1.3 Context-free languages

Example

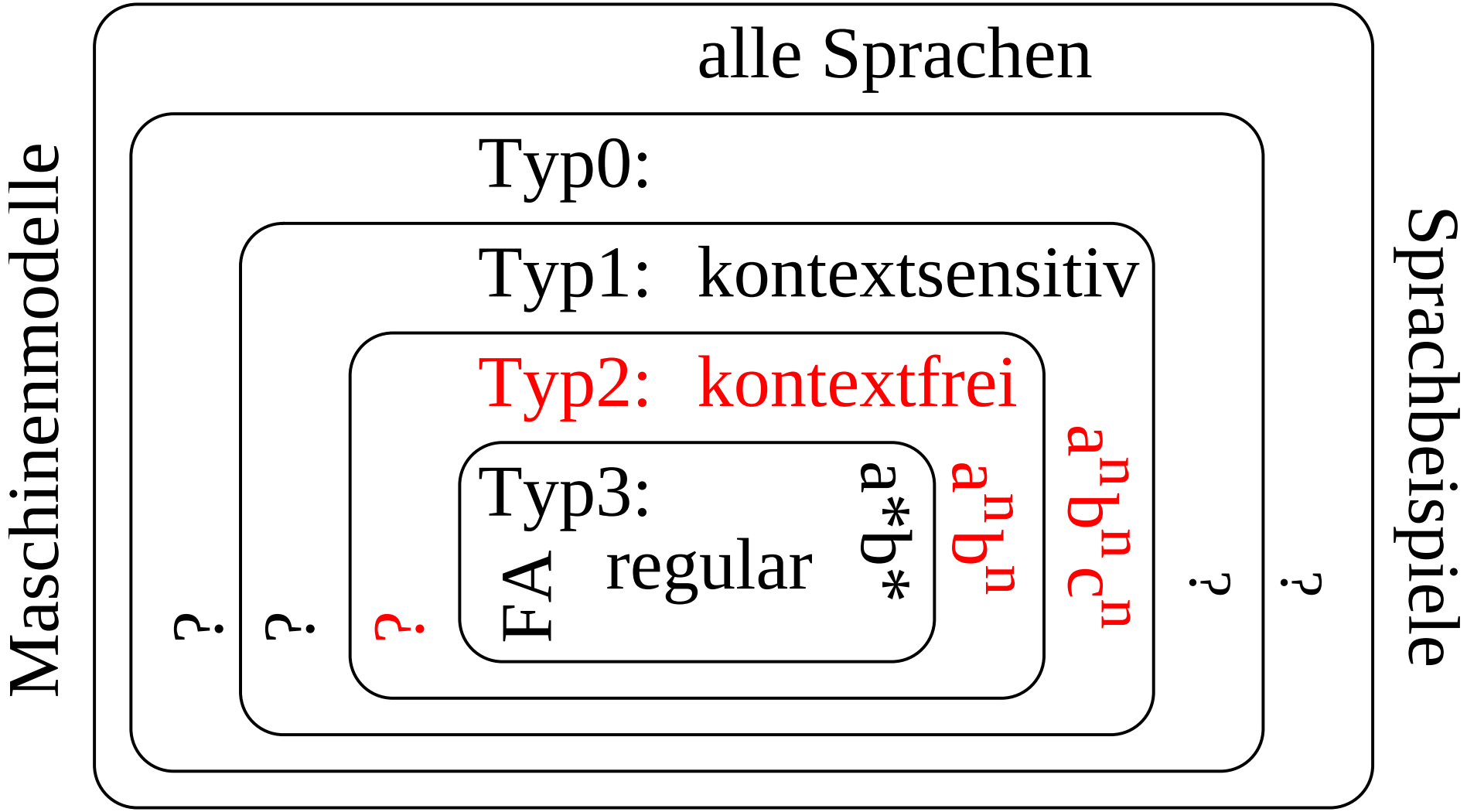
Arithmetical expressions: $G = (\{E,T,F\},\{a,+,*,(,)\},P,E)$ with

$$P = \{E \to T, E \to E+T, T \to F, T \to T*F, F \to a, F \to (E)\}$$

$\{a^n b^n : n \geq 1\}$: $G = (\{S\},\{a,b\},\{S \to ab, S \to aSb\},S)$

Syntax of program languages: Pascal, C, ...

Chomsky Hierarchy



Overview

1. Normal forms
2. Negative results by means of the Pumping-Lemma
3. Closer properties
4. Word problem
5. Pushdown automata

1.3.1 Normal forms

- ε-elimination
- Chomsky normal form (very simple productions)
- Greibach normal form

ε-Elimination

$\forall G = (V, \Sigma, P, S)$ with $P \subseteq V \times (V \cup \Sigma)^*$:

$\exists G' : L(G) = L(G'), G'$ is of type 2

ε elimination for $G = (V, \Sigma, P, S)$

$V_\varepsilon := \{\}$

while $\exists X \to \alpha \in P : X \notin V_\varepsilon \wedge \alpha \in V_\varepsilon^*$ **do** $V_\varepsilon := V_\varepsilon \cup \{X\}$

assert $V_\varepsilon = \left\{X \in V : X \overset{*}{\Rightarrow} \varepsilon\right\}$

while $\exists X \to \alpha Y \beta \in P : Y \in V_\varepsilon \wedge X \to \alpha\beta \notin P$ **do**

$P := P \cup X \to \alpha\beta$ // **invariant** : $L(G)$

$P := P \setminus (V \times \{\varepsilon\})$ // **invariant** : $L(G) \setminus \{\varepsilon\}$

if $S \in V_\varepsilon$ **then return** $(V \cup \{S'\}, \Sigma, P \cup \{S' \to \varepsilon, S' \to S\}, S')$

else return (V, Σ, P, S)

Exercise: Linear time algorithm for detecting of V_ε.

$(\mathcal{O}(|V| + \sum_{X \to r \in P} |r|))$

Exercise a^*b^*

$$G = (\{S,A,B\},\{a,b\},P,S),$$
$$P = \{S \to AB, A \to aA, A \to \varepsilon, B \to bB, B \to \varepsilon\}$$

while $\exists X \to \alpha \in P : X \notin V_\varepsilon \wedge \alpha \in V_\varepsilon^*$ **do** $V_\varepsilon := V_\varepsilon \cup \{X\}$

$$V_\varepsilon : \{\} \rightsquigarrow \{A\} \rightsquigarrow \{A,B\} \rightsquigarrow \{A,B,S\}$$

Exercise a^*b^*

$$G = (\{S,A,B\},\{a,b\},P,S),$$
$$P = \{S \to AB, A \to aA, A \to \varepsilon, B \to bB, B \to \varepsilon\}$$
$$V_\varepsilon = \{A,B,S\}$$

while $\exists X \to \alpha Y \beta \in P : Y \in V_\varepsilon \wedge X \to \alpha\beta \notin P$ **do** $P := P \cup X \to \alpha\beta$

$A \to aA, A \in V_\varepsilon \rightsquigarrow A \to a$

$B \to bB, B \in V_\varepsilon \rightsquigarrow B \to b$

$S \to AB, A \in V_\varepsilon \rightsquigarrow S \to B$

$S \to AB, B \in V_\varepsilon \rightsquigarrow S \to A$

$S \to A, A \in V_\varepsilon \rightsquigarrow S \to \varepsilon$

$S \to B, B \in V_\varepsilon$ but $S \to \varepsilon \in P$

$$P = \{S \to AB, A \to aA, A \to \varepsilon, B \to bB, B \to \varepsilon,$$
$$A \to a, B \to b, S \to B, S \to A, S \to \varepsilon\}$$

