

COMPSCI/SFWRENG 2FA3
Discrete Mathematics with Applications II
Winter 2020

3 Predicate Logic

William M. Farmer

Department of Computing and Software
McMaster University

February 9, 2020



Outline

1. Orders revisited
2. Logic
3. MSFOL, a many-sorted first-order logic
 - a. Syntax
 - b. Semantics
 - c. Theories
 - d. Proof systems
4. Summary

1. Orders Revisited

Pre-Orders

- A **pre-order** is a mathematical structure (S, \leq) where \leq is a binary relation on S that is:
 - ▶ **Reflexive**: $\forall x \in S . x \leq x$.
 - ▶ **Transitive**: $\forall x, y, z \in S . x \leq y \wedge y \leq z \Rightarrow x \leq z$.
- **Example**: (F, \Rightarrow) is a pre-order where F is a set of formulas and \Rightarrow is implication.
- A pre-order can have cycles.
- Every binary relation R on a set S can be extended to a pre-order on S by taking the reflexive and transitive closure of R .
- A **equivalence relation** is a pre-order (S, E) that is:
 - ▶ **Symmetric**: $\forall x, y \in S . x E y \Rightarrow y E x$.

Partial Orders

- A **weak partial order** is a mathematical structure (S, \leq) where \leq is a binary relation on S that is:
 - ▶ **Reflexive**: $\forall x \in S. x \leq x$.
 - ▶ **Antisymmetric**: $\forall x, y \in S. (x \leq y \wedge y \leq x) \Rightarrow x = y$.
 - ▶ **Transitive**: $\forall x, y, z \in S. (x \leq y \wedge y \leq z) \Rightarrow x \leq z$.
- A **strict partial order** is a mathematical structure $(S, <)$ where $<$ is a binary relation on S that is:
 - ▶ **Irreflexive**: $\forall x \in S. \neg(x < x)$.
 - ▶ **Asymmetric**: $\forall x, y \in S. x < y \Rightarrow \neg(y < x)$.
 - ▶ **Transitive**: $\forall x, y, z \in S. (x < y \wedge y < z) \Rightarrow x < z$.
- **Examples**: $(\mathcal{P}(S), \subseteq)$ and $(\mathcal{P}(S), \subset)$ are weak and strict partial orders.
- A partial order does not have cycles.
- Every pre-order can be interpreted as a partial order.

Weak vs. Strict Orders (iClicker)

If (S, \leq) is a weak partial order, then $(S, <)$ will be the same order expressed as strict partial order if $<$ is defined as

- A. $a < b$ iff $a \leq b \wedge a = b$.
- B. $a < b$ iff $a \leq b \vee a = b$.
- C. $a < b$ iff $a \leq b \wedge a \neq b$.
- D. $a < b$ iff $a \leq b \vee a \neq b$.

Some Basic Order Definitions

- Let (P, \leq) be a weak partial order and $S \subseteq P$.
- A **maximal element** [minimal element] of S is a $M \in S$ [$m \in S$] such that $\neg(M < x)$ [$\neg(x < m)$] for all $x \in S$.
- The **maximum element** or **greatest element** [minimum element or least element] of S , if it exists, is the $M \in S$ [$m \in S$] such that $x \leq M$ [$m \leq x$] for all $x \in S$.
- An **upper bound** [lower bound] of S is a $u \in P$ [$l \in P$] such that $x \leq u$ [$l \leq x$] for all $x \in S$.
- The **least upper bound** or **supremum** [greatest lower bound or infimum] of S , if it exists, is a $U \in P$ [$L \in P$] such that U is an upper bound of S and, if u is an upper bound of S , then $U \leq u$ [L is a lower bound of S and, if l is a lower bound of S , then $l \leq L$].

Review

- Four kinds of recursion and induction:
 1. Natural number
 2. Structural
 3. Ordinal
 4. Well-founded
- Orders

Total Orders

- A **weak total order** is a mathematical structure (S, \leq) where \leq is a binary relation on S that is:
 - ▶ **(Reflexive)**: $\forall x. x \leq x$.
 - ▶ **Antisymmetric**: $\forall x, y \in S. (x \leq y \wedge y \leq x) \Rightarrow x = y$.
 - ▶ **Transitive**: $\forall x, y, z \in S. (x \leq y \wedge y \leq z) \Rightarrow x \leq z$.
 - ▶ **Total**: $\forall x, y \in S. x \leq y \vee y \leq x$.
 - A **strict total order** is a mathematical structure $(S, <)$ where $<$ is a binary relation on S that is:
 - ▶ **Irreflexive**: $\forall x \in S. \neg(x < x)$.
 - ▶ **(Asymmetric)**: $\forall x, y \in S. x < y \Rightarrow \neg(y < x)$.
 - ▶ **Transitive**: $\forall x, y, z \in S. (x < y \wedge y < z) \Rightarrow x < z$.
 - ▶ **Trichotomous**: $\forall x, y \in S. x < y \vee y < x \vee x = y$.
- Examples:** (\mathbb{N}, \leq) , (\mathbb{Z}, \leq) , (\mathbb{Q}, \leq) , and (\mathbb{R}, \leq) are weak total orders.

Well-Orders

- A **well-order** is a strict total order $(S, <)$ such that every nonempty subset of S has a minimum element with respect to $<$.
- **Proposition.** Every well-order $(S, <)$ is Noetherian, i.e., S contains no infinite descending sequences of the form
$$\cdots < x_2 < x_1 < x_0.$$
- **Examples.**
 1. $(\mathbb{N}, <)$ is a well-order where $<$ is the usual order on \mathbb{N} .
 2. $(\mathbb{Z}, <)$ is **not** a well-order where $<$ is the usual order on \mathbb{Z} .
 3. $(S, <)$ is a well-order where S is
$$\{0, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \dots, 1, 1\frac{1}{2}, 1\frac{2}{3}, 1\frac{3}{4}, \dots\}$$
and $<$ is the usual order on \mathbb{Q} .
 4. $(\mathbb{N} \times \mathbb{N}, <_{\text{lex}})$ is a well-order where $<_{\text{lex}}$ is lexicographic order on $\mathbb{N} \times \mathbb{N}$.

2. Logic

What is Logic?

1. The study of the principles underlying sound reasoning.
 - ▶ Central idea: **logical consequence**.
2. The branch of mathematics underlying mathematical reasoning and computation.

Fundamental Distinctions

Logic makes several fundamental distinctions:

- Syntax vs. semantics.
- Language vs. metalanguage.
- Theory vs. model.
- Truth vs. proof.

