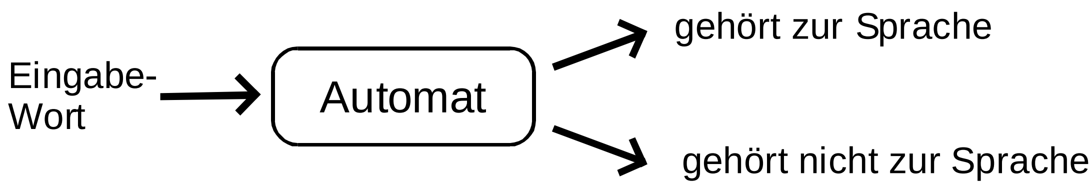
André Winkler, Matr. 2207914

Erkennung und Übersetzung syntaktischer Strukturen

Fokus: Entwicklung von mathematischen Modellen von Rechenmaschinen
Automaten

Eingabemuster



Typen

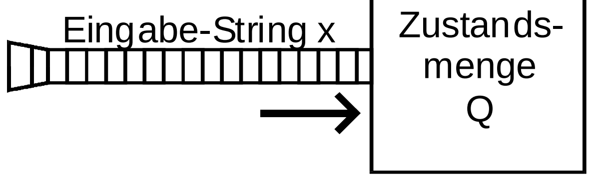
- Turingautomat
- Linearer Automat
- Push-Down-Automat
- Endlicher Automat

Endliche Automaten

- Beschreibung durch 5er-Tupel Д = (Q, $\sum$, δ, q_0, F)

Q	Zustandsmenge
$\sum$	Alphabet
δ	Abbildung von $Q \times \sum$ nach 2^Q
q_0	Startzustand
F	Endzustandsmenge (Teilmenge von Q)

Automat Д



→ Sprache L

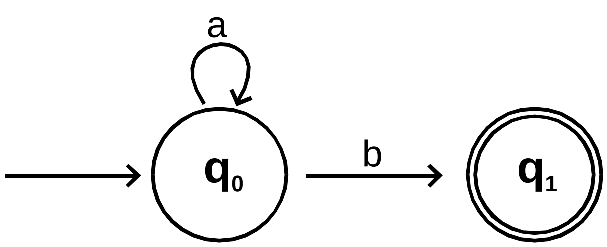
$L(Д_f) = \{x|x \text{ in } \sum^+, x \text{ wird von } Д_f \text{ akzeptiert}\}$

Bedingung:
- Start bei q_0
- Ende bei F
- Der komplette Eingabe-String wird gescannt

Bsp. 1:

$Q = \{q_0, q_1\}$ $\sum = \{a, b\}$ $F = \{q_1\}$

$\delta(q_0, a) = \{q_0\}$
$\delta(q_0, b) = \{q_1\}$
$\delta(q_1, a) = ø$
$\delta(q_1, b) = ø$



Von diesem Automaten akzeptierte Wörter: $a^n b$ mit n = {0; 1; 2; …}

→Eine Sprache wird von einem endlichen Automaten genau dann erkannt, wenn sie mit einer <u>regulären Grammatik</u> erstellt wurde.

Reguläre Grammatik **$G = (N, \sum, P, x_0)$**

$G = (\{S\}, \{a,b\}, P, S)$

Produktionen $S \rightarrow aS$ $S \rightarrow b$

Entsprechender Automat:
$Q = \{q_0, q_1\}$
$F = \{q_1\}$
$\sum = \{a, b\}$

$\delta(q_0, b) = q_1,$ $\delta(q_1, a) = ø,$
$\delta(q_0, a) = q_0,$ $\delta(q_1, b) = ø$

Bsp. 2:

Д = ({q_0, q_1, q_2}, {0, 1}, δ, q_0, {q_2}))

$\delta(q_0, 0) \rightarrow \{ q_2\}$	$\delta(q_0, 1) \rightarrow \{ q_1\}$
$\delta(q_1, 0) \rightarrow \{ q_2\}$	$\delta(q_1, 1) \rightarrow \{ q_0\}$
$\delta(q_2, 0) \rightarrow \{ q_2\}$	$\delta(q_2, 1) \rightarrow \{ q_1\}$

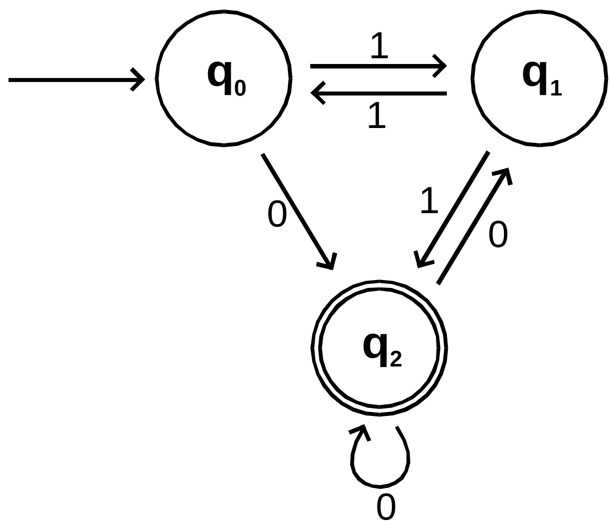
Grammatik G

$G = (\{X_0, X_1, X_2\}, \{0, 1\}, P, X_0)$

$X_0 \rightarrow 1X_1$	$X_1 \rightarrow 1X_0$	$X_2 \rightarrow 1X_1$
$X_0 \rightarrow 0X_2$	$X_1 \rightarrow 0X_2$	$X_2 \rightarrow 0X_2$
$X_0 \rightarrow 0$	$X_1 \rightarrow 0$	$X_2 \rightarrow 0$

Beispielwort: 1100

$X_0 \rightarrow 1X_1 \rightarrow 11X_0 \rightarrow 110X_2 \rightarrow 1100$



→ Zu jedem endlichen Automaten mit einer Sprache L existiert ein vollständig spezifizierter, deterministischer endlicher Automat der Sprache L.