Exercise a^*b^*

$$G = (\{S,A,B\},\{a,b\},P,S),$$
$$P = \{S \to AB, A \to aA, A \to \varepsilon, B \to bB, B \to \varepsilon\}$$
$$V_\varepsilon = \{A,B,S\}$$
$$P := \{S \to AB, A \to aA, A \to \varepsilon, B \to bB, B \to \varepsilon,$$
$$A \to a, B \to b, S \to B, S \to A, S \to \varepsilon\}$$

$$P := P \setminus (V \times \{\varepsilon\})$$

$$P := \{S \to AB, A \to aA, B \to bB,$$
$$A \to a, B \to b, S \to B, S \to A\}$$

Return $(\{S',S,A,B\},\{a,b\},P,S'),$

$$P := \{S' \to \varepsilon, S' \to S, S \to AB, A \to aA, B \to bB,$$
$$A \to a, B \to b, S \to B, S \to A\}$$

Proof of correctness (sketch)

- $X \in V_\varepsilon \longrightarrow X \stackrel{*}{\Rightarrow} \varepsilon$: given derivation
- $X \stackrel{*}{\Rightarrow} \varepsilon \longrightarrow X \in V_\varepsilon$: induction by the length of derivation
- Loop invariant
- Termination !
- Elimination of the ε-productions does not change $L(G) \setminus \{\varepsilon\}$:

  Induction by the length of derivation. Replace the derivation $\gamma X \delta \stackrel{X \to \alpha Y \beta}{\Rightarrow} \gamma \alpha Y \beta \delta \stackrel{Y \to \varepsilon}{\Rightarrow} \gamma \alpha \beta \delta$ by $\gamma X \delta \stackrel{X \to \alpha \beta}{\Rightarrow} \gamma \alpha \beta \delta$.

Proof of correctness — termination

assert $V_\varepsilon = \left\{ X \in V : X \overset{*}{\Rightarrow} \varepsilon \right\}$

while $\exists X \to \alpha Y \beta \in P : Y \in V_\varepsilon \wedge X \to \alpha\beta \notin P$ **do**

$P := P \cup \{X \to \alpha\beta\}$

Let $k := \max\{|r| : X \to r \in P\}$.

Observation: There are new productions $X \to w \in P$, $|w| < k$.

But there are only finitely many productions of bounded length.

Elimination of cyclic unit productions

$G = (V, \Sigma, P, S)$ context-free with no ε-productions

Consider the Graph $U = (V, P \cap V \times V)$ of the unit productions.

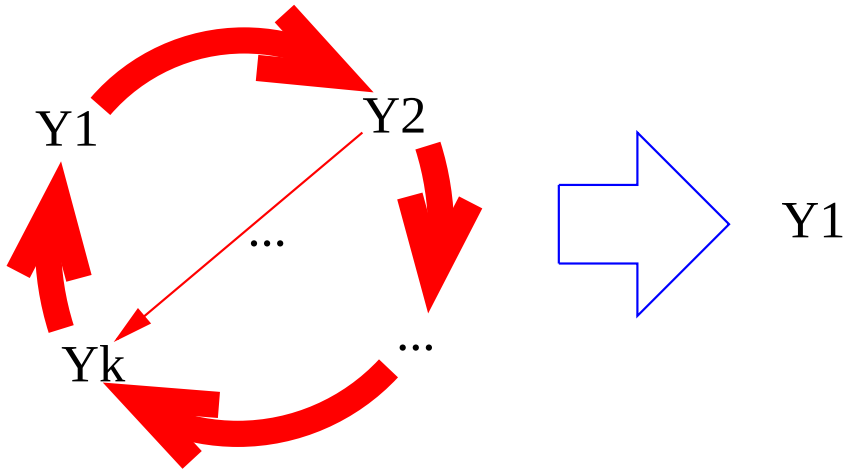
while $\exists$cycle $\{Y_1, \ldots, Y_k\}$ in U, $k \geq 1$ **do**

invariant : $L(G)$

replace $Y_2, \ldots, Y_k$ in G by Y_1

$P := P \setminus \{Y_1 \rightarrow Y_1\}$

assert U is cycle-free



Graph theoretic point of view:

Contraction of strongly connected components

Elimination of noncyclic unit productions

invariant Unit productions graph U is cycle free

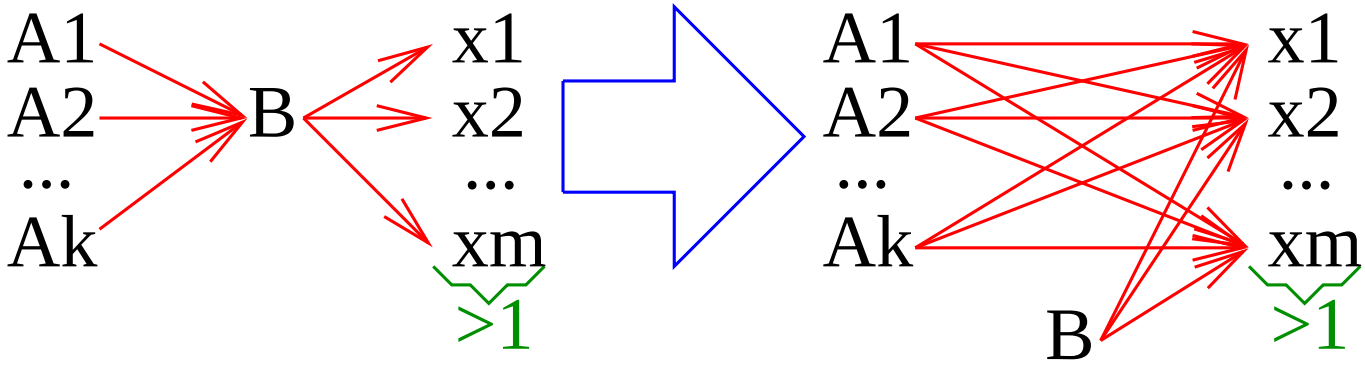
while $\exists X \to Y \in P \cap V \times V : P \cap \{Y\} \times V = \emptyset$ **do**

invariant : $L(G)$

$P := P \cup \{X \to x : X \to Y \in P \wedge Y \to x \in P\}$

$P := P \setminus V \times \{Y\}$

assert U is leer



Graph theoretic point of view: Handle in reverse topological sorting order

Chomsky normal form

A grammar $G' = (V, \Sigma, P, S)$ is in Chomsky normal form if $P \subseteq V \times \Sigma \cup V \times VV$.

Proposition: For every context-free grammar G with $\varepsilon \notin L(G)$, $\exists G'$ in Chomsky normal form, so that $L(G) = L(G')$.

Proof of the Proposition: Step-by-step change the grammar G.

Invariant: $L(G)$ remains unchangeable.

