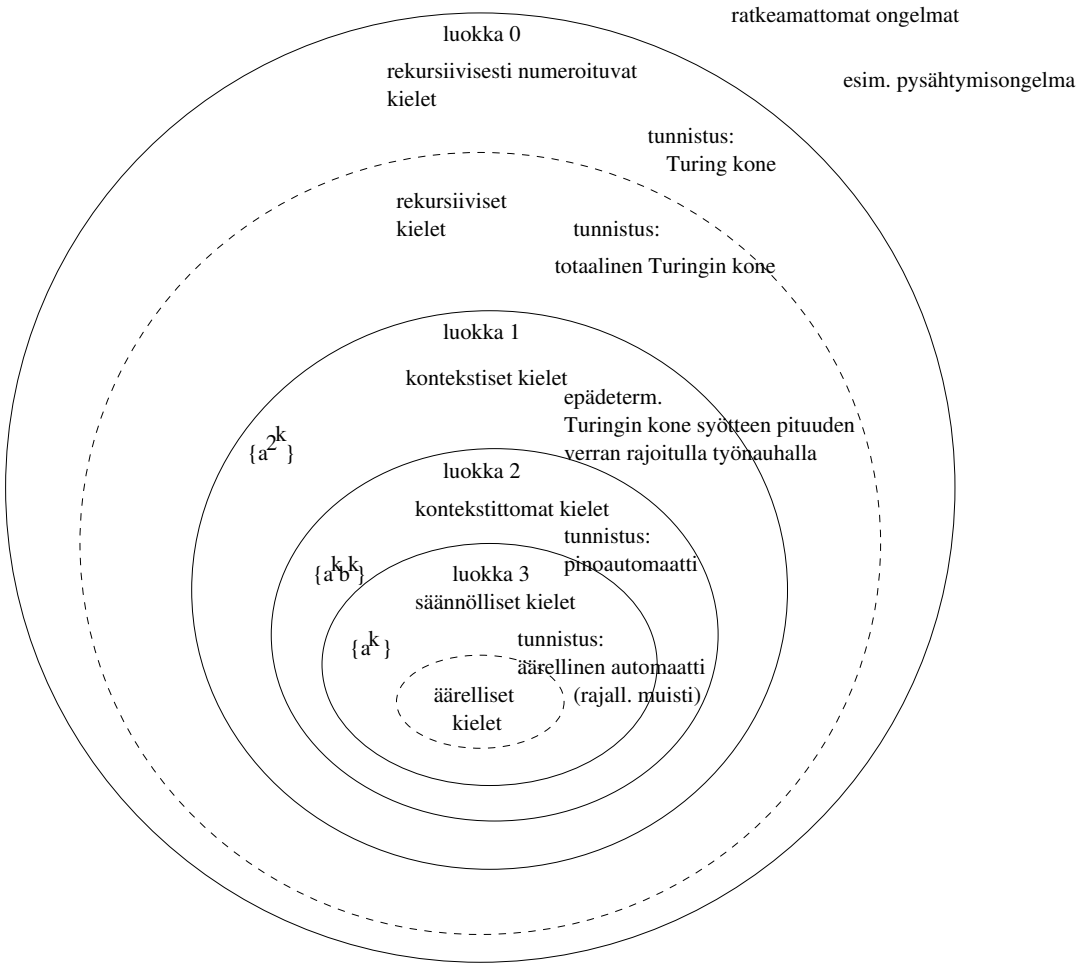
Chomsky language hierarchy

From 0 to 3:

0: Unristrected languages: recursive enumerable and recursive languages. (recognition: Turing machines)
1: Context-sensitive languages.(Turing machines, with bounded working space)
2: Context-free languages. (Pushdown automata)
3: Regular languages (the subset of which are finite languages) (Finite automata).



