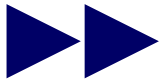
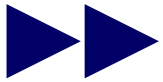
Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Gliederung der Vorlesung

0. Grundbegriffe
1. Formale Sprachen/Automatentheorie
    - 1.1. Grammatiken
    - 1.2. Reguläre Sprachen
    - **1.3. Kontextfreie Sprachen**
2. Berechnungstheorie
    - 2.1. Berechenbarkeitsmodelle
    - 2.2. Die Churchsche These
    - 2.3. Unentscheidbarkeit
3. Komplexitätstheorie
    - 3.1. Nicht-deterministische Turing Maschinen
    - 3.2. Komplexitätsmaße
    - 3.3. Das P=NP? Problem



Kontextfreie Sprachen

algorithmischer Aspekt

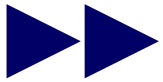
Beispiel: prüfen, ob der Quellcode eines Programm syntaktisch korrekt ist

Annahme: unser Rechner schafft 10^9 Operationen pro Sekunde

Länge des Quellcodes	CYK-Algorithmus
100 Zeichen	0.001s
1000 Zeichen	1s
10000 Zeichen	≥ 16 min

... man sollte über Alternativen nachdenken !!!

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Kontextfreie Sprachen

Fakt

endliche Automaten können nicht alle kontextfreien Sprachen akzeptieren

Frage/Ansatz

Wie viel muß ein endlicher Automat mehr können?

