6. COMPUTABILITY THEORY

The Church–Turing thesis: Any arbitrary (sufficiently powerful) computation machine $\equiv$ the Turing machine.

Computability theory: We examine what can and especially what *cannot* be computed by a Turing machine.

An important dichotomy: Halting and non-halting Turing machines.

Definition 6.1 A Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$$

is *total*, if it halts on all inputs. A formal language A is *recursively enumerable*, if it can be recognized by some Turing machine, and *recursive*, if it can be recognized by a total Turing machine.

Alternative terminology: Recall the connection between decision problems (I/O-mappings with binary output) and formal languages: the formal language A_Π that corresponds to the decision problem Π consists of those inputs x for which the answer to Π is "yes" (that is, the desired output = 1).

A decision problem Π is *decidable*, if the corresponding formal language A_Π is recursive, and *partially decidable*, if A_Π is recursively enumerable. A problem that is not decidable, is *undecidable*. (*Note*: An undecidable problem can be partially decidable.)

In other words a decision problem is decidable, if there is a (total) algorithm that halts on all inputs for deciding it, and partially decidable, if there is an algorithm that always answers correctly in the "yes" case, but might never stop in the "no" case.

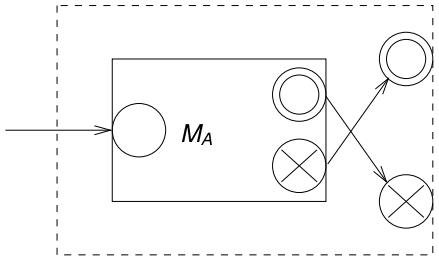
6.2 Basic properties of recursive and recursively enumerable languages

Theorem 6.1 Let $A, B \subseteq \Sigma^*$ be recursive. Then also $\bar{A} = \Sigma^* - A$, $A \cup B$ ja $A \cap B$ are recursive.

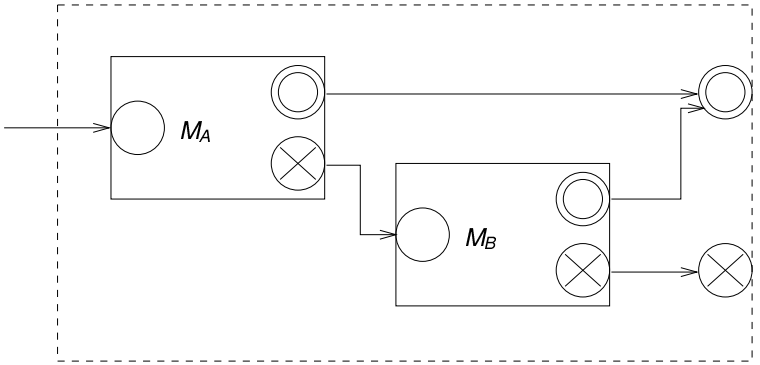
Proof.

(i) Let M_A be a total Turing machine, for which $L(M_A) = A$. A total Turing machine that recognizes $\bar{A}$

can be obtained by switching the accepting and rejecting state of M_A.



(ii) Let M_A and M_B be the total Turing machines for which $L(M_A) = A$, $L(M_B) = B$. A total Turing machine M that recognizes $A \cup B$ is obtained by combining M_A ja M_B in series: if M_A accepts the input, M accepts also; if M_A rejects, M then simulates M_B.



(iii) $A \cap B = \overline{\bar{A} \cup \bar{B}}$. $\square$

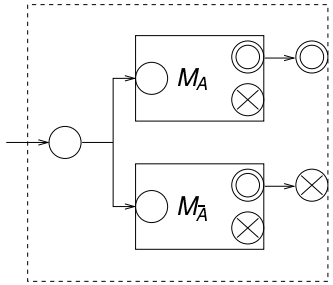
Theorem 6.2 Let $A, B \subseteq \Sigma^*$ be recursively enumerable. Then also $A \cup B$ ja $A \cap B$ are recursively enumerable.

Proof. $A \cap B$ as for Theorem 6.1 and $A \cup B$ as for Theorem 6.3. □

Theorem 6.3 A language $A \subseteq \Sigma^*$ is recursive if and only if the languages A and $\bar{A}$ are recursively enumerable.

Proof. From left to right, see Theorem 6.1(i). We prove the theorem from right to left.

Let M_A and $M_{\bar{A}}$ be the Turing machines that recognize A and $\bar{A}$. For all $x \in \Sigma^*$ *either* M_A *or* $M_{\bar{A}}$ *halts* and accepts x. We combine M_A and $M_{\bar{A}}$ "in parallel" to obtain a total two-tape recognizer M: M simulates M_A on tape 1 and $M_{\bar{A}}$ on tape 2.



