Foundations of Mathematics, Lecture 3

András Kornai

BMETE91AM35 Fall 2023-24

Operations

- Well, what are operations? Operations are like addition, multiplication, negation. . . How can we define operations?
- We don't need new machinery! *Binary* operations are **functions** with two variables. *Unary* operations are functions with one variable (minus, reciprocal, . . .) *Nullary operations* are functions that don't depend on any variable, **constants**.
- A **structure** is a set S and some operations. For example groups have a nullary operation (the unit e), a unary operation $(^{-1})$, and a binary operation (multiplication) which satisfy some identities (group axioms). On occasion, we don't insist that an operation be everywhere defined
- One set of operations that matters in PL are the Boolean $\neg, \wedge, \vee$
- These are 'truth functional' – only the truth of the operands matters for establishing the truth of the result

Boolean operations

- When there is a base set (such as the set of integers) everything works!
- We get the de Morgan identities: $\overline{A \cup B} = \overline{A} \cap \overline{B}$ and $\overline{A \cap B} = \overline{A} \cup \overline{B}$
- Complementation is an involution $\overline{\overline{A}} = A$
- $\cup$ and $\cap$ are commutative, associative, idempotent
- Two kinds of distributive laws
- There is a 0 (the empty set) and a 1 (the universe)
- Will be generalized to *lattices* later on

Trouble with negation

- There is no "set of all sets"
- Why not? Because it would not be well founded (doable with AFA, but not with ZFC)
- Because it would give rise to Russel's Paradox: by Comprehension we could form the set of all sets that don't contain themselves!
- Paradox can be avoided by (a) positive comprehension (b) type theory
- Most math is done with (b) – types are built into ZFC
- Strongly related to type checking in programming languages
- You can have the *class* of all sets, and other classes (NGB set theory)

Subscripting, indexing

- The basic idea: instead of $A, B, C, \ldots, Z$ and running out of letters after 26, let's do $A_1, A_2, \ldots, A_{777}$ because we never run out of numbers
- But what if we do? There are things for which we don't have enough numbers, e.g. points in the interval (0,1)
- Let S be a set of indexes, with members α. (Note that α is a *variable* in this usage.) Further, for each member of S let us assume we already have some set A_α. We define $\bigcup_\alpha A_\alpha$ and $\bigcap_\alpha A_\alpha$ exactly how?
- The special cases: when the index set is empty
- What are variables?
- What are families of sets?

Partitioning

- A *partition* of a set A is a family of sets A_α such that for any $\alpha \neq \beta$ we have $A_\alpha \cap A_\beta = \emptyset$ and $\bigcup_\alpha A_\alpha = A$. By definition, we never consider the empty set a part of any partitition, so in the definition we may write "a family of *nonempty* sets"
- The partition can be finite (e.g. the sets of *even* and *odd* numbers partition the set of integers) or infinite
- There are two *trivial* partitions, when all elements are in the same set, and when all go in their own set
- Partitions are 1-1 related to *equivalence relations*

Functions as relations

- All functions are relations, but not all relations are functions!
- The big difference is that functions have *unique output*, a relation F will be called a function only if $aFb \wedge aFc \Rightarrow b = c$
- Definitions of domain, codomain, range, and composition are the same. We don't write 30° cos $\sqrt{3}/2$, we write $\cos 30^\circ = \sqrt{3}/2$
- CPZ devotes Chapter 10 to functions, we will cover this in class today, but the entire chapter is homework to read. Exercises similar to those in CPZ Ch 1, 9, and 10 will be on the midterm
- Composition of functions is just like composition of relations: if $f : A \to B$ and $g : B \to C$ then $g \circ f : A \to C$
- Sometimes (often) more lax terminology is used, permitting functions to be defined only on a subset of their domain. For example, most people will talk about $\sqrt{\ }$ as an $\mathbb{R} \to \mathbb{R}$ function, even though its *natural domain* is only $\mathbb{R}_0^+$
- Other tricky point about $\sqrt{\ }$ is that output (depending on definition) is not unique

