Kapitel 2: Formale Sprachen

Gliederung

Einordnung

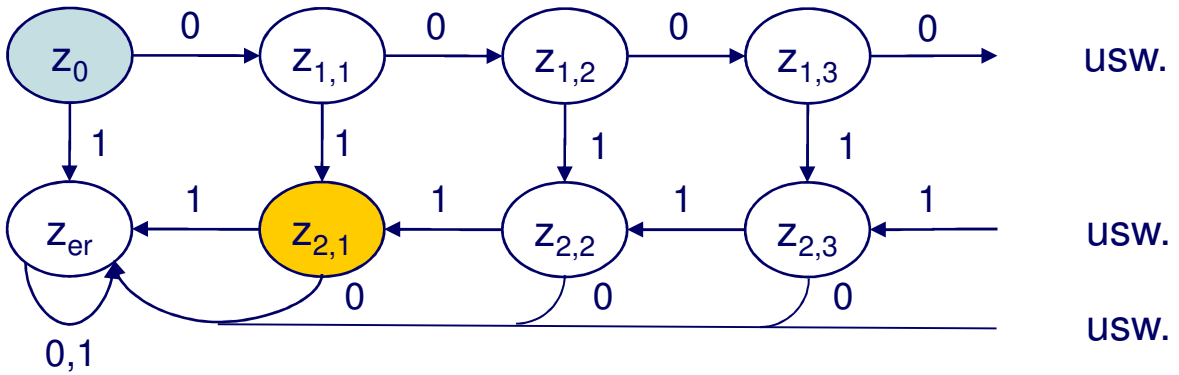
- algorithmischer Aspekt
  - Welche algorithmischen Ideen kann / sollte man verwenden, um das Wortproblem für kontextfreie Sprachen in den Griff zu bekommen?
- Aufgabenstellung
  - gegeben: eine kontextfreie Grammatik $G = [\Sigma,V,S,R]$
  - gesucht: ein Algorithmus, der bei Eingabe eines beliebigen Wortes $w \in \Sigma^*$, die Antwort „ja“ liefert, falls $w \in L(G)$ gilt, und die Antwort „nein“ liefert, falls $w \notin L(G)$ gilt

... und zwar anders / besser als mit dem CYK-Algorithmus

Beobachtung

Endliche Automaten genügen als Berechnungsmodell nicht.

- $L = \{ 0^n 1^n \mid n > 0 \}$ → Es gibt keinen endlichen Automaten A mit L(A) = L
- ... aber es gibt einen „unendlichen Automaten“ B mit L(B) = L



Endliche Automaten können nicht „unbeschränkt zählen“.

Mit unendliche Automaten kann man schlecht rechnen.

- Beobachtung

Endliche Automaten genügen als Berechnungsmodell nicht.

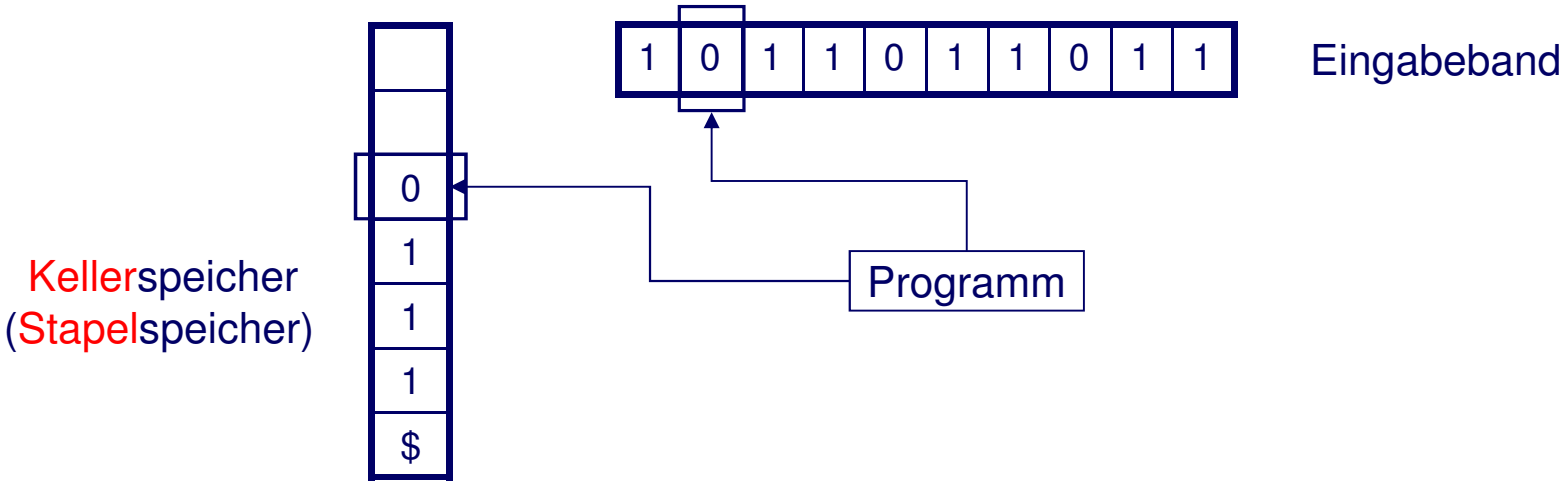
- $L = \{ a_1 ... a_n \# a_n ... a_1 \mid n > 0, a_i \in \{ 0,1 \} \} \}$

→ Es gibt keinen endlichen Automaten A mit L(A) = L.

Endliche Automaten merken sich nicht nur keine (unbeschränkten) Anzahlen, sondern auch (bzw. erst recht) keine (unbeschränkten) Reihenfolgen.

- Kellerautomaten als einfaches Berechnungsmodell

  - Welche elementare Operationen stehen zur Verfügung?
  - Wie viel Speicher steht zur Verfügung, und wie geht man damit um?
  - Wie wird die Eingabe eingegeben?
  - Wie wird die Ausgabe bestimmt?



Kelleroperationen

Im Keller (auf dem Stapel, hier als ungenannte globale Datenstruktur behandelt) kann man …

- das oberste Zeichen „sehen“:
  x := oberstes Zeichen **top**(x)
  … falls leer: den Kellerboden – eine Konstante
- das oberste Zeichen entfernen **pop**
  … falls leer, geschieht nichts
- ein weiteres Zeichen y darauf legen **push**(y)

oder – push und pop zusammenfassend –

- das oberste Zeichen durch einen gegebenen Teilstapel ersetzen
  replace_top_by(w)

Wieso *replace_top_by* gleichmächtig zu *push* & *pop*?

