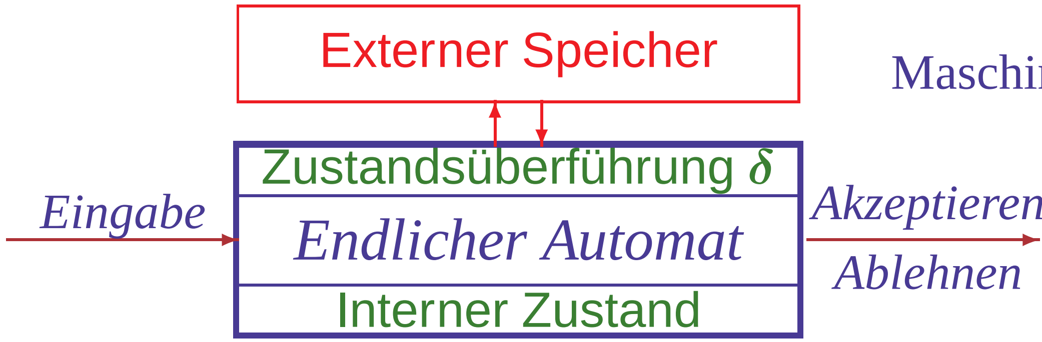
Theoretische Informatik I

Einheit 3.2

Pushdown Automaten

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



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 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
    · Ohne Zwischenspeicher können endliche Automaten L_1 nicht erkennen

Typ-2 Maschinenmodell benötigt externen Speicher

WELCHES SPEICHERMODELL BRAUCHEN TYP-2 SPRACHEN?

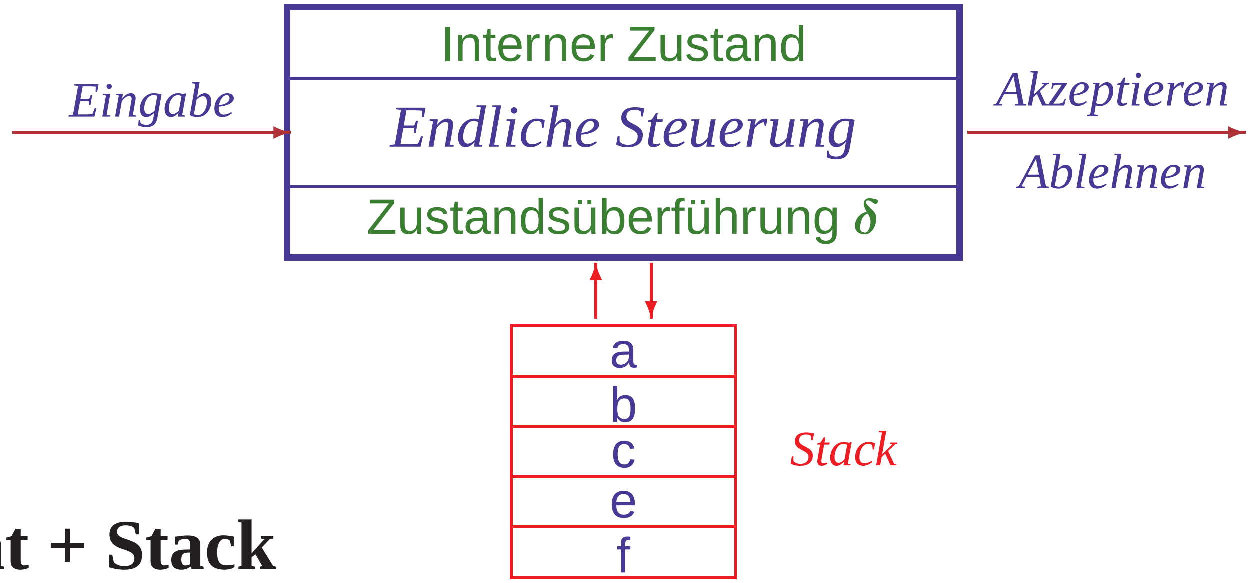
Analysiere das Verhalten von Linksableitungen

