Fragestellungen

- Wie hängen endliche Automaten und kfG's zusammen?
- Sind kfG's kompositional?
- Ist für kfG's das Wortproblem schnell lösbar?
- Was können kfG's nicht?

Abschlusseigenschaften regulärer und kontextfreier Sprachen

	Regulär	Kontextfrei
Vereinigung	$+$	?
Konkatenation	$+$	?
Kleene Hülle	$+$	?
Schnitt	$+$	?
Komplement	$+$	?

Vereinigung

Seien $G_1 = (N_1, T, P_1, S_1)$, $G_2 = (N_2, T, P_2, S_2)$ kfG's mit $N_1 \cap N_2 = \emptyset$.

$$G_1 + G_2 = \begin{array}{l}(N_1 \cup N_2 \cup \{S_{\text{neu}}\}, T, P_1 \cup P_2 \cup \\ \{S_{\text{neu}} ::= S_1 \mid S_2\}, S_{\text{neu}})\end{array}$$

Satz

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

Konkatenation

Seien $G_1 = (N_1, T, P_1, S_1)$, $G_2 = (N_2, T, P_2, S_2)$ kfG's mit $N_1 \cap N_2 = \emptyset$.

$$\boxed{\begin{aligned} G_1 \circ G_2 = \;& (N_1 \cup N_2 \cup \{S_{\text{neu}}\}, T, P_1 \cup P_2 \cup \\ & \{S_{\text{neu}} ::= S_1 S_2\}, S_{\text{neu}}) \end{aligned}}$$

Satz

$L(G_1 \circ G_2) = L(G_1)L(G_2)$.

Kleene Hülle

Sei $G = (N, T, P, S)$ eine kfG.

$$G_* = (N \cup \{S_{\mathrm{neu}}\}, T, P \cup \{S_{\mathrm{neu}} ::= \lambda \mid S S_{\mathrm{neu}}\}, S_{\mathrm{neu}})$$

Satz

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

Abschlusseigenschaften regulärer und kontextfreier Sprachen

	Regulär	Kontextfrei
Vereinigung	+	+
Konkatenation	+	+
Kleene Hülle	+	+
Schnitt	+	(-)
Komplement	+	(-)

Kellerautomaten

Spracherkennung (Syntaxanalyse)

- Algorithmus gesucht, der für $L \subseteq T^*$ (möglichst schnell) entscheidet, ob $w \in L$ (Lösung des Wortproblems)

Grammatik	Automat	Aufwand
rechtslinear	endlich	linear
kontextfrei	?	?

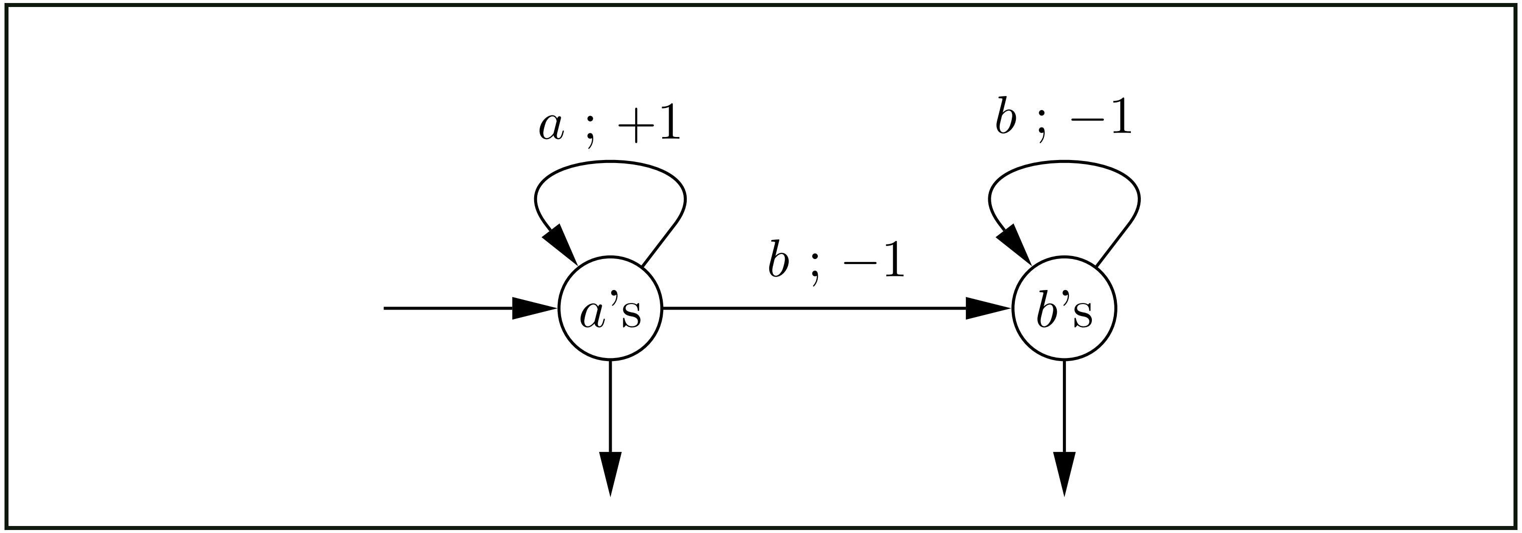
Problem: endliche Automaten können nicht unbeschränkt mitzählen und sich keine unbeschränkt langen Teilwörter merken

