Mathematical Logic

Lecture 2

Harbin, 2023

Intro

Mathematical logic is the study of mathematical reasoning. We do this by developing an abstract model of the process of reasoning in mathematics. We then study this model and determine some of its properties.

Mathematical reasoning is deductive; that is, it consists of drawing (correct) inferences from given or already established facts. Thus the basic concept is that of a statement being a logical consequence of some collection of statements.

In our model of mathematical reasoning we will need to precisely define logical consequence.

Intro

Definition 1. Let $\mathcal{A}$ be any abstract set. We call $\mathcal{A}$ an alphabet.
Finite sequences of elements of $\mathcal{A}$ are called words in $\mathcal{A}$.
Finite sequences of words are called texts.
The length of the word ω, denoted $lh(\omega)$, is the length of ω as a sequence of symbols.

Examples.
1.) Consider the alphabet $\mathcal{A} = \{a, b, c\}$. Then $baaac$ is a word in the alphabet $\mathcal{A}$.
2.) A word that does not contain any character (that is, a sequence of length 0) is called an empty word and is denoted by ϵ.

Definition 2. If α and β are words in the alphabet $\mathcal{A}$, then the word $\alpha\beta$ (the result of adding the word β to the end of the word α) is called the concatenation of the words α and β.

Remark. If α is a word and $n \in \mathbb{N}$, then α^n stands for the word $\underbrace{\alpha\alpha\ldots\alpha}_{n}$.

The set of all words in the alphabet $\mathcal{A}$ is denoted by $\mathcal{A}^*$.
Here is a simple lemma on the cardinality of $\mathcal{A}^*$.

Proposition 1.

If the alphabet $\mathcal{A}$ is finite or countable, then the set $\mathcal{A}^*$ is countable.

Proof. Indeed, for any finite subset $\mathcal{A}_0 \subseteq \mathcal{A}$, the set of all words of fixed length in the alphabet $\mathcal{A}_0$ is finite. Therefore, $\mathcal{A}_0^*$ is the union of a countable number of finite sets, and hence so is the set $\mathcal{A}^* = \bigcup_{\mathcal{A}_0 \subseteq \mathcal{A}} \mathcal{A}_0^*$. ■

We are going to restrict ourselves to countable alphabets.

First-Order Languages

Now we consider the alphabets of formal languages of a particular kind: the vocabularies of first-order languages.

Definition 3. The vocabulary or alphabet of a first-order language contains the following symbols (and only the following symbols):

1. $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$
2. $\forall, \exists$
3. ("variables") $v_0, v_1, \ldots$
4. $\equiv$
5. for every $n \geq 1$ a (possibly empty) set of $n-$ary predicates P_i^n $(i \in \mathbb{N})$
6. for every $n \geq 1$ a (possibly empty) set of $n-$ary function signs f_i^n $(i \in \mathbb{N})$
7. a (possibly empty) set of constants c_i $(i \in \mathbb{N})$ and parentheses as auxiliary symbols.

Note:

• $\mathcal{A} = \{(1); (2); (3); (4)\}$ is fixed: these symbols are contained in every first-order language.

• $\mathcal{S} = \{(5); (6); (7)\}$ is optional: the choice of $\mathcal{S}$ determines the specific character of a first-order language.

• $\mathcal{A}_{\mathcal{S}} = \mathcal{A} \bigcup \mathcal{S}$ is the actual alphabet of the first-order language that is determined by $\mathcal{S}$.

Examples. 1.) $\mathcal{S}_{Equ} = \{\underbrace{\sim}_{P_0^2}\}$ determines the first-order language of equivalence structures.

2.) $\mathcal{S}_{Gr} = \{\underbrace{\circ}_{f_0^2}, \underbrace{e}_{c_0}\}$ determines the first-order language of group theory.

Remark. Let $\mathcal{S}$ be the specific symbol set of a first-order language (such that $\mathcal{A}_{\mathcal{S}} = \mathcal{A} \bigcup \mathcal{S}$ is the alphabet of that language): $\mathcal{A}_{\mathcal{S}}$ is countable. By Proposition 1, $\mathcal{A}_{\mathcal{S}}^*$ is also countable.

