

Theoretische Informatik I



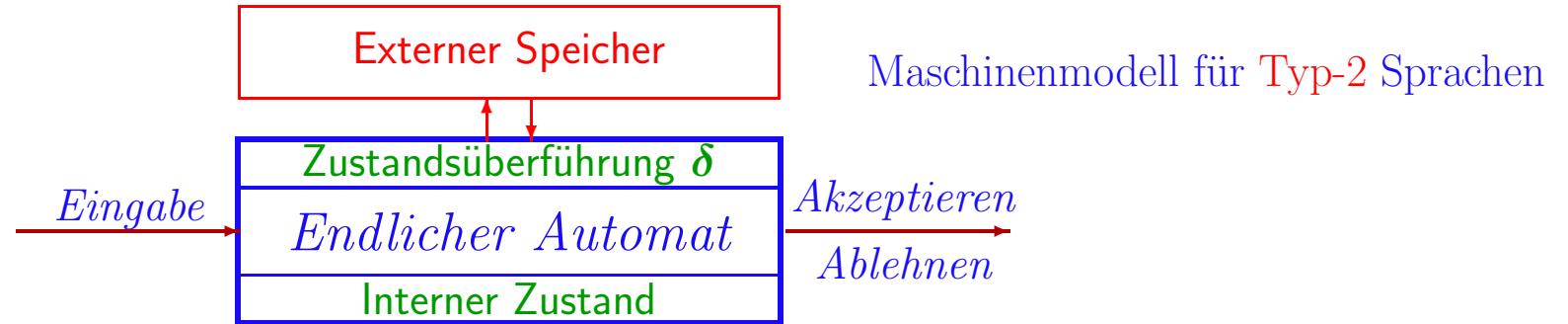
Einheit 3.2

Pushdown Automaten



1. Das Maschinenmodell
2. Arbeitsweise & erkannte Sprache
3. Beziehung zu Typ-2 Sprachen
4. Deterministische PDAs

EIN MASCHINENMODELL FÜR TYP-2 SPRACHEN



• Typ-3 Sprachen werden von NEAs akzeptiert

- Typ-3 Grammatik erzeugt pro Schritt ein Terminalsymbol
 - NEA verarbeitet pro Schritt ein Eingabesymbol
- Erzeugte Terminalsymbole stehen links von der aktuellen Variablen
 - Verarbeitete Eingabesymbole führen zu aktuellem Zustand
- Rechts von der aktuellen Variablen steht noch nichts
 - Im Zustand ist nichts über noch unverarbeitete Eingabesymbole bekannt

• Welches Maschinenmodell paßt zu Typ-2 Sprachen?

- Kontextfreie Grammatiken können $L_1 = \{0^m 1^m \mid m \in \mathbb{N}\}$ erzeugen
- Endliche Automaten haben kein Gedächtnis und können L_1 nicht erkennen

Typ-2 Maschinenmodell benötigt externen Speicher

WELCHES SPEICHERMODELL BRAUCHEN TYP-2 SPRACHEN?

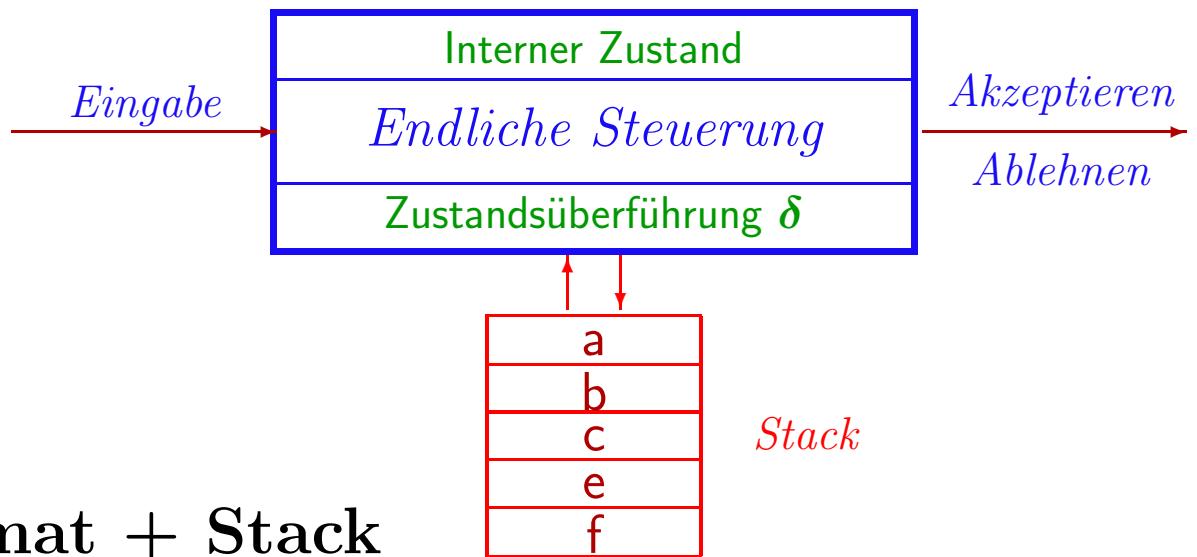
Benutze Analogie der Linksableitungen

- **Links von der aktuellen Variablen A stehen nur erzeugte Terminalsymbole**
 - Entspricht den schon verarbeiteten Eingabesymbolen
- **Aber rechts von A steht bereits Text**
Abarbeitung von A schiebt weiteren Text in die Mitte
 - Automat muß Information speichern, die noch verarbeitet werden muß
 - Information erklärt, was am Ende der Eingabe erwartet wird
- **Wenn A komplett abgearbeitet, springt Linksableitung über Terminalsymbole zur nächsten Variablen**
 - Automat muß zuletzt erzeugte Information zuerst abarbeiten



Speicher des Automaten sollte ein Stack sein

PUSHDOWN-AUTOMATEN INTUITIV



● Endlicher Automat + Stack

- Endliche Steuerung liest Eingabesymbole
- Gleichzeitig kann das oberste Symbol im Stack beobachtet werden