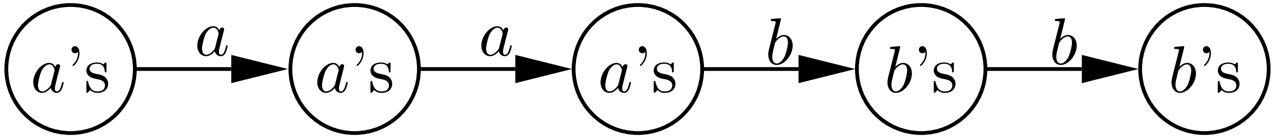
Beispiel: Klammerstrukturen

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$

- nicht regulär
- kontextfrei erzeugt durch $S ::= aSb \mid \lambda$

Idee: Endlicher Automat mit Zähler





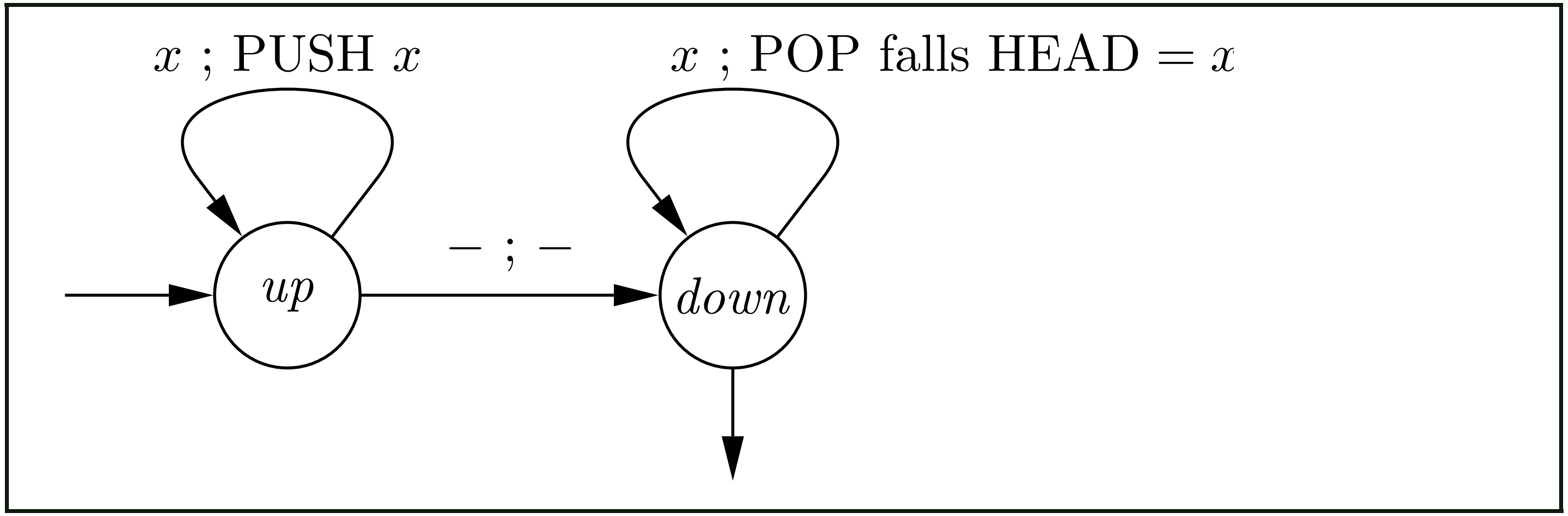
Zähler	0	1	2	1	0

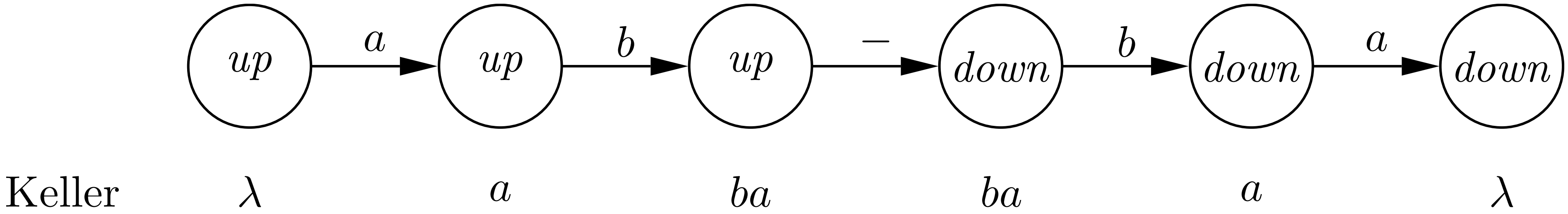
Beispiel: Palindrome

$$L_{pde} = \{w trans(w)\}$$

- nicht regulär
- kontextfrei erzeugt durch $S ::= xSx \mid \lambda$ f.a. $x \in T$

Endlicher Automat mit Keller/Speicher





Nichtdeterministischer Kellerautomat

Idee:

endlichen Automaten mit Zusatzspeicher in Form eines Kellers (Stapel, Stack) mit Speicheroperationen pro Übergang