Metalanguages (iClicker)

Which of the following is a suitable metalanguage for English?

- A. English.
- B. First-order logic.
- C. French.
- D. Latin.
- E. Old English.

What is a Logic?

- Informally, a logic is a system of reasoning.
- Formally, a logic is a family of formal languages with:
 1. A common syntax.
 2. A common semantics.
 3. A notion of logical consequence.
- A logic may include one or more proof systems for mechanically deriving that a given formula is a logical consequence of a given set of formulas.
- Two most common examples:
 - ▶ Propositional logic.
 - ▶ First-order logic.
- First-order logic is called first-order predicate logic and first-order quantificational logic.

What is Predicate Logic?

- Predicate logic is the study of statements about individuals constructed using functions, predicates, propositional connectives, and quantifiers.
- Examples of predicate logics:
 - ▶ First-order logic.
 - ▶ Many-sorted first-order logic.
 - ▶ Second-order logic.
 - ▶ Simple type theory (classical higher-order logic).
 - ▶ Dependent type theory (constructive higher-order logic).
- First-order logic is the leading form of predicate logic.
 - ▶ It is “first-order” because quantification is over individuals but not over higher-order objects such as functions and predicates.
 - ▶ It is suited more for theory than practice.

Many-Sorted First-Order Logic

- Many-sorted first-order logic is a version of first-order logic that admits multiple domains of individuals.
 - ▶ These domains are called **sorts**.
 - ▶ Standard first-order logic has just a single sort.
- This single change makes many-sorted first-order logic much more practical than standard first-order logic.
- We will develop a version of many-sorted first-order logic called **MSFOL**.

3a. MSFOL: Syntax

Types

- Let \mathcal{B} be a nonempty set of symbols called **base types**.
 - ▶ Each base type denotes a **sort**.
- A \mathcal{B} -type is a string of symbols defined inductively by:
 - ▶ (Type of booleans) \mathbb{B} is a \mathcal{B} -type.
 - ▶ (Base type) Each $\alpha \in \mathcal{B}$ is a \mathcal{B} -type.
 - ▶ (Function type) If α and β are \mathcal{B} -types, then $(\alpha \rightarrow \beta)$ is a \mathcal{B} -type.
 - ▶ (Product type) If α and β are \mathcal{B} -types, then $(\alpha \times \beta)$ is a \mathcal{B} -type.
- Thus a \mathcal{B} -type is built from \mathbb{B} and the members of \mathcal{B} by applying the function and product type constructors.
- We may omit matching pairs of parentheses in types when there is no loss of meaning.

Syntax vs. Semantics (iClicker)

Which sentence in English is syntactically incorrect?

- A. They goes to Mac.
- B. His statements infer that he is angry.
- C. She knows the induction principal that is needed.
- D. Long live the king!
- E. All of the above.

Base Cases (iClicker)

How many base cases does the definition of a \mathcal{B} -type have?

- A. 0.
- B. 1.
- C. ☒ 2.
- D. 4.

\mathcal{B} -Types as an Inductive Set

- The set of \mathcal{B} -types can be formally defined as an inductive type.
- Let **Type** be the inductive set defined by the following constructors:
 1. $\mathbb{B} : \text{Type}$.
 2. $\text{Base} : \mathcal{B} \rightarrow \text{Type}$.
 3. $\text{Function} : \text{Type} \times \text{Type} \rightarrow \text{Type}$.
 4. $\text{Product} : \text{Type} \times \text{Type} \rightarrow \text{Type}$.
- Thus the members of **Type** represent the \mathcal{B} -types.

Admin — January 28

- A notetaker for this course is needed for SAS students.
 - ▶ For more information and to register, see <https://sas.mcmaster.ca/volunteer-notetaking/>
 - ▶ Send questions to sasnotes@mcmaster.ca.
- **Midterm Test 1** will be on Wed, Feb 5, at 7:00–9:00 PM.
 - ▶ Format: Two-stage test, 30 multiple-choice questions
 - ▶ Covers the material from Weeks 01, 02, 03, and 04.
 - ▶ Sample test will be posted on Mon, Feb 3.
 - ▶ Review session on Wed, Feb 5, instead of the lecture.
 - ▶ Room assignments will be posted on Avenue.
- **Solutions to Week 03 Exercises**
 - ▶ The ideas behind the solutions are more important than the solutions themselves.
 - ▶ Some of the solutions are a bit rough in various ways.
- Office hours: To see me please send me a note with times.
- **Are there any questions?**

Assignment 2: Question 1

Let **Poly** be the inductive type defined by the following constructors:

$X : \text{Poly}$.

$\text{Coeff} : \mathbb{Q} \rightarrow \text{Poly}$.

$\text{Sum} : \text{Poly} \times \text{Poly} \rightarrow \text{Poly}$.

$\text{Prod} : \text{Poly} \times \text{Poly} \rightarrow \text{Poly}$.

Define

$\text{val} : \text{Poly} \times \mathbb{Q} \rightarrow \mathbb{Q}$

by structural recursion using pattern matching so that, for all $p \in \text{Poly}$ and $q \in \mathbb{Q}$, $\text{val}(p, q)$ is the value of p at q .

Assignment 2: Question 2

Let BinNum be the inductive set defined by:

$\text{Nil} : \text{BinNum}$.

$\text{Join} : \text{BinNum} \times \text{Bit} \rightarrow \text{BinNum}$.

The function $\text{val} : \text{BinNum} \rightarrow \mathbb{N}$ is defined by:

$\text{val}(\text{Nil}) = 0$.

$\text{val}(\text{Join}(u, \text{Zero})) = 2 * \text{val}(u)$.

$\text{val}(\text{Join}(u, \text{One})) = (2 * \text{val}(u)) + 1$.

The function $\text{add} : \text{BinNum} \times \text{BinNum} \rightarrow \text{BinNum}$ is defined using recursion and pattern matching (see Assignment 1).

Prove, for all $u, v \in \text{BinNum}$,

$\text{val}(\text{add}(u, v)) = \text{val}(u) + \text{val}(v)$.

Review

- Orders.
- Logic.
- Types in the MSFOL syntax.

Signatures

- A **formal language** of a logic is a set of **expressions** built from a set of **symbols** that include:
 1. **Variables** that may have different meanings in an interpretation of the language.
 2. **Logical constants** like \wedge and \forall that have the same meaning in every interpretation of every language.
 3. **Nonlogical constants** like 0 and $<$ that have the same meaning within an interpretation of the language but possibly different meanings within different interpretations of the language.
 4. **Punctuation symbols** like parentheses.
- The **signature** S of a formal language L defines the set of nonlogical symbols of L .
 - We will see that S also specifies the expressions of L .