1. ε elimininination
2. unit productions elimination

Elimination of mixed right sides

foreach $a \in \Sigma$ **do**

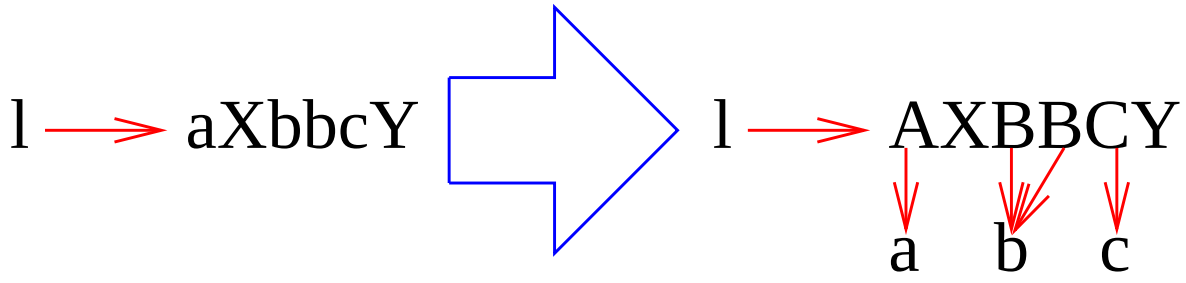
V_a:= new variable

$P := P \cup \{V_a \to a\}$

foreach $\ell \to r \in P$ **do**

if $a \in r \wedge |r| \geq 2$ **then** replace a by V_a in $V \to r$

assert $P \subseteq V \times \Sigma \cup V \times V^*$



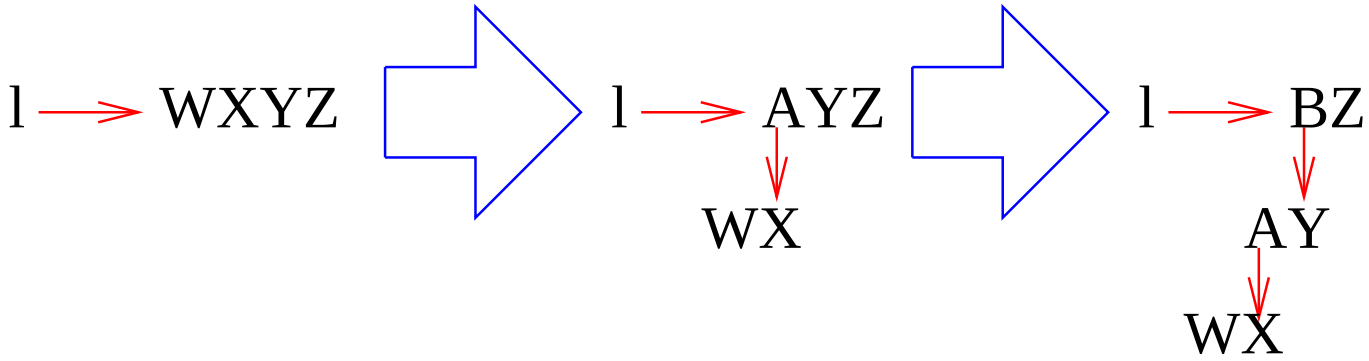
Elimination of longer right sides

while $\exists X \to Y_1 Y_2 Y_3 \cdots Y_k \in P$ with $k \geq 3$ **do**

C:= new variable

$P := P \cup \{C \to Y_1 Y_2, X \to C Y_3 \cdots Y_k\} \setminus \{X \to Y_1 Y_2 \cdots Y_k\}$

The loop halts, since $\sum_{\ell \to r \in P} \max(0, |r| - 2)$ decreases.



Exercise

$$\{E \to E+E, E \to E*E, E \to (E), E \to a\}$$

$\rightsquigarrow$

$$\{E \to EV_+E, E \to EV_*E, E \to V_(EV_), E \to a,$$
$$V_+ \to +, V_* \to *, V_(\to (, V_) \to)\}$$

$\rightsquigarrow$

$$\{C \to EV_+, E \to CE,$$
$$D \to EV_*, E \to DE,$$
$$F \to V_(E, E \to FV_),$$
$$E \to a,$$
$$V_+ \to +, V_* \to *, V_(\to (, V_) \to)\}$$

Greibach normal form

A grammar $G' = (V, \Sigma, P, S)$ is in Greibach normal form if $P \subseteq V \times \Sigma V^*$.

Proposition: For every context-free grammar G with $\varepsilon \notin L(G)$, $\exists G'$ in Greibach normal form, so that $L(G) = L(G')$.

Proof: not here

Idea: a natural generalization of type 3 grammar

1.3.2 The Pumping Lemma

L context-free

$$\rightarrow \exists n \in \mathbb{N} : \forall z \in L : |z| > n$$

$$\rightarrow \exists u, v, w, x, y : z = uvwxy \wedge |vx| \geq 1 \wedge |vwx| \leq n \wedge$$

$$\forall i \in \mathbb{N}_0 : uv^i wx^i y \in L$$

By words:

Sufficiently long words in a context-free language remain in it by iteration of one or two not trivial substring „pumping“.

Proof of the Pumping Lemma

Let $G = (V, \Sigma, P, S)$ be a grammar in Chomsky normal form for $L - \{\varepsilon\}$.

Let $k = |V|$, $n = 2^k$,

$z \in L$ with $|z| = m \geq n$ arbitrary.

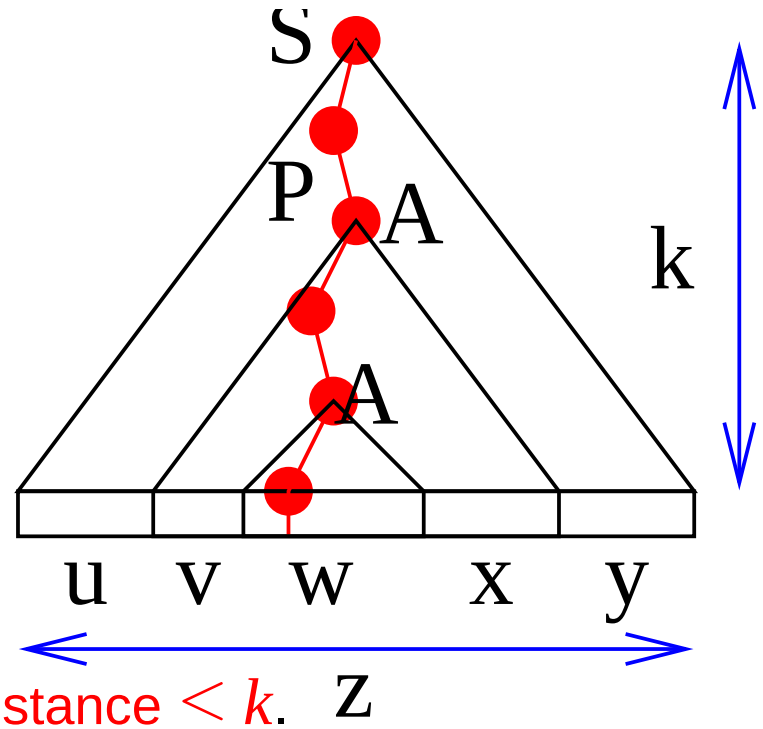
Consider one syntax tree for z.

Max. degree ≤ 2, $\geq n = 2^k$ leaves

$\rightarrow \exists$ path P with a length $|P| > k$.

$\rightarrow \geq k + 1$ variables in P.

