Lecture 8

First-order logic

Syntax and semantics

Dr Christoph Haase
University of Oxford
(with small changes by Javier Esparza)

Limitations of propositional logic

- Can only reason about true or false
- Atomic formulas have no internal structure
- Impossible to express “real” mathematical statements

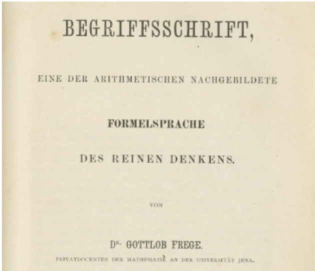
Limitations of propositional logic

- Can only reason about true or false
- Atomic formulas have no internal structure
- Impossible to express “real” mathematical statements

Example

Every natural number x is either odd or even.

Frege's Begriffsschrift


BEGRIFFSSCHRIFT,

EINE DER ARITHMETISCHEN NACHGEBILDETE

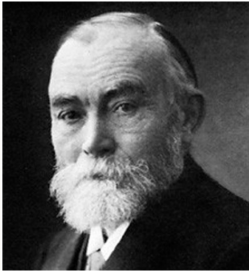
FORMELSPRACHE

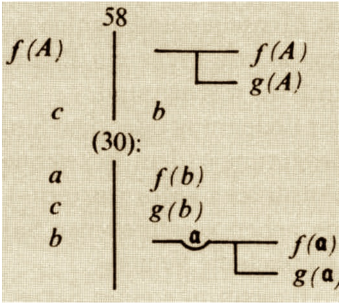
DES REINEN DENKENS.

VON

DR. GOTTLOB FREGE,

PRIVATDOCENTEN DER MATHEMATIK AN DER UNIVERSITÄT JENA.





Signatures

Definition

A **signature** σ is a tuple consisting of

- a set of **constant symbols** (denoted c, d)
- a set of **function symbols** (denoted f, g), and
- a set of **predicate symbols** (denoted P, Q, R).

Each function and predicate symbol has an **arity** $k \geq 1$.

Signatures

Definition

A **signature** σ is a tuple consisting of

- a set of **constant symbols** (denoted c, d)
- a set of **function symbols** (denoted f, g), and
- a set of **predicate symbols** (denoted P, Q, R).

Each function and predicate symbol has an **arity** $k \geq 1$.

Example

The signature of number theory is $\sigma = \langle 0, 1, +, \cdot, = \rangle$, where 0 and 1 are constant symbols, $+$ and $\cdot$ are function symbols of arity two, and $=$ is a predicate symbol of arity two.

Terms

Definition

Let $\mathcal{X}$ be a countably infinite set of **variables** (denoted x, y, z. The **terms** over a signature σ are defined by structural induction:

- Each variable $x \in \mathcal{X}$ is a term.
- Each constant symbol c is a term.
- If $t_1, \ldots, t_k$ are terms and f is a k-ary function symbol then $f(t_1, \ldots, t_k)$ is a term.

Terms

Definition

Let $\mathcal{X}$ be a countably infinite set of **variables** (denoted x, y, z. The **terms** over a signature σ are defined by structural induction:

- Each variable $x \in \mathcal{X}$ is a term.
- Each constant symbol c is a term.
- If $t_1, \ldots, t_k$ are terms and f is a k-ary function symbol then $f(t_1, \ldots, t_k)$ is a term.

Example

Given the signature of number theory σ, we have that $\cdot(+(1, 1), x)$ is a term.

Terms

Definition

Let $\mathcal{X}$ be a countably infinite set of **variables** (denoted x, y, z. The **terms** over a signature σ are defined by structural induction:

- Each variable $x \in \mathcal{X}$ is a term.
- Each constant symbol c is a term.
- If $t_1, \dots, t_k$ are terms and f is a k-ary function symbol then $f(t_1, \dots, t_k)$ is a term.

Example

Given the signature of number theory σ, we have that $\cdot(+(1, 1), x)$ is a term. We often use **infix** notation and write $(1 + 1) \cdot x$ instead.

Formulas

Definition

The set of **formulas** over a signature σ is defined inductively:

- Given terms $t_1, \ldots, t_k$ and a k-ary predicate symbol P then $P(t_1, \ldots, t_k)$ is a formula (**atomic formulas**).
- For each formula F, $\neg F$ is a formula.
- For each pair of formulas F, G, $(F \vee G)$ and $(F \wedge G)$ are both formulas.
- If F is a formula and x is a variable then $\exists x\, F$ and $\forall x\, F$ are both formulas.

