Luku 1

Solvability and unsolvability

1.1 Solvable problems i.e. recursive language

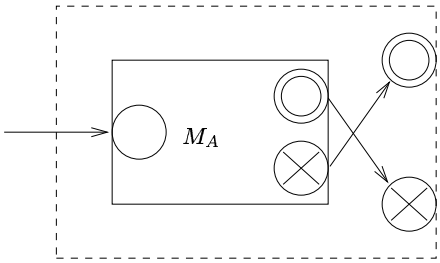
Theorem: Let $A, B \subseteq \Sigma^*$ are recursive. Then also

i) $\bar{A} = \Sigma^* - A$,
ii) $A \cup B$ ja
iii) $A \cap B$

are recursive.

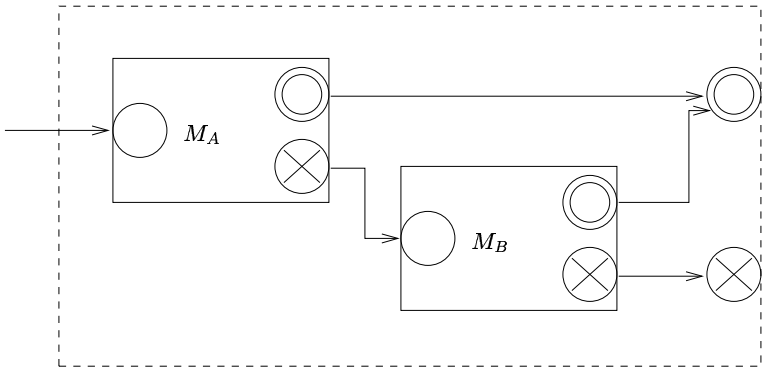
Proof:

(i)Let M_A be a total Turing machine, which recognizes A (i.e. $L(M_A) = A$). A total Turing machine for $\bar{A}$ can be get by changing the accepting and rejecting states of M_A.



Kuva 1.1: Rekursiivisen kielen komplementin tunnistaminen Turingin koneella.

(ii)Let M_A and M_B be total TM's, which recognize languages A and B (i.e. $L(M_A) = A$, $L(M_B) = B$). We get a total machine for $A \cup B$, M by combining M_A and M_B in the following way:



Kuva 1.2: Kahden rekursiivisen kielen yhdisteen tunnistaminen Turingin koneella.

(iii) For $A \cap B$ we get a total TM by de Morganin $A \cap B = \overline{\bar{A} \cup \bar{B}}$. I.e. make first complement machines $M_{\bar{A}}$ ja $M_{\bar{B}}$, combine them into $M_{\bar{A} \cup \bar{B}}$ and finally make a complement machine of that $M_{\overline{\bar{A} \cup \bar{B}}}$. □

(Taks: Draw $M_{\overline{\bar{A} \cup \bar{B}}}$!)

1.2 Partially solvable problems i.e. recursive enumerable language

Theorem: Let $A, B \subseteq \Sigma^*$ recursive enumerable. Then also $A \cup B$ and $A \cap B$ are recursive enumerable.

Proof: Exercise. □

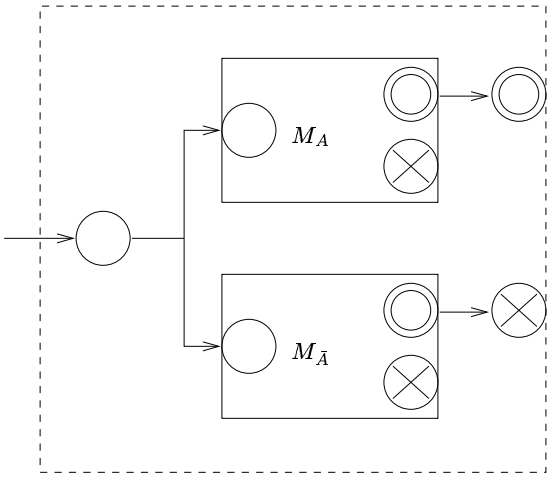
N.B! For recursive enumerable languages it doesn't hold that the complement language would be (necessarily) recursive enumerable! Why we cannot just change the accepting and rejecting final states?

The next important result gives as a method to recognize, when the problem is solvable:

Theorem: Language $A \subseteq \Sigma^*$ is recursive, if and only if languages A and $\bar{A}$ are recursive enumerable.

Todistus. "$\Rightarrow$": follows from previous theorem 1 (i).

"$\Leftarrow$": Let M_A and $M_{\bar{A}}$ be Turing machines for recognizing A and $\bar{A}$. For all $x \in \Sigma^*$ *either* M_A *or* $M_{\bar{A}}$ *halts* and accepts x. Let's combine M_A and $M_{\bar{A}}$ in the following way to get a total machine M:



Kuva 1.3: Totaalisen Turingin koneen muodostaminen kahdesta rinnakkain toimivasta koneesta.