•replace_top_by(ab)	≈	pop; push(b); push(a)
•push(y)	≈	top(x); replace_top_by(y◦x)
•pop	≈	replace_top_by(ε)

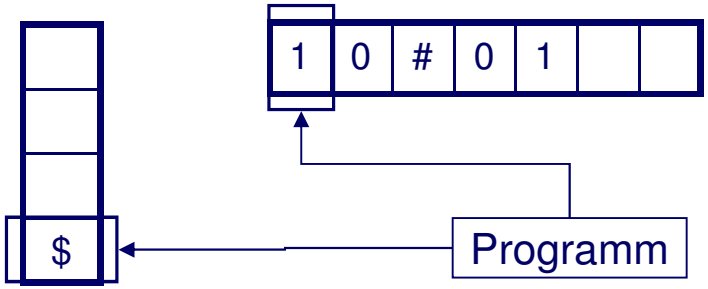
Beispiel 1 (Idee)

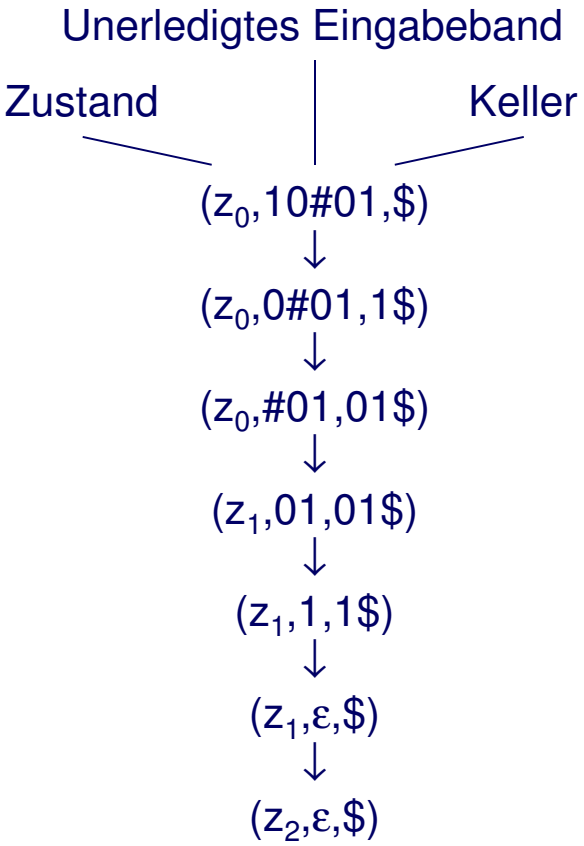
- Es seien $\Sigma = \{ 0,1,\# \}$ und $L = \{ a_1...a_n\#a_n...a_1 \mid n \in N$, alle $a_i \in \{ 0,1 \} \}$

- Wir arbeiten mit drei Zuständen: z_0, z_1, z_2
  - Im Startzustand z_0 werden sich die Zeichen, die auf dem Eingabeband vor dem # gelesen werden, im Keller gemerkt.
  - Wird im Zustand z_0 ein # gelesen, so wird in den Zustand z_1 gewechselt.
  - Im Zustand z_1 wird der Kellerinhalt zeichenweise mit den Zeichen auf dem Eingabeband verglichen:
    - wenn gleich, Zeichen auf dem Eingabeband „abhaken“, Zeichen im Keller entfernen und weiter;
    - wenn ungleich, nicht akzeptieren und „aufgeben“.
  - Wenn in Zustand z_1 festgestellt wird, dass der Keller leer ist, wird in den akzeptierenden Zustand z_2 gewechselt.

Beispiel 1 (gewünschtes Verhalten)

- Eingabe: w = 10#01

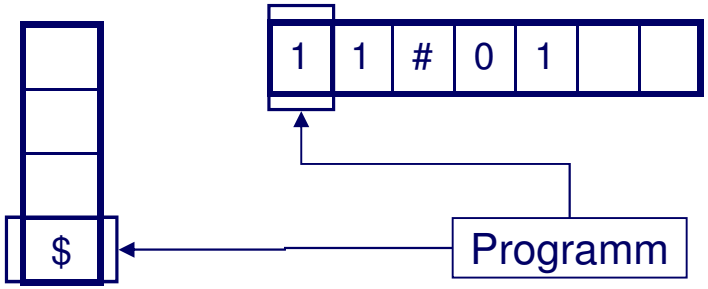




... eine akzeptierende Berechnung

Beispiel 1 (gewünschtes Verhalten)

- Eingabe: w = 11#01

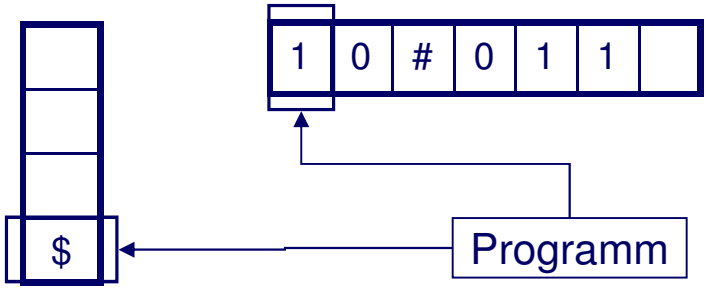


$(z_0,11\#01,\$)$
$\downarrow$
$(z_0,1\#01,1\$)$
$\downarrow$
$(z_0,\#01,11\$)$
$\downarrow$
$(z_1,01,11\$)$

... eine andere Berechnung bei einer anderen Eingabe, die keine akzeptierende Berechnung ist

Beispiel 1 (gewünschtes Verhalten)