Terms and Formulas

We are going to build up well-formed texts in a step-by-step manner:

Definition 4. Let $\mathcal{S}$ be the specific symbol set of a first-order language. $\mathcal{S}-$terms are precisely those words over $\mathcal{A}_{\mathcal{S}}$ that can be generated according to the following rules:

(T1) Every variable is an $\mathcal{S}-$term.

(T2) Every constant in $\mathcal{S}$ is an $\mathcal{S}-$term.

(T3) If $t_1, \ldots, t_n$ are $\mathcal{S}-$terms and f is an $n-$ary function sign in $\mathcal{S}$, then $f(t_1, \ldots, t_n)$ is an $\mathcal{S}-$term.

We denote the set of $\mathcal{S}-$terms by $\mathcal{T}_{\mathcal{S}}$.

Terms and Formulas

This means:
$t \in \mathcal{A}_{\mathcal{S}}^*$ is an $\mathcal{S}-$term if there is a sequence $u_1, \ldots, u_k$of elements of $\mathcal{A}_{\mathcal{S}}^*$, such that $u_k = t$ and for all u_i with $1 \leq i \leq k$ it is the case that:

- u_i is a variable or
- u_i is a constant in $\mathcal{S}$ or
- $u_i = f(t_1, \ldots, t_n)$ and $t_1, \ldots, t_n \in \{u_1, \ldots, u_{i-1}\}$.

Example. Let $\mathcal{S} = \{f, g, c\}$ where f, g are binary function signs, c is a constant. It follows that $g(f(c, v_0), c)$ is an $\mathcal{S}-$term:

c (T2)
v_0 (T1)
$f(c, v_0)$ (T3)
$g(f(c, v_0), c)$ (T4)

Terms and Formulas

Definition 5. Let $\mathcal{S}$ be the specific symbol set of a first-order language. $\mathcal{S}-$formulas are precisely those strings over $\mathcal{A}_{\mathcal{S}}$ that can be generated according to the following rules:

(F1) $\equiv (t_1, t_2)$ (for $\mathcal{S}-$terms t_1, t_2)
(F2) $P(t_1, \ldots, t_n)$ (for $\mathcal{S}-$terms $t_1, t_2, \ldots, t_n$, for n-ary $P \in \mathcal{S}$)
(Formulas which can be generated solely on basis of (F1) and (F2) are called atomic.)
(F3) $\neg\phi$
(F4) $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \to \psi$, $\phi \leftrightarrow \psi$
(F5) $\forall x\phi$, $\exists x\phi$ (for arbitrary variables x)

We denote the set of $\mathcal{S}-$formulas by $\mathcal{F}_{\mathcal{S}}$.

Example. Let $\mathcal{S} = \{R\}$ where R is a binary predicate. It follows that

$$\Big((R(v_0, v_1) \wedge R(v_1, v_2)) \to R(v_0, v_2)\Big)$$

is an $\mathcal{S}-$formula:

- $R(v_0, v_1), R(v_1, v_2), R(v_0, v_2)$ by (F2)
- $R(v_0, v_1) \wedge R(v_1, v_2)$ by (F4)
- $\Big((R(v_0, v_1) \wedge R(v_1, v_2)) \to R(v_0, v_2)\Big)$ by (F4)

Lemma 2.

For all symbol sets $\mathcal{S}$, $\mathcal{T}_{\mathcal{S}}$ and $\mathcal{F}_{\mathcal{S}}$ are countable.

