

0. Einleitung und Grundbegriffe

1. Endliche Automaten

2. Formale Sprachen

3. Berechnungstheorie

4. Komplexitätstheorie

2.1. Chomsky-Grammatiken

2.2. Reguläre Sprachen

2.3. Kontextfreie Sprachen (weiter)

▶ algorithmischer Aspekt

- Welche algorithmischen Ideen kann / sollte man verwenden, um das Wortproblem für kontextfreie Sprachen in den Griff zu bekommen?

▶ Aufgabenstellung

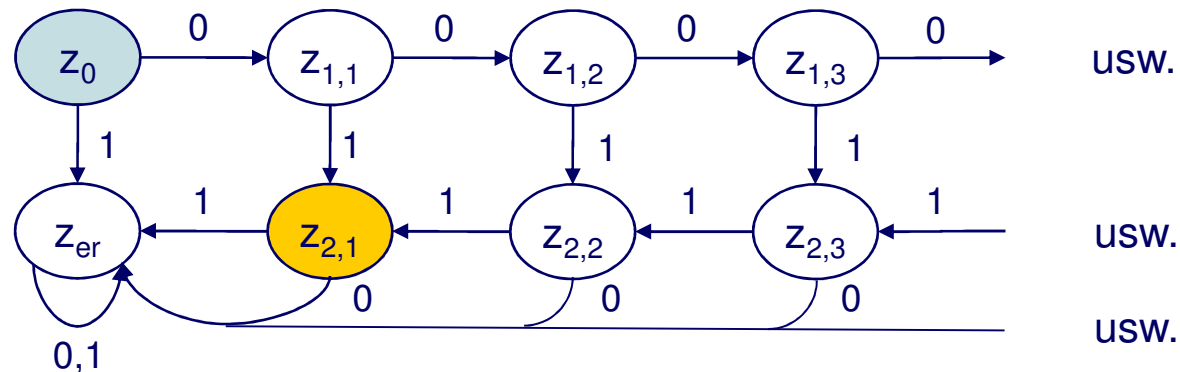
- gegeben: eine kontextfreie Grammatik $G = [\Sigma, V, S, R]$
- gesucht: ein Algorithmus, der bei Eingabe eines beliebigen Wortes $w \in \Sigma^*$, die Antwort „ja“ liefert, falls $w \in L(G)$ gilt, und die Antwort „nein“ liefert, falls $w \notin L(G)$ gilt

... 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.

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

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

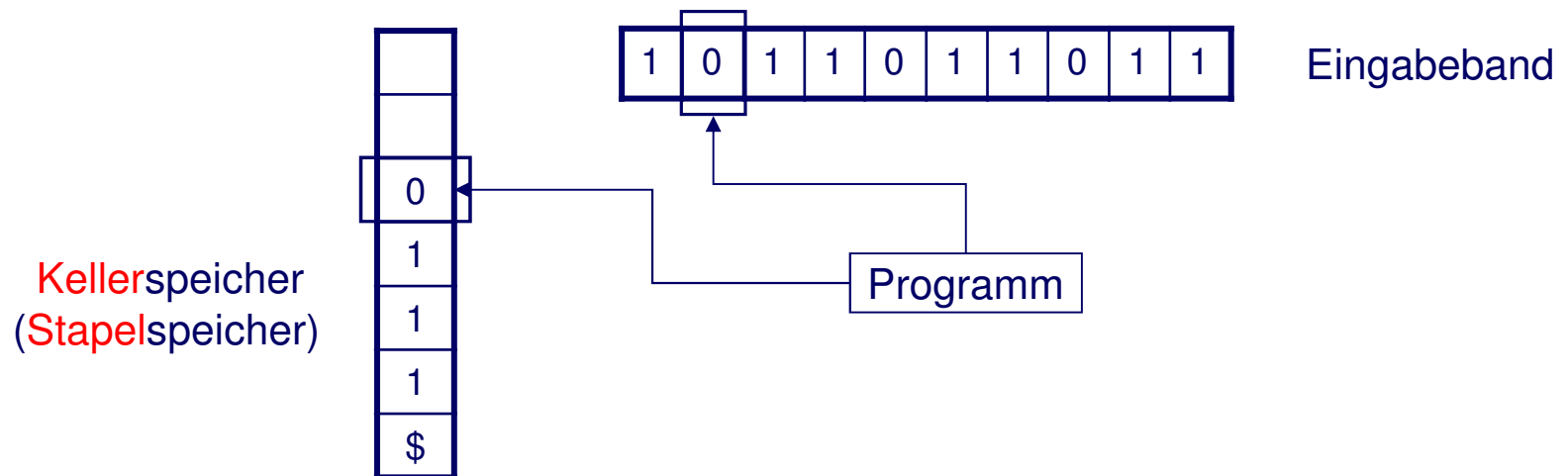
Endliche Automaten merken sich nicht nur keine (unbeschränkten) Anzahlen, sondern auch (bzw. erst recht) keine (unbeschränkten) Reihenfolgen.