● Eingabe und Stack wird gleichzeitig bearbeitet

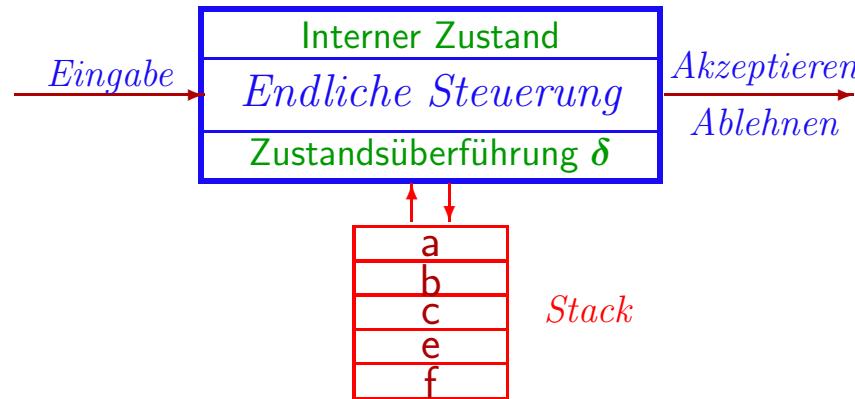
- Gelesenes Symbol wird aus Eingabe “entfernt”
- Zustand kann verändert werden
- Oberstes Stacksymbol wird durch (mehrere) neue Stacksymbole ersetzt
- Nichtdeterministische Entscheidungen und spontane ϵ -Übergänge möglich

PUSHDOWN-AUTOMAT FÜR ‘GERADE’ PALINDROME

$L = \{ww^R \mid w \in \{0, 1\}^*\}$ ist kontextfrei

- **Speichere w in q_0**
 - In q_0 wird je ein Symbol gelesen und auf den Stack gelegt
 - Gelesenes Wort steht von unten nach oben im Stack
- **Spontaner Wechsel “in der Mitte”**
 - Nichtdeterministischer ϵ -Übergang von q_0 nach q_1
 - Im Stack steht w in umgekehrter Reihenfolge
- **Verarbeite w^R in q_1**
 - In q_1 wird je ein Symbol gelesen und mit dem Stacksymbol verglichen
 - Stacksymbol wird bei Gleichheit entfernt
- **Leerer Stack akzeptiert**
 - Wenn Stack leer ist, wurde w^R in q_1 verarbeitet

PUSHDOWN-AUTOMATEN – MATHEMATISCH PRÄZISIERT

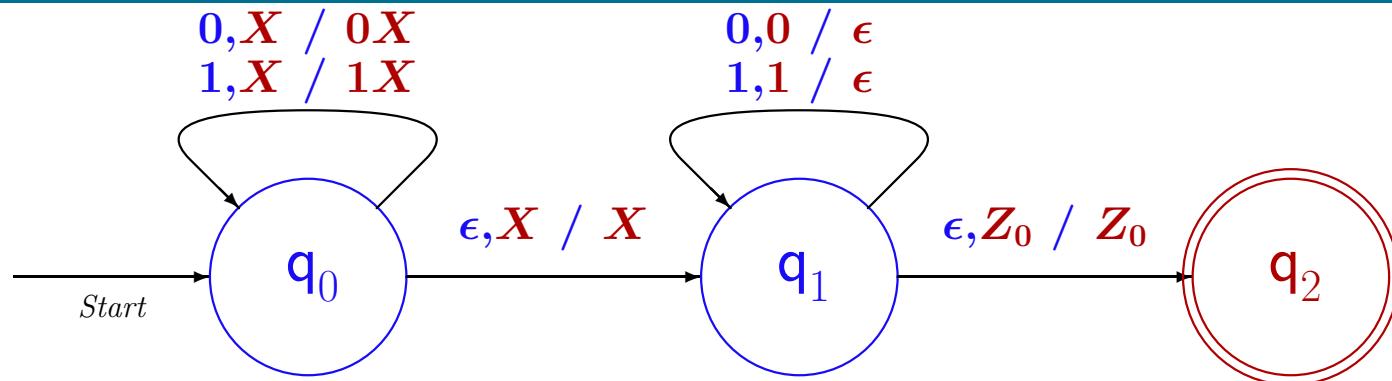


Ein **Pushdown-Automat (PDA, Kellerautomat)** ist ein **7-Tupel** $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ mit

- Q nichtleere endliche **Zustandsmenge**
- Σ endliches **Eingabealphabet**
- Γ endliches **Stackalphabet**
- $\delta: Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma^*)$ **Überführungsfunktion**
- $q_0 \in Q$ **Startzustand** (Anfangszustand)
- $Z_0 \in \Gamma$ **Initialsymbol des Stacks**
- $F \subseteq Q$ Menge von **akzeptierenden Zuständen** (Endzustände)

Pushdown-Automaten sind üblicherweise **nichtdeterministisch!**

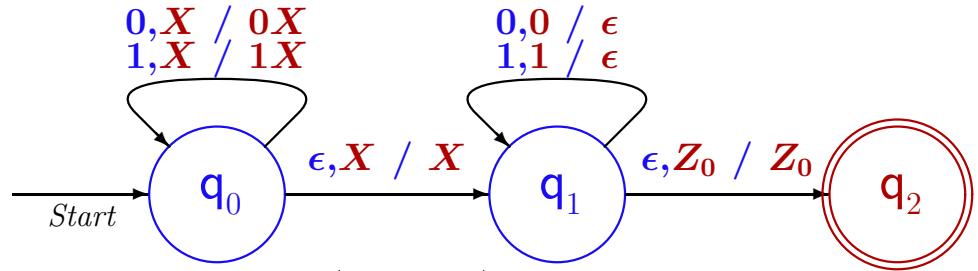
PUSHDOWN-AUTOMAT FÜR $\{ww^R \mid w \in \{0,1\}^*\}$