Formulas

Definition

The set of **formulas** over a signature σ is defined inductively:

- Given terms $t_1, \ldots, t_k$ and a k-ary predicate symbol P then $P(t_1, \ldots, t_k)$ is a formula (**atomic formulas**).
- For each formula F, $\neg F$ is a formula.
- For each pair of formulas F, G, $(F \vee G)$ and $(F \wedge G)$ are both formulas.
- If F is a formula and x is a variable then $\exists x\, F$ and $\forall x\, F$ are both formulas.

$\exists$ and $\forall$ are the **existential** and **universal (first-order) quantifiers**.

Example

A formula over the signature of number theory::

$$\forall x\, \exists y (=(x, ((1+1) \cdot y))) \vee (=(x, (1 + (((1+1) \cdot y))))).$$

Again, use infix notation also for predicate symbols:

$$\forall x\, \exists y ((x = (1+1) \cdot y) \vee (x = 1 + (1+1) \cdot y)).$$

Quantifier depth and bounded variables

Inductive structure of formulas enables structural induction:

Definition

quantifier depth is defined as follows:

$$\begin{aligned}
\mathrm{qd}(P(t_1, \ldots, t_k)) &:= 0 \\
\mathrm{qd}(\neg F) &:= \mathrm{qd}(F) \\
\mathrm{qd}(F \wedge G) = \mathrm{qd}(F \vee G) &:= \max(\mathrm{qd}(F), \mathrm{qd}(G)) \\
\mathrm{qd}(\exists x\, F) = \mathrm{qd}(\forall x\, F) &:= \mathrm{qd}(F) + 1.
\end{aligned}$$

Quantifier depth and bounded variables

Inductive structure of formulas enables structural induction:

Definition

quantifier depth is defined as follows:

$$\begin{aligned} \mathrm{qd}(P(t_1,\dots,t_k)) &:= 0 \\ \mathrm{qd}(\neg F) &:= \mathrm{qd}(F) \\ \mathrm{qd}(F \wedge G) = \mathrm{qd}(F \vee G) &:= \max(\mathrm{qd}(F), \mathrm{qd}(G)) \\ \mathrm{qd}(\exists x\, F) = \mathrm{qd}(\forall x\, F) &:= \mathrm{qd}(F) + 1. \end{aligned}$$

Definition

In formula $\exists x\, G$, we say G is in the **scope** of the quantifier $\exists x$, likewise for $\forall x\, G$. A variable x is **bound** in F if x occurs in scope of $\exists x$ or $\forall x$. If x is not bound then x is **free**. Formula with no free variables is called **closed** or **sentence**.

Exercise

NF: non-formula F: formula, but not closed C: closed

	NF	F	C
$\forall x\ P(c)$			
$\forall x \exists y\ (Q(x, y) \vee R(x, y))$			
$\forall x\ Q(x, x) \rightarrow \exists x\ Q(x, y)$			
$\forall x\ P(x) \vee \forall x\ Q(x, x)$			
$\forall x\ (P(y) \wedge \forall y\ P(x))$			
$P(x) \rightarrow \exists x\ Q(x, P(x))$			
$\forall f\ \exists x\ P(f(x))$			

Exercise

NF: non-formula F: formula, but not closed C: closed

	NF	F	C
$\forall x\ (\neg\forall y\ Q(x,y) \wedge R(x,y))$			
$\exists z\ (Q(z,x) \vee R(y,z)) \rightarrow \exists y\ (R(x,y) \wedge Q(x,z))$			
$\exists x\ (\neg P(x) \vee P(f(c)))$			
$P(x) \rightarrow \exists x\ P(x)$			
$\exists x \forall y\ ((P(y) \rightarrow Q(x,y)) \vee \neg P(x))$			
$\exists x \forall x\ Q(x,x)$			

Semantics of first-order logic

Definition

Given a signature σ, a σ**-structure** (or **assignment**) $\mathcal{A}$ consists of:

- a non-empty set $U_\mathcal{A}$ called the **universe** of the structure;
- for each k-ary predicate symbol P in σ, a k-ary relation

$$P_\mathcal{A} \subseteq \underbrace{U_\mathcal{A} \times \cdots \times U_\mathcal{A}}_{k};$$

- for each k-ary function symbol f in σ, a k-ary function,

$$f_\mathcal{A} \colon \underbrace{U_\mathcal{A} \times \cdots \times U_\mathcal{A}}_{k} \to U_\mathcal{A};$$

- for each constant symbol c, an element $c_\mathcal{A}$ of $U_\mathcal{A}$;
- for each variable x an element $x_\mathcal{A}$ of $U_\mathcal{A}$.