Kapitel 2: Formale Sprachen

Nichtdeterministische Kellerautomaten

► 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?



Kapitel 2: Formale Sprachen

Nichtdeterministische Kellerautomaten

► Kellerooperationen

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

- $\text{replace_top_by}(ab) \approx \text{pop}; \text{push}(b); \text{push}(a)$
- $\text{push}(y) \approx \text{top}(x); \text{replace_top_by}(yx)$
- $\text{pop} \approx \text{replace_top_by}(\epsilon)$

Replace-Argument
links ≈ oben

► Beispiel 1 (Idee)

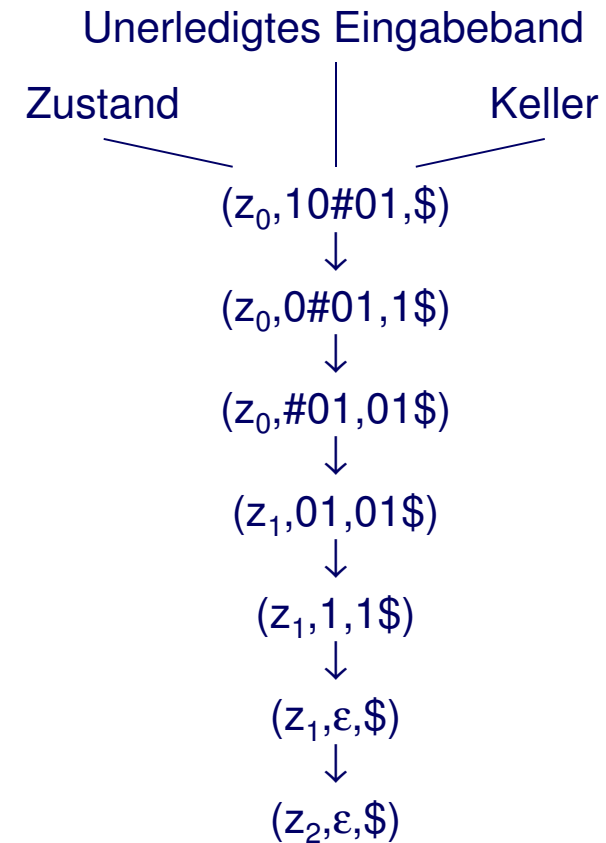
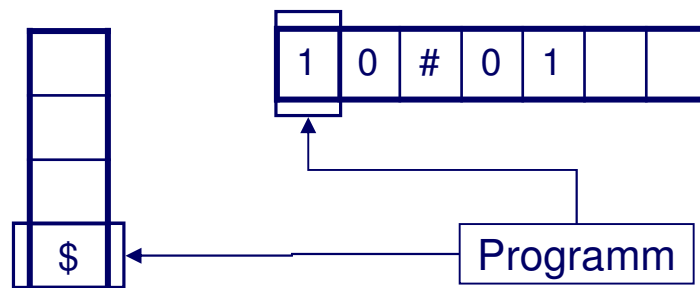
- Es seien $\Sigma = \{ 0,1,\# \}$ und $L = \{ a_1 \dots a_n \# a_n \dots a_1 \mid n \in \mathbb{N}, \text{ alle } a_i \in \{ 0,1 \} \}$
- Wir arbeiten mit drei Zuständen: z_0, z_1, z_2
 - Im Startzustand z_0 werden sich die Zeichen, die auf dem Eingabeband vor dem $\#$ gelesen werden, im Keller gemerkt.
 - 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.