- Speichere w in q_0
 - Jedes gelesene Symbol wird dem Stack zugefügt
 - $\delta(q_0, a, X) = \{(q_0, aX)\}$ für $a \in \{0,1\}$, $X \in \Gamma$
- Spontaner ϵ -Übergang von q_0 nach q_1
 - $\delta(q_0, \epsilon, X) = \{(q_1, X)\}$ für $X \in \Gamma$
- Verarbeite w^R in q_1
 - Jedes gelesene Symbol wird dem obersten Stacksymbol verglichen
 - $\delta(q_1, a, a) = \{(q_1, \epsilon)\}$ für $a \in \{0,1\}$
- “Leerer” Stack akzeptiert (ϵ -Übergang nach q_2)
 - $\delta(q_1, \epsilon, Z_0) = \{(q_2, Z_0)\}$

$$P = (\{q_0, q_1, q_2\}, \{0,1\}, \{0,1, Z_0\}, \delta, q_0, Z_0, \{q_2\})$$

• Übergangsdiagramme



- Jeder Zustand in Q wird durch einen Knoten (Kreise) dargestellt
- Für $(p, \alpha) \in \delta(q, a, X)$, $a \in (\Sigma \cup \epsilon)$ hat das Diagramm eine Kante $q \xrightarrow{a, X / \alpha} p$ (mehrere Beschriftungen derselben Kante möglich)
- q_0 wird durch einen mit *Start* beschrifteten Pfeil angezeigt
- Endzustände in F werden durch doppelte Kreise gekennzeichnet
- Σ und Γ implizit durch die Diagramm bestimmt, Initialsymbol heißt Z_0

• Übergangstabellen

- Tabellarische Darstellung der Funktion δ
- Kennzeichnung von q_0 durch einen Pfeil
- Kennzeichnung von F durch Sterne
- Σ , Γ und Q implizit durch die Tabelle bestimmt
- Wildcard (*, **,...) für $a \in \Sigma$ oder $X \in \Gamma$ erlaubt

Q	$\Sigma \cup \epsilon$	Γ	Resultat
$\rightarrow q_0$	0	*	$q_0, 0*$
$\rightarrow q_0$	1	*	$q_0, 1*$
$\rightarrow q_0$	ϵ	*	$q_1, *$
q_1	0	0	q_1, ϵ
q_1	1	1	q_1, ϵ
q_1	ϵ	Z_0	q_2, Z_0
*	q_2		

Generalisiere $\hat{\delta}$ zu Konfigurationsübergängen

- **Konfiguration:** der wirkliche ‘Zustand’ des PDA

- Mehr als $q \in Q$: auch Inhalt des Stacks und unverarbeitete Eingabe zählt
- Formal dargestellt als Tripel $\mathbf{K} = (\mathbf{q}, \mathbf{w}, \gamma) \in Q \times \Sigma^* \times \Gamma^*$

- **Konfigurationsübergang** \vdash^*

- Wechsel zwischen Konfigurationen durch Abarbeitung von Worten
- $(\mathbf{q}, aw, X\beta) \vdash (p, w, \alpha\beta)$, falls $(p, \alpha) \in \delta(q, a, X)$
- $\mathbf{K}_1 \vdash^* \mathbf{K}_2$, falls $K_1 = K_2$ oder
 - es gibt eine Konfiguration K mit $K_1 \vdash K$ und $K \vdash^* K_2$

- **Konfigurationsübergänge für NEAs definierbar**

- Konfigurationen sind Paare $\mathbf{K} = (q, w) \in Q \times \Sigma^*$
 - $(q, aw) \vdash (p, w)$, falls $p \in \delta(q, a)$, $K_1 \vdash^* K_2$ definiert wie oben
- Allgemeinere, aber für endliche Automaten weniger intuitive Notation

ABARBEITUNG DES PALINDROM PDA

Verarbeitung von 1111

$(q_0, 1111, Z_0)$

$(q_0, 111, 1Z_0)$

$(q_0, 11, 11Z_0)$

$(q_0, 1, 111Z_0)$

$(q_0, \epsilon, 1111Z_0)$

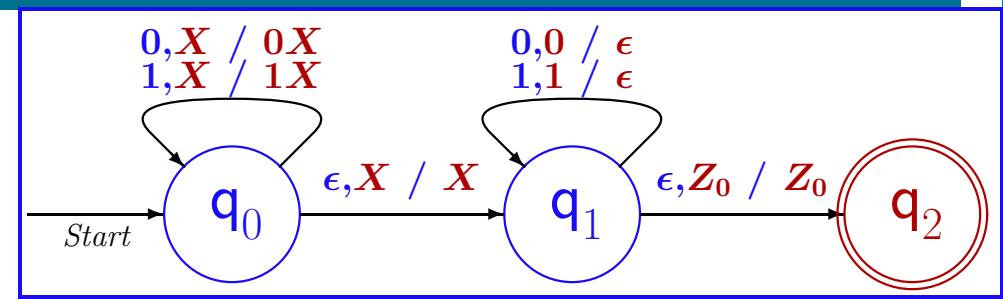
$(q_1, \epsilon, 1111Z_0)$

$(q_1, 111, Z_0) \longrightarrow (q_2, 1111, Z_0)$

$(q_1, 11, 1Z_0) \longrightarrow (q_1, 11, Z_0)$

$(q_1, 1, 11Z_0) \longrightarrow (q_2, 11, Z_0)$

$(q_1, \epsilon, 111Z_0) \longrightarrow (q_1, \epsilon, Z_0)$



(q_2, ϵ, Z_0)

WICHTIGE EINSICHTEN ZU KONFIGURATIONSÜBERGÄNGEN

- Gilt $(q,x,\alpha) \xrightarrow{*} (p,y,\beta)$ dann gilt auch
 $(q,xw,\alpha\gamma) \xrightarrow{*} (p,yw,\beta\gamma)$ für alle $w \in \Sigma^*$, $\gamma \in \Gamma^*$

