Informatik III

2.2 Grammatiken und Kontextfreie Sprachen

Christian Schindelhauer

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Wintersemester 2007/08

Formale Sprachen und

Endliche Automaten

Wortersetzung und Grammatik

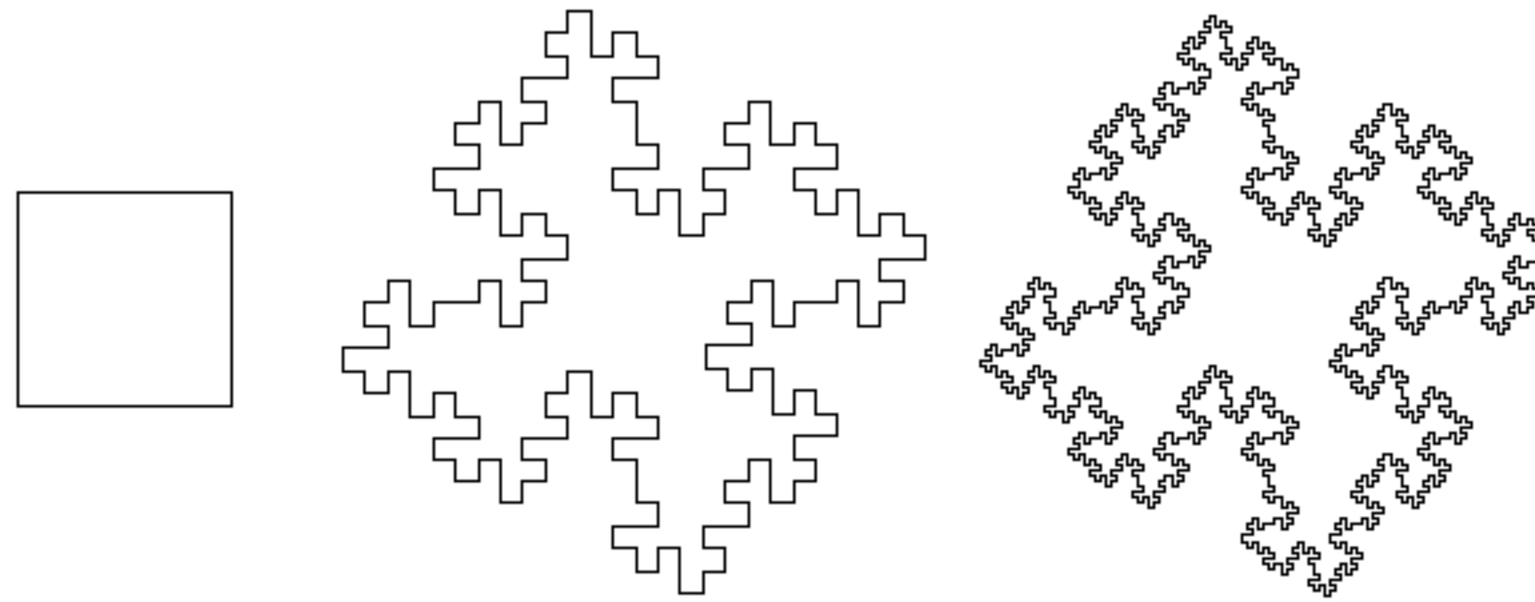
Lindenmayer-System

- **Start**
  - A
- **Regeln:**
  - A → B
  - B → AB
- **Was passiert, wenn man die Regeln simultan anwendet?**
- **So ein Wortersetzungssystem wird Lindenmayer-System genannt**

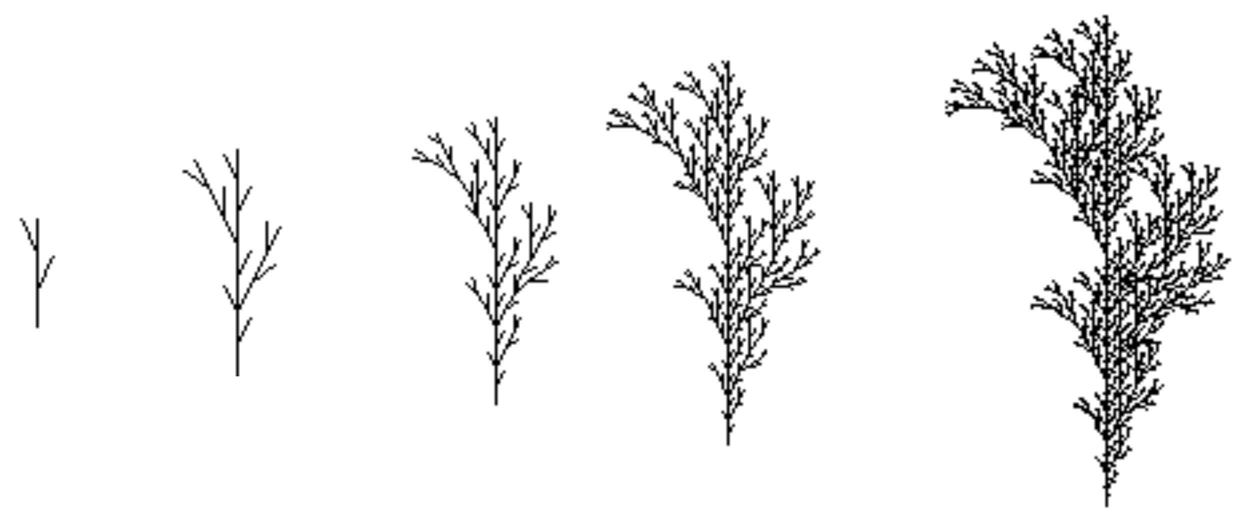
- **A ⇒ B**
  ⇒ AB
  ⇒ BAB
  ⇒ ABBAB
  ⇒ BABABBAB
  ⇒ ABBABBABABBAB
  ⇒ ...

Lindenmayer-System Beispiel

- **Schildkrötenalgorithmus**
  - F: Gehe einen schritt vorwärts
  - +: 90°-Drehung nach rechts
  - -:90°-Drehung nach links
- **Start:**
  - F+F+F+F
- **Regeln**
  - p: F -> F+F-F-FF+F+F-



- **Anderes Beispiel:**



- **http://www.biologie.uni-hamburg.de/b-online/e28_3/lsys.html**

Chomsky-Grammatik

- **Beispiel:**
  - Start
    - A
  - Regeln:
    - $A \rightarrow B$
    - $B \rightarrow AB$
- **Was passiert, wenn man die Regeln *nicht simultan* anwendet?**
- **So eine Grammatik wird Chomsky-Grammatik genannt**

- **Ableitungen:**
  - $A \Rightarrow AB$
    $\Rightarrow ABB$
    $\Rightarrow ABBB$
    $\Rightarrow ...$
  - oder
    $A \Rightarrow AAB$
    $\Rightarrow AAAB$
    $\Rightarrow AAAAB$
    $\Rightarrow ...$

Wortproblem

- **Chomsky-Grammatik**
  - Start
    - A
  - Regeln:
    - A → B
    - B → AB
- **Wortproblem**
  - Kann man das Wort ABBABB ableiten?

Formale Sprachen und

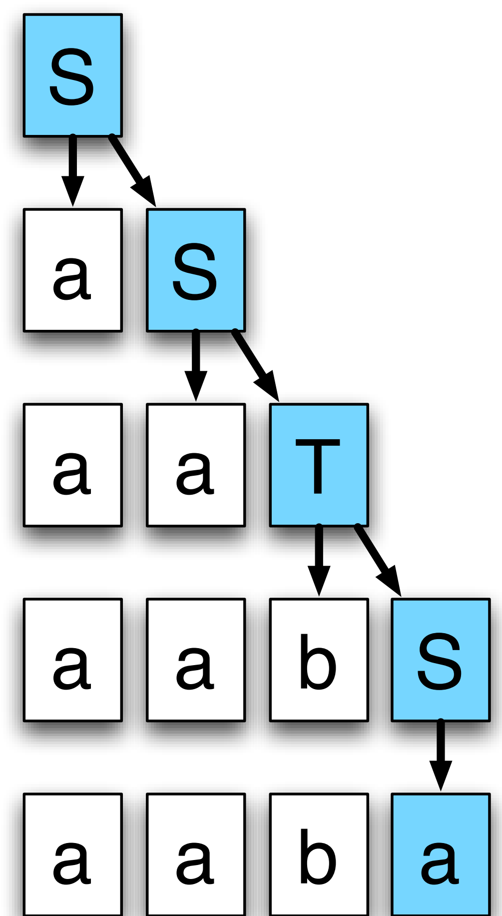
Endliche Automaten

Reguläre Grammatik

7

Reguläre Grammatik

$$S \rightarrow aS$$
$$S \rightarrow a$$
$$S \rightarrow aT$$
$$T \rightarrow bS$$
$$T \rightarrow bT$$



Reguläre Grammatik

- **Definition**
  - Eine (rechts)-**reguläre Grammatik** ist ein Vierer-Tupel $G = (V,\Sigma,R,S)$ mit
    - V ist die endliche Menge der **Variablen** (Nichtterminale)
    - Σ ist die endliche Menge der **Terminale** (Terminalsymbole)
    - R ist eine endliche Menge an **Ersetzungsregeln** (Regeln/ Produktionen)
      - Jede Regel bildet eine Variable auf ein Terminal $A \to a$, mit $A \in V$ und $a \in \Sigma$
      - oder auf ein Wort aus einem Terminal und einer Variable ab $A \to aB$ mit $A,B \in V$ und $a \in \Sigma$
      - oder die Startvariable wird auf das leere Wort abgebildet $S \to \varepsilon$
    - $S \in V$ ist die **Startvariable**

- **Beispiel**
  - $(\{S,T\},$
    $\{a,b\},$
    $\{S \to aS,\ S \to a,$
    $S \to aT,\ T \to bS,$
    $T \to bT\},$
    $S\}$

$$
\begin{aligned}
S &\to aS \\
S &\to a \\
S &\to aT \\
T &\to bS \\
T &\to bT
\end{aligned}
$$

Ableitung

- **Ableitung**
  - Falls die Regel $A \rightarrow w$ in R ist, dann ist $\mathbf{uAv} \Rightarrow \mathbf{uwv}$,
    - d.h. uAv kann zu uwv in einem Schritt **abgeleitet** werden
  - Wir sagen das u zu v abgeleitet werden kann oder $\mathbf{u} \Rightarrow^* \mathbf{v}$, wenn es ein $k \geq 0$ gibt

    mit
    - $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v$
- **Sprache der Grammatik G**
  - $L(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* v\}$

- **Beispiel**
  - $(\{S,T\},$
    $\{a,b\},$
    $\{S \rightarrow aS, S \rightarrow a,$
    $S \rightarrow aT, \ T \rightarrow bS,$
    $T \rightarrow bT\},$
    $S\}$
- $\mathbf{S} \Rightarrow \mathbf{aS}$

  $\Rightarrow \mathbf{aaT}$

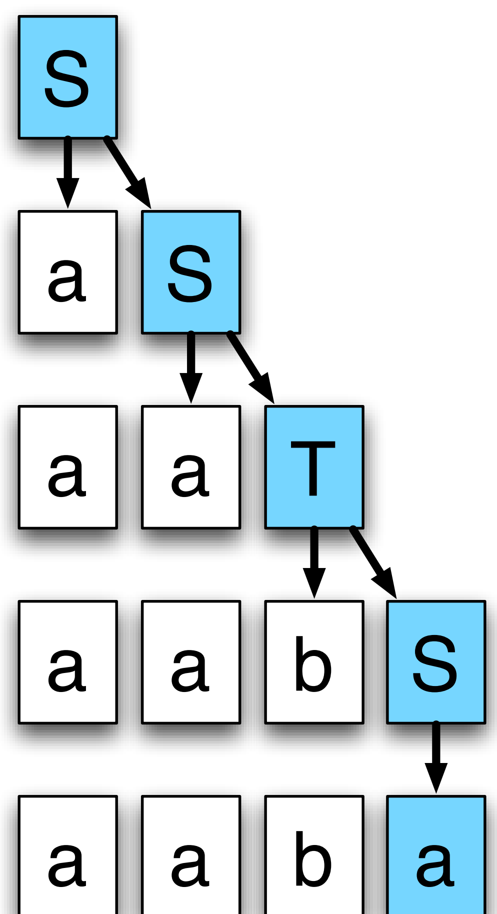
  $\Rightarrow \mathbf{aabS}$

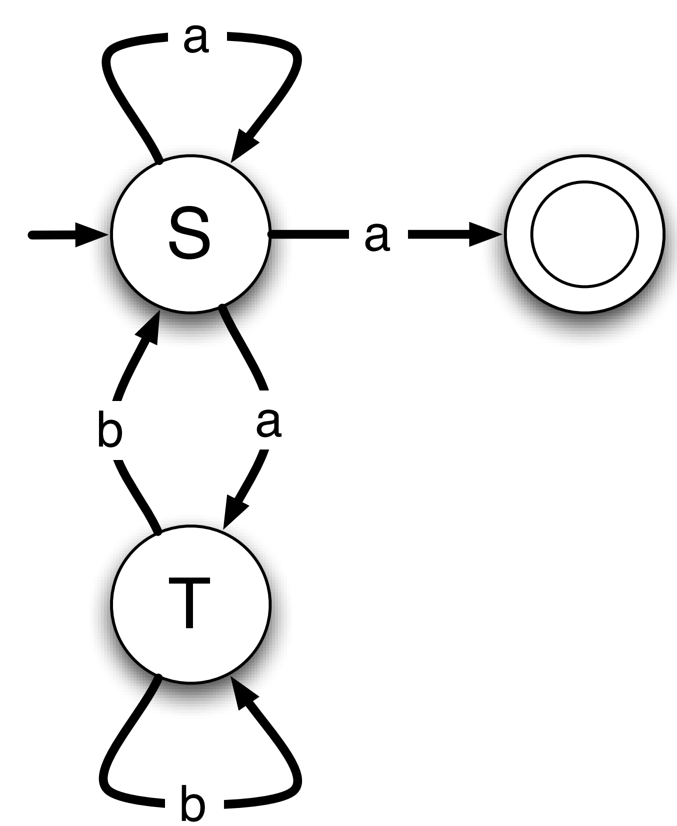
  $\Rightarrow \mathbf{aaba}$

Reguläre Grammatiken beschreiben REG

- **Theorem**
  - Die Menge der regulären Sprachen werden durch die rechts-regulären Grammatiken beschrieben.

$S \rightarrow aS$
$S \rightarrow a$
$S \rightarrow aT$
$T \rightarrow bS$
$T \rightarrow bT$

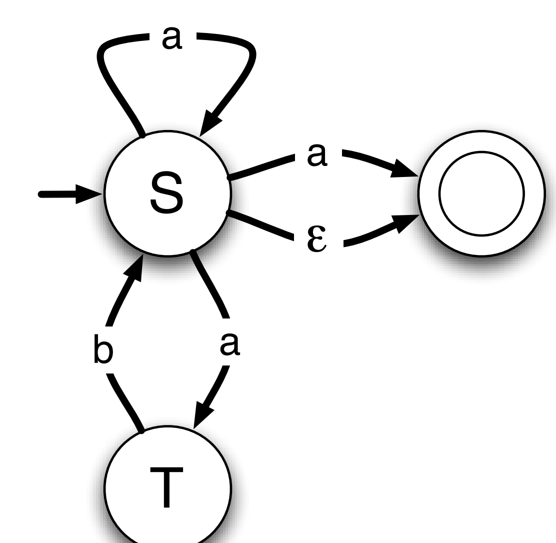




Reguläre Grammatiken beschreiben REG

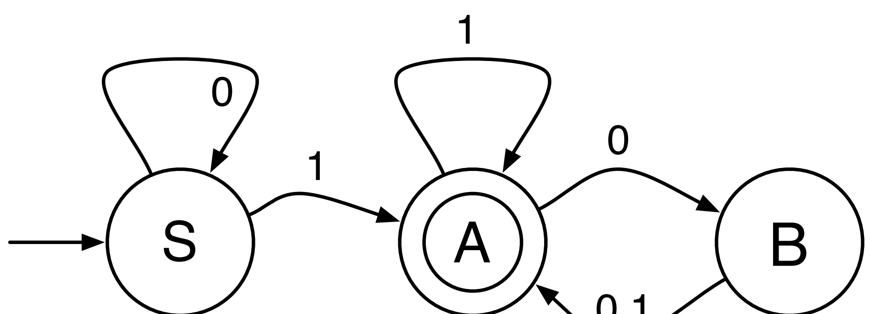
- **Theorem**
  - Rechts-regulären Grammatiken erzeugen reguläre Sprachen.
- **Beweis**
  - Konstruiere NFA $(Q, \Sigma, \delta, q_0, F)$ für Grammatik $G = (V,\Sigma,R,S)$
  - Zustandsmenge: $Q = V \cup \{Q_{akz}\}$
    - Startzustand: S
    - Akzeptierende Zustandsmenge: $\{Q_{akz}\}$
    - Für jede Regel $A \rightarrow aB$ füge B zur Menge von $\delta(A,a)$ hinzugefügt, so dass
      - $B \in \delta(A,a) \Leftrightarrow (A \rightarrow aB) \in R$
    - Bei Regel $A \rightarrow a$ wird Q_{akz} zur Menge von $\delta(A,a)$ hinzugefügt, so dass
      - $Q_{akz} \in \delta(A,a) \Leftrightarrow (A \rightarrow a) \in R$
    - Bei Regel $S \rightarrow \varepsilon$ wird Q_{akz} zur Menge von $\delta(S, \varepsilon)$ hinzugefügt, so dass
      - $Q_{akz} \in \delta(S, \varepsilon) \Leftrightarrow (S \rightarrow \varepsilon) \in R$

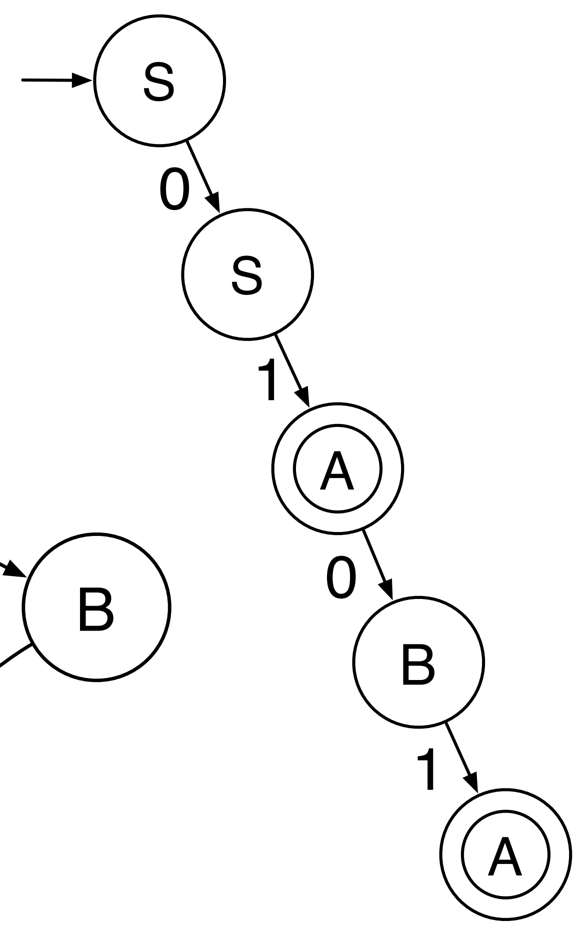
$S \rightarrow \epsilon$
$S \rightarrow aS$
$S \rightarrow a$
$S \rightarrow aT$
$T \rightarrow bS$

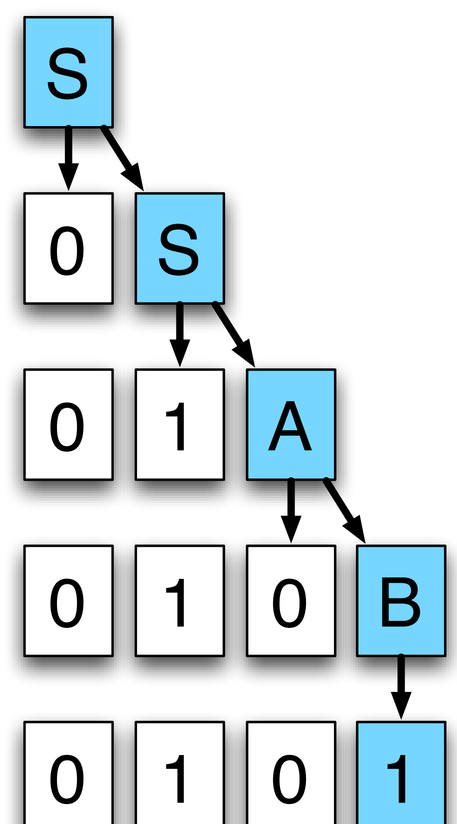


Reguläre Grammatiken beschreiben REG

- **Theorem**
  - Jede reguläre Sprachen kann durch eine rechts-reguläre Grammatik erzeugt werden.







$S \rightarrow 0S$

$S \rightarrow 1A$ $S \rightarrow 1$

$A \rightarrow 1A$ $A \rightarrow 1$

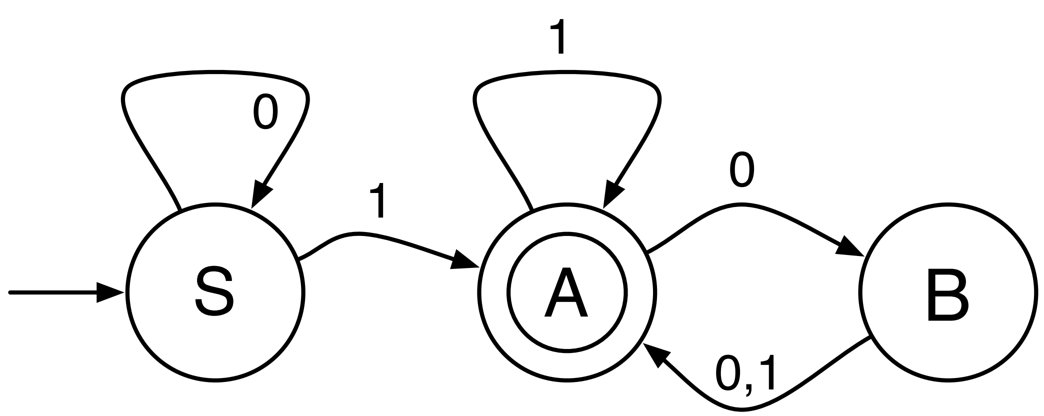
$A \rightarrow 0B$

$B \rightarrow 0A$ $B \rightarrow 0$

$B \rightarrow 1A$ $B \rightarrow 1$

Reguläre Grammatiken beschreiben REG

- **Theorem**
  - Jede reguläre Sprachen kann durch ein rechts-regulären Grammatik erzeugt werden.
- **Beweis**
  - Deterministische endliche Automaten können durch rechts-reguläre Grammatiken beschrieben werden
    - Variablen: $V = Q$
    - Startsymbol ist der Startzustand
    - Terminale sind das Alphabet der Sprache
    - Falls $\delta(A,a) = B$ füge Regel $A \rightarrow aB$ hinzu
      - $\delta(A,a) = B \Leftrightarrow (A \rightarrow aB) \in R$
    - Falls $\delta(A,a) = B$ und B ist akzeptierender Zustand füge Regel $A \rightarrow a$ hinzu
      - $\delta(A,a) = B$ und B ist akzeptierend $\Leftrightarrow (A \rightarrow a) \in R$



$S \rightarrow 0S$
$S \rightarrow 1A$ $\quad S \rightarrow 1$
$A \rightarrow 1A$ $\quad A \rightarrow 1$
$A \rightarrow 0B$
$B \rightarrow 0A$ $\quad B \rightarrow 0$
$B \rightarrow 1A$ $\quad B \rightarrow 1$

Reguläre Grammatiken beschreiben REG

- **Theorem**
  - Die Menge der regulären Sprachen werden durch die rechts-regulären Grammatiken beschrieben
- **Analog:**
  - Definition der links-regulären Grammatik
  - Die linksregulären Grammatiken beschreiben ebenfalls die neuen regulären Sprachen.

$S \to S0$	
$S \to A1$	$S \to 1$
$A \to A1$	$A \to 1$
$A \to B0$	
$B \to A0$	$B \to 0$
$B \to A1$	$B \to 1$

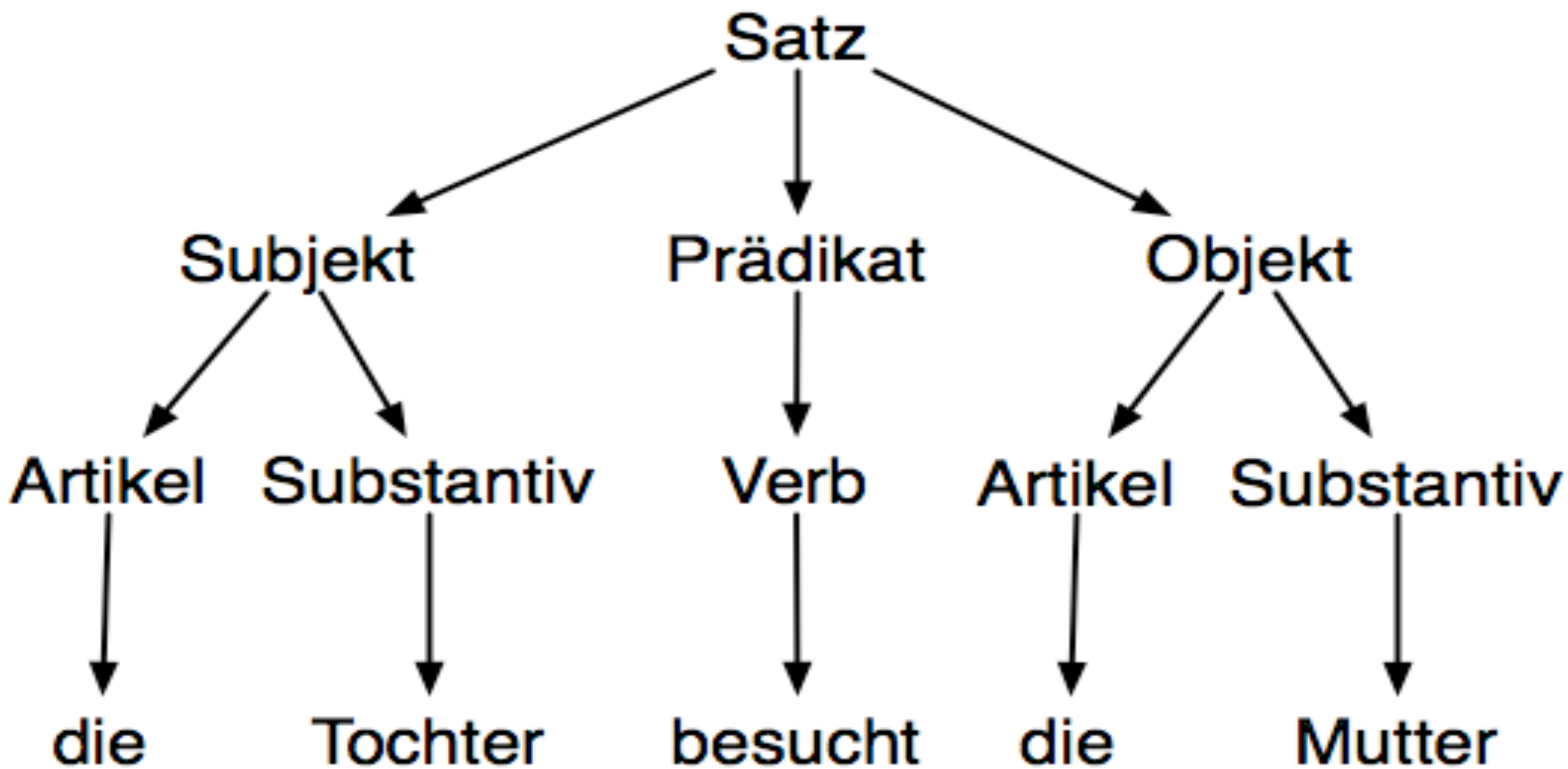
$S \to 0S$	
$S \to 1A$	$S \to 1$
$A \to 1A$	$A \to 1$
$A \to 0B$	
$B \to 0A$	$B \to 0$
$B \to 1A$	$B \to 1$

Formale Sprachen und
Endliche Automaten

Kontextfreie Grammatik

16

Grammatik - Was ist das?



Grammatik - Was ist das?