Proof. We have already seen that $\mathcal{A}_{\mathcal{S}}^*$ is countable. Since $\mathcal{T}_{\mathcal{S}}, \mathcal{F}_{\mathcal{S}} \subseteq \mathcal{A}_{\mathcal{S}}^*$, it follows that $\mathcal{T}_{\mathcal{S}}, \mathcal{F}_{\mathcal{S}}$ are countable, too.■

At some points we will need to sort out those variables which occur freely in a formula, i.e., which occur not bound by any quantifier:

Definition 6. Let $\phi \in \mathcal{F}_{\mathcal{S}}$ (for arbitrary symbol set $\mathcal{S}$): $free(\phi)$, i.e., the set of variables which occur freely in ϕ, can be defined recursively as follows:

- $free(t_1 \equiv t_2) := var(t_1) \cup var(t_2)$ (let $var(t)$ be the set of variables in t)
- $free(P(t_1, \ldots, t_n) = var(t_1) \cup \ldots \cup var(t_n)$
- $free(\neg\phi) := free(\phi)$
- $free(\phi \wedge \psi) := free(\phi) \cup free(\psi)$ (analogously for $\vee, \rightarrow, \leftrightarrow$)
- $free(\forall\phi x) := free(\phi) \setminus \{x\}$ (analogously for $\exists$)

Example.

1.) $free(\forall x(P(y) \to Q(y))) = \{y\}$

2.) $free(\exists x P(x)) = \varnothing$

3.) $free((\exists x P(x) \wedge P(x))) = \{x\}$

We see that according to our definition, $free(\phi) = \varnothing$ is the set of variables that occur freely at some place within ϕ.

A formula ϕ without free variables, i.e., for which $free(\phi) = \varnothing$, is called a sentence.

Methods

Remark. This is simply a version of proof by complete induction over natural numbers. One actually shows by complete induction over n :
For all $n \in \mathbb{N}$, for all Ξ, if Ξ is derivable in the term-/formula-calculus in n steps, then Ξ has the property P.

Example. All terms in $\mathcal{T}_{\mathcal{S}}$ contain as many opening parentheses as they contain closing parantheses; i.e.: all terms Ξ have the property P.
Proof. Induction basis: Variables have the property P (since they do not contain parentheses at all). Constants have the property P (they do not contain parentheses at all).
Inductive assumption: Assume terms $t_1, \ldots, t_n$ have the property P. So t_1 contains as many opening parentheses as closing parentheses, t_2 contains as many opening parentheses as closing parentheses, ... But then also $f(t_1, \ldots, t_n)$ does so (where f is an arbitrary function sign in $\mathcal{S}$).■

Now let us consider a slightly more complex application of induction over terms:

We call $X \subseteq \mathcal{A}_{\mathcal{S}}^*$ closed under the rules of the term calculus (for a given symbol set $\mathcal{S}$) if and only if:

1.) all variables are contained in X
2.) all constants in $\mathcal{S}$ are contained in X
3.) if $t_1, \ldots, t_n$ are in X, then $f(t_1, \ldots, t_n)$ is in X (where f is an arbitrary function sign in $\mathcal{S}$).

Examples.
(I) $\mathcal{A}_{\mathcal{S}}^*$ is closed under the rules of the term calculus.
(II) $\mathcal{T}_{\mathcal{S}}$ is closed under the rules of the term calculus.

Lemma 3.

$$\mathcal{T}_{\mathcal{S}} = \bigcap_{X \text{ closed}} X$$

Thus, $\mathcal{T}_{\mathcal{S}}$ is the least subset of $\mathcal{A}_{\mathcal{S}}^{*}$ that is closed under the rules of the term calculus.

Proof. Let X is closed under the rules of the term calculus. Firstly note that all variables and constants are contained in X. Assume $t_1, \ldots, t_n \in \mathcal{T}_{\mathcal{S}}$ are contained in X, then $f(t_1, \ldots, t_n)$ in X. Thus, $\mathcal{T}_{\mathcal{S}} \subseteq X$ for any such X, from which $\mathcal{T}_{\mathcal{S}} \subseteq \bigcap_{X \text{ closed}} X$. Variables and constants are in $\mathcal{T}_{\mathcal{S}}$; now suppose that $t_1, \ldots, t_n$ are in $\mathcal{T}_{\mathcal{S}} \Rightarrow$ there are term derivations of the form

$$u_1^1, \ldots, u_{k_1}^1 \quad \text{with } u_{k_1}^1 = t_1$$

$$\ldots\ldots\ldots\ldots$$

$$u_1^n, \ldots, u_{k_n}^n \quad \text{with } u_{k_n}^n = t_n$$

But then $u_1^1, \ldots, u_{k_1}^1, u_1^2, \ldots, u_{k_2}^2, \ldots, u_1^n, \ldots, u_{k_n}^n, f(t_1, \ldots, t_n)$ is a derivation of $f(t_1, \ldots, t_n)$ in the term calculus $\Rightarrow f(t_1, \ldots, t_n) \in \mathcal{T}_{\mathcal{S}}$