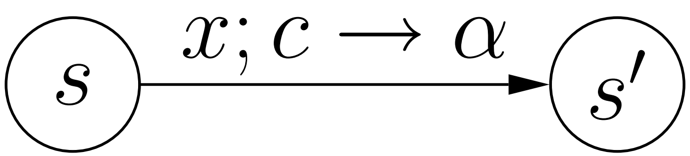
Keller über X:

X^* mit den Operationen push, head, pop (d.h. $push(x,u) = xu$, $head(xu) = x$, $pop(xu) = u$ für alle $x \in X$, $u \in X^*$)

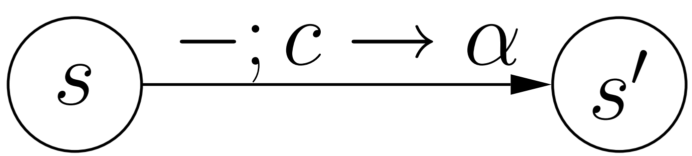
Ein nichtdeterministischer Kellerautomat ist ein System $K = (Z, I, C, d, s_0, F, c_0)$ mit

- Z: endliche Menge von Zuständen,
- I: endliches Eingabealphabet mit $- \notin I$,
- C: endliche Menge von Kellersymbolen,
- $d\colon Z \times (I \cup \{-\}) \times C \rightsquigarrow Z \times C^*$: Zustandsüberführung,
- $s_0 \in Z$: Startzustand,
- $F \subseteq Z$: Endzustände,
- $c_0 \in C$: initiales Kellersymbol.