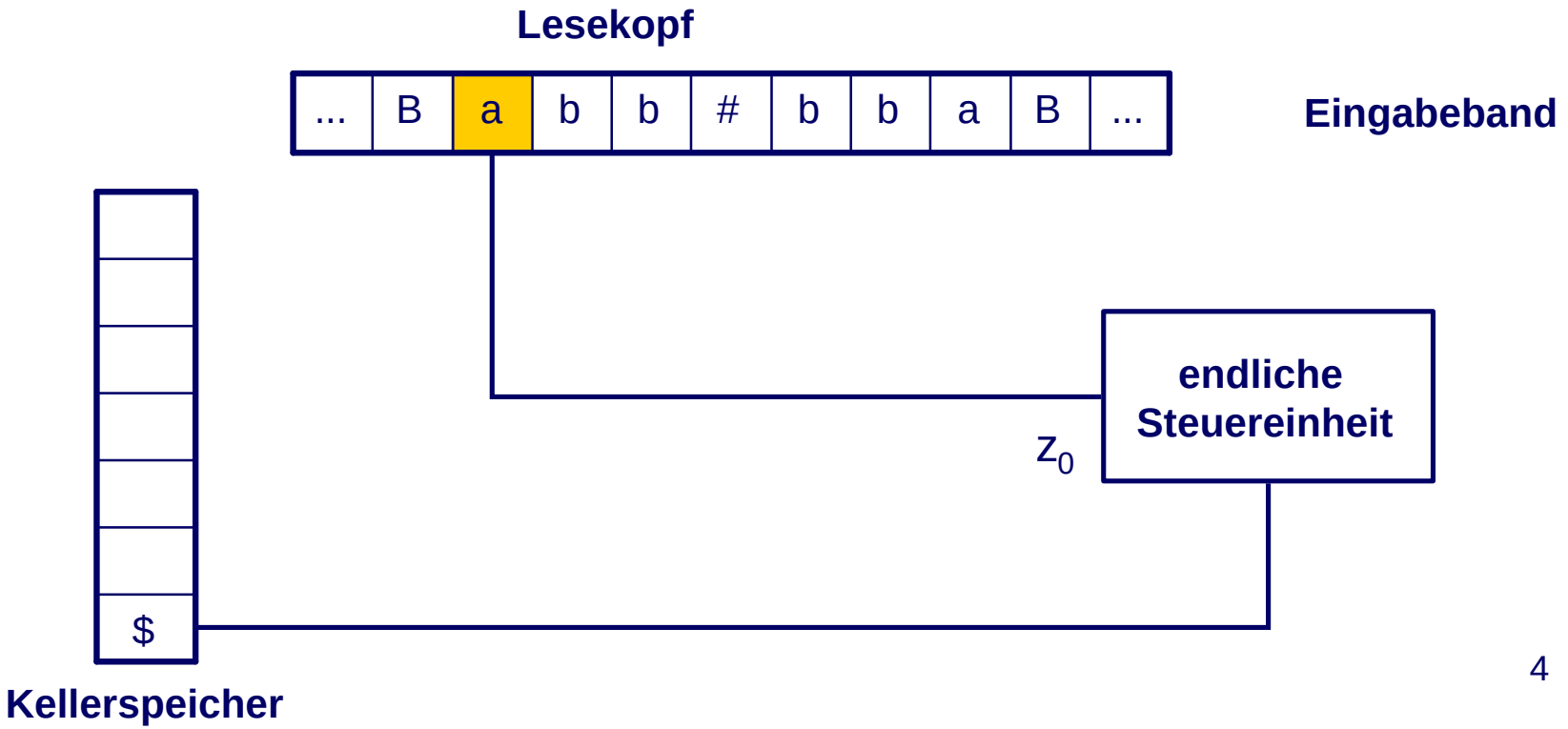
Randbedingung

... wir wollen nicht „mit Kanonen auf Spatzen schießen“

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

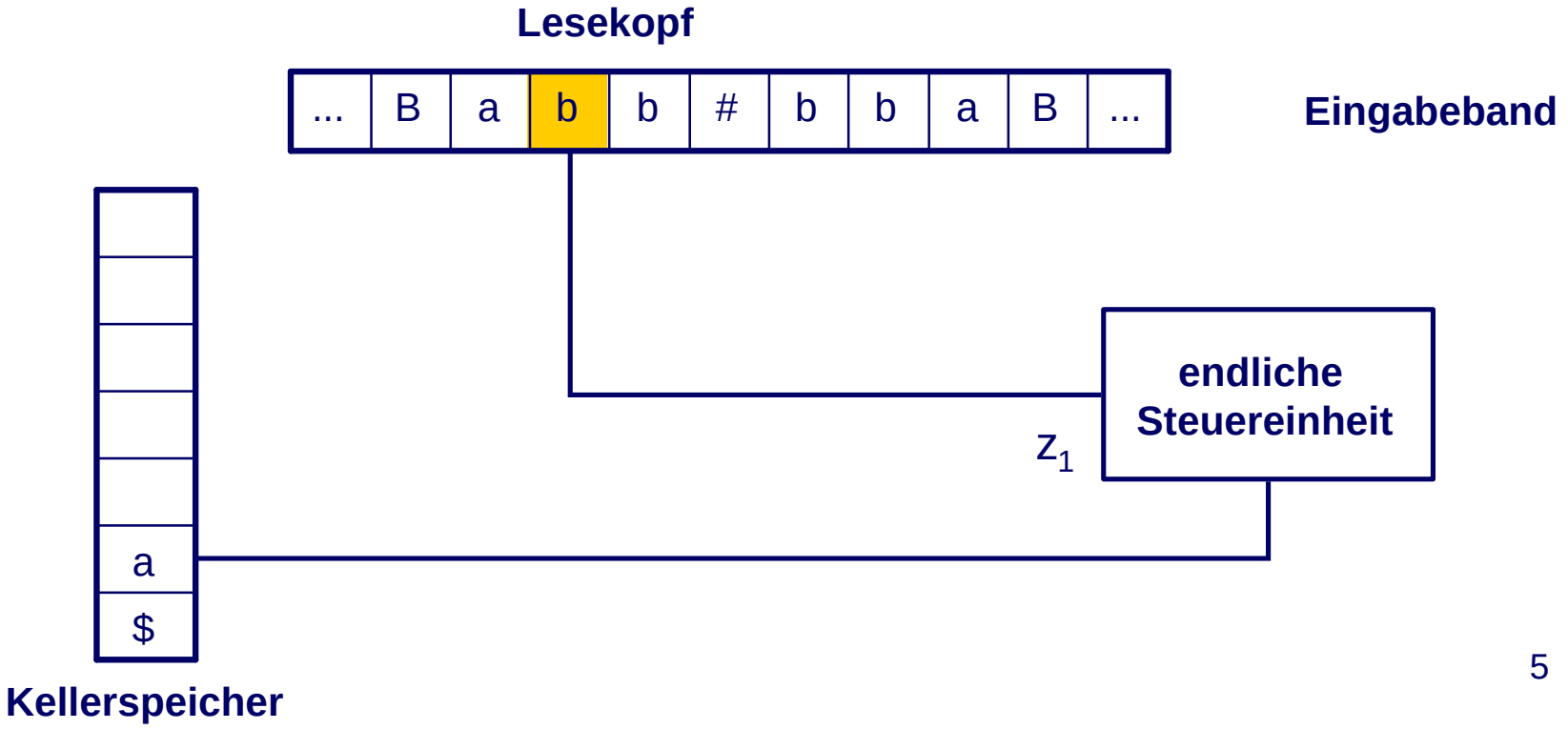
Kontextfreie Sprachen

Beispiel L = { w#$\underline{w}$ | w ∈ { a,b }* \ { ε }; $\underline{w}$ ist die gespiegelte Version von w }



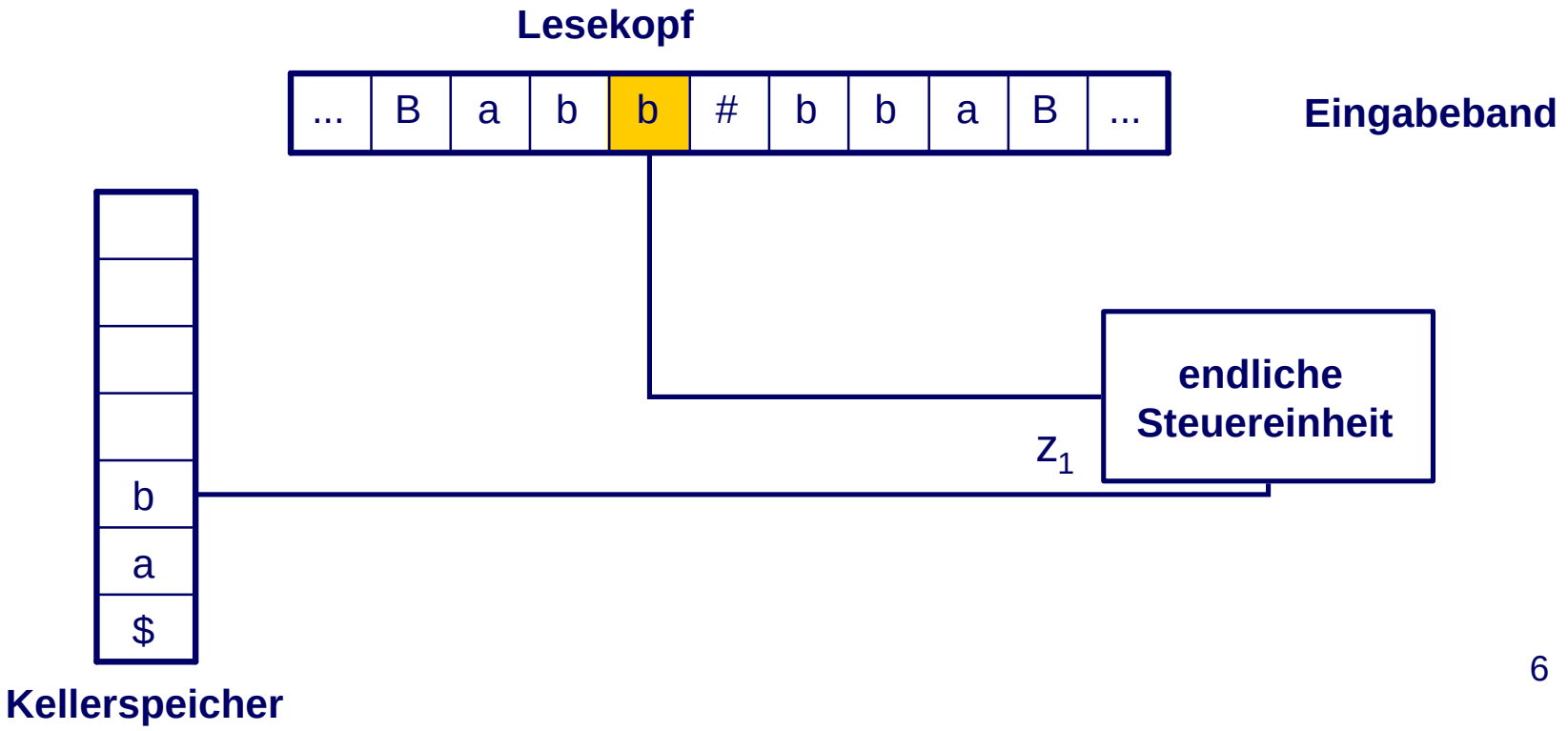
Kontextfreie Sprachen

Beispiel L = { w#$\underline{w}$ | w ∈ { a,b }* \ { ε }; $\underline{w}$ ist die gespiegelte Version von w }