<SATZ> → <SUBJEKT> <PRÄDIKAT> <OBJEKT>

<SUBJEKT> → <ARTIKEL> <SUBSTANTIV>

<OBJEKT> → <ARTIKEL> <SUBSTANTIV>

<PRÄDIKAT> → <VERB>

<ARTIKEL> → `die`

<SUBSTANTIV> → `Mutter`

<SUBSTANTIV> → `Tochter`

<SUBSTANTIV> → `Professorin`

<SUBSTANTIV> → `Studentin`

<VERB> → `langweilt`

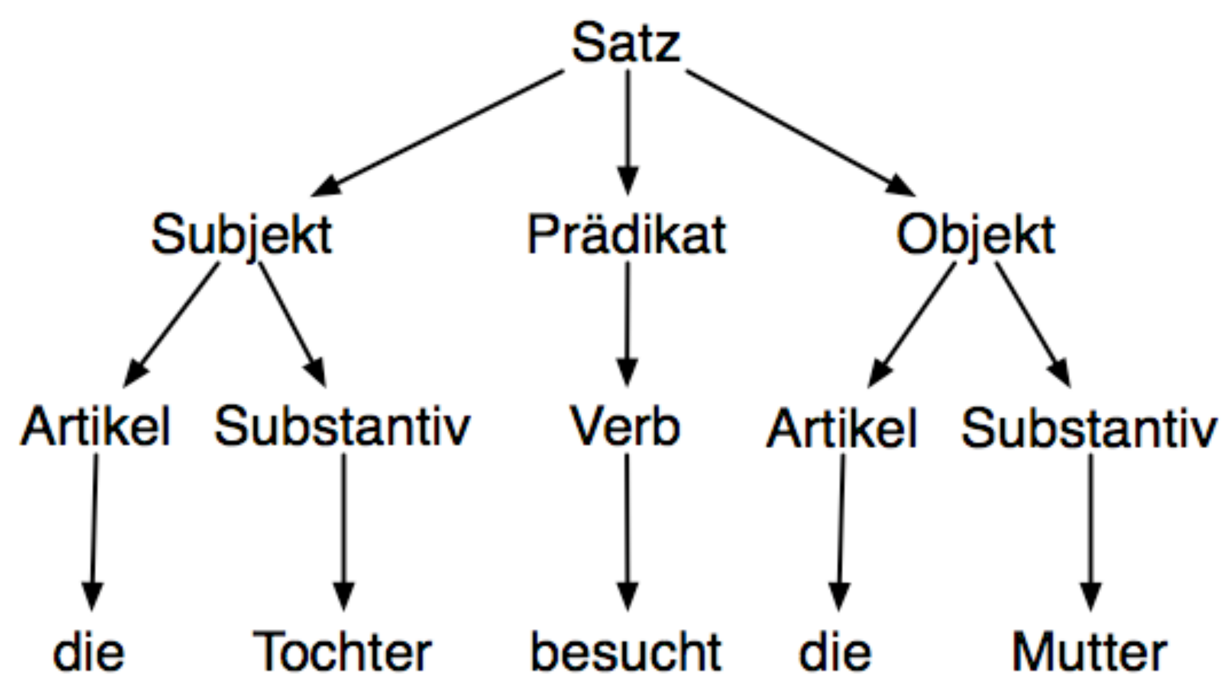
<VERB> → `prüft`

<VERB> → `belehrt`

<VERB> → `besucht`

<VERB> → `beleidigt`

<VERB> → `tötet`



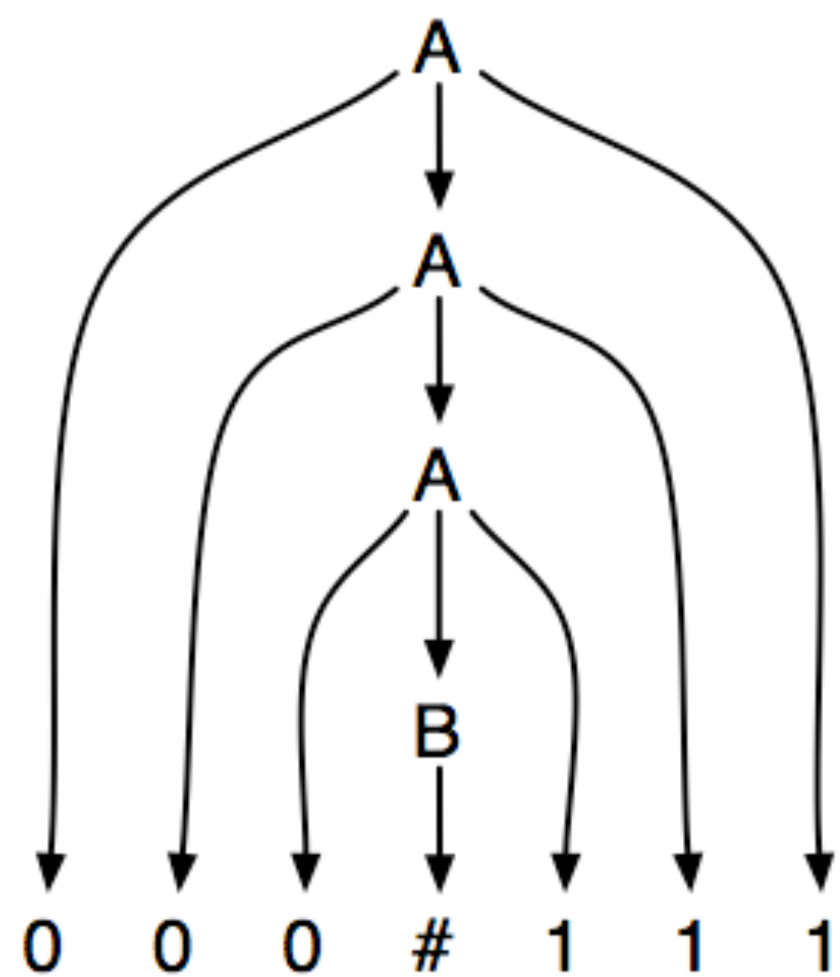
Syntax und Semantik

- **Grammatik:**
  - <SATZ> → <SUBJEKT> <PRÄDIKAT> <OBJEKT>
  - <SUBJEKT> → <ARTIKEL> <SUBSTANTIV>
  - <OBJEKT> → <ARTIKEL> <SUBSTANTIV>
  - <PRÄDIKAT> → <VERB>
  - <ARTIKEL> → die
  - <SUBSTANTIV> → Mutter | Tochter
  - <SUBSTANTIV> → Professorin | Studentin
  - <VERB> → langweilt | prüft | belehrt
  - <VERB> → besucht | beleidigt
  - <VERB> → tötet

- **Mögliche Ableitungen:**
  - die Studentin besucht die Professorin
  - die Professorin belehrt die Studentin
  - die Professorin langweilt die Studentin
  - die Studentin beleidigt die Professorin
  - die Professorin prüft die Studentin
  - die Tochter besucht die Mutter
  - die Mutter besucht die Professorin
  - die Professorin belehrt die Mutter
  - die Mutter tötet die Professorin

Beispiele kontextfreier Grammatiken

- **Kontextfreie Grammatiken sind "Wortersetzer"**
  - A → 0A1
  - A → B
  - B → #
- **Ersetzungsregeln bestehen aus Produktionen**
- **Variablen, Terminalsymbole (Terminale)**
- **Startvariable: A**
  - A
  - 0A1
  - 00A11
  - 000A111
  - 000B111
  - 000#111 .... und Schluss
- **Das Wort 000#111 ist ein Wort der Grammatik**



Ableitungsbaum

Formale Definition einer kontextfreien Grammatik

- **Definition**
  - Eine kontextfreie Grammatik ist ein Vierer-Tupel $G=(V,\Sigma,R,S)$, wobei
    - V ist die endliche Menge der **Variablen** (Nichtterminale)
    - Σ ist die endliche Menge der **Terminale** (Terminalsymbole)
      - V und Σ sind disjunkte Mengen
    - R ist eine endliche Menge an **Ersetzungsregeln** (Regeln/ Produktionen)
      - Jede Regel besteht aus einer Variable und einer Zeichenkette aus Variablen und Terminalen,
        - $A \rightarrow w$, mit $A \in V$, $w \in (V \cup \Sigma)^*$
    - $S \in V$ ist die **Startvariable**
- **Ableitung**
  - Falls die Regel $A \rightarrow w$ in R ist, dann ist $uAv \Rightarrow uwv$, d.h.
    - uAv kann zu uwv in einem Schritt abgeleitet werden
  - Wir sagen das u zu v abgeleitet werden kann oder
    $u \Rightarrow^* v$, wenn es ein $k \geq 0$ gibt mit
    - $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v$
- **Sprache der Grammatik G**
  - $L(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* w\}$
- **Die kontextfreien Sprachen werden durch kontextfreien Grammatiken erzeugt.**

Beispiel einer kontextfreien Grammatik

- **Definition**
  - Eine kontextfreie Grammatik ist ein Vierer-Tupel $G=(V,\Sigma,R,S)$
    - V: Variablen
    - Σ: Terminale
      - V und Σ sind disjunkt
    - R : Ersetzungsregeln
      - $A \to w$ mit $A \in V$, $w \in (V \cup \Sigma)^*$
    - $S \in V$: Startvariable
- **Ableitung**
  - Falls $A \to w$ in R,
    dann ist $uAv \Rightarrow uwv$
  - $u \Rightarrow^* v$, wenn es ein $k \geq 0$ gibt mit
    - $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v$

- **Sprache der Grammatik G**
  - $L(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* v\}$
  - $G = (\{A,B\}, \{0,1,\#\}, R, A)$
    - $R = \{A \to 0A1, A \to B, B \to \#\}$
  - Alternativ: $R = \{A \to 0A1 \mid B, B \to \#\}$
  - $A \Rightarrow 0A1$
    $\Rightarrow 00A11$
    $\Rightarrow 000A111$
    $\Rightarrow 000B111$
    $\Rightarrow 000\#111$
  - Also: $A \Rightarrow^* 000\#111$
  - Das Wort 000#111 in der Sprache L(G)
  - $L(G) = \{\#, 0\#1, 00\#11, 000\#111, \ldots\}$

Probleme mit kontextfreien Grammatiken

- **Mehrdeutigkeit:**
  - Beispielgrammatik ({S}, {+,x,3}, R, A}
  - R = {S → S + S, S → S x S, S → 3}
  - Betrachte das Wort: w = 3+3x3
    - $S \Rightarrow S+S \Rightarrow S + SxS \Rightarrow^* 3 + 3x3$
    - $S \Rightarrow SxS \Rightarrow S+S \text{ x } S \Rightarrow^* 3+3 \text{ x } 3$
  - Die Ableitung ist jeweils verschieden
    - damit ist auch die Interpretation verschieden

Probleme mit kontextfreien Grammatiken

- **Epsilon-Regeln**
  - Beispielgrammatik:
    $(\{S,A,B,C\}, \{a,b,c\}, R, S\ \})$
  - $R = \{\ S \to ABC \mid \varepsilon,$
    $A \to BaB \mid \varepsilon,$
    $B \to \varepsilon \mid c,$
    $C \to BAbACB \mid \varepsilon\ \}$

- $S \Rightarrow ABC$
  $\Rightarrow ABBAbACB$
  $\Rightarrow ABBAbA\ BAbACB\ B$
  $\Rightarrow ABBAbA\ BAbA\ BAbACB\ B\ B$
  $\Rightarrow ABBAbA\ cAbA\ BAbACB\ B\ B$
  $\Rightarrow^{*}\ \varepsilon\varepsilon\varepsilon\varepsilon b\varepsilon c\varepsilon b\varepsilon\varepsilon\varepsilon b\varepsilon\varepsilon\varepsilon\varepsilon\varepsilon\varepsilon$
  $= bcbba$

- **Zwischenergebnisse können gigantisch lang werden und dann wieder verschwinden**

Formale Sprachen und Endliche Automaten

Chomsky-Normalform

25

Ist ein Wort in der Sprache?

- **Die Entscheidung, ob ein Wort in einer kontextfreien Grammatik abgeleitet werden kann ist nicht trivial:**
  - Mehrdeutige Ableitung
  - Riesige Worte, die aufgrund der Epsilon-Regeln wieder verschwinden
- **Wenig aussichtsreiche Ansätze:**
  1. Versuche Terminalsymbole von links nach rechts zu erzeugen
     - Führt für viele Sprachen in Sackgassen
  2. Probiere alle Ableitungsmöglichkeiten aus
     - Grammatik mit Epsilon-Regeln: Unendliche Laufzeit
     - Grammatik ohne Epsilon-Regeln: Unglaublich schlechte (= exponentielle) Laufzeit
- **Die Lösungsstrategie:**
  - Chomsky-Normalform
    - Jede Grammatik lässt sich (praktisch) ohne Epsilon-Regeln in einer einfachen Form darstellen
  - Cocke-Younger-Kasami-Algorithms
    - Durch dynamische Programmierung lässt sich die Laufzeit reduzieren

Chomsky-Normalform

- **Definition**
  - Eine kontextfreie Grammatik ist in Chomsky-Normalform, falls jede Regel die Form
    - $A \rightarrow BC$ oder
    - $A \rightarrow a$ oder
    - $S \rightarrow \varepsilon$ hat
  - wobei
    - A,B,C,S sind Variablen
    - a ist ein Terminal
    - B,C sind nicht das Startsymbol,
    - S ist das Startsymbol.

- **Beispiel**
  - G = ({A,B,C,N,E}, {0,1,#}, R, S)
  - R = { S → NC,
N → 0,
S → #,
A → NC,
C → AE,
E → 1,
A → #}

Chomsky-Normalform

- **Definition**
  - Eine kontextfreie Grammatik ist in Chomsky-Normalform, falls jede Regel die Form
    - $A \rightarrow BC$ oder
    - $A \rightarrow a$ oder
    - $S \rightarrow \varepsilon$ hat
  - wobei
    - A,B,C,S sind Variablen
    - a ist ein Terminal
    - B,C sind nicht das Startsymbol,
    - S ist das Startsymbol.

- **Theorem**
  - Jede kontextfreie Sprache kann in Chomsky-Normalform dargestellt werden.

- **Beispiel: Kontextfreie Grammatik**
  - $G = (\{A,B\}, \{0,1,\#\}, R, A)$
  - $R = \{A \rightarrow 0A1, A \rightarrow B, B \rightarrow \#\}$
- **Grammatik mit gleicher Sprache in Chomski-Normalform**
  - $G' = (\{A,B,C,N,E\}, \{0,1,\#\}, R, S)$
  - $R = \{S \rightarrow NC, N \rightarrow 0, S \rightarrow \#,$
    $A \rightarrow NC, C \rightarrow AE, E \rightarrow 1, A \rightarrow \#\}$

Chomsky-Normalform

- **Chomsky-Normalform**
  - $A \rightarrow BC$
  - $A \rightarrow a$
  - $S \rightarrow \varepsilon$
- **Theorem**
  - Jede kontextfreie Sprache kann in Chomsky-Normalform dargestellt werden.
- **Beweisidee:**
  - Forme alle Produktionsregeln um, welche gegen die Normalform verstoßen
  - Führe ggf. neue Variablen ein

- **Vier Probleme**
  - Startsymbol auf der rechten Seite
    - Lösung: Neues Startsymbol
  - Epsilon-Regeln: $A \rightarrow \varepsilon$
    - Lösung: Falls A in einer Regel rechts vorkommt, füge neue Regeln ohne A rechts ein
  - Eins-zu-eins-Regeln $A \rightarrow B$
    - Lösung: In Ziele einsetzen
  - Lange und gemischte Regeln: $A \rightarrow aBcAbA$
    - Lösung: neue Variablen&neue Regeln für schrittweise Ableitung

Konstruktion der Chomsky-Normalform

- **Theorem**
  - Jede kontextfreie Sprache kann in Chomsky-Normalform dargestellt werden.
- **Beweis:**
  - Gegeben eine kontextfreie Grammatik $G=(V,\Sigma,R,S)$
  - Konstruiere äquivalente Grammatiken G_1,G_2,G_3,G_4
  - G_4 ist dann eine kontextfreie Grammatik in Chomsky-Normalform
    - $A \rightarrow BC$ oder
    - $A \rightarrow a$ oder
    - $S \rightarrow \varepsilon$
- **Konstruiere äquiv. kontextfreie Grammatik G_1, in der das Startsymbol S nicht auf der rechten Seite vorkommt**
  - $G_1 = (V,\Sigma,R',S')$
  - R' besteht aus allen Regeln von R und der neuen Regel $S' \rightarrow S$
  - G_1 ist äquivalent zu G und erfüllt Bedingung 1.

Beweis Fortsetzung

- **Gegeben**
  - eine kontextfreie Grammatik G_1 = (V,Σ,R,S), in der das Startsymbol S nicht auf der rechten Seite vorkommt
  - Konstruiere eine äquiv. Grammatik G_2 = (V,Σ,R',S) ohne Regeln der Form A → ε, für A ∈ V \ {S}
- **R' entsteht durch Modifikation von R**
  - Für jede Regel A → ε aus R mit A ∈ V \ {S}
    - Lösche Regel A → ε aus R
    - Falls A auf der rechten Seite einer Regel einfach vorkommt,
      - d.h. B → uAv
    - füge Regel B → uv zu R hinzu
      - Falls uv = ε (also B → A)
      - füge B → ε nur hinzu, falls B → ε noch nicht behandelt wurde
    - Falls A auf der rechten Seite einer Regel mehrfach vorkommt,
      - z.B. B → uAvAw
    - füge jede Kombination der Ersetzungen von A durch ε zu R' hinzu
      - Also: B → uAvAw, B → uvAw, B → uAvw, B → uvw
      - Falls uvw= ε füge B → ε nur hinzu, falls B → ε noch nicht behandelt wurde

Beweis 2. Fortsetzung

- **Gegeben eine kontextfreie Grammatik $G_2 = (V,\Sigma,R,S)$,**
  - in der das Startsymbol S nicht auf der rechten Seite vorkommt
  - ohne Regeln der Form
    - $A \to \varepsilon$, für $A \in V \setminus \{S\}$
- **Konstruiere eine äquiv. Grammatik $G_3 = (V,\Sigma,R',S)$ ohne Regeln der Form $A \to B$**
  - R' entsteht durch folgende Modifikation von R
  - Für jede Regel $A \to B$ aus R
    - Lösche Regel $A \to B$ aus R
    - Für jede Regel $B \to u$
      - füge Regel $A \to u$ hinzu
      - außer
        - $A \to u$ hat die Form $A \to C$ für $A,C \in V$
        - und diese Regel wurde schon behandelt
- **Korrektheitsbeweis**
  - folgt durch Betrachtung der Ableitung eines Wortes in G_2 und G_3

Beweis 3. Fortsetzung

- **Gegeben eine kontextfreie Grammatik $G_3 = (V,\Sigma,R,S)$,**
  - in der das Startsymbol S nicht auf der rechten Seite vorkommt
  - ohne Regeln $A \to \varepsilon$, für $A \in V \setminus \{S\}$
  - ohne Regeln $A \to B$ für $A,B \in V$
- **Konstruiere kontextfreie Grammatik $G_4 = (V',\Sigma,R',S)$ ohne Regeln**
  - $A \to u_1 u_2 \ldots u_m$ für $m \geq 2$, wobei ein Symbol u_i ein Terminal ist
- **Führe für jedes Terminal a eine Variable V[a] ein**
  - Füge Regeln $V[a] \to a$ für jedes Terminal a hinzu
  - Ersetze in jeder Regel $A \to u_1 u_2 \ldots u_n$ für $n \geq 2$ die Vorkommen der Terminale u_i durch die Hilfsvariablen $V[u_i]$

Beweis 4. Fortsetzung und Schluss

- **Gegeben eine kontextfreie Grammatik $G_4 = (V,\Sigma,R,S)$,**
  - in der das Startsymbol S nicht auf der rechten Seite vorkommt
  - ohne Regeln $A \rightarrow \varepsilon$, für $A \in V\setminus\{S\}$
  - ohne Regeln $A \rightarrow B$ für $A,B \in V$
  - ohne Regeln $A \rightarrow u_1u_2...u_m$ für $m \geq 2$, wobei ein Symbol u_i ein Terminal ist

- **Konstruiere kontextfreie Grammatik $G_5 = (V',\Sigma,R',S)$ in Chomsky-Normalform, d.h.**
  - ohne Regeln $A \rightarrow u_1u_2...u_m$ für $m \geq 3$

- **Für jede Regel $A \rightarrow u_1u_2...u_m$ mit $m \geq 3$**
  - Lösche diese Regel
  - Füge neue Variablen $V[u_2...u_m]$, $V[u_3...u_m]$, $V[u_{m-1}u_m]$ hinzu, sowie die Regeln
    - $A \rightarrow u_1V[u_2...u_m]$,
    - $V[u_2...u_m] \rightarrow u_2\, V[u_2...u_m]$,
    - $V[u_3...u_m] \rightarrow u_3V[u_4...u_m]$,

      ...
    - $V[u_{m-1}u_m] \rightarrow u_{m-1}u_m$
- **Entstandene Grammatik G_4 ist äquivalent und in Chomsky-Normalform.**

Beispiel: 1. Schritt

- **Betrachte**
  - kontextfreie Grammatik (V,Σ,R,S) mit den Regeln mit
  - $V = \{A,B,S\}$
  - $\Sigma = \{a,b\}$
  - $R = \{ S \rightarrow ASA,$
    $S \rightarrow aB,$
    $A \rightarrow B,$
    $A \rightarrow S,$
    $B \rightarrow b,$
    $B \rightarrow \varepsilon \}$

- **Äquivalente kontextfreie Grammatik $G' = (V,\Sigma,R,S')$**
  - ohne Vorkommen des Startsymbols auf der rechten Seite
  - S' neues Startsymbol
  - $R = \{ S' \rightarrow S,$
    $S \rightarrow ASA,$
    $S \rightarrow aB,$
    $A \rightarrow B,$
    $A \rightarrow S,$
    $B \rightarrow b,$
    $B \rightarrow \varepsilon \}$

Beispiel: 2. Schritt

- **Ausgangsregeln**
  - R = { S' → S,
    S → ASA,
    S → aB,
    A → B,
    A → S,
    B → b,
    B → ε }
- **Entfernen von B → ε, neue Regeln:**
  - R' = { S' → S,
    S → ASA,
    S → aB, S → a,
    A → B,
    A → ε,
    A → S,
    B → b }
- **Entfernen von A → ε, neue Regeln:**
  - R" = { S' → S,
    S → ASA,
    S → SA,
    S → AS,
    S → S,
    S → aB,
    S → a,
    A → B,
    A → S,
    B → b }

Beispiel: 3a1. Schritt

- **Start:**
  - R = { $S' \rightarrow S$,
    $S \rightarrow ASA$,
    $S \rightarrow SA$,
    $S \rightarrow AS$,
    $S \rightarrow S$,
    $S \rightarrow aB$,
    $S \rightarrow a$,
    $A \rightarrow B$,
    $A \rightarrow S$,
    $B \rightarrow b$ }

- **Entferne $S \rightarrow S$**
  - R' = { $S' \rightarrow S$,
    $S \rightarrow ASA$,
    $S \rightarrow SA$,
    $S \rightarrow AS$,
    $S \rightarrow aB$,
    $S \rightarrow a$,
    $A \rightarrow B$,
    $A \rightarrow S$,
    $B \rightarrow b$ }

Beispiel: 3a2. Schritt

- **Start**
  - $R' = \{\ S' \to S,\ S \to ASA,\ S \to SA,\ S \to AS,\ S \to aB,\ S \to a,\ A \to B,\ A \to S,\ B \to b\ \}$

- **Entferne $S' \to S$, neue Regeln:**
  - $R'' = \{\ S' \to ASA,\ S' \to SA,\ S' \to AS,\ S' \to aB,\ S' \to a\ S \to ASA,\ S \to SA,\ S \to AS,\ S \to aB,\ S \to a,\ A \to B,\ A \to S,\ B \to b\ \}$

Beispiel: 3a3. Schritt

- **Start**
  - R” = { S’ → ASA,
    S’ → SA,
    S’ → AS,
    S’ → aB,
    S’ → a
    S → ASA,
    S → SA,
    S → AS,
    S → aB,
    S → a,
    A → B,
    A → S,
    B → b }

- **Entferne A → B, neue Regeln:**
  - R”’ = { S’ → ASA,
    S’ → SA
    S’ → AS,
    S’ → aB,
    S’ → a,
    S → ASA,
    S → SA,
    S → AS,
    S → aB,
    S → a,
    A → b,
    A → S,
    B → b }

Beispiel: 3b. Schritt

- **Start:**
  - R' = { S' → ASA,
    S' → SA
    S' → AS,
    S' → aB,
    S' → a,
    S → ASA,
    S → SA,
    S → AS,
    S → aB,
    S → a,
    A → b,
    A → S,
    B → b }

- **Entferne A → S, neue Regeln**
  - R" = { S' → ASA,
    S' → SA
    S' → AS,
    S' →aB,
    S' → a,
    S → ASA,
    S → SA,
    S → AS,
    S → aB,
    S → a,

-
  A → b,
  A → ASA,
  A → SA,
  A → AS,
  A → aB,
  A → a,
  B → b }

Beispiel: 4. Schritt

- **Start**
  - R = {S' → ASA
    S' → SA
    S' → AS
    S' → aB
    S' → a
    S → ASA
    S → SA
    S → AS
    S → aB
    S → a
    A → b
    A → ASA
    A → SA
    A → AS
    A → aB
    A → a
    B → b }

- **Füge Variablen V[SA] V[a] hinzu**
  - R' = {S' → A V[SA]
    V[SA] → SA
    S' → SA
    S' → AS
    S' → V[a]B
    S' → a
    S → A V[SA]
    S → SA
    S → AS
    S → V[a]B
    V[a] → a
    S → a

  - Fortsetzung

    A → b
    A → A V[SA]
    A → SA
    A → AS
    A → V[a] B
    A → a
    B → b }

- **Grammatik R' ist in Chomsky-Normalfom**

Formale Sprachen und
Endliche Automaten

Das Wortproblem von CFG & der CYK-Algorithmus

42

Das Wort-Problem der kontextfreien Sprachen

- **Gegeben:**
  - Eine kontextfreie Grammatik $G=(V,\Sigma,R,S)$ in Chomsky-Normalform und
  - ein Wort $w \in \Sigma^*$
- **Entscheide:**
  - Ist $w \in L(G)$
- **Oder:**
  - Kann w aus dem Startsymbol S abgeleitet werden?