Kapitel 2: Formale Sprachen

Nichtdeterministische Kellerautomaten

► Beispiel 1 (gewünshtes Verhalten)

- Eingabe: $w = 10\#01$



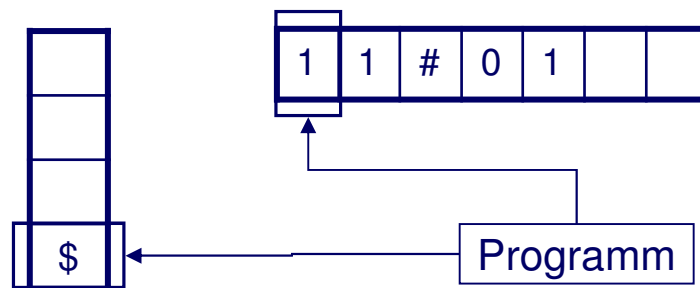
... eine akzeptierende Berechnung

Kapitel 2: Formale Sprachen

Nichtdeterministische Kellerautomaten

► Beispiel 1 (gewünshtes Verhalten)

- Eingabe: $w = 11\#01$


$$\begin{aligned} & (z_0, 11\#01, \$) \\ & \downarrow \\ & (z_0, 1\#01, 1\$) \\ & \downarrow \\ & (z_0, \#01, 11\$) \\ & \downarrow \\ & (z_1, 01, 11\$) \end{aligned}$$

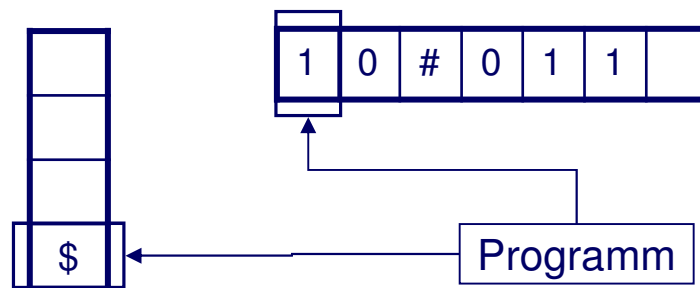
... eine andere Berechnung bei einer anderen Eingabe, die keine akzeptierende Berechnung ist

Kapitel 2: Formale Sprachen

Nichtdeterministische Kellerautomaten

► Beispiel 1 (gewünshtes Verhalten)