If simulation 1 accepts, M accepts; if simulation 2 accepts, M rejects the input. □

Corollary 6.4 Let $A \subseteq \Sigma^*$ be a recursively enumerable language that is not recursive. Then $\bar{A}$ is not recursively enumerable. □

6.3 Encoding Turing machines

Let us examine standard model Turing machines with input alphabet $\Sigma = \{0, 1\}$. Every such machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$$

may be represented as a binary string:

Assume that $Q = \{q_0, q_1, \ldots, q_n\}$, where $q_{\text{acc}} = q_{n-1}$ and $q_{\text{rej}} = q_n$; and that $\Gamma \cup \{>, <\} = \{a_0, a_1, \ldots, a_m\}$, where $a_0 = 0$, $a_1 = 1$, $a_2 = >$ and $a_3 = <$. We also write $\Delta_0 = L$ ja $\Delta_1 = R$.

Encoding the values of the transition function δ: the rule $\delta(q_i, a_j) = (q_r, a_s, \Delta_t)$ is encoded as

$$c_{ij} = 0^{i+1}10^{j+1}10^{r+1}10^{s+1}10^{t+1}.$$

The whole machine M is encoded as

$$\begin{aligned} c_M \quad = \quad & 111c_{00}11c_{01}11\ldots11c_{0m}11c_{10}11\ldots11c_{1m}11 \\ & \ldots11c_{n-2,0}11\ldots11c_{n-2,m}111. \end{aligned}$$

Conversely each binary string c can be seen as an encoding of a Turing machine M_c. Binary strings that are not legal encodings of the form described above are thought to encode a trivial Turing machine M_{triv} that rejects all inputs.

We define:

$$M_c = \begin{cases} \text{the TM } M, \text{ for which } c_M = c, \text{ if } c \text{ is a legal encoding of a TM} \\ \text{the TM } M_{\text{triv}}, \text{ otherwise.} \end{cases}$$

We obtain a listing of all Turing machines over the alphabet $\{0, 1\}$, and indirectly also of all recursive languages over $\{0, 1\}$. The machines are

$$M_\varepsilon, M_0, M_1, M_{00}, M_{01}, \ldots,$$

and the languages

$$L(M_\varepsilon), L(M_0), L(M_1), L(M_{00}), L(M_{01}), \ldots$$

(indices in canonical order). Each language may appear more than once in the list.

A language that is not recursively enumerable

Lemma 6.5 The language

$$D = \{c \in \{0,1\}^* \mid c \notin L(M_c)\}$$

is not recursively enumerable.

Proof. Suppose that $D = L(M)$ for some standard model Turing machine M. Let d be the binary encoding of M, i.e., $D = L(M_d)$. Now

$$d \in D \quad \Leftrightarrow \quad d \notin L(M_d) = D.$$

By this contradiction, D cannot be recursively enumerable. □

The decision problem corresponding to D: "Is it true that the machine whose encoding is c does not accept c?" More natural examples will follow.

A pictorial description of D: if the characteristic functions of $L(M_\varepsilon)$, $L(M_0)$, $L(M_1)$, ... are presented as a table, then D differs from each language on the diagonal:

$D \searrow$	$L(M_\varepsilon)$	$L(M_0)$	$L(M_1)$	$L(M_{00})$	$\cdots$
ε	$\not{0}^1$	0	0	0	$\cdots$
0	0	$\not{1}^0$	1	0	$\cdots$
1	0	0	$\not{1}^0$	1	$\cdots$
00	0	0	0	$\not{0}^1$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

6.4 Universal Turing machines

The *universal language* U over an alphabet $\{0,1\}$ is defined by:

$$U = \{c_M w \mid w \in L(M)\}.$$

Let A be a recursively enumerable language over the alphabet $\{0,1\}$ and let M be a standard type Turing machine that recognizes A. Now

$$A = \{w \in \{0,1\}^* \mid c_M w \in U\}.$$

Also the language U is recursively enumerable. Turing machines that recognize U are called *universal Turing machines*.

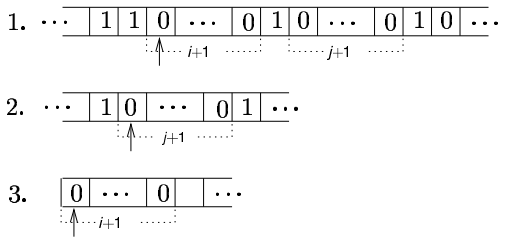
Theorem 6.6 The language U is recursively enumerable.

Proof. It is easiest to describe the universal machine M_U that recognizes U as a three-tape machine. At the beginning, the input is placed at the beginning of tape 1.