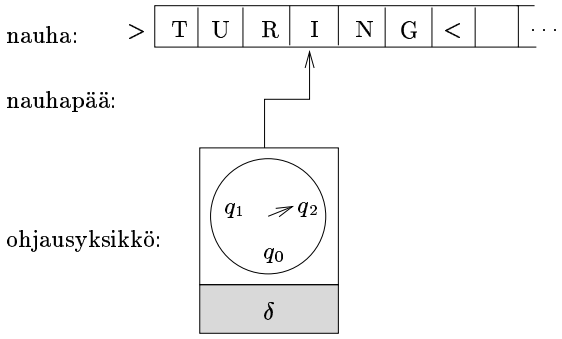
Kuva 1: Chomsky language hierarchy completed with classes of Recursive and Recursive enumerable languages

Luku 1

Turing machines

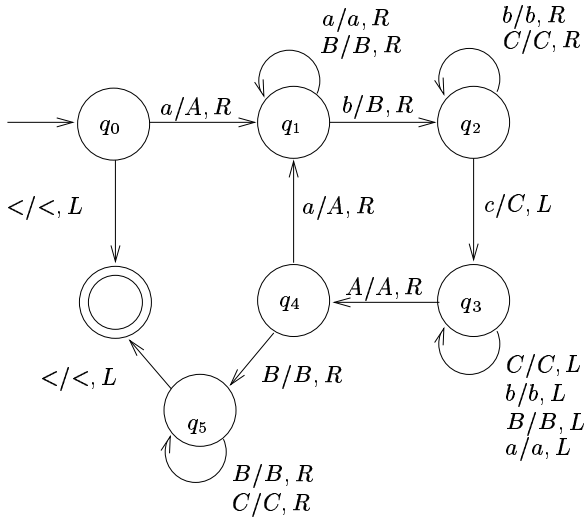
- In the beginning we have the input string on the tape (We use special characters $>$ and $<$ to mark the beginning and end of the tape. The beginning character cannot be moved, but the end character moves further, when we write into end.), the reading head points to the first character and the machine is started in state q_0.
- In each step the machine reads the character in the current position of the reading head, decides by its transition function the next state, writes a new character into reader head position and transfers one step left or right (although we cannot move to the left of the beginning mark $>$).
- The machine has an accepting final state q_{yes} and rejecting final state q_{no}. The machine halts, when it enters final state.

E.g. a machine which recognizes language $\{a^k b^k c^k | k \geq 0\}$



Kuva 1.1: Turing machine.

- Idea: we count a's, b's and c's by changing them one by one to A, B and C
- when the last a is changed to A, we have to check that there are no more b's or c's.



Kuva 1.2: Turning machine for recognizing $\{a^k b^k c^k \mid k \geq 0\}$.

Definition: *Turing machine* is a seven-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{yes}}, q_{\text{no}}),$$

missÃď

- Q is finite set of *states*
- Σ is *input alphabet*;
- $\Gamma \supseteq \Sigma$ is *tape alphabet*; suppose that $>, < \notin \Gamma$;
- $\delta : (Q - \{q_{\text{yes}}, q_{\text{no}}\}) \times (\Gamma \cup \{>, <\}) \to Q \times (\Gamma \cup \{>, <\}) \times \{L, R\}$ is *transition function*;
- $q_0 \in Q$ is *initial state*;
- $q_{\text{yes}} \in Q$ is *accepting* and $q_{\text{no}} \in Q$ *rejecting final state.*

The values of the transition function

$$\delta(q, a) = (q', b, \Delta)$$

are required:

(i) if $b = >$, then $a = >$;

(ii) if $a = >$, then $b = >$ and $\Delta = R$;

(iii) if $b = <$, then $a = <$ and $\Delta = L$.

- Interpretation of $\delta(q, a) = (q', b, \Delta)$:
  - When in state q and reading tape symbol (or special beginning or end mark) a, the machine transfers to state q', writes character b, and moves the reading head one position into direction Δ ($L \sim$ "left", $R \sim$ "right").
  - The written characters and transition directions are restricted, if we read $a =$ '$>$' or '$<$', and the value of transition function is not defined, if $q = q_{\text{yes}}$ or $q = q_{\text{no}}$. When the machine enters either of these states, if halts immediately.
- The *configuration* of the machine is

  $$(q, u, a, v) \in Q \times \Gamma^* \times (\Gamma \cup \{\epsilon\}) \times \Gamma^*,$$

  in which can be $a = \epsilon$, iáÿ§ also $u = \epsilon$ or $v = \epsilon$.