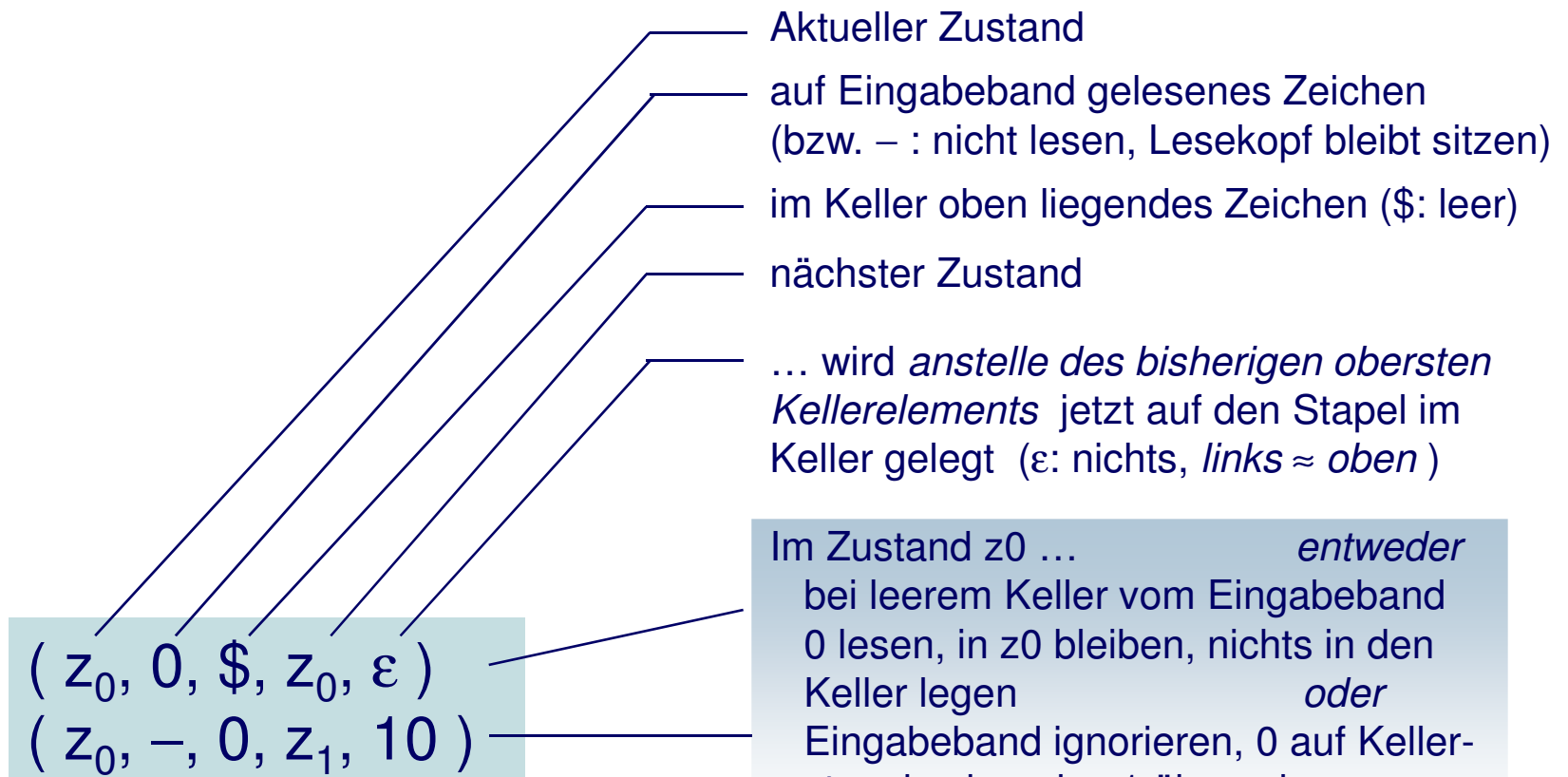
- Eingabe: $w = 10\#011$



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

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

► Transitionen eines Kellerautomaten – Beispiel zweier Transitionen)



Im Zustand z_0 ... *entweder*
bei leerem Keller vom Eingabeband
0 lesen, in z_0 bleiben, nichts in den
Keller legen *oder*
Eingabeband ignorieren, 0 auf Keller-
stapel sehen, in z_1 übergehen,
1 auf die 0 im Kellerstapel legen.

► Beispiel 1 (Beschreibung des Kellerautomaten)

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

Zustandsmenge $Z = \{ z_0, z_1, z_2 \}$; Eingabealphabet $\Sigma = \{ 0, 1, \# \}$;

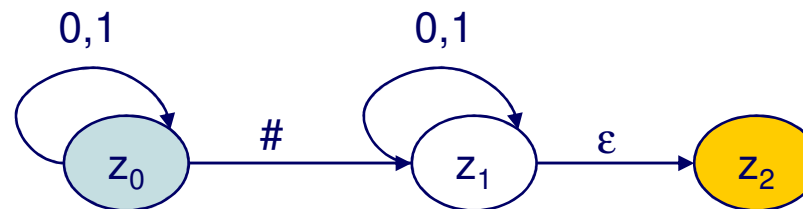
Kellularphabet $\Gamma = \{ 0, 1 \}$; Anfangszustand z_0 ; Leerer-Keller-Symbol $\$$;

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:



Kapitel 2: Formale Sprachen

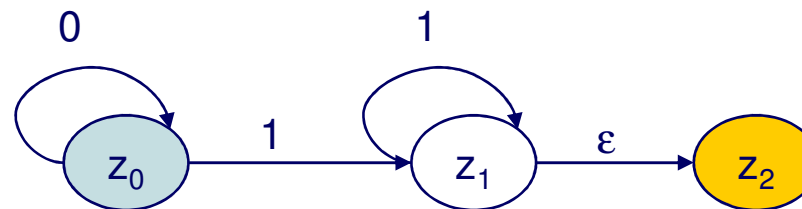
Nichtdeterministische Kellerautomaten

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

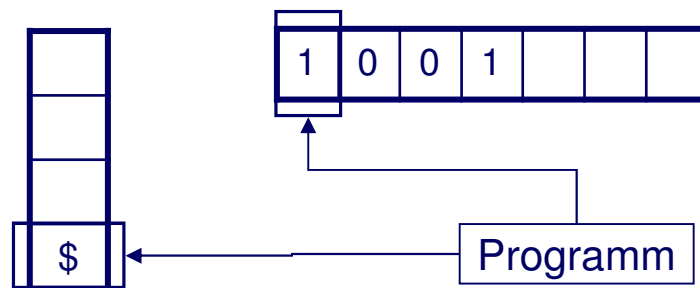
*Dieser Kellerautomat ist offenbar nichtdeterministisch:
Es gibt Reaktionsalternativen in derselben Situation.*