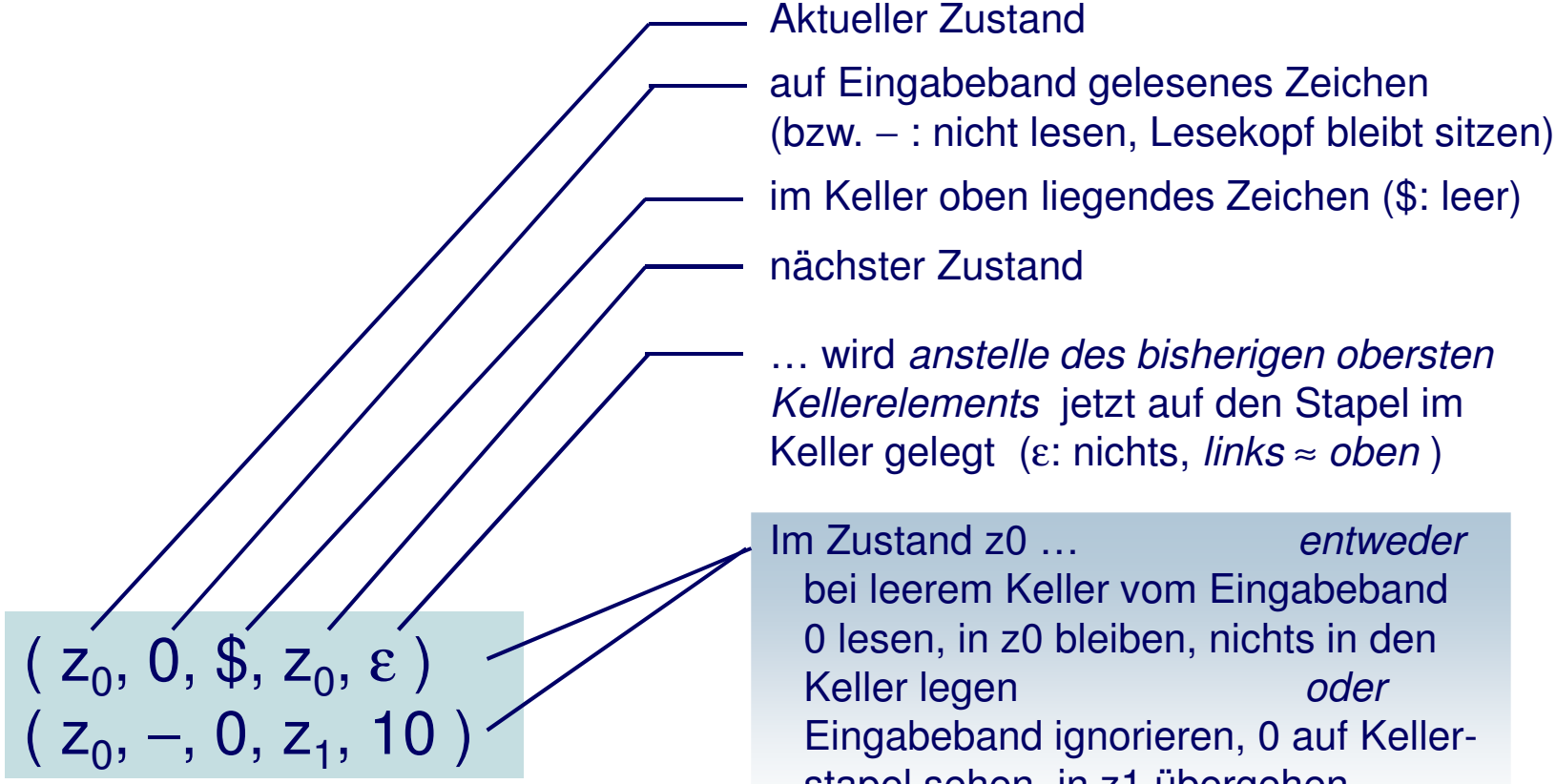
- Eingabe: w = 10#011



$(z_0,10\#011,\$)$
↓
$(z_0,0\#011,1\$)$
↓
$(z_0,\#011,01\$)$
↓
$(z_1,011,01\$)$
↓
$(z_1,11,1\$)$
↓
$(z_1,1,\$)$
↓
$(z_2,1,\$)$

... eine weitere Berechnung, die keine akzeptierende Berechnung ist

- Transitionen eines Kellerautomaten – Beispiel zweier Transitionen)



Beispiel 1 (Beschreibung des Kellerautomaten)

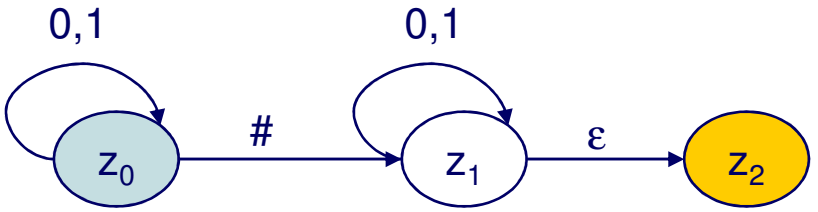
- $L = \{ a_1...a_n\#a_n...a_1 \mid n \in N, \text{alle } a_i \in \{ 0,1 \} \}$

Zustandsmenge $Z = \{ z_0,z_1,z_2 \}$; Eingabealphabet $\Sigma = \{ 0,1,\# \}$;
Kelleralphabet $\Gamma = \{ 0,1 \}$; Anfangszustand z_0; Leerer-Keller-Symbol \$;
Menge akzeptierender Zustände $F = \{ z_2 \}$; Transitionen in Transitionenmenge Δ:

$\{ (z_0,0,\$,z_0,0),$
$(z_0,1,\$,z_0,1),$
$(z_0,0,0,z_0,00),$
$(z_0,0,1,z_0,01),$
$(z_0,1,0,z_0,10),$
$(z_0,1,1,z_0,11),$
$(z_0,\#,0,z_1,0),$
$(z_0,\#,1,z_1,1),$

$(z_1,0,0,z_1,\varepsilon),$
$(z_1,1,1,z_1,\varepsilon),$
$(z_1,-,\$,z_2,\varepsilon) \}$

„Automat der Zustände ohne Keller“ – relativ nichtssagend, nur grobe Strukturübersicht:

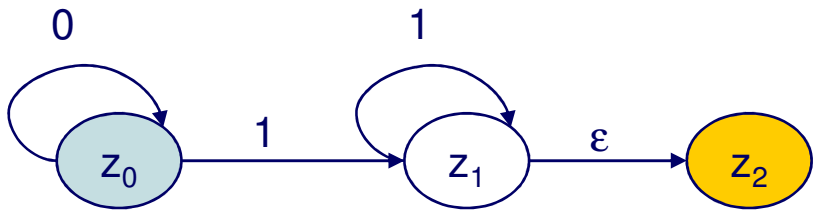


- Beispiel 2 (Beschreibung des Kellerautomaten)

$L = \{ 0^n 1^n \mid n > 0 \}$

$Z = \{ z_0, z_1, z_2 \};\ \Sigma = \{ 0,1 \};\ \Gamma = \{ 0,1 \};\ z_0;\ \$;\ F = \{ z_2 \};\ \Delta:$

$\{ (z_0,0,\$,z_0,0),$
$(z_0,0,0,z_0,00),$
$(z_0,1,0,z_1,\varepsilon),$
$(z_1,1,0,z_1,\varepsilon),$
$(z_1,-,\$,z_2,\varepsilon) \}$



Beispiel 3 (Sprache und Idee für Kellerautomat)

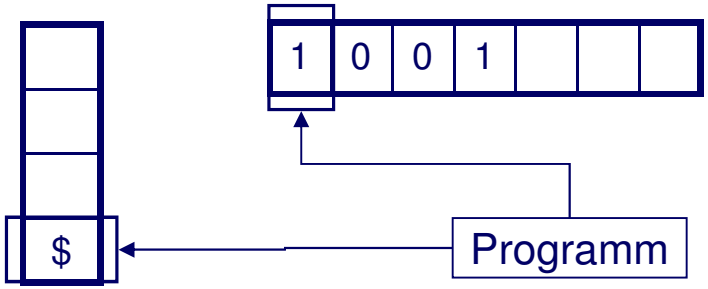
- Es seien $\Sigma = \{ 0,1 \}$ und $L = \{ a_1 ... a_n a_n ... a_1 \mid n \in N, \text{alle } a_i \in \{ 0,1 \} \}$.