Der Cocke-Younger-Kasami-Algorithmus (CYK-Algorithmus)

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \to \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3. Für jede Variable v
4. Teste, ob $v \to b$ eine Regel ist, wobei $b = w_i$
5. Falls ja, füge v zu $T(i, i)$ hinzu.
6. Für $\ell = 2$ bis n
7. Für $i = 1$ bis $n - \ell + 1$
8. Setze $j = i + \ell - 1$
9. Für $k = i$ bis $j - 1$
10. Für jede Regel $v \to uz$
11. Falls $u \in T(i, k)$ und $z \in T(k + 1, j)$, füge v zu $T(i, j)$ hinzu.
12. Falls S in $T(1, n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge der Teilfolge

i = Startindex der Teilfolge

j = Schlussindex der Teilfolge

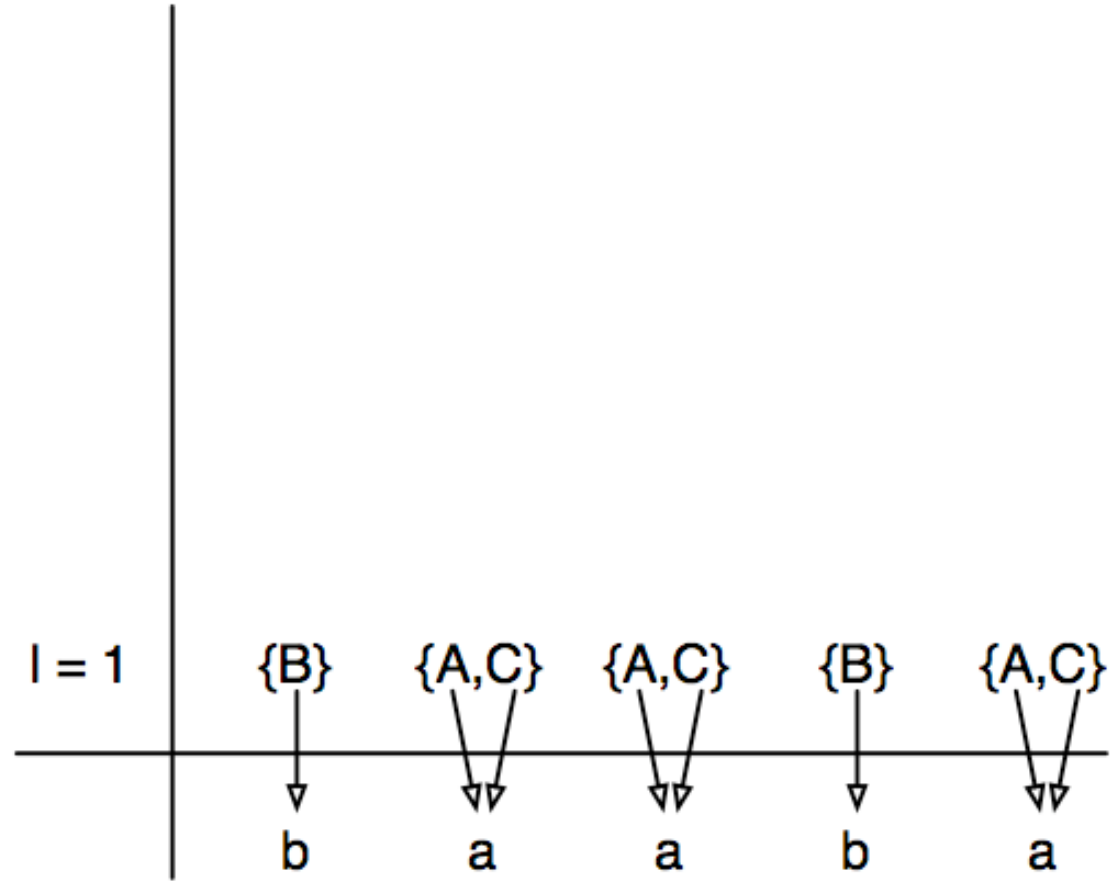
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \to \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3. Für jede Variable v
4. Teste, ob $v \to b$ eine Regel ist, wobei $b = w_i$
5. Falls ja, füge v zu $T(i,i)$ hinzu.
6. Für $\ell = 2$ bis n
7. Für $i = 1$ bis $n - \ell + 1$
8. Setze $j = i + \ell - 1$
9. Für $k = i$ bis $j - 1$
10. Für jede Regel $v \to uz$
11. Falls $u \in T(i,k)$ und $z \in T(k+1,j)$, füge v zu $T(i,j)$ hinzu.
12. Falls S in $T(1,n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = { **S → AB, S → BC,**
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

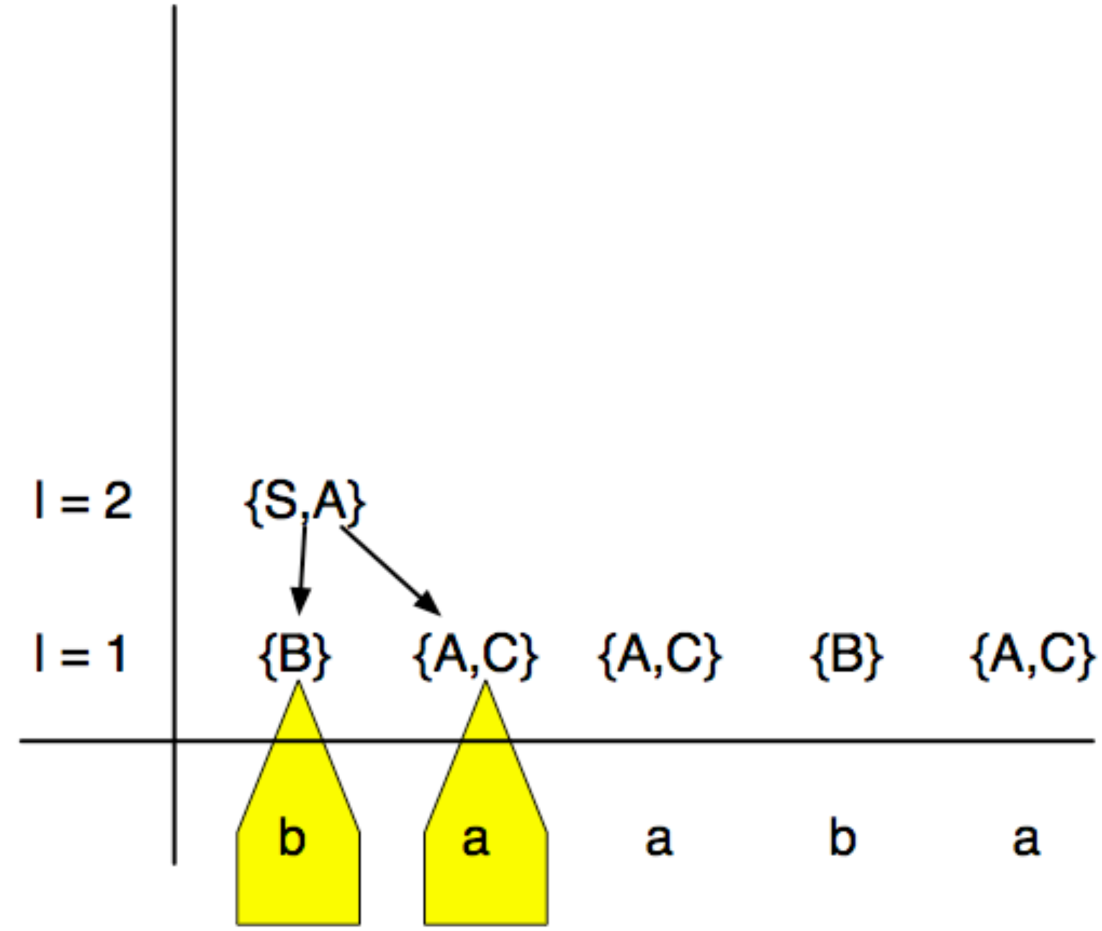
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \to \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3.   Für jede Variable v
4.   Teste, ob $v \to b$ eine Regel ist, wobei $b = w_i$
5.   Falls ja, füge v zu $T(i,i)$ hinzu.
6. Für $\ell = 2$ bis n
7.   Für $i = 1$ bis $n - \ell + 1$
8.   Setze $j = i + \ell - 1$
9.   Für $k = i$ bis $j - 1$
10.   Für jede Regel $v \to uz$
11.   Falls $u \in T(i,k)$ und $z \in T(k+1,j)$, füge v zu $T(i,j)$ hinzu.
12. Falls S in $T(1,n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = { **S → AB, S → BC,**
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

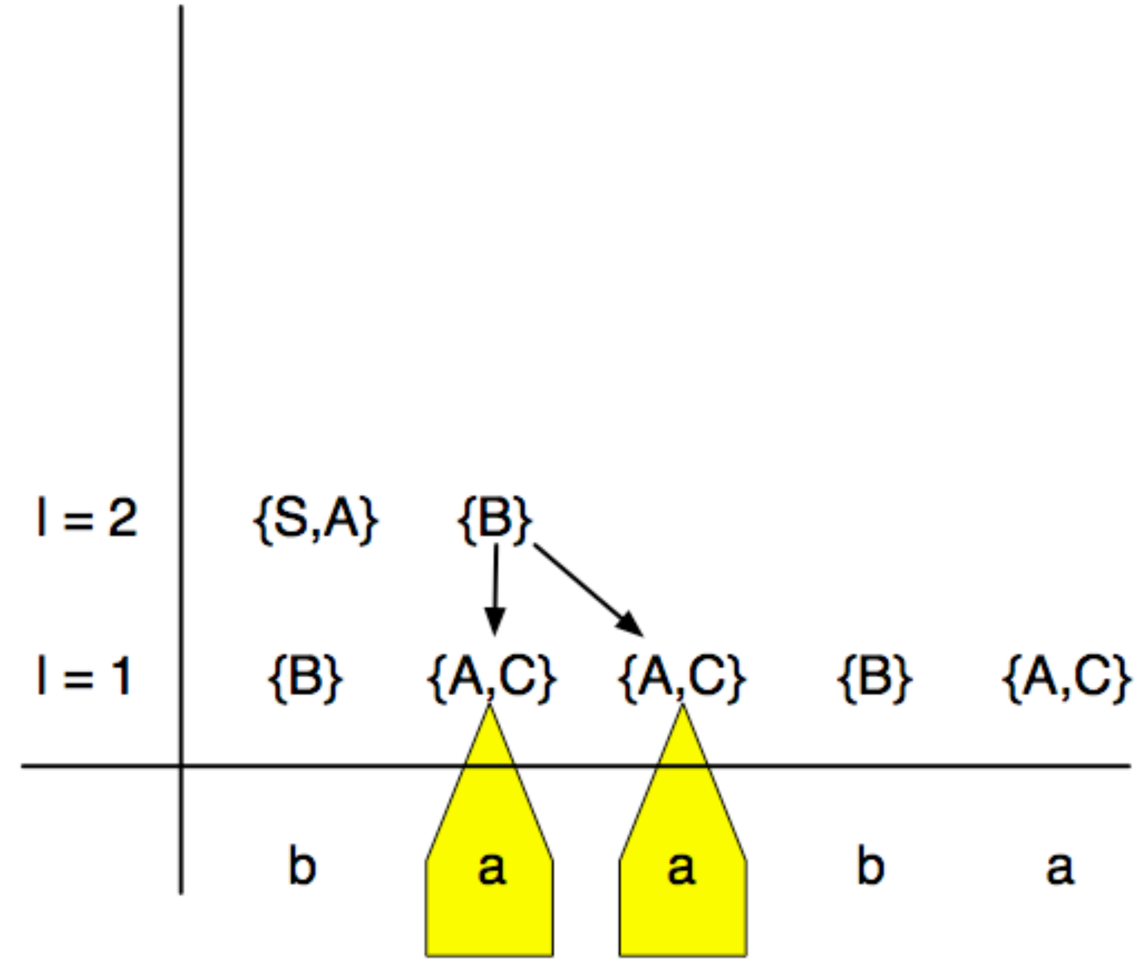
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \to \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3. Für jede Variable v
4. Teste, ob $v \to b$ eine Regel ist, wobei $b = w_i$
5. Falls ja, füge v zu $T(i,i)$ hinzu.
6. Für $\ell = 2$ bis n
7. Für $i = 1$ bis $n - \ell + 1$
8. Setze $j = i + \ell - 1$
9. Für $k = i$ bis $j - 1$
10. Für jede Regel $v \to uz$
11. Falls $u \in T(i,k)$ und $z \in T(k+1,j)$, füge v zu $T(i,j)$ hinzu.
12. Falls S in $T(1,n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = { **S → AB, S → BC,**
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

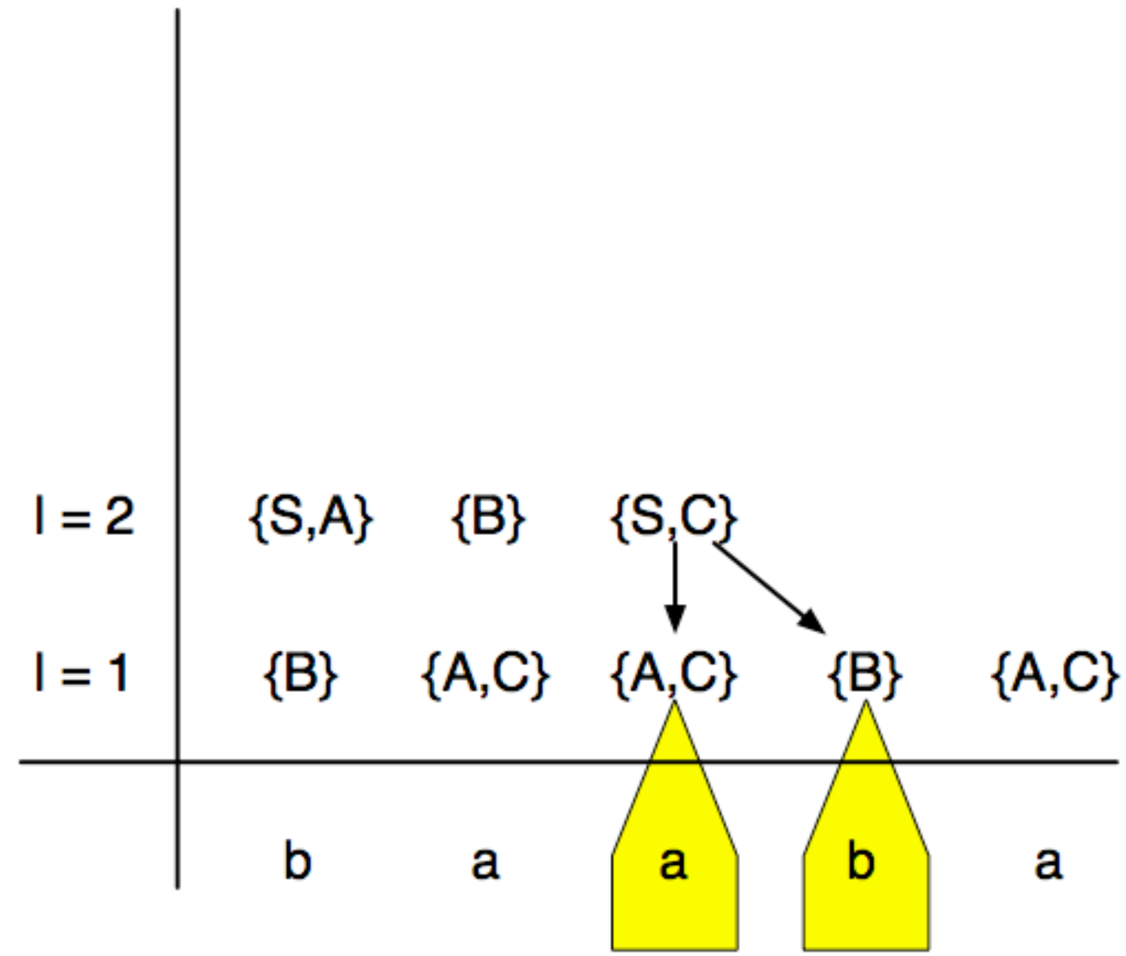
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \to \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3. Für jede Variable v
4. Teste, ob $v \to b$ eine Regel ist, wobei $b = w_i$
5. Falls ja, füge v zu $T(i,i)$ hinzu.
6. Für $\ell = 2$ bis n
7. Für $i = 1$ bis $n - \ell + 1$
8. Setze $j = i + \ell - 1$
9. Für $k = i$ bis $j - 1$
10. Für jede Regel $v \to uz$
11. Falls $u \in T(i,k)$ und $z \in T(k+1,j)$, füge v zu $T(i,j)$ hinzu.
12. Falls S in $T(1,n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = { **$S \to AB$, $S \to BC$,**
$A \to BA$
$B \to CC$,
$C \to AB$,
$A \to a$, $B \to b$, $C \to a$ }

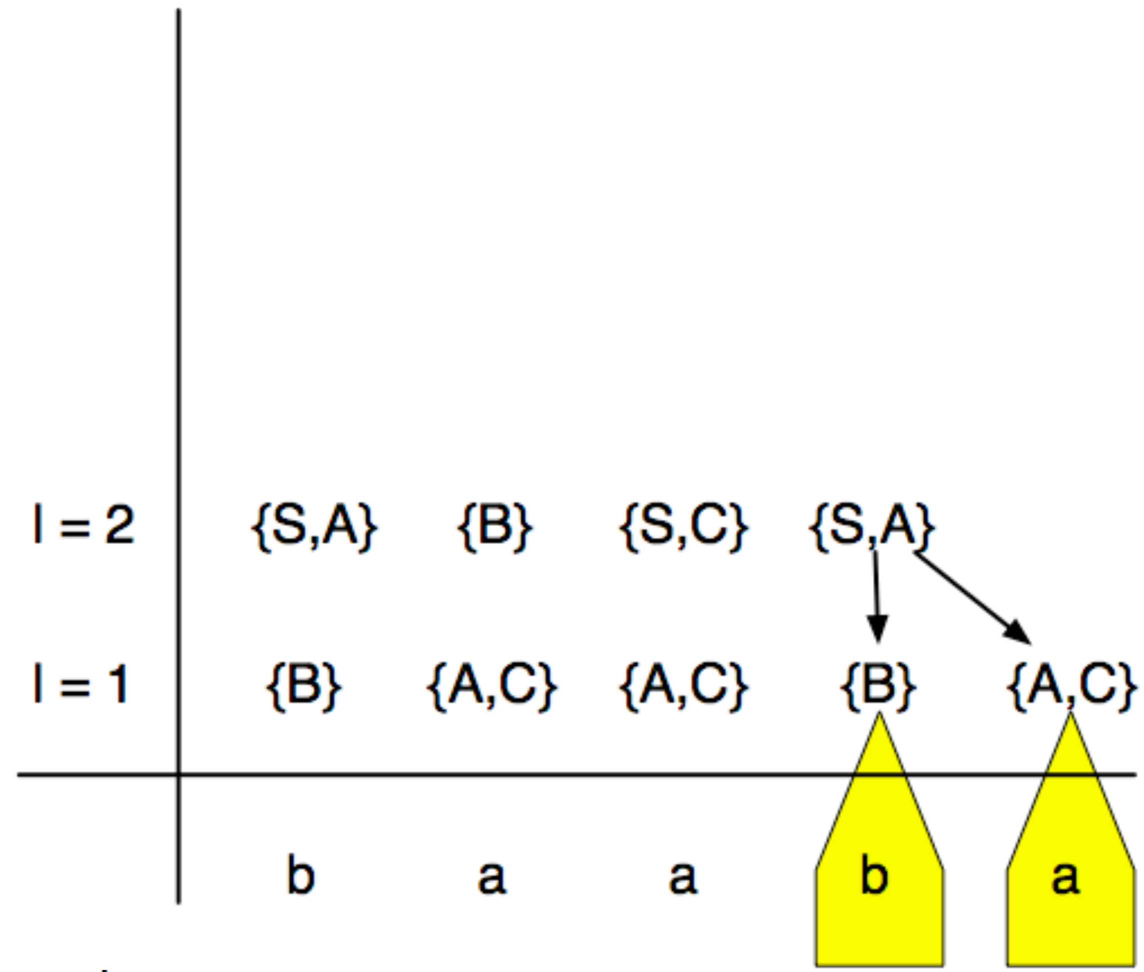
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \to \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3. Für jede Variable v
4. Teste, ob $v \to b$ eine Regel ist, wobei $b = w_i$
5. Falls ja, füge v zu $T(i,i)$ hinzu.
6. Für $\ell = 2$ bis n
7. Für $i = 1$ bis $n - \ell + 1$
8. Setze $j = i + \ell - 1$
9. Für $k = i$ bis $j - 1$
10. Für jede Regel $v \to uz$
11. Falls $u \in T(i,k)$ und $z \in T(k+1,j)$, füge v zu $T(i,j)$ hinzu.
12. Falls S in $T(1,n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = { S → AB, S → BC,
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

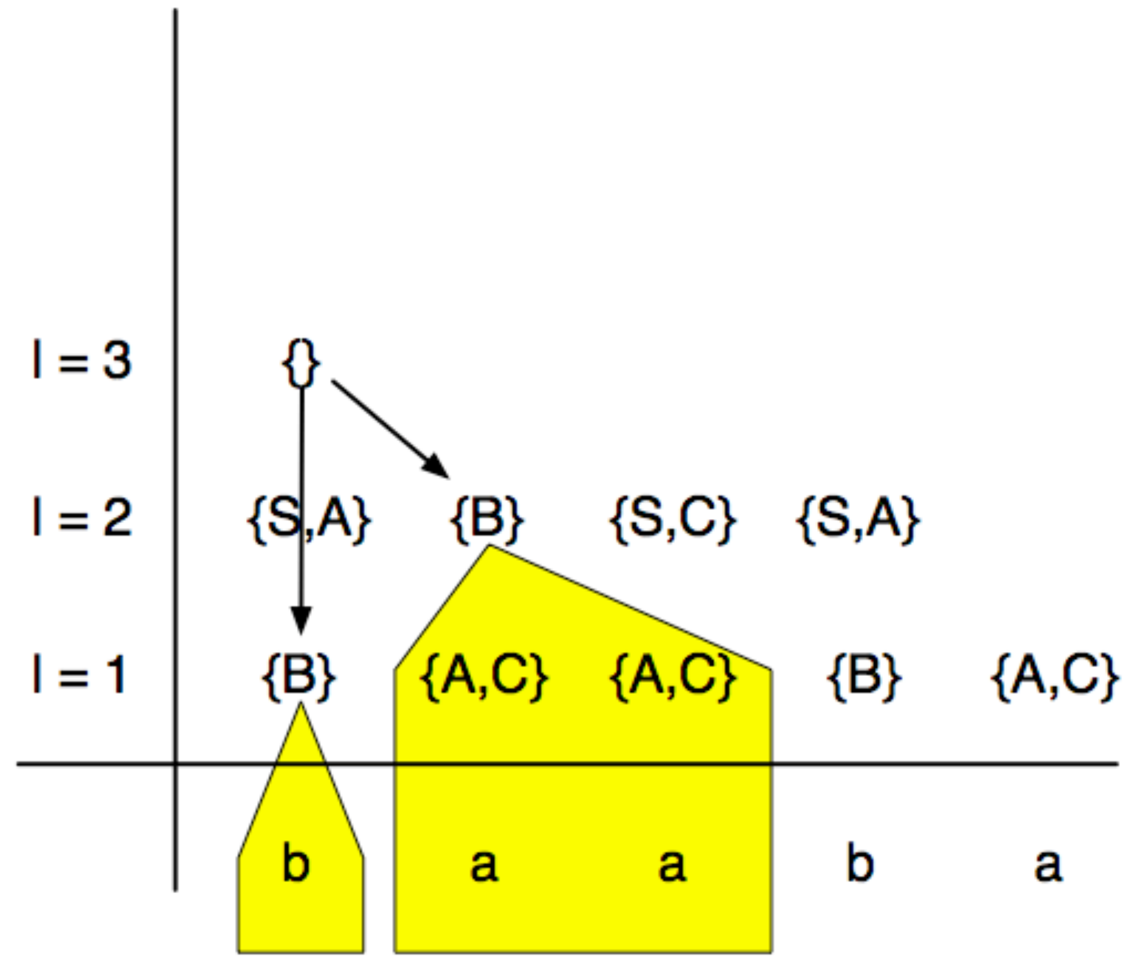
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \rightarrow \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3. Für jede Variable v
4. Teste, ob $v \rightarrow b$ eine Regel ist, wobei $b = w_i$
5. Falls ja, füge v zu $T(i, i)$ hinzu.
6. Für $\ell = 2$ bis n
7. Für $i = 1$ bis $n - \ell + 1$
8. Setze $j = i + \ell - 1$
9. Für $k = i$ bis $j - 1$
10. Für jede Regel $v \rightarrow uz$
11. Falls $u \in T(i, k)$ und $z \in T(k + 1, j)$, füge v zu $T(i, j)$ hinzu.
12. Falls S in $T(1, n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = {
S → AB, S → BC,
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

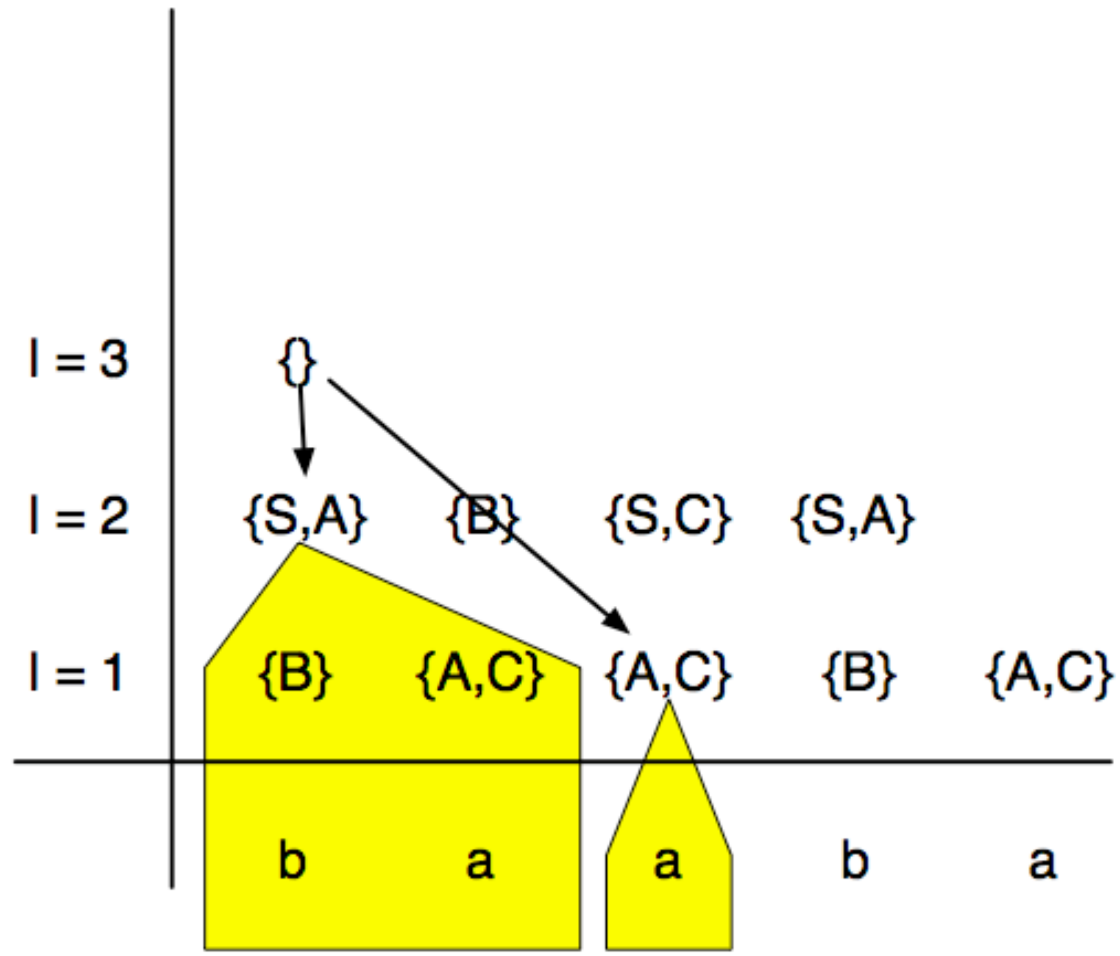
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \to \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3. Für jede Variable v
4. Teste, ob $v \to b$ eine Regel ist, wobei $b = w_i$
5. Falls ja, füge v zu $T(i,i)$ hinzu.
6. Für $\ell = 2$ bis n
7. Für $i = 1$ bis $n - \ell + 1$
8. Setze $j = i + \ell - 1$
9. Für $k = i$ bis $j - 1$
10. Für jede Regel $v \to uz$
11. Falls $u \in T(i,k)$ und $z \in T(k+1,j)$, füge v zu $T(i,j)$ hinzu.
12. Falls S in $T(1,n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = { **S → AB, S → BC,**
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

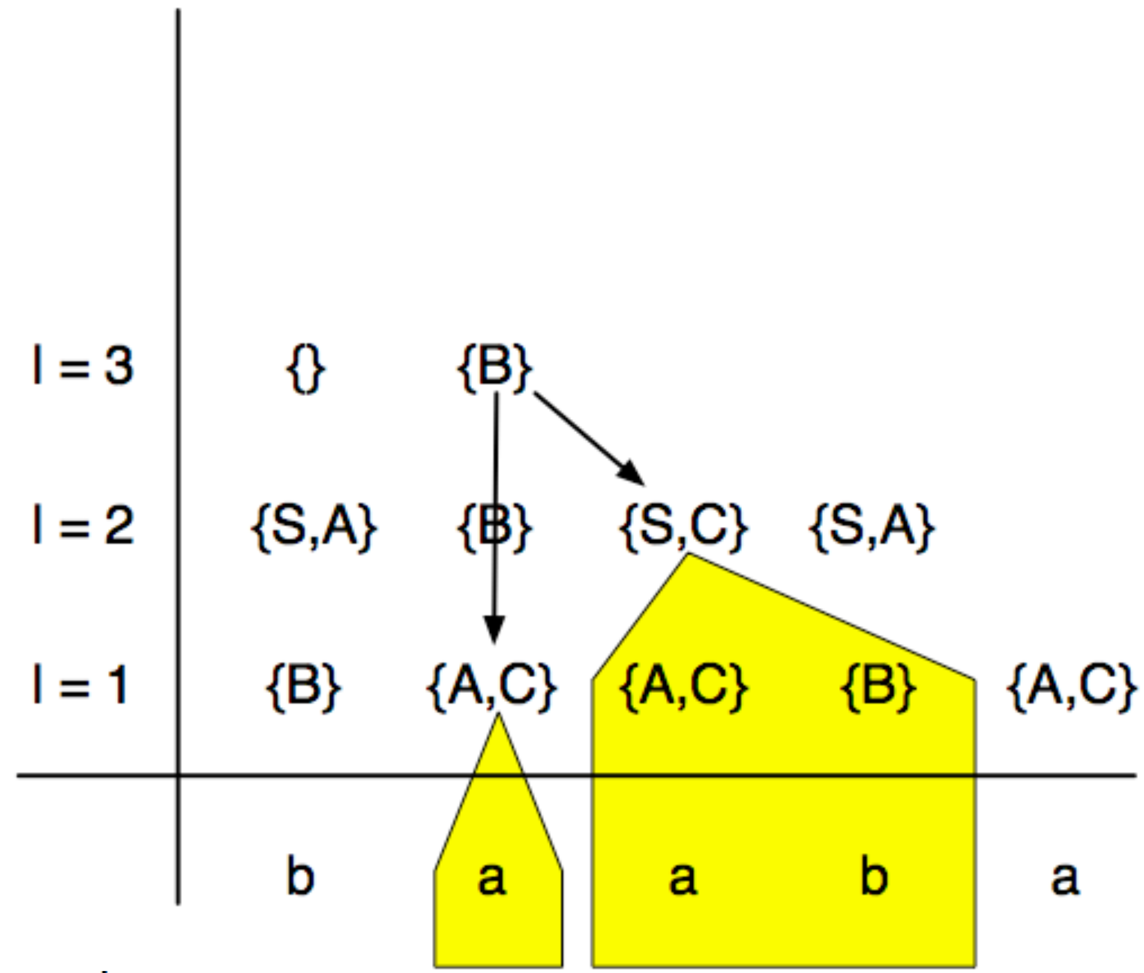
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \to \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3. Für jede Variable v
4. Teste, ob $v \to b$ eine Regel ist, wobei $b = w_i$
5. Falls ja, füge v zu $T(i,i)$ hinzu.
6. Für $\ell = 2$ bis n
7. Für $i = 1$ bis $n - \ell + 1$
8. Setze $j = i + \ell - 1$
9. Für $k = i$ bis $j - 1$
10. Für jede Regel $v \to uz$
11. Falls $u \in T(i,k)$ und $z \in T(k+1,j)$, füge v zu $T(i,j)$ hinzu.
12. Falls S in $T(1,n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = { **S → AB, S → BC,**
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