Example

Let σ be the signature of number theory. The natural σ-structure $\mathcal{A}$ is:

- $U_{\mathcal{A}} := \mathbb{N} = \{0, 1, \ldots\}$
- $0_{\mathcal{A}} := 0$, $1_{\mathcal{A}} := 1$
- $+_{\mathcal{A}} := (m, n) \mapsto m + n$
- $\cdot_{\mathcal{A}} := (m, n) \mapsto m \cdot n$
- $=_{\mathcal{A}} := \{(n, n) : n \in \mathbb{N}\}$

Example

Let σ be the signature of number theory. The natural σ-structure $\mathcal{A}$ is:

- $U_{\mathcal{A}} := \mathbb{N} = \{0, 1, \ldots\}$
- $0_{\mathcal{A}} := 0$, $1_{\mathcal{A}} := 1$
- $+_{\mathcal{A}} := (m, n) \mapsto m + n$
- $\cdot_{\mathcal{A}} := (m, n) \mapsto m \cdot n$
- $=_{\mathcal{A}} := \{(n, n) : n \in \mathbb{N}\}$

BUT the following $\mathcal{B}$ is also a σ-structure:

- $U_{\mathcal{B}} := \{A, B, 5\}$
- $0_{\mathcal{B}} := A$, $1_{\mathcal{A}} := 5$
- $+_{\mathcal{B}} := (m, n) \mapsto 5$
- $\cdot_{\mathcal{B}} := (m, n) \mapsto A$
- $=_{\mathcal{B}} := \{(A, B), (B, B)\}$

Semantics of first-order logic

Definition

Value $\mathcal{A}(t) \in U_{\mathcal{A}}$ of term t inductively defined as follows:

- For a constant symbol c, $\mathcal{A}(c) := c_{\mathcal{A}}$.
- For a variable x, $\mathcal{A}(x) := x_{\mathcal{A}}$.
- For a term $f(t_1, \ldots, t_k)$, where f is k-ary function symbol and $t_1, \ldots, t_k$ are terms,

$$\mathcal{A}(f(t_1, \ldots, t_k)) := f_{\mathcal{A}}(\mathcal{A}(t_1), \ldots, \mathcal{A}(t_k)).$$

Semantics of first-order logic

Definition

Value $\mathcal{A}(t) \in U_{\mathcal{A}}$ of term t inductively defined as follows:

- For a constant symbol c, $\mathcal{A}(c) := c_{\mathcal{A}}$.
- For a variable x, $\mathcal{A}(x) := x_{\mathcal{A}}$.
- For a term $f(t_1, \ldots, t_k)$, where f is k-ary function symbol and $t_1, \ldots, t_k$ are terms,

$$\mathcal{A}(f(t_1, \ldots, t_k)) := f_{\mathcal{A}}(\mathcal{A}(t_1), \ldots, \mathcal{A}(t_k)).$$

Definition

Define the **satisfaction relation** $\mathcal{A} \models F$ ($\mathcal{A}$ **satisfies** F, or $\mathcal{A}$ is a **model** of F) by structural induction:

- $\mathcal{A} \models P(t_1, \ldots, t_k)$ if and only if $(\mathcal{A}(t_1), \ldots, \mathcal{A}(t_k)) \in P_{\mathcal{A}}$.
- $\mathcal{A} \models (F \wedge G)$ if and only if $\mathcal{A} \models F$ and $\mathcal{A} \models G$.
- $\mathcal{A} \models (F \vee G)$ if and only if $\mathcal{A} \models F$ or $\mathcal{A} \models G$.
- $\mathcal{A} \models \neg F$ if and only if $\mathcal{A} \not\models F$.
- $\mathcal{A} \models \exists x\, F$ if and only if there exists $a \in U_{\mathcal{A}}$ such that $\mathcal{A}_{[x \mapsto a]} \models F$.
- $\mathcal{A} \models \forall x\, F$ if and only if $\mathcal{A}_{[x \mapsto a]} \models F$ for all $a \in U_{\mathcal{A}}$.