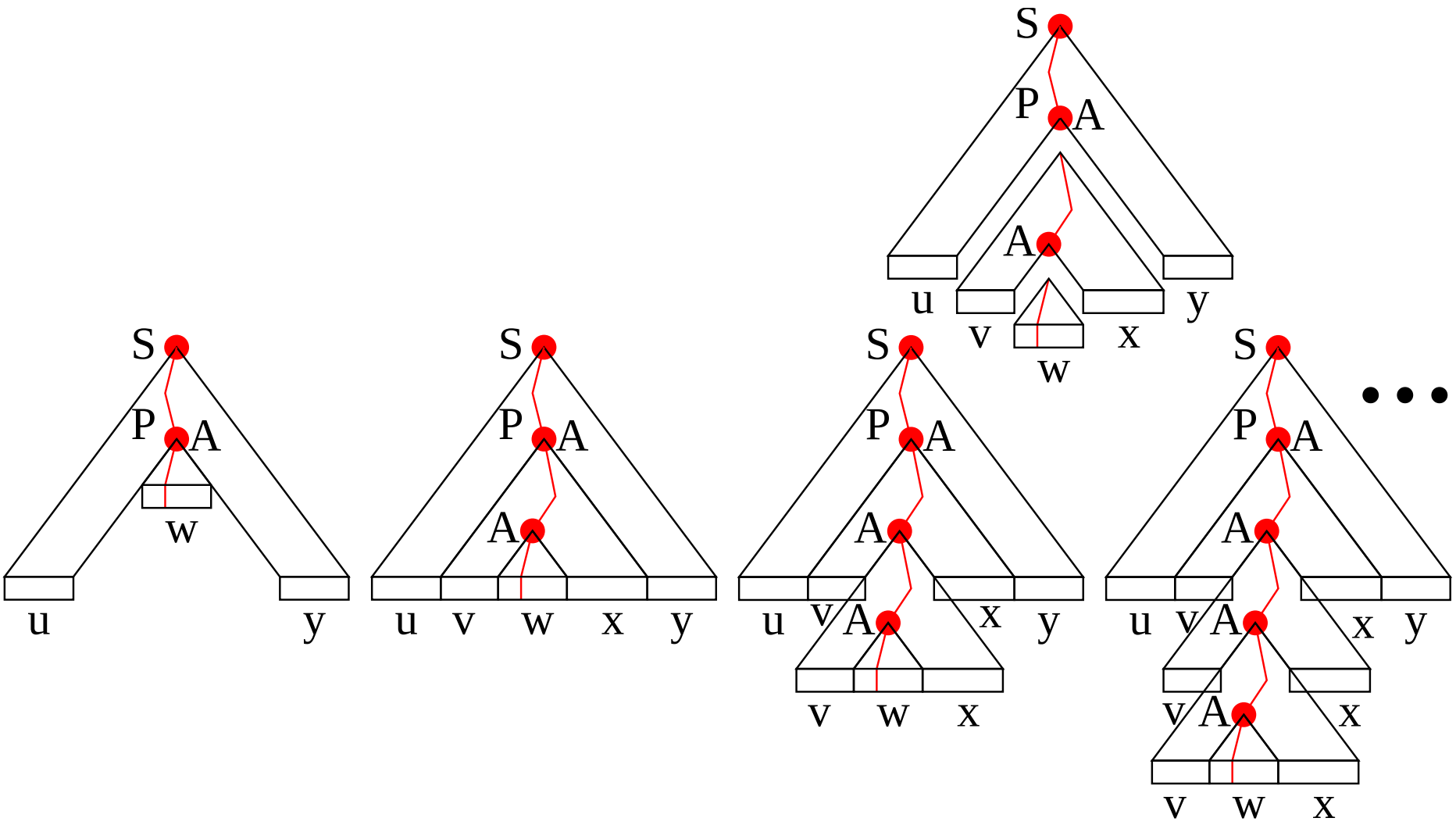
$\rightarrow \exists$ variable $A \in P$: A appears $\geq 2\times$.



The second appearance from down up has distance $\leq k$.

$\rightarrow$ stands for the substring vwx with length $\leq n$

Construction of the iterations:



Lemma: A binary tree (knots with degree 0 or 2) with $\geq 2^k$ leaves contains a path P with length $\geq k$.

Proof: Induction on k.

Case $k = 0$: 1 leave, path length 0.

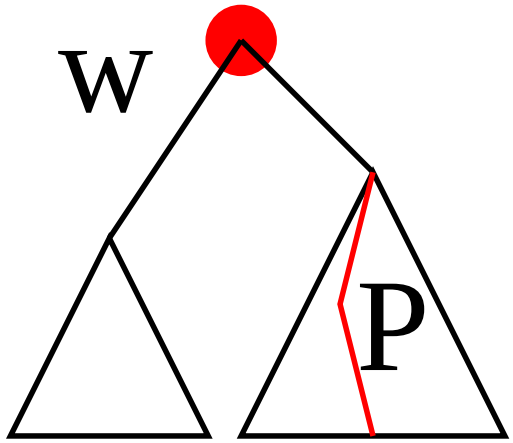
Case $k \rightsquigarrow k+1$:

$\geq 2^{k+1}$ leaves. The root w has degree 2.

At least one subtree has $\geq 2^k$ leaves

and contains a path P of length $\geq k$.

So wP has length $\geq k+1$.



$L = \{a^m b^m c^m : m \geq 1\}$ is not context-free

Assume that L is context-free.

Let n be the number from the Pumping lemma.

Consider $z = a^n b^n c^n$ and decomposition

$z = uvwxy$ by the Pumping lemma with

$|vwx| \leq n$, $|vx| \geq 1$, $uv^0wx^0y = uwy \in L$.

vx could not contain a-s, b-s and c-s.

$\longrightarrow$ the balance of a-s, b-s and c-s in uwy does not hold.

$\longrightarrow uwy \notin L$

A contradiction

Corollary: type 2$\neq$type 1

$L = \{ww : w \in \{a,b\}^*\}$ is not context-free

Assume that L is context-free.

Let n be the number from the Pumping lemma.

Consider $z = a^n b^n a^n b^n$ and a decomposition

$z = uvwxy$ by the Pumping lemma with $|vwx| \leq n$, $|vx| \geq 1$,

$z' := uwy \in L$.

Case $vx = a^k b^j$ is in the left half:

$\longrightarrow z' = a^{n-k} b^{n-j} a^n b^n$. A contradiction.

Case vx lies in the right half: analoguesly

or (in the middle): $vx = b^k a^j$

$\longrightarrow z' = a^n b^{n-k} a^{n-j} b^n$. A contradiction.

Rules for the proof by the Pumping Lemma

1. Let n be the number from the Pumping lemma.
2. Consider $z = ???$ ($|z| \geq n$) and a decomposition $z = uvwxy$ by the Pumping lemma with $|vwx| \leq n$, $|vx| \geq 1$
   - Every z with $|z| \geq n$ is allowed. The “creative” part !
   - The choice of the word enable us to make the proof possible/simple
   - Since $|vwx| \leq n$ contains blocks of length n we have the following cases
3. Cases for all possible decompositions $z = uvwxy$. For each case:
   Find an $i \geq 0$, so that $uv^i wx^i y \notin L(G)$.
   Typical values: $i = 0$, $i = 2$.
   Challenge: Make the number of cases smaller .

1.3.3 Closer properties

Context-free languages are

closed under

- union $\cup$
- concatenation $\cdot$
- star operation $*$

not closed under

- intersection $\cap$
- complement $\overline{\cdot}$

Closure of CFG under $\cup$

Consider

$G_1 = (V_1, \Sigma, P_1, S_1)$,

$G_2 = (V_2, \Sigma, P_2, S_2)$,

Let $V_1 \cap V_2 = \emptyset$ and

$G = (\{S\} \cup V_1 \cup V_2, \Sigma, \{S \to S_1, S \to S_2\} \cup P_1 \cup P_2)$.

Evidently it holds

$L(G) = L(G_1) \cup L(G_2)$.