$\Rightarrow$ *Corollary:* If $A \subseteq \Sigma^*$ is a recursive enumerable language, which is not recursive, then $\bar{A}$ is not recursive enumerable (i.e. the problem is fully unsolvable)!

1.3 Unsolvability

Idea:

1) Turing machines are finite
2) Thus we can enumerate all TM's 3) The set of all problems is uncountable (Diagonalization argument)

$\Rightarrow$ There exists problems for which we cannot give a solving TM (and no other computational solution by the Church-Turing thesis).

1.4 Universal machine and universal language

- Universal machine
  - gets as its input a code of M and its input w
  - halts only if M halts on w
  - outputs the same result as M by w
- We need a way to code TM's such that

  - any TM can be represented as a code in finite alphabet
  - the universal machine can decode it

- it is enough to code the transition function of TM – it describes the behaviour of the machine (and it doesn't matter if we rename the states)!

1.4.1 Coding Turing machines as integers (binary strings)

Let's consider the standard Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{yes}}, q_{\text{no}})$, whose input alphabet is $\Sigma = \{0, 1\}$.

Let $Q = \{q_0, q_1, \ldots, q_n\}$, in which $q_{\text{yes}} = q_{n-1}$ and $q_{\text{no}} = q_n$.

$\Gamma \cup \{>, <\} = \{a_0, a_1, \ldots, a_m\}$, in which $a_0 = 0$, $a_1 = 1$, $a_2 = >$ and $a_3 = <$.

Denote $\Delta_0 = L$ ja $\Delta_1 = R$.

Let's now define codes for these:

States	q_0	0
	q_1	00
	q_2	000
	.	.
	.	.
	.	.
	$q_{n-1} = q_{yes}$	0^n
	$q_n = q_{no}$	0^{n+1}
Characters $\Gamma \cup \{>, <\}$	0	0
	1	00
	$>$	000
	$<$	0000
	a_2	$00000 = 0^5$
	a_3	0^6
	.	.
	.	.
	.	.
	a_m	0^{m+1}
Directions	L	0
	R	00

Each value of the transition function δ can now be coded in the following way:

Rule $\delta(q_i, a_j) = (q_r, a_s, \Delta_t)$ has code

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

i.e. $c_{ij} = 1(q_i\text{'s code})1(a_j\text{'s code})1(q_r\text{'s code})1(a_s\text{'s code})1(\Delta_t\text{'s code})$

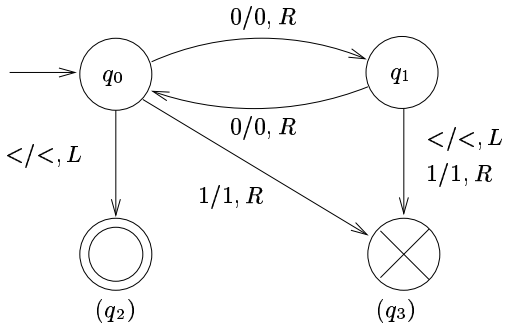
The whole code for machine M can be made by combining the codes of transition functions into one string, using 11 to separate transition function values and 111 to mark the beginning and end of the code.

Thus code of M is:

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

E.g. Machine that recognizes language $\{0^{2k} \mid k \geq 0\}$ has code:

$$c_M = 111\underbrace{01010010100}_{\delta(q_0,0)=(q_1,0,R)}11\underbrace{01001000010010}_{\delta(q_0,1)=(q_3,1,R)}11\ldots$$



Kuva 1.4: Kielen $\{0^{2k} \mid k \geq 0\}$ tunnistava Turingin kone.

- Each TM M has code c_M
- Each binarystring c can be associated a TM M_c.
- N.B.! Binarystrings, which don't represent any legal TM, can be associated some trivial, all inputs rejecting (i.e. recognizing $\emptyset$ -language) machine M_{triv}.

Let's define:

$$M_c = \begin{cases} \text{machine } M, \text{ for which } c_M = c, \text{ if } c \text{ is legal code;} \\ \text{machine } M_{\text{triv}}, \text{ otherwise.} \end{cases}$$

1.4.2 Unsolvability of the universal language

- *Universal language* U in alphabet $\{0,1\}$ is defined as

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

- Language U is recursive enumerable. TM's, which recognize U, are called *universal Turing machines.*
- But U is not recursive (and the complement of it $\overline{U}$ is not recursive enumerablee)
  $\Rightarrow$ we cannot recognize unsolvable problems computationally!

These are just the basic results, try also to study the proofs and how universal machine works!