- **Links von der aktuellen Variablen A stehen nur erzeugte Terminalsymbole**
  - Entspricht den schon verarbeiteten Eingabesymbolen des Automaten
- **Rechts von A können bereits Terminalsymbole stehen Abarbeitung von A schiebt weiteren Text in die Mitte**
  - Bei Verarbeitung eines Eingabewortes muß der Automat Information speichern, welche Symbole am Ende des Wortes kommen müssen
- **Ist A komplett abgearbeitet, so "springt" die Ableitung über Terminalsymbole zur nächsten Variablen**
  - Automat muß zuletzt erzeugte Information zuerst abarbeiten

$$\Downarrow$$

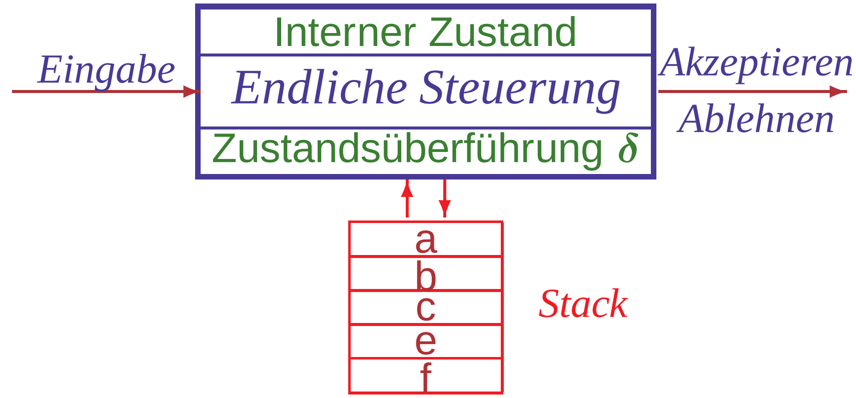
Speicher des Automaten sollte ein Stack sein

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 / spontane ϵ-Übergänge möglich

PUSHDOWN-AUTOMATEN – MATHEMATISCH PRÄZISIERT

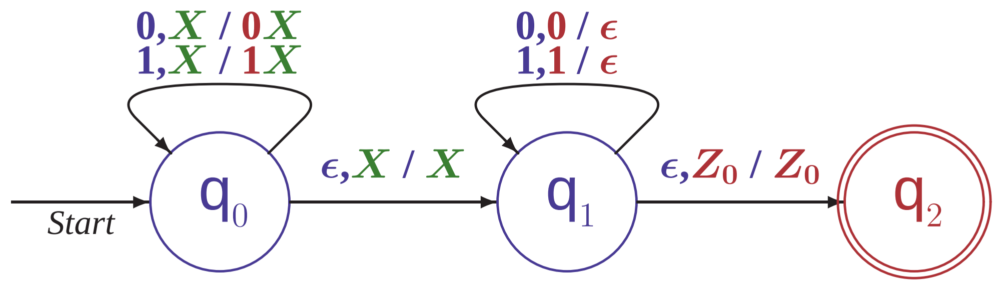


Ein **Pushdown-Automat** (**PDA, Kellerautomat**)
ist ein 7-Tupel $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ mit

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

Pushdown-Automaten sind üblicherweise nichtdeterministisch!