- Interpretation: in state q, the tape contents left to reading head is u, in the current position of the reading head there is character a and tape contents to the right of the head is v.

- Possibly $a = \epsilon$, if the head is in the beginning or end of the used part of tape. In the first case the machine "notices" special character '$>$' and in the second case special character '$<$'.

- *Initial configuration with input* $x = a_1 a_2 \ldots a_n$ is

$$(q_0, \epsilon, a_1, a_2 \ldots a_n).$$

- Configuration (q, u, a, v) is usually denoted simply $(q, u\underline{a}v)$, and the initial configuration with input x by $(q_0, \underline{x})$.

- Configuration (q, w) *leads directly* to configuration (q', w'), mark

$$(q, w) \underset{M}{\vdash} (q', w'),$$

if any of the following conditions hold: for all $q, q' \in Q$, $u, v \in \Gamma^*$, $a, b \in \Gamma$ and $c \in \Gamma \cup \{\epsilon\}$:

if $\delta(q, a) = (q', b, R)$, then $(q, u\underline{a}cv) \underset{M}{\vdash} (q', ub\underline{c}v)$;

if $\delta(q, a) = (q', b, L)$, then $(q, uc\underline{a}v) \underset{M}{\vdash} (q', u\underline{c}bv)$;

if $\delta(q, >) = (q', >, R)$, then $(q, \underline{\epsilon}cv) \underset{M}{\vdash} (q', \underline{c}v)$;

if $\delta(q, <) = (q', b, R)$, then $(q, u\underline{\epsilon}) \underset{M}{\vdash} (q', ub\underline{\epsilon})$;

if $\delta(q, <) = (q', b, L)$, then $(q, uc\underline{\epsilon}) \underset{M}{\vdash} (q', u\underline{c}b)$;

if $\delta(q, <) = (q', <, L)$, then $(q, uc\underline{\epsilon}) \underset{M}{\vdash} (q', u\underline{c})$.

- Configurations of form (q_{yes}, w) or (q_{no}, w) don't lead to any other configuration. In those cases the machine halts.

- Conf. (q, w) *leads to conf.* (q', w'), mark

$$(q, w) \underset{M}{\vdash^*} (q', w'),$$

if there exists a queue of conf's (q_0, w_0), (q_1, w_1), $\ldots$, (q_n, w_n), $n \geq 0$, such that

$$(q, w) = (q_0, w_0) \underset{M}{\vdash} (q_1, w_1) \underset{M}{\vdash} \cdots \underset{M}{\vdash} (q_n, w_n) = (q', w').$$

- Turing machine M *accepts* string $x \in \Sigma^*$, if

$$(q_0, \underline{x}) \vdash_M^* (q_{\text{yes}}, w) \qquad \text{for some } w \in \Gamma^*;$$

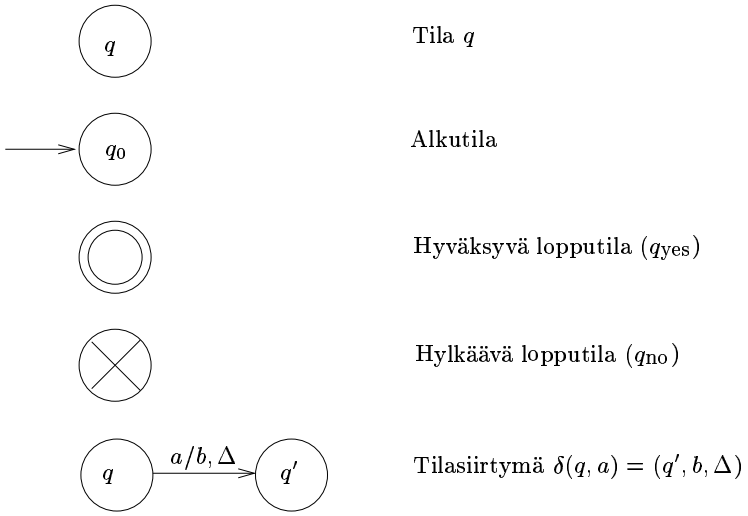
  otherwise M *rejects* x.

