1.3 Kontextfreie Sprachen

Beispiele

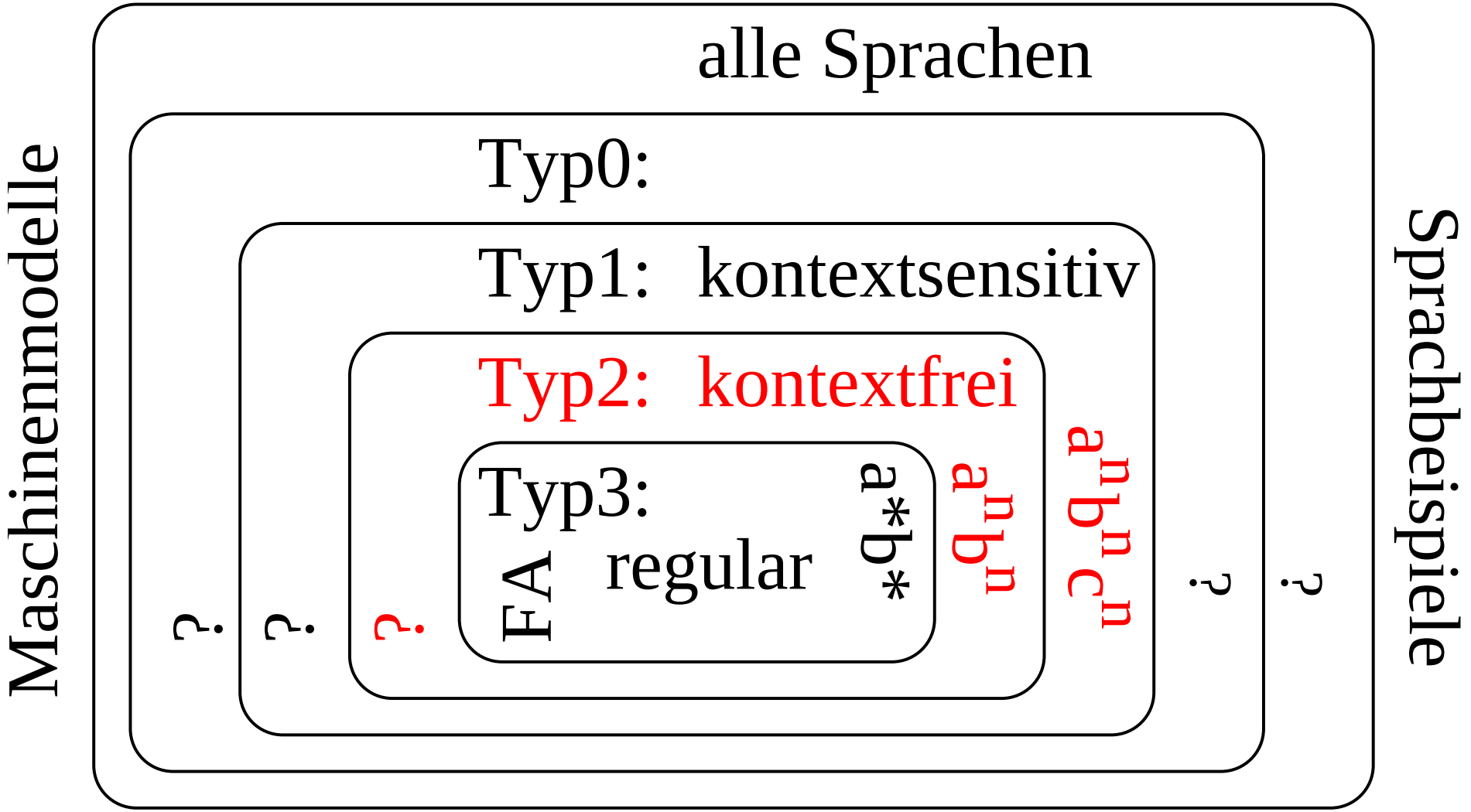
Klammerausdrücke: $G = (\{E,T,F\},\{a,+,*,(,)\},P,E)$ mit

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

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

Programmiersprachensyntax: Pascal, C, ...

Chomsky Hierarchie



Überblick

1. Normalformen
2. Unmöglichkeitsresultate mittels Pumping-Lemma
3. Abschlusseigenschaften
4. Wortproblem
5. Kellerautomaten

1.3.1 Normalformen

- ε-Eliminiation
- Umbenennungen eliminieren
- Chomsky Normalform (sehr einfache Produktionen)
- Greibach-Normalform

ε-Eliminiation

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

$\exists G' : L(G) = L(G')$, G' hat Typ 2

ε eliminieren für $G = (V, \Sigma, P, S)$

$V_\varepsilon := \{\}$

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)

Übung: Linearzeitalgorithmus zur Bestimmung von V_ε.

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

Beispiel 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** $V_\varepsilon := V_\varepsilon \cup \{X\}$

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

Beispiel 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** $P := P \cup X \to \alpha\beta$

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

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

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

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

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

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

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

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

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

Korrektheitsbeweis (Skizzenskizze)

- $X \in V_\varepsilon \longrightarrow X \stackrel{*}{\Rightarrow} \varepsilon$: Ableitung angeben
- $X \stackrel{*}{\Rightarrow} \varepsilon \longrightarrow X \in V_\varepsilon$: Induktion über Ableitungslänge
- Schleifeninvariante
- Terminierung !
- Entfernen der ε-Produktionen verändert $L(G) \setminus \{\varepsilon\}$ nicht:

  Induktion über Ableitungslänge. Ersetze Teilableitung
  $\gamma X \delta \stackrel{X \to \alpha Y \beta}{\Rightarrow} \gamma \alpha Y \beta \delta \stackrel{Y \to \varepsilon}{\Rightarrow} \gamma \alpha \beta \delta$ durch
  $\gamma X \delta \stackrel{X \to \alpha \beta}{\Rightarrow} \gamma \alpha \beta \delta$.

Korrektheitsbeweis — Terminierung

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

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

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

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

Beobachtung: Neue Produktionen in $X \to w \in P$ haben $|w| < k$.