Graphische Darstellung

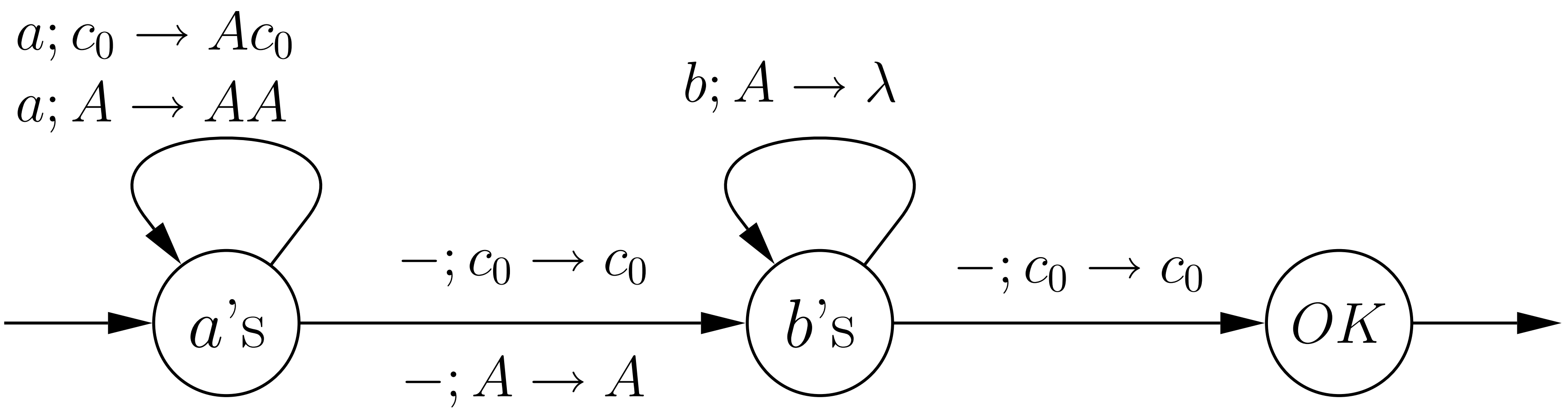


für $(s', \alpha) \in d(s, x, c)$



für $(s', \alpha) \in d(s, -, c)$

Beispiel


$a; c_0 \longrightarrow Ac_0$
$a; A \longrightarrow AA$
$b; A \longrightarrow \lambda$
$-; c_0 \longrightarrow c_0$
$-; A \longrightarrow A$
$-; c_0 \longrightarrow c_0$
a's
b's
OK

