9 Decidability

9.1 Hilbert’s tenth problem

- The Hilbert’s tenth problem (proposed in 1900 among a list of 23 open problems for the new century): Devise a procedure with a finite number of operations that tests whether a polynomial has an integral root. What Hilbert meant by “a process with a finite number of operations” is an algorithm.
- Formulating Hilbert’s problem with today’s terminology: Is there an algorithm to test whether a polynomial has an integral root? (If yes, give the algorithm.) Or, define a language $D = \{ p | p$ is a polynomial with integral root}. Is there a Turing machine to decide $D$? Here $p$, although a polynomial, is treated as a string.
  Note: In 1970, it was proved that $D$ is not Turing-decidable (or undecidable).

9.2 A binary encoding scheme for TMs

- TM $\leftrightarrow$ binary number.
  $Q = \{q_1, q_2, \ldots, q_{|Q|} \}$ with $q_1$ to be the start state, $q_2$ to be the accept state, and $q_3$ to be the reject state.
  $\Gamma = \{X_1, X_2, \ldots, X_{|\Gamma|}\}$.
  $D = \{D_1, D_2\}$ with $D_1$ to be $L$ and $D_2$ to be $R$.
  A transition $\delta(q_i, X_j) = (q_k, X_m, D_m)$ is coded as $0^i10^j10^k10^m$. A TM is coded as $C_111C_211\cdots11C_n$, where each $C$ is the code for a transition.
- TM $M$ with input $w$ is represented by $< M, w >$ and coded as $M111w$.
- Using similar schemes, we can encode DFA, NFA, PDA, RE, and CFG.

9.3 Decidable languages

Reading Sipser 4.1 (pp. 166-173)
The following languages are decidable by TMs.
- $A_{DFA} = \{ < B, w > | B$ is a DFA that accepts string $w \}$.
- $A_{NFA} = \{ < B, w > | B$ is an NFA that accepts string $w \}$.
- $A_{REX} = \{ < R, w > | R$ is a regular expression that generates string $w \}$.
- $E_{DFA} = \{ < B > | B$ is a DFA and $L(B) = \emptyset \}$.
- $E_{Q_{DFA}} = \{ < B_1, B_2 > | B_1$ and $B_2$ are DFAs and $L(B_1) = L(B_2) \}$.
- $A_{CFG} = \{ < G, w > | G$ is a CFG that generates string $w \}$.
- $E_{CFG} = \{ < G > | G$ is a CFG and $L(G) = \emptyset \}$.
- Every CFL is decidable.

9.4 Diagonalization

Reading: Sipser 4.2 (pp. 174-179)
- The size of an infinite set: Countably infinite and uncountably infinite.
- Diagonalization to prove a set to be uncountably infinite.
  Example (Sipser p. 175): $\mathbb{Q}$, the set of positive rational numbers, is countably infinite.
  Example (Sipser p. 177): $\mathbb{R}$, the set of real numbers, is uncountably infinite.
- Some languages are not Turing-recognizable. (Or equivalently, there are more languages than Turing machines. Since the number of Turing machines is countable, we wish to prove that the number of languages over an alphabet is uncountable.)
9.5 A language that is not Turing-recognizable

- Enumerating binary strings: \( \epsilon, 0, 1, 00, 01, 10, 11, \cdots \). The \( i \)th string, \( w_i \), is the \( i \)th string in the above lexicographic ordering.

- Let the \( i \)th TM, \( M_i \), be the TM whose code is \( w_i \), the \( i \)th binary string. If \( w_i \) is not a valid TM code, then let \( M_i \) be the TM that immediately rejects any input, i.e., \( L(M_i) = \emptyset \).

- Define the diagonalization language \( A_D = \{ w_i | w_i \not\in L(M_i) \} \). A boolean table where the \((i, j)\) entry indicates whether TM \( M_i \) accepts string \( w_j \). Language \( A_D \) is made by complementing the diagonal.

- \( A_D \) is not Turing-recognizable.

  Proof: Suppose, by contradiction, there is a TM \( M \) such that \( A_D = L(M) \). Then \( M = M_i \) with code \( w_i \) for some \( i \). \( w_i \in A_D \) iff \( w_i \not\in L(M_i) \) by definition of \( A_D \). \( w_i \in A_D \) iff \( w_i \in L(M_i) \) by \( A_D = L(M_i) \). A contradiction.

9.6 A language that is Turing-recognizable but not Turing-decidable

Reading: Sipser 4.2 (pp. 173-174 and 179-182)

- A universal TM:
  - Each TM (among those discussed) can only solve a single problem, however, a computer can run arbitrary algorithms. Can we design a general-purposed TM that can solve a wide variety of problems?
  - Theorem: There is a universal TM \( U \) which simulates an arbitrary TM \( M \) with input \( w \) and produces the same output.
  - TM \( U \) is an abstract model for computers just as TM \( M \) is a formal notion for algorithms.

- Let \( A_{TM} = \{ < M, w > | M \text{ is a TM and } M \text{ accepts string } w \} \)

  \( A_{TM} \) is Turing-recognizable since it can be recognized by TM \( U \). \( A_{TM} \) is called the universal language.

- \( A_{TM} \) is not Turing-decidable.

  Proof: Assume that \( A_{TM} \) is decided by TM \( T \). Then on input \( < M, w > \), \( T \) accepts iff \( M \) accepts \( w \).

  Define TM \( D \), which on input \( < M > \), runs \( T \) on input \( < M, < M > > \) and accepts iff \( T \) rejects \( < M, < M > > \).

  Feed \( < D > \) to \( D \). We see that \( D \) accepts \( < D > \) iff \( T \) rejects \( < D, < D > > \) iff \( D \) does not accept \( < D > \). A contradiction.

  Diagonalization is used in this proof. Why?