Kapitel 2: Formale Sprachen

Nichtdeterministische Kellerautomaten

► Beispiel 3 (gewünshtes Verhalten)

- Eingabe: $w = 1001$



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

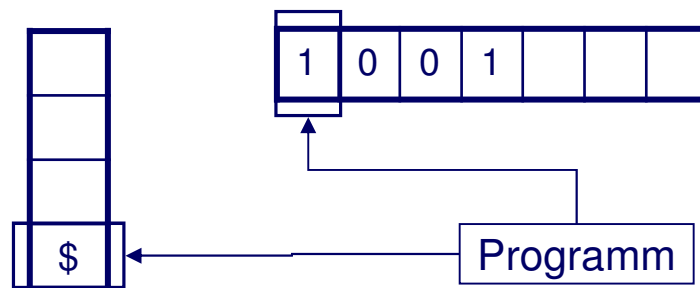
... eine akzeptierende Berechnung

Kapitel 2: Formale Sprachen

Nichtdeterministische Kellerautomaten

► Beispiel 3 (in Kauf genommenes Verhalten)

- Eingabe: $w = 1001$



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

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

Kapitel 2: Formale Sprachen

Nichtdeterministische Kellerautomaten

► Beispiel 3 (Beschreibung des Kellerautomaten)

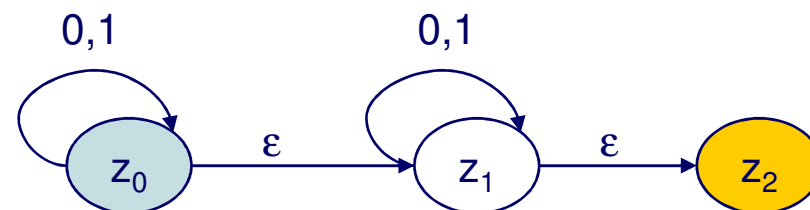
$$L = \{ a_1 \dots a_n a_n \dots a_1 \mid n > 0, a_i \in \{ 0, 1 \} \}$$

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

Transitionenmenge $\Delta =$

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

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



- ▶ 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)

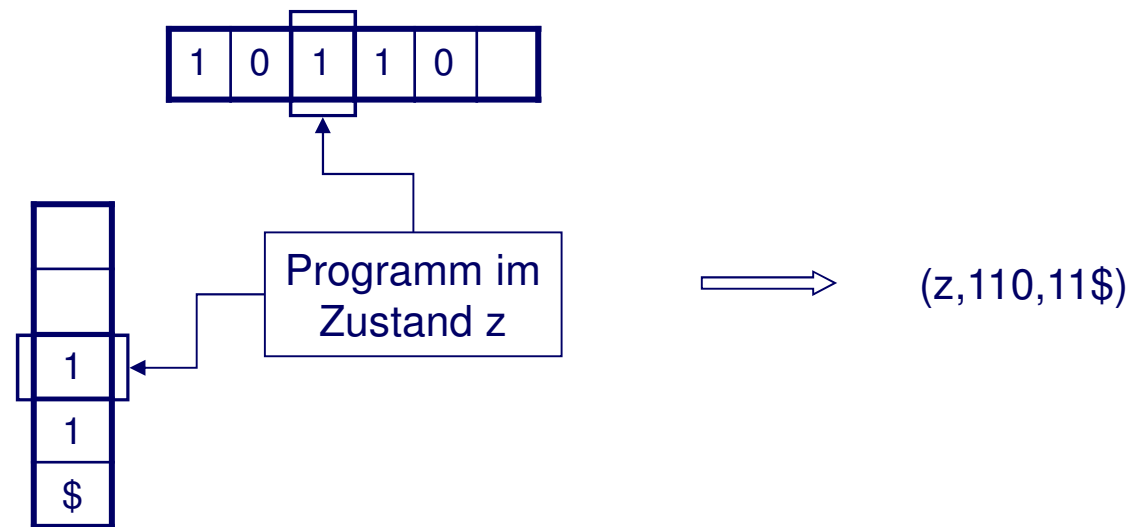
► Erinnerung: Verwendung eines nichtdeterministischen Kellerautomaten

- 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 \approx 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.