Signatures in MSFOL

- Let \mathcal{V} be a countably infinite set of symbols called **variable symbols**.
- A **signature of MSFOL** is a tuple $\Sigma = (\mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{P}, \tau)$ where:
 1. \mathcal{B} is a nonempty set of **base types**.
 2. \mathcal{C} is a set of symbols called **constant symbols**.
 3. \mathcal{F} is a set of symbols called **function symbols**.
 4. \mathcal{P} is a set of symbols called **predicate symbols**.
 5. $\mathcal{V}, \mathcal{B}, \mathcal{C}, \mathcal{F}$, and \mathcal{P} are pairwise disjoint.
 6. τ is a function that maps the members of $\mathcal{C} \cup \mathcal{F} \cup \mathcal{P}$ to \mathcal{B} -types such that:
 - a. $\tau(c) \in \mathcal{B}$ for each $c \in \mathcal{C}$.
 - b. $\tau(f)$ has the form $(\alpha_1 \times \dots \times \alpha_n) \rightarrow \alpha$ for each $f \in \mathcal{F}$ where $\alpha_1, \dots, \alpha_n, \alpha \in \mathcal{B}$.
 - c. $\tau(p)$ has the form $(\alpha_1 \times \dots \times \alpha_n) \rightarrow \mathbb{B}$ for each $p \in \mathcal{P}$ where $\alpha_1, \dots, \alpha_n \in \mathcal{B}$.
- Σ is **finite** if $\mathcal{B}, \mathcal{C}, \mathcal{F}$, and \mathcal{P} are finite.

Examples of Signatures [1/3]

- $\Sigma_{\text{eq}} = (\mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{P}, \tau)$ where

- ▶ $\mathcal{B} = \{U\}$.
- ▶ $\mathcal{C} = \emptyset$.
- ▶ $\mathcal{F} = \emptyset$.
- ▶ $\mathcal{P} = \emptyset$.

is the signature of a language for **equality**.

- $\Sigma_{\text{ord}} = (\mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{P}, \tau)$ where

- ▶ $\mathcal{B} = \{U\}$.
- ▶ $\mathcal{C} = \emptyset$.
- ▶ $\mathcal{F} = \emptyset$.
- ▶ $\mathcal{P} = \{<\}$ with $\tau(<) = U \times U \rightarrow \mathbb{B}$.

is the signature of a language for a **strict order**.

Examples of Signatures [2/3]

- $\Sigma_{\text{nat}}^1 = (\mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{P}, \tau)$ where

- ▶ $\mathcal{B} = \{\mathbb{N}\}$.
- ▶ $\mathcal{C} = \{0\}$ with $\tau(0) = \mathbb{N}$.
- ▶ $\mathcal{F} = \{S, +, *\}$ with $\tau(S) = \mathbb{N} \rightarrow \mathbb{N}$ and $\tau(+)=\tau(*)=\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$.
- ▶ $\mathcal{P} = \{<\}$ with $\tau(<) = \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{B}$.

is the signature of a language for **natural number arithmetic**.

- $\Sigma_{\text{nat}}^2 = (\mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{P}, \tau)$ where

- ▶ $\mathcal{B} = \{\mathbb{N}\}$.
- ▶ $\mathcal{C} = \{0, 1, 2, \dots\}$ with $\tau(0) = \tau(1) = \tau(2) = \dots = \mathbb{N}$.
- ▶ $\mathcal{F} = \{+, *\}$ with $\tau(+)=\tau(*)=\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$.
- ▶ $\mathcal{P} = \{<\}$ with $\tau(<) = \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{B}$.

is the signature of an alternate language for **natural number arithmetic**.

Examples of Signatures [3/3]

- $\Sigma_{\text{stack}} = (\mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{P}, \tau)$ where:

- ▶ $\mathcal{B} = \{\text{Element}, \text{Stack}\}$.
- ▶ $\mathcal{C} = \{\text{error}, \text{bottom}\}$ with $\tau(\text{error}) = \text{Element}$ and $\tau(\text{bottom}) = \text{Stack}$.
- ▶ $\mathcal{F} = \{\text{push}, \text{pop}, \text{top}\}$ with $\tau(\text{push}) = \text{Element} \times \text{Stack} \rightarrow \text{Stack}$, $\tau(\text{pop}) = \text{Stack} \rightarrow \text{Stack}$, and $\tau(\text{top}) = \text{Stack} \rightarrow \text{Element}$.
- ▶ $\mathcal{P} = \emptyset$.

is the signature of a language for **stacks of abstract elements**.

Terms

- In MSFOL, a **term** is an expression that denotes a **member of a sort**.
- Let $\Sigma = (\mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{P}, \tau)$ be a signature of MSFOL and $\alpha \in \mathcal{B}$.
- A **Σ -term of type α of MSFOL** is a string of symbols defined inductively by:
 1. **(Variable)** If $x \in \mathcal{V}$, then $x : \alpha$ is a Σ -term of type α .
 2. **(Constant)** If $c \in \mathcal{C}$ with $\tau(c) = \alpha$, then c is a Σ -term of type α .
 3. **(Function Application)** If $f \in \mathcal{F}$ with $\tau(f) = (\alpha_1 \times \dots \times \alpha_n) \rightarrow \alpha$ and t_1, \dots, t_n are Σ -terms of type $\alpha_1, \dots, \alpha_n$, respectively, then $f(t_1, \dots, t_n)$ is a Σ -term of type α .
- Let T_Σ be the set of Σ -terms.

Terms (iClicker)

Which of the following is a Σ_{nat}^1 -term? Note that $+$, $*$, and $<$ are written as infix operators.

- A. x where $x \in \mathcal{V}$.
- B. S .
- C. $S(0)$.
- D. $0 + 1$.
- E. $0 < 0$.

Terms as a Family of Inductive Sets

- Let $\Sigma = (\mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{P}, \tau)$ be a finite signature of MSFOL.
- The set of Σ -terms can be formally defined as a family of inductive sets, one for each $\alpha \in \mathcal{B}$.
- Let $\{\text{Term}_\alpha \mid \alpha \in \mathcal{B}\}$ be the set of inductive sets simultaneously defined by the following constructors:
 1. For all $\alpha \in \mathcal{B}$, $\text{Var}_\alpha : \mathcal{V} \rightarrow \text{Term}_\alpha$.
 2. For all $c \in \mathcal{C}$, $\text{Con}_c : \text{Term}_{\tau(c)}$.
 3. For all $f \in \mathcal{F}$,
$$\text{FunApp}_f : \text{Term}_{\alpha_1} \times \cdots \times \text{Term}_{\alpha_n} \rightarrow \text{Term}_\alpha$$
where $\tau(f) = (\alpha_1 \times \cdots \times \alpha_n) \rightarrow \alpha$.
- Thus, for each $\alpha \in \mathcal{B}$, Term_α is an inductive set representing the Σ -terms of type α of MSFOL.