Kontextfreie Sprachen

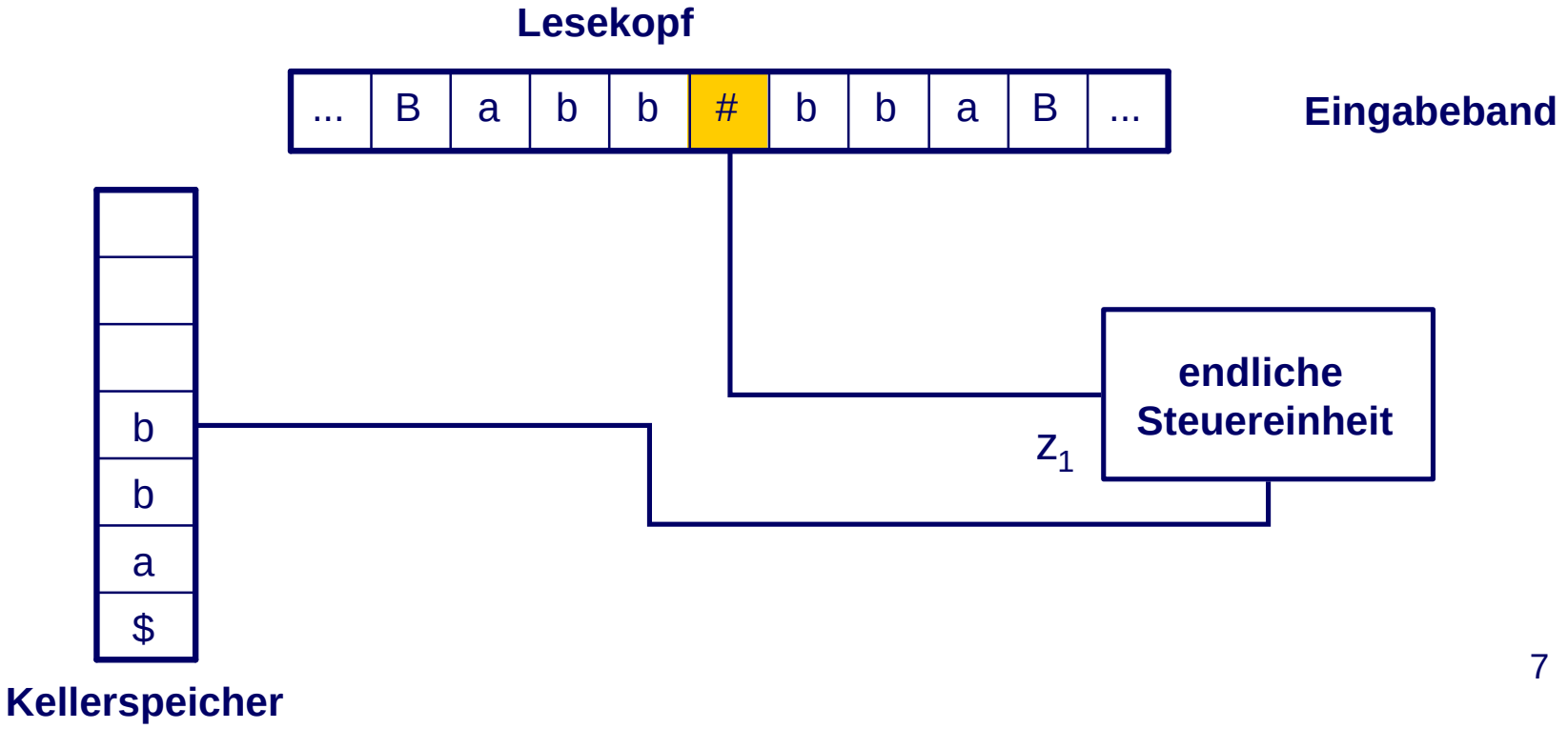
Beispiel L = { w#$\underline{w}$ | w ∈ { a,b }* \ { ε }; $\underline{w}$ ist die gespiegelte Version von w }



Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Kontextfreie Sprachen

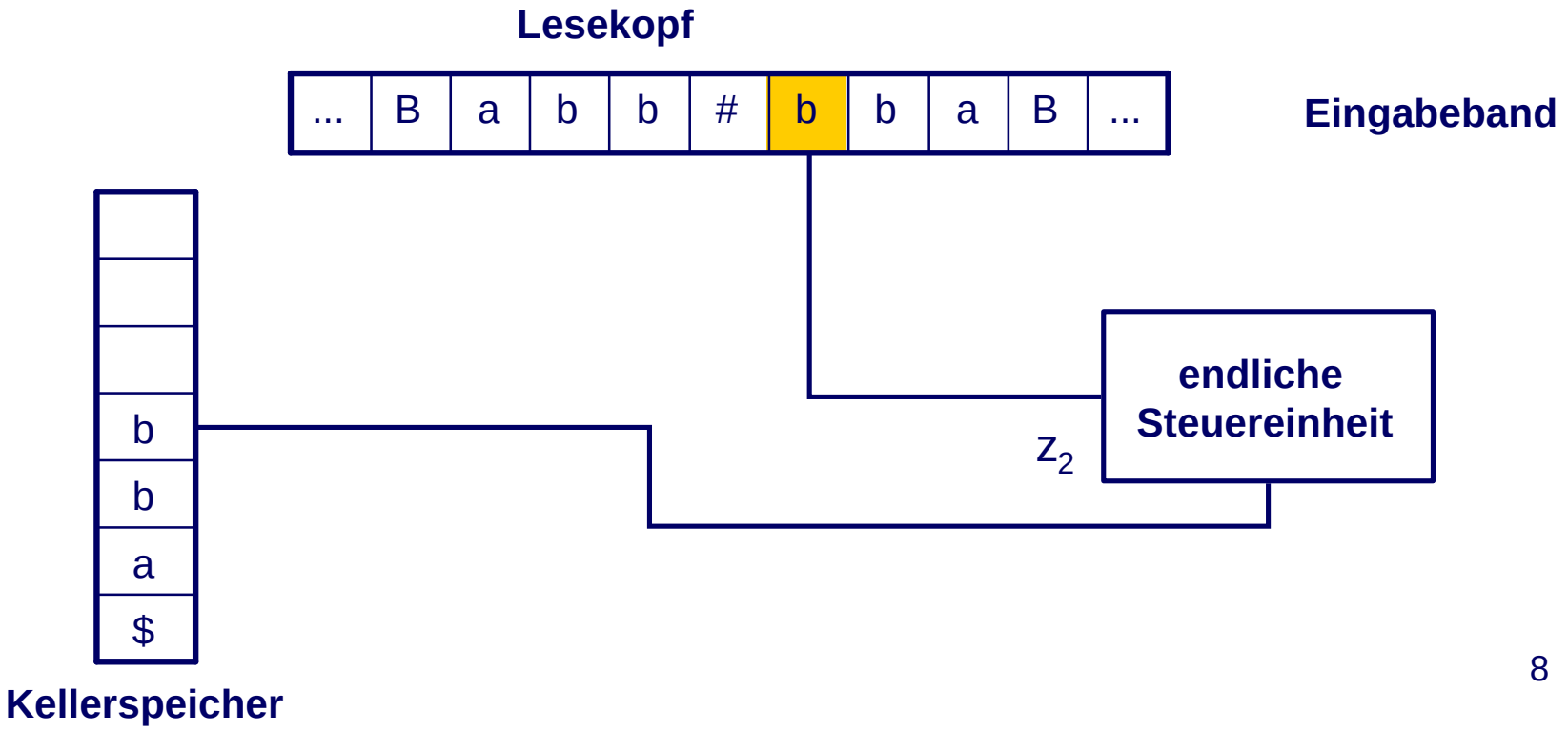
Beispiel L = { w#$\underline{w}$ | w ∈ { a,b }* \ { ε }; $\underline{w}$ ist die gespiegelte Version von w }



Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

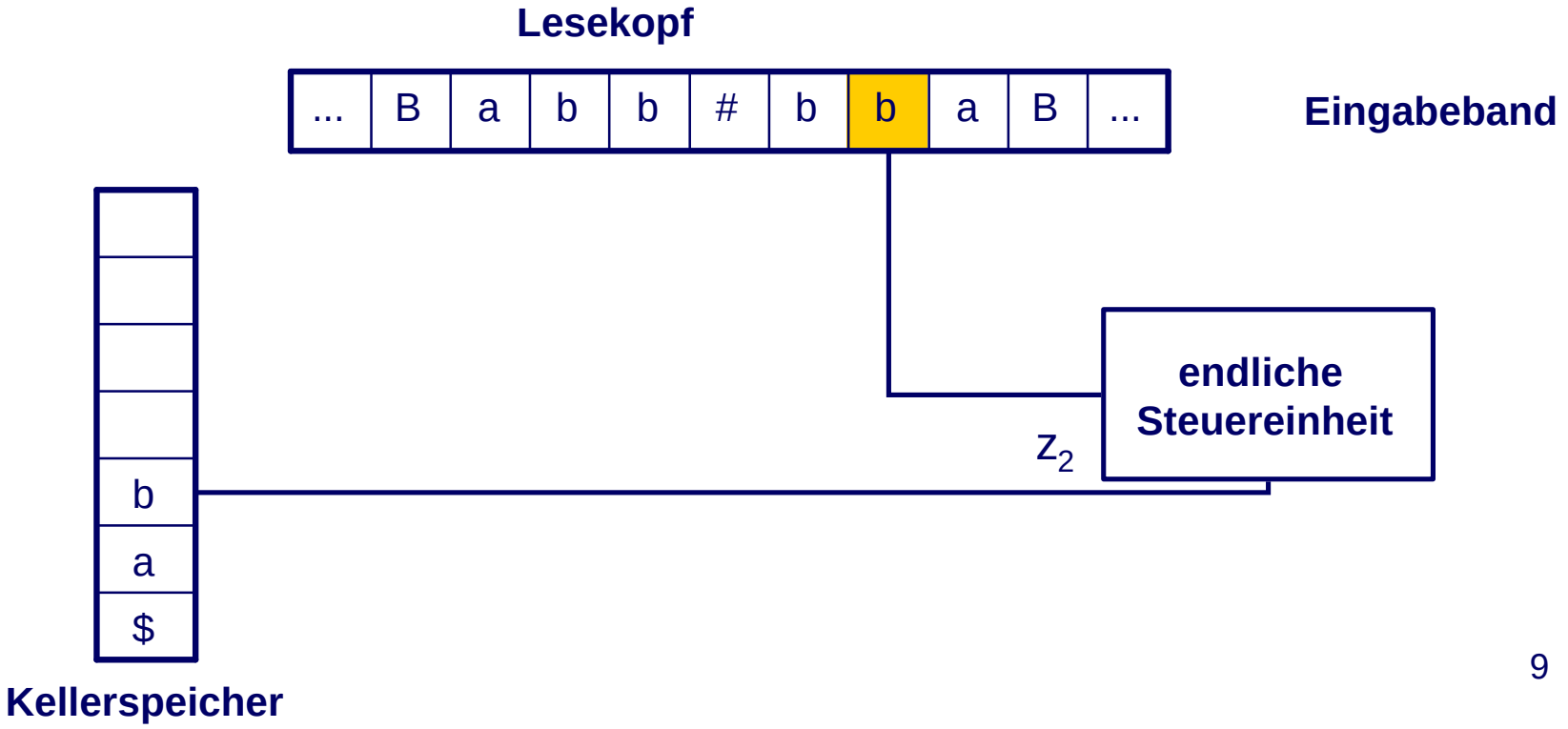
Kontextfreie Sprachen

Beispiel L = { w#$\underline{w}$ | w ∈ { a,b }* \ { ε }; $\underline{w}$ ist die gespiegelte Version von w }