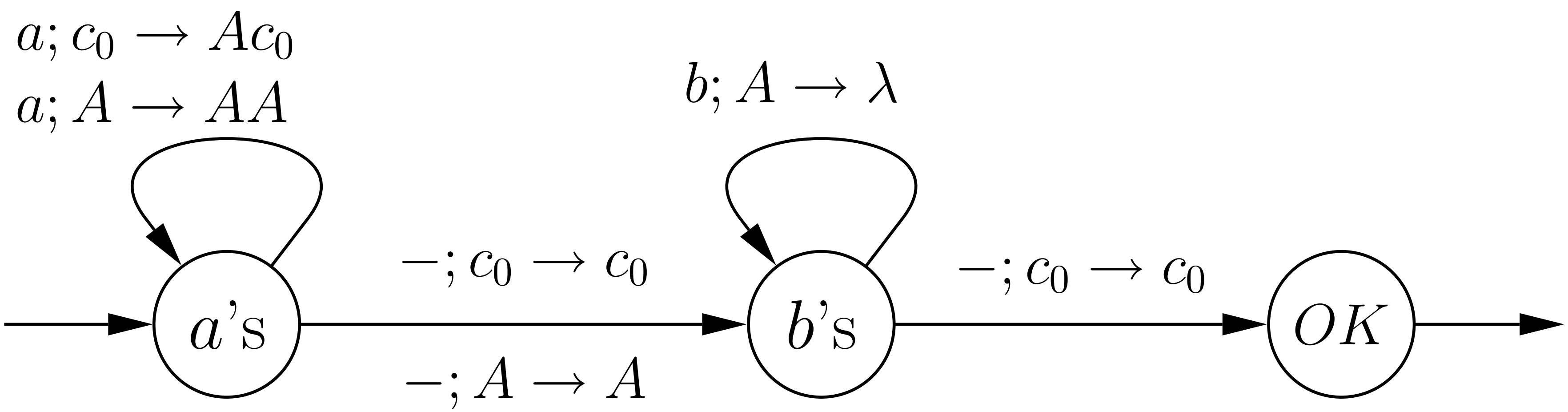
Konfiguration

Eine Konfiguration $con = (s, w, \gamma)$ besteht aus einem Zustand $s \in Z$, einem Wort $w \in I^*$ und einem Kellerwort $\gamma \in C^*$.

Wichtige Konfigurationen:

- Anfangskonfiguration: $con_0 = (s_0, w, c_0)$
- Endkonfiguration: $con_F = (s, \lambda, \gamma)$ mit $s \in F$

Beispiel



- Konfiguration: $(b's, bb, AAc_0)$
- Anfangskonfiguration: $(a's, aabb, c_0)$
- Endkonfiguration: (OK, λ, c_0)

Folgekonfiguration

$(s, xv, c\gamma) \longmapsto (s', v, \alpha\gamma)$, falls $(s', \alpha) \in d(s, x, c)$

$(s, v, c\gamma) \longmapsto (s', v, \alpha\gamma)$, falls $(s', \alpha) \in d(s, -, c)$

- Schreibweise:

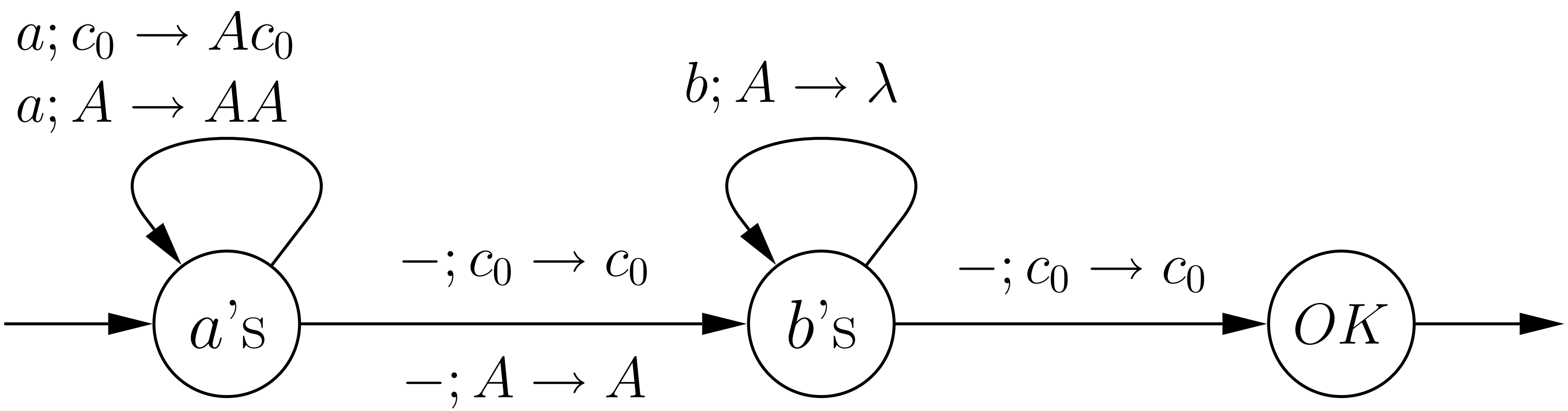
$$\boxed{con = con_0 \longmapsto con_1 \longmapsto \cdots \longmapsto con_n = con'}$$

kann durch

$$\boxed{con \overset{n}{\longmapsto} con'} \text{ oder } \boxed{con \overset{*}{\longmapsto} con'}$$

abgekürzt werden.

