3.8 On limitations of context-free languages

As for regular languages, a pumping lemma exists also for context-free languages. Now, however, the string must be pumped simultaneously at two locations.

Lemma 3.9 ("The uvwxy lemma") Let L be a context-free language. Then there is an $n \geq 1$ such that any $z \in L$, $|z| \geq n$, can be split into five parts $z = uvwxy$ such that

(i) $|vx| \geq 1$,

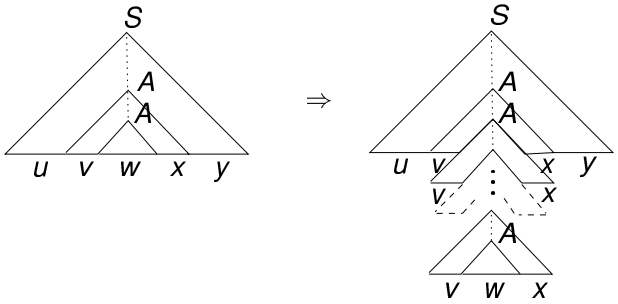
(ii) $|vwx| \leq n$,

(iii) $uv^i wx^i y \in L$ for all $i = 0, 1, 2, \ldots$.

Proof. Let $G = (V, \Sigma, P, S)$ be a grammar for L in the Chomsky normal form. Then every parse tree of G with height at most h has at most 2^h leaves. In other words, every parse tree of any $z \in L$ contains a path of length at least $\log_2 |z|$.

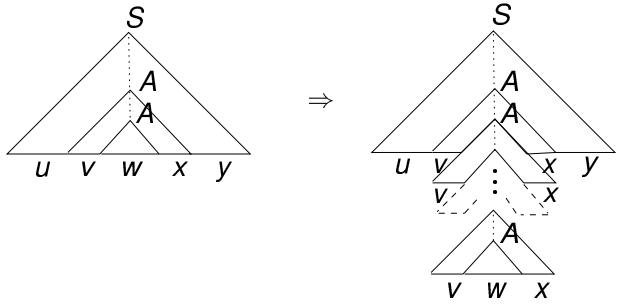
Let $k = |V - \Sigma|$ be the number of non-terminals in G. Let $n = 2^{k+1}$. Choose any $z \in L$, $|z| \geq n$, and examine its parse tree.

Based on the above, the tree contains a path of length $\geq k + 1$; on this path some non-terminal must be repeated — in fact already among the $k + 2$ lowest vertices. Let A such a non-terminal.



The string z can now be partitioned as $z = uvwxy$, where w is derived from the lowest occurrence of A and vwx is derived from the second lowest occurrence of A; the partial strings are obtained from the derivation

$$S \Rightarrow^* uAy \Rightarrow^* uvAxy \Rightarrow^* uvwxy.$$



Since $S \Rightarrow^* uAy$, $A \Rightarrow^* vAx$ and $A \Rightarrow^* w$, the partial strings v and x can be "pumped" around w:

$$S \Rightarrow^* uAy \Rightarrow^* uvAxy \Rightarrow^* uv^2Ax^2y \Rightarrow^* \ldots \Rightarrow^* uv^iAx^iy \Rightarrow^* uv^iwx^iy.$$

Thus $uv^i wx^i y \in L$ for all $i = 0, 1, 2, \ldots$.

Since G is in Chomsky normal form and $A \Rightarrow^* vAx$, we must have $|vx| \geq 1$.

Further, since A is chosen so that its second lowest occurrence is no higher than at height $k+1$ from the leaves of the parse tree, the length of the string derived from the second lowest occurrence of A is bounded by $|vwx| \leq 2^{k+1} = n$. □

Example. Observe the language

$$L = \{a^k b^k c^k \mid k \geq 0\}.$$

Assume that L were context-free; choose the parameter n according to the lemma and examine the string $z = a^n b^n c^n \in L$.

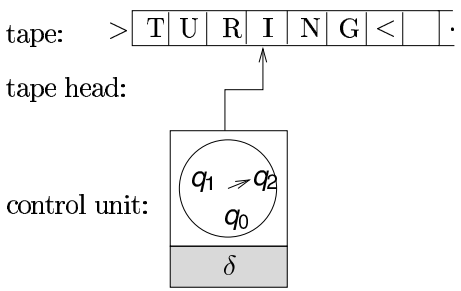
By Lemma 3.9 z can be split into parts

$$z = uvwxy, \qquad |vx| \geq 1, \qquad |vwx| \leq n.$$

By the last condition the string vx cannot contain both a's, b's, c's. The string $uv^0wx^0y = uwy$ has an excess of some symbol, and it cannot be of the form required for belonging to L, even though according to the lemma we should have $uwy \in L$.