Es gibt aber nur endlich viele Produktionen begrenzter Länge.

Eliminiation von zyklischen Einheitsproduktionen

$G = (V, \Sigma, P, S)$ kontextfrei ohne ε-Produktionen

Betrachte den Graphen $U = (V, P \cap V \times V)$ der Einheitsproduktionen.

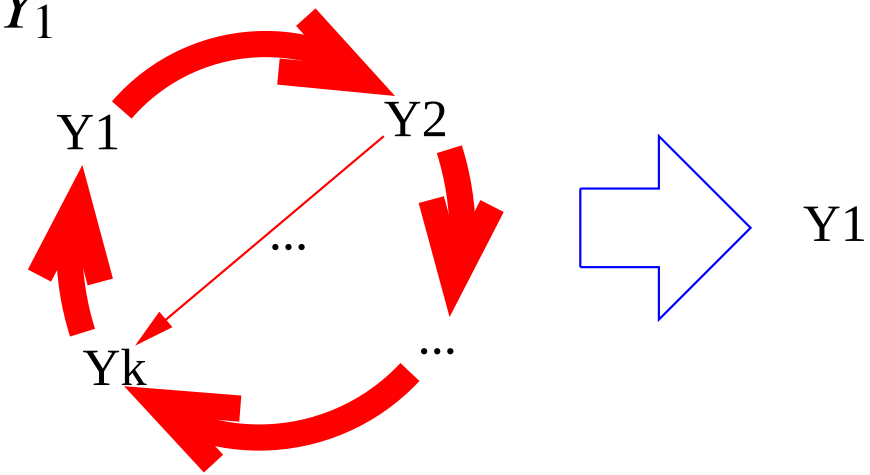
while $\exists$Kreis $\{Y_1, \ldots, Y_k\}$ in U, $k \geq 1$ **do**

invariant : $L(G)$

ersetze $Y_2, \ldots, Y_k$ in G durch Y_1

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

assert U ist kreisfrei



Graphentheoretische Sichtweise:

Kontraktion von starken Zusammenhangskomponenten

Eliminiation von nichtzyklischen Einheitsproduktionen

invariant Einheitsproduktionengraph U ist kreisfrei

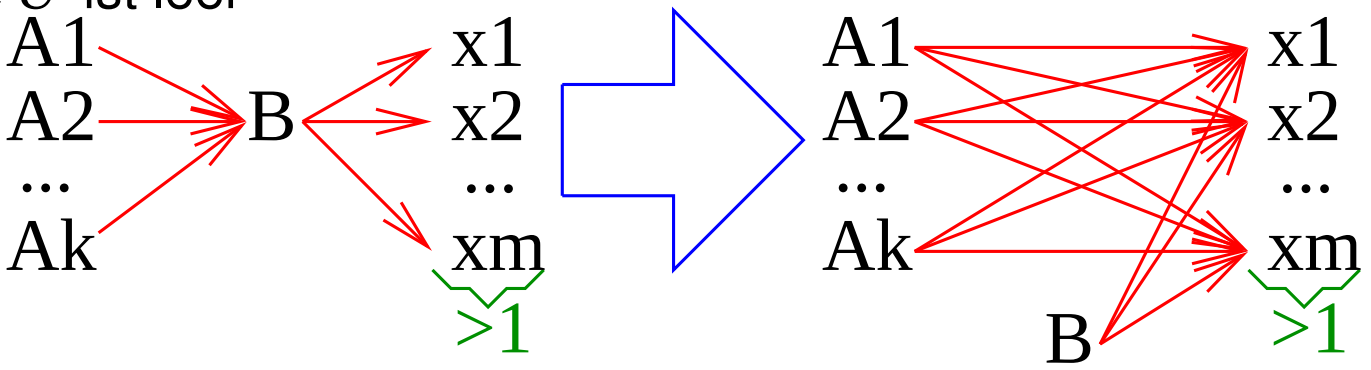
while $\exists X \to Y \in P \cap V \times V : P \cap \{Y\} \times V = \emptyset$ **do**

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 ist leer



Graphentheoretische Sichtweise: Verarbeitung in umgekehrter topologisch sortierter Reihenfolge

Chomsky Normalform

Eine Grammatik $G' = (V, \Sigma, P, S)$ ist in Chomsky Normalform falls $P \subseteq V \times \Sigma \cup V \times VV$.

Satz: Zu jeder kontextfreien Grammatik G mit $\varepsilon \notin L(G)$,
$\exists G'$ in Chomsky Normalform, so dass $L(G) = L(G')$.

Beweisansatz: schrittweise Veränderung der Grammatik G.

Invariante: $L(G)$ bleibt unverändert.