Formulas

- A **formula** is an expression that denotes a **boolean value** (i.e., true or false).
- Let $\Sigma = (\mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{P}, \tau)$ be a signature of MSFOL.
- A **Σ -formula of MSFOL** is string of symbols defined inductively by:
 1. (**Equality**) If t_1 and t_2 are Σ -terms of $\alpha \in \mathcal{B}$, then $(t_1 = t_2)$ is a Σ -formula.
 2. (**Predicate Application**) If $p \in \mathcal{P}$ with $\tau(p) = (\alpha_1 \times \cdots \times \alpha_n) \rightarrow \mathbb{B}$ and t_1, \dots, t_n are Σ -terms of type $\alpha_1, \dots, \alpha_n$, respectively, then $p(t_1, \dots, t_n)$ is a Σ -formula.
 3. (**Negation and Implication**) If A and B are Σ -formulas, then $\neg A$ and $(A \Rightarrow B)$ are Σ -formulas.
 4. (**Universal Quantification**) If $x \in \mathcal{V}$, $\alpha \in \mathcal{B}$, and A is a Σ -formula, then $(\forall x : \alpha . A)$ is a Σ -formula.
- Let F_Σ be the set of Σ -formulas.

Formulas (iClicker)

Which of the following is a Σ_{nat}^1 -formula (without using any notational definitions or conventions)? Note that $+$, $*$, and $<$ are written as infix operators.

- A. $0 < 0$.
- B. $0 = 0$.
- C. $\forall x : \mathbb{N} . 0 = x$
- D. $0 + S(0)$
- E. $0 < S(0) \wedge S(0) < 0$.

Formulas as an Inductive Set

- Let $\Sigma = (\mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{P}, \tau)$ be a finite signature of MSFOL.
- The set of Σ -formulas can be formally defined as an inductive set.
- Let **Form** be the inductive set defined by the following constructors:
 1. For all $\alpha \in \mathcal{B}$, $\text{Eq}_\alpha : \text{Term}_\alpha \times \text{Term}_\alpha \rightarrow \text{Form}$.
 2. For all $p \in \mathcal{P}$,
 $\text{PredApp}_p : \text{Term}_{\alpha_1} \times \cdots \times \text{Term}_{\alpha_n} \rightarrow \text{Form}$
where $\tau(p) = (\alpha_1 \times \cdots \times \alpha_n) \rightarrow \mathbb{B}$.
 3. $\text{Neg} : \text{Form} \rightarrow \text{Form}$.
 4. $\text{Implies} : \text{Form} \times \text{Form} \rightarrow \text{Form}$.
 5. $\text{Forall} : \mathcal{V} \times \mathcal{B} \times \text{Form} \rightarrow \text{Form}$.
- Thus **Form** is an inductive set representing the Σ -formulas of MSFOL.

Admin — January 29

- You can nominate instructors and TAs for an McMaster Student Union (MSU) teaching award until Feb 2 at <https://www.msumcmaster.ca/services-directory/29-teaching-awards/surveys>
- A notetaker for this course is needed for SAS students.
 - ▶ For more information and to register, see <https://sas.mcmaster.ca/volunteer-notetaking/>
 - ▶ Send questions to sasnotes@mcmaster.ca.
- **Midterm Test 1** will be on Wed, Feb 5, at 7:00–9:00 PM.
 - ▶ Format: Two-stage test, 30 multiple-choice questions
 - ▶ Covers the material from Weeks 01, 02, 03, and 04.
 - ▶ Sample test will be posted on Mon, Feb 3.
 - ▶ Review session on Wed, Feb 5, instead of the lecture.
 - ▶ Room assignments will be posted on Avenue.
- Office hours: To see me please send me a note with times.
- **Are there any questions?**

Review

MSFOL syntax.

- \mathcal{B} -types.
- Signatures.
- Σ -terms.
- Σ -formulas.

Notational Definitions

$(s \neq t)$	stands for	$\neg(s = t)$.
\top	stands for	$(\forall x : \mathbb{B} . (x : \mathbb{B} = x : \mathbb{B}))$.
F	stands for	$\neg \top$.
$(A \vee B)$	stands for	$(\neg A \Rightarrow B)$.
$(A \wedge B)$	stands for	$\neg(\neg A \vee \neg B)$.
$(A \Leftrightarrow B)$	stands for	$((A \Rightarrow B) \wedge (B \Rightarrow A))$.
$(\exists x : \alpha . A)$	stands for	$\neg(\forall x : \alpha . \neg A)$.

Syntactic Conventions

- A pair of matching parentheses in a term or a formula may be dropped if there is no resulting ambiguity.
- A variable $x : \alpha$ occurring in the body A of $(\Box x : \alpha . A)$, where $\Box \in \{\forall, \exists\}$, may be written as x if there is no resulting ambiguity.
 - ▶ So $\forall x : \mathbb{N} . 0 = x : \mathbb{N}$ can be written as $\forall x : \mathbb{N} . 0 = x$.
- Let \Box be \forall or \exists . $(\Box x_1 : \alpha_1 \cdots (\Box x_n : \alpha_n . A) \cdots)$ may be written as

$$(\Box x_1 : \alpha_1, \dots, x_n : \alpha_n . A).$$

Similarly, $(\Box x_1 : \alpha \cdots (\Box x_n : \alpha . A) \cdots)$ may be written as

$$(\Box x_1, \dots, x_n : \alpha . A).$$

Variable Binders

- A **variable binder** is an operator applied to a variable x and an expression B that binds the occurrences of x in B to a collection of values.
- **Examples of variable binders in logic:**
 1. Universal quantification: $\forall x . B$.
 2. Existential quantification: $\exists x . B$.
 3. Function abstraction: $\lambda x . B$.
 4. Definite description: $\iota x . B$.
 5. Indefinite description: $\epsilon x . B$.
- The only primitive variable binder in MSFOL is \forall .
 - ▶ \exists is a notational definition derived from \forall .
- **Examples of variable binders in mathematics:**

$$\sum_{x=m}^n B, \prod_{x=m}^n B, B[x \mapsto a], \int_a^b B dx, \lim_{x \rightarrow a} B, \{x \mid B\}$$

Bound and Free Variables [1/3]

- Let $\Sigma = (\mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{P}, \tau)$ be a signature, t be a Σ -term, and A be a Σ -formula.
- An occurrence of a variable $x : \alpha$ in A is **bound** [**free**] if it is [not] in a subformula of A of the form $\forall x : \alpha . B$.
- A variable $x : \alpha$ is **bound** [**free**] in A if there is a bound [free] occurrence of $x : \alpha$ in B .
- A variable $x : \alpha$ is **free** in t if it occurs in t
- A Σ -term or Σ -formula is **open** [**closed**] if are [no] variables free in it.
- A Σ -sentence is a closed Σ -formula.
- t is **free for** $x : \alpha$ **in** A if no free occurrence of $x : \alpha$ in A is within a subformula of A of the form $\forall y : \beta . B$ such that $y : \beta$ is free in t (i.e., **if no variable captures occurs**).

