s(x) = s(z)$
5.	$0 + s(x) = s(0 + x)$	PA4
6.	$s(0 + x) = s(x) \to \underbrace{C[x/z]}_{0+s(x)=s(x)}$	mp(4, 5)
7.	$\underbrace{(0 + x = x)}_{A} \to \underbrace{0 + s(x) = s(x)}_{A[s(x)/x]}$	transitivity of $\to$ (3,4)
8.	$\forall x.(A \to A[s(x)/x])$	gen(7)

Models of PA

Standard model of PA. $\mathcal{N}$ defined by:

- $D := \mathbb{N}$
- $0^{\mathcal{N}} := 0 \in \mathbb{N}$
- $s^{\mathcal{N}} := n \mapsto n + 1$
- $+^{\mathcal{N}} := (n, m) \mapsto n + m$
- $\cdot^{\mathcal{N}} := (n, m) \mapsto n \cdot m$

Theorem $\mathcal{N} \vDash \mathrm{PA}$

Models of PA

Standard model of PA. $\mathcal{N}$ defined by:

- $D := \mathbb{N}$
- $0^{\mathcal{N}} := 0 \in \mathbb{N}$
- $s^{\mathcal{N}} := n \mapsto n + 1$
- $+^{\mathcal{N}} := (n, m) \mapsto n + m$
- $\cdot^{\mathcal{N}} := (n, m) \mapsto n \cdot m$

Theorem $\mathcal{N} \models \mathrm{PA}$

Question: Is any model of PA **isomorphic**? Does PA "capture" structure of natural numbers?

Models of PA

Standard model of PA. $\mathcal{N}$ defined by:

- $D := \mathbb{N}$
- $0^{\mathcal{N}} := 0 \in \mathbb{N}$
- $s^{\mathcal{N}} := n \mapsto n + 1$
- $+^{\mathcal{N}} := (n, m) \mapsto n + m$
- $\cdot^{\mathcal{N}} := (n, m) \mapsto n \cdot m$

Theorem $\mathcal{N} \models \mathrm{PA}$

Question: Is any model of PA **isomorphic**? Does PA "capture" structure of natural numbers?

Answer: No, unfortunately! We can find non-standard models of PA:

- As application of the compactness theorem for first-order logic
- As application of Gödel's incompleteness theorem for PA

References

- ▷ Buss, S. R., editor (1998). *Handbook of Proof Theory*, volume 137 of Studies in Logic and the Foundations of Mathematics. Elsevier.
- ▷ Smullyan, R. M. (1968). *First-Order Logic*. Springer-Verlag.
- ▷ Troelstra, A. S. and Schwichtenberg, H. (1996). *Basic Proof Theory*. Cambridge University Press, New York, NY, USA.

Course on Proof Theory - Lecture 1

Exercises - Introduction to Propositional and First-order Logic

Gianluca Curzi, Marianna Girlando

University of Birmingham

Midlands Graduate School
Nottingham, 10-14 April 2022

Part 1: Propositional logic

1. Show $\vdash \bot \to A$ (*ex falso quodlibet*)

2. Show $\vdash (A \to (B \to C)) \to B \to A \to C$ (*exchange*).

3. We can extend HF to include conjunction $\wedge$ by adding the following axioms:

   $$\begin{array}{rl} \text{HF8.} & A \to B \to (A \wedge B) \\ \text{HF9.} & (A \wedge B) \to A \\ \text{HF10.} & (A \wedge B) \to B \end{array}$$

   Show $\vdash (A \to (B \to C)) \leftrightarrow ((A \wedge B) \to C)$, where

   $$A \leftrightarrow B := (A \to B) \wedge (B \to A)$$

4. We haven't yet used the axiom (*neg*)! Show that HF proves:

   $$\begin{array}{ll} (a) & \neg A \to (A \to B) \\ (b) & (\neg B \to \neg A) \to (A \to B) \\ (c) & ((A \to B) \to A) \to A \end{array}$$

Hint for part 1: Use the deduction theorem to reduce any proof of $A \to B$ to a proof of B with hypothesis A.

Part 2: Predicate logic

1. We can extend HF to include existential quantifier $\exists$ by adding the following axioms:

$$
\begin{array}{lll}
\text{HF11.} & A[t/x] \to \exists x.A & \\
\text{HF12.} & \forall x.(A \to B) \to (\exists x.A \to B) & x \notin FV(B)
\end{array}
$$

   Show the following equivalences:

$$
\begin{array}{lll}
(a) & \mathcal{M}, \sigma \vDash \exists x.A \iff \mathcal{M}, \sigma \vDash \neg\forall x.\neg A & \\
(b) & \mathcal{M}, \sigma \vDash (\exists x.A \to B) \iff \mathcal{M}, \sigma \vDash \forall x.(A \to B) & x \notin FV(B) \\
(c) & \vdash (\exists x.A \to B) \leftrightarrow \forall x.(A \to B) & x \notin FV(B)
\end{array}
$$

2. Outline a first-order theory whose models are the partial orders. Adapt this theory to characterise
   - total orders e.g. $(\mathbb{Z}, \leq)$
   - total orders with a minimum element e.g. $(\mathbb{N}, \leq, 0)$
   - dense total orders e.g. $(\mathbb{Q}, \leq)$
   - Is it possible to characterise well-founded partial orders?