Weder w noch γ werden bei der Verarbeitung angesehen

- Beweis durch Induktion über Anzahl der Konfigurationsschritte
- Kernargument: $(q,aw,X\gamma) \xrightarrow{*} (p,w,\beta\gamma)$, falls $(p,\beta) \in \delta(q,a,X)$
was hinter a bzw. X kommt, bleibt unverändert

- Gilt $(q,xw,\alpha) \xrightarrow{*} (p,yw,\beta)$ dann gilt auch
 $(q,x,\alpha) \xrightarrow{*} (p,y,\beta)$ für alle $w \in \Sigma^*$

Wenn w bisher nicht gelesen wurde, dann spielt es (noch) keine Rolle
Dagegen kann es von Bedeutung sein, ob im Stack hiner α etwas steht

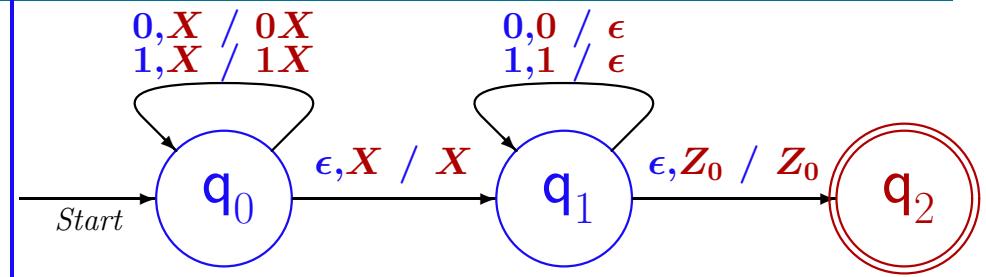
● Zwei alternative Definitionen

- Akzeptanz durch akzeptierende Endzustände (Standarddefinition)
 - $L_F(P) = \{ w \in \Sigma^* \mid \exists q \in F. \exists \beta \in \Gamma^*. (q_0, w, Z_0) \vdash^* (q, \epsilon, \beta) \}$
- Akzeptanz durch leeren Stack (oft praktischer)
 - $L_\epsilon(P) = \{ w \in \Sigma^* \mid \exists q \in Q. (q_0, w, Z_0) \vdash^* (q, \epsilon, \epsilon) \}$

● Beide Akzeptanzdefinitionen sind äquivalent

- Zu jedem PDA $P_\epsilon = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, \emptyset)$ kann ein PDA P_F konstruiert werden mit $L_\epsilon(P_\epsilon) = L_F(P_F)$
- Zu jedem PDA $P_F = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ kann ein PDA P_ϵ konstruiert werden mit $L_F(P_F) = L_\epsilon(P_\epsilon)$

SPRACHEN DES PALINDROMAUTOMATEN



- $L_F(P) = \{ww^R \mid w \in \{0,1\}^*\}$

⊳: Durch Induktion über Länge von w zeige, daß für jedes Wort ww^R gilt

$$(q_0, ww^R, Z_0) \vdash^* (q_0, w^R, w^R Z_0) \vdash (q_1, w^R, w^R Z_0) \vdash^* (q_1, \epsilon, Z_0) \vdash (q_2, \epsilon, Z_0)$$

⊲: Durch Induktion über Länge von x zeige

Wenn $(q_0, x, \alpha) \vdash^* (q_1, \epsilon, \alpha)$ dann $x = ww^R$ für ein $w \in \{0,1\}^*$

Kernidee: $(q_0, x_1..x_n, \alpha) \vdash^* (q_0, x_2..x_n, x_1\alpha) \vdash^* (q_1, x_i..x_n, \beta x_1\alpha)$
 $\vdash^* (q_1, x_n, x_1\alpha) \vdash^* (q_1, \epsilon, \alpha)$

impliziert $(q_0, x_1..x_{n-1}, \alpha) \vdash^* (q_0, x_2..x_{n-1}, x_1\alpha) \vdash^* \dots \vdash^* (q_1, \epsilon, x_1\alpha)$

und $x_1..x_n = x_1x_2..x_{n-1}x_1 = x_1vv^Rx_1$ für ein $v \in \{0,1\}^*$ Siehe HMU §6.2.1

- $L_\epsilon(P) = \emptyset$

– Einfaches Argument: Z_0 wird nie gelöscht

– Modifikation: Ändere Kantenbeschriftung von q_1 nach Q_2 zu $\epsilon, Z_0 / \epsilon$

Dann gilt $L_\epsilon(P') = L_F(P) = \{ww^R \mid w \in \{0,1\}^*\}$

TRANSFORMATION VON L_ϵ IN L_F

Zu jedem PDA $P_\epsilon = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, \emptyset)$ kann ein PDA P_F konstruiert werden mit $L_\epsilon(P_\epsilon) = L_F(P_F)$

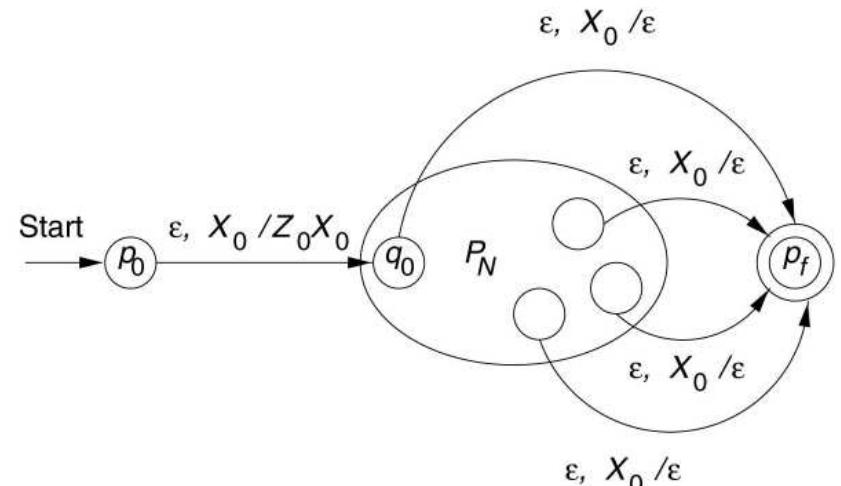
- Bei leerem Stack wechsele in Endzustand