1. ε eliminieren
2. Einheitsproduktionen eliminieren

Eliminiation gemischter rechter Seiten

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

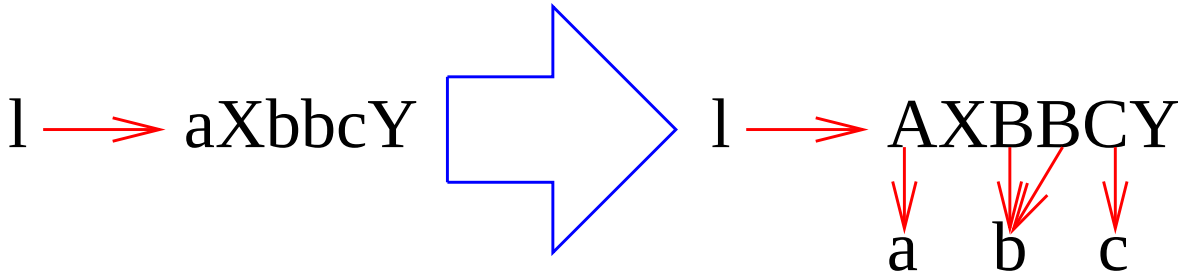
V_a:= new variable

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

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



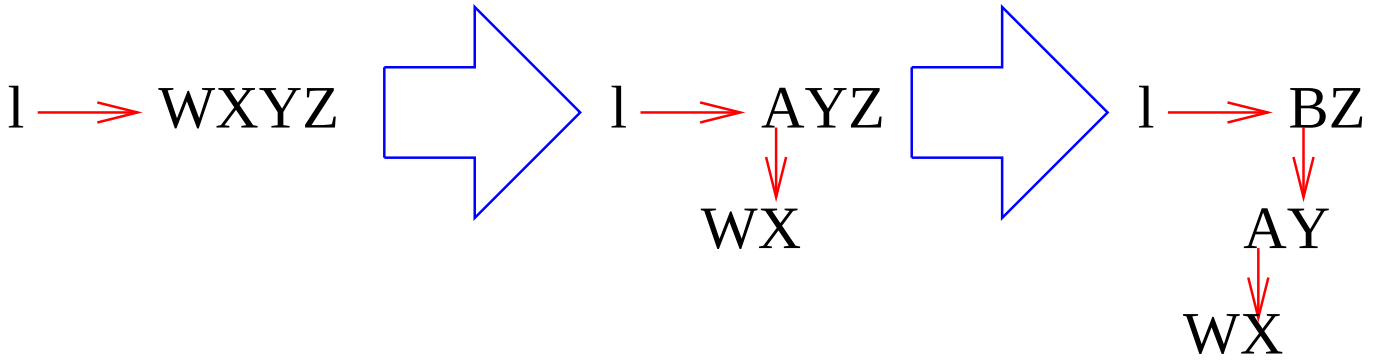
Elimination langer rechter Seiten

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

C:= new variable

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

Schleife terminiert, weil $\sum_{\ell \to r \in P} \max(0, |r| - 2)$ abnimmt.



Beispiel

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

$\rightsquigarrow$

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

$\rightsquigarrow$

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

Greibach Normalform

Eine Grammatik $G' = (V, \Sigma, P, S)$ ist in Greibach Normalform falls $P \subseteq V \times \Sigma V^*$.

Satz: Zu jeder kontextfreien Grammatik G mit $\varepsilon \notin L(G)$, $\exists G'$ in Greibach Normalform, so dass $L(G) = L(G')$.

Beweis: nicht hier

Idee: natürliche Verallgemeinerung von Typ 3 Grammatiken

1.3.2 Das Pumping Lemma

L kontextfrei

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

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

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

In Worten:

Hinreichend lange Worte einer kontextfreien Sprache lassen sich durch Wiederholung von ein oder zwei nichttrivialen Mittelteilen „aufpumpen“.

Beweis Pumping Lemma

Sei $G = (V, \Sigma, P, S)$ Grammatik in Chomsky Normalform für $L - \{\varepsilon\}$.

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

$z \in L$ mit $|z| = m \geq n$ beliebig.

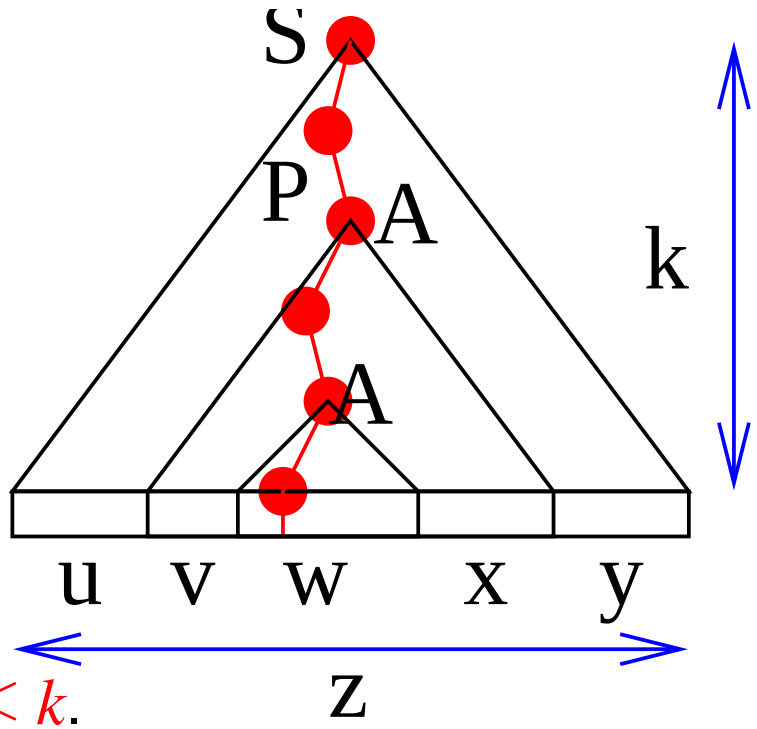
Betrachte einen Syntaxbaum für w.

Max. Grad ≤ 2, $\geq n = 2^k$ Blätter

$\rightarrow \exists$ Pfad P der Länge $|P| > k$.

$\rightarrow \geq k+1$ Variablen liegen auf P.