- Wir verwenden drei Zustände z_0, z_1, z_2
  - In z_0 werden sich die Zeichen, die auf dem Eingabeband gelesen werden, im Keller gemerkt.
  - In z_0 kann - ohne ein Zeichen auf dem Eingabeband zu lesen - in den Zustand z_1 gewechselt werden.
  - Im Zustand z_1 wird der Kellerinhalt zeichenweise mit den Zeichen auf dem Eingabeband verglichen.
  - Wenn in z_1 festgestellt wird, dass der Keller leer ist, wird in den akzeptierenden Zustand z_2 gewechselt.

Dieser Kellerautomat ist offenbar nichtdeterministisch:
Es gibt Reaktionsalternativen in derselben Situation.

Beispiel 3 (gewünschtes Verhalten)

- Eingabe: w = 1001

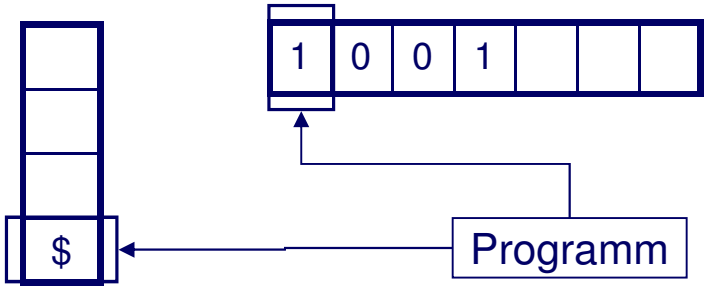


$(z_0,1001,\$)$
↓
$(z_0,001,1\$)$
↓
$(z_0,01,01\$)$
↓
$(z_1,01,01\$)$
↓
$(z_1,1,1\$)$
↓
$(z_1,\varepsilon,\$)$
↓
$(z_2,\varepsilon,\$)$

... eine akzeptierende Berechnung

Beispiel 3 (in Kauf genommenes Verhalten)

- Eingabe: w = 1001



$(z_0,1001,\$)$
$\downarrow$
$(z_0,001,1\$)$
$\downarrow$
$(z_0,01,01\$)$
$\downarrow$
$(z_0,1,001\$)$
$\downarrow$
$(z_0,\varepsilon,1001\$)$
$\downarrow$
$(z_1,\varepsilon,1001\$)$

... eine andere Berechnung, die – bei gleicher Eingabe (!) – keine akzeptierende Berechnung ist

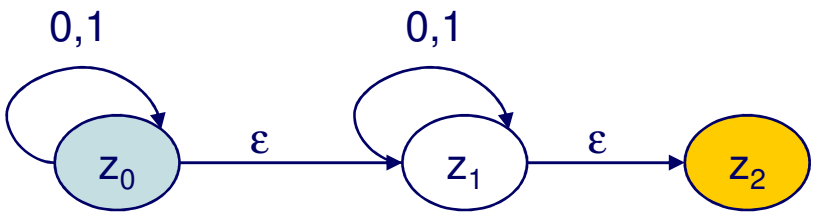
- Beispiel 3 (Beschreibung des Kellerautomaten)

$L = \{ a_1 ... a_n a_n ... a_1 \mid n > 0, a_i \in \{ 0,1 \} \}$

$Z = \{ z_0, z_1, z_2 \}$; $\Sigma = \{ 0,1 \}$; $\Gamma = \{ 0,1 \}$; z_0; \$; $F = \{ z_2 \}$;
Transitionenmenge Δ =

$\{ (z_0,0,\$,z_0,0),$
$(z_0,1,\$,z_0,1),$
$(z_0,0,0,z_0,00),$
$(z_0,0,1,z_0,01),$
$(z_0,1,0,z_0,10),$
$(z_0,1,1,z_0,11),$
$(z_0,-,0,z_1,0),$
$(z_0,-,1,z_1,1),$

$(z_1,0,0,z_1,\varepsilon),$
$(z_1,1,1,z_1,\varepsilon),$
$(z_1,-,\$,z_2,\varepsilon) \}$



- Arbeitsweise eines nichtdeterministischen Kellerautomaten, informell
  - Eingabeband
    - linearer Speicher; bestehend aus Zellen; je Zelle ein Zeichen
  - Lesekopf
    - kann den Inhalt der aktuellen Zelle lesen und muss dann sofort eine Zelle nach rechts gehen,
      kann aber auch vorläufig die aktuelle Zelle ignorieren und auf der aktuellen Zelle sitzen bleiben
  - Kellerspeicher
    - Sichtbar ist das oberste Zeichen (= das zuletzt gespeicherte unter den verbliebenen); Sonderfall: „Keller leer“
    - Eine Transition löscht dieses Zeichen, sofern nicht „Keller leer“, und kann dafür neue Zeichen in einer bestimmten Reihenfolge in den Keller schreiben (replace_top_by).

- Arbeitsweise eines nichtdeterministischen Kellerautomaten, informell
  - Programm steuert die Verarbeitung. Es befindet sich in unterschiedlichen Zuständen.
    - Fall 1: der aktuelle Zustand, das aktuelle Zeichen auf dem Eingabeband und das zuletzt gespeicherte Zeichen im Keller legen fest, wie es weiter geht.
    - Fall 2: der aktuelle Zustand und das oberste Zeichen im Keller legen bereits fest, wie es weiter geht; das aktuelle Zeichen auf dem Eingabeband wird momentan „ignoriert“ (und dabei auch nicht „verbraucht“).
  - Es ist keine Ausgabemedium vorgesehen.
    - Die Unterteilung in akzeptierende und nicht akzeptierende Zustände wird als Ausgabeverhalten (als ja/nein-Entscheidung) interpretiert.

Formale Definition der nichtdeterministischen Kellerautomaten

- Bestandteile eines nichtdeterministischen Kellerautomaten
  $K = [Z,\Sigma,\Gamma,z_0,\$,F,\Delta]$
  - endliche Menge Z von Zuständen
    - ausgezeichneter Anfangszustand $z_0 \in Z$
    - Teilmenge $F \subseteq Z$ von akzeptierenden Zuständen
  - endliches Eingabealphabet Σ
    plus ein ausgezeichnetes Symbol $- \notin \Sigma$
  - endliches Kelleralphabet Γ
    plus ausgezeichnetes Symbol $\$ \notin \Gamma$, das „im „leeren Keller steht“
  - Menge von Transitionen $\Delta \subseteq Z \times (\Sigma \cup \{-\}) \times (\Gamma \cup \{\$\}) \times Z \times \Gamma^*$
    (eine fünfstellige Relation)