Kontextfreie Sprachen

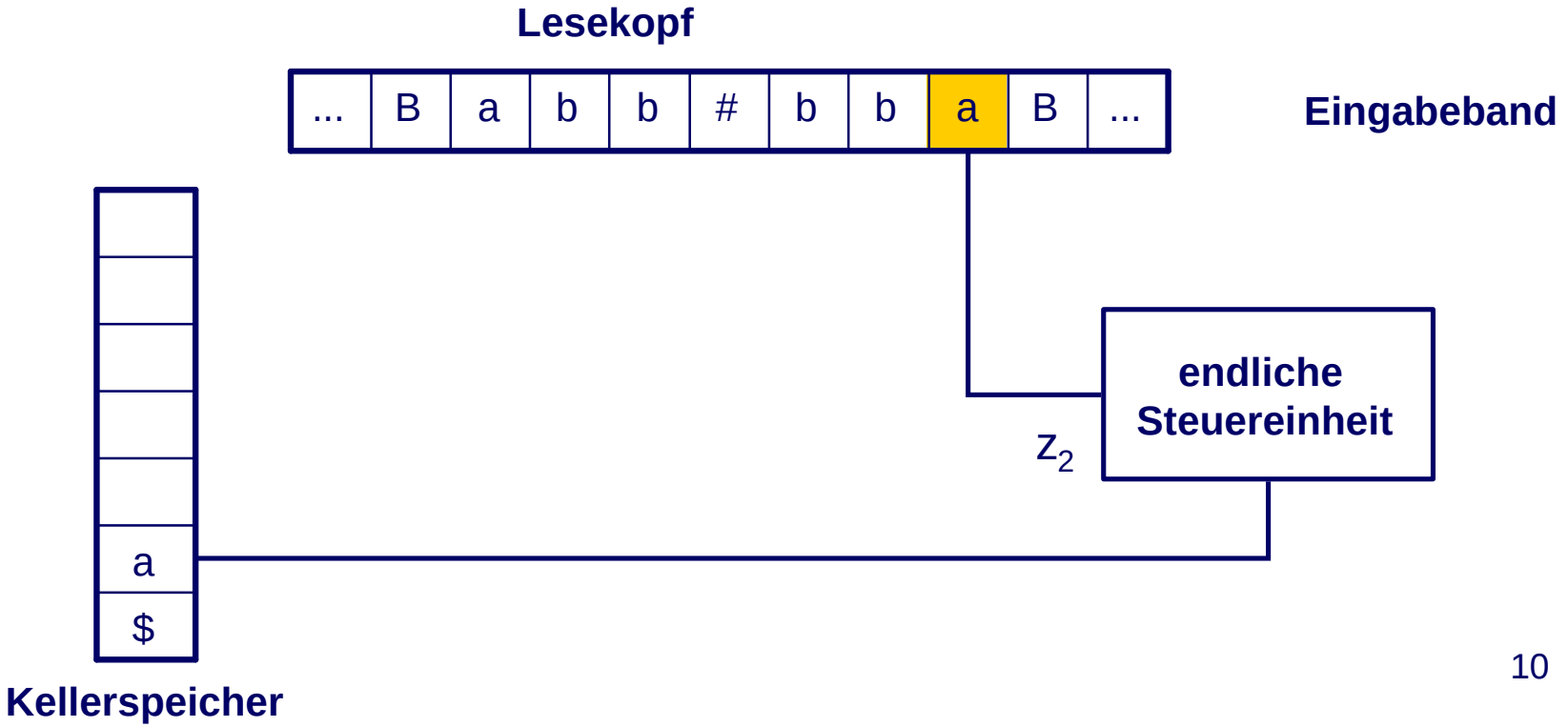
Beispiel $L = \{ w\#\underline{w} \mid w \in \{ a,b \}^* \setminus \{ \varepsilon \};$ $\underline{w}$ ist die gespiegelte Version von $w \}$



Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

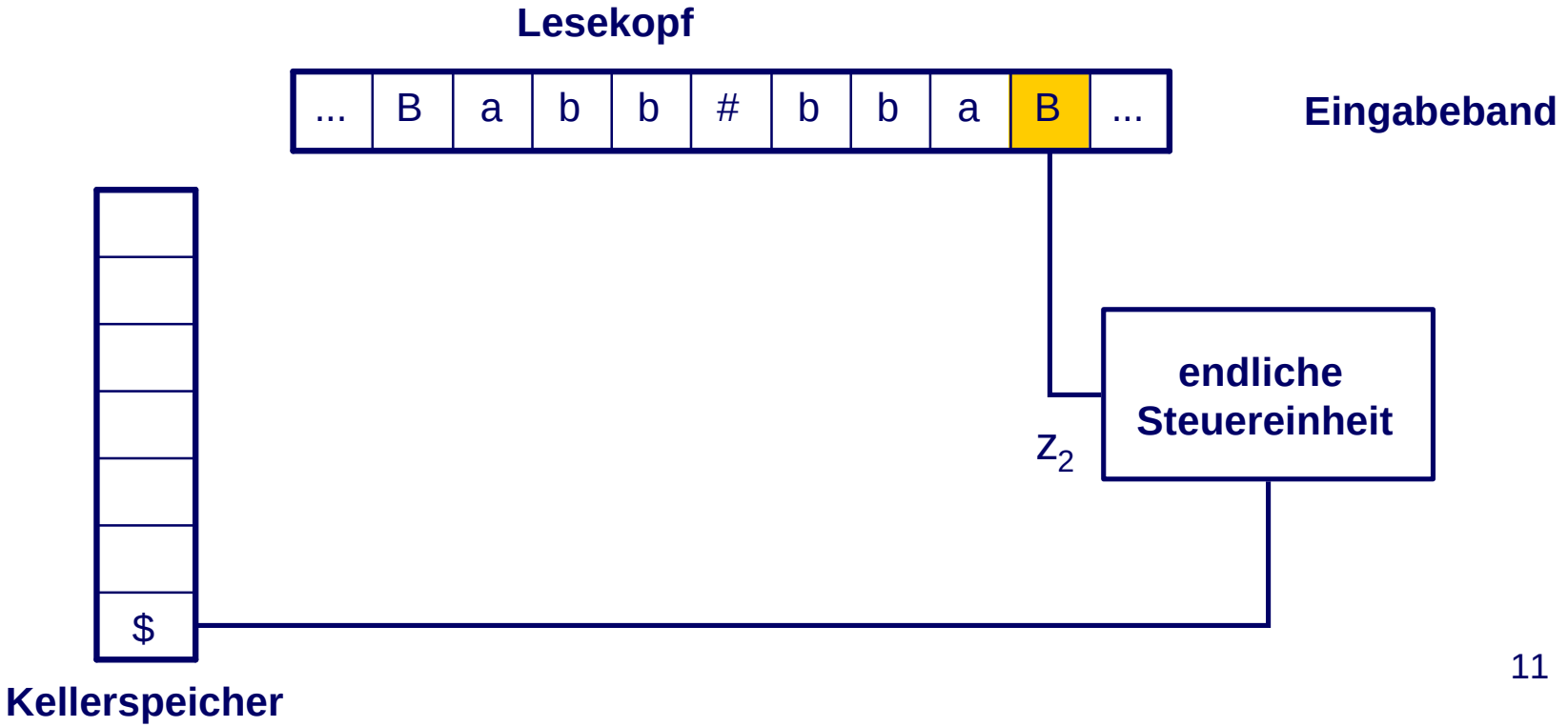
Kontextfreie Sprachen

Beispiel L = { w#$\underline{w}$ | w ∈ { a,b }* \ { ε }; $\underline{w}$ ist die gespiegelte Version von w }



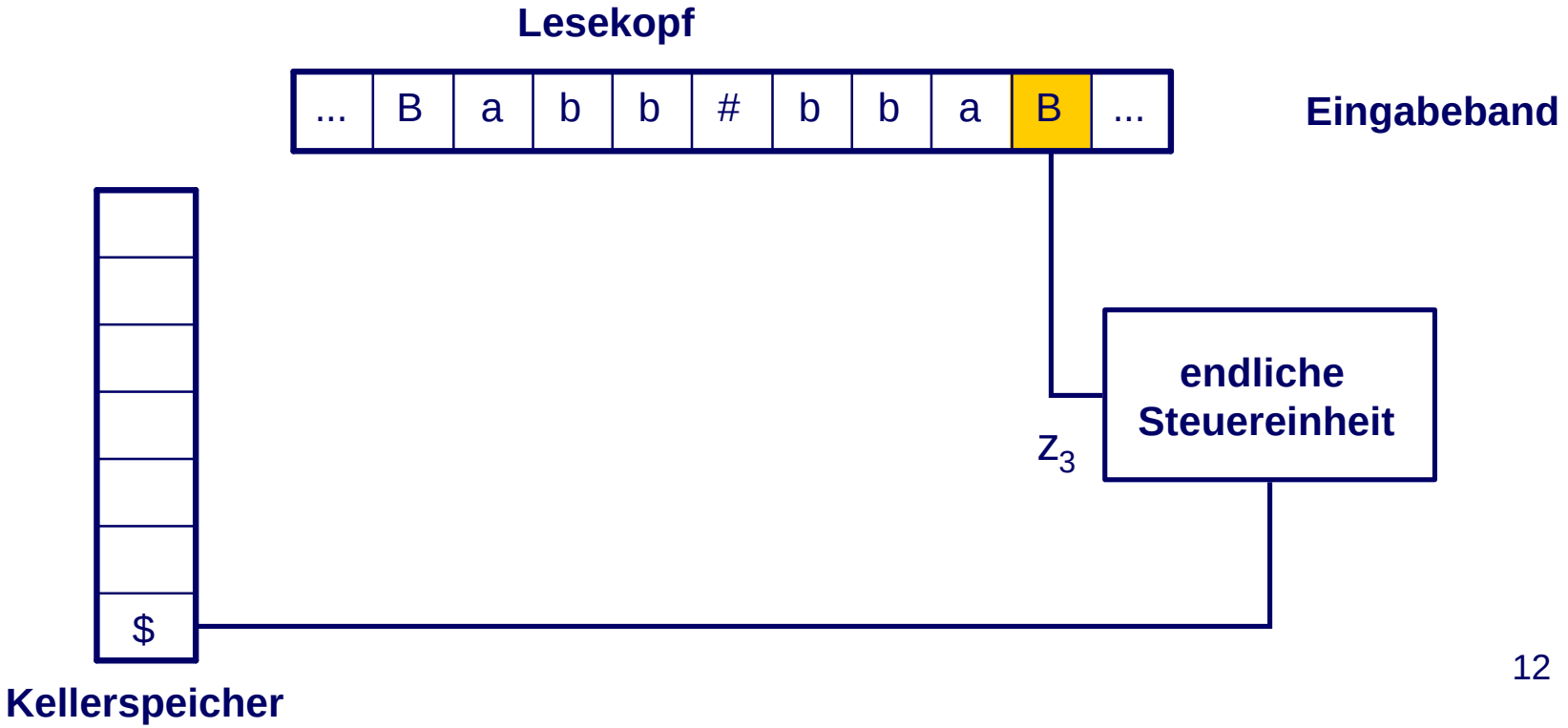
Kontextfreie Sprachen

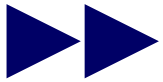
Beispiel L = { w#$\underline{w}$ | w ∈ { a,b }* \ { ε }; $\underline{w}$ ist die gespiegelte Version von w }



Kontextfreie Sprachen

Beispiel L = { w#$\underline{w}$ | w ∈ { a,b }* \ { ε }; $\underline{w}$ ist die gespiegelte Version von w }



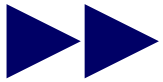


Kontextfreie Sprachen

algorithmischer Aspekt

solcherart Automaten können benutzt werden, um das Membership-Problem für kontextfreie Sprachen (/* in vielen Fällen auch effizient */) zu lösen