Main function types

- Functions are *defined* or *given* by their graphs, which are the set of (input, output) pairs. But we often think of functions as little machines that take some input and produce some output
- The input may be of different type than the output. Examples: distance travelled as a function of time; temperature as a function of space; force of gravity as a function of masses and distance, ...
- **Multivariate** functions don't depend on a single variable but several. For example, current is a function of both voltage and resistance (Ohm's Law)
- **Vector-valued** sometimes functions produce a k-tuple of values simultaneously. For example, at any given point in space gravity has both a magnitude and a direction (total of four numbers)
- These can happen at the same time: functions from n-tuples to k-tuples are often used

Variety of functions

- The central types are **numerical functions** from numbers to numbers. You will be seeing a lot of examples of *arithmetic* functions: domain $\mathbb{N}$ but range can be $\mathbb{R}$ or even $\mathbb{C}$
- Also very frequent are **real functions** with domain and range $\mathbb{R}$
- You will love **complex functions** with domain and range $\mathbb{C}$
- **Functionals** are functions whose domain are functions, and range is typically $\mathbb{R}$ or $\mathbb{C}$
- **Operators** are functions from functions to functions
- All of these are heavily used in physics/engineering
- But there is more! Not all functions involve numbers, for example the truth function maps formulas onto the set {true, false }
- We will also have a lot to say about **operations** in algebra

Main properties of functions

1. Injective: different x-es map on different y-s: $f(x) = f(y) \Rightarrow x = y$
2. Surjective: codomain = range (codomain $\supset$ range is true by definition)
3. Bijective: both injective and surjective
4. **Theorem:** a function f is *invertible* $\Leftrightarrow$ it is *bijective*
5. **Proof:** We need to prove both $\Rightarrow$ and $\Leftarrow$. For $\Rightarrow$ we need to *verify* that the bijective properties follow from invertibility. For $\Leftarrow$ we will construct the inverse of a bijective function.
6. ($\Rightarrow$) What do we suppose? What do we need to prove?
7. ($\Leftarrow$) What do we suppose? What do we need to prove?
8. Discussion of CPZ Ex 10.18

The Peano axioms

- $1 \in \mathbb{N}$
- $\forall i \in \mathbb{N}\ \exists! i' \in \mathbb{N}$
- $\nexists i \in \mathbb{N}\ i' = 1$
- $i' = j' \rightarrow i = j$
- $\forall P \subset \mathbb{N}\ 1 \in P \wedge (i \in P \rightarrow i' \in P) \rightarrow P = \mathbb{N}$

The usual $\mathbb{N}$

- Succession is just a technical device, what we want are the standard arithmetic operations (addition, subtraction, multiplication, division)
- These will be *defined*. Addition is defined inductively: (i) $x + 1 = x'$ (ii) $x + y' = x + y + 1$
- By convention we call 1+1 2, we call 2+1 3, etc.
- HW4.1 Prove 2+3=3+2
- HF4.2–7 CPZ 2.21–2.26
- HF4.8 Use the definition of addition to define multiplication

Logic

- The twin pillars of the foundation are set theory and logic
- We already started to build relations and functions on set theory, and we will continue with operations and structures. But now we turn to logic
- Set theory has variants (ZFC, NGB, KP . . .), logic has many more variants!
- To define a logic we will need four things:
  1. A language to write formulas
  2. A notion of truth
  3. A notion of what the formulas mean ‘model theory’
  4. A deduction procedure ‘proof theory’
- We will cover each in turn, starting with ‘language’
- We have a set Σ called the *alphabet*, it’s elements are called *letters*
- Putting letters one after the other we obtain *strings*

Rudiments of formal language theory

- Given an alphabet Σ, the set of all strings formed from these is denoted Σ^*. There is a special element λ called the *empty string*.
- Length of λ is 0, length of $a \in \Sigma$ is 1, length of α denoted $|\alpha|$ satisfies $|\alpha\beta| = |\alpha| + |\beta|$
- The main operation on strings is *concatenation* (writing them is sequence). For example, if $\alpha = abc$ and $\beta = AB$ then $\alpha\beta = abcAB$
- Concatenation is *not* commutative, $\beta\alpha = ABabc \neq \alpha\beta$
- We abbreviate $\alpha\alpha$ as α^2, similarly for α^3 etc.
- A **language** over the alphabet Σ is a subset of Σ^*