Semantics of first-order logic

Example

Let $\mathcal{A}$ be the natural σ-structure of number theory, then

$$\mathcal{A} \models \forall x \, \exists y ((x = (1 + 1) \cdot y) \vee (x = 1 + (1 + 1) \cdot y)).$$

Semantics of first-order logic

Example

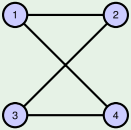
Let $\mathcal{A}$ be the natural σ-structure of number theory, then

$$\mathcal{A} \models \forall x\, \exists y((x = (1+1) \cdot y) \vee (x = 1 + (1+1) \cdot y)).$$

Every natural number is odd or even

Semantics of first-order logic

Example



- Undirected graph as σ-structure with one binary relation symbol E interpreted as the edge relation.
- Above graph represented by structure $\mathcal{A}$ with universe $U_{\mathcal{A}} = \{1, 2, 3, 4\}$ and irreflexive symmetric binary relation

$$E_{\mathcal{A}} = \{(1,2), (2,3), (3,4), (4,1), (2,1), (3,2), (4,3), (1,4)\}\,.$$

- Edge relation is irreflexive and symmetric:

$$\forall x\, \neg E(x,x) \wedge \forall x\, \forall y\, (E(x,y) \rightarrow E(y,x))\,.$$

- Every pair of nodes are connected by a path of length 3:

$$\forall x\, \forall y\, \exists z_1\, \exists z_2\, (E(x,z_1) \wedge E(z_1,z_2) \wedge E(z_2,y)).$$

The relevance lemma

Lemma

Suppose that $\mathcal{A}$ and $\mathcal{A}'$ are σ-assignments with the same universe and identical interpretations of the predicate, function, and constant symbols in σ. If $\mathcal{A}$ and $\mathcal{A}'$ give the same interpretation to each variable occurring free in some σ-formula F then $\mathcal{A} \models F$ if and only if $\mathcal{A}' \models F$.

The relevance lemma

Lemma

Suppose that $\mathcal{A}$ and $\mathcal{A}'$ are σ-assignments with the same universe and identical interpretations of the predicate, function, and constant symbols in σ. If $\mathcal{A}$ and $\mathcal{A}'$ give the same interpretation to each variable occurring free in some σ-formula F then $\mathcal{A} \models F$ if and only if $\mathcal{A}' \models F$.

Proof.

By structural induction on terms and formulas. □

Validity, satisfiability, consequence, equivalence

Let F be a formula over a signature σ.

1. F ist **valid** if every σ-structure is a model of F, denoted $\models F$.
2. F is **satisfiable** if it has at least one model.

Let $F_1, \ldots, F_k, G$ be formulas over the same signature.

G is a **consequence of** or **entailed by** $F_1, \ldots, F_k$, denoted $F_1, \ldots, F_k \models G$, if every model of $\{F_1, \ldots, F_k\}$ is also model of G.

Two formulas F and G over the same signature are **equivalent**, denoted $F \equiv G$, if they have the same models.

Exercise

V: valid S: satisfiable, but not valid U: unsatisfiable

	V	S	U
$\forall x\, P(a)$			
$\exists x\, (\neg P(x) \vee P(a))$			
$P(a) \to \exists x\, P(x)$			
$P(x) \to \exists x\, P(x)$			
$\forall x\, P(x) \to \exists x\, P(x)$			
$\forall x\, P(x) \wedge \neg \forall y\, P(y)$			

Exercise

V: valid S: satisfiable, but not valid U: unsatisfiable

	V	S	U
$\forall x \ (P(x,x) \to \exists x \forall y \ P(x,y))$			
$\forall x \forall y \ (x = y \to f(x) = f(y))$			
$\forall x \forall y \ (f(x) = f(y) \to x = y)$			
$\exists x \exists y \exists z \ (f(x) = y \land f(x) = z \land y \neq z)$			