Bound and Free Variables (iClicker)

In which of the following Σ_{nat}^2 -formulas is $x : \mathbb{N}$ is free?

- A. $2 < 4 \wedge 4 < 6$.
- B. $y : \mathbb{N} = z : \mathbb{N}$.
- C. $x : \mathbb{N} = z : \mathbb{N}$.
- D. $\exists x : \mathbb{N} . 0 < x : \mathbb{N}$.
- E. $(\forall x : \mathbb{N} . 0 < x : \mathbb{N}) \wedge (0 < x : \mathbb{N})$.

Closed Formulas (iClicker)

Which of the following Σ_{nat}^2 -formulas is closed?

- A. $x : \mathbb{N} = y : \mathbb{N}$.
- B. $2 + 3 = 6$.
- C. $\forall y : \mathbb{N} . (x : \mathbb{N} + y : \mathbb{N})$.
- D. $(3 = 7) \vee (\forall x : \mathbb{N} . 0 < 7)$.
- E. $\forall x : \mathbb{N} . ((x : \mathbb{N} = y : \mathbb{N}) \wedge (\forall y : \mathbb{N} . x : \mathbb{N} < y : \mathbb{N}))$

Bound and Variables [2/3]

- The **set of free variables** of t , written $\text{fvar}(t)$, is defined by pattern matching as follows:
 1. $\text{fvar}(x : \alpha) = \{x : \alpha\}$.
 2. $\text{fvar}(c) = \emptyset$.
 3. $\text{fvar}(f(t_1, \dots, t_n)) = \text{fvar}(t_1) \cup \dots \cup \text{fvar}(t_n)$.
- The **set of free variables** of A , written $\text{fvar}(A)$, is defined by pattern matching as follows:
 1. $\text{fvar}(s = t) = \text{fvar}(s) \cup \text{fvar}(t)$.
 2. $\text{fvar}(p(t_1, \dots, t_n)) = \text{fvar}(t_1) \cup \dots \cup \text{fvar}(t_n)$.
 3. $\text{fvar}(\neg B) = \text{fvar}(B)$.
 4. $\text{fvar}(A \Rightarrow B) = \text{fvar}(A) \cup \text{fvar}(B)$.
 5. $\text{fvar}(\forall x : \alpha . A) = \text{fvar}(A) \setminus \{x : \alpha\}$.

Bound and Free Variables [3/3]

- The **set of bound variables** of A , written $\text{bvar}(A)$, is defined by pattern matching as follows:
 1. $\text{bvar}(s = t) = \emptyset$.
 2. $\text{bvar}(p(t_1, \dots, t_n)) = \emptyset$.
 3. $\text{bvar}(\neg B) = \text{bvar}(B)$.
 4. $\text{bvar}(A \Rightarrow B) = \text{bvar}(A) \cup \text{bvar}(B)$.
 5. $\text{bvar}(\forall x : \alpha . A) = \text{bvar}(A) \cup \{x : \alpha\}$.

Substitutions [1/2]

- The result of **substituting a Σ -term t of type α for the variable $x : \alpha$ in a Σ -term s** , written $s[x \mapsto t]$, is defined by pattern matching on s as follows:
 1. $(x : \alpha)[x \mapsto t] = t$.
 2. $(y : \beta)[x \mapsto t] = y : \beta$
when x is not y or α is not β .
 3. $c[x \mapsto t] = c$.
 4. $f(t_1, \dots, t_n)[x \mapsto t] = f(t_1[x \mapsto t], \dots, t_n[x \mapsto t])$.

Substitutions [2/2]

- The result of substituting a Σ -term t of type α for the free occurrences of the variable $x : \alpha$ in a Σ -formula A , written $A[x \mapsto t]$, is defined by pattern matching on A as follows:

- $(s_1 = s_2)[x \mapsto t] = (s_1[x \mapsto t] = s_2[x \mapsto t])$.
- $p(t_1, \dots, t_n)[x \mapsto t] = p(t_1[x \mapsto t], \dots, t_n[x \mapsto t])$.
- $(\neg A)[x \mapsto t] = \neg A[x \mapsto t]$.
- $(A \Rightarrow B)[x \mapsto t] = (A[x \mapsto t] \Rightarrow B[x \mapsto t])$.
- $(\forall x : \alpha . A)[x \mapsto t] = (\forall x : \alpha . A)$.
- $(\forall y : \beta . A)[x \mapsto t] = (\forall y : \beta . A[x \mapsto t])$
when x is not y or α is not β .

Substitution (iClicker)

Which of the following substitutions is not correct?

- A. $(0 < x : \alpha)[x \mapsto y : \alpha] = (0 < y : \alpha)$.
- B. $(\forall x : \alpha . 1 = x : \alpha)[x \mapsto y : \alpha] = (\forall x : \alpha . 1 = y : \alpha)$.
- C. $(\forall x : \alpha . 1 = x : \alpha)[x \mapsto y : \alpha] = (\forall x : \alpha . 1 = x : \alpha)$.
- D. $(\forall y : \alpha . 1 = x : \alpha)[x \mapsto y : \alpha] = (\forall y : \alpha . 1 = y : \alpha)$.

Note that a variable capture occurs in D.

3b. MSFOL: Semantics

Mathematical Structures

- Loosely speaking, a **mathematical structure** is a set of mathematical values that are structured in some manner.
- A typical mathematical structure consists of:
 - A finite set of nonempty **domains** (sets) of values: D_1, \dots, D_m .
 - A finite set of **distinguished values** in the domains: a_1, \dots, a_n .
 - A finite set of **functions** whose inputs and outputs are in the domains: f_1, \dots, f_q .
 - A finite set of **relations** over the domains: R_1, \dots, R_r .
- Such a mathematical structure may be written as a tuple: $(D_1, \dots, D_m, a_1, \dots, a_n, f_1, \dots, f_q, R_1, \dots, R_r)$.
- Example:** The integers with addition and multiplication: $(\mathbb{Z}, 0, 1, +, -, *, =, <)$.