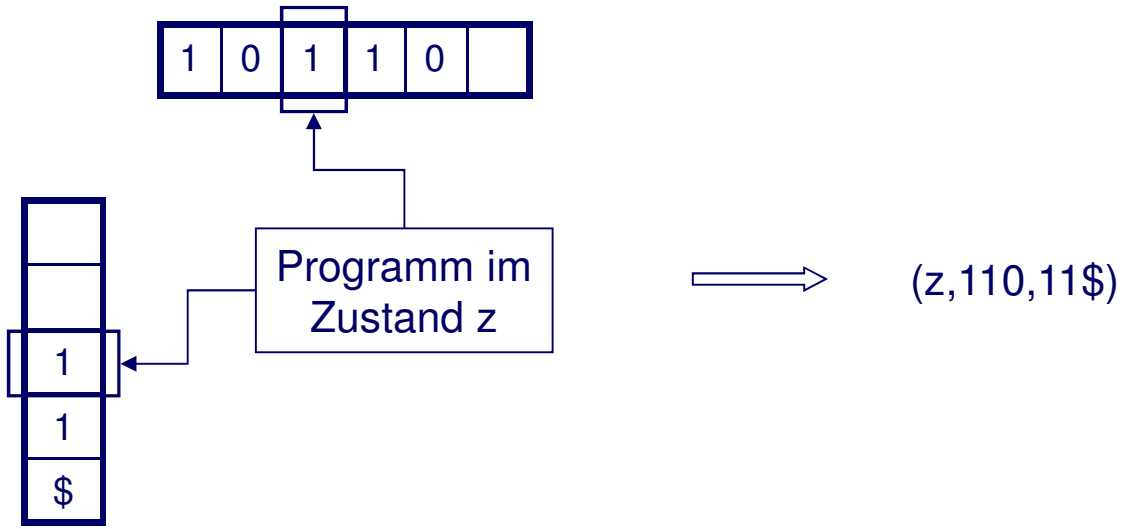
- Erinnerung: Verwendung eines nichtdeterministischen Kellerautomaten

  - Wörter $w \in \Sigma^*$ können auf dem Eingabeband stehen.
  - Das erste Zeichen auf dem Eingabeband wird im Zustand z_0 verarbeitet, wobei im Keller zunächst nur das Symbol \$ steht.
  - Programm $\approx$ Transitionenmenge Δ.
  - Ausgabeverhalten: Unterteilung in F und $Z \setminus F$.

... zentrale Begriffe: Rechenschritt, akzeptierende Berechnung

Berechnungen eines nichtdeterministischen Kellerautomaten

- Es sei $K = [Z,\Sigma,\Gamma,z_0,\$,F,\Delta]$ ein nichtdeterministischer Kellerautomat.
- Eine Situation in einer Berechnung von K wird beschrieben durch ein Tripel (z,u,v$) mit $z \in Z$, $u \in \Sigma^*$ und $v \in \Gamma^*$.
  - z ist der aktuelle Zustand von K;
  - u ist die „restliche“ (noch nicht gelesene) Zeichenkette auf dem Eingabeband;
  - v ist die Zeichenkette im Keller (links ≈ oben).



Berechnungen eines nichtdeterministischen Kellerautomaten (weiter)

- Es sei $K = [Z,\Sigma,\Gamma,z_0,\$,F,\Delta]$ ein nichtdeterministischer Kellerautomat.
- K definiert eine Relation $\rightarrow_K$ über der Menge $Z \times \Sigma^* \times \Gamma^*\{\$\}$.

  Sie bestimmt die möglichen einzelnen Rechenschritte, d.h. die Situationsänderungen von K.

- Es seien $a \in \Sigma$, $u \in \Sigma^*$, $x \in \Gamma$, $v \in \Gamma^*$ und $v' \in \Gamma^*$

Dann gilt:

- $(z,au,xv\$) \rightarrow_K (z',u,v'v\$)$, falls $(z,a,x,z',v') \in \Delta$
- $(z,au,\$) \rightarrow_K (z',u,v'\$)$, falls $(z,a,\$,z',v') \in \Delta$
- $(z,u,xv\$) \rightarrow_K (z',u,v'v\$)$, falls $(z,-,x,z',v') \in \Delta$
- $(z,u,\$) \rightarrow_K (z',u,v'\$)$, falls $(z,-,x,z',v') \in \Delta$

Wir schreiben $(z,u,v) \rightarrow_K^* (z',u',v')$, falls K die Situation (z,u,v) in endlich vielen Rechenschritten in die Situation (z',u',v') überführen kann. $\rightarrow_K^*$ ist die reflexive, transitive Hülle der Relation $\rightarrow_K$.

- Berechnungen eines nichtdeterministischen Kellerautomaten (weiter)
  - Es sei $K = [Z,\Sigma,\Gamma,z_0,\$,F,\Delta]$ ein nichtdeterministischer Kellerautomat.
  - Die von K definierte Relation $\rightarrow_K^*$ über der Menge $Z \times \Sigma^* \times \Gamma^*\{\$\}$ beschreibt die möglichen Berechnungen von M .
  - Es seien $w,w',v \in \Sigma^*$ und $z \in Z$.

Eine Berechnung $(z_0,w,\$) \rightarrow_K^* (z,w',v\$)$ von K heißt akzeptierende Berechnung von K bei Eingabe w, falls $z \in F$ und $w' = \varepsilon$ gilt.

  - d.h. bei einer akzeptierenden Berechnung erreicht K nach „vollständiger Verarbeitung“ von w einen akzeptierenden Zustand. Der verbleibende Kellerinhalt ist dabei gleichgültig.

Ein anderer „Dialekt“ von Kellerautomaten akzeptiert genau bei leerem Keller, unabhängig vom Zustand, wird hier nicht behandelt.

- Beispiel 3 (0-1-Palindrome gerader Länge, mögliche Berechnungen)

$Z = \{ z_0,z_1,z_2 \}; \Sigma = \{ 0,1 \}; \Gamma = \{ 0,1 \}; z_0; \$; F = \{ z_2 \};\ \Delta =$

$\{ (z_0,0,\$,z_0,0),\ (z_0,1,\$,z_0,1),\ (z_0,0,0,z_0,00),\ (z_0,0,1,z_0,01),$

$(z_0,1,0,z_0,10),\ (z_0,1,1,z_0,11),\ (z_0,-,0,z_1,0),\ (z_0,-,1,z_1,1),$

$(z_1,0,0,z_1,\varepsilon),\ (z_1,1,1,z_1,\varepsilon),\ (z_1,-,\$,z_2,\varepsilon) \}$

Akzeptierender Zustand aber keine akzeptierende Berechnung

$(z_0,1111,\$) \to_K (z_0,111,1\$) \to_K (z_1,111,1\$) \to_K (z_1,11,\$) \to_K (z_2,11,\$)$

$\downarrow_K$

$(z_0,11,11\$) \to_K (z_1,11,11\$) \to_K (z_1,1,1\$) \to_K (z_1,\varepsilon,\$) \to_K (z_2,\varepsilon,\$)$

$\downarrow_K$

$(z_0,1,111\$) \to_K (z_1,1,111\$) \to_K (z_1,\varepsilon,11\$)$

$\downarrow_K$

$(z_0,\varepsilon,1111\$) \to_K (z_1,\varepsilon,1111\$)$

- ... von einem nichtdeterministischen Kellerautomaten beschriebene Sprache

  - Es sei $K = [Z,\Sigma,\Gamma,z_0,\$,F,\delta]$ ein nichtdeterministischer Kellerautomat.
  - Es sei $w \in \Sigma^*$.