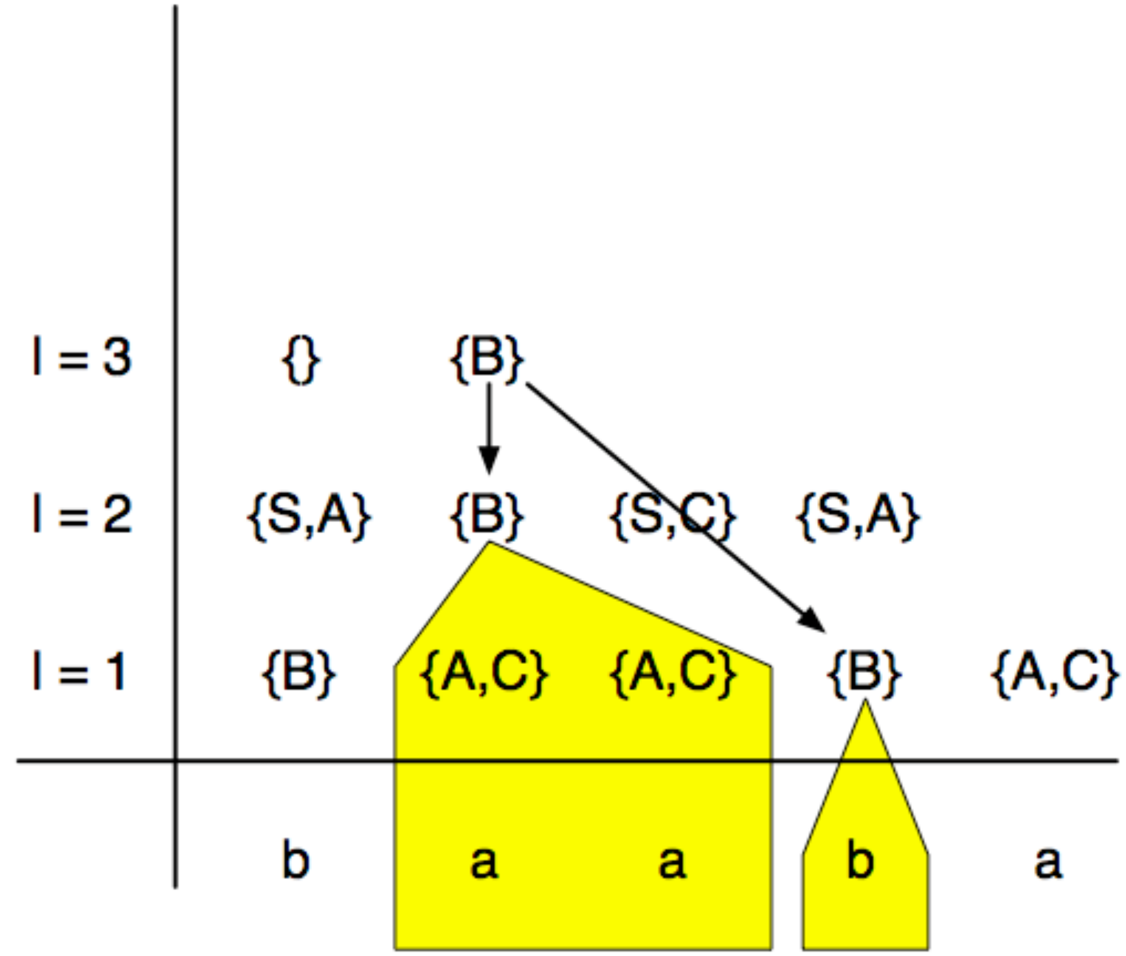
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

Falls $w = \epsilon$ und $S \to \epsilon$ eine Regel ist, akzeptiere.
Für $i = 1$ bis n
  Für jede Variable v
    Teste, ob $v \to b$ eine Regel ist, wobei $b = w_i$
    Falls ja, füge v zu $T(i,i)$ hinzu.
Für $\ell = 2$ bis n
  Für $i = 1$ bis $n - \ell + 1$
    Setze $j = i + \ell - 1$
    Für $k = i$ bis $j - 1$
      Für jede Regel $v \to uz$
        Falls $u \in T(i,k)$ und $z \in T(k+1,j)$,
        füge v zu $T(i,j)$ hinzu.
Falls S in $T(1,n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = { **$S \to AB$, $S \to BC$,**
$A \to BA$
$B \to CC$,
$C \to AB$,
$A \to a$, $B \to b$, $C \to a$ }

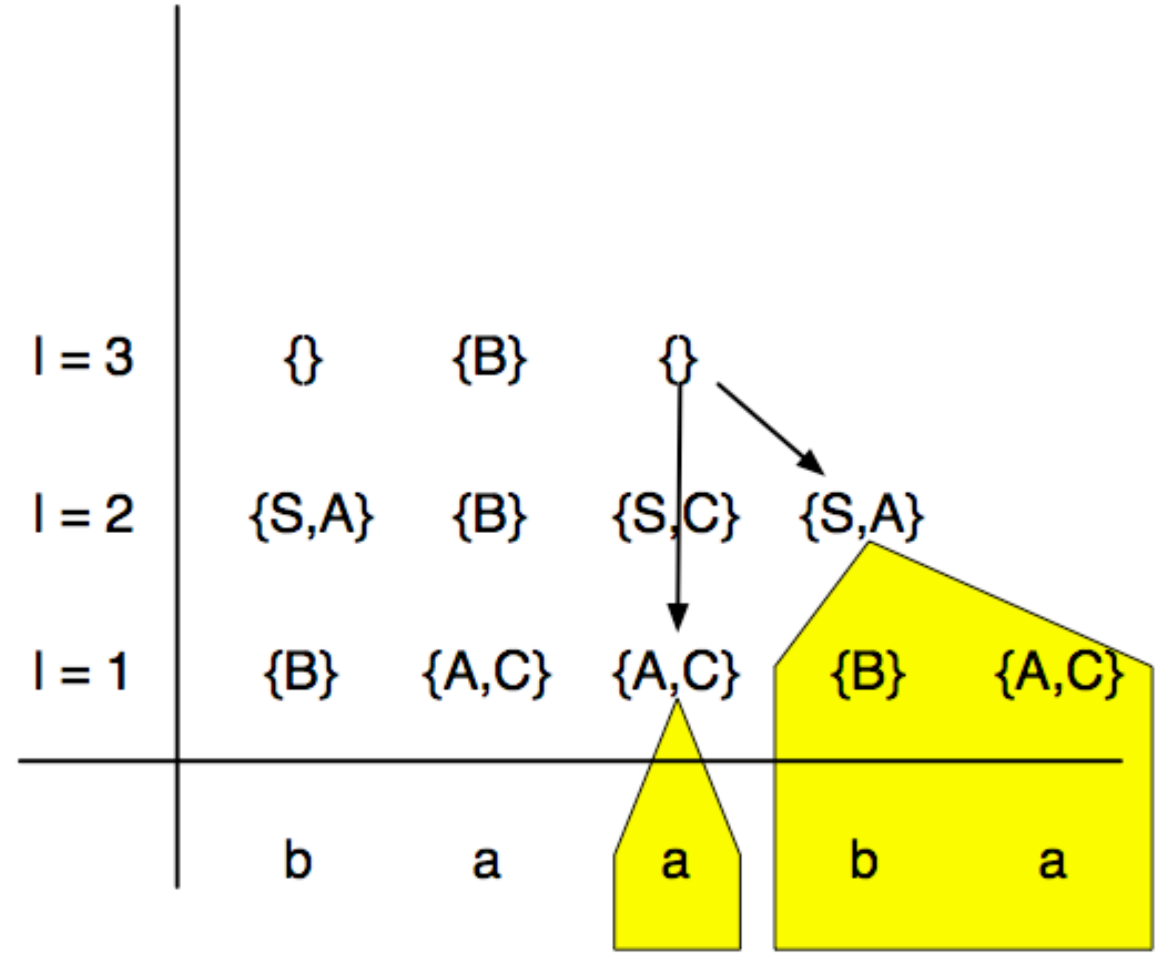
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \to \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3. Für jede Variable v
4. Teste, ob $v \to b$ eine Regel ist, wobei $b = w_i$
5. Falls ja, füge v zu $T(i,i)$ hinzu.
6. Für $\ell = 2$ bis n
7. Für $i = 1$ bis $n - \ell + 1$
8. Setze $j = i + \ell - 1$
9. Für $k = i$ bis $j - 1$
10. Für jede Regel $v \to uz$
11. Falls $u \in T(i,k)$ und $z \in T(k+1,j)$, füge v zu $T(i,j)$ hinzu.
12. Falls S in $T(1,n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = {
S → AB, S → BC,
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

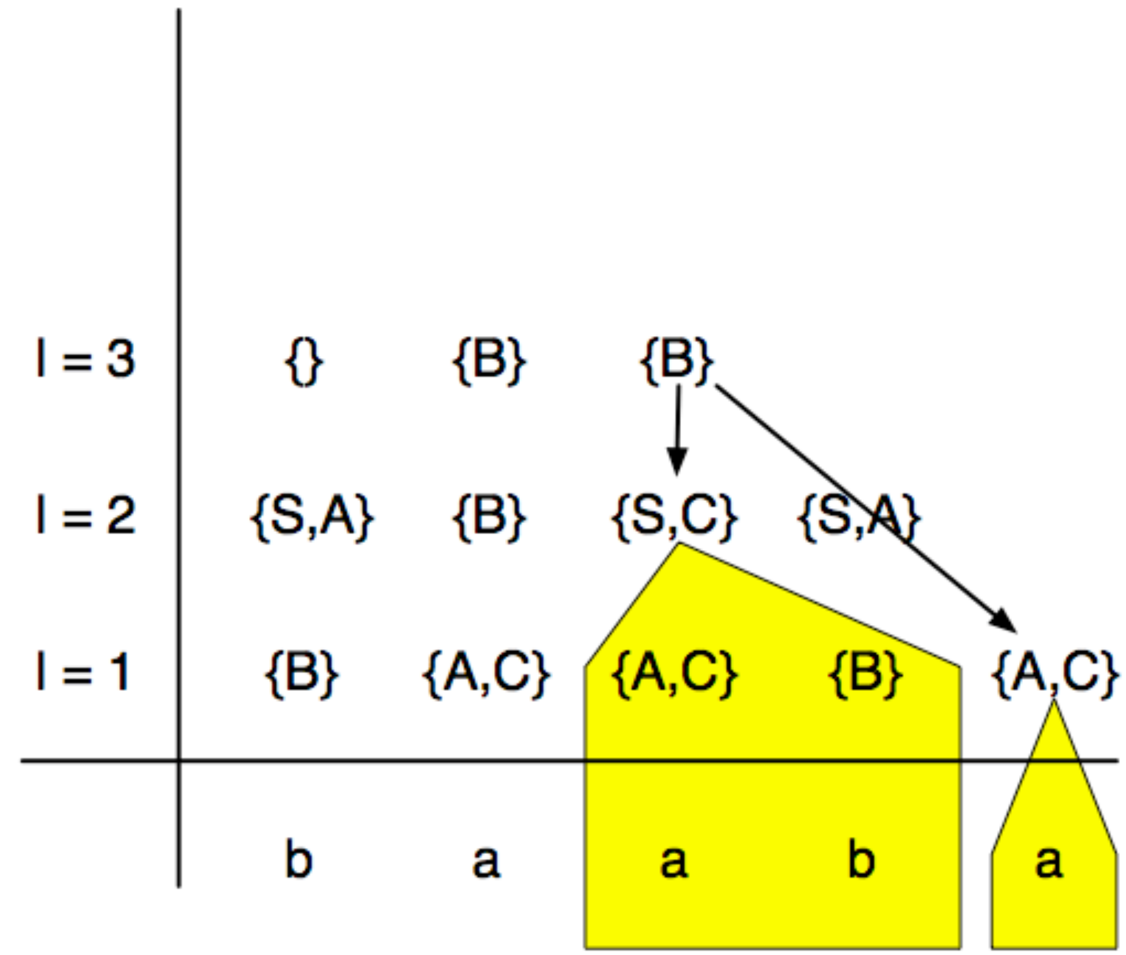
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \rightarrow \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3. Für jede Variable v
4. Teste, ob $v \rightarrow b$ eine Regel ist, wobei $b = w_i$
5. Falls ja, füge v zu $T(i,i)$ hinzu.
6. Für $\ell = 2$ bis n
7. Für $i = 1$ bis $n - \ell + 1$
8. Setze $j = i + \ell - 1$
9. Für $k = i$ bis $j - 1$
10. Für jede Regel $v \rightarrow uz$
11. Falls $u \in T(i,k)$ und $z \in T(k+1,j)$, füge v zu $T(i,j)$ hinzu.
12. Falls S in $T(1,n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = {
S → AB, S → BC,
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

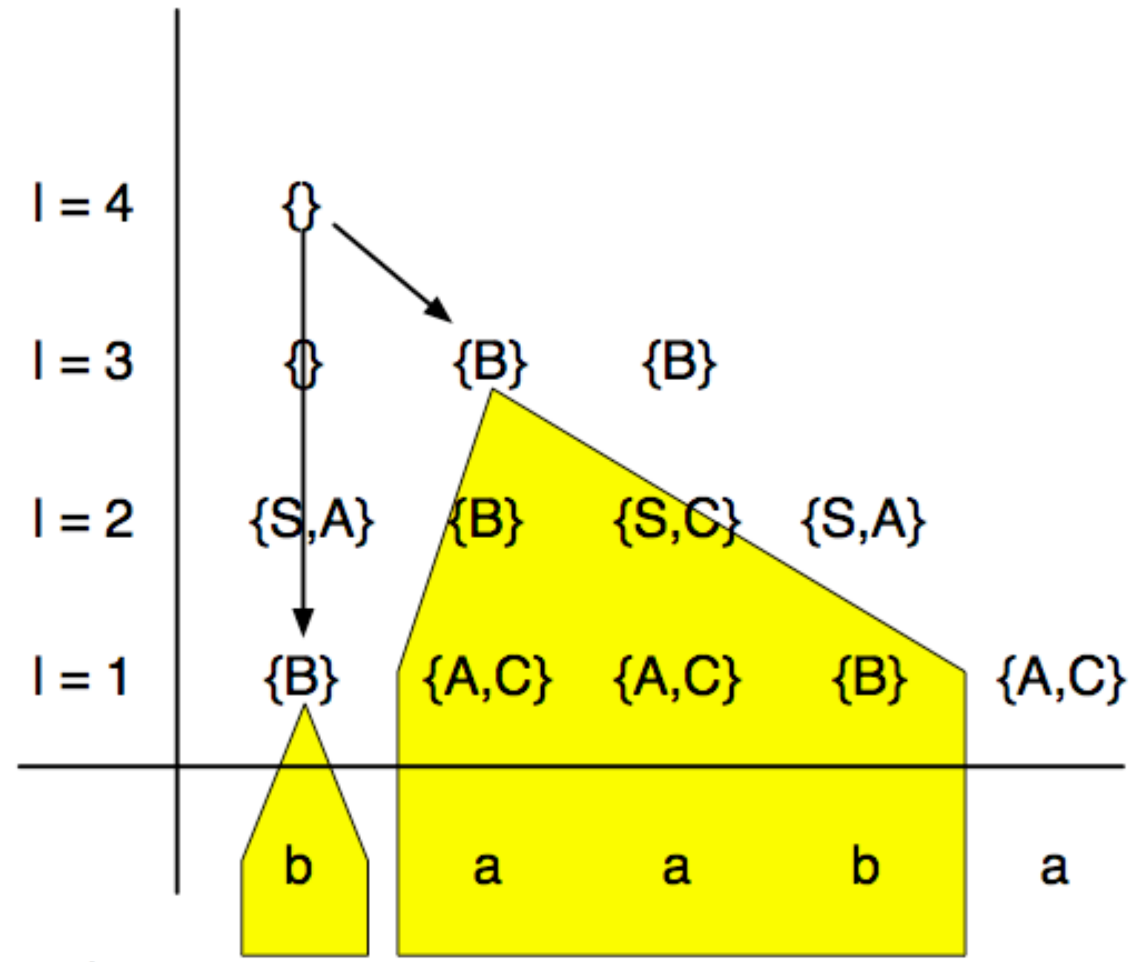
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \to \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3. Für jede Variable v
4. Teste, ob $v \to b$ eine Regel ist, wobei $b = w_i$
5. Falls ja, füge v zu $T(i,i)$ hinzu.
6. Für $\ell = 2$ bis n
7. Für $i = 1$ bis $n - \ell + 1$
8. Setze $j = i + \ell - 1$
9. Für $k = i$ bis $j - 1$
10. Für jede Regel $v \to uz$
11. Falls $u \in T(i,k)$ und $z \in T(k+1,j)$, füge v zu $T(i,j)$ hinzu.
12. Falls S in $T(1,n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = { **S → AB, S → BC,**
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

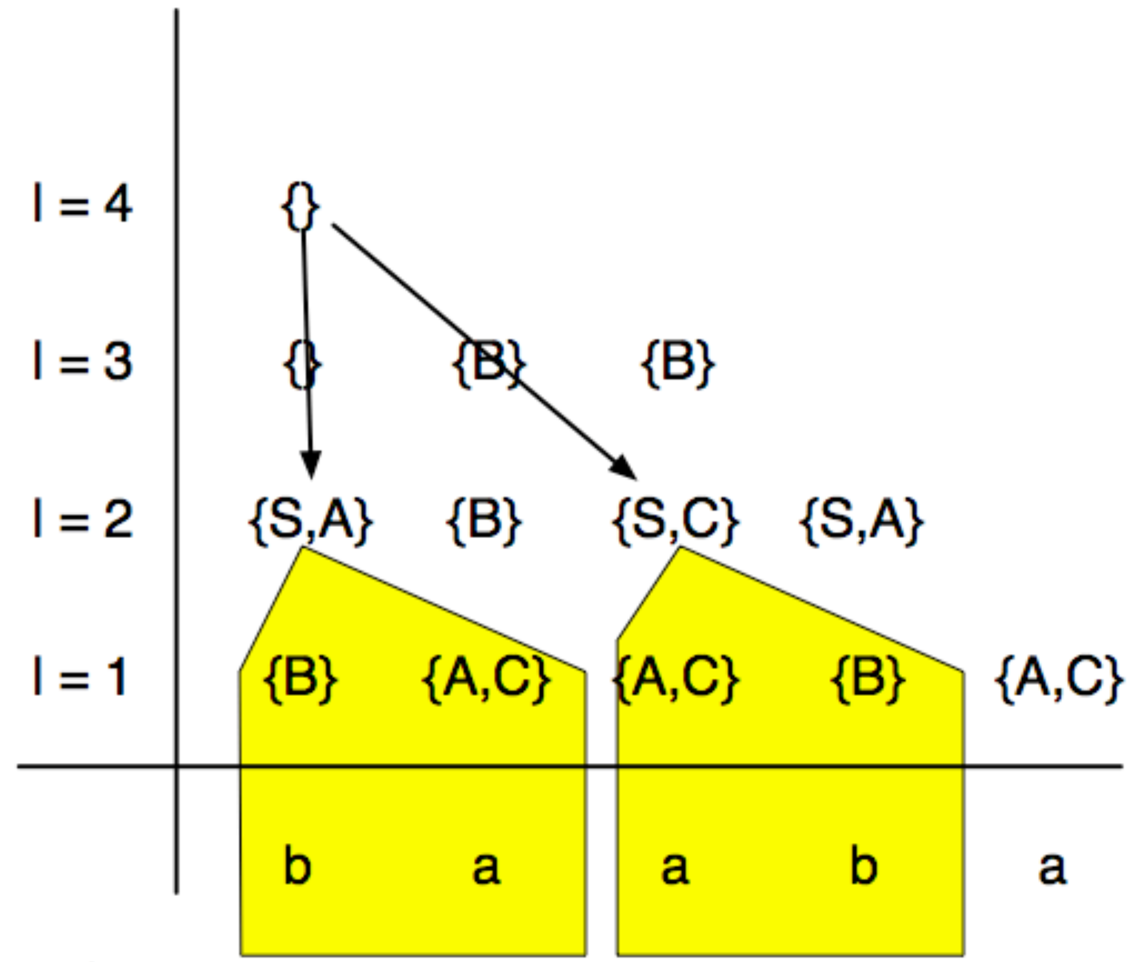
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \rightarrow \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3.   Für jede Variable v
4.   Teste, ob $v \rightarrow b$ eine Regel ist, wobei $b = w_i$
5.   Falls ja, füge v zu $T(i,i)$ hinzu.
6. Für $\ell = 2$ bis n
7.   Für $i = 1$ bis $n - \ell + 1$
8.   Setze $j = i + \ell - 1$
9.   Für $k = i$ bis $j - 1$
10.   Für jede Regel $v \rightarrow uz$
11.   Falls $u \in T(i,k)$ und $z \in T(k+1,j)$, füge v zu $T(i,j)$ hinzu.
12. Falls S in $T(1,n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = {   **S → AB, S → BC,**
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

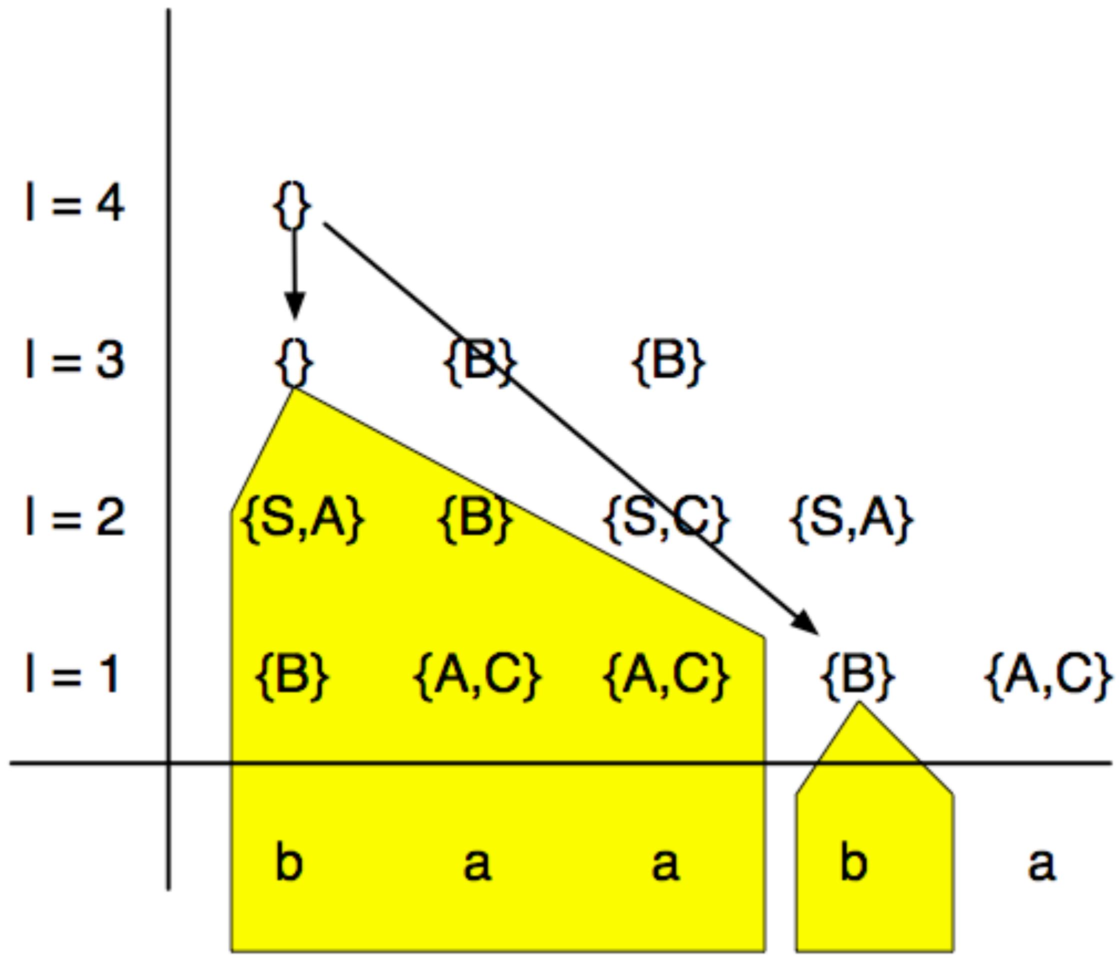
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \to \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3. Für jede Variable v
4. Teste, ob $v \to b$ eine Regel ist, wobei $b = w_i$
5. Falls ja, füge v zu $T(i, i)$ hinzu.
6. Für $\ell = 2$ bis n
7. Für $i = 1$ bis $n - \ell + 1$
8. Setze $j = i + \ell - 1$
9. Für $k = i$ bis $j - 1$
10. Für jede Regel $v \to uz$
11. Falls $u \in T(i, k)$ und $z \in T(k + 1, j)$, füge v zu $T(i, j)$ hinzu.
12. Falls S in $T(1, n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = { S → AB, S → BC,
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

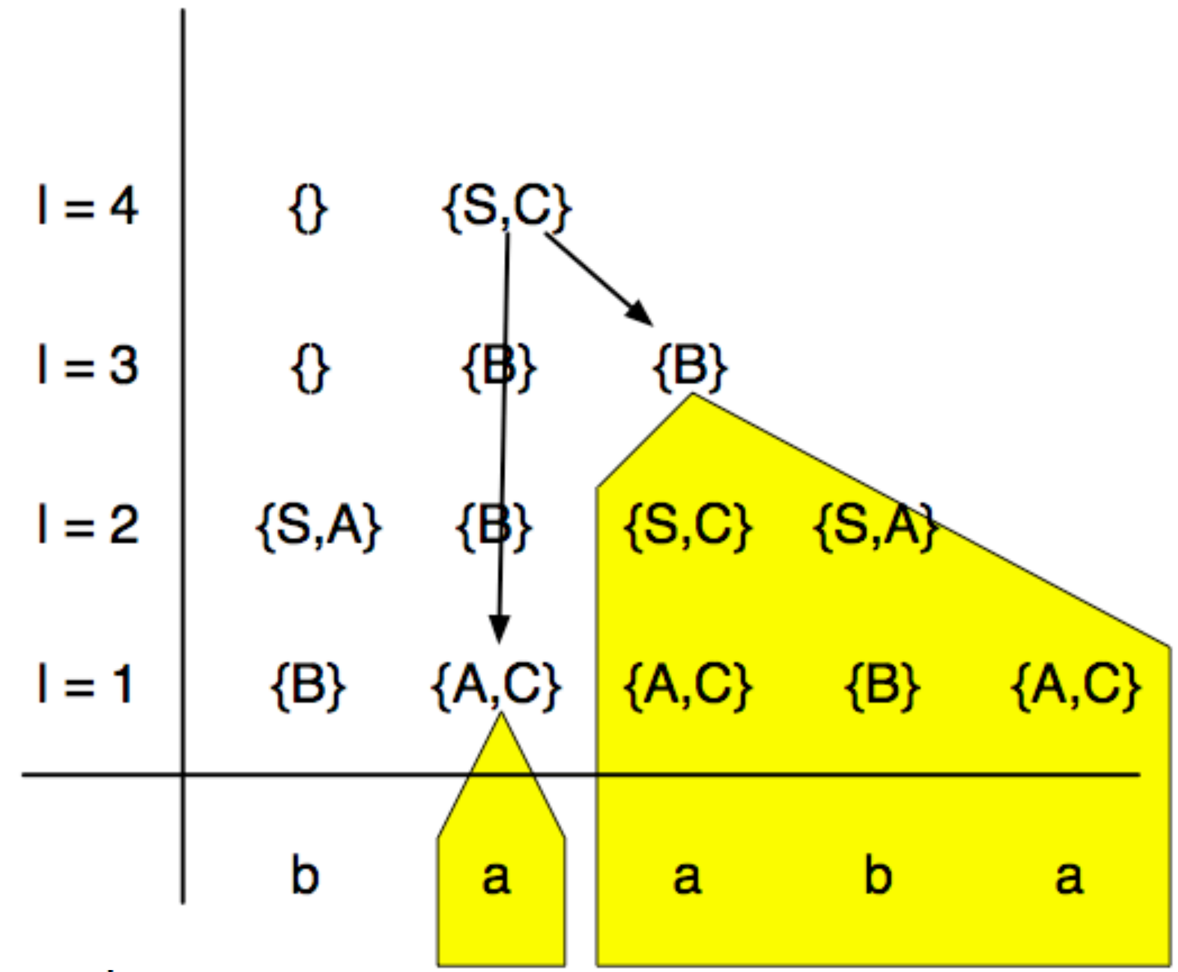
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \to \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3.   Für jede Variable v
4.   Teste, ob $v \to b$ eine Regel ist, wobei $b = w_i$
5.   Falls ja, füge v zu $T(i, i)$ hinzu.
6. Für $\ell = 2$ bis n
7.   Für $i = 1$ bis $n - \ell + 1$
8.   Setze $j = i + \ell - 1$
9.   Für $k = i$ bis $j - 1$
10.   Für jede Regel $v \to uz$
11.   Falls $u \in T(i, k)$ und $z \in T(k+1, j)$,
                füge v zu $T(i, j)$ hinzu.
12. Falls S in $T(1, n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = {
$S \to AB$, $S \to BC$,
$A \to BA$
$B \to CC$,
$C \to AB$,
$A \to a$, $B \to b$, $C \to a$ }