... notwendig, Wissen über Kellerspeicher „auffrischen“



Kontextfreie Sprachen

Kellerspeicher (/* angepaßt an unsere Bedürfnisse */)

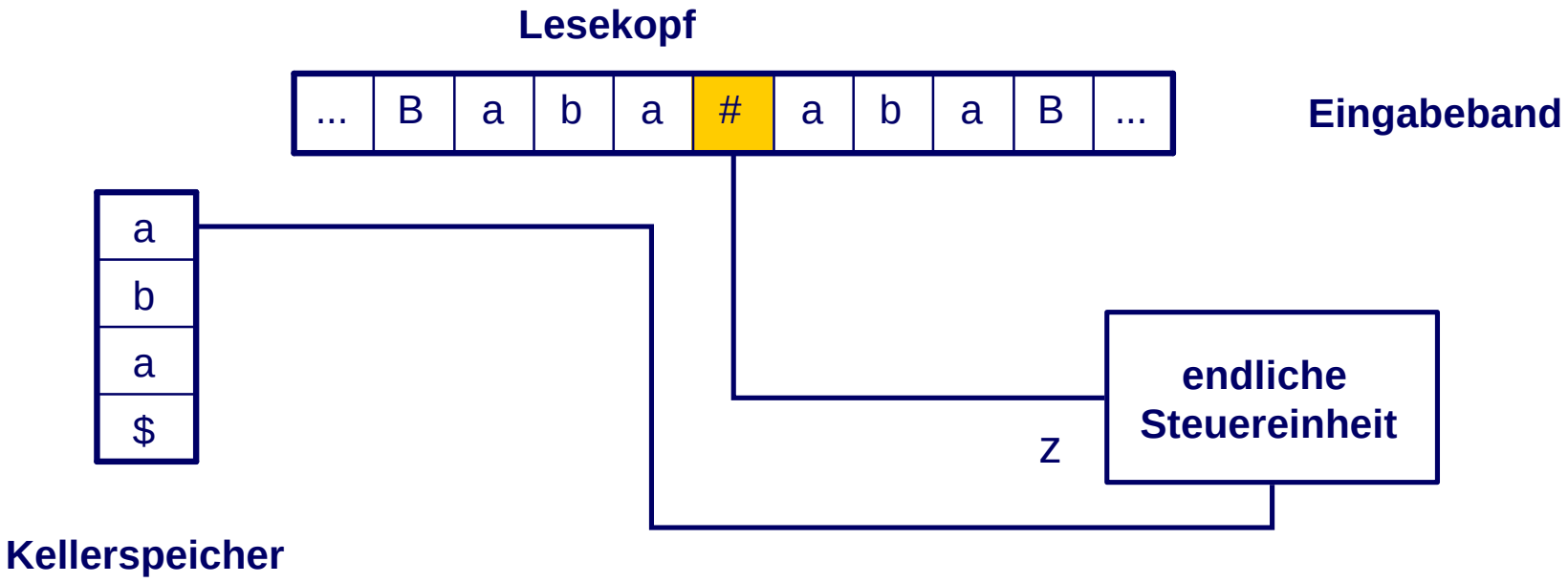
- implementiert das LIFO-Prinzip
- erlaubt Datenelemente zu speichern
- eingeschränkter Zugriff: es kann nur auf das zuletzt gespeicherte Datenelement zugegriffen werden

- Keller K; initial steht ein $ im Keller K
- endliches Kelleralphabet Γ mit $\$ \notin \Gamma$
- Operationen: pop, push

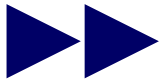
Eigenschaften:

```
pop(push(x,K)) = K
pop($) = Error
```

Kontextfreie Sprachen

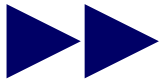


- auf dem Eingabeband steht das Wort w, für welches zu entscheiden ist, ob es zur Sprache L gehört oder nicht
- das Wort w wird zeichenweise von links nach rechts gelesen
- falls sich der Kellerautomat nach vollständiger Verarbeitung des Wortes w in einem Endzustand befindet, so gehört w zu L; andernfalls gehört w nicht zu L



Kontextfreie Sprachen

- das Eingabeband ist in Felder unterteilt; jedes Feld enthält ein Zeichen
- endliche Steuereinheit; mit einem Lesekopf und einem Kellerspeicher verbunden
  - bekommt Information über den Inhalt eines Feldes und mittels pop das zuletzt im Kellerspeicher gespeicherte Zeichen
  - befindet sich in einem von endlich vielen Zuständen
  - legt in Abhängigkeit vom aktuellen Zustand, vom Inhalt des aktuell gesehenen Feldes und vom zuletzt im Kellerspeicher gespeicherte Zeichen fest, was im nächsten Arbeitsschritt passiert
- ein Arbeitsschritt besteht aus:
  - Bewegung des Lesekopfs um ein Feld nach rechts bzw. Lesekopf nicht bewegen
  - Speichern von endlich vielen Zeichen im Kellerspeicher
  - Festlegung des nächsten Zustands



Kontextfreie Sprachen

es sei G eine kontextfreie Grammatik

... man verwendet einen Kellerautomaten, um für ein gegebenes Wort w zu überprüfen, ob es eine Linksableitung für w gibt

Beispiel $L = \{ w\#\underline{w} \mid w \in \{ a,b \}^* \setminus \{ \varepsilon \};\ \underline{w}$ ist die gespiegelte Version von $w \}$

$\Sigma = \{ a, b, \# \}$
$V = \{ S \}$
S

$S \rightarrow_G aSa \rightarrow_G ab\#ba$

$S \rightarrow a\#a \mid b\#b$
$S \rightarrow aSa \mid bSb$

Kontextfreie Sprachen

Beispiel L = { w#$\underline{w}$ | w ∈ { a,b }* \ { ε }, $\underline{w}$ ist die gespiegelte Version von w }