Examples of Mathematical Structures

- Number systems.
- Orders.
- Equivalence relations.
- Algebraic structures.
- Lattices.
- Graphs.
- Trees.
- Geometries.
- Topologies.
- Measures.
- Abstract data types (ADTs) used in computing (e.g, ADTs for strings, lists, streams, arrays, records, stacks, and queues).

Admin — February 4

- **Midterm Test 1** will be on Wed, Feb 5, at 7:00–9:00 PM.
 - ▶ Format: Two-stage test, 30 multiple-choice questions
 - ▶ Covers the material from Weeks 01, 02, 03, and 04.
- **Room assignments for Midterm Test 1:**
 - ▶ CNH 104: Last names beginning with A–P.
 - ▶ KTH B135: Last names beginning with Q–Z.
- **Sample test** is posted on Avenue.
- **Review session** on Wed, Feb 5, instead of the lecture.
 - ▶ There will be a regular lecture on Friday.
- **TA help session** on Tue, Feb 4, 3:30-5:00 in ABB 271.
- Office hours: To see me please send me a note with times.
- **Are there any questions?**

Assignment 3

Question 1. Let subformulas : $F_{\Sigma} \rightarrow \mathcal{P}(F_{\Sigma})$ be the function that maps a formula $A \in F_{\Sigma}$ to the set of subformulas of A . Define subformulas by structural recursion using pattern matching.

Question 2. Suppose F is the set of partial and total functions $f : \mathbb{N} \rightarrow \mathbb{N}$.

1. Show that (F, \sqsubseteq) is a weak partial order but not a weak total order.
2. Describe the set of minimal elements of (F, \sqsubseteq) .
3. Describe the set of maximal elements of (F, \sqsubseteq) .
4. Does (F, \sqsubseteq) have a minimum element? If so, what is it?
5. Does (F, \sqsubseteq) have a maximum element? If so, what is it?

Review

- Notational definitions.
- Syntactic conventions.
- Variable binders.
- Bound and free variables.
- Substitutions.
- Mathematical structures.

Finite Signatures as Tuples of Symbols

- Let $\Sigma = (\mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{P}, \tau)$ be a finite signature of MSFOL where:

- $\mathcal{B} = \{\alpha_1, \dots, \alpha_m\}$.
- $\mathcal{C} = \{c_1, \dots, c_n\}$.
- $\mathcal{F} = \{f_1, \dots, f_q\}$.
- $\mathcal{P} = \{p_1, \dots, p_r\}$.

- Σ can then be written as

$$(\alpha_1, \dots, \alpha_m, \\ (c_1 : \tau(c_1)), \dots, (c_n : \tau(c_n)), \\ (f_1 : \tau(f_1)), \dots, (f_q : \tau(f_q)), \\ (p_1 : \tau(p_1)), \dots, (p_r : \tau(p_r)))$$

or, when τ is known, as

$$(\alpha_1, \dots, \alpha_m, c_1, \dots, c_n, f_1, \dots, f_q, p_1, \dots, p_r).$$

Signatures (iClicker)

A signature

$$(\alpha_1, \dots, \alpha_m, c_1, \dots, c_n, f_1, \dots, f_q, p_1, \dots, p_r)$$

is

- A. A language.
- B. A mathematical structure.
- C. An interpretation of a mathematical structure.
- D. Interpreted by a mathematical structure.

Semantics [1/5]

- Let $\Sigma = (\mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{P}, \tau)$ be a signature of MSFOL.
- A **Σ -structure** is a pair $\mathcal{M} = (\mathcal{D}, I)$ where \mathcal{D} is a collection $\{D_\alpha \mid \alpha \in \mathcal{B}\}$ of nonempty domains and I is a function on $\mathcal{C} \cup \mathcal{F} \cup \mathcal{P}$ (called the **interpretation function** of \mathcal{M}) such that:
 - $I(c) \in D_{\tau(c)}$ for each $c \in \mathcal{C}$.
 - $I(f) : D_{\alpha_1} \times \dots \times D_{\alpha_n} \rightarrow D_\alpha$ is an n -ary function for each $f \in \mathcal{F}$ with $\tau(f) = \alpha_1 \times \dots \times \alpha_n \rightarrow \alpha$.
 - $I(p) : D_{\alpha_1} \times \dots \times D_{\alpha_n} \rightarrow \{\mathbf{T}, \mathbf{F}\}$ is an n -ary predicate for each $p \in \mathcal{P}$ with $\tau(p) = \alpha_1 \times \dots \times \alpha_n \rightarrow \mathbb{B}$.

Semantics [2/5]

- A Σ -structure interprets the **symbols** of Σ as **mathematical values**.
- If Σ is
$$(\alpha_1, \dots, \alpha_m, c_1, \dots, c_n, f_1, \dots, f_q, p_1, \dots, p_r),$$
then a Σ -structure $\mathcal{M} = (\mathcal{D}, I)$ can be viewed as the mathematical structure
$$(D_{\alpha_1}, \dots, D_{\alpha_m}, I(c_1), \dots, I(c_n), I(f_1), \dots, I(f_q), I(p_1), \dots, I(p_r))$$
(in which the predicates $I(p_i)$ are considered as relations).
- We need to define the **value** of a Σ -term and a Σ -formula in a Σ -structure.

Σ -Structures (iClicker)

Let \mathcal{M} be a Σ -structure. Which statement is true?

- A. Each Σ -formula has a unique value in \mathcal{M} .
- B. Each open Σ -formula has a unique value in \mathcal{M} .
- C. Each closed Σ -formula (i.e., Σ -sentence) has a unique value in \mathcal{M} .

Semantics [3/5]

- Let $\mathcal{M} = (\mathcal{D}, I)$ be a Σ -structure.
- A **variable assignment** into \mathcal{M} is a function that maps each $x : \alpha$ to an element of D_α (where $x \in \mathcal{V}$ and $\alpha \in \mathcal{B}$).
- Let $\text{assign}(\mathcal{M})$ be the set of variable assignments into \mathcal{M} .
- Given $\varphi \in \text{assign}(\mathcal{M})$, $x \in \mathcal{V}$, $\alpha \in \mathcal{B}$, and $d \in D_\alpha$, let $\varphi[x : \alpha \mapsto d]$ be the $\varphi' \in \text{assign}(\mathcal{M})$ such that $\varphi'(x : \alpha) = d$ and $\varphi'(v) = \varphi(v)$ for all $v \neq x : \alpha$.