Das Wort w gehört zur von K beschriebenen Sprache L(K) gdw.
es bei Eingabe w eine akzeptierende Berechnung von K gibt.

Wenn K nach „vollständiger“ Verarbeitung von w einen akzeptierenden Zustand erreichen kann, so gehört w zur Sprache L(K).

- Grundlegender Zusammenhang

Es seien Σ ein endliches Alphabet und $L \subseteq \Sigma^*$. Dann sind die folgenden Aussagen äquivalent:

1. L ist eine kontextfreie Sprache.
2. Es gibt einen nichtdeterministischen Kellerautomaten K mit L(K) = L.

Es handelt sich um zwei Implikationen

a) Wenn L kontextfrei ist, so gibt es einen nichtdeterministischen Kellerautomaten K mit L(K) = L .

b) Wenn es einen nichtdeterministischen Kellerautomaten K mit L(K) = L gibt, so ist L kontextfrei.
*(Den Teil (b) zeigen wir hier **nicht**, siehe aber z.B. Lehrbücher, YouTube etc.)*

Implikation kontextfrei → ND-Kellerautomat, Beispiel

$\Sigma = \{ 0,1 \}$
$V = \{ S,A \}$
S

$S \rightarrow 0S1 \mid 0A1$
$A \rightarrow 00$

- Wir versuchen, einen nichtdeterministischen Kellerautomaten K zu konstruieren, der die durch G beschriebene Sprache akzeptiert, d.h. es soll L(K) = L(G) gelten.
- Er soll einen *Anfangszustand* z_0, einen *Arbeitszustand* z und einen *akzeptierenden* Zustand z* haben.
- Grundidee ist,
  - im Arbeitszustand oben im Keller von K die Anwendung der Regeln von G in einer Linksableitung zu „simulieren“ und
  - die produzierten Terminalzeichen mit denen des vorgelegten Wortes zu vergleichen.

Implikation kontextfrei → ND-Kellerautomat, Beispiel

$\Sigma = \{ 0,1 \}$
$V = \{ S,A \}$
S

$S \rightarrow 0S1 \mid 0A1$
$A \rightarrow 00$

- Im Zustand z wird jede obenliegende (Top-)Variable durch eine rechte Seite ersetzt (i.a. nichtdeterministisch):
  - (z,–,S,z,0S1)
  - (z,–,S,z,0A1)
  - (z,–,A,z,00)
- „Darum herum“ …
  - wird anfangs S in den Keller geschrieben:
    - (z_0,–,$,z,S),
  - werden alle oben auf dem Stapel liegenden Terminalzeichen mit denen des Eingabeworts „verglichen“:
    - (z,0,0,z,ε)
    - (z,1,1,z,ε)
  - signalisiert der leere Keller, dass insgesamt ein Terminalwort produziert wurde, das zur Eingabe passte:
    - (z,–,$,z*,ε)

Implikation kontextfrei → ND-Kellerautomat, Beispiel (weiter)

$\Sigma = \{ 0,1 \}$
$V = \{ S,A \}$
S

$S \rightarrow 0S1 \mid 0A1$
$A \rightarrow 00$

- Sei w = 0001
- Das Wort w kann wie folgt aus S abgeleitet werden (Linksableitung):
  - $S \rightarrow_G 0A1 \rightarrow_G 0001$
- Eine akzeptierende Berechnung von K bei Eingabe des Wortes w:
  - $(z_0,0001,\$) \rightarrow_K (z,0001,S\$)$
    $\rightarrow_K (z,0001,0A1\$)$
    $\rightarrow_K (z,001,A1\$)$
    $\rightarrow_K (z,001,001\$)$
    $\rightarrow_K (z,01,01\$)$
    $\rightarrow_K (z,1,1\$)$
    $\rightarrow_K (z,\varepsilon,\$)$
    $\rightarrow_K (z^*,\varepsilon,\$)$

$(z_0,-,\$,z,S)$
$(z,-,S,z,0S1)$
$(z,-,S,z,0A1)$
$(z,-,A,z,00)$
$(z,0,0,z,\varepsilon)$
$(z,1,1,z,\varepsilon)$
$(z,-,\$,z^*,\varepsilon)$

Implikation a) (kontextfrei → ND-Kellerautomat, allgemein)

- Es sei $L \subseteq \Sigma^*$ eine kontextfreie Sprache.
- Es sei $G = [\Sigma,V,S,R]$ eine kontextfreie Grammatik mit $L(G) = L$.

- Bilde einen nichtdeterministischer Kellerautomaten $K = [Z,\Sigma,\Gamma,z_0,\$,F,\Delta]$ mit $L(K) = L$ wie folgt:

- $Z = \{ z_0,z,z^* \}$
- $\Gamma = V \cup \Sigma$
- $F = \{ z^* \}$
- Δ enthält folgende Transitionen:
  - $(z_0,-,\$,z,S)$ und $(z,-,\$,z^*,\varepsilon)$
  - (z,x,x,z,ε) für alle $x \in \Sigma$
  - (z,x,V,z,r) für alle Regeln $V \rightarrow r$ in R

Anmerkungen

- Wir sind jetzt an einem Punkt, den wir schon aus dem Kapitel Reguläre Sprachen kennen:
  - Anhand einer kontextfreien Grammatik G können wir einen nichtdeterministischen Kellerautomaten K mit L(K) = L(G) konstruieren.
- Da man mit einem nichtdeterministischen Kellerautomaten K in der Praxis nichts anfangen kann (d.h. man kann K nicht benutzen, um das Wortproblem für die Sprache L(K) *effizient* zu lösen), benötigen wir wieder eine deterministische Variante dieses Berechnungsmodells.

... im Kapitel Reguläre Sprachen hatten wir uns deshalb überlegt, wie man anhand eines endlichen nichtdeterministischen Automaten A einen endlichen Automaten A‘ mit L(A‘) = L(A) konstruieren kann (Stichwort: Zustandsmengenautomat)

Anmerkungen (weiter)

- Im Unterschied zum Kapitel Reguläre Sprachen ist hier alles ein wenig komplizierter, weil
  - es kontextfreie Sprachen L gibt, für die es überhaupt keinen deterministischen Kellerautomaten K mit L(K) = L gibt,
  - man nicht entscheiden kann, ob es zu einer kontextfreien Sprache L einen deterministischen Kellerautomaten K mit L(K) = L gibt.
- Zusätzlich wird das Ganze ein wenig „verzwickter“, weil man einem deterministischen Kellerautomaten K nicht sofort ansieht, dass man mit seiner Hilfe das Wortproblem für die Sprache L(K) in Zeit O(n) lösen kann.