Beschreibung von Pushdown-Automaten

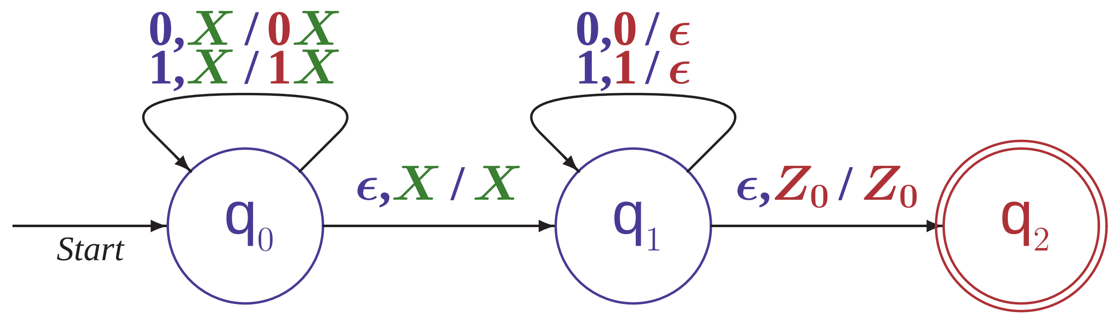


- **Übergangsdiagramme**
  - Jeder Zustand in Q wird durch einen Knoten (Kreise) dargestellt
  - Für $(p,\alpha)\in\delta(q,a,X)$, $a\in\Sigma\cup\{\epsilon\}$ hat das Diagramm eine Kante $q\xrightarrow{a,X/\alpha}p$
    (Mehrfachbeschriftungen und Platzhalter für beliebige $a\in\Sigma\cup\{\epsilon\}$, $X\in\Gamma$ möglich)
  - q_0 wird durch einen mit *Start* beschrifteten Pfeil angezeigt
  - Endzustände in F werden durch doppelte Kreise gekennzeichnet
  - Σ und Γ implizit durch 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
  - Platzhalter für $a\in\Sigma\cup\{\epsilon\}$, $X\in\Gamma$ erlaubt

	Q	$\Sigma+\epsilon$	Γ	Resultat
$\rightarrow$	q_0	0	X	q_0, 0X
$\rightarrow$	q_0	1	X	q_0, 1X
$\rightarrow$	q_0	ϵ	X	q_1, X
	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			

Pushdown-Automat für $\{ww^R \mid w \in \{0,1\}^*\}$



- **Speichere w in q_0**
  - Es wird je ein Symbol gelesen und auf den Stack gelegt
    - $\delta(q_0,a,X) = \{(q_0,aX)\}$ für $a \in \{0,1\}$, $X \in \Gamma$
- **Spontaner Wechsel "in der Mitte"**
    - $\delta(q_0,\epsilon,X) = \{(q_1,X)\}$ für $X \in \Gamma$ (nichtdeterministischer ϵ-Übergang)
- **Verarbeite w^R in q_1** (w steht in umgekehrter Reihenfolge im Stack)
  - Jedes gelesene Symbol wird dem obersten Stacksymbol verglichen
    - $\delta(q_1,a,a) = \{(q_1,\epsilon)\}$ für $a \in \{0,1\}$
- **"Leerer" Stack akzeptiert**
  - Wenn Stack leer ist, wurde w^R in q_1 verarbeitet
    - $\delta(q_1,\epsilon,Z_0) = \{(q_2,Z_0)\}$ (deterministischer ϵ-Übergang)

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

Arbeitsweise von Pushdown-Automaten

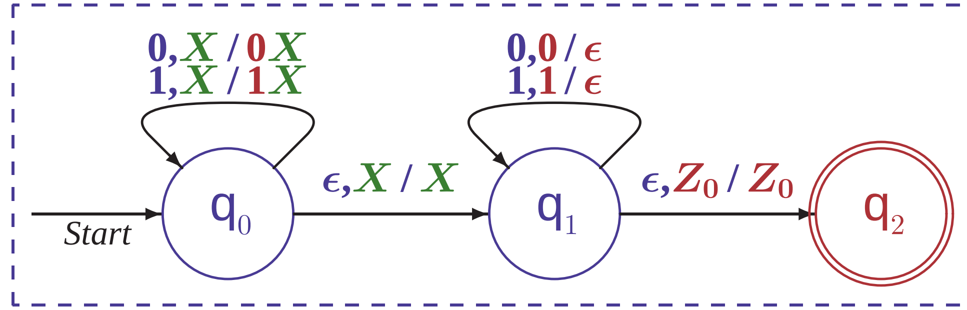
Generalisiere Konzept der Konfigurationsübergänge