nicht deterministischer, nicht vollständiger Automat $Д_f = (Q, \sum, \delta, q_0, F)$

vollst. spezifizierter deterministischer endlicher Automat $Д_f^{'} = (Q', \sum', \delta', q_0', F')$

Bsp. 3:

geg.: $Д_f = (\{q_0, q_1\}, \{a, b\}, \delta, q_0, \{q_1\})$ (aus Bsp. 1)

ges.: $Д_f^{'} = (Q', \{a, b\}, \delta', q_0', F')$ mit $L(Д_f) = L(Д_f^{'})$,

$Д_f^{'}$...deterministisch und vollständig spezifiziert

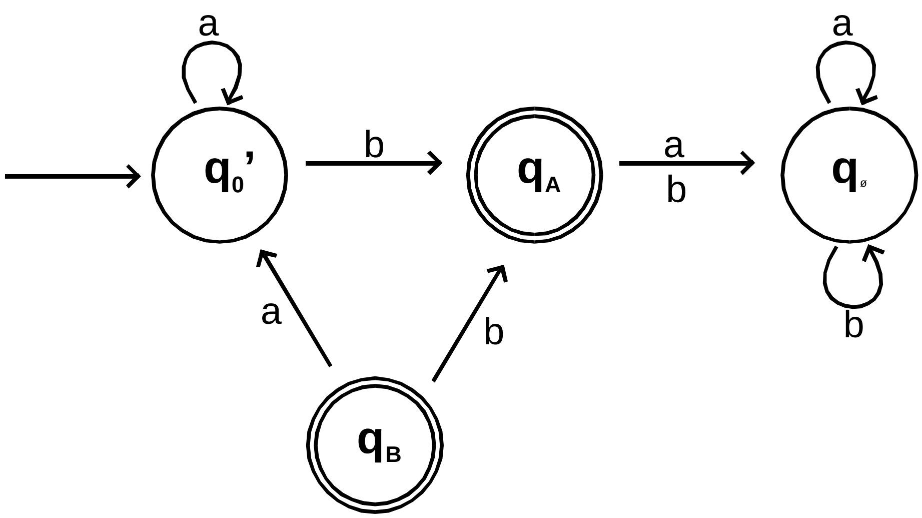
$Q' = \{ø, \{q_0\}, \{q_1\}, \{q_0, q_1\}\}$
$q_0' = \{q_0\}$
$F' = \{ \{q_1\}, \{q_0, q_1\} \}$

$q_ø = ø$ (Trap State)
$q_0' = \{q_0\}$
$q_A = \{q_1\}$
$q_B = \{q_0, q_1\}$

$F' = \{q_A, q_B\}$

$\delta'(q_0', a) = q_0'$
$\delta'(q_A, a) = q_ø$
$\delta'(q_B, a) = q_0'$
$\delta'(q_ø, a) = q_ø$

$\delta'(q_0', b) = q_A$
$\delta'(q_A, b) = q_ø$
$\delta'(q_B, b) = q_A$
$\delta'(q_ø, b) = q_ø$



→ q_B : Unzugänglicher Zustand, kann entfernt werden.

Beispielwörter:	aab	:	funktioniert
	abb	:	kommt zu $q_ø$ → kein akzeptierter Zustand

Für jeden spezifischen, deterministischen endlichen Automaten Д$_f$ gibt es einen äquivalenten Automaten Д$_{fm}$ mit minimaler Anzahl von Zuständen, und dieser ist eindeutig bestimmt (bis auf Umbenennung der Zustände).

Minimierung von endlichen Automaten

Test der Äquivalenz zweier Zustände q_i und q_j:

Bedingungen:

(I) q_i und q_j müssen beide in F oder beide in ┐F sein
(II) für jedes Eingabesymbol a ϵ ∑ müssen $\delta(q_i, a)$ und $\delta(q_j, b)$ gleich sein

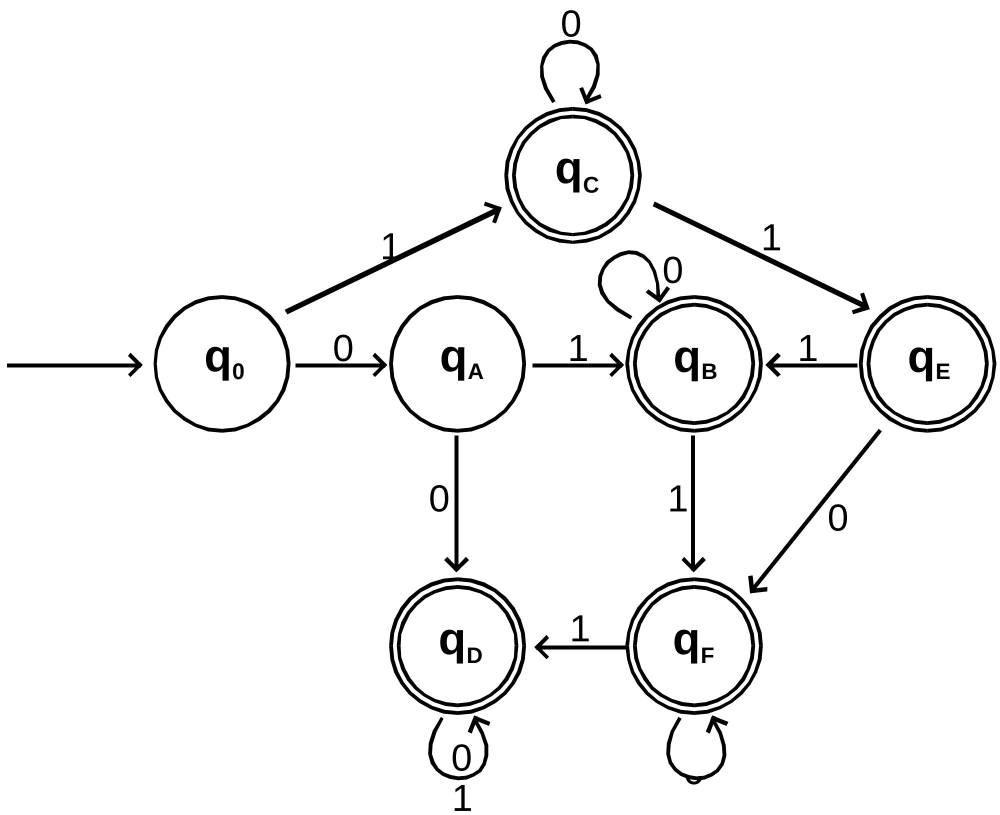
Minimierungsschritte

(A) Entfernung aller nicht zugänglichen Zustände
(B) Finden und Zusammenfassen aller äquivalenten Zustände