- Neues Initialsymbol X_0 für P_F markiert unteres Ende des Stacks
- Neuer Anfangszustand p_0 für P_F schreibt Initialsymbol von P_ϵ auf Stack
- Neuer Endzustand p_f in den bei “leerem” Stack gewechselt wird

- $P_F = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, p_0, X_0, \delta_F, \{p_f\})$

- $\delta_F(p_0, \epsilon, X_0) = \{(q_0, Z_0 X_0)\}$
- $\delta_F(q, a, X) = \delta(q, a, X)$ für alle $q \in Q, X \in \Gamma$
- $\delta_F(q, \epsilon, X_0) = \{(p_f, \epsilon)\}$ für alle $q \in Q$

Korrektheitsbeweis durch Detailanalyse



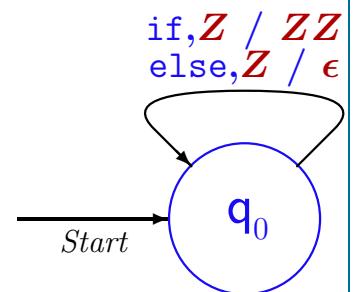
UMWANDLUNG EINES L_ϵ -PDA IN EINEN L_F -PDA

- Gegeben $P_\epsilon = (\{q\}, \{\text{if, else}\}, \{Z\}, q, Z, \delta, \emptyset)$

mit $\delta(q, \text{if}, Z) = \{(q, ZZ)\}$

$\delta(q, \text{else}, Z) = \{(q, \epsilon)\}$

- Erkennt, daß ein (Teil-)Ausdruck mehr **else** als **if** enthält



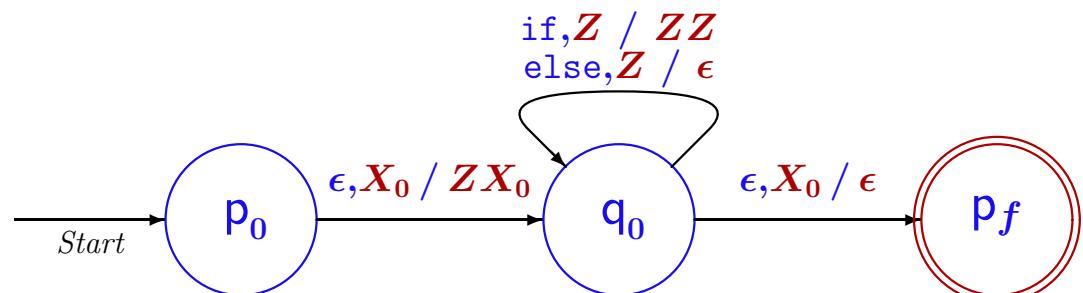
- $P_F = (\{p_0, q, p_f\}, \{\text{if, else}\}, \{X_0, Z\}, p_0, X_0, \delta_F, \{p_f\})$

– $\delta_F(p_0, \epsilon, X_0) = \{(q, ZX_0)\}$

– $\delta_F(q, \text{if}, Z) = \{(q, ZZ)\}$

– $\delta_F(q, \text{else}, Z) = \{(q, \epsilon)\}$

– $\delta_F(q, \epsilon, X_0) = \{(p_f, \epsilon)\}$



TRANSFORMATION VON L_F IN L_ϵ

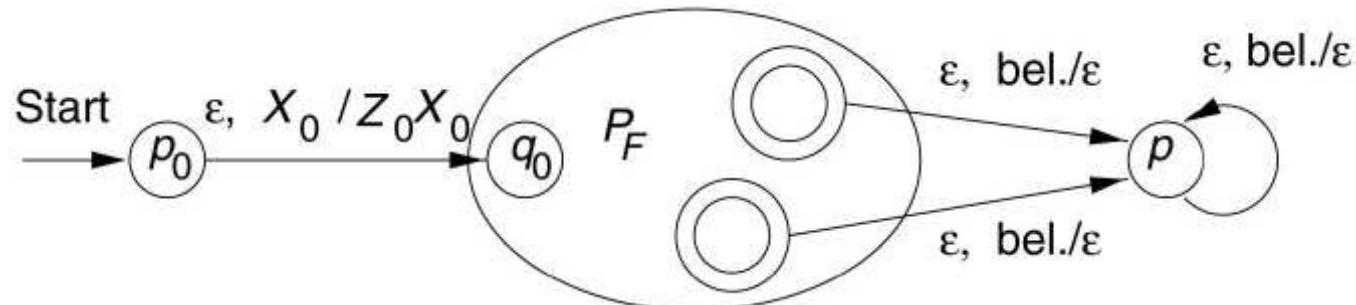
Zu jedem PDA $P_F = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ kann ein PDA P_ϵ konstruiert werden mit $L_F(P_F) = L_\epsilon(P_\epsilon)$

- Im Endzustand leere den Stack

- Neuer Stacklösch-Zustand p , in von Endzuständen gewechselt wird
- Neues Initialsymbol X_0 für P_ϵ verhindert irrtümliches Leeren des Stacks
- Neuer Anfangszustand p_0 für P_ϵ schreibt Initialsymbol von P_F auf Stack

- $P_F = (Q \cup \{p_0, p\}, \Sigma, \Gamma \cup \{X_0\}, q_0, X_0, \delta_\epsilon, \emptyset)$

- $\delta_\epsilon(p_0, \epsilon, X_0) = \{(q_0, Z_0 X_0)\}$
- $\delta_\epsilon(q, a, X) = \delta(q, a, X)$ für alle $q \in Q, X \in \Gamma$
- $\delta_\epsilon(q, \epsilon, X_0) = \{(p, \epsilon)\}$ für alle $q \in F$
- $\delta_\epsilon(p, \epsilon, X) = \{(p, \epsilon)\}$ für alle $X \in \Gamma \cup \{X_0\}$