Exercise

1. $\forall x\ P(x) \vee \forall x\ Q(x,x)$
2. $\forall x\ (P(x) \vee Q(x,x))$
3. $\forall x\ (\forall z P(z) \vee \forall y\ Q(x,y))$

	Y	N
$1 \models 2$		
$2 \models 3$		
$3 \models 1$		

Exercise

1. $\exists y \forall x \; P(x, y)$
2. $\forall x \exists y \; P(x, y)$

	Y	N
$1 \models 2$		
$2 \models 1$		

Exercise

	Y	N
$\forall x \forall y\ F \equiv \forall y \forall x\ F$		
$\forall x \exists y\ F \equiv \exists x \forall y\ F$		
$\exists x \exists y\ F \equiv \exists y \exists x\ F$		
$\forall x\ F \vee \forall x\ G \equiv \forall x\ (F \vee G)$		
$\forall x\ F \wedge \forall x\ G \equiv \forall x\ (F \wedge G)$		
$\exists x\ F \vee \exists x\ G \equiv \exists x\ (F \vee G)$		
$\exists x\ F \wedge \exists x\ G \equiv \exists x\ (F \wedge G)$		

Predicate logic with equality

Predicate logic
$+$
distinguished predicate symbol "$=$" of arity 2.

Semantics: a structure $\mathcal{A}$ of predicate logic with equality always maps the predicate symbol $=$ to the identity relation:

$$\mathcal{A}(=) \; = \; \{(d, d) \mid d \in U_{\mathcal{A}}\} \, .$$

Formalizing statements

What does it mean to “formalize” a statement in predicate logic?

Formalizing statements

What does it mean to "formalize" a statement in predicate logic?
It means to give

- a formula F over a signature σ, and
- a **partial** structure $\mathcal{A}$ assigning meaning to some symbols of σ,

such that the statement holds iff **every** σ **structure that extends** $\mathcal{A}$ is a model of F.

Formalizing statements

What does it mean to "formalize" a statement in predicate logic?
It means to give

- a formula *F* over a signature σ, and
- a **partial** structure $\mathcal{A}$ assigning meaning to some symbols of σ,

such that the statement holds iff **every** σ **structure that extends** $\mathcal{A}$ is a model of *F*.

Intuitively, the symbols interpreted in $\mathcal{A}$ are those that the formalizer assumes are known by whoever is going to read the formula. *F* may contain other symbols, but then *F* must define what they mean (see next slides).

Formalizing statements

What does it mean to “formalize” a statement in predicate logic?
It means to give

- a formula F over a signature σ, and
- a **partial** structure $\mathcal{A}$ assigning meaning to some symbols of σ,

such that the statement holds iff **every σ structure that extends $\mathcal{A}$** is a model of F.

Intuitively, the symbols interpreted in $\mathcal{A}$ are those that the formalizer assumes are known by whoever is going to read the formula. F may contain other symbols, but then F must define what they mean (see next slides).

Typically, the formalizer chooses names for the symbols that suggest their meaning. The structure is often omitted, because it is assumed to be known (danger!).

Formalizing statements

What does it mean to “formalize” a statement in predicate logic?
It means to give

- a formula F over a signature σ, and
- a **partial** structure $\mathcal{A}$ assigning meaning to some symbols of σ,

such that the statement holds iff **every** σ **structure that extends** $\mathcal{A}$ is a model of F.

Intuitively, the symbols interpreted in $\mathcal{A}$ are those that the formalizer assumes are known by whoever is going to read the formula. F may contain other symbols, but then F must define what they mean (see next slides).

Typically, the formalizer chooses names for the symbols that suggest their meaning. The structure is often omitted, because it is assumed to be known (danger!).

We give different formalizations of the statement

There are infinitely many prime numbers

Formalization I

If the meanings of “prime” and “‘greater-than” are known, then we can take a signature with a unary predicate symbol *Pr* and a binary predicate symbol $>$:

Formula F_1: $\forall x \exists y \, (Pr(y) \wedge y > x)$

Structure $\mathcal{A}_1$:

$$\begin{array}{rcl} U_{\mathcal{A}_1} & = & \mathbb{N} \\ Pr_{\mathcal{A}_1} & = & \{n \in \mathbb{N} \mid n \text{ is prime}\} \\ >_{\mathcal{A}_1} & = & \{(n, m) \in \mathbb{N} \mid n > m\} \end{array}$$