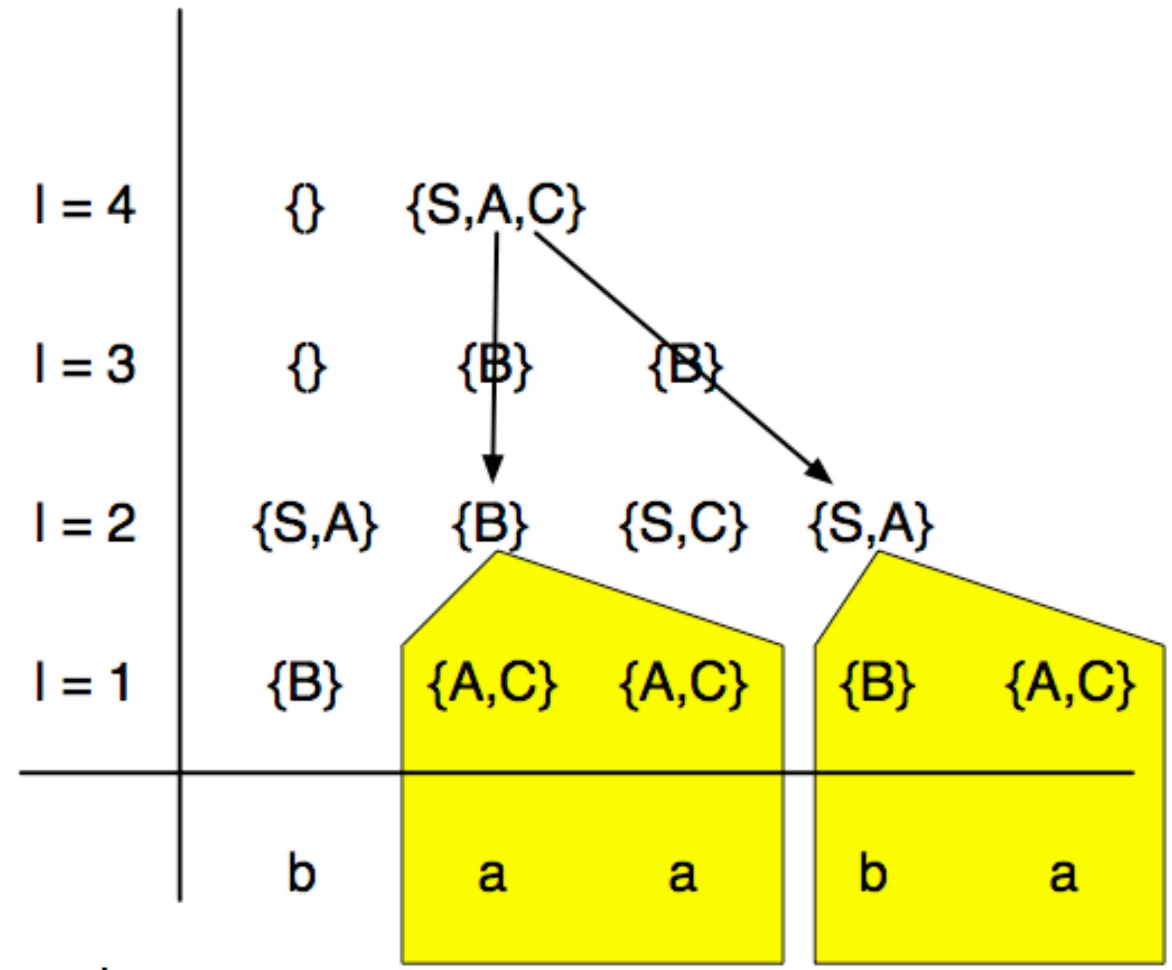
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \to \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3. Für jede Variable v
4. Teste, ob $v \to b$ eine Regel ist, wobei $b = w_i$
5. Falls ja, füge v zu $T(i,i)$ hinzu.
6. Für $\ell = 2$ bis n
7. Für $i = 1$ bis $n - \ell + 1$
8. Setze $j = i + \ell - 1$
9. Für $k = i$ bis $j - 1$
10. Für jede Regel $v \to uz$
11. Falls $u \in T(i,k)$ und $z \in T(k+1,j)$, füge v zu $T(i,j)$ hinzu.
12. Falls S in $T(1,n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = { **S → AB, S → BC,**
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

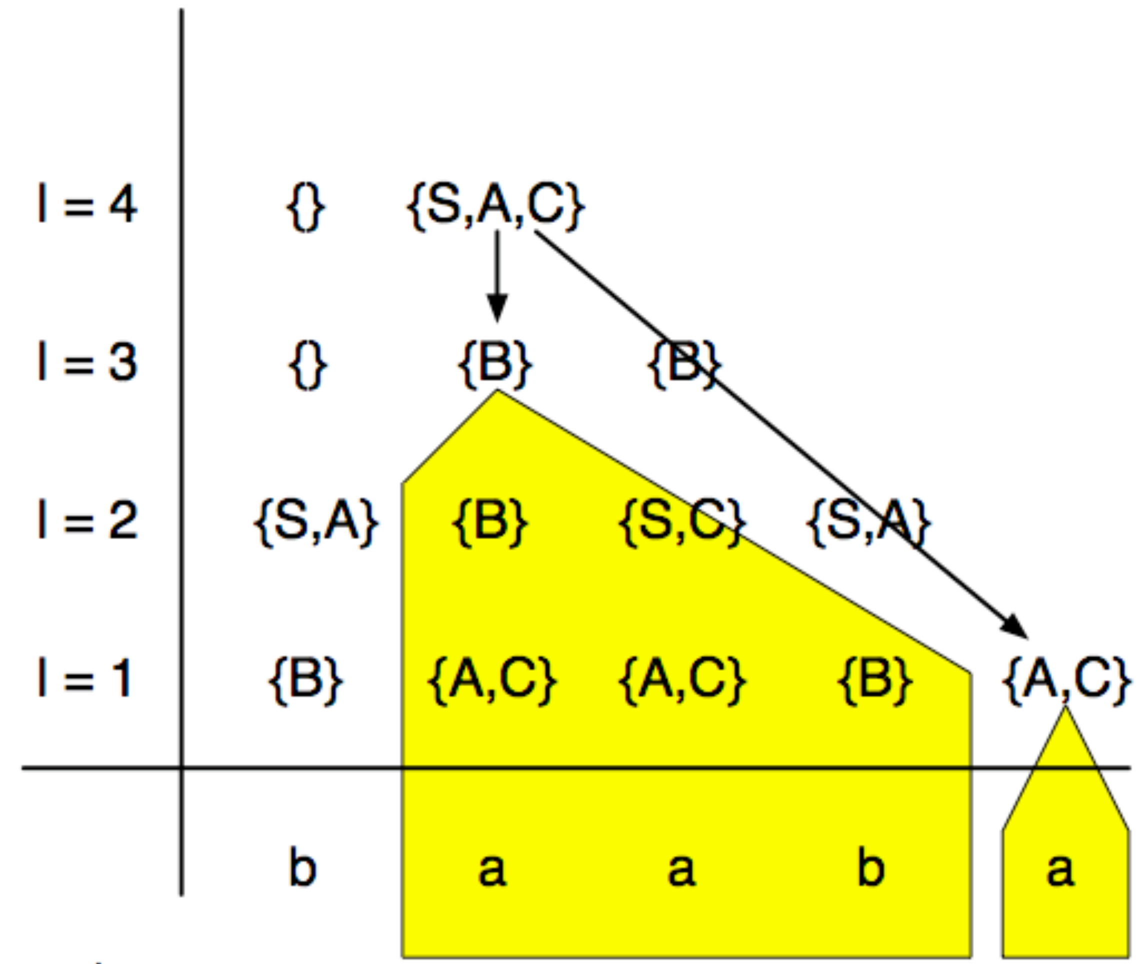
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \to \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3. Für jede Variable v
4. Teste, ob $v \to b$ eine Regel ist, wobei $b = w_i$
5. Falls ja, füge v zu $T(i, i)$ hinzu.
6. Für $\ell = 2$ bis n
7. Für $i = 1$ bis $n - \ell + 1$
8. Setze $j = i + \ell - 1$
9. Für $k = i$ bis $j - 1$
10. Für jede Regel $v \to uz$
11. Falls $u \in T(i, k)$ und $z \in T(k + 1, j)$, füge v zu $T(i, j)$ hinzu.
12. Falls S in $T(1, n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = { **S → AB, S → BC,**
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

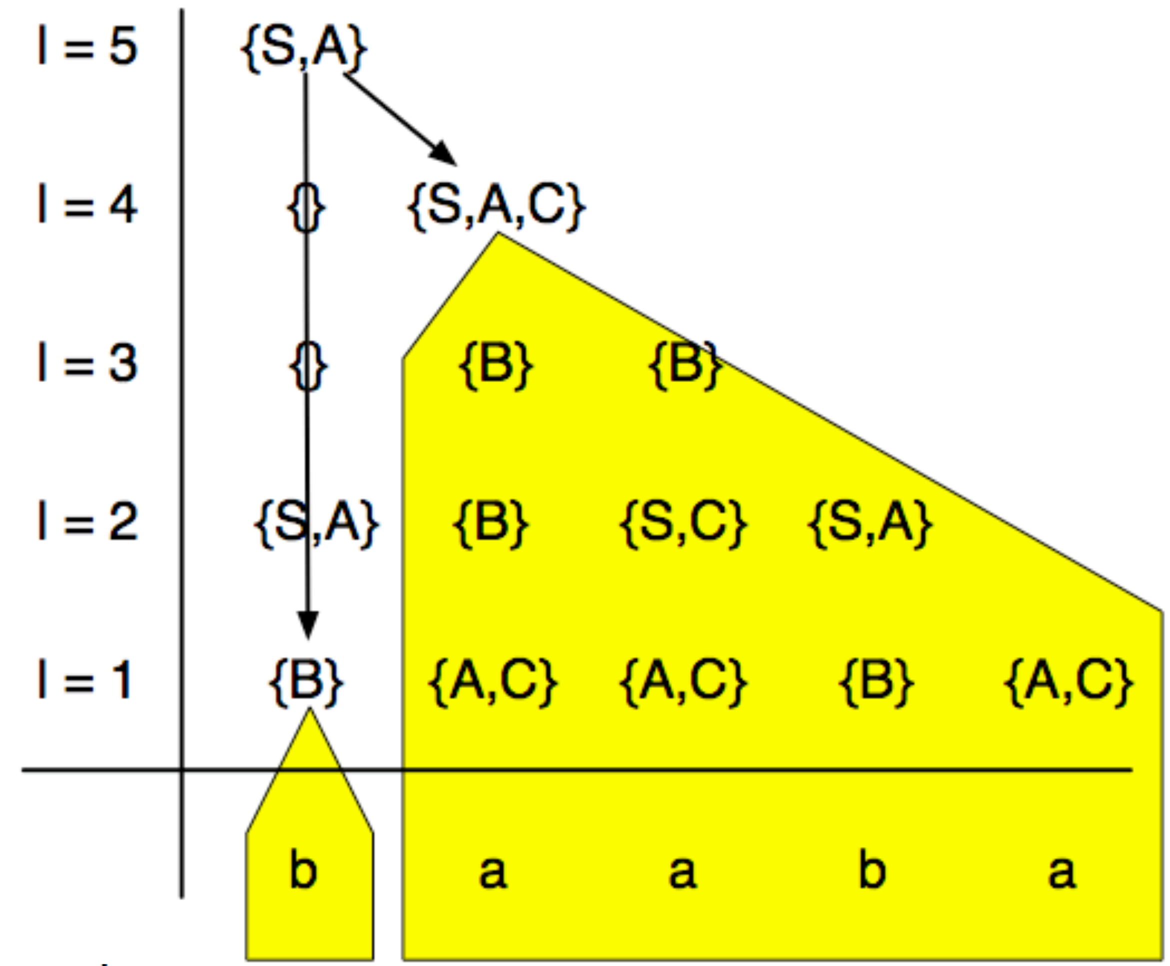
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \rightarrow \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3. Für jede Variable v
4. Teste, ob $v \rightarrow b$ eine Regel ist, wobei $b = w_i$
5. Falls ja, füge v zu $T(i,i)$ hinzu.
6. Für $\ell = 2$ bis n
7. Für $i = 1$ bis $n - \ell + 1$
8. Setze $j = i + \ell - 1$
9. Für $k = i$ bis $j - 1$
10. Für jede Regel $v \rightarrow uz$
11. Falls $u \in T(i,k)$ und $z \in T(k+1,j)$, füge v zu $T(i,j)$ hinzu.
12. Falls S in $T(1,n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = { S → AB, S → BC,
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

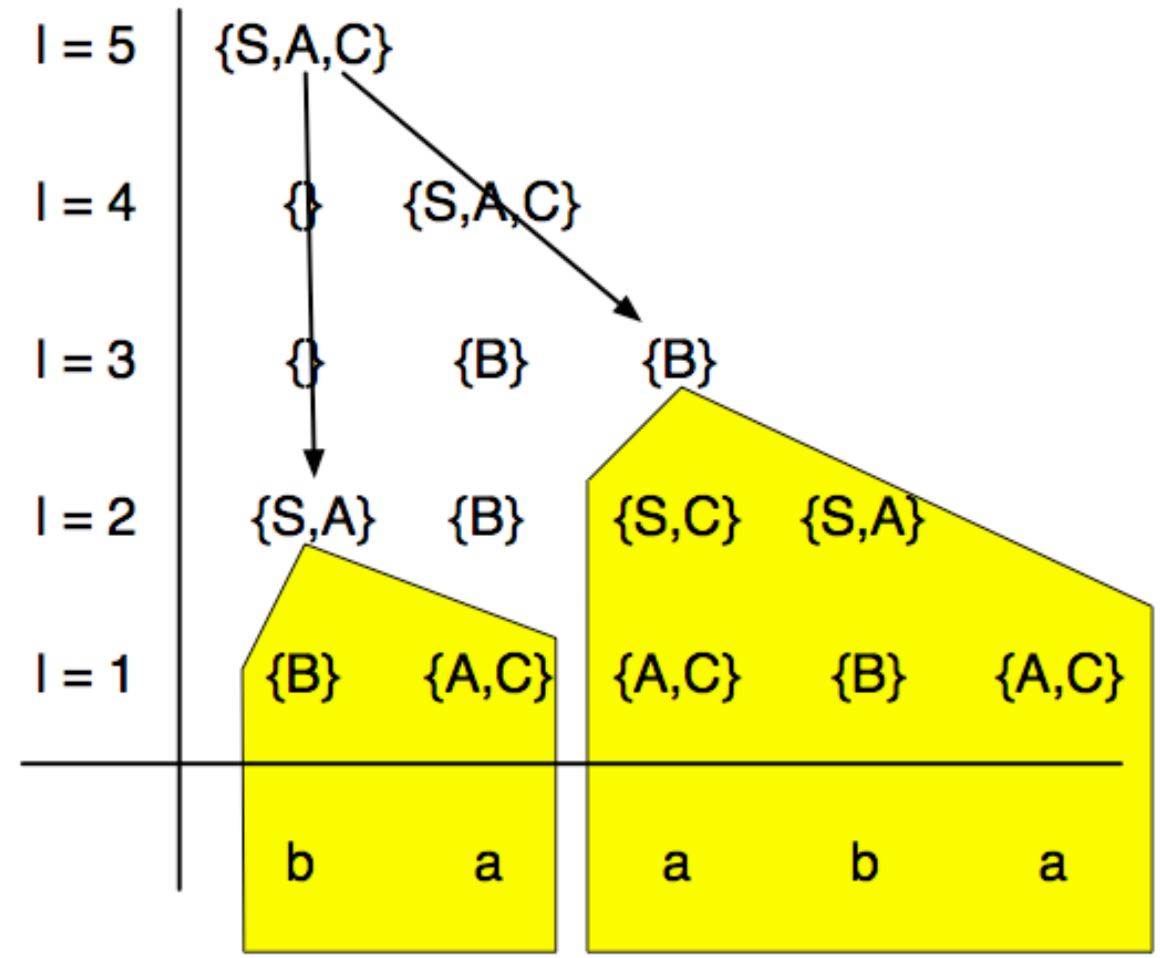
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \to \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3. Für jede Variable v
4. Teste, ob $v \to b$ eine Regel ist, wobei $b = w_i$
5. Falls ja, füge v zu $T(i,i)$ hinzu.
6. Für $\ell = 2$ bis n
7. Für $i = 1$ bis $n - \ell + 1$
8. Setze $j = i + \ell - 1$
9. Für $k = i$ bis $j - 1$
10. Für jede Regel $v \to uz$
11. Falls $u \in T(i,k)$ und $z \in T(k+1,j)$, füge v zu $T(i,j)$ hinzu.
12. Falls S in $T(1,n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = { **S → AB, S → BC,**
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

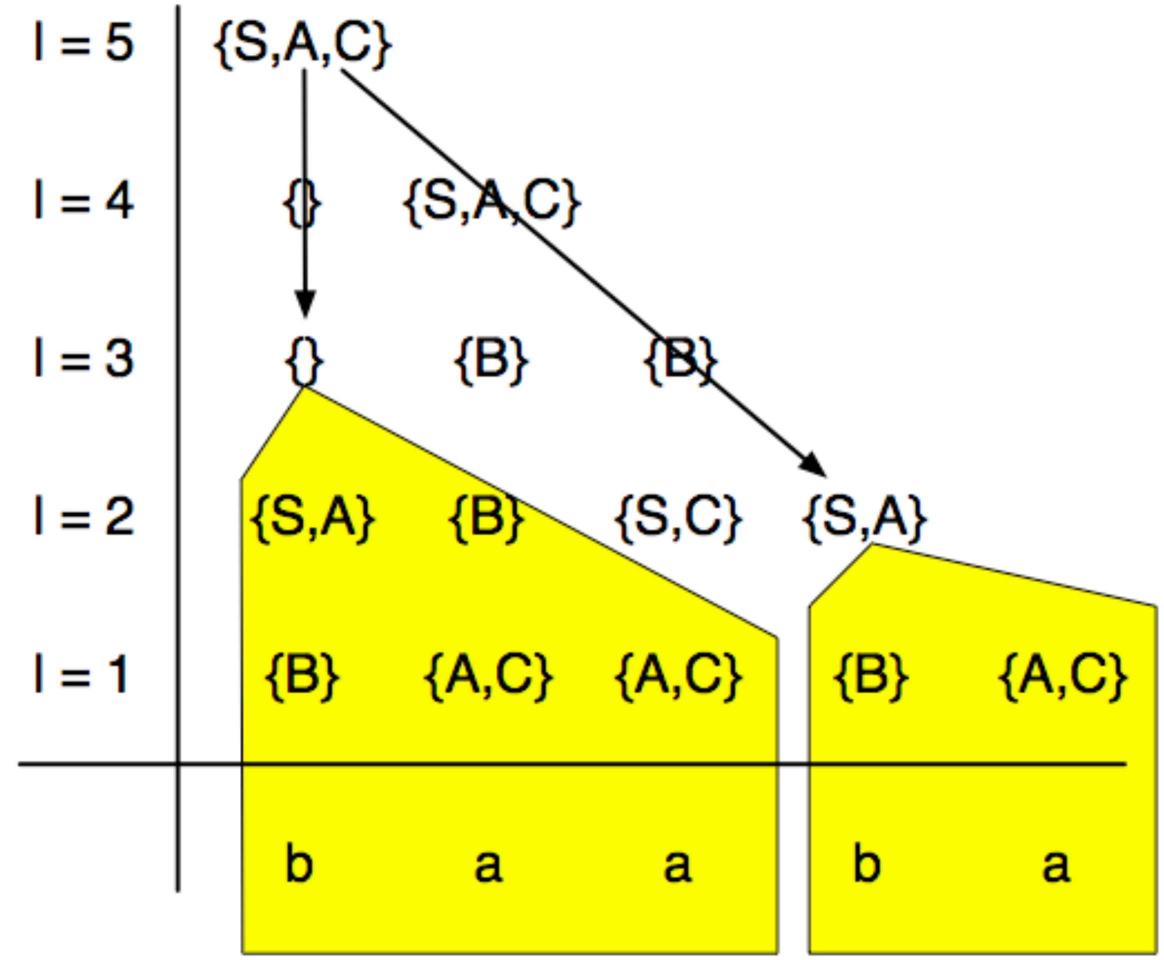
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \rightarrow \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3.   Für jede Variable v
4.   Teste, ob $v \rightarrow b$ eine Regel ist, wobei $b = w_i$
5.   Falls ja, füge v zu $T(i, i)$ hinzu.
6. Für $\ell = 2$ bis n
7.   Für $i = 1$ bis $n - \ell + 1$
8.   Setze $j = i + \ell - 1$
9.   Für $k = i$ bis $j - 1$
10.   Für jede Regel $v \rightarrow uz$
11.   Falls $u \in T(i, k)$ und $z \in T(k + 1, j)$, füge v zu $T(i, j)$ hinzu.
12. Falls S in $T(1, n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = {
S → AB, S → BC,
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

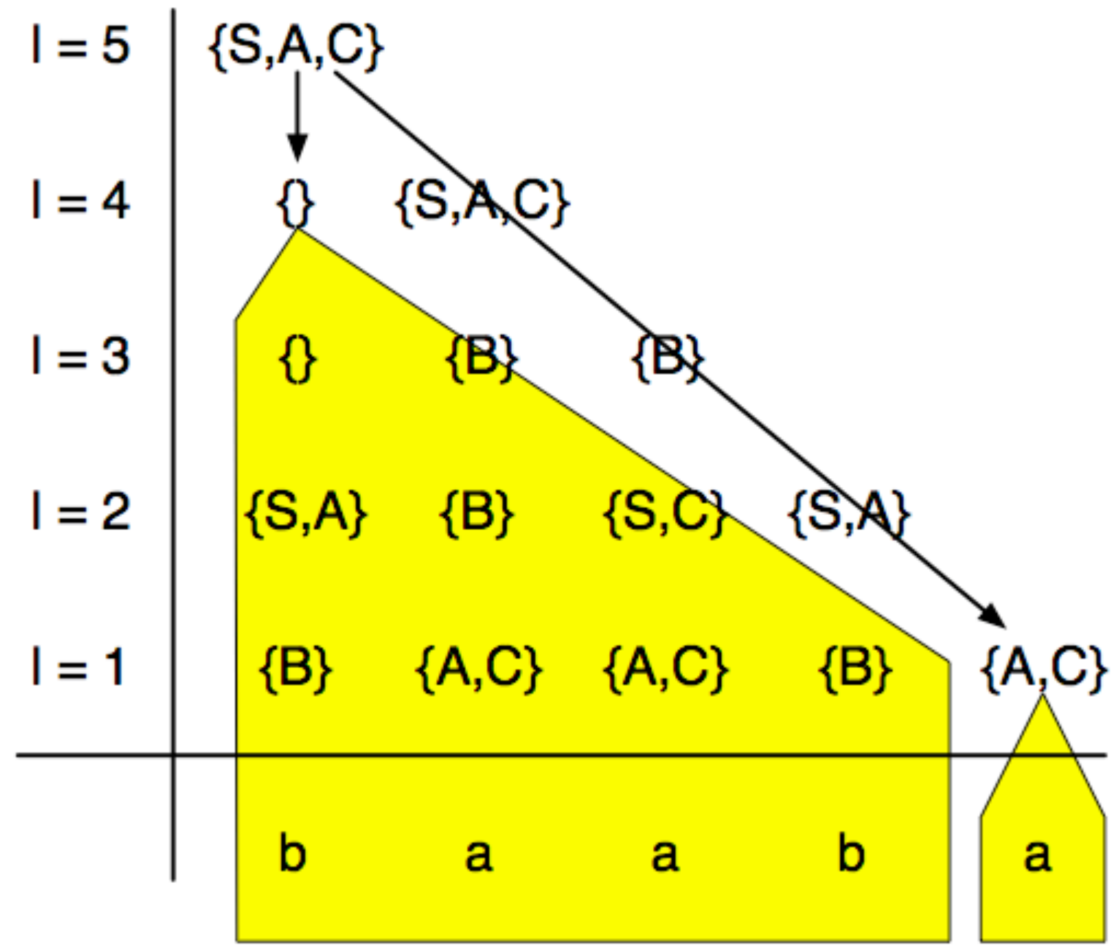
Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \rightarrow \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3. Für jede Variable v
4. Teste, ob $v \rightarrow b$ eine Regel ist, wobei $b = w_i$
5. Falls ja, füge v zu $T(i, i)$ hinzu.
6. Für $\ell = 2$ bis n
7. Für $i = 1$ bis $n - \ell + 1$
8. Setze $j = i + \ell - 1$
9. Für $k = i$ bis $j - 1$
10. Für jede Regel $v \rightarrow uz$
11. Falls $u \in T(i, k)$ und $z \in T(k + 1, j)$, füge v zu $T(i, j)$ hinzu.
12. Falls S in $T(1, n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex



G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = {
S → AB, S → BC,
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

Cocke-Younger-Kasami-Algorithmus
Beispiel

CYK-Algorithmus

Bei Eingabe $w = w_1 \cdots w_n$:

1. Falls $w = \epsilon$ und $S \to \epsilon$ eine Regel ist, akzeptiere.
2. Für $i = 1$ bis n
3. Für jede Variable v
4. Teste, ob $v \to b$ eine Regel ist, wobei $b = w_i$
5. Falls ja, füge v zu $T(i,i)$ hinzu.
6. Für $\ell = 2$ bis n
7. Für $i = 1$ bis $n - \ell + 1$
8. Setze $j = i + \ell - 1$
9. Für $k = i$ bis $j - 1$
10. Für jede Regel $v \to uz$
11. Falls $u \in T(i,k)$ und $z \in T(k+1,j)$, füge v zu $T(i,j)$ hinzu.
12. Falls S in $T(1,n)$ enthalten ist, akzeptiere. Sonst lehne ab.

l = Länge
i = Startindex
j = Schlussindex

l = 5	{S,A,C}				
l = 4	{}	{S,A,C}			
l = 3	{}	{B}	{B}		
l = 2	{S,A}	{B}	{S,C}	{S,A}	
l = 1	{B}	{A,C}	{A,C}	{B}	{A,C}
	b	a	a	b	a

G = (V , Σ, S, P), wobei V = {S, A, B, C } und Σ = {a, b}.

P = { S → AB, S → BC,
A → BA
B → CC,
C → AB,
A → a, B → b, C → a }

baaba ist in der Sprache

Formale Sprachen und

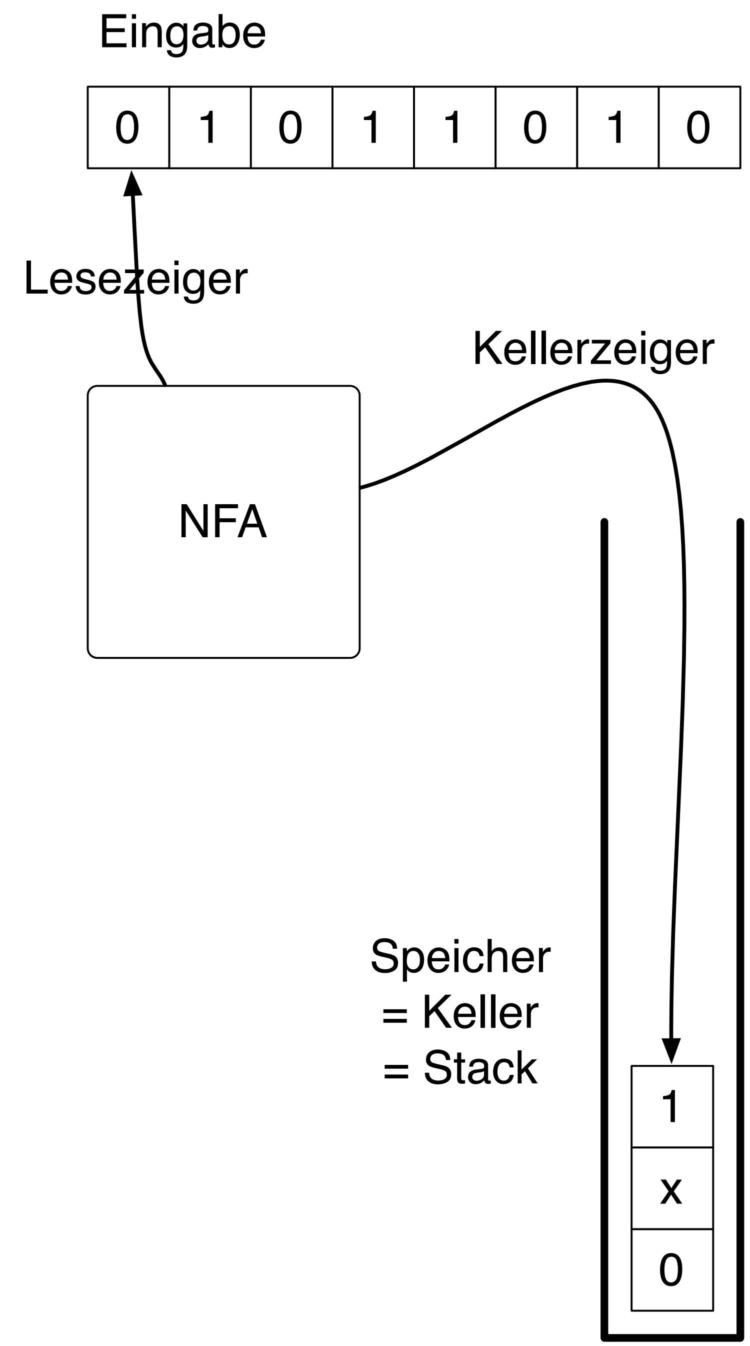
Endliche Automaten

Der Kellerautomat

67

Prinzip des Kellerautomats
Push-Down-Automaton (PDA)

- **Ein Kellerautomat vereinigt einen**
  - NFA mit einem
  - Keller (Stack)
- **Der Keller kann potenziell beliebig viele Zeichen speichern**
- **Zugriff ist eingeschränkt:**
  - Pop: Auslesen des obersten Zeichens (inklusive Entfernen)
  - Push: Hineinlegen eines Zeichens
- **Auf den Übergängen des NFA wird der Zugriff auf den Keller festgelegt**
  - zusätzlich zum aktuellen Zeichen der Eingabe,
  - die weiterhin von links nach rechts gelesen wird.