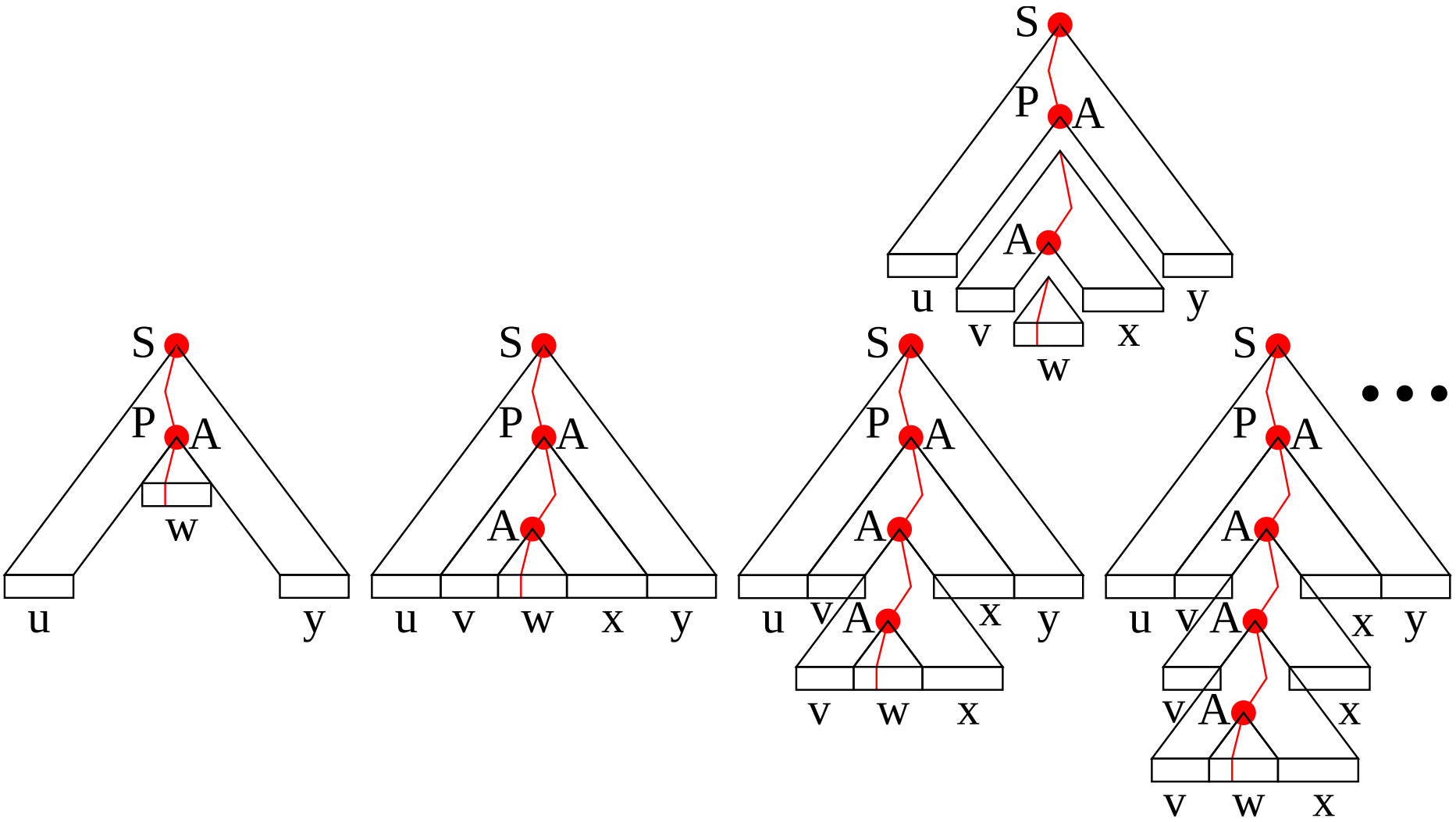
$\rightarrow \exists$ Variable $A \in P$: A kommt $\geq 2\times$ vor.



Zweites Vorkommen von unten hat Abstand $\leq k$.

$\rightarrow$ steht für Teilwort w der Länge $\leq n$

Konstruktion von Wiederholungen:



Lemma: Ein Binärbaum (Knotengrade 0 oder 2) mit $\geq 2^k$ Blättern enthält einen Pfad P der Länge $\geq k$.

Beweis: Induktion über k.

Fall $k = 0$: 1 Blatt, Pfadlänge 0.

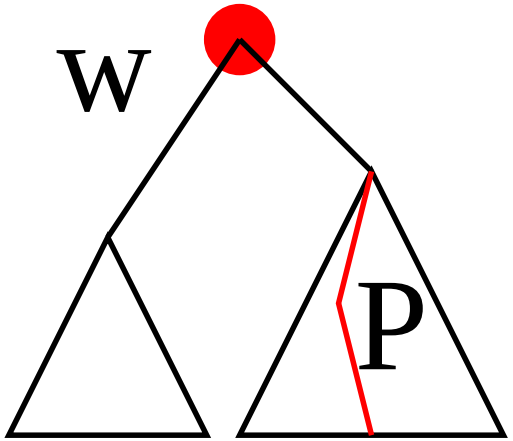
Fall $k \rightsquigarrow k+1$:

$\geq 2^{k+1}$ Blätter. Wurzel w hat Grad 2.

Mindestens ein Unterbaum hat $\geq 2^k$ Blätter

und enthält Pfad P der Länge $\geq k$.

Also hat wP Länge $\geq k+1$.



$L = \{a^m b^m c^m : m \geq 1\}$ ist nicht kontextfrei

Annahme L ist kontextfrei.

Sei n der Wert ab dem das Pumpinglemma gilt.

Betrachte $z = a^n b^n c^n$ und Zerlegung

$z = uvwxy$ nach dem Pumping Lemma mit

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

vx kann nicht a-s, b-s und c-s enthalten.

$\rightarrow$ die Balance von a-s, b-s und c-s in uwy stimmt nicht.

$\rightarrow uwy \notin L$

Widerspruch

Korollar: Typ2$\neq$Typ1

$L = \{ww : w \in \{a,b\}^*\}$ ist nicht kontextfrei

Annahme L ist kontextfrei.

Sei n der Wert ab dem das Pumpinglemma gilt.

Betrachte $z = a^n b^n a^n b^n$ und eine Zerlegung

$z = uvwxy$ nach dem Pumping Lemma mit $|vwx| \leq n$, $|vx| \geq 1$,

$z' := uwy \in L$.

