Foundations of Mathematics, Lecture 6 (incl. midterm)

András Kornai

BMETE91AM35 Fall 2020-21

Organization of this class

- First half (12:15 – 13.00) lecture. The material covered will be relevant for the rest of the course, but it is obviously not part of the midterm
- Second half (13:00 – 13:45) Midterm. Use paper and pencil. You need to take a picture of your solutions and email it to me `kornai@math.bme.hu`
- Within the next 24 hours you can submit LaTeXversion of your solutions for extra points (but you can get 100% without this step)
- Midterm will be screenshared, and also available at the course website (don't look now, I will put it up at 1PM)

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^*
- Since languages are sets, it is meaningful to speak of their union, intersection, and complement (relative to Σ^*)
- The **product** of languages R and S, written RS, is $\{\alpha\beta | \alpha \in R, \beta \in S\}$
- The set $\cup_{i=0}^{\infty} R^i$ is written R^* and is called the **Kleene closure** of R.

Main types of formal languages

- The simplest formal languages are the ones where we list all members explicitly, e.g. $L = \{a, ab, ba, baab\}$
- **Regular** or **rational** languages are built from these using only Boolean and Kleene operations
- The most complex formal languages we can deal with are called **recursively enumerable**. For each of these, there is an algorithm (Turing machine) that halts whenever the string $\alpha \in L$ is where we start the computation
- Difference between recursively enumerable and recursive
- We can *prove* the existence of languages that are not r.e. but we cannot *construct* any!
- Summary: there are actually more complex formal languages than we can deal with

Between the extremes

1. Between the extremely simple regular (Type 3) and the extremely complex recursively enumerable (Type 0) languages there are intermediary types
2. Linear bounded languages (context-sensitive languages) are Type 1
3. These require as much memory as the size of the input, but not more. These will be sufficient for logic
4. Context-free languages (Type 2) are even less powerful, sufficient for most of logic, except quantifier scope
5. Mildly context-sensitive languages (Type 1.5) include indexed grammars, linear indexed grammars
6. Complexity of the languages used in logic somewhere between 1 and 1.5