Thus $\mathcal{T}_{\mathcal{S}}$ is itself closed under the rules of the term calculus, whence $\mathcal{T}_{\mathcal{S}} \supseteq \bigcap_{X \text{ closed}} X$ ■

Consider the most important first-order languages: the Zermelo–Fraenkel language of set theory $L_1 Set$, and the Peano language of arithmetic $L_1 Ar$.

Translation from $L_1 Set$ to "ordinary language".

$\forall x(\neg(x \in \varnothing))$:

"for all (sets) x it is false that x is an element of (the set) $\varnothing$"

$\forall x(y \in z)$:

The literal translation "for all x it is true that y is an element of z" sounds a little strange. Later we shall see that from the point of view of "truth" or "deducibility," such a formula is equivalent to the formula $y \in z$.

Translation from argot to L_1Ar.

$x < y : \exists z(y = (x + z) + 1)$. Here the variables are names for nonnegative integers.

"x is a divisor of y": $\exists z(y = xz)$.

Remark (Higher-order languages). Let L be any first-order language. Its modes of expression are limited in principle by one important consideration: we are not allowed to speak of arbitrary properties of objects of the theory, that is, arbitrary subsets of the set of all objects.

Syntactically, this is reflected in the prohibition against forming expressions such as $\forall p(p(x))$, where p is a relation of degree 1; relations must stand for fixed rather than variable properties.

Of course, certain properties can be defined using nonatomic formulas. For example, in L_1Ar instead of "x is even" we may write $\exists y(x = (1 + 1)y)$. However, there is a continuum of subsets of the integers but only a countable set of definable properties so there are automatically properties that cannot be defined by formulas. Thus, it is impossible to replace the forbidden expression $\forall p(p(x))$ by a sequence of expressions $P_1(x), P_2(x), \ldots$.

Languages in which quantifiers may be applied to properties and/or functions (and also, possibly, to properties of properties, and so on) are called higher-order languages.

Additional Example-exercise: Prove that the following strings are $\mathcal{S}-$terms (for given $\mathcal{S}$ with $c, f, g \in \mathcal{S}$, where f is a binary function sign, g is a unary function sign, x and y are variables):

(a) $f(x, c)$

(b) $g(f(x, c))$

(c) $f(f(x, c), f(x, f(x, y)))$

Proof. (a) By (T1), (T2) and (T3).

(b) By (a) and (T3).

(c) By (a), (T1), and (T3) thrice.

Exercises

Exercise I. Prove that the following strings are $\mathcal{S}-$formulas (for given $\mathcal{S}$ with $c, f, g, P, Q \in \mathcal{S}$, where f is a binary function sign, g is a unary function sign, x and y are variables, P is a unary predicate and Q is a binary predicate):
(a) $\neg P(f(x, c))$
(b) $\exists x \forall y(P(g(f(x, c))) \rightarrow Q(y, y))$

Exercise II. Prove by induction: the string $\forall x f(x, c)$ is not an $\mathcal{S}-$term (where $\mathcal{S}$ is an arbitrary symbol set).

Exercises

Exercise I. Prove that the following strings are $\mathcal{S}-$formulas (for given $\mathcal{S}$ with $c, f, g, P, Q \in \mathcal{S}$, where f is a binary function sign, g is a unary function sign, x and y are variables, P is a unary predicate and Q is a binary predicate):
(a) $\neg P(f(x, c))$
(b) $\exists x \forall y(P(g(f(x, c))) \rightarrow Q(y, y))$

Exercise II. Prove by induction: the string $\forall x f(x, c)$ is not an $\mathcal{S}-$term (where $\mathcal{S}$ is an arbitrary symbol set).