Bsp. 4:

Д$_f$ = ($\{q_0, q_A, q_B, q_C, q_D, q_E, q_F\}$, $\{0, 1\}$, δ, q_0, F)

F = $\{q_B, q_C, q_E, q_F\}$



<u>Minimierungsschritte</u>

(A) Entfernung aller nicht zugänglichen Zustände

(B) Finden und Zusammenfassen aller äquivalenten Zustände

→ (A) nicht möglich
→ (B) Äquivalenz testen
(am Beispiel q_0 und q_A):

$\delta(q_0, 0) = q_A$ $\delta(q_0, 1) = q_C$

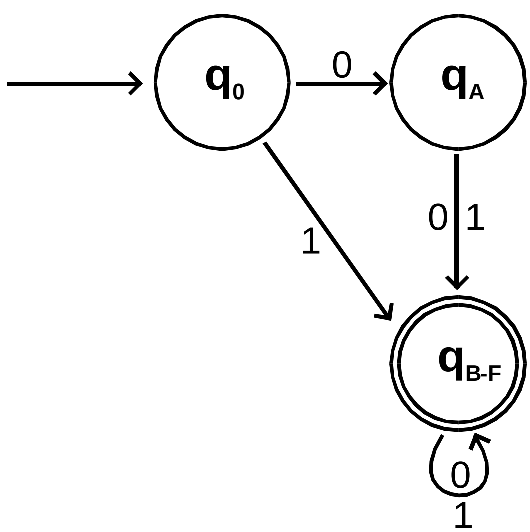
$\delta(q_A, 0) = q_D$ **A-D** $\delta(q_A, 1) = q_B$ **C-B**

Die Äquivalenten A-D und C-B werden ins oberste linke Feld der Tabelle eingetragen. Analog die anderen Felder. Da q_A und q_D nicht äquivalent sind, wird dieses Feld in der rechten Tabelle gestrichen.

	q_0	q_A	q_B	q_C	q_D	q_E
q_A	A-D C-B					
q_B						
q_C			B-C F-E			
q_D			B-D F-D	C-D E-D		
q_E			B-F F-B	C-F E-B	D-F D-B	
q_F			B-F F-D	C-F E-B	D-F D-D	F-F B-D

	q_0	q_A	q_B	q_C	q_D	q_E
q_A	~~A-D C-B~~					
q_B						
q_C			B-C F-E			
q_D			B-D F-D	C-D E-D		
q_E			B-F F-B	C-F E-B	D-F D-B	
q_F			B-F F-D	C-F E-B	D-F D-D	F-F B-D

Wir erhalten am Ende als Menge äquivalenter Zustände: $\{q_0\}$, $\{q_A\}$, $\{q_B, q_C, q_C, q_E, q_F\}$.
So kann man einen minimalen Automaten bilden:



Pushdown-Automaten (nicht deterministisch)

Endliche Automaten können keine kontextfreien irregulären Sprachen erkennen

z.B.: $S \rightarrow uAy$ $\quad A \rightarrow vAx$ $\quad A \rightarrow w$

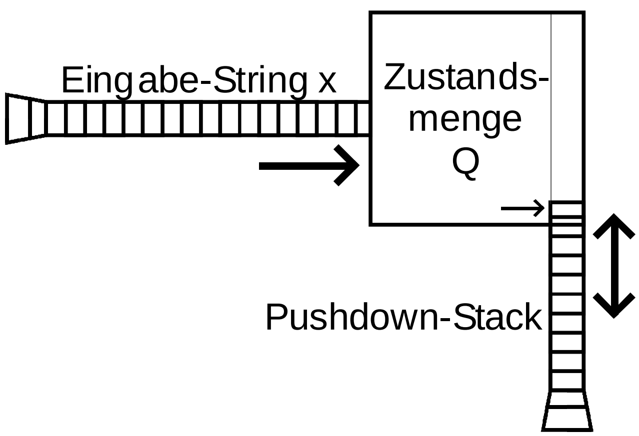
$S \rightarrow uAy \rightarrow uvAxy \rightarrow uvvAxxy \rightarrow uv^nAx^ny \rightarrow uv^nwx^ny$

Wörter wie uv^nwx^ny mit $n \geq 0$ können durch Pushdown-Automaten erkannt werden

→ Pushdown-Liste (LIFO-Stack)

$Д_P = (Q, \sum, \Gamma, \delta, q_0, Z_0, F)$

Γ... Pushdown-List-Alphabet
Z_0…initiales Pushdown-List Symbol



Sprache $L = \{x \mid x \text{ in } \sum^+, x \text{ wird von } Д_f \text{ akzeptiert}\}$

Bedingung für Akzeptieren von x:
- (I) Start bei q_0 mit Symbol Z_0 auf jedem Stack
- (II) Ende bei vollständig gescanntem x und Zustand in F

Bsp. 5:

$L = \{x \mid x = ca^ndb^n, n\geq0\}$

$Д_P = (\{q_0\}, \{a, b, c, d\}, \{S, A, B, C, D\}, \delta, q_0, S, ø)$

$\delta(q_0, c, S) = \{ (q_0, DAB), (q_0, C) \}$
$\delta(q_0, d, C) = \{ (q_0, \lambda) \}$
$\delta(q_0, b, B) = \{ (q_0, \lambda) \}$
$\delta(q_0, a, D) = \{ (q_0, \lambda) \}$
$\delta(q_0, a, A) = \{ (q_0, AB), (q_0, CB) \}$

Bei „caadbb“:	Stack
	S
$\delta(q_0, c, S) = \{ \underline{(q_0, DAB)}, (q_0, C) \}$	DAB
$\delta(q_0, a, D) = \{ (q_0, \lambda) \}$	AB
$\delta(q_0, a, A) = \{ (q_0, AB), \underline{(q_0, CB)} \}$	CBB
$\delta(q_0, d, C) = \{ (q_0, \lambda) \}$	BB
$\delta(q_0, b, B) = \{ (q_0, \lambda) \}$	B
$\delta(q_0, b, B) = \{ (q_0, \lambda) \}$	ø

Bsp 2:

Gegeben ist eine kontextfreie Grammatik.