- **Erweitere Begriff der Konfiguration**
  - Aktueller Zustand, unverarbeitete Eingabe und Inhalt des Stacks zählt
  - Formal dargestellt als Tripel $\boldsymbol{K} = (\boldsymbol{q},\boldsymbol{w},\boldsymbol{\gamma}) \in Q\times\Sigma^*\times\Gamma^*$
- **Modifiziere Konfigurationsübergangsrelation** $\vdash^*$
  - Wechsel zwischen Konfigurationen durch Abarbeitung von Wörtern
  - $(\boldsymbol{q}, \boldsymbol{aw}, \boldsymbol{X\beta}) \vdash (\boldsymbol{p}, \boldsymbol{w}, \boldsymbol{\alpha\beta})$, falls $(p,\alpha)\in\delta(q,a,X)$
  - $(\boldsymbol{q}, \boldsymbol{w}, \boldsymbol{X\beta}) \vdash (\boldsymbol{p}, \boldsymbol{w}, \boldsymbol{\alpha\beta})$, falls $(p,\alpha)\in\delta(q,\epsilon,X)$

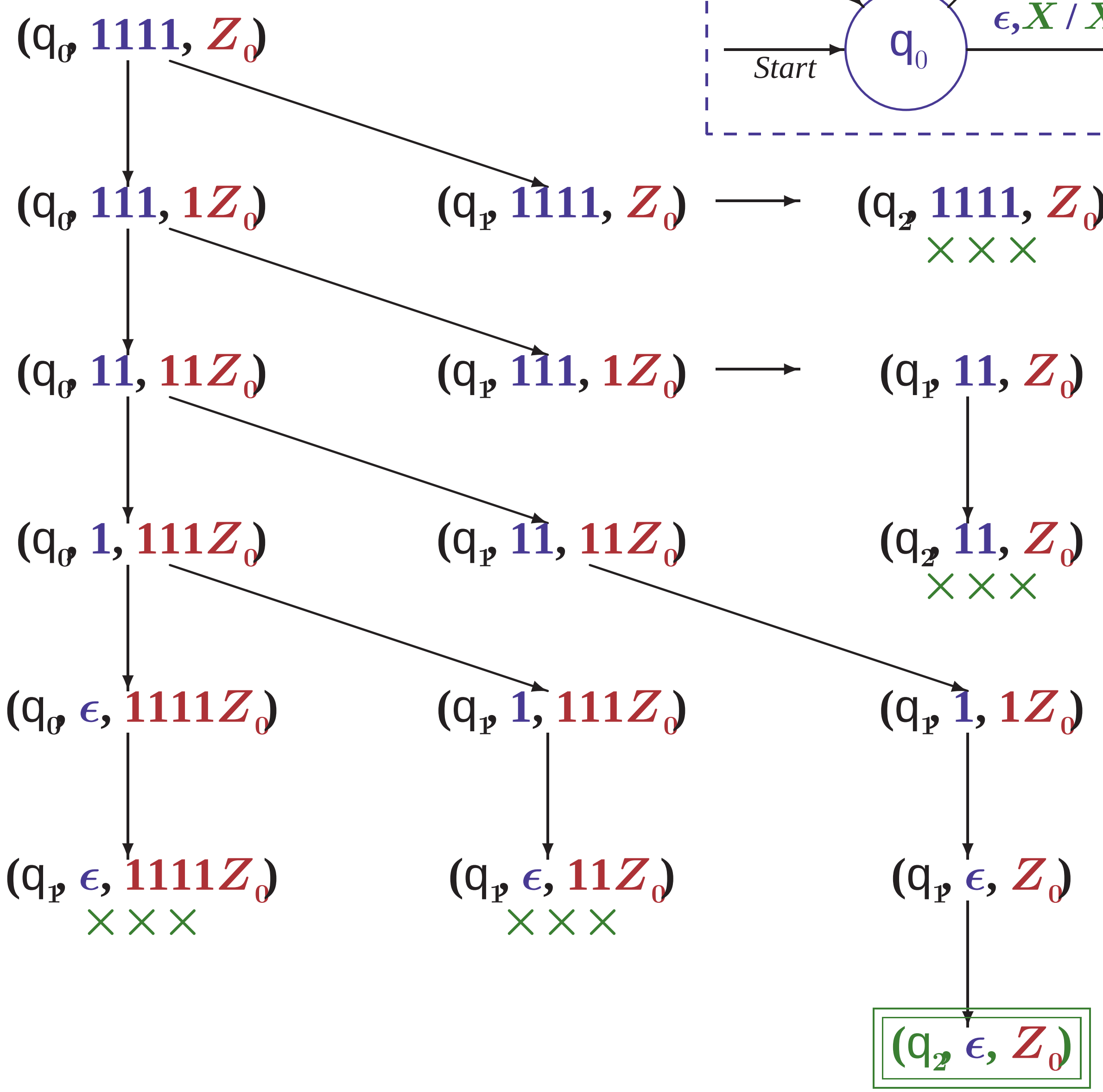
    (Im Zustand q ist a das erste Eingabesymbol und X oben im Stack. a wird abgearbeitet, X durch α ersetzt, der Rest bleibt stehen)
  - $\boldsymbol{K_1} \vdash^* \boldsymbol{K_2}$, falls $K_1 = K_2$ oder

