3P17/4P17 COMPUTABILITY 2014-2015 Lecture Handout 1

Alphabets, strings, languages

An **alphabet** Σ is a finite set of symbols. E.g., $\Sigma = \{\texttt{0}, \texttt{1}\}$, $\Sigma = \{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$.

A **string σ over an alphabet** Σ is a sequence of symbols from Σ. E.g., $\sigma = \texttt{gniksjfnd}$.

The **length** $|\sigma|$ of a string σ is the number of symbols that it is formed of.

Concatenation of strings $\sigma_1 = \texttt{abc}$, $\sigma_2 = \texttt{xyz}$ is $\sigma_1\sigma_2 = \texttt{abcxyz}$

A **substring** is a string within a string, appearing consecutively, e.g., `bbb` is a substring of `abbbaa` but not a substring of `babbaa`.

The **empty string** ε is for strings like the number zero in arithmetic: If σ is a string, $\varepsilon\sigma = \sigma\varepsilon = \sigma$ and $|\varepsilon| = 0$.

A **language** is a set of strings, can be finite or infinite.

Given an alphabet Σ, the **star operation** gives Σ^*, the set of all strings over Σ. E.g., if $\Sigma = \{\texttt{a}, \texttt{b}\}$, then $\Sigma^* = \{\varepsilon, \texttt{a}, \texttt{b}, \texttt{aa}, \texttt{bb}, \texttt{ab}, \texttt{ba}, \texttt{aaa}, \ldots\}$.

Turing machines

A **Turing machine** M is a 7-tuple: $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ where Q, Σ, Γ are finite sets and

1. Q is the set of **states** (always includes $q_0, q_{\text{acc}}, q_{\text{rej}}$),
2. Σ is the **input alphabet**; it does not include the **blank symbol** $\sqcup$, or **end-of-tape** symbol $\triangleright$
3. Γ is the **tape alphabet**, where $\sqcup, \triangleright \in \Gamma$ and $\Sigma \subset \Gamma$,
4. $\delta : Q \setminus \{q_{\text{acc}}, q_{\text{rej}}\} \times \Gamma \to Q \times \Gamma \times \{\text{L}, \text{R}\}$ is the **transition function**,
5. $q_0 \in Q$ is the **start state**,
6. $q_{\text{acc}} \in Q$ is the **accept state**, and
7. $q_{\text{rej}} \in Q$ is the **reject state**.

A Turing machine has a read/write head and a tape, which is an infinite array of squares/locations, with a first location, a second location, ...

Each square/location holds exactly one symbol (which may be the blank symbol $\sqcup$).

The left end of the tape is occupied by $\triangleright$. It is never replaced by another symbol.

The Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ performs computations as follows:

-Initially there is an input string $w = w_1w_2, \ldots w_n \in \Sigma^*$ on the n leftmost squares after $\triangleright$.

-The rest of the tape is blank (i.e., filled with $\sqcup$).

- The read/write head starts at the first square after the $\triangleright$, i.e., on the first symbol of the input.

-At each step, M does the following: If the current state is q_{acc} or q_{rej}, then halt. Otherwise, read the symbol under the head and do the following as per the transition function: replace the symbol on the square underneath the head, move the head, change state.

-If the read/write head is ever over $\triangleright$, it moves right in the next step. It cannot replace $\triangleright$ (but can change state).

At any given step, the triple (current state, current head location, current tape contents) is called a **configuration**.

Configuration C_1 **yields** configuration C_2 if C_2 follows C_1 from the the rules of operation. We have **starting configurations**, **accepting configurations**, **rejecting configurations**...the latter two are **halting configurations**.

A Turing machine can:

- recognise/decide (strings in) languages
- compute functions on strings

Recognising/deciding languages

The collection of strings that cause a Turing machine M to reach an accepting state (accepting configuration) is called **the language of M**, or **the language recognised by M**, and is denoted $L(M)$.

A language L_1 is called **Turing-recognisable** if some Turing machine recognises it, i.e., if there exists a Turing machine M with $L(M) = L_1$.

Note, if $w \in L_1$, then M will always halt with accept. If $w \notin L_1$, then M may halt with reject, or may **loop** i.e., go on forever.

A Turing machine that halts (with accept or reject) on **every** input is called a **decider**.

A language L_1 is called **Turing-decidable**, or simply **decidable** if there is a decider M such that $L(M) = L_1$.

Computing functions on strings

A function $f : \Sigma^* \to \Sigma^*$ is called **a function on Σ-strings**

We say that a Turing machine M **computes the function** f if for every $w \in \Sigma^*$,

- M halts on input w
- after halting, the tape has $f(w)$ on the leftmost squares of the tape and the rest are blank

A function f is called **Turing-computable**, or simply **computable** if there exists some Turing machine M that computes it.

This captures a broad class of computations, since the strings may be encoded to represent numbers, graphs, matrices, ...