Fall $vx = a^k b^j$ liegt in linker Hälfte:

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

Fall vx liegt in rechter Hälfte: analog

sonst (Mittellage): $vx = b^k a^j$

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

Faustregeln für Beweise mit dem Pumping Lemma

1. Sei n der Wert ab dem das Pumpinglemma gilt.
2. Betrachte $z = ???$ ($|z| \geq n$) und eine Zerlegung
   $z = uvwxy$ nach dem Pumping Lemma mit $|vwx| \leq n$, $|vx| \geq 1$
   - Jedes z mit $|z| \geq n$ ist erlaubt. Der "kreativste" Teil !
   - Auswahl soll Beweis möglich/einfach machen
   - Wegen $|vwx| \leq n$ vereinfachen Blöcke der Länge n die folgende Fallunterscheidung.
3. Fallunterscheidung über alle möglichen Zerlegungen $z = uvwxy$.
   Für jeden Fall: Finde ein $i \geq 0$, so dass $uv^iwx^iy \notin L(G)$.
   Typische Werte: $i = 0$, $i = 2$.
   Herausforderung: Anzahl Fälle klein halten.

1.3.3 Abschlusseigenschaften

Abgeschlossen unter

- ☐ Vereinigung $\cup$
- ☐ Produkt $\cdot$
- ☐ Stern $*$

Nicht abgeschlossen unter

- ☐ Schnitt $\cap$
- ☐ Komplement $\overline{\cdot}$

Abgeschlossenheit von KFG unter $\cup$

Betrachte

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

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

OBda mit $V_1 \cap V_2 = \emptyset$ sowie

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

Offenbar gilt

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

Abgeschlossenheit von KFG unter $\cdot$

Betrachte

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

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

OBda mit $V_1 \cap V_2 = \emptyset$ sowie

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

Offenbar gilt

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

Abgeschlossenheit von KFG unter *

Betrachte

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

wobei OBdA S_1 auf keiner rechten Seite in P vorkommt. Sowie

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

Offenbar gilt

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

Nichtabgeschlossenheit von KFG unter ∩

Betrachte die kontextfreien Sprachen

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

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

$L_1 \cap L_2 = \{a^i b^i c^i : i > 0\}$ ist nicht kontextfrei!

Nichtabgeschlossenheit von KFG unter $\bar{\cdot}$

Annahme:

Abgeschlossenheit unter $\cup$ und $\bar{\cdot}$.

$\rightarrow$

Abgeschlossenheit unter $\cap$ wegen

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

Widerspruch

1.3.4 Der CYK-Algorithmus — Das Wortproblem für kontextfreie Grammatiken

Gegeben: Grammatik $G = (V, \Sigma, P, S)$,
Wort $x = x_1 \cdots x_n \in \Sigma^*$.

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

Algorithmus von Cocke, Younger und Kasami

OBdA: G ist in Chomsky Normalform:

Spezialfall $x = \varepsilon$ ist einfach,

sonst zunächst in Chomsky NF konvertieren.

CYK Algorithmus

Wir lösen ein allgemeineres Problem:

Für jedes Teilwort $x_i \cdots x_{i+j-1}$ der Länge j von x,

von welchen Variablensymbolen ist $x_i \cdots x_{i+j-1}$ ableitbar?

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

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

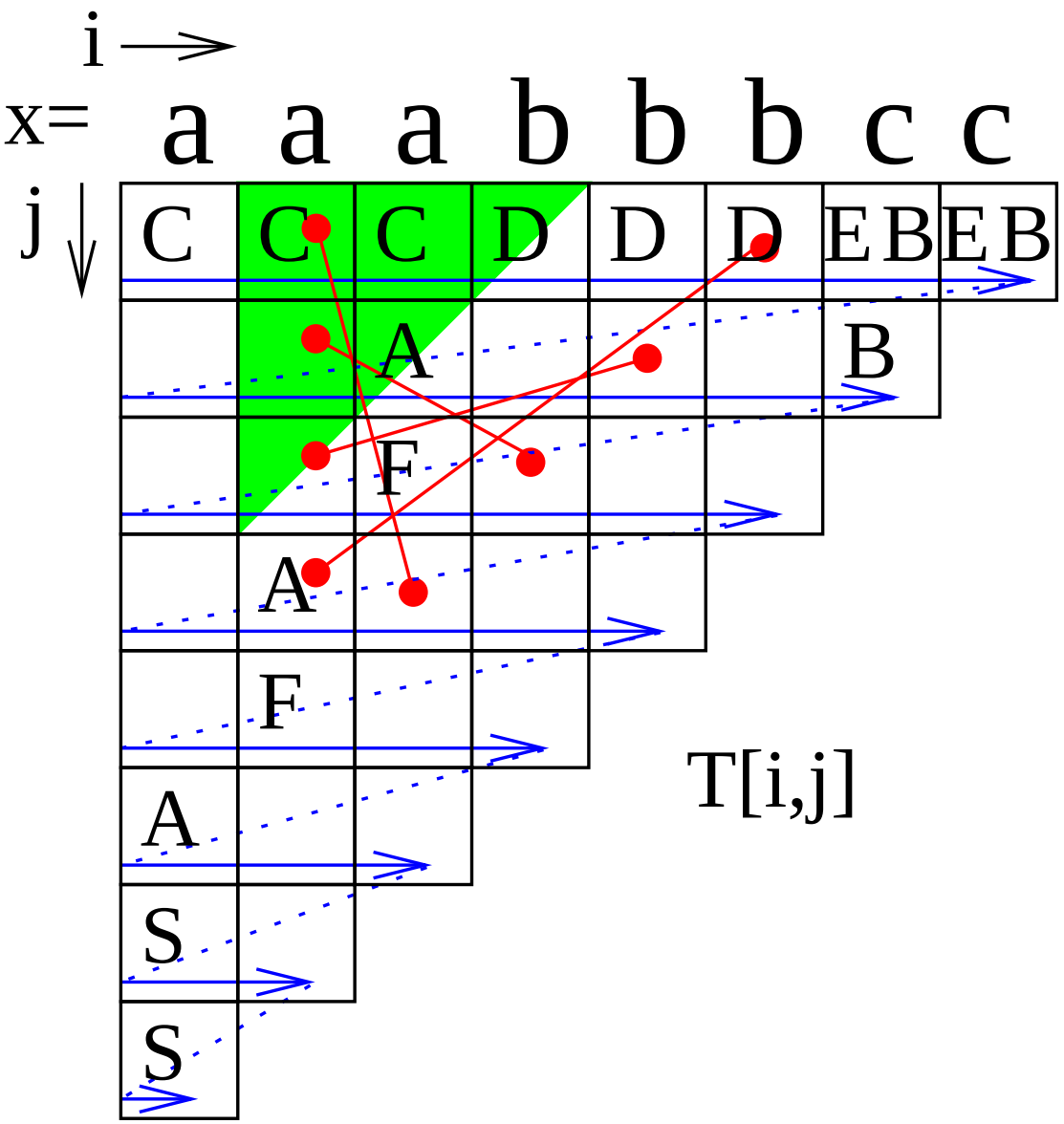
Sonst:

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

Am Ende: $S \in T[1,n]$?