  (i.e. to accept it is enough that we enter the accepting final state – we don't even have to read the whole string!)

- *The language recognized by* M is:

$$L(M) = \{x \in \Sigma^* \mid (q_0, \underline{x}) \vdash_M^* (q_{\text{yes}}, w) \text{ jollakin } w \in \Gamma^*\}.$$

1.1 Representation as transition diagrams



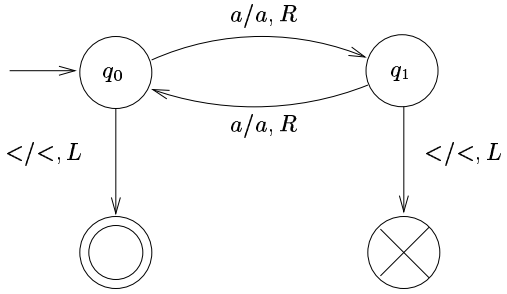
Kuva 1.3: Symbols for transition diagrams.

E.g.: language $\{a^{2k} \mid k \geq 0\}$ can be recognized by Turing machine

$$M = (\{q_0, q_1, q_{\text{yes}}, q_{\text{no}}\}, \{a\}, \{a\}, \delta, q_0, q_{\text{yes}}, q_{\text{no}}),$$

in which

$$\begin{aligned} \delta(q_0, a) &= (q_1, a, R), \\ \delta(q_1, a) &= (q_0, a, R), \\ \delta(q_0, <) &= (q_{\text{yes}}, <, L), \\ \delta(q_1, <) &= (q_{\text{no}}, <, L). \end{aligned}$$



Kuva 1.4: Turing machine which recognizes $\{a^{2k} \mid k \geq 0\}$.

Computation of M on input aaa:

$$\begin{aligned}(q_0, \underline{a}aa) &\underset{M}{\vdash} (q_1, a\underline{a}a) \underset{M}{\vdash} (q_0, aa\underline{a}) \\ &\underset{M}{\vdash} (q_1, aaa\underline{\epsilon}) \underset{M}{\vdash} (q_{\text{no}}, aa\underline{a}).\end{aligned}$$

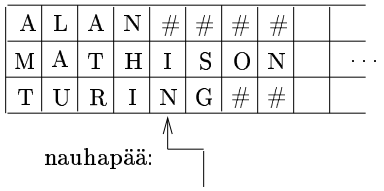
Machine halts in q_{no}, thus $aaa \notin L(M)$.

E.g. 2: Language $\{a^k b^k c^k \mid k \geq 0\}$, input $aabbcc$:

$$\begin{array}{lll}
(q_0, \underline{a}abbcc) & \vdash & (q_2, AABB\underline{C}c) \quad \vdash \\
(q_1, A\underline{a}bbcc) & \vdash & (q_2, AABBC\underline{c}) \quad \vdash \\
(q_1, Aa\underline{b}bcc) & \vdash & (q_3, AABB\underline{C}C) \quad \vdash \\
(q_2, AaB\underline{b}cc) & \vdash & (q_3, AAB\underline{B}CC) \quad \vdash \\
(q_2, AaBb\underline{c}c) & \vdash & (q_3, AA\underline{B}BCC) \quad \vdash \\
(q_3, AaB\underline{b}Cc) & \vdash & (q_3, A\underline{A}BBCC) \quad \vdash \\
(q_3, Aa\underline{B}bCc) & \vdash & (q_4, AA\underline{B}BCC) \quad \vdash \\
(q_3, A\underline{a}BbCc) & \vdash & (q_5, AAB\underline{B}CC) \quad \vdash \\
(q_3, \underline{A}aBbCc) & \vdash & (q_5, AABB\underline{C}C) \quad \vdash \\
(q_4, A\underline{a}BbCc) & \vdash & (q_5, AABBC\underline{C}) \quad \vdash \\
(q_1, AA\underline{B}bCc) & \vdash & (q_5, AABBCC\underline{\epsilon}) \quad \vdash \\
(q_1, AAB\underline{b}Cc) & \vdash & (q_{\text{yes}}, AABBC\underline{C}).
\end{array}$$