SIND PDAs WIRKLICH DIE MASCHINEN FÜR TYP-2 SPRACHEN?

$$\mathcal{L}_2 = \mathcal{L}_{PDA} = \{ L \mid \exists P: \text{PDAs. } L = L_\epsilon(P) \}$$

• Konfigurationsübergänge $\hat{=}$ Linksableitungen

- $(q_0, xy, Z_0) \vdash^* (q, y, A\alpha)$ bedeutet, daß P nach Verarbeitung von x im Zustand q ist und noch y und den Stack $A\alpha$ zu verarbeiten hat
- $A\alpha$ muß gespeichert und beim Lesen von y komplett abgearbeitet werden
- Linksableitung $S \xrightarrow{*} xA\alpha \xrightarrow{*} xy$ erzeugt aus dem Startsymbol zuerst das Wort $xA\alpha$ und muß dann y aus $A\alpha$ ableiten

• Grammatik \longrightarrow Pushdown-Automat

- PDA muß Linksableitung auf Stack simulieren
- Erzeugte linke Terminalteilworte müssen mit Teil der Eingabe verglichen werden um nächste Variable freizulegen

• Pushdown-Automat \longrightarrow Grammatik

- Grammatik muß Abarbeitung von Symbolen des Stacks simulieren
- Regeln beschreiben wie PDA zur Abarbeitung von X mit δ Zwischenworte im Stack auf- und schließlich wieder abbaut

VON GRAMMATIKEN ZU PUSHDOWN-AUTOMATEN

Zu jeder kontextfreien Grammatik $G = (V, T, P_G, S)$
kann ein PDA P konstruiert werden mit $L(G) = L_\epsilon(P)$

● Stack simuliert Linksableitungen von G

- Beginne mit Startsymbol von G
- $A \in V$ wird durch die rechte Seite β einer Regel $A \rightarrow \beta$ ersetzt
- $a \in T$ wird vom Stack entfernt, wenn es als Eingabesymbol erscheint um nächste Variable der Linksableitungen im Stack zu identifizieren

● $P = (\{q\}, T, V \cup T, q, S, \delta, \emptyset)$

- $\delta(q, \epsilon, A) = \{(q, \beta) \mid A \rightarrow \beta \in P_G\}$ für alle $A \in V$
- $\delta(q, a, a) = \{(q, \epsilon)\}$ für alle $a \in T$

● Korrektheitsbeweis $L(G) = L_\epsilon(P)$

- Zeige:
 - (\subseteq) Wenn $S = x_1 A_1 \alpha_1 \xrightarrow{L} \dots x_m A_m \alpha_m \xrightarrow{L} w \in T^*$ dann gibt es y_i mit $(q, w, S) \vdash^* (q, y_i, A_i \alpha_i)$ und $x_i y_i = w$
 - (\supseteq) Wenn $(q, w, X) \vdash^* (q, \epsilon, \epsilon)$ dann $X \xrightarrow{*} w$

Wenn $S = x_1 A_1 \alpha_1 \xrightarrow{L} \dots x_m A_m \alpha_m \xrightarrow{L} w \in T^*$ ($x_i \in T^*$, $A_i \in V$)
dann gibt es y_i mit $(q, w, S) \vdash^* (q, y_i, A_i \alpha_i)$ und $x_i y_i = w$

- Beweis durch Induktion über Länge i der Linksableitung
- Basisfall $i = 1$: $S = x_1 A_1 \alpha_1 \xrightarrow{L} w \in T^*$
 - Es folgt $S = A_1$ und $x_1 = \alpha_1 = \epsilon$, also $y_1 = w$
 - $(q, w, S) \vdash^* (q, w, S)$ gilt mit 0 Konfigurationsübergängen
- Induktionsschritt: $S \dots \xrightarrow{L} x_i A_i \alpha_i \xrightarrow{L} x_{i+1} A_{i+1} \alpha_{i+1} \xrightarrow{L} w \in T^*$
 - $x_i A_i \alpha_i \xrightarrow{L} x_{i+1} A_{i+1} \alpha_{i+1}$ verlangt $A_i \xrightarrow{\beta} \in P_G$ für ein β
 - Also $(q, \beta) \in \delta(q, \epsilon, A_i)$ also $(q, y_i, A_i \alpha_i) \vdash^* (q, y_i, \beta \alpha_i)$
 - Zerlege $\beta \alpha_i$ in $x A_{i+1} \alpha_{i+1}$. Dann kann y_i in $x y_{i+1}$ zerlegt werden
 - Es folgt $(q, x y_{i+1}, x A_{i+1} \alpha_{i+1}) \vdash^* (q, y_{i+1}, A_{i+1} \alpha_{i+1})$ (PDA arbeitet x ab)
 - Mit Induktionsannahme: $(q, w, S) \vdash^* (q, y_i, A_i \alpha_i) \vdash^* (q, y_{i+1}, A_{i+1} \alpha_{i+1})$
- Schlußfolgerung: $S = x_1 A_1 \alpha_1 \xrightarrow{L} \dots x_{m+1} A_{m+1} \alpha_{m+1} = w \in T^*$
 - $x_{m+1} = w$ und $A_{m+1} = \alpha_{m+1} = y_{m+1} = \epsilon$
 - Also $(q, w, S) \vdash^* (q, \epsilon, \epsilon)$, d.h. $w \in L_\epsilon(P)$

