MATH3306

Set Theory & Mathematical Logic

Semester 2, 2022

Professor Benjamin Burton

Rohan Boas

Contents

1 Introductory notes

1.1 Gödel's incompleteness theorem

Theorem 1.1.1 (Gödel's incompleteness theorem, informal version)**.** *There are true mathematical statements that cannot be proven.*

"Proof". Take the statement "This statement has no proof."
Assume it is false. This implies that the statement has a proof. If the statement has a proof, it must be true, contradiction!
Assume it is true. This implies that the statement has no proof. Therefore, the statement cannot be proven. □

1.2 The halting problem

We would like to be able to know if an algorithm or program will halt or will loop forever. Can we write an algorithm which can tell us whether or not a given program will halt on a given input? This is known as the halting problem. The halting problem is undecidable.

Proof by contradiction. Say there does exist some program H which solves the halting problem. Let us define H. H takes as inputs a program, x, and an input for that program, y.

$$\mathrm{H}(x,y) = \begin{cases} \text{YES}, & \text{if } x \text{ halts on input } y \\ \text{NO}, & \text{if } x \text{ loops forever on input } y \end{cases}$$

Let us define a program Foo.

$$\mathrm{Foo}(x) = \begin{cases} \text{loops forever}, & \text{if } \mathrm{H}(x,x) \text{ is YES} \\ \text{halts}, & \text{if } \mathrm{H}(x,x) \text{ is NO} \end{cases}$$

TODO: finish proof, maybe rewrite with diagram □

1.3 Defining algorithms

In defining algorithms, Turing machines and recursive functions will be the primary focus. Grammars and code are also alternatives.

Definition 1.1 (Church-Turing thesis, informal version)**.** Any reasonable definitions of "algorithm" are equivalent.

2 Finite and deterministic state automata

2.1 Finite state automata

Definition 2.1 (Alphabet)**.** An alphabet A is a finite set of symbols.

Definition 2.2 (Word)**.** A word is a sequence of symbols from A.

Theorem 2.1.1. *Words can be concatenated. E.g. for words α representing "bob", and β representing "cat", $\alpha\beta\alpha$ represents "bobcatbob".*

Theorem 2.1.2. *The set of words length m is $A^m = A \times A \times ... \times A$.*

Theorem 2.1.3. *The empty word (i.e. of length 0) is ε.*

Theorem 2.1.4. *A^* is the set of all words over A. $A^* = \bigcup_{m \geq 0} A^m$.*

Theorem 2.1.5. *A^+ is the set of all non-empty words over A. $A^+ = \bigcup_{m \geq 1} A^m$.*

Definition 2.3 (Language)**.** A language is a subset of A^*

Definition 2.4 (Finite state automata)**.** A finite state automaton (FSA) can be defined as the 5-tuple (Q, F, A, τ, q_o) where

- Q is a finite set of states,
- $F \subseteq Q$ is the set of final/accepting states,
- A is a finite alphabet,
- $\tau \subseteq Q \times A \times Q$ is the set of transitions, and
- q_0 is the initial state.

Definition 2.5 (Computation)**.** A computation is a sequence $q_0 a_1 q_1 a_2 ... a_n q_n$ such that each $q_i a_{i+1} q_{i+1} \in \tau$.

Definition 2.6 (Successful)**.** A computation is successful if $q_n \in F$.

Definition 2.7 (Accepted)**.** A word $\alpha = a_1 a_2 ... a_n$ is accepted by the FSA if there is a successful computation $q_0 a_1 q_1 a_2 ... a_n q_n$.

Definition 2.8 (Recognised)**.** The language recognised by an FSA is the set of all words it accepts.

2.2 Deterministic state automata

Definition 2.9 (Deterministic state automata)**.** A deterministic state automaton (DSA) is an FSA where $\forall q \in Q, \forall a \in A, \exists! q' \in Q$ s.t. $(q, a, q') \in \tau$

Theorem 2.2.1. *The definition of DSA implies there exists a function $\delta : Q \times A \to A$.*