$G = (\{S, A\}, \{a, b, c, d\}, P, S\}$

Produktionen:

$S \rightarrow cA$ $\qquad$ $A \rightarrow aAb$ $\qquad$ $A \rightarrow d$

Daraus konstruierter Automat:

$\text{Д}_P = (\{q_0\}, \{a, b, c, d\}, \{S, A, a, b, c, d\}, \delta, q_0, S, ø)$

δ ist die Abbildung

$\delta(q_0, \lambda, S) = \{ (q_0, cA) \}$
$\delta(q_0, \lambda, A) = \{ (q_0, aAb), (q_0, d) \}$
$\delta(q_0, a, a) = \{ (q_0, \lambda) \}$
$\delta(q_0, b, b) = \{ (q_0, \lambda) \}$
$\delta(q_0, c, c) = \{ (q_0, \lambda) \}$
$\delta(q_0, d, d) = \{ (q_0, \lambda) \}$

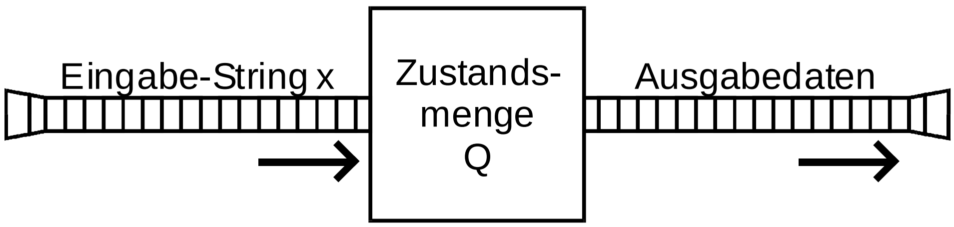
Beispielwort „caadbb“:

	S
$\delta(q_0, \lambda, S) = \{ (q_0, cA) \}$	cA
$\delta(q_0, c, c) = \{ (q_0, \lambda) \}$	A
$\delta(q_0, \lambda, A) = \{ \underline{(q_0, aAb)}, (q_0, d) \}$	aAb
$\delta(q_0, a, a) = \{ (q_0, \lambda) \}$	Ab
$\delta(q_0, \lambda, A) = \{ \underline{(q_0, aAb)}, (q_0, d) \}$	aAbb
$\delta(q_0, a, a) = \{ (q_0, \lambda) \}$	Abb
$\delta(q_0, \lambda, A) = \{ (q_0, aAb), \underline{(q_0, d)} \}$	dbb
$\delta(q_0, d, d) = \{ (q_0, \lambda) \}$	bb
$\delta(q_0, b, b) = \{ (q_0, \lambda) \}$	b
$\delta(q_0, b, b) = \{ (q_0, \lambda) \}$	ø

Automaten für einfache syntaxgerichtete Übersetzung

Endliche Automaten und Pushdown-Automaten werden zum Erkennen von regulären und kontextfreien Sprachen benutzt

→ Der Eingabewert soll nun verarbeitet und Daten zur Weiterverarbeitung ausgegeben werden



Endliche Signalumformer

- Eingabe- **und** Ausgabealphabet

$$\mathbf{\text{Ъ}_f = (Q, \sum, \Delta, \delta, q_0, F)}$$

$\sum$… Eingabealphabet
Δ... Ausgabealphabet

→ Der Automat selektiert Ausgabe-String und Zustand in Abhängigkeit von Eingabewert und vom aktuellen Zustand.

(Eingabe-) Sprache $L(\text{Ъ}_f) = \{x \mid x \text{ in } \sum^+, x \text{ wird von } \text{Ъ}_f \text{ akzeptiert}\}$

x wird akzeptiert ↔ reguläre Übersetzung

→ Übersetzungspaar (x, y)
(!) Aber: Länge von x muss nicht zwangsläufig der Länge von y entsprechen

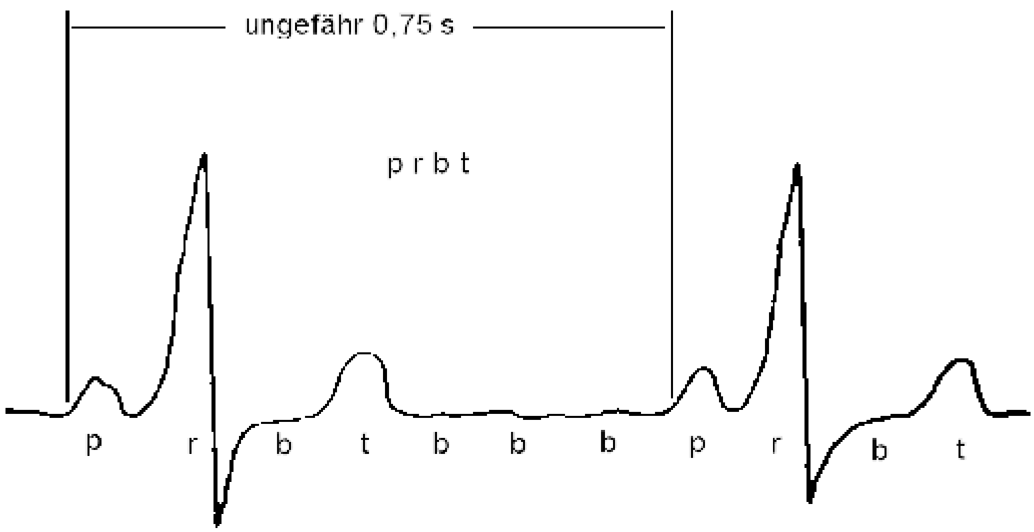
Darstellungsmöglichkeiten von Signalumformern:
- State Transition Diagram
- Output Diagram