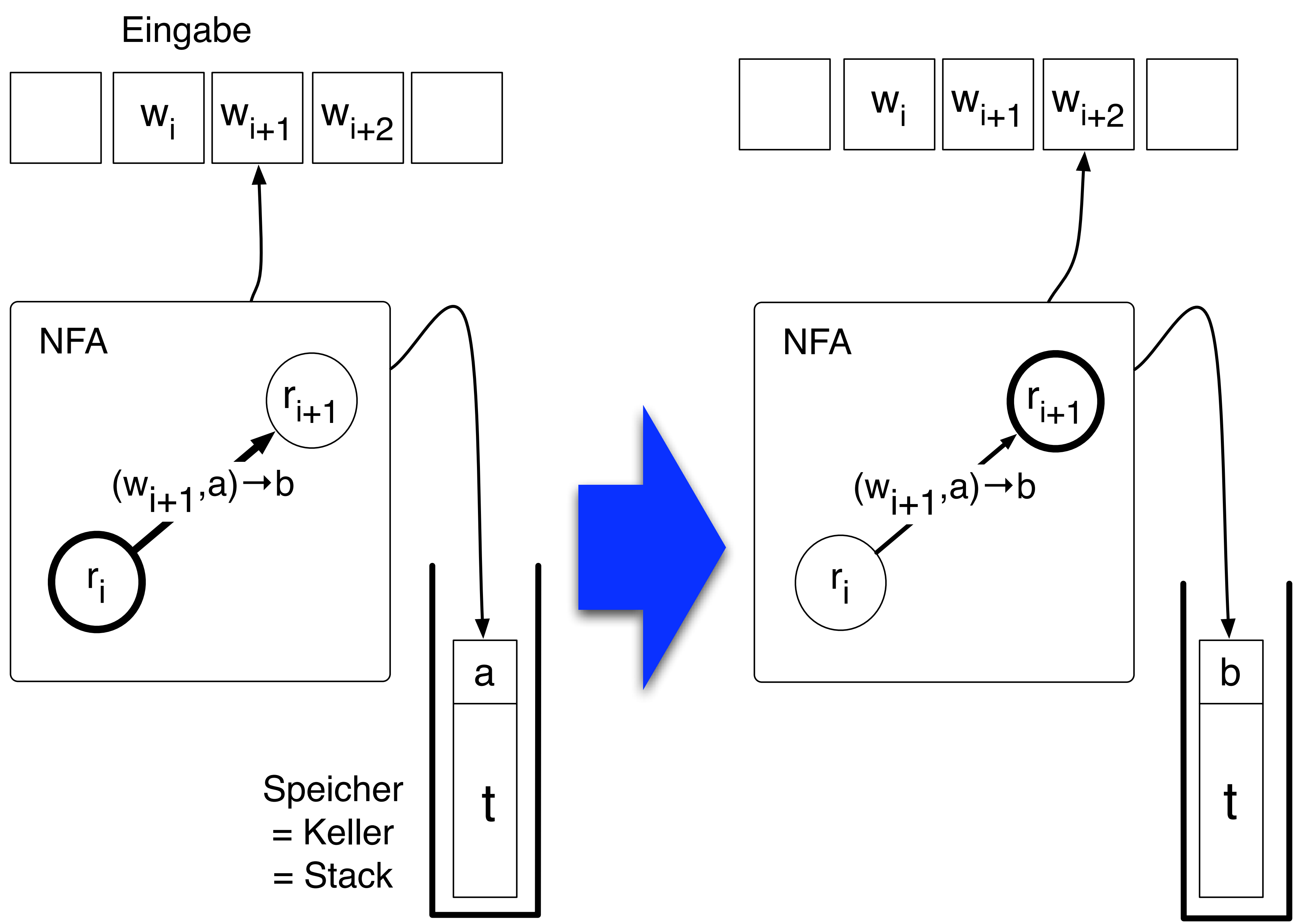


Keller-Automat: Formale Definition

- **Definition**
  - Ein **Kellerautomat** (pushdown automaton - PDA) wird durch ein Sechser-Tupel $(Q, \Sigma, \Gamma, \delta, q_0, F)$, wobei Q, Σ, Γ, F endliche Mengen sind und
  - Q ist die Menge der **Zustände**
  - Σ ist das **Eingabealphabet**
  - Γ ist das **Kelleralphabet**
  - $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to P(Q \times \Gamma_\varepsilon)$ ist die **Übergangsfunktion**
  - q_0 ist der **Startzustand**
  - $F \subseteq Q$ ist die Menge der **akzeptierenden Zustände**

- **Ein PDA akzeptiert die Eingabe w,**
  - wenn es eine Darstellung $w = w_1 w_2 \ldots w_m$ mit $w_i \in \Sigma_\varepsilon$ gibt
  - wenn es eine Zustandsfolge $q = r_0 r_1 \ldots r_m$ mit $r_i \in Q$ gibt
  - wenn es Zeichenketten $s_0, s_1, \ldots, s_m \in \Gamma_\varepsilon^*$ gibt, so dass
  - $r_0 = q_0$ und $s_0 = \varepsilon$
  - Startzustand mit leeren Keller
    - Für $i = 0, \ldots, m-1$ gilt:
      - $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$, wobei
      - $s_i = at$ und $s_{i+1} = bt$ für passende $a, b \in \Gamma_\varepsilon$, $t \in \Gamma_\varepsilon^*$
    - Übergang mit Kellerverhalten:
      - Lies a, Schreibe b
  - $r_m \in F$: Ein akzeptierender Zustand erscheint als Endzustand

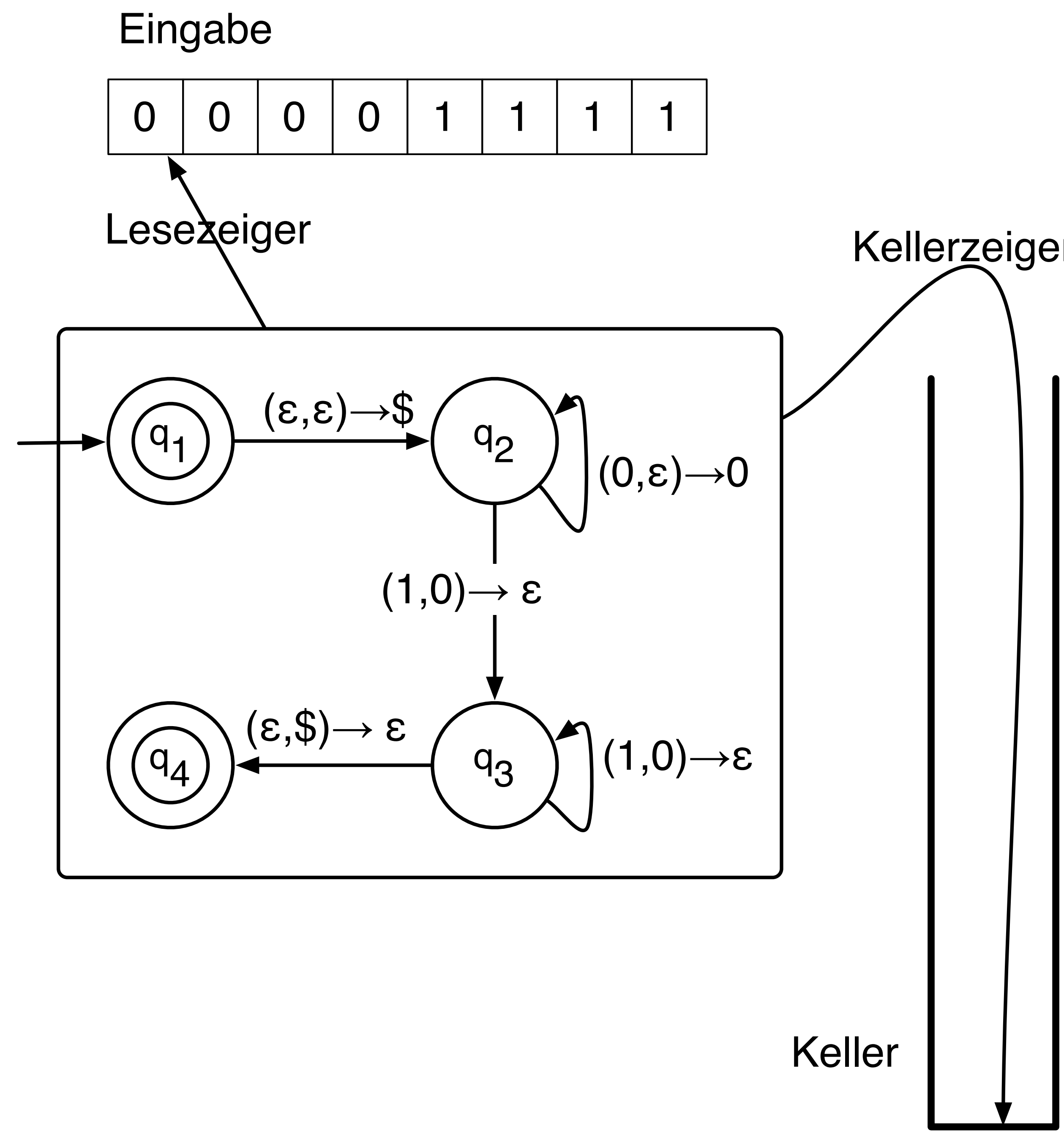
Übergang



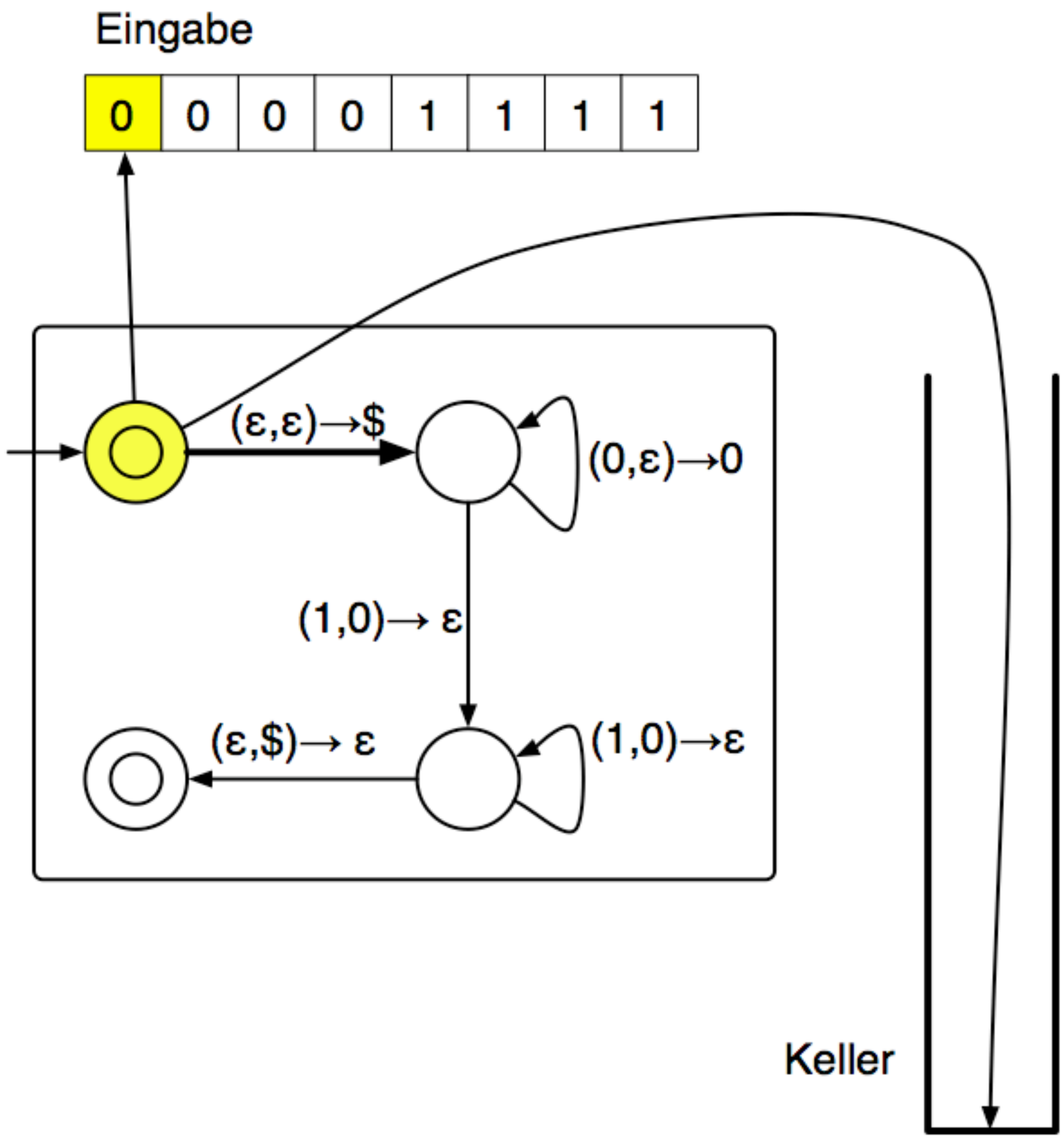
Eingabe
w_i
w_{i+1}
w_{i+2}
NFA
r_{i+1}
$(w_{i+1},a)\rightarrow b$
r_i
a
t
Speicher
= Keller
= Stack
w_i
w_{i+1}
w_{i+2}
NFA
r_{i+1}
$(w_{i+1},a)\rightarrow b$
r_i
b
t

Keller-Automat: Beispielberechnung

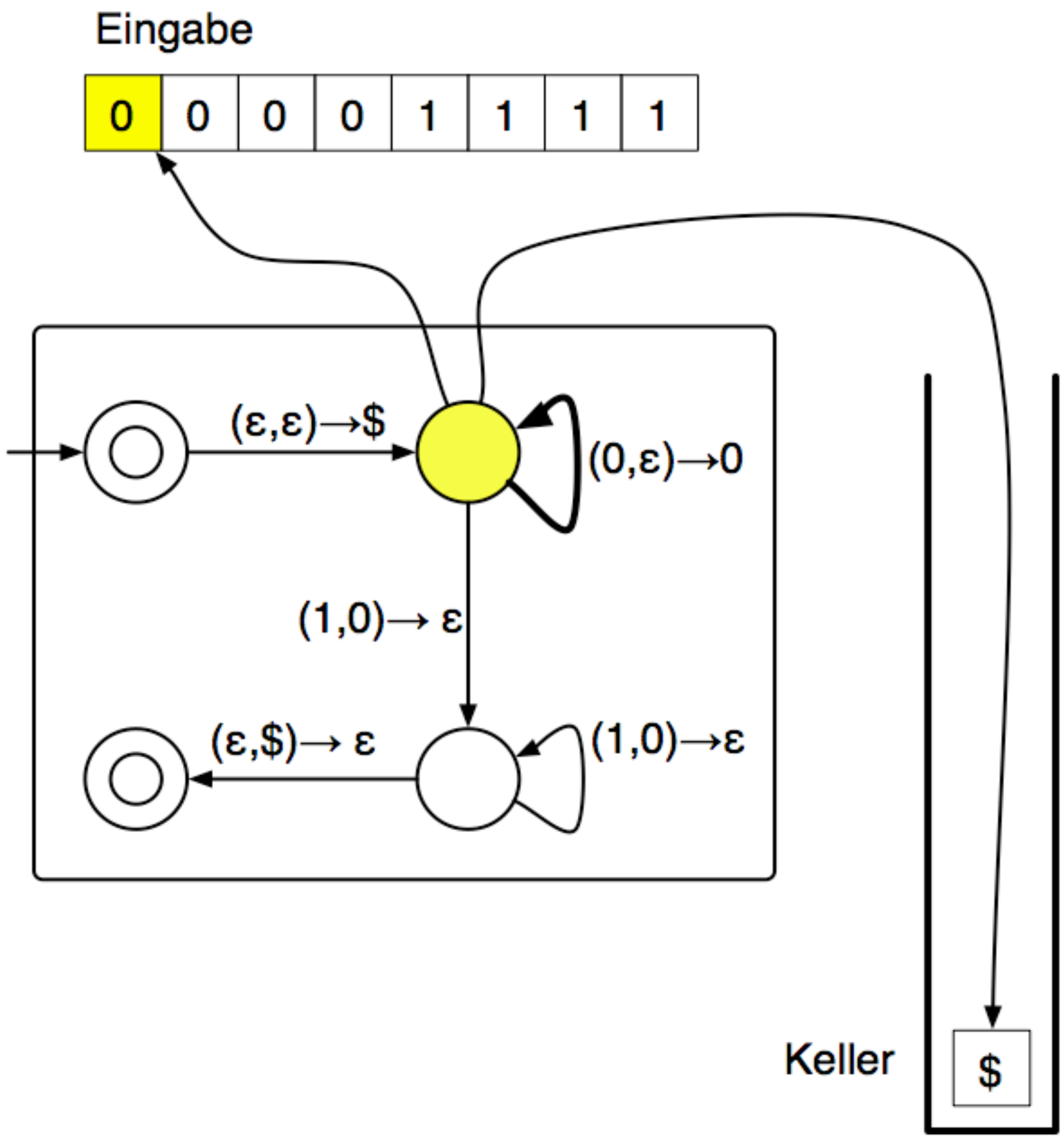
- **Der PDA $M = (Q, \Sigma, \Gamma, \delta, q_1, F)$ akzeptiert die Sprache $\{0^n 1^n \mid n \geq 0\}$**
  - $Q = \{q_1, q_2, q_3, q_4\}$
  - $\Sigma = \{0,1\}$
  - $\Gamma = \{0,\$\}$
  - $F = \{q_1, q_4\}$
  - $\delta(q_1,\varepsilon,\varepsilon) = \{(q_2,\$)\}$ q_1: Push \$, Gehe zu q_2
  - $\delta(q_2,0,\varepsilon) = \{(q_2,0)\}$ q_2: Falls Lese = 0: Push 0; Gehe zu q_2
  - $\delta(q_2,1,0) = \{(q_3,\varepsilon)\}$ q_2: Falls Lese=1 und Pop=0 gehe zu q_3
  - $\delta(q_3,1,0) = \{(q_3,\varepsilon)\}$ q_3: Falls Lese=1 gehe zu q_3
  - $\delta(q_3,\varepsilon,\$) = \{(q_4,\varepsilon)\}$ q_3: Falls Pop=\$ gehe zu q_4
  - $\delta(q,a) = \{\}$, für alle anderen $q \in Q, a \in \Sigma_\varepsilon$