KORREKTHEITSBEWEIS $L(G) \supseteq L_\epsilon(P)$

Wenn $(q, w, X) \vdash^* (q, \epsilon, \epsilon)$ dann $X \xrightarrow{*} w$

- Beweis durch Induktion über Länge der PDA Berechnung
- Basisfall: $(q, w, X) \vdash (q, \epsilon, \epsilon)$
 - Es folgt $X \xrightarrow{\epsilon} \in P_G$ und $w = \epsilon$, also $X \xrightarrow{*} w$
- Induktionsschritt: $(q, w, X) \vdash^{n+1} (q, \epsilon, \epsilon)$
 - Da X oben im Stack steht, muß der erste Schritt die Form $(q, w, X) \vdash (q, w, Y_1..Y_k)$ für ein $X \xrightarrow{} Y_1..Y_k \in P_G$ haben
 - Dann gibt eine Zerlegung $w = w_1..w_k$ mit
$$(q, w_1..w_k, Y_1..Y_k) \vdash^* (q, w_2..w_k, Y_2..Y_k) \vdash^* (q, \epsilon, \epsilon)$$
 - Es folgt $(q, w_i..w_k, Y_i) \vdash^* (q, w_{i+1}..w_k, \epsilon)$ also $(q, w_i, Y_i) \vdash^* (q, \epsilon, \epsilon)$
 - Per Induktionsannahme folgt $Y_i \xrightarrow{*} w_i$ für alle i
also $X \xrightarrow{} Y_1..Y_k \xrightarrow{*} w_1..w_k = w$
- $L(G) \supseteq L_\epsilon(P)$ folgt nun mit $w \in L_\epsilon(P)$ und $X = S$

UMWANDLUNG EINER GRAMMATIK IN EINEN PDA

- $G_6 = (\{E, I\}, \{a, b, 0, 1, +, *, (,)\}, P_G, E)$
mit $P_G = \{ E \rightarrow I \mid E+E \mid E*E \mid (E) \mid I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \}$
- Erzeuge $P = (\{q\}, T, V \cup T, q, E, \delta, \emptyset)$
mit $V = \{E, I\}$ und $T = \{a, b, 0, 1, +, *, (,)\}$
 - $\delta(q, \epsilon, E) = \{(q, I), (q, E+E), (q, E*E), (q, (E))\}$
 - $\delta(q, \epsilon, I) = \{(q, a), (q, b), (q, Ia), (q, Ib), (q, I0), (q, I1)\}$
 - $\delta(q, a, a) = \{(q, \epsilon)\}$ – $\delta(q, +, +) = \{(q, \epsilon)\}$
 - $\delta(q, b, b) = \{(q, \epsilon)\}$ – $\delta(q, *, *) = \{(q, \epsilon)\}$
 - $\delta(q, 0, 0) = \{(q, \epsilon)\}$ – $\delta(q, (, ()) = \{(q, \epsilon)\}$
 - $\delta(q, 1, 1) = \{(q, \epsilon)\}$ – $\delta(q, (,)) = \{(q, \epsilon)\}$

VON PUSHDOWN-AUTOMATEN ZU GRAMMATIKEN

Zu jedem PDA $P = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ kann eine kontextfreie Grammatik G konstruiert werden mit $L_\epsilon(P) = L(G)$

- **Simuliere Abarbeitung eines Symbols vom Stack**

- Verarbeite Variablen der Form (p, X, q) :
“Entfernen von X kann von Zustand p zu Zustand q führen”
- Entfernen von X kann heißen, zuerst ein $Y_1..Y_m$ auf- und dann abzubauen
- Beginne mit Erzeugung von Z_0 und zeige, daß Z_0 entfernt werden kann

- **$G = (\Sigma, \{S\} \cup Q \times \Gamma \times Q, P_G, S)$**

- $S \rightarrow (q_0, Z_0, q) \in P_G$ für alle $q \in Q$
- $(p, X, q_m) \rightarrow a (p, Y_1, q_1) \dots (q_{m-1}, Y_m, q_m) \in P_G$, für beliebige $q_1, \dots, q_m \in Q$, falls $(p, Y_1..Y_m) \in \delta(q, a, X)$

- **Korrektheitsbeweis $L_\epsilon(P) = L(G)$** (viele Details)

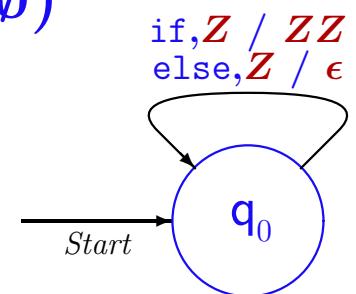
- Zeige: $(p, X, q) \xrightarrow{*} w \in \Sigma^*$ genau dann, wenn $(p, w, X) \vdash^* (q, \epsilon, \epsilon)$
 \subseteq : Induktion über Länge der PDA Berechnung
 \supseteq : Induktion über Länge der Ableitung

UMWANDLUNG EINES PDA IN EINE GRAMMATIK

- Gegeben $P = (\{q\}, \{\text{if, else}\}, \{Z\}, q, Z, \delta, \emptyset)$

mit $\delta(q, \text{if}, Z) = \{(q, ZZ)\}$

$\delta(q, \text{else}, Z) = \{(q, \epsilon)\}$



- $G = (\{\text{if, else}\}, \{S, (q, Z, q)\}, P_G, S)$

mit $P_G = S \rightarrow (q, Z, q)$

$(q, Z, q) \rightarrow \text{if } (q, Z, q)(q, Z, q)$

$(q, Z, q) \rightarrow \text{else}$

Kurzschreibweise A für Hilfssymbol (q, Z, q) ergibt elegantere Darstellung

$G = (\{\text{if, else}\}, \{S, A\}, P_G, S)$ mit $P_G = S \rightarrow A$

$A \rightarrow \text{if } AA$

$A \rightarrow \text{else}$

BRAUCHEN WIR NICHTDETERMINISTISCHE AUTOMATEN?

- **Grammatiken sind nichtdeterministisch**

- Nichtdeterministische Automaten sind das “natürliche” Gegenstück
 - Grammatikregeln führen zu mengenwertiger Überführungsfunktion
- “Wirkliche” Automaten müssen deterministisch sein

- **Typ-3 Sprachen haben deterministische Modelle**

- NEAs können in äquivalente DEAs umgewandelt werden
- Teilmengenkonstruktion kann Automaten exponentiell vergrößern

- **Reichen deterministische PDAs für Typ-2 Sprachen?**

- Überführungsfunktion $\delta: Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$ muß eindeutig sein
- Gibt es für PDAs immer äquivalente deterministische PDAs?