Für endliche (deterministischen) Automaten war das einfacher.

Spezialfall deterministische Kellerautomaten

- Es sei $K = [Z,\Sigma,\Gamma,z_0,\$,F,\Delta]$ ein nichtdeterministischer Kellerautomat.
- K ist ein deterministischer Kellerautomat, falls
  für alle $z \in Z$, $x \in \Sigma \cup \{-\}$ und $y \in \Gamma \cup \{\$\}$ gilt:
  - die Menge $\{ (z,\xi,y,z',v) \in \Delta \mid \xi \in \{x,-\}, z' \in Z, v \in \Gamma^* \}$ enthält höchstens eine Transition.

… kurz und gut:
In jeder Situation kann höchstens eine Transition benutzt werden.

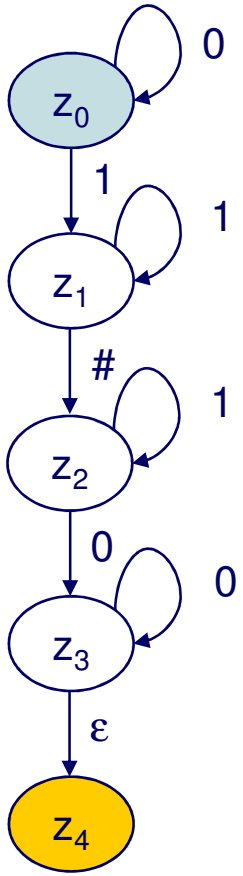
Es kann und muss Transitionen geben, in denen ein „–“ vorkommt (also kann K auf einer Eingabe w mehr als |w| viele Rechenschritte machen).

Beispiel 5

- $L = \{ 0^n \circ 1^m \circ \# \circ 1^m \circ 0^n \mid n,m > 0 \}$

$Z = \{ z_0,z_1,z_2,z_3,z_4 \}$; $\Sigma = \{ 0,1,\# \}$; $\Gamma = \{ 0,1 \}$; z_0; \$; $F = \{ z_4 \}$;
Transitionen in Δ:

- $(z_0,0,\$,z_0,0)$
- $(z_0,0,0,z_0,00)$
- $(z_0,1,0,z_1,10)$
- $(z_1,1,1,z_1,11)$
- $(z_1,\#,1,z_2,1)$
- $(z_2,1,1,z_2,\varepsilon)$
- $(z_2,0,0,z_3,\varepsilon)$
- $(z_3,0,0,z_3,\varepsilon)$
- $(z_3,-,\$,z_4,\varepsilon)$

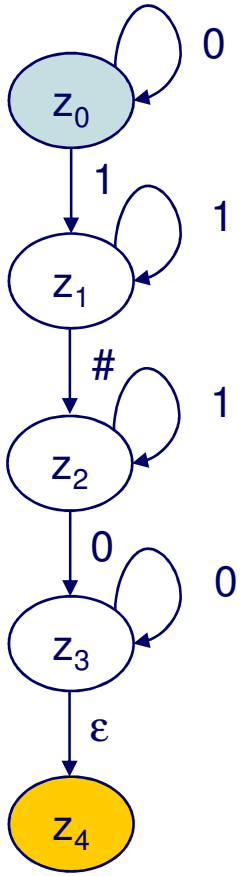


Beispiel 5

- $L = \{ 0^n \circ 1^m \circ \# \circ 1^m \circ 0^n \mid n,m > 0 \}$

$Z = \{ z_0,z_1,z_2,z_3,z_4 \}$; $\Sigma = \{ 0,1,\# \}$; $\Gamma = \{ 0,1 \}$; z_0; \$; $F = \{ z_4 \}$;
Transitionen in Δ:

- $(z_0,0,\$,z_0,0)$
- $(z_0,0,0,z_0,00)$
- $(z_0,1,0,z_1,10)$
- $(z_1,1,1,z_1,11)$
- $(z_1,\#,1,z_2,1)$
- $(z_2,1,1,z_2,\varepsilon)$
- $(z_2,0,0,z_3,\varepsilon)$
- $(z_3,0,0,z_3,\varepsilon)$
- $(z_3,-,\$,z_4,\varepsilon)$



Grobe Beschreibung:
z_0: ***push*** *Eingabe-0en, auch die erste Eingabe-1, dann …*
z_1: ***push*** *Eingabe-1en, auch das erste Eingabe-#, dann …*
z_2: *solange mögl., lies Eing.-0,* ***pop*** *Keller-0, auch E-1/K-1, dann …*
z_3: *solange mögl., lies Eing.-1,* ***pop*** *Keller-1; wenn Keller leer, dann …*
z_4: *akzeptiere und* ***stop***

- Diskussion (Rolle des Nichtdeterminismus)

> Es sei Σ ein endliches Alphabet mit mindestens zwei Zeichen. Dann gibt es kontextfreie Sprachen $L \subseteq \Sigma^*$ für die gilt:
> Es gibt keinen deterministischen Kellerautomaten K mit L(K) = L.

... Alphabetgröße >1, da jede kontextfreie Sprache über einem Alphabet der Größe 1 auch regulär ist, also auch Sprache eines endlichen Automaten, der mit (z,x,z') → ((z,x,$,z',ε) leicht zum deterministischen Kellerautomaten erweitert werden kann.

- Beispiel

  - Für die kontextfreie Sprache $L = \{ a_1...a_n a_n...a_1 \mid n \in N, a_i \in \{ 0,1 \} \}$ gibt es überhaupt keinen deterministischen Kellerautomaten K mit L(K) = L.

- Diskussion (Rolle des Ignorierens „–“)

Es sei Σ ein Alphabet mit mindestens zwei Zeichen
Dann gibt es eine kontextfreie Sprache $L \subseteq \Sigma^*$ der Art, dass
jeder deterministische Kellerautomaten K mit L(K) = L
mindestens eine Transition der Form (z,–,x,z‘,v) enthalten muss.

Ein deterministischer Kellerautomat muss das aktuell gelesene Zeichen auf dem Eingabeband „ignorieren“ können dürfen.