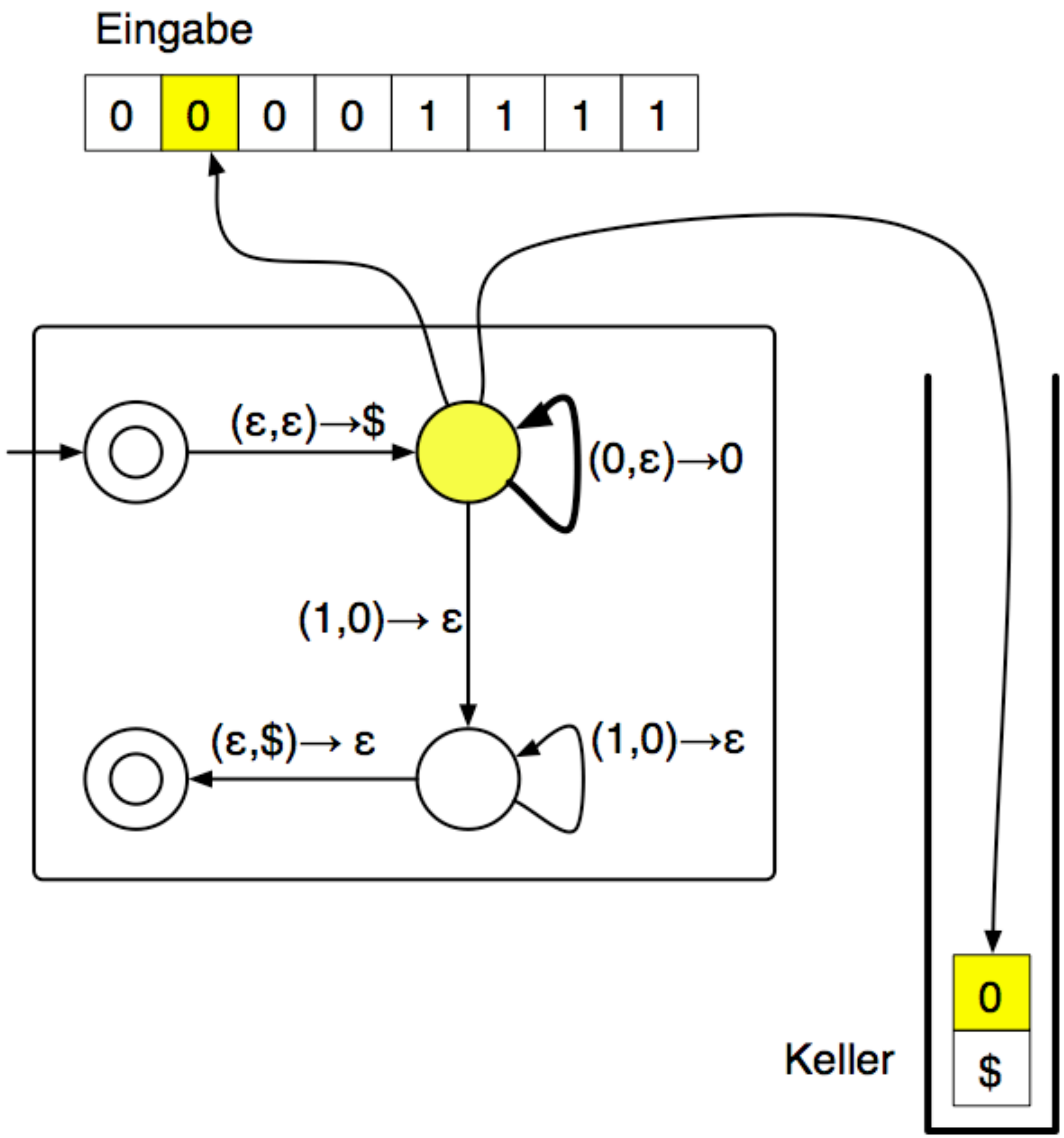
Keller-Automat: Beispielberechnung



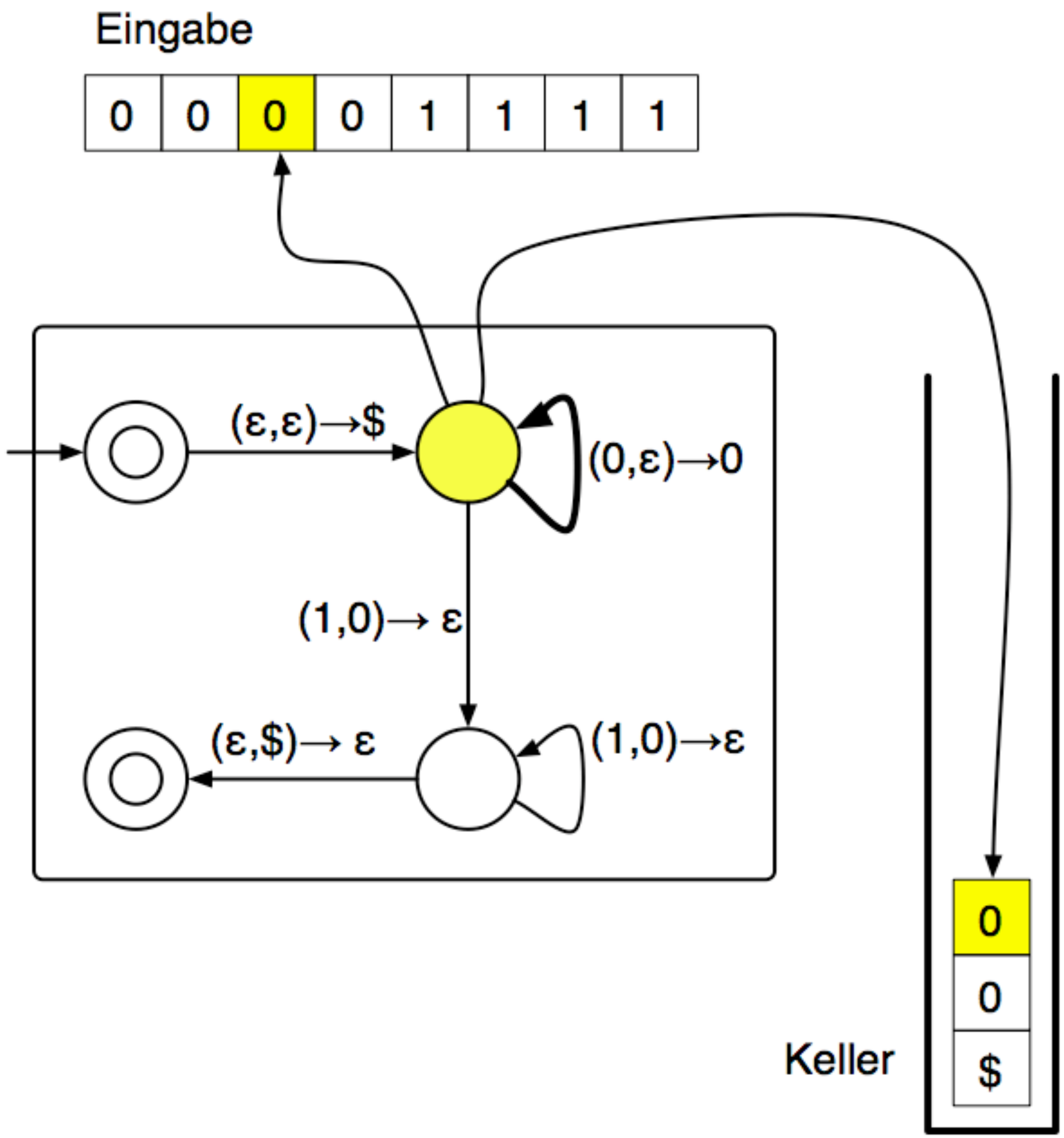
Keller-Automat: Beispielberechnung



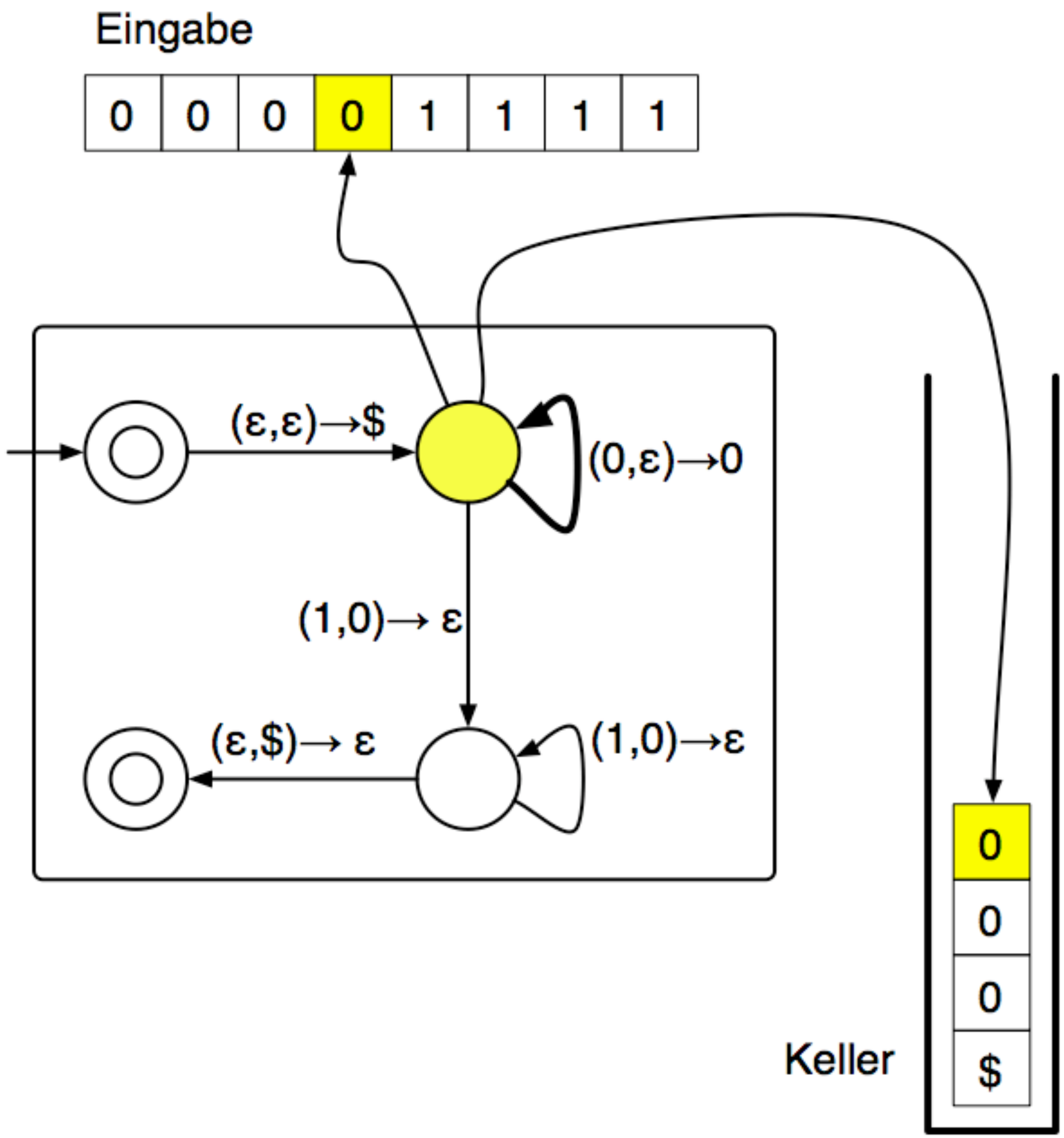
Keller-Automat: Beispielberechnung



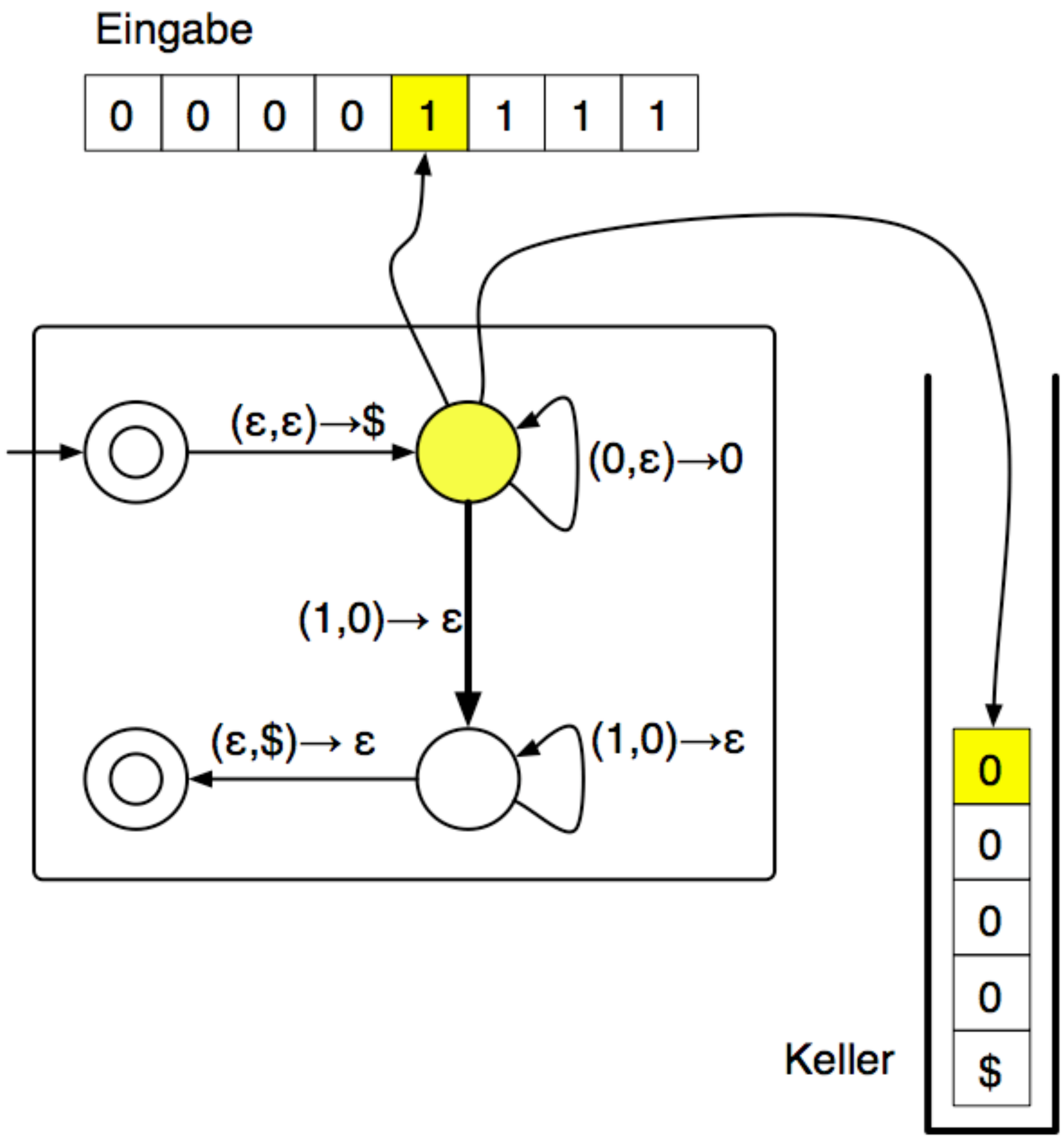
Keller-Automat: Beispielberechnung



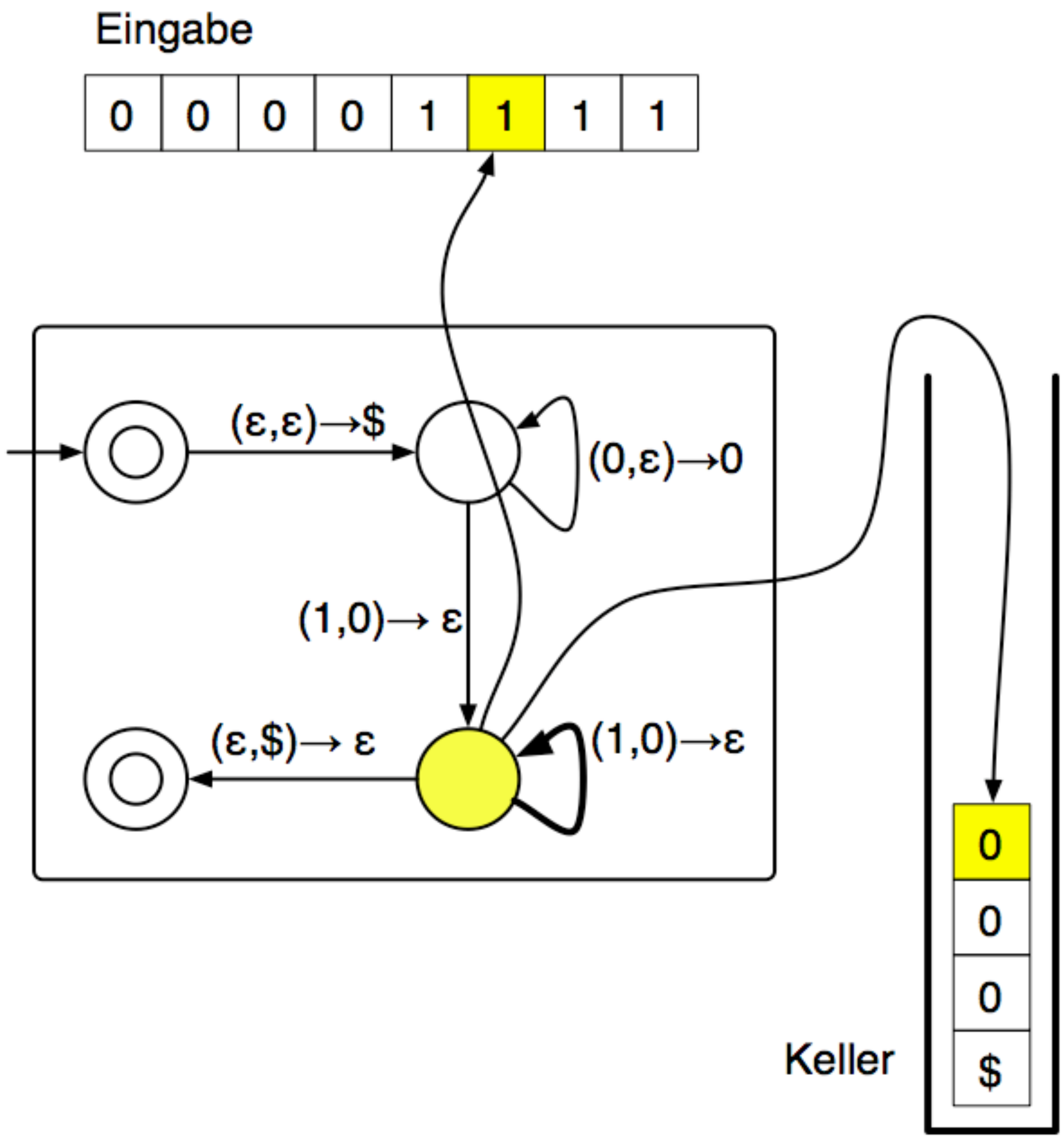
Keller-Automat: Beispielberechnung



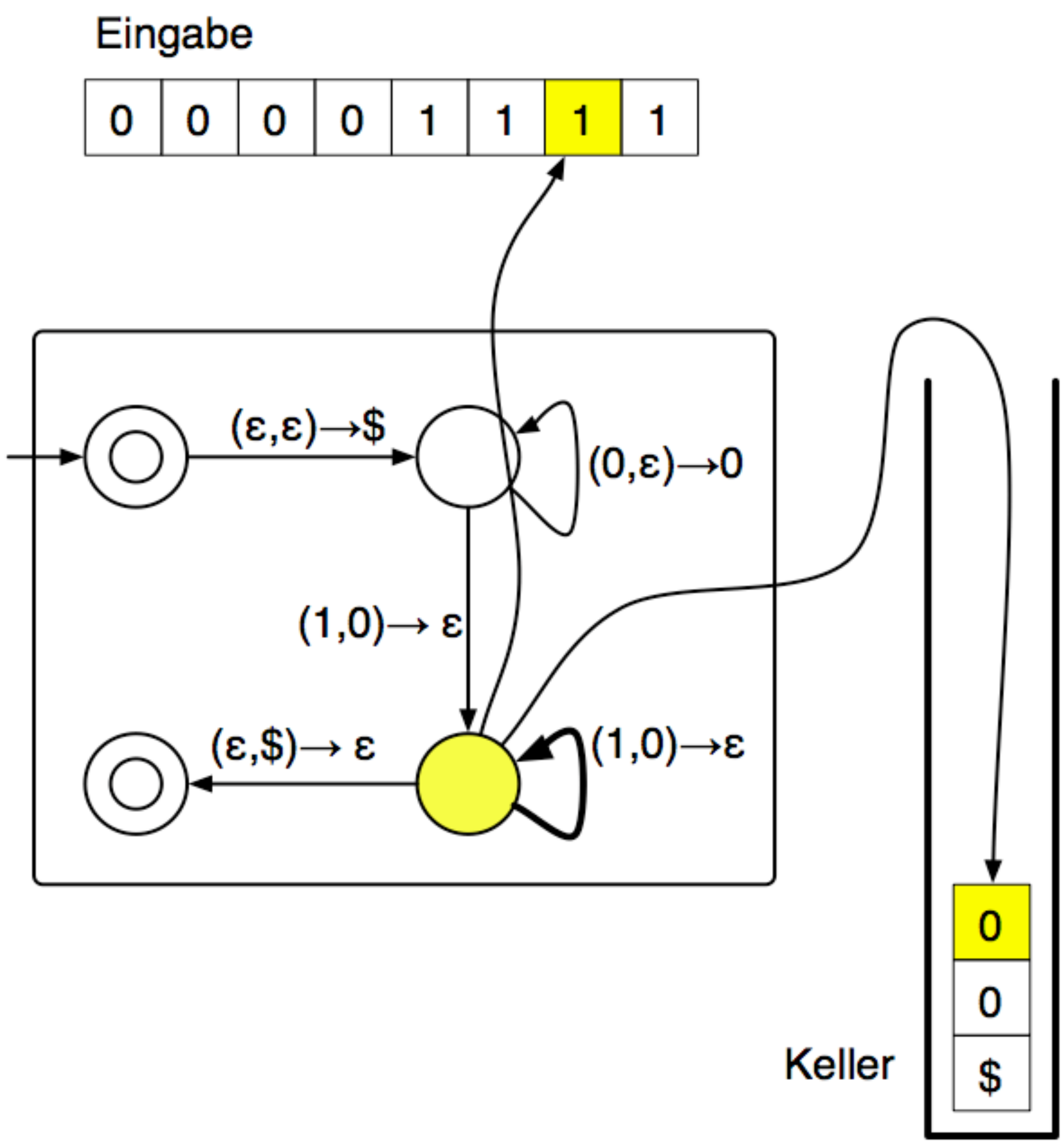
Keller-Automat: Beispielberechnung



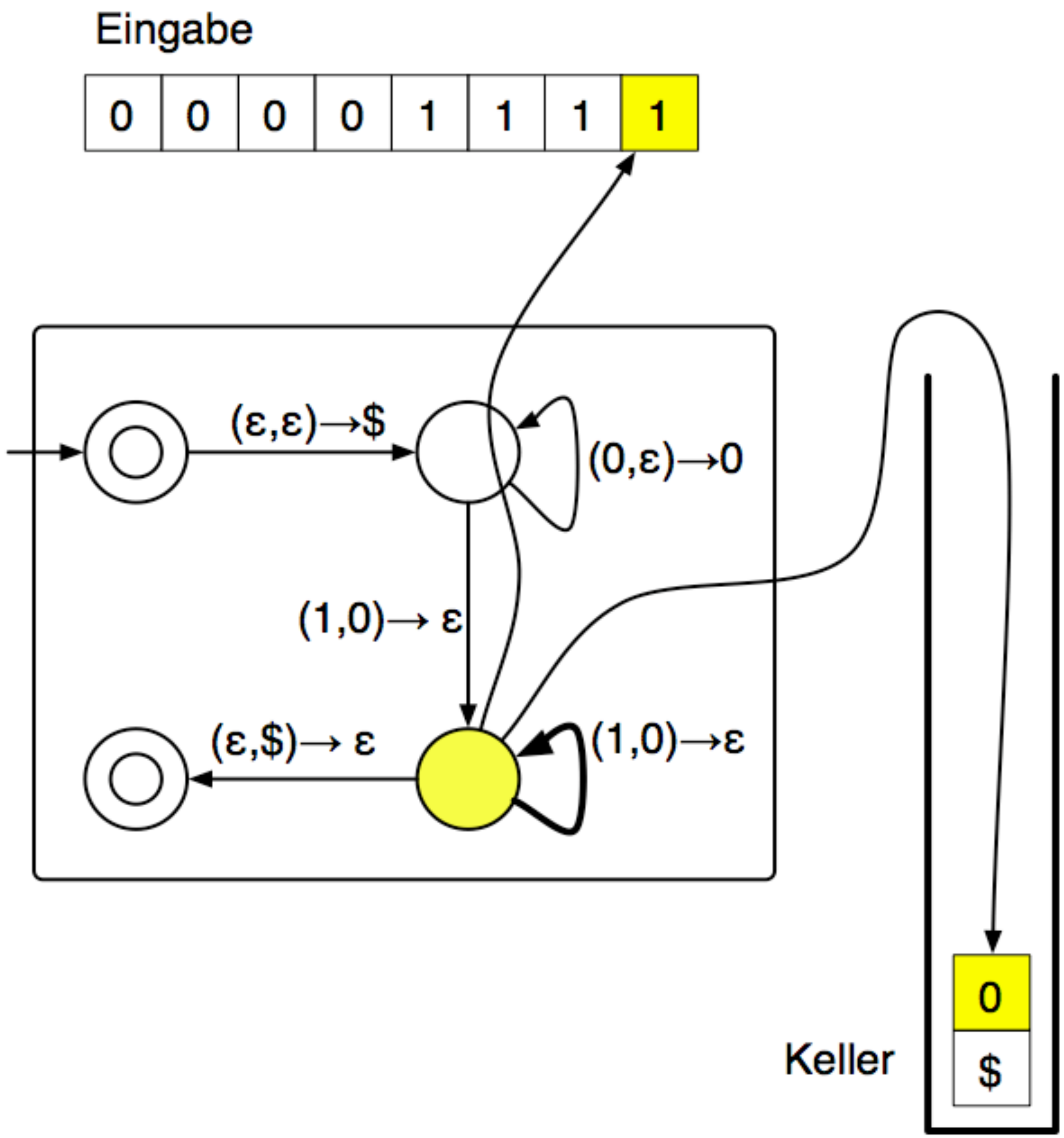
Keller-Automat: Beispielberechnung



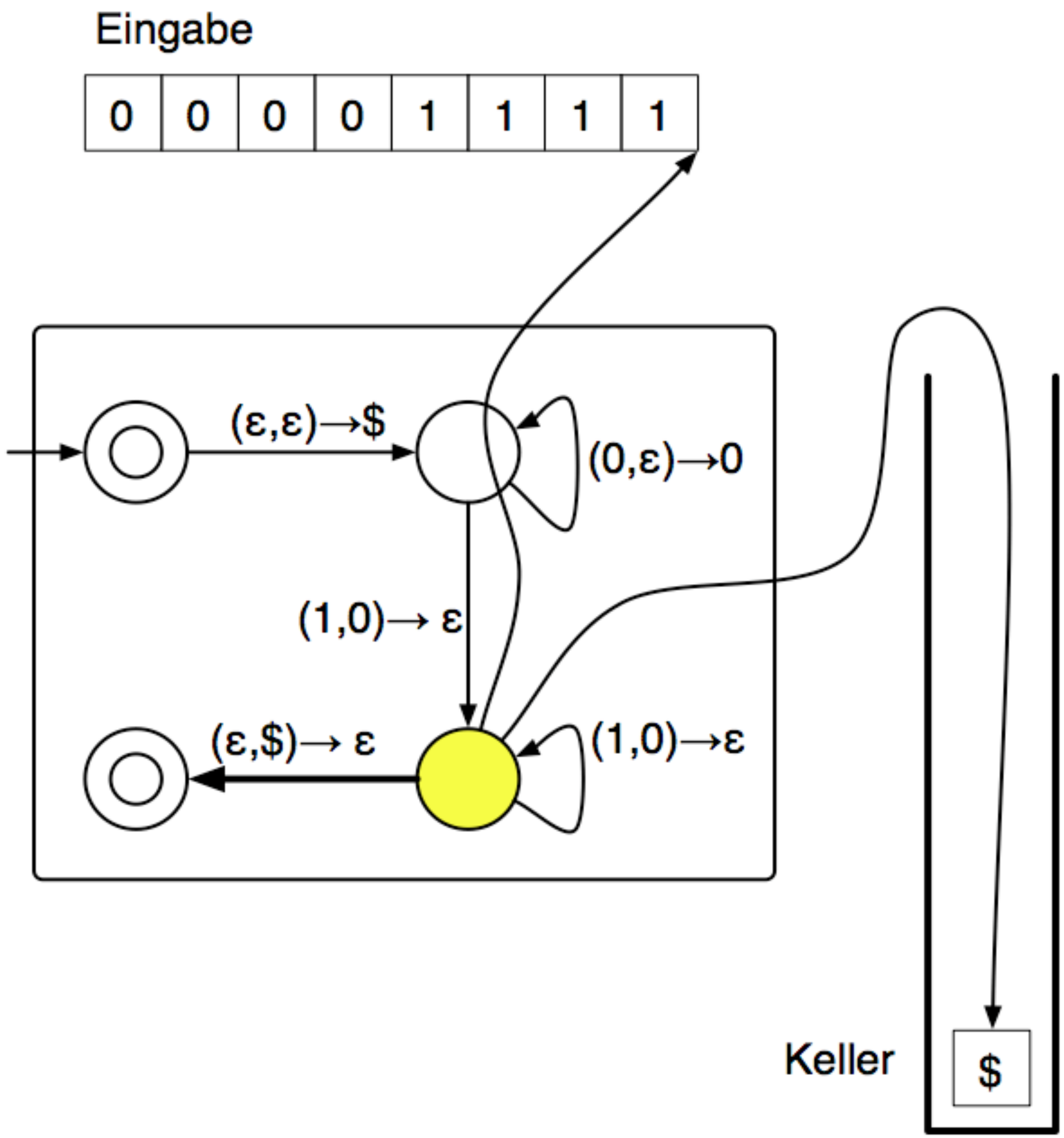
Keller-Automat: Beispielberechnung



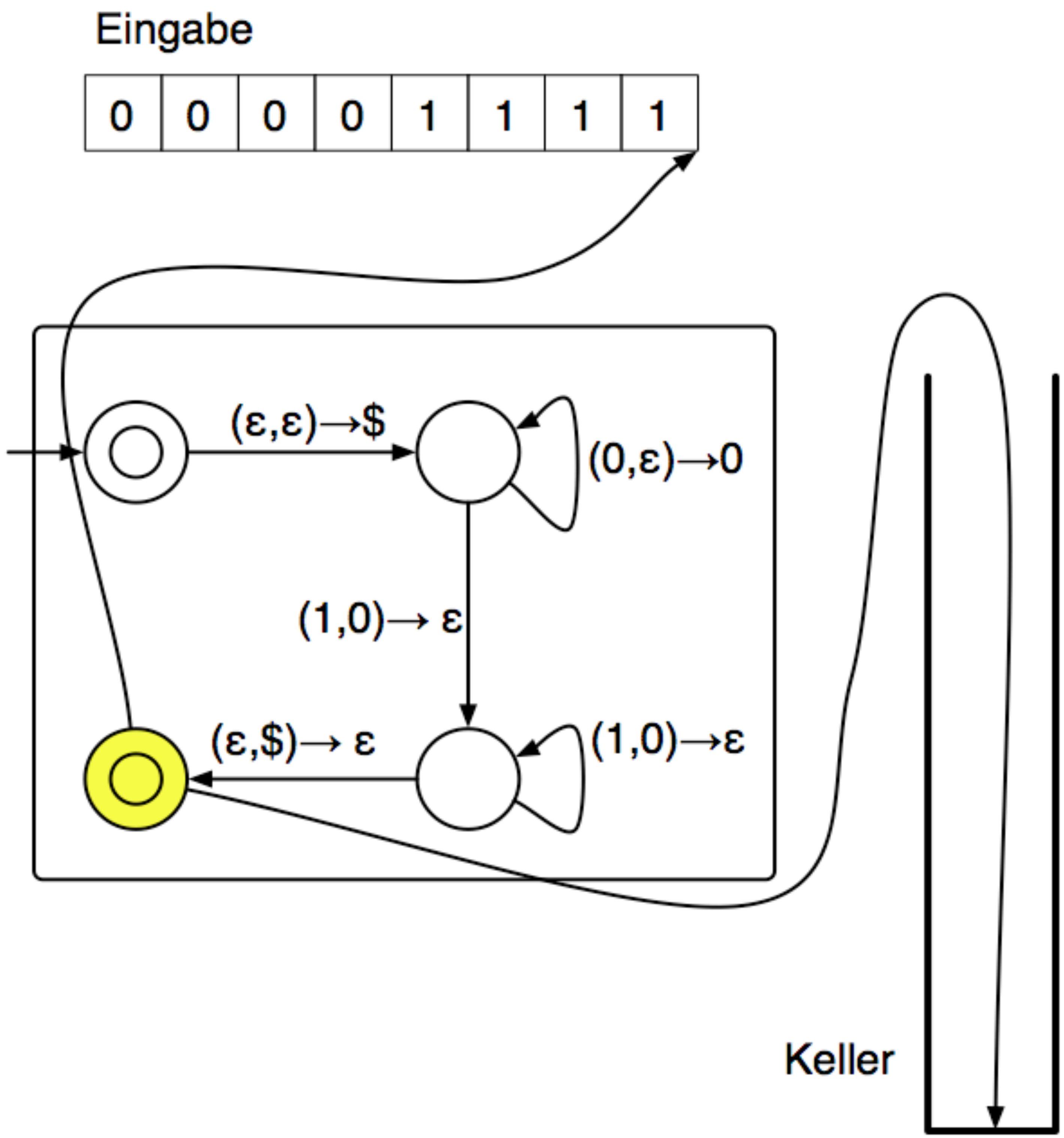
Keller-Automat: Beispielberechnung



Keller-Automat: Beispielberechnung



Keller-Automat: Beispielberechnung
Automat akzeptiert!



Formale Sprachen und
Endliche Automaten

Kellerautomaten erkennen genau CFL

83

Keller-Automaten beschreiben genau die kontextfreien Sprachen

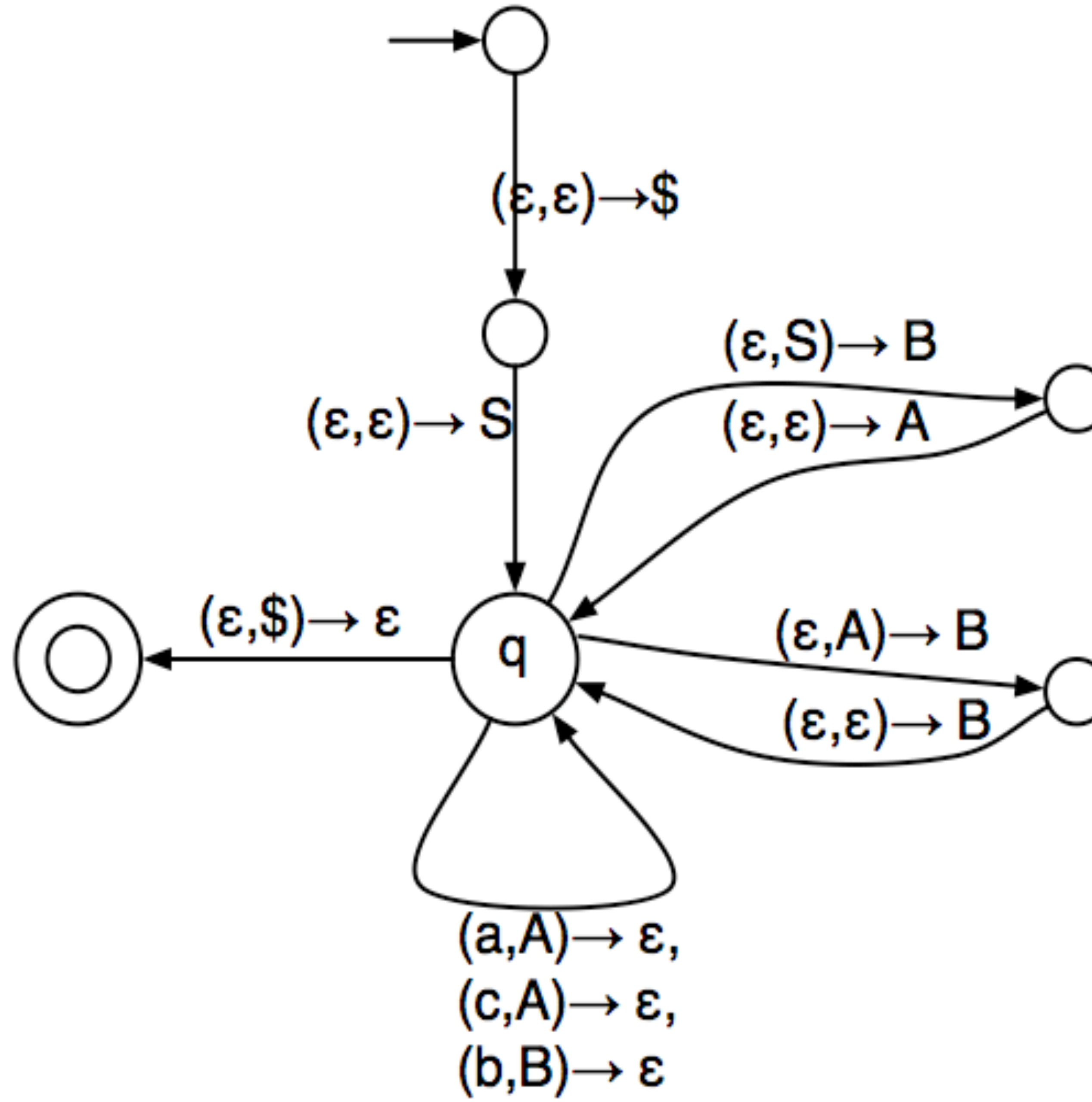
- **Theorem 6.1**
  - Eine Sprache ist genau dann kontextfrei, wenn ein Kellerautomat sie erkennt
- **Lemma 6.1**
  - Ist eine Sprache kontextfrei, dann erkennt Sie ein Kellerautomat.
- **Beweisidee**
  - Die Mehrdeutigkeit der kontextfreien Sprache wird durch den Nichtdeterminismus des PDA gelöst
  - Die Ableitung der Worte steht im Keller
  - Im Keller werden die möglichen Übergänge der Ableitung geraten
  - Oben im Keller = Links, Unten im Keller = Rechts.
  - Sobald oben im Keller ein Terminal steht, wird es mit der Eingabe verglichen und jeweils gestrichen
  - Sobald oben im Keller eine Variable steht, wird eine mögliche Ersetzung im Keller durchgeführt.