Ein **Deterministischer Pushdown-Automat (DPDA)**
ist ein **7-Tupel** $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ mit

- Q nichtleere endliche **Zustandsmenge**
- Σ endliches **Eingabealphabet**
- Γ endliches **Stackalphabet**
- $\delta: Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$ **Überführungsfunktion**
 - $\delta(q, \epsilon, X)$ nur definiert, wenn $\delta(q, a, X)$ für alle $a \in \Sigma$ undefiniert
- $q_0 \in Q$ **Startzustand** (Anfangszustand)
- $Z_0 \in \Gamma$ **Initialsymbol des Stacks**
- $F \subseteq Q$ Menge von **akzeptierenden Zuständen** (Endzustände)

Erkannte Sprache

- $L_F(P) = \{ w \in \Sigma^* \mid \exists q \in F. \exists \beta \in \Gamma^*. (q_0, w, Z_0) \xrightarrow{*} (q, \epsilon, \beta) \}$
- $L_\epsilon(P) = \{ w \in \Sigma^* \mid \exists q \in Q. (q_0, w, Z_0) \xrightarrow{*} (q, \epsilon, \epsilon) \}$

DPDAS SIND NICHT MÄCHTIG GENUG

- **DPDA-Sprachen sind eine echte Teilklasse von \mathcal{L}_2**

1. $L(DPDA) \subseteq \mathcal{L}_2$: Jeder DPDA ist ein spezieller PDA

2. DPDAs können $\{ww^R \mid w \in \{0, 1\}^*\}$ nicht erkennen

DPDA P kann nicht entscheiden, wo die Mitte eines Wortes liegt

- Wenn $0^n 1 1 0^n$ (großes n) gelesen ist, ist Stack durchs Zählen geleert
- Wenn noch einmal $0^n 1 1 0^n$ gelesen wird, muß P akzeptieren
- Wenn stattdessen $0^m 1 1 0^m$ ($m \neq n$) kommt, darf P nicht akzeptieren
- Aber die Information über n ist nicht mehr gespeichert (Details aufwendig)

- **DPDAs erkennen nur eindeutige Typ-2 Sprachen**

1. Für jeden DPDA P hat $L_\epsilon(P)$ eine eindeutige Grammatik

Für DPDAs ergibt die Umwandlung eine eindeutige Typ-2 Grammatik

- Folge der Konfigurationsübergänge bestimmt Linksableitung eindeutig)

2. Für jeden DPDA P hat $L_F(P)$ eine eindeutige Grammatik

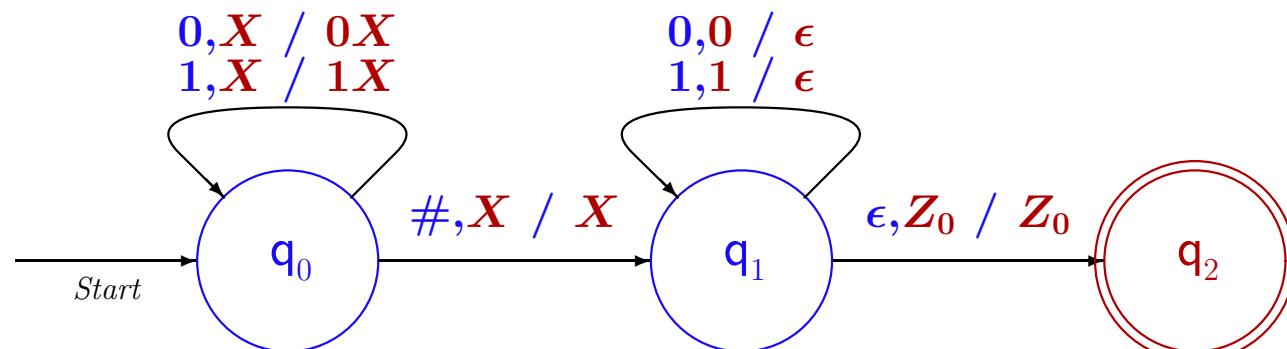
Umwandlung in L_ϵ – DPDA kann deterministisch gemacht werden

DPDAs SIND MÄCHTIGER ALS ENDLICHE AUTOMATEN

- $\mathcal{L}_3 = L(DEA) \subseteq L_F(DPDA)$
 - Jeder DEA ist ein spezieller DPDA
 - Aussage gilt nur für Erkennung mit Endzustand
- $L = \{w\#w^R \mid w \in \{0, 1\}^*\} \in L_F(DPDA) - L(DEA)$
 - L ist nicht regulär

· Beweis durch Pumping Lemma, analog zu $\{ww^R \mid w \in \{0, 1\}^*\}$

– $L = L_F(P)$ für folgenden DPDA P



– P ist deterministisch, da ϵ -Übergang in q_1 genau bei Stacksymbol Z_0

- $\{0\}^* \notin L_\epsilon(DPDA)$
 - Wenn der Stack einmal leer ist, kann ein DPDA nicht mehr weiterarbeiten

- **Maschinenmodell für kontextfreie Sprachen**

- Nichtdeterministischer endlicher Automat mit Stack und ϵ -Übergängen
- Erkennung von Worten durch Endzustand oder leeren Stack
- Erkennungsmodelle sind ineinander transformierbar

- **Verhaltensanalyse durch Konfigurationsübergänge**

- Konfigurationen beschreiben ‘Gesamtzustand’ von Pushdown-Automaten
- Konfigurationsübergänge verallgemeinern Überführungsfunktionen

- **Äquivalent zu kontextfreien Grammatiken**

- Umwandlung von Konfigurationsübergängen in Regeln und umgekehrt

- **Deterministische PDAs sind weniger mächtig**

- DPDAs erkennen nur eindeutige Typ-2 Sprachen
- L_ϵ -DPDAs können nicht einmal alle regulären Sprachen erkennen