Kapitel 2: Formale Sprachen

Nichtdeterministische Kellerautomaten

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

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

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

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

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

*Akzeptierender
Zustand aber keine
akzeptierende
Berechnung*

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

\downarrow_K

$(z_0, 11, 11\$) \rightarrow_K (z_1, 11, 11\$) \rightarrow_K (z_1, 1, 1\$) \rightarrow_K (z_1, \epsilon, \$) \rightarrow_K (z_2, \epsilon, \$)$

\downarrow_K

$(z_0, 1, 111\$) \rightarrow_K (z_1, 1, 111\$) \rightarrow_K (z_1, \epsilon, 11\$)$

\downarrow_K

$(z_0, \epsilon, 1111\$) \rightarrow_K (z_1, \epsilon, 1111\$)$

► ... 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 \rightarrow 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.

Kapitel 2: Formale Sprachen

Nichtdeterministische Kellerautomaten und kontextfreie Grammatiken

► Implikation kontextfrei \rightarrow 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):
- „Darum herum“ ...
 - wird anfangs S in den Keller geschrieben:
 - werden alle oben auf dem Stapel liegenden Terminalzeichen mit denen des Eingabeworts „verglichen“:
 - signalisiert der leere Keller, dass insgesamt ein Terminalwort produziert wurde, das zur Eingabe passte:

$$\bullet (z, -, S, z, 0S1)$$

$$\bullet (z, -, S, z, 0A1)$$

$$\bullet (z, -, A, z, 00)$$

$$\bullet (z_0, -, \$, z, S),$$

$$\bullet (z, 0, 0, z, \varepsilon)$$

$$\bullet (z, 1, 1, z, \varepsilon)$$

$$\bullet (z, -, \$, z^*, \varepsilon)$$

Kapitel 2: Formale Sprachen

Nichtdeterministische Kellerautomaten und kontextfreie Grammatiken

► Implikation kontextfrei \rightarrow ND-Kellerautomat, Beispiel (weiter)

$$\Sigma = \{ 0,1 \}$$

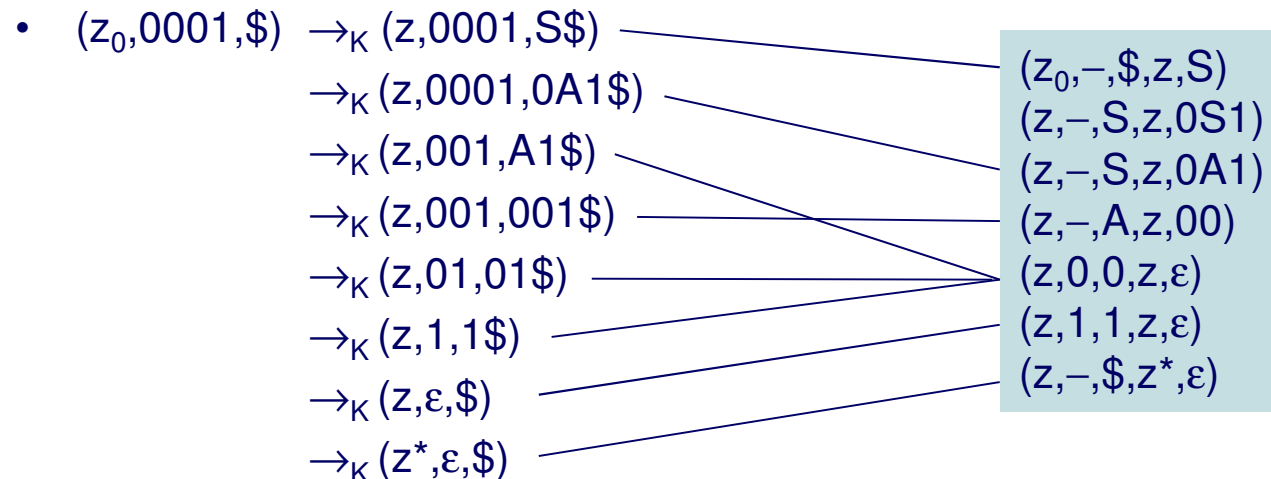
$$V = \{ S,A \}$$

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 :



► Implikation a) (kontextfrei \rightarrow 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 $(z, -, \$, z^*, \varepsilon)$
 - $(z, x, x, z, \varepsilon)$ 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)

Kapitel 2: Formale Sprachen

Deterministische Kellerautomaten

► 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 „sollte“ Transitionen geben, in denen ein „-“ vorkommt.