Closure of CFG under $\cdot$

Consider

$G_1 = (V_1, \Sigma, P_1, S_1)$,

$G_2 = (V_2, \Sigma, P_2, S_2)$,

Let $V_1 \cap V_2 = \emptyset$ and

$G = (\{S\} \cup V_1 \cup V_2, \Sigma, \{S \to S_1 S_2\} \cup P_1 \cup P_2)$.

Evidently it holds

$L(G) = L(G_1) \cdot L(G_2)$.

Closure of CFG under *

Consider

$G_1 = (V_1, \Sigma, P_1, S_1)$

and let S_1 appear in no right side in P . And

$G = (\{S\} \cup V_1, \Sigma, \{S \to \varepsilon, S \to S_1, S_1 \to S_1 S_1\} \cup P_1 \setminus \{S_1 \to \varepsilon\})$.

Clearly it holds

$L(G) = L(G_1)^*$.

Nonclosure of CFG under ∩

Consider the Context-free languages

$L_1 = \{a^i b^j c^j : i, j > 0\}$

$L_2 = \{a^i b^i c^j : i, j > 0\}$.

$L_1 \cap L_2 = \{a^i b^i c^i : i > 0\}$ is not context-free!

Nonclosure of CFG under $\bar{\cdot}$

Assume:

closure under $\cup$ and $\bar{\cdot}$.

$\rightarrow$

closure under $\cap$ will

$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

A contradiction

1.3.4 The CYK-algorithm — The word problem fr context-free grammars

Given: A grammar $G = (V, \Sigma, P, S)$,

A word $x = x_1 \cdots x_n \in \Sigma^*$.

Question: $x \in L(G)$?

Algorithm of Cocke, Younger and Kasami

Let G be in Chomsky normal form:

Special case $x = \varepsilon$ is easy,

since apriori is converted in Chomsky NF.

CYK Algorithm

We are solving more general problem:

For each substring $x_i \cdots x_{i+j-1}$ of the length j of x,

from which variable symbols is $x_i \cdots x_{i+j-1}$ derivable?

$$T[i,j] := \left\{ A \in V : A \overset{*}{\Rightarrow} x_i \cdots x_{i+j-1} \right\}$$

Case $j = 1$: $T[i,1] = \{A \in V : A \to x_i \in P\}$

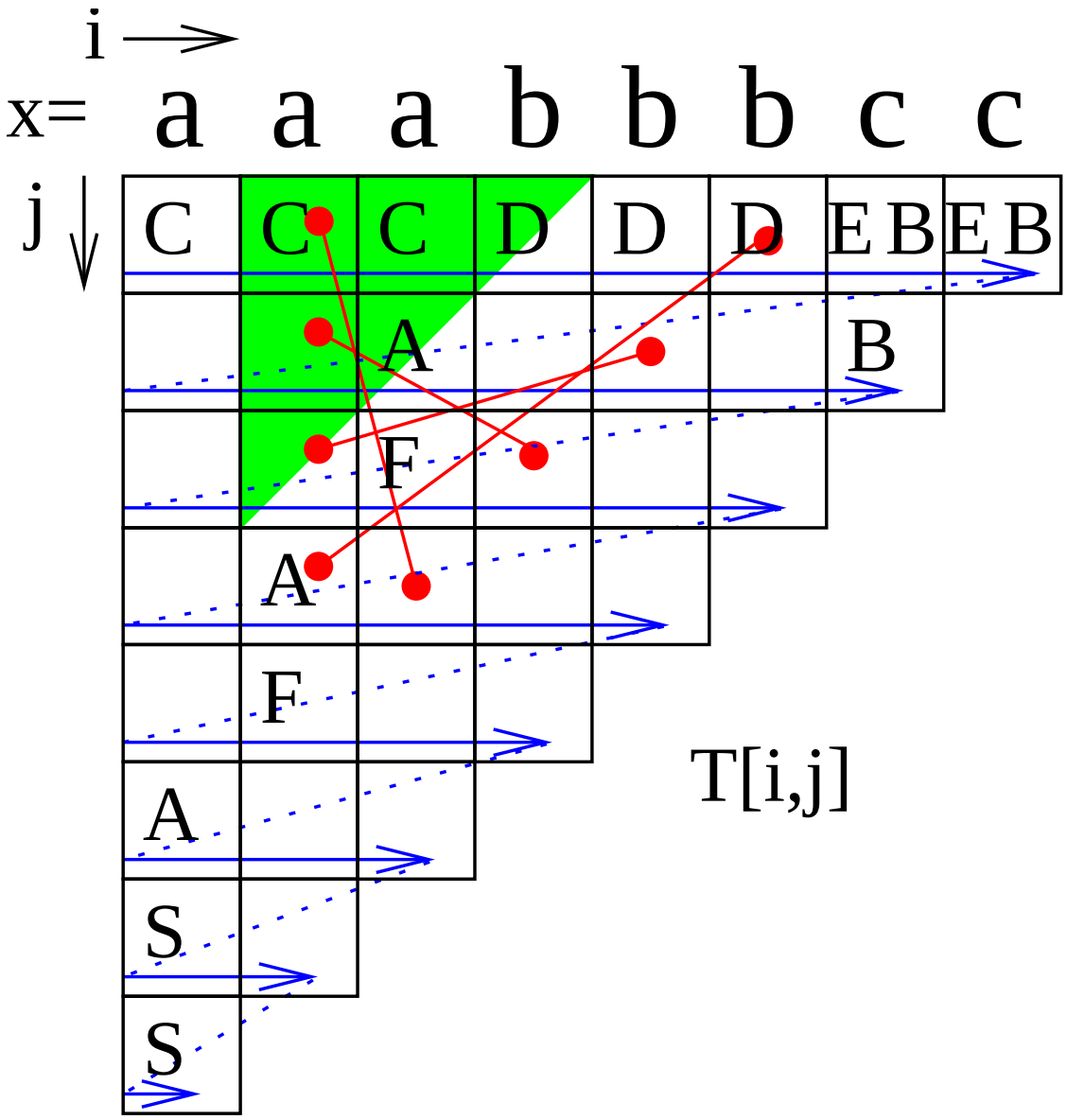
Otherwise:

$$T[i,j] := \{A \in V : \exists A \to BC \in P : \exists k \in \{1,\ldots,j-1\} : B \in T[i,k] \wedge C \in T[i+k, j-k]\}$$

In the end: $S \in T[1,n]$?

Exercise

$G = (\{S, A, B, C, D, E, F\},$
$\{a, b, c\}, P, S),$
$P = \{$
$S \rightarrow AB,$
$A \rightarrow CD,$
$A \rightarrow CF,$
$B \rightarrow c,$
$B \rightarrow EB,$
$C \rightarrow a,$
$D \rightarrow b,$
$E \rightarrow c,$
$F \rightarrow AD\}$