Jede kontextfreie Sprache kann von einem PDA erkannt werden

- **Lemma 6.1**
  - Ist eine Sprache kontextfrei, dann gibt es einen Kellerautomat für sie.
- **Konstruktion des Kellerautomaten**
  - Wir können davon ausgehen, dass die Sprache als kontextfreie Grammatik $G=(V,\Sigma,R,S)$ in Chomsky-Normalform vorliegt.
  - Ist $S \rightarrow \varepsilon$ in der Grammatik so gibt es einen ist der Startzustand des PDA akzeptierend
  - Zuerst wird der Stack mit den Symbolen \$S initialisiert
  - Sei q der Zustand nach der Initialisierung
  - Für jede Regel der Form
    - $A \rightarrow BC$
    - wird ein Übergang von q zu Zustand q[BC] hergestellt, der A vom Keller holt, ε von der Eingabe liest und C auf den Keller legt
    - Der Übergang von q[BC] nach q liest ε von der Eingabe und legt B auf den Keller ab
  - Für jede Regel der Form
    - $A \rightarrow a$
    - wird ein Übergang von q zu q angelegt, der von der Eingabe a liest und A vom Keller holt
  - Eine weitere Regel liest \$ vom Keller und geht in den einzigen akzeptierenden Zustand q[akz]

Beispielkonstruktion

- **Grammatik: G=(V,Σ,R,S) mit**
  - V={S,A,B}
  - Σ={a,b,c}
- **R = { S → AB,**
  A → BB,
  A → a,
  A → c,
  B → b }
- **Beispiel:**
  - S Keller: S$ Eingabe: cb
    ⇒ AB Keller: AB$ Eingabe: cb
    ⇒ cB Keller: B$ Eingabe: b
    ⇒ cb Keller: $ Eingabe: -



PDAs erkennen nur kontextfreie Sprachen

- **Lemma 6.2**
  - Die Sprache, die von einem Kellerautomat erkannt wird, ist kontextfrei
- **Vorbereitung:**
  - Ein PDA kann so modifiziert werden ohne seine Sprache zu verändern, dass die folgenden Eigenschaften gelten:
    1. Es gibt nur einen einzigen akzeptierenden Zustand q_{akz}
    2. Bevor der PDA akzeptiert, leert der PDA seinen Keller
    3. In jedem Übergang wird
       - entweder ein Zeichen vom Keller geholt oder
       - ein Zeichen abgelegt.
       - aber nicht beides zugleich
  - Die Variable A[rs] beschreibt alle Worte, die der PDA abarbeiten kann, wenn er mit einem leeren Keller bei Zustand r beginnt und bei Zustand s wieder mit einem leeren Keller endet

PDAs erkennen nur kontextfreie Sprachen

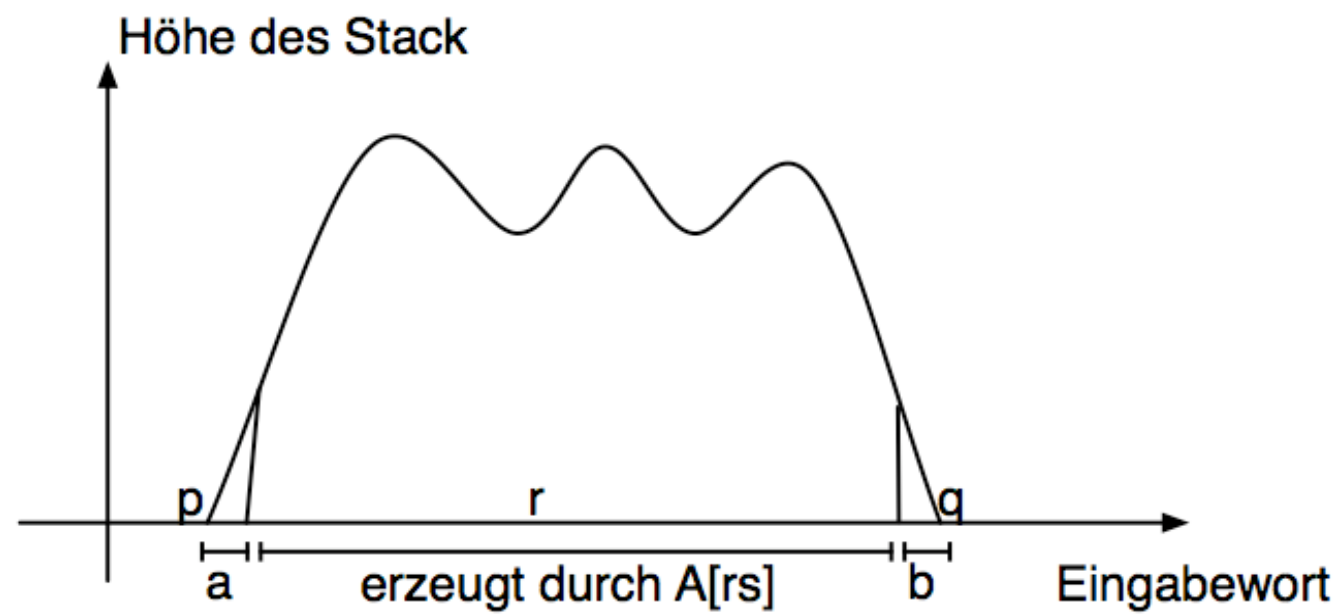
- **Lemma 6.2**
  - Die Sprache, die von einem Kellerautomat erkannt wird, ist kontextfrei
- **Beweis**
  - Für $P = (Q, \Sigma, \Gamma, \delta, q_0, \{q_{akz}\})$ konstruieren wir eine kontextfreie Grammatik G
  - Die Variablen von G sind: $\{A[pq] \mid p,q \in Q\}$
  - Die Startvariable ist $A[q_0 q_{akz}]$

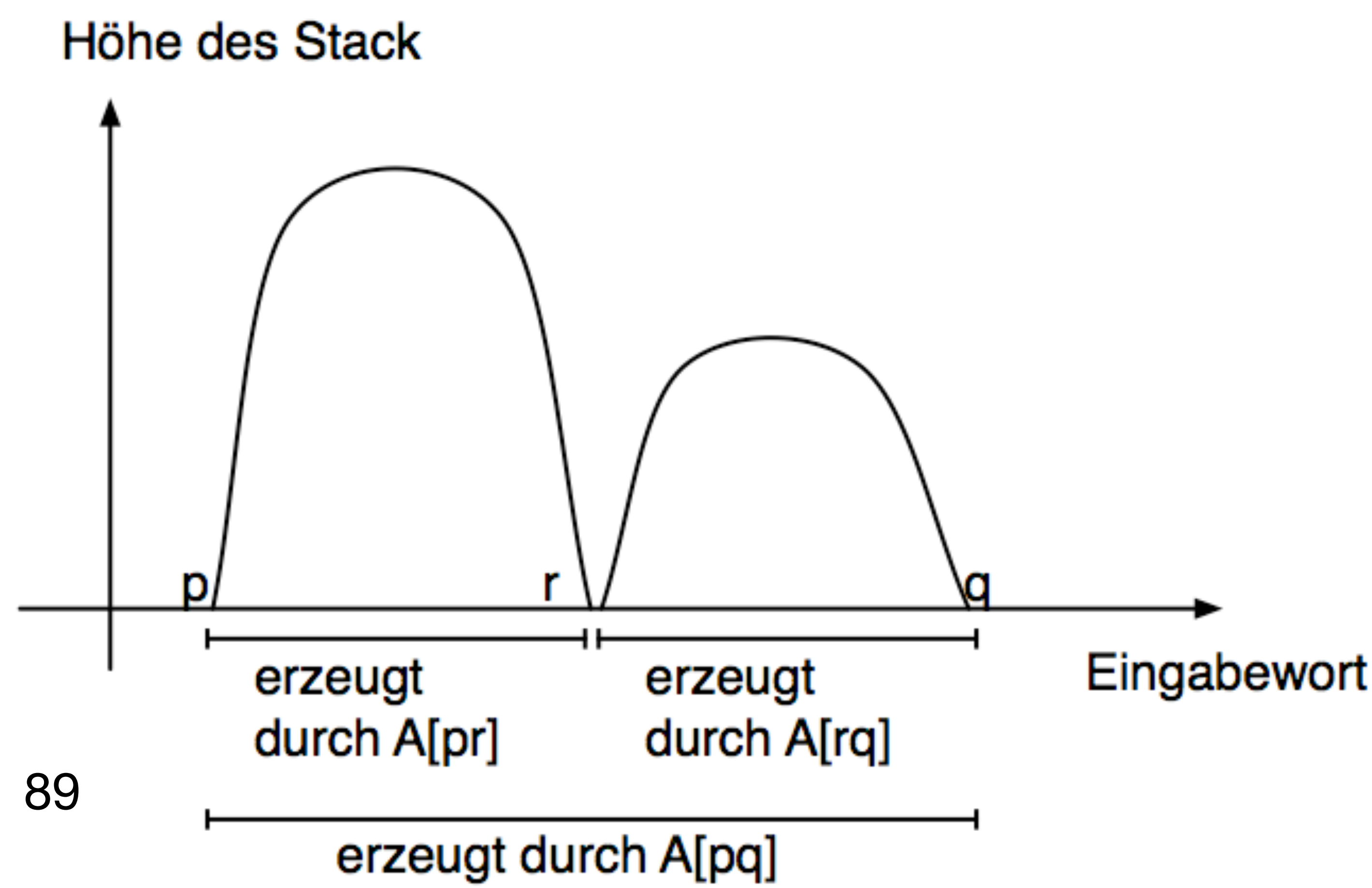
- Die Ersetzungsregeln sind:
  - Für jedes $p,q,r,s \in Q$, $t \in \Gamma$ und $a,b \in \Sigma$
    - falls $\delta(p,a,\varepsilon)$ das Element (r,t) enthält und
    - falls $\delta(s,b,t)$ das Element (q, ε),
      - füge $A[pq] \rightarrow a\ A[rs]\ b$ zu G hinzu
  - Für jedes $p,q,r \in Q$ füge
    - $A[pq] \rightarrow A[pr]\ A[rq]$ zu G hinzu
  - Für jedes $p \in Q$ füge die Regel
    - $A[pp] \rightarrow \varepsilon$ zu G hinzu

PDAs erkennen nur kontextfreie Sprachen
Intuition zur Konstruktion

Die Ersetzungsregeln sind:

- Für jedes p,q,r,s $\in$ Q, t $\in \Gamma$ und a,b $\in \Sigma$
  - falls δ(p,a,ε) das Element (r,t) enthält und
  - falls δ(s,b,t) das Element (q, ε),
  - füge A[pq] $\rightarrow$ aA[rs]b zu G hinzu
- Für jedes p,q,r $\in$ Q füge
  - A[pq] $\rightarrow$ A[pr] A[rq] zu G hinzu





PDAs erkennen nur kontextfreie Sprachen - Korrektheit

- **Ergebnis der Konstruktion:**
  - Wenn A[pq] ein Wort x erzeugt, so kann x den PDA P vom Zustand p in den Zustand q überführen, wobei am Anfang und Ende der Stapel jeweils leer ist
    - Beweis per Induktion über die Anzahl der Ableitungsschritte, die zur Ableitung von x aus der Variablen A[pq] benötigt werden
  - Kann x den PDA P vom Zustand p in den Zustand q überführen, wobei am Anfang und Ende der Stapel jeweils leer ist, so kann x aus A[pq] abgeleitet werden
    - Beweis per Induktion über die Anzahl der Schritte in der Rechnung, die p in q überführen
  - Aus der Startvariable können genau diejenigen Worte $x \in \Sigma^*$ abgeleitet werden, die P aus dem Zustand q_0 in den Zustand q_{accept} überführen, wobei am Anfang und am Ende der Stapel jeweils leer ist.
- **Dies sind genau die Worte, die von P akzeptiert werden.**
- **Somit ist L = L(G) = L(P)**

Reguläre Sprachen vs. kontextfreie Sprachen

- **Korollar 6.3:**
  - Jede reguläre Sprache L ist auch kontextfrei.
- **Beweis:**
  - Jede reguläre Sprache L besitzt mindestens einen DFA A, der diese Sprache akzeptiert
  - Jeder DFA ist auch ein NFA
  - Jeder NFA ist auch ein PDA
  - Folglich gibt es einen PDA (und zwar A), der L akzeptiert
  - Damit ist L auch kontextfrei

Formale Sprachen und

Endliche Automaten

Pumping-Lemma für CFG

92

Das Pumping-Lemma für reguläre Sprachen

- **Pumping-Lemma für reguläre Sprachen**
  - Sei A eine reguläre Sprache.
    - Dann gibt es eine Zahl $p>0$
    - so dass für jedes Wort $s \in A$ mit $|s| \geq p$
    - s in drei Teile geteilt werden kann: $s = xyz$, wobei gilt
      - für alle $i \geq 0$: $xy^iz \in A$
      - $|y| > 0$
      - $|xy| \leq p$.

Das Pumping-Lemma für kontextfreie Sprachen

- **Pumping-Lemma für kontextfreie Sprachen:**
  - Sei L eine kontextfreie Sprache
    - Dann existiert eine Zahl $p>0$
    - So dass für jedes Wort $s \in L$ mit $|s| \geq p$
    - s in fünf Teile geteilt werden kann $s = uvxyz$, wobei gilt
      - für alle $i \geq 0$: $uv^ixy^iz \in L$
      - $|vy| \geq 1$
      - $|vxy| \leq p$

Vergleich der beiden Pumping Lemmata

- **Pumping-Lemma für reguläre Sprachen**
  - Sei A eine reguläre Sprache.
    - Dann gibt es eine Zahl p>0
    - so dass für jedes Wort $s \in A$ mit $|s| \geq p$
    - s in drei Teile geteilt werden kann: s = xyz, wobei gilt
      - für alle $i \geq 0$: $xy^iz \in A$
      - $|y| > 0$
      - $|xy| \leq p$.

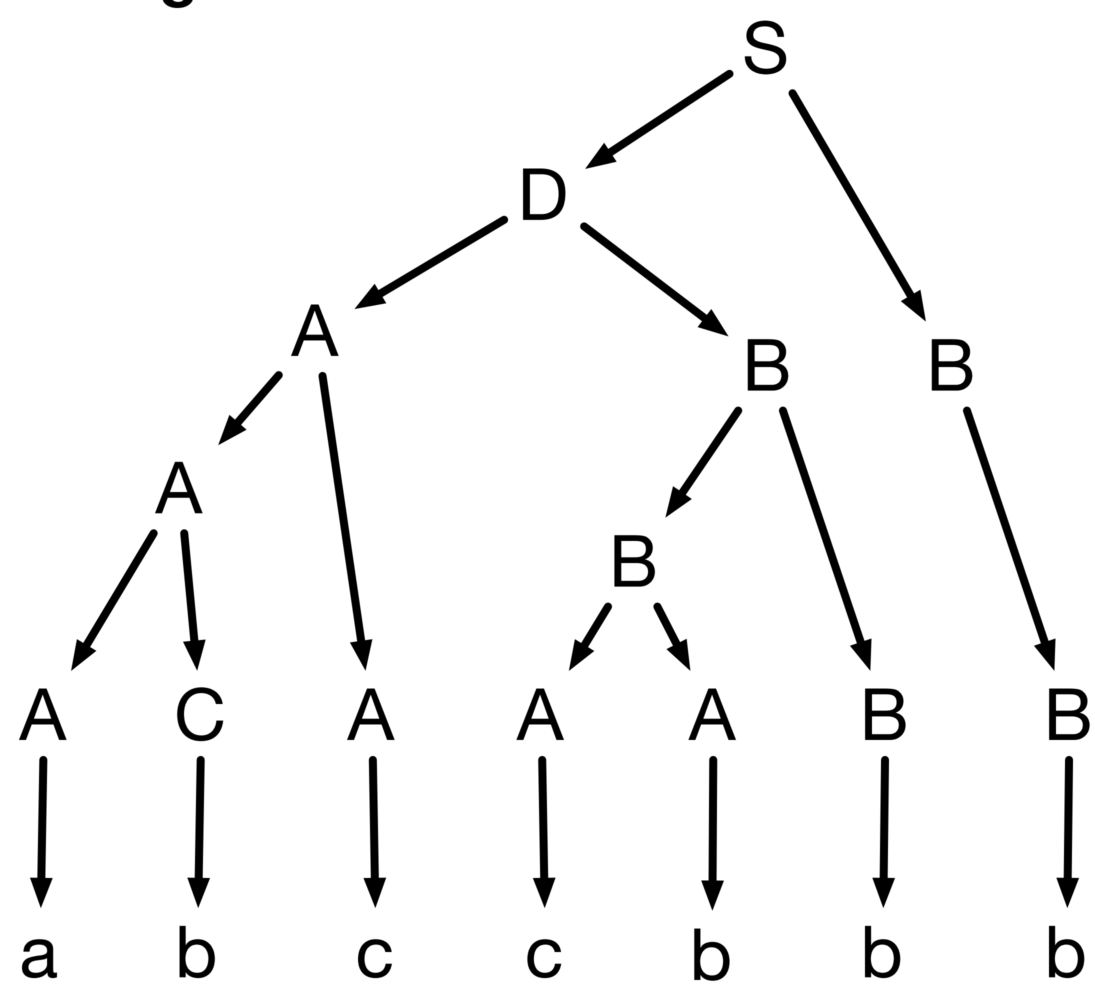
- **Pumping-Lemma für kontextfreie Sprachen:**
  - Sei L eine kontextfreie Sprache
    - Dann existiert eine Zahl p>0
    - So dass für jedes Wort $s \in L$ mit $|s| \geq p$
    - s in fünf Teile geteilt werden kann s = uvxyz, wobei gilt
      - für alle $i \geq 0$: $uv^ixy^iz \in L$
      - $|vy| \geq 1$
      - $|vxy| \leq p$

Beweis des Pumping-Lemmas für kontextfreie Sprachen - Beweisidee

- Verwende Chomsky-Normalform als Grammatik für L
- Betrachte Ableitungsbaum als Binärbaum
- Betrachte ein Wort w mit $|w| \geq p = 2^{|V|}$
- Auf dem Weg von einem Blatt zur Wurzel liegen $|V|+1$ Variablen, also kommt eine Variable doppelt vor
- Pumpe diesen Teilbaum auf

Beweis des Pumping-Lemmas für kontextfreie Sprachen

- **Sei G eine Grammatik in Chomsky-Normalform, die L erzeugt**
  - Solch eine Grammatik existiert immer
- **Betrachte Ableitungsbaum eines Wortes**
- **Betrachte Ableitungsbaum als Binärbaum**
  - Chomsky-Normalform
- **Setze $p=2^{|V|}$**

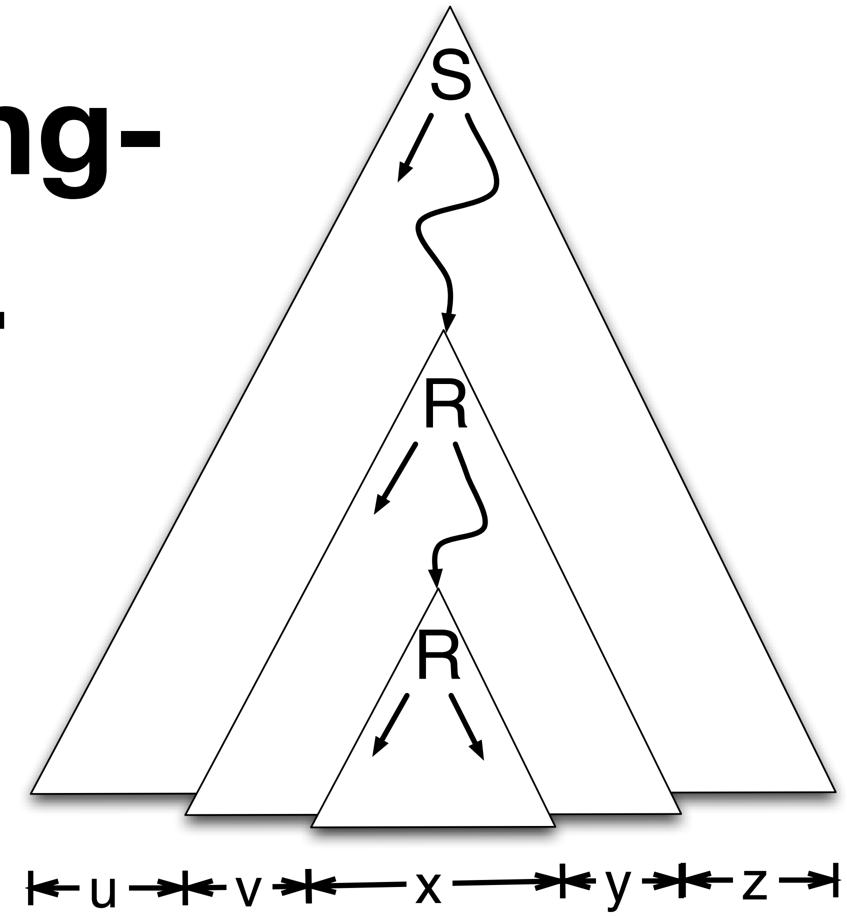


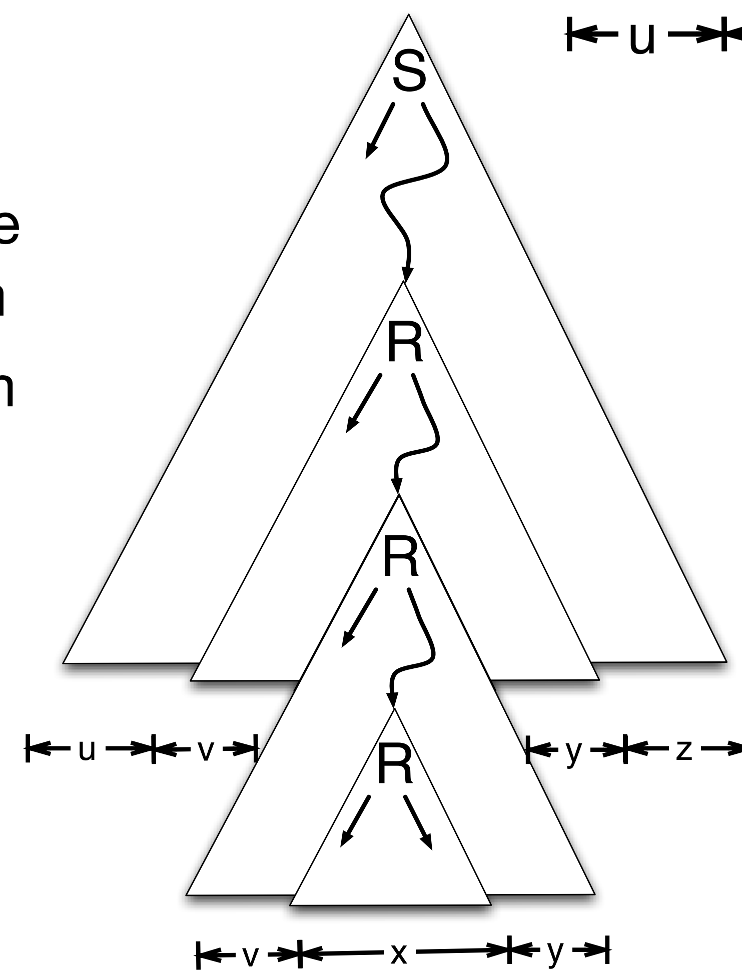
Beweis des Pumping-Lemmas für kontextfreie Sprachen

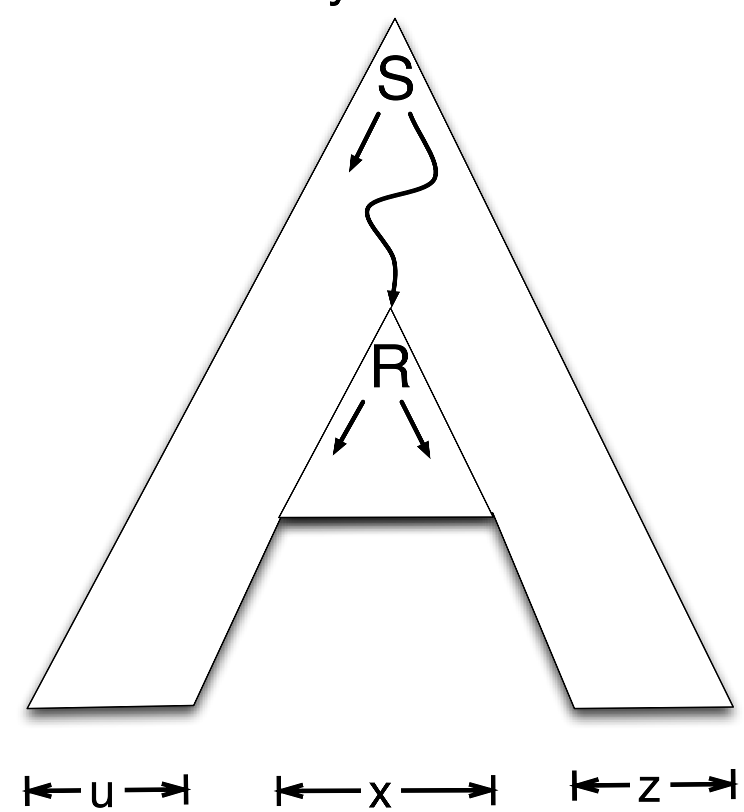
- **Betrachte Wort w der Länge p**
- **Ein beliebiger Ableitungsbaum T für w muss nun genau |w|≥p Blätter besitzen (Binärbaum zu T muss ≥p Blätter besitzen)**
  - Binärbaum muss Tiefe mindestens |V| haben, d.h. es muss einen Pfad von der Wurzel zu einem Blatt geben, auf dem mindestens |V| Kanten liegen
  - Wähle einen Pfad der Länge mindestens |V| in dem Binärbaum des Ableitungsbaums T aus
- **Auf dem Pfad liegen mindestens |V|+1 viele Knoten mit Variablen aus V**
  - Auf dem Weg vom Blatt zur Wurzel treffen wir eine Variable R doppelt an (Schubfachprinzip)
- **Betrachte nun Teilbaum, der R als Wurzel hat**
  - Ordne dem Teilbaum das Teilwort des Wortes zu, das sich aus den Terminalen an den Blättern des Teilbaums ergibt

Beweis des Pumping-Lemmas für CFL

- Somit haben wir nun **w=uvxyz**
- **Diese erfüllt**
  - $|vy| \geq 1$ (da Chomsky-Normalform)
  - $|vxy| \leq p$ (Der Teilbaum mit Wurzel R hat höchstens Tiefe $|V|$)
  - – für alle $i \geq 0$: $uv^ixy^iz \in A$
    - i>1: Platziere an Stelle R den Teilbaum mit Wurzel R und erhalte wieder korrekten Ableitungsbaum
    - i=0: Ersetze den Teilbaum, der am ersten mit R bezeichneten Teilbaum hängt, durch den Teilbaum, der am zweiten mit R bezeichneten Teilbaum hängt







Beispiel: Pumping Lemma

- **Beispiel:**
  - Die Sprache $L=\{a^n b^n c^n \mid n \geq 1\}$ ist nicht kontextfrei.
- **Beweis**:
  - Sei $p>0$ beliebig
  - Wähle $w= a^p b^p c^p \in L$
  - Sei $w=uvxyz$ eine Aufteilung von w in fünf Teile mit
    - $|vy|\geq 1$
    - $|vxy|\leq p$
    - Aus $|vxy|\leq p$ folgt, daß nur zwei der drei Symbole enthalten sein können
    - Führe Fallunterscheidung basierend auf fehlendem Buchstaben durch

Fallunterscheidung

- **a‘s fehlen:**
  - Betrachte das Wort $w=uv^0xy^0z$
  - da $|vy|\geq 1$ fehlen nun einige der Symbole b,c, die in w auftauchen
  - vy enthält kein a, daher fehlt kein a
  - damit enthält $w=uv^0xy^0z$ mehr a‘s als b‘s oder c‘s und liegt nicht in L
- **b‘s fehlen:**
  - Betrachte das Wort $w=uv^2xy^2z$
  - Da $|vy|\geq 1$ taucht das Symbol a oder das Symbol c in vy auf
  - damit enthält $w=uv^2xy^2z$ mehr a‘s oder c‘s als b‘s und liegt nicht in L
- **c‘s fehlen:**
  - Betrachte das Wort $w=uv^2xy^2z$
  - Da $|vy|\geq 1$ taucht das Symbol a oder das Symbol b in vy auf
  - damit enthält $w=uv^2xy^2z$ mehr a‘s oder b‘s als c‘s und liegt nicht in L

Fehlversuch: Pumping Lemma

- **Beispiel:**
  - L ={ w w | w ∈ {0,1}*} ist nicht kontextfrei.
- **Pumping-Lemma für kontextfreie Sprachen:**
  - Sei L eine kontextfreie Sprache
    - Dann existiert eine Zahl p>0
    - So dass für jedes Wort s ∈ L mit |s|≥p
    - s in fünf Teile geteilt werden kann s = uvxyz, wobei gilt
      - für alle i≥0: $uv^ixy^iz \in L$
      - |vy|≥1
      - |vxy|≤p

- **Betrachte:**
  - w = 0......0 1 0......0 1
  - also w = 0^p10^p1
- **Jetzt führt die Wahl:**

$$\underbrace{\mathbf{0000......0}}_{u}\ \underbrace{\mathbf{0}}_{v}\ \underbrace{\mathbf{1}}_{x}\ \underbrace{\mathbf{0}}_{y}\ \underbrace{\mathbf{0000......001}}_{z}$$

- **zu keinem Widerspruch!**
- **So geht das nicht.**

Beispiel
Pumping Lemma

- **Beispiel:**
  - Die Sprache L={ w w | w ∈ {0,1}*} ist nicht kontextfrei.
- **Pumping-Lemma für kontextfreie Sprachen:**
  - Sei L eine kontextfreie Sprache
    - Dann existiert eine Zahl p>0
    - So dass für jedes Wort s mit |s|≥p
    - s in fünf Teile geteilt werden kann s = uvxyz, wobei gilt
      - für alle i≥0: $uv^ixy^iz \in L$
      - |vy|≥1
      - |vxy|≤p

- **Betrachte:**
  - w = 0...0 1...1 0...0 1...1
  - also w = $0^p1^p0^p1^p$
- **0000000 1111111 0000000 1111111**

u v x y z

- **0000000 1111111 0000000 1111111**

u v x y z

- **0000000 1111111 0000000 1111111**
  - ...

Kontextfreie Sprachen und Grammatiken

Ende