    es gibt eine Konfiguration K mit $K_1 \vdash K$ und $K \vdash^* K_2$

Abarbeitung des Palindrom PDA



Verarbeitung von 1111



Akzeptierte Sprache eines Pushdown-Automaten

Zwei alternative Definitionen möglich

- **Akzeptanz durch akzeptierende Endzustände**

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

  – Standarddefinition: Nach Abarbeitung der Eingabe entscheidet der Zustand, ob das Wort akzeptiert wird

- **Akzeptanz durch leeren Stack**

  $$L_\epsilon(P) = \{\, w \in \Sigma^* \mid \exists q \in Q.\ (q_0, w, Z_0) \vdash^* (q, \epsilon, \epsilon)\,\}$$

  – Oft praktischer: Nach Abarbeitung der Eingabe sind auch alle zwischengelagerten Symbole verarbeitet

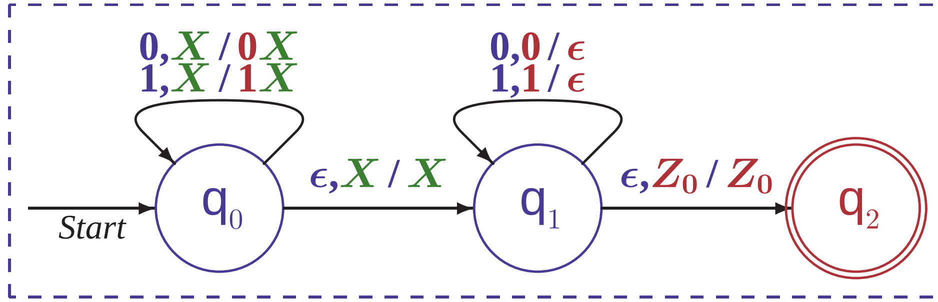
- **Definitionen haben verschiedene Effekte**

  – Sprachen können für konkrete PDAs sehr verschieden ausfallen

- **Beide Definitionen sind gleichmächtig**

  – PDA kann passend zur anderen Definition umgewandelt werden

Die beiden Sprachen des Palindromautomaten



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

  $\supseteq$: Durch strukturelle Induktion zeige, daß für jedes Wort w gilt

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

  $\subseteq$: Durch strukturelle Induktion über $x = x_1..x_n$ zeige

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

  Kernidee des Induktionsschrittes (Details in HMU §6.2.1)

  Wenn $(q_0,x_1..x_n,\alpha') \vdash (q_0,x_2..x_n,x_1\alpha') \vdash^* (q_1,x_i..x_n,\beta x_1\alpha')$
  $\vdash^* (q_1,x_n,x_1\alpha') \vdash (q_1,\epsilon,\alpha')$ für $\alpha',\beta \in \Gamma^*$,

  dann folgt $(q_0,x_1..x_{n-1},\alpha') \vdash (q_0,x_2..x_{n-1},x_1\alpha') \vdash^* \ldots \vdash^* (q_1,\epsilon,x_1\alpha')$
  und $x_1{=}x_n$ und per Induktion $x_2..x_{n-1} = vv^R$ für ein $v \in \{0{,}1\}^*$