Implementation by

dynamic programming

for $i := 1$ **to** n **do** $T[i,1] := \{A \in V : A \to x_i \in P\}$

for $j := 2$ **to** n **do**

  for $i := 1$ **to** $n - j + 1$ **do**

    $T[i,j] := \emptyset$

    for $k := 1$ **to** $j - 1$ **do**

      $T[i,j] = \cup\{A : \exists A \to BC \in P : B \in T[i,k] \wedge C \in T[i+k, j -$

return $S \in T[1,n]$

Analysis

Let $V = 1..|V|$

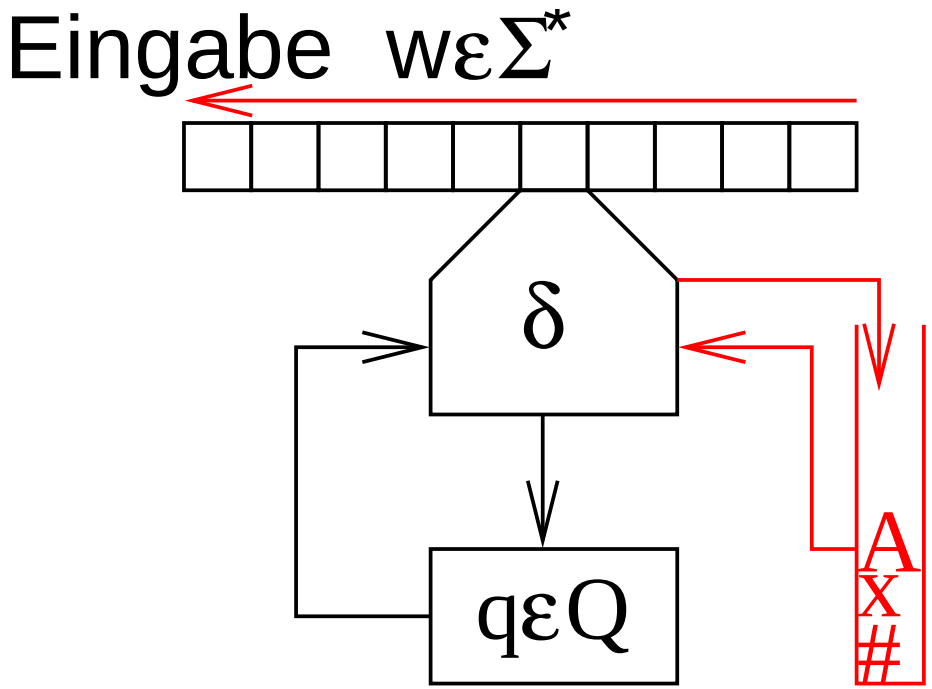
for $i := 1$ **to** n **do** $T[i,1] := \{A \in V : A \to x_i \in P\}$ // "cheep"
for $j := 2$ **to** n **do** // $\leq n$ times
  for $i := 1$ **to** $n - j + 1$ **do** // $\leq n$ times
    $T[i,j] := \emptyset$ // Binary vector of the size $|V| \leq |P|$
    for $k := 1$ **to** $j - 1$ **do** // $\leq n$ times
      foreach $A \to BC \in P$ **do** // $\leq |P|$ times
        if $B \in T[i,k] \wedge C \in T[i+k, j-k]$ **then** // $\mathcal{O}(1)$
          insert A into $T[i,j]$ // $\mathcal{O}(1)$
return $S \in T[1,n]$

Time: $\mathcal{O}(n \times n \times (|V| + n \times |P|)) = \mathcal{O}\left(n^3 |P|\right)$

1.3.5 Push down (stack) automata

$K = (Q, \Sigma, \Gamma, \delta, s, \#)$:

- Q, states
- Σ, alphabet
- Γ stack alphabet,
  $\Sigma \cup \{\#\} \subseteq \Gamma$
- $\delta : Q \times (\Sigma \cup \varepsilon) \times \Gamma \to 2^{Q \times \Gamma^*}$,
  transition function; (finite sets)
- $s \in Q$, input state
- $\# \notin \Sigma$: end of stack,

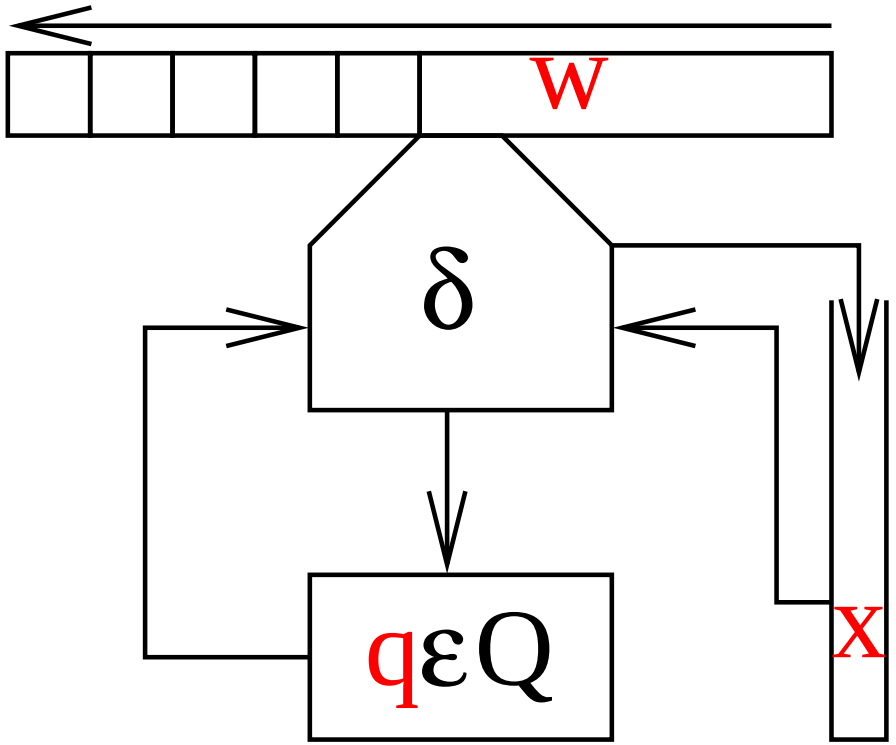


$\approx \varepsilon$NFA $+$ stack memory $-$ final states

$\approx$ special 2-tape-Turing machine

Configuration of a stack machine

$(q, w, x) \in Q \times \Sigma^* \times \Gamma^*$



Functionality of a stack machine

Possible transitions between configurations.

Consumption of an input symbol a:

$$(q, aw, bx) \overset{(q',x')\in\delta(q,a,b)}{\vdash} (q', w, x'x)$$

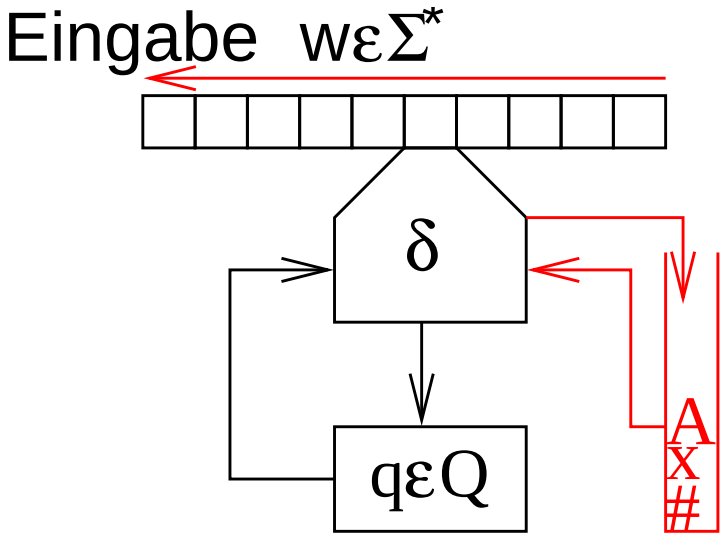
ε-transition:

$$(q, w, bx) \overset{(q',x')\in\delta(q,\varepsilon,b)}{\vdash} (q', w, x'x)$$

Stack machine as an acceptor

$K = (Q, \Sigma, \Gamma, \delta, s, \#)$.