Σ = { a, b }
V = { S }
S

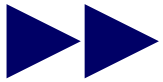
S → a#a | b#b
S → aSa | bSb

S $\rightarrow_G$ aSa $\rightarrow_G$ ab#ba

Keller	Eingabewort
$	ab#ba
$\underline{S}$$	ab#ba
$\underline{a}$Sa$	ab#ba
$\underline{S}$a$	b#ba
$\underline{b}$#ba$	b#ba
$\underline{\#}$ba$	#ba
$\underline{b}$a$	ba
$\underline{a}$$	a
$	

Hinweis: zuletzt im Kellerspeicher gespeichertes Zeichen ist unterstrichen

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Kontextfreie Sprachen

Beispiel $L = \{ a^n b^n \mid n \geq 1 \}$

$\Sigma = \{ a, b \}$
$V = \{ S \}$
S

$S \rightarrow ab \mid aSb$

$S \rightarrow_G aSb \rightarrow_G aabb$

Keller	Eingabewort
$	aabb
S$	aabb
aSb$	aabb
Sb$	abb
abb$	abb
bb$	bb
b$	b
$	

Kontextfreie Sprachen

Beispiel $L = \{ a^n b^n \mid n \geq 1 \}$

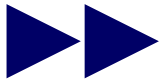
$\Sigma = \{ a, b \}$
$V = \{ S \}$
S

$S \rightarrow ab \mid aSb$

$S \rightarrow_G aSb \rightarrow_G aabb$

Keller	Eingabewort
$	aabb
S$	aabb
aSb$	aabb
Sb$	abb
aSbb$	abb
Sbb$	bb
aSbb$	bb
...	...

... dieser Automat muß zwangsläufig nicht-deterministisch arbeiten



Kontextfreie Sprachen

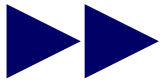
Zusammenhang zwischen nicht-deterministischen Kellerautomaten und kontextfreien Grammatiken

Zu jeder kontextfreien Grammatik G gibt es einen nicht-deterministischen Kellerautomaten C mit L(C) = L(G).

Zu jedem nicht-deterministischen Kellerautomaten C gibt es eine kontextfreie Grammatik G mit L(C) = L(G).

aber ...

nicht-deterministische Kellerautomaten kann man „nicht“ sinnvoll implementieren (/* Laufzeit */)



Kontextfreie Sprachen

Spezialfall: Deterministische Kellerautomaten

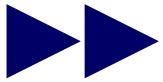
Idee: ... in jeder Situation ist nur eine „Aktion“ möglich

mögliche Situationen sind definiert durch

- den aktuellen Zustand, das Zeichen auf dem Eingabeband, das der Lesekopf sieht, und das zuletzt in den Kellerspeicher geschriebene Zeichen

mögliche Aktionen haben zur Folge, daß sich

- der Zustand und der Inhalt des Kellerspeichers ändert und der Lesekopf ein Zeichen nach rechts geht
- der Zustand und der Inhalt des Kellerspeichers ändert und der Lesekopf auf demselben Zeichen steht



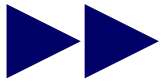
Kontextfreie Sprachen

der „Nutzen“ von deterministische Kellerautomaten

es sei L eine kontextfreie Sprache, für die es einen deterministischen Kellerautomaten D mit L(D) = L gibt

Dann kann man einen Algorithmus angeben, mit dem man zu jedem Wort $w \in \Sigma^*$ in Zeit O(|w|) entscheiden kann, ob $w \in L$ oder $w \notin L$ gilt.

wie das im Detail geht: siehe Vorlesung „Compilerbau“



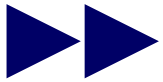
Kontextfreie Sprachen

Beispiel: prüfen, ob der Quellcode eines Programm syntaktisch korrekt ist *)

Annahme: unser Rechner schafft 10^9 Operationen pro Sekunde

Länge des Quellcodes	es gibt einen determ. Kellerautomat für L	es gibt keinen determ. Kellerautomat für L
100 Zeichen	0.0000001 s	0.001 s
1000 Zeichen	0.000001 s	1 s
10000 Zeichen	0.00001 s	≥ 16 min

*) ... alle syntaktisch korrekten Programme bilden eine kontextfreie Sprache

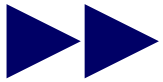


Kontextfreie Sprachen

Zwischenfazit

- die Syntax von Programmiersprachen beschreibt man derart, daß sicher gestellt ist, daß es für die beschriebene Sprache einen deterministischen Kellerautomaten gibt

... das erfordert sowohl Verständnis für die zugrunde liegende Problematik als auch Kreativität !!!



Kontextfreie Sprachen

Begründung (/* Teil 1 */)

Es gibt kontextfreie Sprachen L mit folgender Eigenschaft:

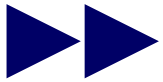
Es gibt keinen deterministischen Kellerautomaten D mit L(D) = L.

Beispiel: $L = \{ w\underline{w} \mid w \in \{ a,b \}^* \setminus \{ \varepsilon \}$, $\underline{w}$ ist die gespiegelte Version von w $\}$

Einsicht:

man muß beim „Design“ aufpassen

... das gilt nicht nur für die Syntax von Programmiersprachen



Kontextfreie Sprachen

Begründung (/* Teil 2 */)

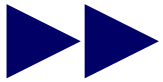
Es gibt keinen Algorithmus, der folgendes leistet:

- der Algorithmus akzeptiert als Eingabe eine kontextfreie Grammatik G
- falls es einen deterministischen Kellerautomat D mit L(D) = L(G) gibt, so liefert der Algorithmus das Resultat „ja“
- andernfalls liefert der Algorithmus das Resultat „nein“

Einsicht:

man kann nicht damit rechnen, daß einem die Arbeit durch „Computer-Tools“ entscheidend vereinfacht wird

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Kontextfreie Sprachen

Begriffe:

- kontextfreie Grammatik, Linksableitung, Chomsky-Normalform
- deterministischer/nicht-deterministischer Kellerautomat

Sätze / Zusammenhänge:

- kf. Grammatik und Chomsky-Normalform
- kf. Grammatik und nicht-det. KA
- det. KA können echt „weniger“
- Pumping-Lemma

Methoden / Techniken / Algorithmen:

- Übersetzung (/* kf. Grammatik → Chomsky-Normalform, kf. Grammatik → nicht-det. KA */)
- CYK-Algorithmus
- Anwendung des Pumping-Lemma