Das formulieren wir gleich genauer.

Jedenfalls kann K bei einer Eingabe w wegen der „- -Transitionen“ ggf. mehr als $|w|$ Rechenschritte machen (im Gegensatz zum endlichen Automaten).

Kapitel 2: Formale Sprachen

Deterministische Kellerautomaten

► Beispiel 5

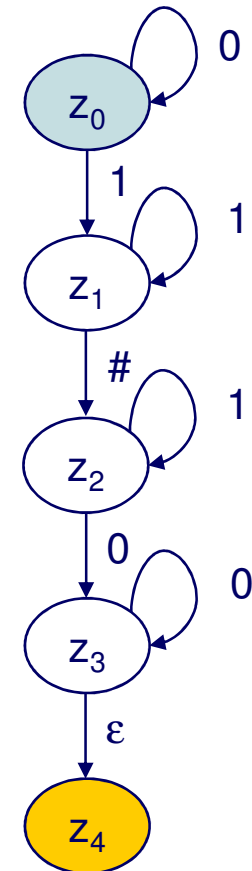
- $L = \{ 0^n \cdot 1^m \cdot \# \cdot 1^m \cdot 0^n \mid n, m > 0 \}$

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

Transitionen in Δ :

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

- $(z_2, 1, 1, z_2, \epsilon)$
- $(z_2, 0, 0, z_3, \epsilon)$
- $(z_3, 0, 0, z_3, \epsilon)$
- $(z_3, -, \$, z_4, \epsilon)$



Kapitel 2: Formale Sprachen

Deterministische Kellerautomaten

► Beispiel 5

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

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

Transitionen in Δ :

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

- $(z_2, 1, 1, z_2, \epsilon)$
- $(z_2, 0, 0, z_3, \epsilon)$
- $(z_3, 0, 0, z_3, \epsilon)$
- $(z_3, -, \$, z_4, \epsilon)$

Grobe Beschreibung:

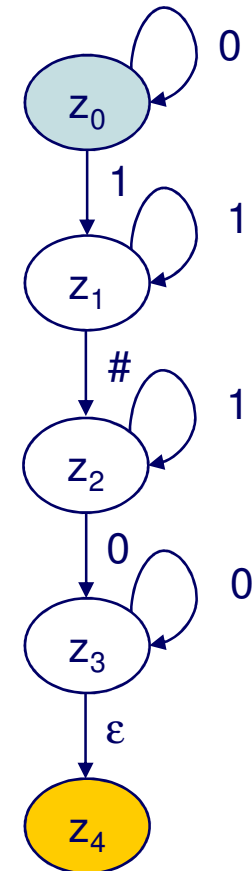
z_0 : **push** Eingabe-0en, auch die erste Eingabe-1, dann ...

z_1 : **push** Eingabe-1en, auch das erste Eingabe-#, dann ...

z_2 : solange mögl., lies Eing.-1, **pop** Keller-1, auch E-0/K-0, dann ...

z_3 : solange mögl., lies Eing.-0, **pop** Keller-0; wenn Keller leer, dann ...

z_4 : akzeptiere und **stop**



► Anmerkungen (weiter)

- Im Unterschied zum Kapitel Reguläre Sprachen ist hier alles ein wenig komplizierter, weil
 - man einen ND-Kellerautomaten nicht immer in einen gleichwertigen deterministischen Kellerautomaten umbauen kann, da ... es kontextfreie Sprachen L gibt, für die es gar keinen deterministischen Kellerautomaten K mit $L(K) = L$ gibt,
 - man nicht einmal algorithmisch entscheiden kann, ob es zu einer kontextfreien Sprache L einen deterministischen Kellerautomaten K mit $L(K) = L$ gibt.
- Zusätzlich wird das Ganze dadurch ein wenig schwieriger, dass man einem deterministischen Kellerautomaten K nicht ohne weiteres ansieht, dass man tatsächlich mit seiner Hilfe das Wortproblem für die Sprache $L(K)$ in Zeit $O(n)$ lösen kann.

Kapitel 2: Formale Sprachen

Deterministische Kellerautomaten

▶ Stellenwert 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 $(z, x, z') \rightarrow ((z, x, \$, z', \epsilon))$ leicht zum deterministischen Kellerautomaten erweitert werden kann.