$L(K)$?



Definition:

K accepts $w \in \Sigma^*$ iff

$\exists$ a sequence of (by δ admitted)

configuration transitions

$(s, w, \#) \vdash \cdots \vdash (q, \varepsilon, \varepsilon)$ with $q \in Q$ arbitrary.

“Acceptation by an empty stack”

$L(K) := \{w \in \Sigma^* : K \text{ accepts } w\}$.

Exercise: $\{w\$w^R : w \in \{a,b\}^*\}$

$K = (\{0,1\}, \{a,b,\$\}, \{a,b,\#\}, \delta, 0, \#)$

$\delta(0,\$,k) = \{(1,k)\}$

$\delta(0,i,k) = \{(0,ik)\}$ for $i \in \{a,b\}$

$\delta(1,i,i) = \{(1,\varepsilon)\}$

$\delta(1,\varepsilon,\#) = \{(1,\varepsilon)\}$

$(0, ba\$ab, \#) \vdash$

$(0, a\$ab, b\#) \vdash$

$(0, \$ab, ab\#) \vdash$

$(1, ab, ab\#) \vdash$

$(1, b, b\#) \vdash$

$(1, \varepsilon, \#) \vdash$

$(1, \varepsilon, \varepsilon)$

Exercise: $\left\{ww^R : w \in \{a,b\}^*\right\}$

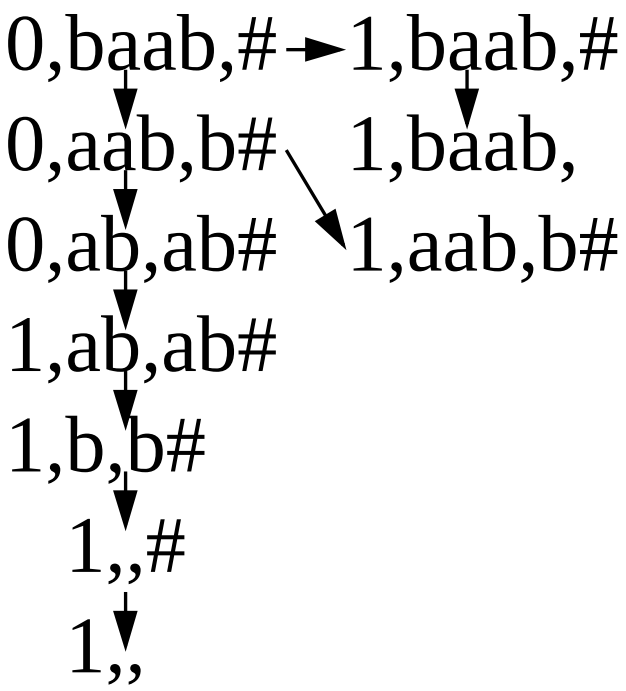
$K = (\{0,1\},\{a,b\},\{a,b,\#\},\delta,0,\#)$

$\delta(0,\varepsilon,k) = \{(1,k)\}$

$\delta(0,i,k) = \{(0,ik)\}$

$\delta(1,i,i) = \{(1,\varepsilon)\}$

$\delta(1,\varepsilon,\#) = \{(1,\varepsilon)\}$



Proposition: L is context-free iff $\exists$ a nondeterministic stack automaton (NstackA) $M : L(M) = L$

Proof: L is context-free $\longrightarrow \exists$ NstackA $M : L(M) = L$

Let $G = (V, \Sigma, P, S)$ be a grammar with $L(G) = L$.

Consider NstackA $M = (\{z\}, \Sigma, V \cup \Sigma, \delta, z, S)$ with

$$\forall A \to \alpha \in P : (z, \alpha) \in \delta(z, \varepsilon, A),$$

$$\forall a \in \Sigma : (z, \varepsilon) \in \delta(z, a, a)$$

Idea (Invariant): stack stores the proposition form of one derivation minus by the input terminal symbols.

$G = (V, \Sigma, P, S)$ grammar with $L(G) = L$.

$M = (\{z\}, \Sigma, V \cup \Sigma, \delta, z, S)$ with

$\forall A \to \alpha \in P : (z, \alpha) \in \delta(z, \varepsilon, A), \forall a \in \Sigma : (z, \varepsilon) \in \delta(z, a, a)$

$x = x_1 \cdots x_n \in L(G)$

$\longrightarrow \exists$ left derivation $S \Rightarrow \cdots \Rightarrow x$

We construct a sequence of configurations of M

$(z, x, S) \vdash \cdots \vdash (z, \varepsilon, \varepsilon)$:

- Configuration $(z, x_i \cdots x_n, Vy)$: left derivation replacement
  $V \to \alpha \in P$ gives us $(z, \alpha) \in \delta(z, \varepsilon, V)$.
  So the new configuration is $(z, x_i \cdots x_n, \alpha y)$.
- Configuration $(z, x_i \cdots x_n, x_i y)$:
  New configuration $(z, x_{i+1} \cdots x_n, y)$

$G = (V, \Sigma, P, S)$ grammar with $L(G) = L$.

$M = (\{z\}, \Sigma, V \cup \Sigma, \delta, z, S)$ with

$\forall A \to \alpha \in P : (z, \alpha) \in \delta(z, \varepsilon, A)$ and $\forall a \in \Sigma : (z, \varepsilon) \in \delta(z, a, a)$

$x = x_1 \cdots x_n \in L(M)$

$\longrightarrow \exists$ a sequence of configurations of M

$(z, x, S) \vdash \cdots \vdash (z, \varepsilon, \varepsilon)$:

We construct a left derivation $S \Rightarrow \cdots \Rightarrow x$.

The configuration transitions without consumption of input symbols give us turning towards the substitutions.

Proof: $\exists$ NstackA $M : L(M) = L \longrightarrow L$ is context-free.

Let $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$ be NstackA with $L(M) = L$.

not here. (A little technical)

Proposition: For each NstackA with one state $M = (\{z\}, \Sigma, \Gamma, \delta, z)$ there is a grammar G with $L(G) = L(M)$.

Proof: Set $G = (\Gamma, \Sigma, P, \#)$ with

$P = \{A \to a\alpha : (z, \alpha) \in \delta(z, a, A)\}$ $a = \varepsilon$ allowed!

Part $L(G) \subseteq L(M)$:

$x \in L(G) \longrightarrow \exists$ left derivation $A = \# \overset{*}{\Rightarrow} x$.

We prove by induction:

For each prefix w of x: $\# \overset{*}{\Rightarrow} w\alpha$ with $w \in \Sigma^*$, $\alpha \in \Gamma^*$

$\exists$ a configuration sequence $(z, x, \#) \vdash \cdots \vdash (z, y, \alpha)$ of M with

$wy = x$.

The induction base: Prefix $\langle \# \rangle$ corresponds to configuration sequence $\langle (z, x, \#) \rangle$ ($y = x, w = \varepsilon, \alpha = \#$)

Induction step: $\# \stackrel{*}{\Rightarrow} wA\beta \Rightarrow wa\alpha\beta$. $a \in \varepsilon \cup \Sigma$.

By IH $\exists$ conf.-sequence $(z, x, \#) \vdash \cdots \vdash (z, ay, A\beta)$ with $way = x$.

Because of $A \longrightarrow a\alpha \in P$ it should be $(z, \alpha) \in \delta(z, a, A)$.

So, $(z, ay, A\beta) \vdash (z, y, \alpha\beta)$.

Hence $\# \stackrel{*}{\Rightarrow} x \longrightarrow (z, x, \#) \vdash^* (z, \varepsilon, \varepsilon)$.