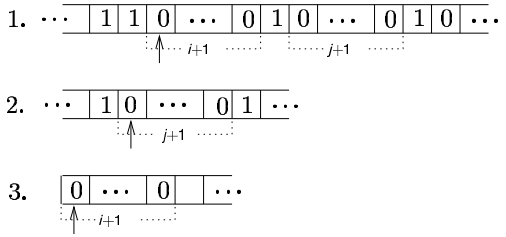
After this the machine proceeds as follows:

1. First M_U checks that the input is of the form cw, where c is a legal encoding of a Turing machine. If the encoding is not legal, M_U rejects the input; otherwise it copies the string $w = a_1 a_2 \ldots a_k \in \{0,1\}^*$ onto tape 2 in the form

$$00010^{a_1+1}10^{a_2+1}1 \ldots 10^{a_k+1}10000.$$



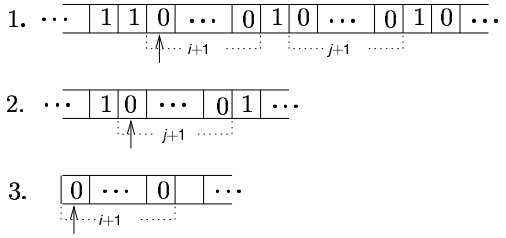
2. If the input is of the form cw, where $c = c_M$ for some machine M, M_U must find out, whether M would accept w. For this purpose M_U maintains the description c of the machine M on tape 1, simulates the tape of M on tape 2, and keeps the simulated state of M on tape 3 in the form $q_i \sim 0^{i+1}$ (at the start M_U writes the code 0 of the state q_0 onto tape 3).



3. After initialization M_U repeatedly simulates single steps of the machine M. For each step, M_U first searches the description of M on tape one for the location that corresponds with the simulated state q_i of M and the alphabet a_j.

Let tape 1 contain $0^{i+1}10^{j+1}10^{r+1}10^{s+1}10^{t+1}$.

Then M_U replaces the string 0^{i+1} on tape 3 by the string 0^{r+1}, the string 0^{j+1} on tape 2 by the string 0^{s+1}, and moves the tape head on tape 2 a step to the left if $t = 0$ or to the right if $t = 1$.

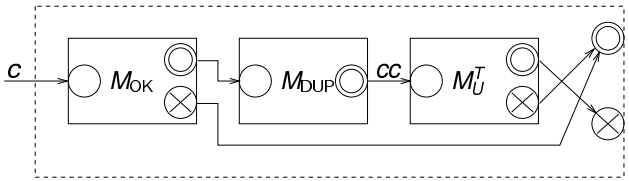


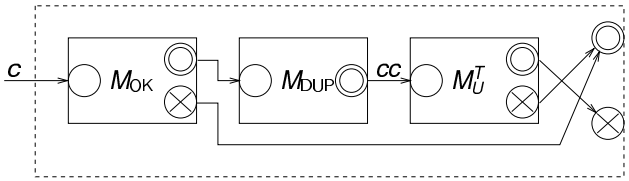
If tape 1 contains no code related to the simulated state q_i the simulated machine M has entered an accepting or rejecting final state. Then $i = k + 1$ or $i = k + 2$, where q_k is the last state described on tape 1. The machine M_U enterst the corresponding final state q_{acc} or q_{rej}. □

Theorem 6.7 The language U is not recursive.

Todistus. Assume that U had a total recognizer M_U^T. Then we could form a total recognizel for the language D in Lemma 6.5 as follows.

Let M_{OK} be a total Turing machine that tests whether the input is a legal encoding of a Turing machine, and let M_{DUP} be a total Turing machine, that changes the input string c to cc. The machine M_D can be combined from the machines M_U^T, M_{OK} and M_{DUP} as follows:





Obviously M_D is total if M_U^T is, and

$$\begin{aligned} c \in L(M_D) &\Leftrightarrow c \notin L(M_{OK}) \text{ or } cc \notin L(M_U^T) \\ &\Leftrightarrow c \notin L(M_c) \\ &\Leftrightarrow c \in D. \end{aligned}$$

But according to Lemma 6.5 the language D is not recursive; a contradiction. □.

Corollary 6.8 The language

$$\tilde{U} = \{c_M w \mid w \notin L(M)\}$$

is not recursively enumerable.

Proof. The language $\tilde{U}$ is essentially the same as the complement $\bar{U}$ of U; to be exact, $\bar{U} = \tilde{U} \cup \text{ERR}$, where ERR is an easily recognizable recursive language

$$\begin{aligned} \text{ERR} &= \{x \in \{0,1\}^* \mid x \text{ contains no legal} \\ &\quad \text{encoding of a Turing machine as a prefix}\}. \end{aligned}$$

If $\tilde{U}$ would be recursively enumerable, then so would be $\bar{U}$. Since U is recursively enumerable, it would follow that U would even be recursive. However, this contradicts the previous theorem, and thus $\tilde{U}$ cannot be recursively enumerable. □