▶ Beispiel

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

► Stellenwert 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 Kellerautomat K mit $L(K) = L$ mindestens eine Transition der Form $(z, -, x, z', v)$ enthalten muss.
- Praktisch ausgedrückt:
Ein deterministischer Kellerautomat muss das aktuell gelesene Zeichen auf dem Eingabeband „vorübergehend ignorieren können dürfen“; sonst kann er unnötigerweise für viele Sprachen das Wortproblem nicht lösen.

Man will aber gerade – wo immer möglich – deterministische Kellerautomaten dazu verwenden, weil sie das so schnell erledigen ...

► Stellenwert des Ignorierens „–“, Beweisidee

- Sei $L = L_1 \cup L_2$ mit $L_1 = \{ 0^n \circ 1^m \circ \# \circ 1^m \circ 0^n \circ \# \mid n, m \in \mathbb{N} \}$ und $L_2 = \{ 0^n \circ 1^m \circ \# \circ 0^n \circ \# \mid n, m \in \mathbb{N} \}$
- 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:

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

- $(z_2, 1, 1, z_2, \varepsilon)$
- $(z_2, 0, 0, z_3, \varepsilon)$
- $(z_3, 0, 0, z_3, \varepsilon)$
- $(z_3, \#, \$, z_4, \varepsilon)$

- $(z_2, 0, 1, z_5, \varepsilon)$
- $(z_5, -, 1, z_5, \varepsilon)$
- $(z_5, -, 0, z_3, \varepsilon)$
- $(z_5, -, \$, z_3, \varepsilon)$

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. ($0 - 0$: „verzögerter Ausgleich“)

► 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 (c eine Konstante unabhängig von w , abhängig von K) 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 – Beweisideen (1)
 - 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“.
 - Ansonsten gibt es aber gewisse Komplikationen:
 - Der die Sprache L akzeptierende DKA K trifft bezüglich der Lösung des Wortproblems nur eine *Halbentscheidung*:
Er „sagt ja“, wenn $w \in L$.
Wenn $w \notin L$, bleibt er irgendwo außerhalb des akzeptierenden Endzustandes stecken oder rechnet unendlich lange weiter.
 - Selbst wenn er w akzeptiert (also wenn $w \in L$) könnte er sich vorher evtl. beliebig lange mit ignorierenden Transitionen beschäftigen.

- ▶ Effizienzvorteile deterministischer Kellerautomaten – Beweisideen (2)
 - Zum Trödeln mit ignorierende Transitionen:
 - Es gibt eine vom deterministischen Kellerautomaten K abhängige Konstante $c(K)$ derart, dass K beim Akzeptieren eines Wortes w dies stets innerhalb von höchstens $c(K) \cdot |w|$ Schritten erledigt.
Der Beweis ist nicht ganz trivial (\rightarrow siehe sog. Parsing-Theorie).
 - Ausbau der Halbentscheidung des Wortproblems $w \in L?$ zur Entscheidung:
 - Man schreibt ein Programm, das den Kellerautomaten schrittweise ausführt und die Schritte mitzählt.
Akzeptiert er, sagt das Programm JA.
Stoppt er in einem nicht akzeptierenden Zustand oder macht er mehr als $c(K) \cdot |w|$ Schritte, sagt das Programm NEIN.

Kapitel 2: Formale Sprachen

Deterministische Kellerautomaten

► 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 bei $c(K)=100$ ($O(n)$)	CYK-Algorithmus ($O(n^3)$)
100	0.00001 s	0.001 s
1000	0.0001 s	1 s
10000	0.001 s	≥ 16 min

Kapitel 2: Formale Sprachen

Deterministische Kellerautomaten

► Zum Schluss: eine Verständnisübung zum Umgang mit Kellerautomaten

Gegeben seien

- ein deterministischer Kellerautomat K_1 und
- ein („echt“) nicht-deterministischer Kellerautomat K_2
- mit $Z = \{ z_0, z_1, z_2 \}$, $\Sigma = \Gamma = \{ a \}$, mit 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 ?

Kapitel 2: Formale Sprachen

Deterministische Kellerautomaten

► Zum Schluss: eine Verständnisübung zum Umgang mit Kellerautomaten

Gegeben seien

- ein deterministischer Kellerautomat K_1 und
- ein („echt“) nicht-deterministischer Kellerautomat K_2
- mit $Z = \{ z_0, z_1, z_2 \}$, $\Sigma = \Gamma = \{ a \}$, mit 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 ?
- Antwort: beide $\{ a^{2n} \mid n \in \mathbb{N} \}$