- $L_\epsilon(P) = \emptyset$ weil Z_0 nie gelöscht wird

  Modifikation von P: Ändere Kantenbeschriftung von q_1 nach q_2 in $\epsilon, Z_0 / \epsilon$

  Für den resultierenden PDA P' gilt: $L_\epsilon(P') = L_F(P) = \{ww^R \mid w \in \{0,1\}^*\}$

Wichtige Erkenntnisse zu Aussagen über Konfigurationsübergänge in Beweisen

- **Ungelesene Eingaben können ignoriert werden**

  Gilt $(q,xw,\alpha) \vdash^* (p,yw,\beta)$ **dann gilt auch**

  $(q,x,\alpha) \vdash^* (p,y,\beta)$ **für alle** $w \in \Sigma^*$

  Dagegen kann es von Bedeutung sein, ob im Stack hinter α etwas steht

  – Beweis durch Induktion über Anzahl der Konfigurationsschritte

  – Kernargument: $(q,ayw,X\beta) \vdash (p,yw,\gamma\beta)$ verlangt $(p,\gamma) \in \delta(q,a,X)$

  also $(q,ay,X\beta) \vdash (p,y,\gamma\beta)$

- **Erweiterung von Eingabe oder Stack ändert nichts**

  Gilt $(q,x,\alpha) \vdash^* (p,y,\beta)$ **dann gilt auch**

  $(q,xw,\alpha\gamma) \vdash^* (p,yw,\beta\gamma)$ **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: $(q,aw,X\gamma) \vdash (p,w,\beta\gamma)$, falls $(p,\beta) \in \delta(q,a,X)$

  was hinter a bzw. X kommt, bleibt unangetastet

Erkennen mit leerem Stack ist oft einfacher

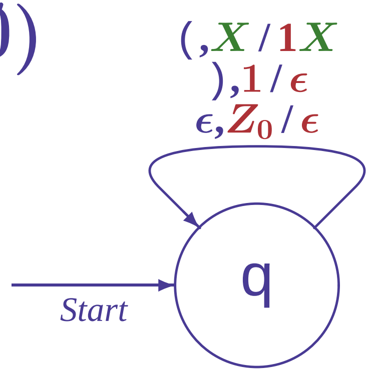
Konstruiere PDA für korrekte Klammerausdrücke

- **Rahmenbedingungen an Eingabewörter**
  - Anzahl geöffneter und geschlossener Klammern muß gleich sein
  - In keinen Anfangssegment dürfen mehr (als) vorkommen
- **Zähle Überschuß geöffneter Klammern im Stack**
  - Jedes (erhöht die Anzahl, jedes) erniedrigt sie
  -) ist nicht erlaubt, wenn der Stackboden erreicht ist
  - Am Ende des Wortes wird der Stackboden entfernt
- **Setze** $P_1 = (\{q\}, \{(,)\}, \{Z_0, 1\}, \delta, q, Z_0, \emptyset)$