→ gerichteter Graph mit Zuständen als Knoten

Praktische Anwendung: EKG des menschlichen Herzen (Elektrokardiogramm)

	Puls	
menschlicher Herzschlag	p	Atria (kleiner Herzmuskel)
	r	Ventricles (großer Herzmuskel)
	t	Repolarisierung des Ventricles
	b	Pause

Ein normaler Herzschlag im EKG-Ausschnitt besteht aus
$p \rightarrow r \rightarrow b \rightarrow t \rightarrow b^n$ mit $n \geq 0$



Ausgedrückt in einer Grammatik G bedeutet das:

G = ({S, A, B, C, D, E, H}, {p, r, b, t}, P, S)

Produktionen

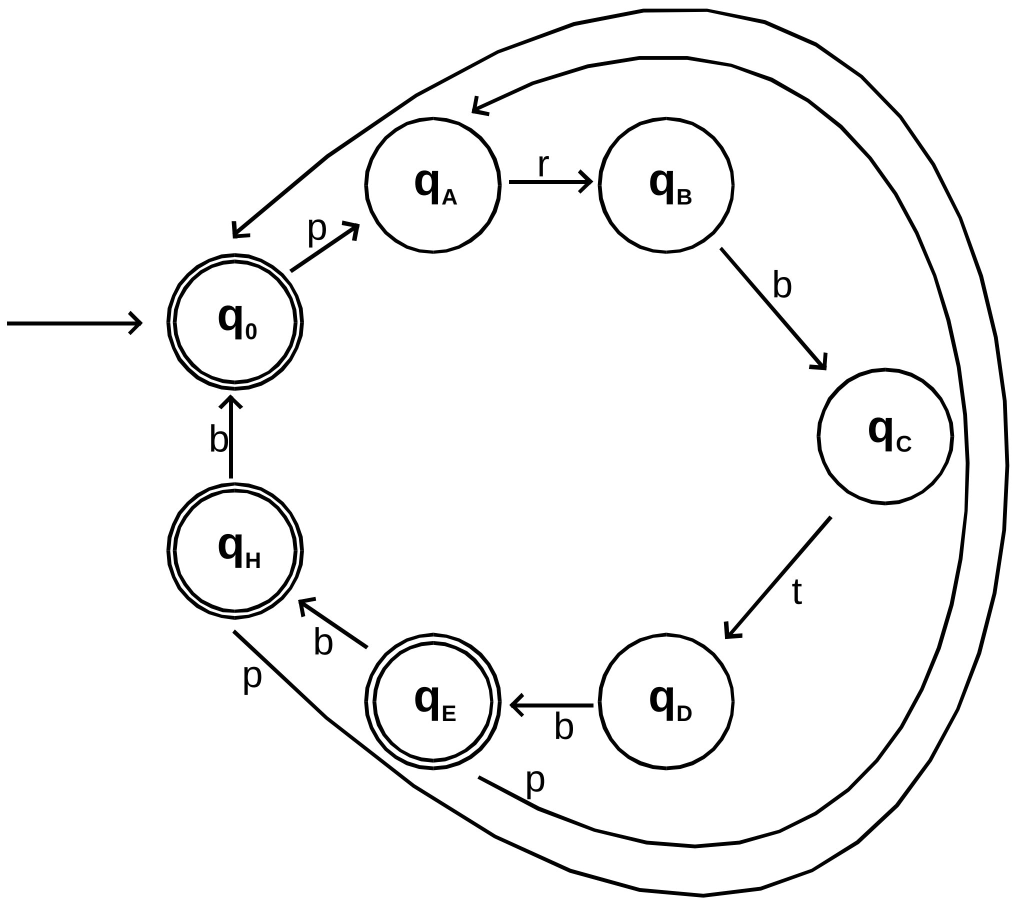
$S \rightarrow pA$
$A \rightarrow rB$
$B \rightarrow bC$
$C \rightarrow tD$
$D \rightarrow b$
$D \rightarrow bE$