Beispiel

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



Implementierung durch dynamische Programmierung

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**
            $T[i,j] \cup = \{A : \exists A \to BC \in P : B \in T[i,k] \wedge C \in T[i+k, j-k]\}$
return $S \in T[1,n]$

Analyse

OBdA $V = 1..|V|$

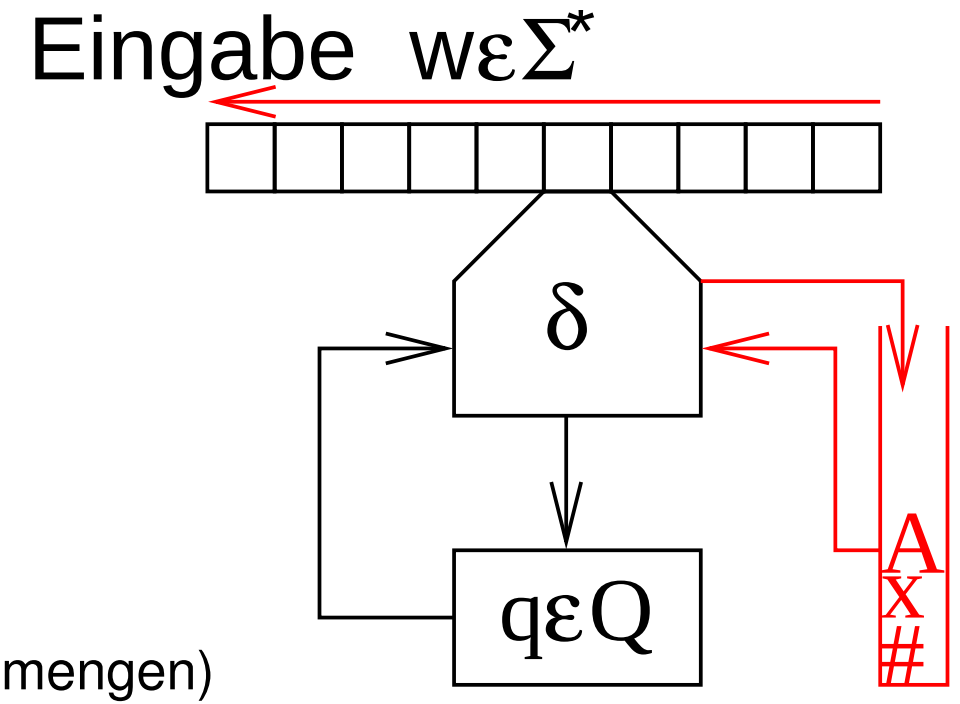
for $i := 1$ **to** n **do** $T[i,1] := \{A \in V : A \to x_i \in P\}$ // "billig"
for $j := 2$ **to** n **do** // $\leq n$ mal
  for $i := 1$ **to** $n - j + 1$ **do** // $\leq n$ mal
    $T[i,j] := \emptyset$ // Bitvektor der Größe $|V| \leq |P|$
    for $k := 1$ **to** $j - 1$ **do** // $\leq n$ mal
      foreach $A \to BC \in P$ **do** // $\leq |P|$ mal
        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 $S \in T[1,n]$

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

1.3.5 Kellerautomaten

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

- Q, Zustände
- Σ, Eingabealphabet
- Γ Kelleralphabet,
  $\Sigma \cup \{\#\} \subseteq \Gamma$
- $\delta : Q \times (\Sigma \cup \varepsilon) \times \Gamma \to 2^{Q \times \Gamma^*}$,
  Übergangsfunktion; (endliche Teilmengen)
- $s \in Q$, Startzustand
- $\# \notin \Sigma$: Kellerende,

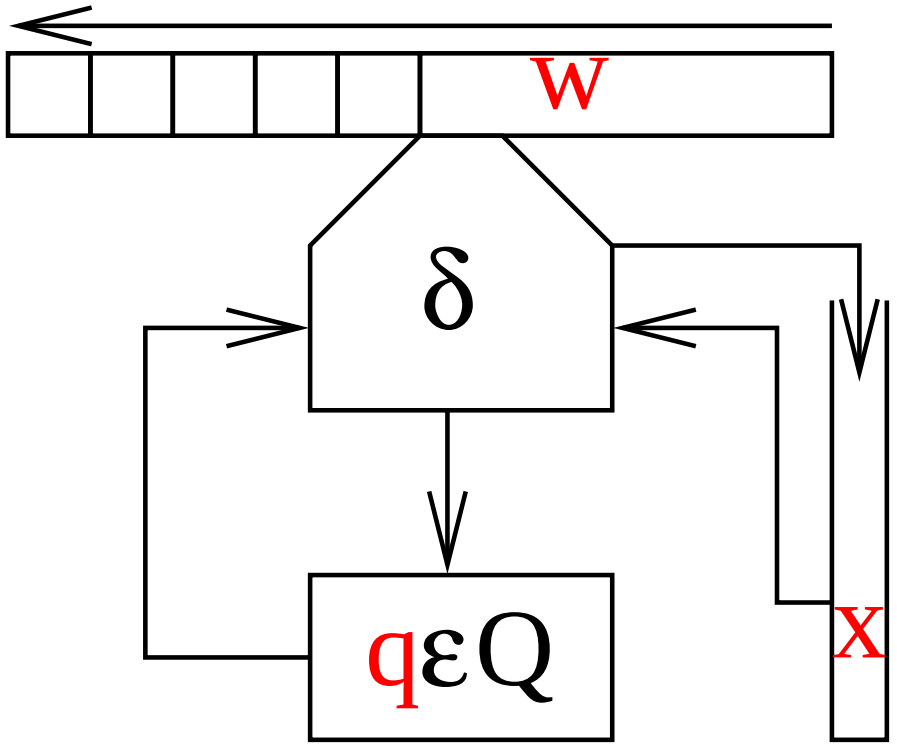


$\approx \varepsilon$NFA $+$ Kellerspeicher $-$ Endzustände

$\approx$ spezielle 2-Band-Turingmaschine

Konfiguration einer Kellermaschine

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



Funktionsweise einer Kellermaschine

mögliche Übergänge zwischen Konfigurationen.