- Diskussion (Rolle des Ignorierens „–“, Beweisidee)

  - Sei $L = L_1 \cup L_2$ mit $L_1 = \{ 0^n \circ 1^m \circ \# \circ 1^m \circ 0^n \circ \# \mid n,m \in N \}$ und $L_2 = \{ 0^n \circ 1^m \circ \# \circ 0^n \circ \# \mid n,m \in N \}$
  - Deterministischer Kellerautomat mit $Z = \{ z_0,z_1,z_2,z_3,z_4,z_5 \}$, $\Sigma = \{ 0,1,\# \}$, $\Gamma = \{ 0,1 \}$, Startzustand z_0, akzeptierendem Zustand z_4 und den folgenden Transitionen:

• $(z_0,0,\$,z_0,0)$	• $(z_2,1,1,z_2,\varepsilon)$	• $(z_2,0,1,z_5,\varepsilon)$
• $(z_0,0,0,z_0,00)$	• $(z_2,0,0,z_3,\varepsilon)$	• $(z_5,-,1,z_5,\varepsilon)$
• $(z_0,1,0,z_1,10)$	• $(z_3,0,0,z_3,\varepsilon)$	• $(z_5,-,0,z_3,\varepsilon)$
• $(z_1,1,1,z_1,11)$	• $(z_3,\#,\$,z_4,\varepsilon)$	• $(z_5,-,\$,z_3,\varepsilon)$
• $(z_1,\#,1,z_2,1)$		

Ein DKA K muss sich die Nullen und Einsen vor dem ersten # im Keller merken. da auf dem Eingabeband ein w aus L_1 stehen kann. Wenn w aber aus L_2 ist, merkt K das an der Eingabe 0 und der 1 im Keller. Jetzt muss K so lange die nächste Null „ignorieren“, bis er die Einsen aus dem Keller geräumt hat und endlich an die im Keller gemerkten Nullen kommt.

Effizienzvorteile deterministischer Kellerautomaten

- Es sei $L \subseteq \Sigma^*$ eine kontextfreie Sprache, für die es einen deterministischen Kellerautomaten K mit $L(K) = L$ gibt (d.h. L ist eine sogenannte deterministische kontextfreie Sprache)

Dann kann man mit Hilfe des deterministischen Automaten K einen Algorithmus zur Lösung des Wortproblems für L angeben, der bei der Eingabe eines Worts w höchstens $c \cdot |w|$ Rechenschritte benötigt.

Wenn man eine Sprache mit Hilfe von kontextfreien Grammatiken definiert, ist es vorteilhaft, wenn sich die Sprache mit Hilfe eines deterministischen Kellerautomaten beschreiben lässt, da das Wortproblem praktisch genauso effizient lösbar ist wie bei einer regulären Sprache.

Wenn das nicht gelingt, muss man den „langsameren“ CYK-Algorithmus zur Lösung des Membership-Problems verwenden.

Effizienzvorteile deterministischer Kellerautomaten – Begründung

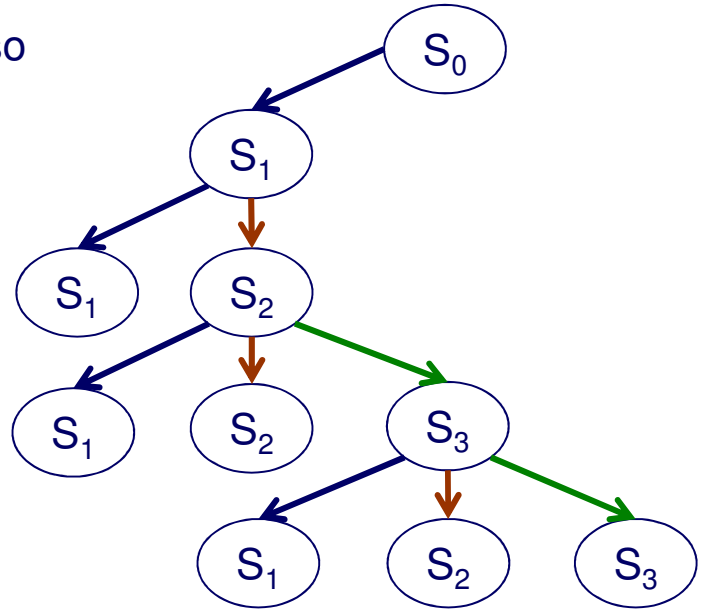
- Wenn es in der Transitionenmenge Δ des deterministischen Kellerautomaten K keine ignorierende Transition (also der Form (z,–,x,z‘,v)) gibt, gilt offenbar c = 1: K kann jedes Zeichen auf dem Eingabeband genau einmal lesen und kann es nie „ignorieren“.

- Wenn es in Δ jedoch Transition der Form (z,–,x,z‘,v) gibt, so ist c = t + 1, wobei t die Anzahl dieser Transitionen ist, weil
  - K ein deterministischer Kellerautomat ist und deshalb ein Zeichen auf dem Eingabeband höchsten t-mal nacheinander „ignorieren“ kann, bevor er in eine Endlosschleife läuft

... um zu entscheiden, ob w zu L(K) gehört, lässt man parallel einen Zähler laufen, in dem die Rechenschritte gezählt werden. Falls K „zu lange“ rechnet, wird abgebrochen und das Ergebnis „w gehört nicht zu L(K)“ ausgegeben.

Illustration der Begründung

- sei K ein deterministischer Kellerautomat mit drei Transitionen der Form (z,–,x,z‘,v), sagen wir Blau, Rot und Grün
- **Annahme:** es gibt eine Situation S_0, so dass K in drei aufeinander folgenden Rechenschritten eine diese drei Transitionen benutzen kann (also das aktuell gelesene Zeichen der Eingabe w dreimal nacheinander „ignoriert“)



... da K ein deterministischer Kellerautomat ist, würde K jeweils in eine Endlosschleife geraten und mehr als 4|w| viele Rechenschritte durchführen*

Laufzeitvergleich

- Es sei $L \subseteq \Sigma^*$ eine kontextfreie Sprache
- Es sei K ein deterministische Kellerautomat mit L(K) = L.
- Wie lange dauert die Prüfung, ob ein Wort zu L gehört?

Annahme: unser Rechner schafft 10^9 Operationen pro Sekunde

Länge n des zu testenden Worts	Algorithmus auf der Grundlage von K (O(n))	CYK-Algorithmus ($O(n^3)$)
100	0.0000001 s	0.001 s
1000	0.000001 s	1 s
10000	0.00001 s	≥ 16 min

- Zum Schluss: eine Übung zum Umgang mit der neuen Technik

Gegeben seien

- Ein deterministischer Kellerautomat K_1 und ein („echt“) nicht-deterministischer Kellerautomat K_2 mit $Z = \{ z_0,z_1,z_2 \}$, $\Sigma = \Gamma = \{ a \}$, Startzustand z_0, akzeptierendem Zustand z_2 und den folgenden Transitionen:

für K_1:

- $(z_0,a,\$,z_1,a)$
- $(z_1,a,a,z_2,\varepsilon)$
- $(z_2,a,\$,z_1,a)$

für K_2:

- $(z_0,a,\$,z_0,a)$
- (z_0,a,a,z_0,aa)
- $(z_0,–,a,z_1,a)$
- $(z_1,a,a,z_1,\varepsilon)$
- $(z_1,–,\$,z_2,\varepsilon)$

- Welche Sprache akzeptiert K_1, welche K_2?