$E \rightarrow b$
$E \rightarrow bH$
$E \rightarrow pA$

$H \rightarrow b$
$H \rightarrow bS$
$H \rightarrow pA$

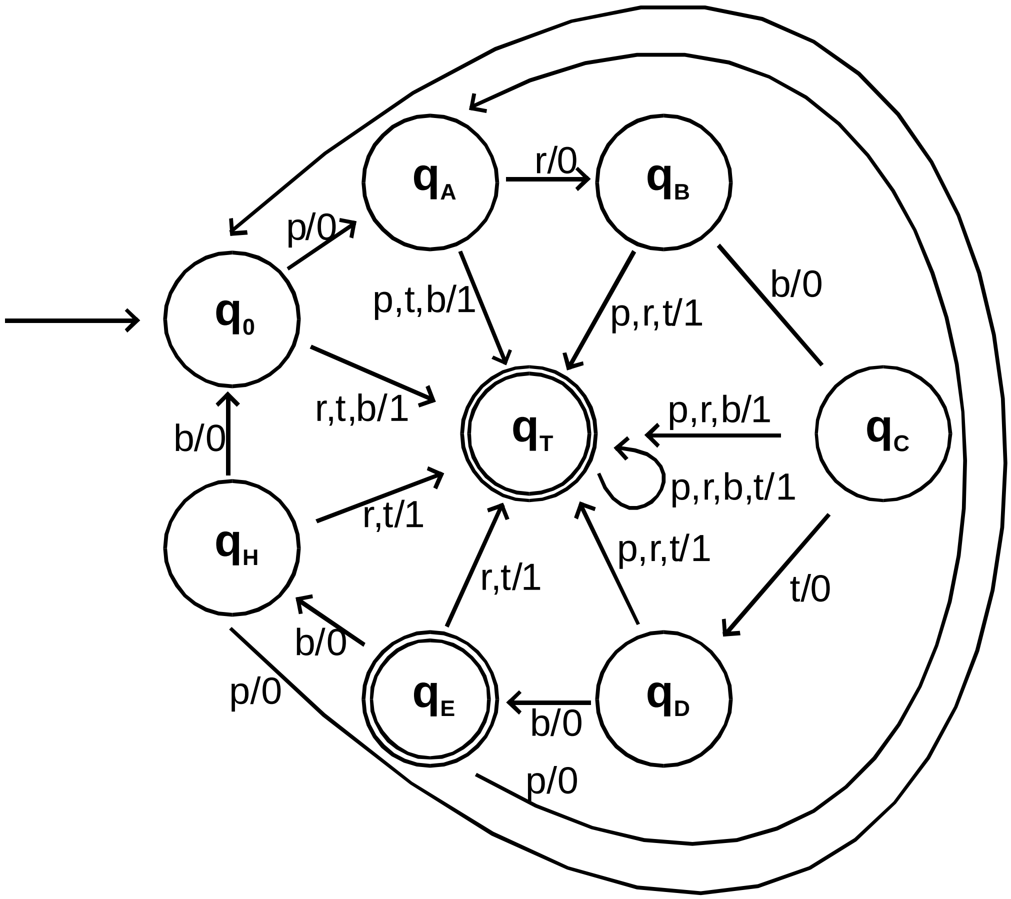
Beispielwort: p r b t bbb prbt bb

Zustandsdiagramm ohne Ausgangsdaten (Vereinfachung: regelmäßiger Herzschlag)



q_0, q_H, q_E … Endzustände

→ um Echtzeitalarm ausgeben zu können, muss ein zusätzlicher „Alarm“-Zustand q_T sowie der Ausgabestring hinzugefügt werden.



q_T… Endzustand

Eine Menge T in $\sum^* \times \Delta^*$ ist die Übersetzung eines endlichen Signalumformers, wenn T durch ein reguläres, syntaxgerichtetes Überstzungsschema erzeugt werden kann.

Um ein reguläres Schema von

von $\sum = \{p, r, b, t\}$ nach $\Gamma = \{0, 1\}$ zu erzeugen
(mit derselben Wirkung wie der zuletzt gezeigt Signalumformer)

sind 8 Nonterminals für die 8 Zustände nötig

$Q = \{q0, q_A, q_B, q_C, q_D, q_E, q_H, q_T\}$
$N = \{S, A, B, C, D, E, H,, T\}$

$\delta(q_0, p) = \{ (q_A, 0) \}$
:
:

$\delta(q_0, r) = \delta(q_0, t) = \delta(q_0, b) = \{ (q_T, 1) \}$

So kommt man zu:

$S \rightarrow pA, 0A$
$S \rightarrow rT, 1T$
$S \rightarrow tT, 1T$
$S \rightarrow bT, 1T$

Da q_T Endzustand ist, ergeben sich noch die Terminierer:

$S \rightarrow r, 1$
$S \rightarrow t, 1$
$S \rightarrow b, 1$

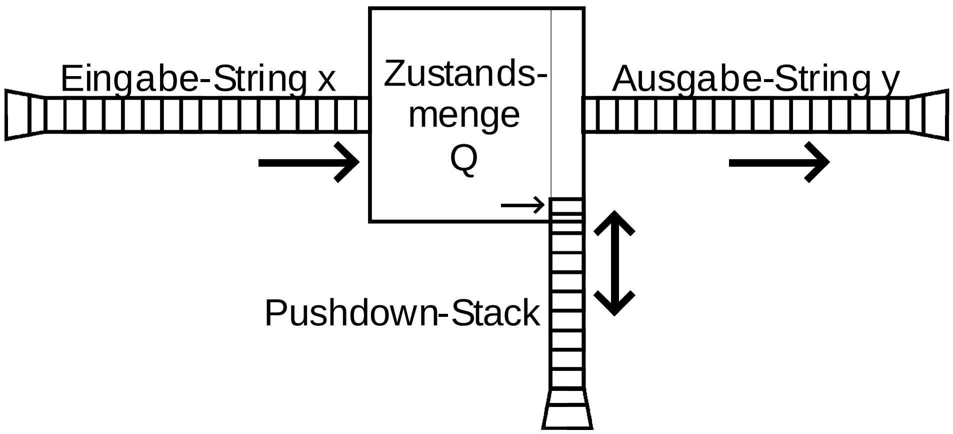
$A \rightarrow rB, 0B$	$C \rightarrow rT, 1T$	$E \rightarrow cT, 1T$
$A \rightarrow tT, 1T$	$C \rightarrow p, 1$	$E \rightarrow t, 1$
$A \rightarrow p, 1$	$C \rightarrow b, 1$	$H \rightarrow bS, 0S$
$A \rightarrow b, 1$	$C \rightarrow pT, 1T$	$H \rightarrow rT, 1T$
$A \rightarrow pT, 1T$	$C \rightarrow r, 1$	$H \rightarrow r, 1$
$A \rightarrow bT, 1T$	$D \rightarrow pT, 1T$	$H \rightarrow pA, 0A$
$A \rightarrow t, 1$	$D \rightarrow tT, 1T$	$H \rightarrow tT, 1T$
$B \rightarrow pT, 1T$	$D \rightarrow r, 1$	$H \rightarrow t, 1$
$B \rightarrow tT, 1T$	$D \rightarrow bE, 0E$	$T \rightarrow pT, 1T$
$B \rightarrow r, 1$	$D \rightarrow rT, 1t$	$T \rightarrow tT, 1T$
$B \rightarrow bC, 0C$	$D \rightarrow p, 1$	$T \rightarrow p, 1$
$B \rightarrow rT, 1T$	$D \rightarrow t, 1$	$T \rightarrow t, 1$
$B \rightarrow p, 1$	$E \rightarrow pA, 0A$	$T \rightarrow rT, 1T$
$B \rightarrow t, 1$	$E \rightarrow rT, 1T$	$T \rightarrow bT, 1T$
$C \rightarrow tT, 1T$	$E \rightarrow r, 1$	$T \rightarrow r, 1$
$C \rightarrow tD, 0D$	$E \rightarrow bH, 0H$	$T \rightarrow b, 1$

Pushdown – Signalumformer

→ um eine irreguläre kontextfreie Übersetzung zu gewährleisten, ist eine endliche, aber kapazitätslose Pushdown-Liste notwendig.