Verbrauch von Eingabezeichen a:

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

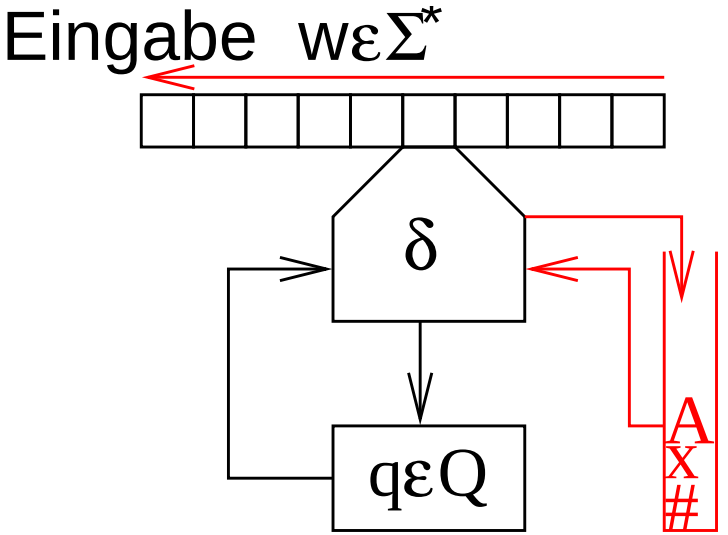
ε-Übergang:

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

Kellermaschine als Akzeptor

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

$L(K)$?



Definition:

K akzeptiert $w \in \Sigma^*$ gdw

$\exists$ Folge von (durch δ zugelassenen)

Konfigurationsübergangen

$(s, w, \#) \vdash \cdots \vdash (q, \varepsilon, \varepsilon)$ mit $q \in Q$ beliebig.

“Akzeption über den leeren Keller”

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

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

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

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

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

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

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

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

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

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

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

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

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

$(1,\varepsilon,\varepsilon)$

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

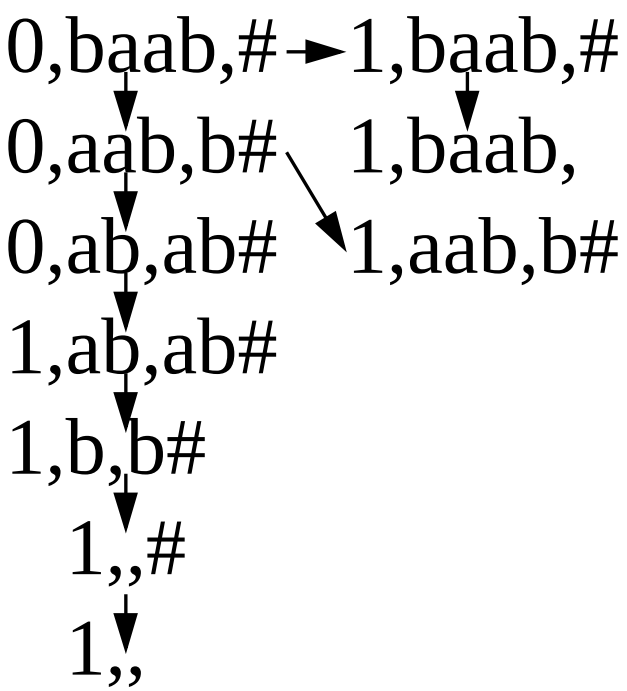
$K = (\{0,1\}, \{a,b\}, \{a,b,\#\}, \delta, 0, \#)$

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

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

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

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



Satz: L ist kontextfrei gdw. $\exists$ NKellerA $M : L(M) = L$

Beweis: L ist kontextfrei $\longrightarrow \exists$ NKellerA $M : L(M) = L$

Sei $G = (V, \Sigma, P, S)$ Grammatik mit $L(G) = L$.

Betrachte NKellerA $M = (\{z\}, \Sigma, V \cup \Sigma, \delta, z, S)$ mit

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

Idee (Invariante): Keller speichert Satzform einer Ableitung minus mit der Eingabe gematchte Terminalzeichen.

$G = (V, \Sigma, P, S)$ Grammatik mit $L(G) = L$.

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

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

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

$\longrightarrow \exists$ Linksableitung $S \Rightarrow \cdots \Rightarrow x$

Wir konstruieren daraus eine Folge von Konfigurationen von M

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

- Konfiguration $(z, x_i \cdots x_n, Vy)$:
  Linksableitungsersetzung $V \to \alpha \in P$ ergibt $(z, \alpha) \in \delta(z, \varepsilon, V)$.
  Also neue Konfiguration $(z, x_i \cdots x_n, \alpha y)$.
- Konfiguration $(z, x_i \cdots x_n, x_i y)$:
  Neue Konfiguration $(z, x_{i+1} \cdots x_n, y)$

$G = (V, \Sigma, P, S)$ Grammatik mit $L(G) = L$.

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

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

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

$\longrightarrow \exists$ Folge von Konfigurationen von M

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

Wir konstruieren daraus eine Linksableitung $S \Rightarrow \cdots \Rightarrow x$.

Die Konfigurationsübergänge ohne Verbrauch von Eingabezeichen ergeben die anzuwendenden Ersetzungen.