4. TURING MACHINES

Alan Turing 1935–36.



A Turing machine is like a finite state automaton that can store information on a tape. The machine may move the tape head left or right; it can also read or write the character at the tape head. The tape is unbounded to the right.
The Church–Turing thesis: Any mechanically solvable problem can be solved by a Turing machine.

Computational models equivalent to the Turing machine:

- Gödel–Kleene: recursively defined functions (1936),
- Church: λ-calculus (1936),
- Post (1936) and Markov (1951): string rewriting systems,
- all current programming languages.

Turingin machine $\equiv$ programming language.

Definition 4.1 A *Turing machine* is a 7-tuple

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

where

- Q is a finite set of *state*;
- Σ is the *input alphabet*;
- $\Gamma \supseteq \Sigma$ is the *tape alphabet* (assume that $>, < \notin \Gamma$);
- $\delta : (Q - \{q_{\text{acc}}, q_{\text{rej}}\}) \times (\Gamma \cup \{>, <\}) \to Q \times (\Gamma \cup \{>, <\}) \times \{L, R\}$ is the *transition function*;
- $q_0 \in Q$ is the *initial state*;
- $q_{\text{acc}} \in Q$ is the *accepting* and
- $q_{\text{rej}} \in Q$ the *rejecting final state*.

Interpretation of the transition function

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

From state q if the machine reads from the tape the character a, the machine moves to state q', writes the character b over the character it read, and moves the tape head one position to the direction Δ ($L \sim$ "left", $R \sim$ "right").

If $a =$ '$>$' or '$<$', there are limitations on the character that may be written and the direction the tape head may be moved, and the transition function is always undefined, if $q = q_{\text{acc}}$ or $q = q_{\text{rej}}$. If the machine ends up at either of these two final states, it halts immediately.

Of the values of the transition function

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

it is required that

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

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

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

The *configuration* of a Turing machine is a 4-tuple

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

where we may have $a = \varepsilon$, if also $u = \varepsilon$ or $v = \varepsilon$.

Interpretation: the machine is in state q, the contents of the tape from the beginning up to but not including the tape head is u, at the tape head is the character a, and to the right of the tape head to the end of the used section of tape there is the string v.

We may have $a = \varepsilon$, if the tape head is at the beginning or end of the used part of the tape. In the first case, the machine is thought to "see" the character '$>$' and in the second case the character '$<$'.

The *initial configuration with input* $x = a_1 a_2 \ldots a_n$ is the 4-tuple

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

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

The configuration (q, w) *leads directly* to configuration (q', w'), denoted by

$$(q, w) \vdash_M (q', w'),$$

if one of the following is true: for all $q, q' \in Q$, $u, v \in \Gamma^*$, $a, b \in \Gamma$ ja $c \in \Gamma \cup \{\varepsilon\}$:

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

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

if $\delta(q, >) = (q', >, R)$, then $(q, \underline{\varepsilon}cv) \vdash_M (q', \underline{c}v)$;

if $\delta(q, <) = (q', b, R)$, then $(q, u\underline{\varepsilon}) \vdash_M (q', ub\underline{\varepsilon})$;

if $\delta(q, <) = (q', b, L)$, then $(q, uc\underline{\varepsilon}) \vdash_M (q', u\underline{c}b)$;

if $\delta(q, <) = (q', <, L)$, then $(q, uc\underline{\varepsilon}) \vdash_M (q', u\underline{c})$.

Configurations of the form (q_{acc}, w) or (q_{rej}, w) lead to no other configuration. In such configurations the machine *halts*.

The configuration (q, w) *leads* to configuration (q', w'), denoted by

$$(q, w) \vdash_M^* (q', w'),$$

if there exists a sequence of configurations (q_0, w_0), (q_1, w_1), ..., (q_n, w_n), $n \geq 0$, such that

$$(q, w) = (q_0, w_0) \vdash_M (q_1, w_1) \vdash_M \cdots \vdash_M (q_n, w_n) = (q', w').$$

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

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

otherwise M *rejects* x.