Part $L(M) \subseteq L(G)$:

$x \in L(M) \longrightarrow \exists$ configuration sequence $(z,x,\#) \vdash \cdots \vdash (z,\varepsilon,\varepsilon)$

...

To this corresponds to a left derivation $A = \# \stackrel{*}{\Rightarrow} x$.

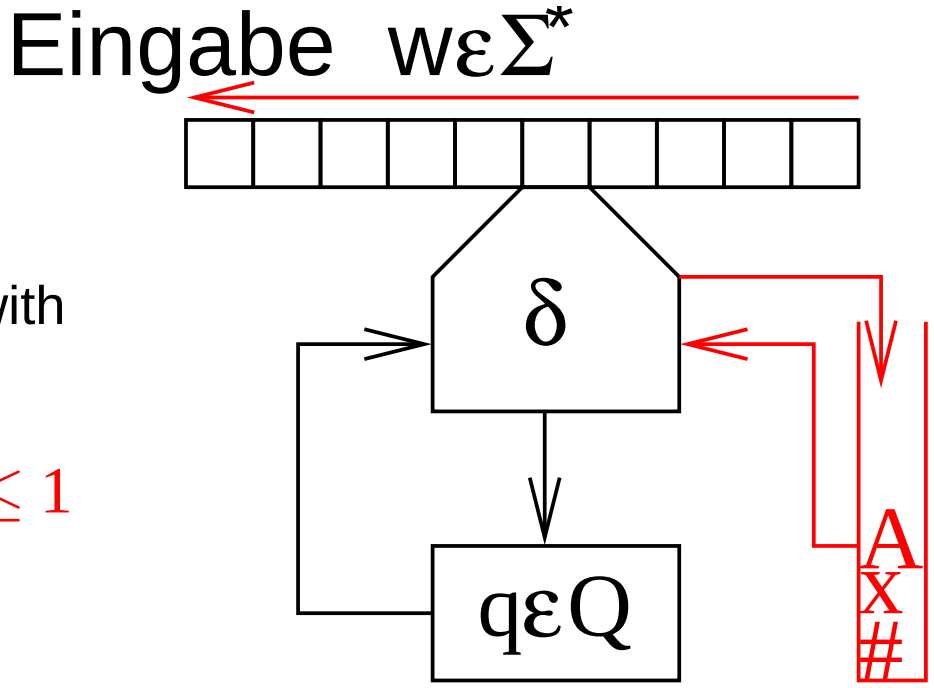
1.3.6 Deterministic context-free languages

$K = (Q, \Sigma, \Gamma, \delta, s, \#, F)$:

- Q, Σ, Γ, s, # we know.
- $\delta : Q \times (\Sigma \cup \varepsilon) \times \Gamma \to 2^{Q \times \Gamma^*}$, with

  $\forall z \in Q, a \in \Sigma, A \in \Gamma :$

  $$|\delta(z,a,A)| + |\delta(z,\varepsilon,A)| \leq 1$$



K accepts $w \in \Sigma^*$ iff

$\exists$ a sequence of (by δ admitted)

configuration transitions

$(s, w, \#) \vdash \cdots \vdash (f, \varepsilon, \varepsilon)$ with $f \in F$.

Proposition:

$\forall$ DstackA K : $\exists$ NstackA with one sate K' : $L(K) = L(K')$.

$\exists$ NstackA with one state K' : $\nexists$ DstackA K : $L(K) = L(K')$.

Proof:

not here.

Compilers

Observation:

The word problem for DstackA we could solve in linear time .

Remark:

The languages for DstackA are exactly die LR(k)-languages.

Such language families are the foundation of the syntax of many program languages.

Closure properties for DstackA

Without Proof:

Closed under: $\bar{\cdot}$, intersection with regular languages.

Not closed under: $\cup, \cap, \cdot, ^*$

1.3.7 Decidability for context-free languages

Emptiness problem

Function isEmpty$(G = (V, \Sigma, P, S))$

Marked $:= \Sigma$

while $\exists A \to \alpha \in P : A \notin$ Marked $\wedge\, \alpha \in$ Marked* **do**

Marked $:=$ Marked $\cup \{A\}$

return $S \notin$ Marked

It holds iff $L(G) = \emptyset$

Finiteness problem

Given: grammar $G = (V, \Sigma, P, S)$

Question: $|L(G)| < \infty$?

Let n be the number of the Pumping lemma.

Observation: $|L(G)| = \infty \Leftrightarrow \exists z \in L(G) : n \leq |z| < 2n$

Proof:

$z \in L(G), n \leq |z| < 2n \longrightarrow$ Pumping lemma gives $|L| = \infty$.

Case $|L(G)| = \infty$ consider $z \in L(G)$ with minimal $|z| \geq n$.

Assume $|z| \geq 2n$.

$\overset{\text{Pumping lemma}}{\longrightarrow} z = uvwxy,\ |vx| \geq 1,\ uwy \in L(G),\ |uwy| \geq n.$

A contradiction with the minimality of $|z|$.

Finiteness problem

Given: a grammar $G = (V, \Sigma, P, S)$

Question: $|L(G)| < \infty$?

Let n be the number of the Pumping lemma.

It holds: $|L(G)| = \infty \Leftrightarrow \exists z \in L(G) : n \leq |z| < 2n$

Brute Force Algorithm:

Word problem for all words z with length $n \leq |z| < 2n$.

Time exposure: $\mathcal{O}\left(8 \cdot 2^{3|V|} \cdot |\Sigma|^{2 \cdot 2^{|V|}}\right)$

!

Remark: There is an efficient algorithm.

Decidability problem for DstackA

- Equivalence: $L(K_1) = L(K_2)$?

Non decidable problems for CFG

- $L(G_1) \cap L(G_2) = \emptyset$? Disjunctives
- $|L(G_1) \cap L(G_2)| = \infty$?
- $L(G_1) \cap L(G_2)$ context-free?
- $L(G_1) \subseteq L(G_2)$?
- $L(G_1) = L(G_2)$?
- Ambiguity: $\exists x \in L(G) : |\{\text{syntax tree}(x)\}| \geq 2$
- Is $\overline{L(G)}$ context-free?
- Is $L(G)$ regular?
- Is $L(G)$ det. context-free?

Non-decidability of $L(G_1) \cap L(G_2) = \emptyset$?

We solve the PCP- Post Correspondence System

$K = (x_1, y_1) \cdots (x_k, y_k) \in (\{a,b\}^* \times \{a,b\}^*)^*$ by means of one assumed below program disjoint(G_1, G_2):

Define $\Sigma = \{a, b, 1, \ldots, k\}$

$G_1 = (\{S\}, \Sigma, P_1, S)$,

$G_2 = (\{S\}, \Sigma, P_2, S)$, with

$P_1 = \{S \to iSx_i, S \to ix_i : i \in 1..k\}$

$P_2 = \{S \to iSy_i, S \to iy_i : i \in 1..k\}$

$L(G_1) = \{i_n \cdots i_1 x_{i_1} \cdots x_{i_n} : i_\ell \in 1..k\}$

$L(G_2) = \{i_n \cdots i_1 y_{i_1} \cdots y_{i_n} : i_\ell \in 1..k\}$

$L(G_1) \cap L(G_2) \neq \emptyset$ iff.

$\exists i_1, \ldots i_n : x_{i_1} \cdots x_{i_n} = y_{i_1} \cdots y_{i_n}$ iff.

K has a solution.