Beispiel



Konfigurationsfolge:

$$(a's, abb, Ac_0) \longmapsto (a's, bb, AAc_0) \longmapsto (b's, bb, AAc_0) \longmapsto (b's, b, Ac_0)$$

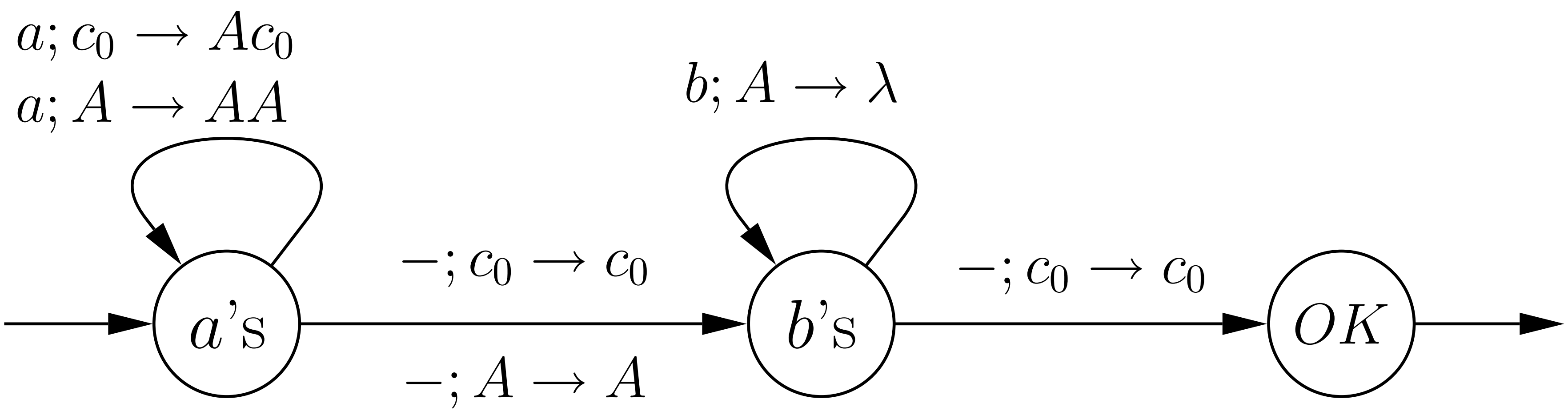
Erkannte Sprache

Gegeben: $K = (Z, I, C, d, s_0, F, c_0)$

Erkannte Sprache

- K erkennt $w \in I^*$, falls $(s_0, w, c_0) \vdash^* (s'', \lambda, \gamma)$ mit $s'' \in F$.
- Die Menge aller von K erkannten Wörter bildet die erkannte Sprache $L(K)$.

Beispiel



Erkannte Sprache: $\{a^n b^n \mid n \in \mathbb{N}\}$

Erkennen von a^2b^2:

$$(a's, aabb, c_0) \vdash\!\!\!\!- (a's, abb, Ac_0) \vdash\!\!\!\!- (a's, bb, AAc_0) \vdash\!\!\!\!-$$
$$(b's, bb, AAc_0) \vdash\!\!\!\!- (b's, b, Ac_0) \vdash\!\!\!\!- (b's, \lambda, c_0) \vdash\!\!\!\!- (OK, \lambda, c_0)$$

Fragestellungen

- Wie hängen endliche Automaten und kfG's zusammen?
- Sind kfG's kompositional?
- Ist für kfG's das Wortproblem schnell lösbar?
- Was können kfG's nicht?

Deterministischer Kellerautomat

- Jede Konfiguration hat höchstens eine Folgekonfiguration.
- Lineare Spracherkennung (wie endl. Aut.)
- Praktischer Einsatz bei Syntaxanalyse von Programmiersprachen
- Nicht jede kontextfreie Sprache wird von deterministischen Kellerautomaten erkannt.