Semantics [4/5]

- The **valuation function** for \mathcal{M} is a binary function $V^{\mathcal{M}}$ that satisfies the following conditions for all $\varphi \in \text{assign}(\mathcal{M})$ and all Σ -terms t and Σ -formulas A :
 1. $V_\varphi^{\mathcal{M}}(x : \alpha) = \varphi(x : \alpha)$.
 2. $V_\varphi^{\mathcal{M}}(c) = I(c)$.
 3. $V_\varphi^{\mathcal{M}}(f(t_1, \dots, t_n)) = I(f)(V_\varphi^{\mathcal{M}}(t_1), \dots, V_\varphi^{\mathcal{M}}(t_n))$.
 4. $V_\varphi^{\mathcal{M}}(s = t) = \text{T}$ if $V_\varphi^{\mathcal{M}}(s) = V_\varphi^{\mathcal{M}}(t)$ and $V_\varphi^{\mathcal{M}}(s = t) = \text{F}$ otherwise.
 5. $V_\varphi^{\mathcal{M}}(p(t_1, \dots, t_n)) = I(p)(V_\varphi^{\mathcal{M}}(t_1), \dots, V_\varphi^{\mathcal{M}}(t_n))$.
 6. $V_\varphi^{\mathcal{M}}(\neg A) = \text{F}$ [T] if $V_\varphi^{\mathcal{M}}(A) = \text{T}$ [F].
 7. $V_\varphi^{\mathcal{M}}(A \Rightarrow B) = \text{F}$ if $V_\varphi^{\mathcal{M}}(A) = \text{T}$ and $V_\varphi^{\mathcal{M}}(B) = \text{F}$ and $V_\varphi^{\mathcal{M}}(A \Rightarrow B) = \text{T}$ otherwise.
 8. $V_\varphi^{\mathcal{M}}(\forall x : \alpha . A) = \text{T}$ if $V_{\varphi[x : \alpha \mapsto d]}^{\mathcal{M}}(A) = \text{T}$ for all $d \in D_\alpha$ and $V_\varphi^{\mathcal{M}}(\forall x : \alpha . A) = \text{F}$ otherwise.

Semantics [5/5]

- Let $A \in F_\Sigma$ and $\Gamma \subseteq F_\Sigma$.
- A is **satisfiable** if $V_\varphi^{\mathcal{M}}(A) = \text{T}$ for some Σ -structure \mathcal{M} and $\varphi \in \text{assign}(\mathcal{M})$.
- A is **valid** in \mathcal{M} or \mathcal{M} is a **model** for A , written $\mathcal{M} \models A$, if $V_\varphi^{\mathcal{M}}(A) = \text{T}$ for all $\varphi \in \text{assign}(\mathcal{M})$.
- A is **(universally) valid**, written $\models A$, if A is valid in every Σ -structure.
- \mathcal{M} is a **model** for Γ , written $\mathcal{M} \models \Gamma$, if $\mathcal{M} \models B$ for all $B \in \Gamma$.
- A sentence A is a **semantic consequence** of a set Γ of sentences, written $\Gamma \models A$, if $\mathcal{M} \models \Gamma$ implies $\mathcal{M} \models A$ for all Σ -structures \mathcal{M} .
 - A somewhat more complicated definition is needed when $\Gamma \cup \{A\}$ contains open formulas.
 - **Semantic consequence** is a form of **logical consequence**.

Valid Formulas (iClicker)

Which of the following Σ -formulas is not valid in MSFOL?

- A. $x : \alpha = x : \alpha$.
- B. $p(a) \Rightarrow p(a)$.
- C. $\exists x : \alpha . x = x$.
- D. $\boxed{\exists x, y : \alpha . x \neq y}$.

Notes on Quantifiers

- The universal and existential quantifiers are **duals** of each other:

$$\neg(\forall x : \alpha . A) \Leftrightarrow \exists x : \alpha . \neg A,$$
$$\neg(\exists x : \alpha . A) \Leftrightarrow \forall x : \alpha . \neg A.$$

- Changing the order of quantifiers in a formula usually changes the meaning of the formula!
 - As a rule, $\forall x : \alpha . \exists y : \beta . A \not\equiv \exists y : \beta . \forall x : \alpha . A$.
- In a formula of the form $\forall x : \alpha . \exists y : \beta . A$, the value of the existentially quantified variable y depends on the value of the universally quantified variable $x : \alpha$.
- A universal statement like “All mice are rodents” is formalized as $\forall x : \text{Animal} . \text{mouse}(x) \Rightarrow \text{rodent}(x)$.
- An existential statement like “Some rodents are mice” is formalized as $\exists x : \text{Animal} . \text{rodent}(x) \wedge \text{mouse}(x)$.

3c. MSFOL: Theories

Theories

- A **theory** of MSFOL is a pair $T = (\Sigma, \Gamma)$ where:
 - Σ is a signature of MSFOL.
 - Γ is a set of sentences in F_Σ called the **axioms** of T .
- \mathcal{M} is a **model** for T , written $\mathcal{M} \models T$, if $\mathcal{M} \models \Gamma$.
- A is **valid** in T , written $T \models A$, if $\Gamma \models A$.
- T is **satisfiable** if there is a model for Γ .
- Examples:**
 - Theories of orders, lattices, and boolean algebras.
 - Theories of monoids, groups, rings, and fields.
 - First-order Peano arithmetic (natural number arithmetic).
 - Theory of real closed fields (real number arithmetic).

Example: Theory of Strict Partial Orders

- Let $\Sigma = (\{U\}, \emptyset, \emptyset, \{<\}, \tau)$ where $\tau(<) = U \times U \rightarrow \mathbb{B}$.
- Let Γ be the following set of Σ -sentences:
 1. $\forall x : U . \neg(x < x)$.
 2. $\forall x, y : U . x < y \Rightarrow \neg(y < x)$.
 3. $\forall x, y, z : U . (x < y \wedge y < z) \Rightarrow x < z$.
- Then $T_{\text{spo}} = (\Sigma, \Gamma)$ is a theory of MSFOL called the **theory of strict partial orders**.
- The models of T_{spo} are all possible strict partial orders.
- If we can show $T_{\text{spo}} \models A$, then we know that A is valid in every strict partial order!