Beweis: $\exists$ NKellerA $M : L(M) = L \longrightarrow L$ ist kontextfrei.

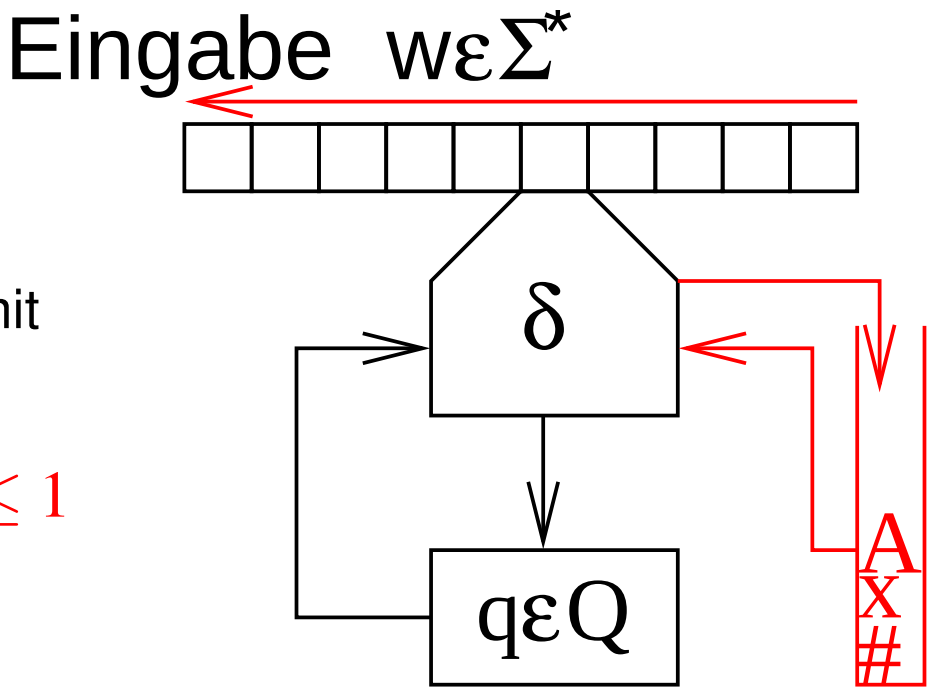
Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$ NKeller A mit $L(M) = L$.

nicht hier. (Etwas technisch)

1.3.6 Deterministisch kontextfreie Sprachen

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

- Q, Σ, Γ, s, # wie gehabt.
- $\delta : Q \times (\Sigma \cup \varepsilon) \times \Gamma \to 2^{Q \times \Gamma^*}$, mit
  $\forall z \in Q, a \in \Sigma, A \in \Gamma :$
  $|\delta(z,a,A)| + |\delta(z,\varepsilon,A)| \leq 1$



K akzeptiert $w \in \Sigma^*$ gdw

$\exists$ Folge von (durch δ vorgeschriebenen) Konfigurationsübergängen

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

Satz:

$\forall$ DKellerA K : $\exists$ 1ZustandNKellerA K' : $L(K) = L(K')$.

$\exists$ 1ZustandNKellerA K' : $\not\exists$ DKellerA K : $L(K) = L(K')$.

Beweis:

nicht hier.

Compiler

Beobachtung:

Das Wortproblem für DKellerA läßt sich in linearer Zeit lösen.

Bemerkung:

Sprachen für DKellerA sind gerade die LR(k)-Sprachen.

Solche und ähnliche Sprachfamilien sind Grundlage für die Syntax vieler Programmiersprachen.

Abgeschlossenheitseigenschaften für DKellerA

Ohne Beweis:

Abgeschlossen unter: $\bar{\cdot}$, Schnitt mit regulärer Sprache.

Nicht abgeschlossen unter: $\cup, \cap, \cdot, ^*$

1.3.7 Entscheidbarkeit für kontextfreie Sprachen

Leerheitsproblem

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

Marked:= Σ

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

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

return $S \notin$Marked

Ergibt wahr gdw $L(G) = \emptyset$

Endlichkeitsproblem

Gegeben: Grammatik $G = (V, \Sigma, P, S)$

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

Sei n die Mindestlänge aus dem Pumpinglemma.

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

Beweis:

$z \in L(G), n \leq |z| < 2n \longrightarrow$ Pumpinglemma garantiert $|L| = \infty$.

Falls $|L(G)| = \infty$ betrachte $z \in L(G)$ mit minimalem $|z| \geq n$.

Annahme $|z| \geq 2n$.

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

Widerspruch zur Minimalität von $|z|$.

Endlichkeitsproblem

Gegeben: Grammatik $G = (V, \Sigma, P, S)$

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

Sei n die Mindestlänge aus dem Pumpinglemma.

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

Brute Force Algorithmus:

Wortproblem für alle Wörter z der Länge $n \leq |z| < 2n$ lösen.

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

!

Bemerkung: Es gibt effizientere Algorithmen.

Entscheidbare Probleme für DKellerA

- Äquivalenz: $L(K_1) = L(K_2)$?

Unentscheidbare Problem für KFG

- $L(G_1) \cap L(G_2) = \emptyset$? Disjunktheit
- $|L(G_1) \cap L(G_2)| = \infty$?
- $L(G_1) \cap L(G_2)$ kontextfrei?
- $L(G_1) \subseteq L(G_2)$?
- $L(G_1) = L(G_2)$?
- Mehrdeutigkeit: $\exists x \in L(G) : |\{\text{Syntaxbäume}(x)\}| \geq 2$
- Ist $\overline{L(G)}$ kontextfrei?
- Ist $L(G)$ regulär?
- Ist $L(G)$ det. kontextfrei?

Unentscheidbarkeit von $L(G_1) \cap L(G_2) = \emptyset$?

Wir lösen das PCP $K = (x_1, y_1) \cdots (x_k, y_k) \in (\{a,b\}^* \times \{a,b\}^*)^*$

mit Hilfe eines angenommenen Unterprogramms disjoint(G_1, G_2):

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

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

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

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

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

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

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

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

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

K hat Lösung.