→ Ђ$_P$ = (Q, $\sum$, Γ, Δ, δ, q_0, Z_0, F)

$\sum$… Eingabealphabet
Δ... Ausgabealphabet



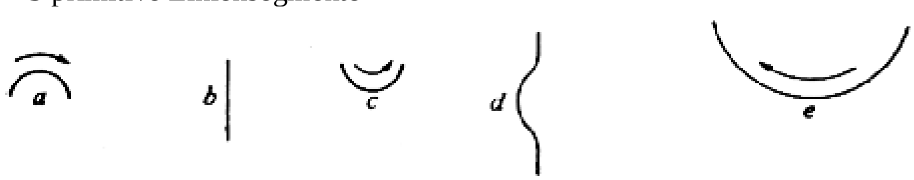
Übersetzungstupel

T (Ђ$_P$) = (x, y)

x von $\sum^*$
y von Δ^* (ist auf dem Ausgabeband)
Ђ$_P$ startet bei q_0 und hält bei leerem Stack an

Praktische Anwendung: Telozentrische Chromosomen (Ledley, 1965)

→ 5 primitive Liniensegmente



→ ergeben 5 terminale Beschreibungen

a	rechte Kurve	<rK>
b	gerade	<gerad>
c	linke Kurve	<lK>
d	Ausbuchtung	<ausb>
e	große rechte Kurve	<grK>

Die Chromosomen werden spezifiziert sich durch 8 nonterminale Beschreibungen:

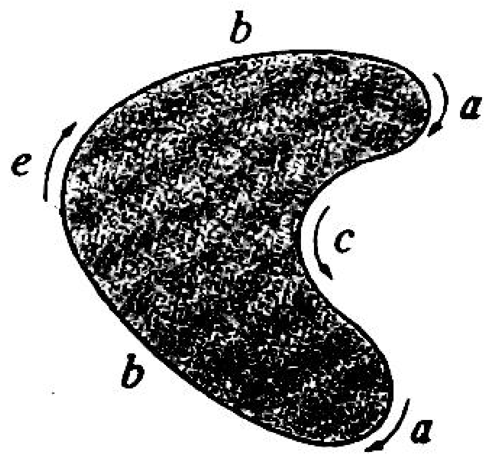
N	Beschreibung	Ausgabe	Produktionen
S…	Chromosomen	<chro>	$S \rightarrow S1$
			$S \rightarrow S2$
S_1…	submedian	<sub>	$S1 \rightarrow AA$
S_2…	telozentrisch	<telo>	$S2 \rightarrow BA$
A…	Armpaar	<armpaar>	$A \rightarrow CA$
			$A \rightarrow AC$
			$A \rightarrow DE$
			$A \rightarrow FD$
B…	Boden	<boden>	$B \rightarrow bB$
			$B \rightarrow Bb$
			$B \rightarrow e$
C…	Seite	<seite>	$C \rightarrow Cb$
			$C \rightarrow bC$
			$C \rightarrow b$
			$C \rightarrow d$
D…	Arm	<arm>	$D \rightarrow bD$
			$D \rightarrow Db$
			$D \rightarrow a$
E…	rechte Seite	<RSeite>	$E \rightarrow cD$
F…	linke Seite	<LSeite>	$F \rightarrow Dc$

→ 5+8 = 13 Symbole im Alphabet
jedes beschreibt eine visuelle Eigenschaft
- entweder Linienzug (Terminale) oder Gruppen derer (Nonterminale)

→ Grammatik $G = (N, \sum, P, S)$

Beispielchromosom: „ebacab“

S → S2 → BA → BbA → ebA → ebDE → ebaE → ebacD → ebacDb → ebacab



(S, S)
→ (S_2, <telo>S_2) → (BA, <telo><boden>B<armpaar>A)
→ (BbA, <telo><boden>B<gerad><armpaar>A)
→ (ebA, <telo><boden><grK><gerad><armpaar>A)
→ (ebDE, <telo><boden><grK><gerad><armpaar><arm>D<RSeite>E)
→ (ebaE, <telo><boden><grK><gerad><armpaar><arm><rK><RSeite>E)
→ (ebacD, <telo><boden><grK><gerad><armpaar><arm><rK><RSeite><lK><arm>D)
→ (ebacDb, <telo><boden><grK><gerad><armpaar><arm><rK><RSeite><lK><arm>D<gerad>
→ (ebacab,
<telo><boden><grK><gerad><armpaar><arm><rK><RSeite><lK><arm><rK><gerad>)

Der Ausgabesatz beschreibt eine komplette Analyse des Eingabesatzes „ebacab“.