Formalization I

If the meanings of “prime” and “‘greater-than” are known, then we can take a signature with a unary predicate symbol *Pr* and a binary predicate symbol $>$:

Formula F_1: $\forall x \exists y \, (Pr(y) \wedge y > x)$

Structure $\mathcal{A}_1$:

$$\begin{aligned} U_{\mathcal{A}_1} &= \mathbb{N} \\ Pr_{\mathcal{A}_1} &= \{n \in \mathbb{N} \mid n \text{ is prime}\} \\ >_{\mathcal{A}_1} &= \{(n, m) \in \mathbb{N} \mid n > m\} \end{aligned}$$

What if the meaning of “prime” is not known?

Formalization II

If the meaning of “divides” is known, then we can take a signature with a constant *one* and two binary predicate symbols $>$, *Dv* (we use predicate logic with equality), and **define** “prime”:

Formula F_2:

$$\forall x\ Pr(x) \leftrightarrow (\forall y\ Dv(y,x) \rightarrow (y = x \vee y = one))$$
$$\rightarrow\ \ \forall x \exists y\ Pr(y) \wedge y > x$$

Structure $\mathcal{A}_2$:

$$U_{\mathcal{A}_2} = \mathbb{N}$$
$$Dv_{\mathcal{A}_2} = \{(n,n) \in \mathbb{N} \mid n \text{ divides } m\}$$
$$>_{\mathcal{A}_2} = \{(n,m) \in \mathbb{N} \mid n > m\}$$

We are now stating “ if we define prime numbers as . . . then there are infinitely many prime numbers”.

The statement “there are infinitely many prime numbers” holds iff **every structure that extends** $\mathcal{A}_2$ satisfies the formula.

Formalization II

If the meaning of “divides” is known, then we can take a signature with a constant *one* and two binary predicate symbols $>$, *Dv* (we use predicate logic with equality), and **define** “prime”:

Formula F_2:

$$\forall x\ Pr(x) \leftrightarrow (\forall y\ Dv(y,x) \rightarrow (y = x \vee y = one))$$
$$\rightarrow\ \forall x \exists y\ Pr(y) \wedge y > x$$

Structure $\mathcal{A}_2$:

$$U_{\mathcal{A}_2} = \mathbb{N}$$
$$Dv_{\mathcal{A}_2} = \{(n,n) \in \mathbb{N} \mid n \text{ divides } m\}$$
$$>_{\mathcal{A}_2} = \{(n,m) \in \mathbb{N} \mid n > m\}$$

We are now stating “ if we define prime numbers as . . . then there are infinitely many prime numbers”.

The statement “there are infinitely many prime numbers” holds iff **every structure that extends** $\mathcal{A}_2$ satisfies the formula.

What if the meaning of “divides” is not known?

Formalization III

If the meaning of “product” is known , then we can take

Formula F_3:

$$
\begin{array}{ll}
 & \forall x\,\forall y\; Dv(x,y) \leftrightarrow \exists z\; prod(x,z) = y \\
\wedge & \forall x\; Pr(x) \leftrightarrow (\forall y\; Dv(y,x) \rightarrow (y = x \vee y = one)) \\
\rightarrow & \forall x \exists y\; Pr(y) \wedge y > x
\end{array}
$$

(the conjunction of the first two formulas implies the third)

Structure $\mathcal{A}_3$:

$$
\begin{array}{rcl}
U_{\mathcal{A}_3} & = & \mathbb{N} \\
>_{\mathcal{A}_3} & = & \{(n,m) \in \mathbb{N} \mid n > m\} \\
one_{\mathcal{A}_3} & = & 1 \\
prod_{\mathcal{A}_3}(n,m) & = & n \cdot m
\end{array}
$$

What if the meaning of “product” is not known ?

Formalization IV

If the meaning of “sum”, “successor”, “one” and “zero” is known, then we can take

Formula F_4:

$$
\begin{array}{ll}
 & \forall x \; prod(x, zero) = zero \\
\wedge & \forall x \forall y \; prod(x, succ(y)) = sum(prod(x, y), y) \\
\wedge & \forall x \; \forall y \; Dv(x, y) \leftrightarrow \exists z \; prod(x, z) = y \\
\wedge & \forall x \; Pr(x) \leftrightarrow (\forall y \; Dv(y, x) \rightarrow (y = x \vee y = one)) \\
\rightarrow & \forall x \exists y \; (Pr(y) \wedge y > x)
\end{array}
$$

Structure $\mathcal{A}_4$ only defines $>$, *sum*, *succ*, *one*, *zero*.