The language M *recognizes* is

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

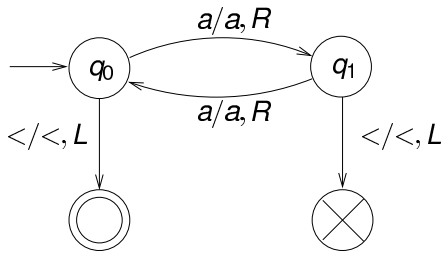
Example 1. The language $\{a^{2^k} \mid k \geq 0\}$ can be recognized by the Turing machine

$$M = (\{q_0, q_1, q_{\text{acc}}, q_{\text{rej}}\}, \{a\}, \{a\}, \delta, q_0, q_{\text{acc}}, q_{\text{rej}}),$$

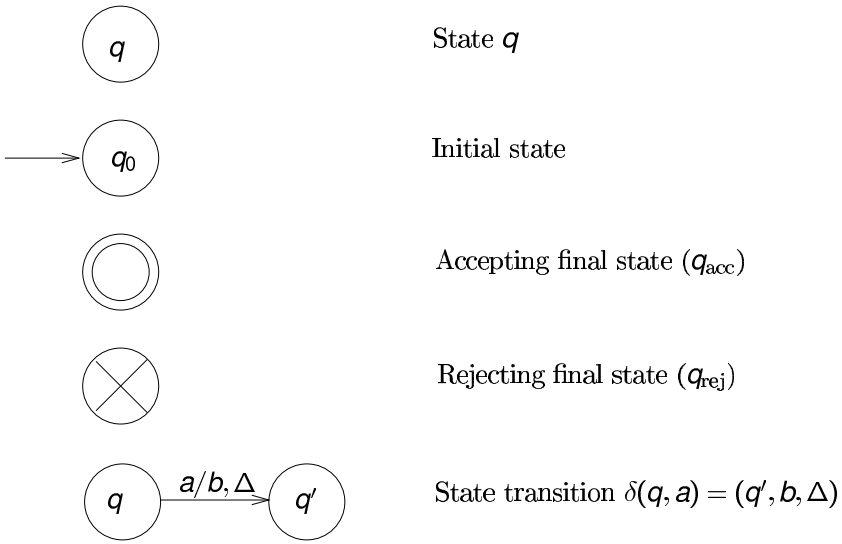
missä

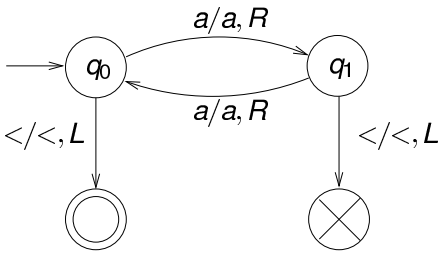
$$\begin{aligned} \delta(q_0, a) &= (q_1, a, R), \\ \delta(q_1, a) &= (q_0, a, R), \\ \delta(q_0, <) &= (q_{\text{acc}}, <, L), \\ \delta(q_1, <) &= (q_{\text{rej}}, <, L). \end{aligned}$$

A diagram presentation:



Notation used in diagram:



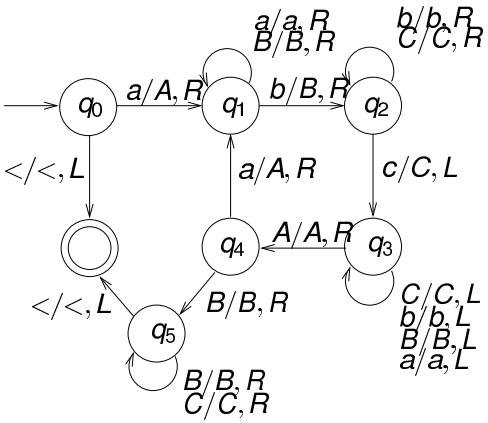


The computation of M with for example input *aaa* proceeds as follows:

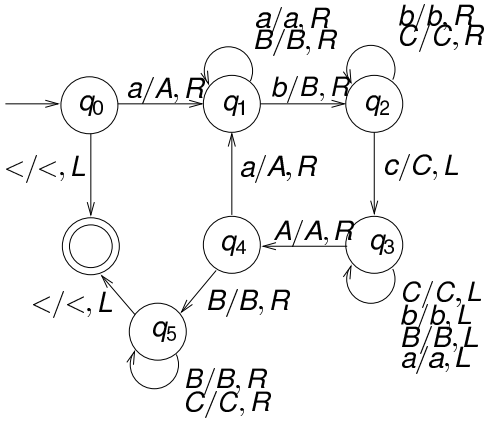
$$(q_0, \underline{a}aa) \underset{M}{\vdash} (q_1, a\underline{a}a) \underset{M}{\vdash} (q_0, aa\underline{a}) \underset{M}{\vdash} (q_1, aaa\underline{\varepsilon}) \underset{M}{\vdash} (q_{\text{rej}}, aa\underline{a}).$$

The machine halts in state q_{rej}, so $aaa \notin L(M)$.

Example 2. A machine that recognizes the language $\{a^k b^k c^k \mid k \geq 0\}$:



For clarity the rejecting final state q_{rej} is not shown explicitly. Then the interpretation is that all "missing" edges lead to this state.

The computation with input *aabbcc*:

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