Eigenschaften
- Telozentrisches Chromosom
- mit einem Boden <grK><gerad>
- verbunden mit einem Armpaar <arm><rK><RSeite><lK>
- verbunden mit einem Arm <rK><gerad>

Nun kann angesichts der zu Grunde liegenden Beschreibungen der Pushdown-Signalumformer beschrieben werden (Pushdown Transducer):

Ђ$_P$ = ({q_0}, ∑, Γ, Δ, δ, q_0, S, ø)

Eingabe: formaler String
Ausgabe: Beschreibung in unserer Sprache

∑ = {a, b, c, d, e}
Δ = { <chro>, <sub>, <telo>, <armpaar>, <boden>, <seite>, <arm>, <RSeite>, <LSeite> }
Γ = Die Menge aller Nonterminals (= {S, S_1, S_2, A, B, C, D, E, F}), zuzüglich ∑ und Δ

N	Produktionen	Pushdown-Beschreibung
S…	S → S1	$\delta(q_0, \lambda, S)$ = { (q_0, <sub>S_1,λ),
	S → S2	(q_0, <telo>S_2,λ) }
S_1…	S1 → AA	$\delta(q_0, \lambda, S_1)$ = { (q_0, <armpaar>A<armpaar>A,λ) }
S_2…	S2 → BA	$\delta(q_0, \lambda, S_2)$ = { (q_0, <boden>B<armpaar>A,λ) }
A…	A → CA	$\delta(q_0, \lambda, A)$ = { (q_0, <seite>CA,λ),
	A → AC	(q_0, A<seite>C,λ),
	A → DE	(q_0, <arm>D<RSeite>E,λ),
	A → FD	(q_0, <LSeite>F<arm>D,λ) }
B…	B → bB	$\delta(q_0, \lambda, B)$ = { (q_0, b<gerad>B,λ),
	B → Bb	(q_0, Bb<gerad>,λ)
	B → e	(q_0, e<grK>,λ) }
C…	C → bC	$\delta(q_0, \lambda, C)$ = { (q_0, b<gerad>C,λ),
	C → Cb	(q_0, Cb<gerad>,λ),
	C → b	(q_0, b<gerad>,λ),
	C → d	(q_0, d<ausb>,λ) }
D…	D → bD	$\delta(q_0, \lambda, D)$ = { (q_0, b<gerad>D,λ),
	D → Db	(q_0, Db<gerad>,λ),
	D → a	(q_0, a<rK>,λ) }
E…	E → cD	$\delta(q_0, \lambda, E)$ = { (q_0, c<lK><arm>D,λ) }
F…	F → Dc	$\delta(q_0, \lambda, F)$ = { (q_0, Dc<lK>,λ) }

Für jedes Symbol a des Eingabealphabets ∑ (Terminale)

$\delta(q_0, a, a)$ = { (q_0, λ,λ) }

Für jedes Symbol s des Ausgabealphabets Δ

$\delta(q_0, s, s)$ = { (q_0, λ,<s>) }

Ausgabesatzbildung vom Beispielwort: „ebacab“

	S
δ(q_0, λ, S) = { (q_0, <sub>S_1,λ), (q_0, <telo>S_2,λ) }	<telo>S_2
δ(q_0, λ, <telo>) = { (q_0, λ, <telo>) }	S_2
δ(q_0, λ, S_2) = { (q_0, <boden>B<armpaar>A,λ) }	<boden>B<armpaar>A
δ(q_0, λ, <boden>) = { (q_0, λ, <boden>) }	B<armpaar>A
δ(q_0, λ, B) = { (q_0, b<gerad>B,λ), (q_0, Bb<gerad>,λ), (q_0, e<grK>,λ) }	Bb<gerad><armpaar>A
δ(q_0, λ, B) = { (q_0, b<gerad>B,λ), (q_0, Bb<gerad>,λ), (q_0, e<grK>,λ) }	e<grK>b<gerad><armpaar>A
δ(q_0, e, e) = { (q_0, λ, λ) }	<grK>b<gerad><armpaar>A
δ(q_0, λ, <grK>) = { (q_0, λ, <grK>) }	b<gerad><armpaar>A
δ(q_0, b, b) = { (q_0, λ, λ) }	<gerad><armpaar>A
δ(q_0, λ, <gerad>) = { (q_0, λ, <gerad>) }	<armpaar>A
δ(q_0, λ, <armpaar>) = { (q_0, λ, <armpaar>) }	A
δ(q_0, λ, A) = { (q_0, <seite>CA,λ), (q_0, A<seite>C,λ), (q_0, <arm>D<RSeite>E,λ), (q_0, <LSeite>F<arm>D,λ) }	<arm>D<RSeite>E
δ(q_0, λ, <arm>) = { (q_0, λ, <arm>) }	D<RSeite>E
δ(q_0, λ, D) = { (q_0, b<gerad>D,λ), (q_0, Db<gerad>,λ), (q_0, a<rK>,λ) }	a<rK><RSeite>E
δ(q_0, a, a) = { (q_0, λ, λ) }	<rK><RSeite>E
δ(q_0, λ, <rK>) = { (q_0, λ, <rK>) }	<RSeite>E
δ(q_0, λ, <RSeite>) = { (q_0, λ, <RSeite>) }	E
δ(q_0, λ, E) = { (q_0, c<lK><arm>D,λ) }	c<lK><arm>D
δ(q_0, c, c) = { (q_0, λ, λ) }	<lK><arm>D
δ(q_0, λ, <lK>) = { (q_0, λ, <lK>) }	<arm>D
δ(q_0, λ, <arm>) = { (q_0, λ, <arm>) }	D
δ(q_0, λ, D) = { (q_0, b<gerad>D,λ), (q_0, Db<gerad>,λ), (q_0, a<rK>,λ) }	Db<gerad>
δ(q_0, λ, D) = { (q_0, b<gerad>D,λ), (q_0, Db<gerad>,λ), (q_0, a<rK>,λ) }	a<rK>b<gerad>
δ(q_0, a, a) = { (q_0, λ, λ) }	<rK>b<gerad>
δ(q_0, λ, <rK>) = { (q_0, λ, <rK>) }	b<gerad>
δ(q_0, b, b) = { (q_0, λ, λ) }	<gerad>
δ(q_0, λ, <gerad>) = { (q_0, λ, <gerad>) }	ø

Benutzte Literatur

Rafael C. Gonzalez / Michael G. Thomason: Syntactic Pattern Recognition. An Introduction. Addison-Wesley, Reading etc. 1978. S. 96-133.