1.2 Variations of Turing machines

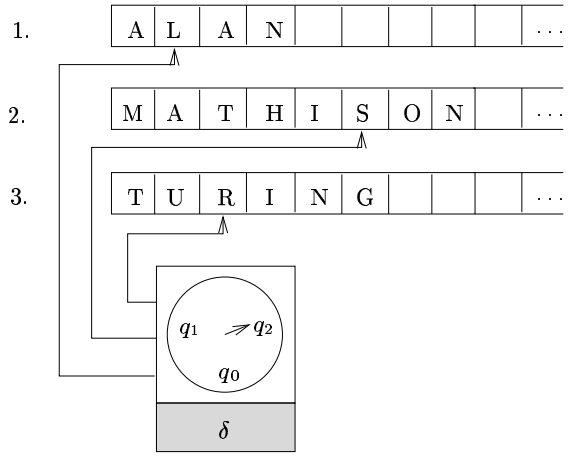
1.2.1 Multiple track TM's



Kuva 1.5: Kolmeuraisen Turingin koneen nauha.

Basic idea ....

1.2.2 Multiple tape TM's



Kuva 1.6: Kolmenauhainen Turingin kone.

Basic idea .....

1.2.3 Nondeterministic TM's

Important!
Definition and idea
Equivalence of deterministic and nondeterministic TM's

.....

Example: Recognizing composite numbers by a nondeterministic TM.

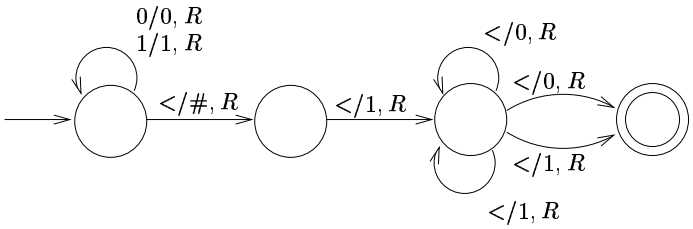
A non-negative integer n is *composite*, if it has integer factors $p, q \geq 2$, for which $pq = n$. A number, which is not composite, is *prime*.

Let's assume a deterministic TM CHECK_MULT, which recognizes language

$$L(\text{CHECK\_MULT}) = \{n\#p\#q \mid n, p, q \text{ binary numbers}, n = pq\}.$$

In addition suppose a det. TM GO_START, which moves the reading head into the beginning of the tape.

Let's also have a nondeterministic TM GEN_INT, which produces an arbitrary binary number greater than one into the end of the tape.

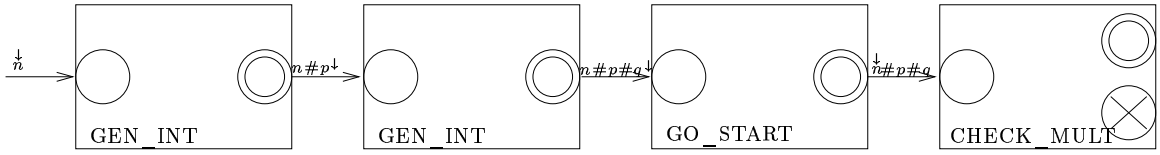


Kuva 1.7: Nondeterministic Turing machine GEN_INT.

A nondet. TM TEST_COMPOSITE, which recognizes language

$$L(\text{TEST\_COMPOSITE}) = \{n \mid n \text{ is a composite number in binary form}\}$$

can now be built by combining these elements.



Kuva 1.8: Nondeterministic Turing machine TEST_COMPOSITE.

The combined TM accepts the input binary number n, if and only if there exists binary numbers $p, q \geq 2$, such that $n = pq$ — so if and only if n is a composite number.