Overview

This week we will focus on reviewing the core concepts involved with undecidability, reducibility, Rice’s theorem, incompleteness of mathematics, and so on.

1 Concept Review

1.1 Undecidability

By a cardinality argument, we know that almost all languages are undecidable. This argument, however, does not give us an explicit construction. The following theorem does just that.

**Theorem 1.1.** The language \( \{\langle M, w \rangle : M \text{ accepts the input } w \} \) is not decidable.

**Proof.** Assume \( \{\langle M, w \rangle : M \text{ accepts the input } w \} \) is decidable, then the language \( D = \{\langle M \rangle : M \text{ does not accepts } \langle M \rangle \} \) is decidable. Suppose \( D \) is decidable by \( M_1 \), then \( \langle M_1 \rangle \in D \) iff \( M_1 \) accepts \( \langle M_1 \rangle \) iff \( \langle M_1 \rangle \in D \), which is a contradiction. (This is the standard diagonalization argument.)

1.2 Reducibility

**Definition 1.1.** A function \( f : \Sigma^*_1 \to \Sigma^*_2 \) is computable if there is a Turing machine such that for every input \( w \in \Sigma^*_1 \), \( M \) halts with just \( f(w) \) on its tape.

**Definition 1.2.** A reduction of \( L_1 \subseteq \Sigma^*_1 \) to \( L_2 \subseteq \Sigma^*_2 \) is a computable function \( f : \Sigma^*_1 \to \Sigma^*_2 \) such that, for any \( w \in \Sigma^*, w \in L_1 \) if and only if \( f(w) \in L_2 \), and we write \( L_1 \leq_m L_2 \).

Intuitively, \( L_1 \) reduces to \( L_2 \) means that \( L_1 \) is not harder than \( L_2 \). More formally, we can express this intuition in the following lemma.

**Lemma 1.1.** If \( L_1 \leq_m L_2 \) and \( L_1 \) is undecidable, then so it \( L_2 \).
1.3 Rice’s theorem

**Theorem 1.2** (Rice’s theorem). Let \( \mathcal{P} \) be any subset of the class of r.e. languages such that \( \mathcal{P} \) and its complement are both nonempty. Then the language \( L_\mathcal{P} = \{ \langle M \rangle : L(M) \in \mathcal{P} \} \) is undecidable.

Intuitively, Rice’s theorem states that Turing machines can not test whether another Turing machine satisfies a (nontrivial) property. For example, let \( \mathcal{P} \) be the subset of the recursively enumerable languages which contains the string \( a \). Then Rice’s theorem claims that there is no Turing machine which can decide whether a Turing machine accepts \( a \).

2 Exercises

**Exercise 2.1.** Reductions can be tricky to get the hang of, and you want to avoid “going the wrong way” with them. In which of these scenarios does \( L_1 \leq_m L_2 \) provide useful information (and in those cases, what may we conclude)?

(a) \( L_1 \)’s decidability is unknown and \( L_2 \) is undecidable
(b) \( L_1 \)’s decidability is unknown and \( L_2 \) is decidable
(c) \( L_1 \) is undecidable and \( L_2 \)’s decidability is unknown
(d) \( L_1 \) is decidable and \( L_2 \)’s decidability is unknown

**Exercise 2.2.** Argue that \( \leq_m \) is a transitive relation.

**Exercise 2.3.** Determine, with proof, whether the following languages are decidable.

(a) \( L = \{ \langle M, x \rangle : At \ some \ point \ it \ its \ computation \ on \ x, \ M \ \text{re-enters \ its \ start \ state} \} \)
(b) \( L = \{ \langle x, y \rangle : f(x) = y \} \) where \( f \) is a fixed computable function.
(c) \( CF_{TM} = \{ \langle M \rangle : L(M) \ is \ context-free \} \)

**Exercise 2.4.** Show \( \{ G : G \ is \ a \ CFG \ generating \ x \} \leq_M \{ G : G \ is \ a \ CFG \ generating \ xy \} \).