Observe however: *prod* is defined inductively. The definition is no longer a macro, in the sense that we cannot produce an “equivalent” formula without the symbol *prod*.

Formalization IV

If the meaning of “sum”, “successor”, “one” and “zero” is known, then we can take

Formula F_4:

$$
\begin{array}{ll}
 & \forall x \; prod(x, zero) = zero \\
\wedge & \forall x \forall y \; prod(x, succ(y)) = sum(prod(x, y), y) \\
\wedge & \forall x \; \forall y \; Dv(x, y) \leftrightarrow \exists z \; prod(x, z) = y \\
\wedge & \forall x \; Pr(x) \leftrightarrow (\forall y \; Dv(y, x) \rightarrow (y = x \vee y = one)) \\
\rightarrow & \forall x \exists y \; (Pr(y) \wedge y > x)
\end{array}
$$

Structure $\mathcal{A}_4$ only defines $>$, *sum*, *succ*, *one*, *zero*.

Observe however: *prod* is defined inductively. The definition is no longer a macro, in the sense that we cannot produce an “equivalent” formula without the symbol *prod*.

What if the meaning of “sum” is not known?

Formalization V

Formula F_5:

$$
\begin{array}{ll}
 & \forall x\ sum(x, zero) = x \\
\wedge & \forall x \forall y\ sum(x, succ(y)) = succ(sum(x, y)) \\
\wedge & \forall x\ prod(x, zero) = zero \\
\wedge & \forall x \forall y\ prod(x, succ(y)) = sum(prod(x, y), y) \\
\wedge & \forall x\ \forall y\ (Div(x, y) \leftrightarrow \exists z\ prod(x, z) = y) \\
\wedge & \forall x\ Pri(x) \leftrightarrow (\forall y\ Div(y, x) \rightarrow (y = x \vee y = one)) \\
\rightarrow & \forall x \exists y\ (Pri(y) \wedge y > x)
\end{array}
$$

Structure $\mathcal{A}_5$ only defines $>$, *succ*, *one*, *zero*.

Formalization V

Formula F_5:

$$
\begin{array}{ll}
 & \forall x\ sum(x, zero) = x \\
\wedge & \forall x \forall y\ sum(x, succ(y)) = succ(sum(x, y)) \\
\wedge & \forall x\ prod(x, zero) = zero \\
\wedge & \forall x \forall y\ prod(x, succ(y)) = sum(prod(x, y), y) \\
\wedge & \forall x\ \forall y\ (Div(x, y) \leftrightarrow \exists z\ prod(x, z) = y) \\
\wedge & \forall x\ Pri(x) \leftrightarrow (\forall y\ Div(y, x) \rightarrow (y = x \vee y = one)) \\
\rightarrow & \forall x \exists y\ (Pri(y) \wedge y > x)
\end{array}
$$

Structure $\mathcal{A}_5$ only defines $>$, *succ*, *one*, *zero*.

What if the meaning of ‘greater than” and “one” is not known?

Formalization VI

Formula F_6:

$$
\begin{array}{ll}
 & one = succ(zero) \\
\wedge & \forall x \forall y\ x > y \leftrightarrow \exists z\ \neg(z = zero) \wedge sum(y,z) = x \\
\wedge & \forall x\ sum(x, zero) = x \\
\wedge & \forall x \forall y\ sum(x, succ(y)) = succ(sum(x,y)) \\
\wedge & \forall x\ prod(x, zero) = zero \\
\wedge & \forall x \forall y\ prod(x, succ(y)) = sum(prod(x,y), y) \\
\wedge & \forall x\ \forall y\ (Div(x,y) \leftrightarrow \exists z\ prod(x,z) = y) \\
\wedge & \forall x\ Pri(x) \leftrightarrow (\forall y\ Div(y,x) \rightarrow (y = x \vee y = one)) \\
\rightarrow & \forall x \exists y\ (Pri(y) \wedge y > x)
\end{array}
$$

Structure $\mathcal{A}_6$ only defines *succ*, *zero*.