Admin — February 7

- Midterm Test 1.
 - ▶ Marks will be posted next week.
 - ▶ Solutions will be posted next week.
- Office hours: To see me please send me a note with times.
- Are there any questions?

Review

- MSFOL syntax
 - ▶ Signatures
 - ▶ Σ -terms
 - ▶ Σ -formulas
 - ▶ Bound and free variables.
 - ▶ Substitution.
- MSFOL semantics.
 - ▶ Σ -structures.
 - ▶ Variable assignments.
 - ▶ Valuation function $V_{\varphi}^{\mathcal{M}}$.
 - ▶ A formula is satisfiable, valid in \mathcal{M} , and valid.
 - ▶ A formula is a semantic consequence of a set of formulas.
- MSFOL theories.
 - ▶ Signature and a set of axioms.
 - ▶ Model for T , formula valid in T , T is satisfiable.

Theories (iClicker)

Suppose \mathcal{M} is a model of T_{spo} . Which of the following statements is true?

- A. \mathcal{M} is a weak partial order.
- B. \mathcal{M} is a strict partial order.
- C. \mathcal{M} is not a strict total order.
- D. \mathcal{M} is a strict total order iff
$$T_{\text{spo}} \models \forall x, y : U . x < y \vee y < x \vee x = y.$$

Theories (iClicker)

Every theory has a model. Is this statement true or false?

- A. True.
- B. ☒ False.

Example: Theory of Monoids

- Let $\Sigma = (\{M\}, \{e\}, \{*\}, \emptyset, \tau)$ where $\tau(e) = M$ and $\tau(*) = M \times M \rightarrow M$.
- Let Γ be the following set of Σ -sentences:
 1. $\forall x, y, z : M . (x * y) * z = x * (y * z)$.
 2. $\forall x : M . e * x = x$.
 3. $\forall x : M . x * e = x$.
- Then $T_{\text{mon}} = (\Sigma, \Gamma)$ is a theory of MSFOL called the **theory of monoids**.
- Examples of models of T_{mon} :
 1. $(\mathbb{N}, 0, +)$, $(\mathbb{N}, 1, *)$, $(\mathbb{Z}, 0, +)$, etc.
 2. (String, "", concatenation).
 3. (IntegerList, [], append).
 4. $(U \rightarrow U, \text{identity-function}, \text{function-composition})$.

Theory Extensions (iClicker)

Let $T = (\Sigma, \Gamma)$ and $T' = (\Sigma, \Gamma')$ be theories of MSFOL. Which of the following statements is true if $\Gamma \subseteq \Gamma'$ (i.e., if T' is an **extension** of T).

- A. Every model of T is a model of T' .
- B. ☒ Every model of T' is a model of T .
- C. If T is satisfiable, then T' is satisfiable.
- D. ☒ If T' is satisfiable, then T is satisfiable.

First-Order Peano Arithmetic

- Let $\Sigma = (\{\mathbb{N}\}, \{0\}, \{S, +, *\}, \emptyset, \tau)$ where $\tau(0) = \mathbb{N}$, $\tau(S) = \mathbb{N} \rightarrow \mathbb{N}$, and $\tau(+) = \tau(*) = \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$.
- Let Γ be the following set of Σ -formulas:
 1. $\forall x : \mathbb{N} . 0 \neq S(x)$.
 2. $\forall x, y : \mathbb{N} . (S(x) = S(y) \Rightarrow x = y)$.
 3. $\forall x : \mathbb{N} . x + 0 = x$.
 4. $\forall x, y : \mathbb{N} . x + S(y) = S(x + y)$.
 5. $\forall x : \mathbb{N} . x * 0 = 0$.
 6. $\forall x, y : \mathbb{N} . x * S(y) = (x * y) + x$.
 7. Each universal closure A of a formula of the form
$$(B[x \mapsto 0] \wedge (\forall x : \mathbb{N} . B \Rightarrow B[x \mapsto S(x)])) \Rightarrow \forall x : \mathbb{N} . B$$
where B is a Σ -formula.
- Then $T_{\text{pa}} = (\Sigma, \Gamma)$ is a theory of MSFOL called **first-order Peano arithmetic**.

Theory vs. Model

- A mathematical structure is a **concrete** mathematical model.
- A theory is an **abstract** mathematical model.
- A theory can be viewed as a specification of its models.
 - ▶ A theory is to a model as a specification is to an implementation.
- Theories fall into two categories:
 - ▶ Those intended to describe a **single model** (e.g., **first-order Peano arithmetic**).
 - ▶ Those intended to describe a **collection of models** (**theory of monoids**).

3d. MSFOL: Proof Systems

Proof Systems

- A **proof system** for MSFOL is a set of **axioms** and **rules of inference** for constructing **proofs** that say a particular formula A follows from a particular set Γ of premises.
- Let P be a proof system for MSFOL.
- A is **syntactic consequence** of Γ in P , written $\Gamma \vdash_P A$, if there is a proof of A from Γ in P .
- A is a **theorem** in P , written $\vdash_P A$, if there is a proof of A from \emptyset in P .
- Γ is **consistent** in P if not every formula is a syntactic consequence of Γ in P .
- P is **sound** if $\Gamma \vdash_P A$ implies $\Gamma \models A$.
- P is **complete** if $\Gamma \models A$ implies $\Gamma \vdash_P A$.
- If $T = (\Sigma, \Gamma)$ is a theory of MSFOL, then A is a **theorem** of T in P , written $T \vdash_P A$, if $\Gamma \vdash_P A$.

4. Summary

Truth vs. Proof

Semantics	Syntax
truth	proof
semantic consequence	syntactic consequence
A is valid $\models A$	A is a theorem in P $\vdash_P A$
A is valid in T $T \models A$	A is a theorem of T in P $T \vdash_P A$
T is satisfiable	T is consistent in P

- Semantic consequence and syntactic consequence are different forms of logical consequence.
- The semantic and syntactic notions are equivalent in the most common logics:
 - ▶ Propositional logic (Bernays, 1918).
 - ▶ First-order logic (Gödel, 1930).
 - ▶ Simple type theory (Henkin, 1950)

Mathematical Problems: Fundamental Form

- Most mathematical problems can be expressed as statements of the form
$$T \models A$$
where T is a theory and A is a sentence.
- There are three basic ways of deciding whether or not $T \models A$:
 1. Model checking:
Show that $\mathcal{M} \models A$ for each model \mathcal{M} of T .
 2. Proof:
Show $T \vdash_P A$ for some sound proof system P .
 3. Counterexample:
Show $\mathcal{M} \models \neg A$ for some model \mathcal{M} of T .