  mit $\delta(q, (, X) = \{(q, 1X)\}$

  $\delta(q,), 1) = \{(q, \epsilon)\}$

  $\delta(q, \epsilon, Z_0) = \{(q, \epsilon)\}$



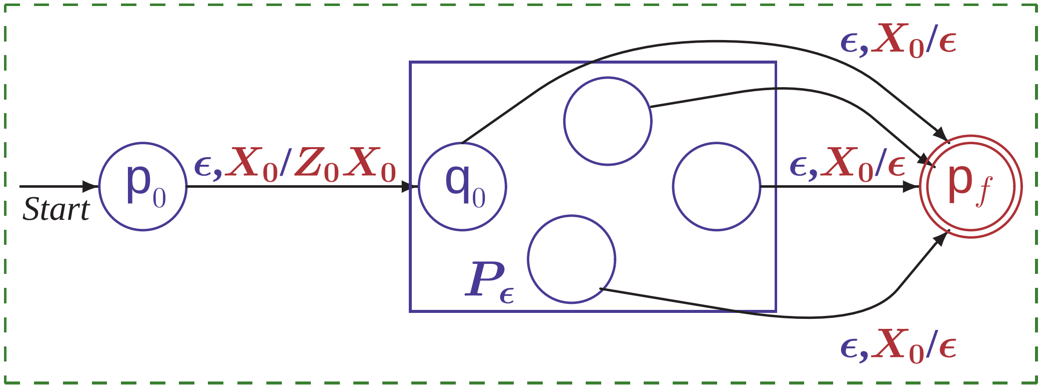
Transformation von L_ϵ– in L_F–Automaten

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

- **Bei leerem Stack wechsele in einen Endzustand**
  - Neues Initialsymbol X_0 markiert unteres Ende des Stacks von P_F
  - 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
- $P_F = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})$
  - $\delta_F(q,a,X) = \delta(q,a,X)$

    für alle $q \in Q$, $X \in \Gamma$
  - $\delta_F(p_0,\epsilon,X_0) = \{(q_0,Z_0X_0)\}$
  - $\delta_F(q,\epsilon,X_0) = \{(p_f,\epsilon)\}$

    für alle $q \in Q$



Korrektheitsbeweis durch Detailanalyse

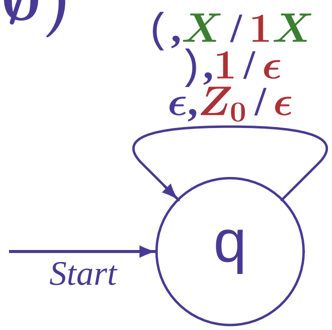
Umwandlung eines L_ϵ-PDA in einen L_F-PDA

- **Gegeben** $P_\epsilon = (\{q\}, \{(,)\}, \{Z_0, 1\}, \delta, q, Z_0, \emptyset)$

  mit $\delta(q, (, X) = \{(q, 1X)\}$

  $\delta(q,), 1) = \{(q, \epsilon)\}$

  $\delta(q, \epsilon, Z_0) = \{(q, \epsilon)\}$



- **Äquivalenter PDA** P_F **mit Endzuständen ist**

  $(\{p_0, q, p_f\}, \{(,)\}, \{X_0, Z_0, 1\}, \delta_F, p_0, X_0, \{p_f\})$

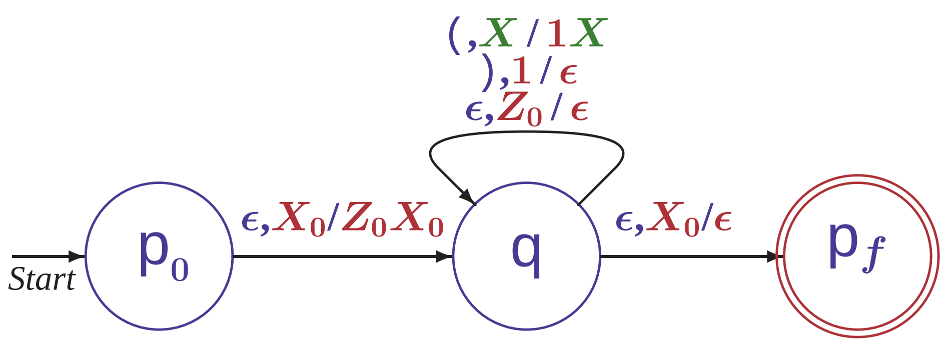
  mit $\delta_F(p_0,\epsilon,X_0) = \{(q,Z_0X_0)\}$

  $\delta_F(q, (, X) = \{(q, 1X)\}$

  $\delta_F(q,), 1) = \{(q, \epsilon)\}$

  $\delta(q, \epsilon, Z_0) = \{(q, \epsilon)\}$

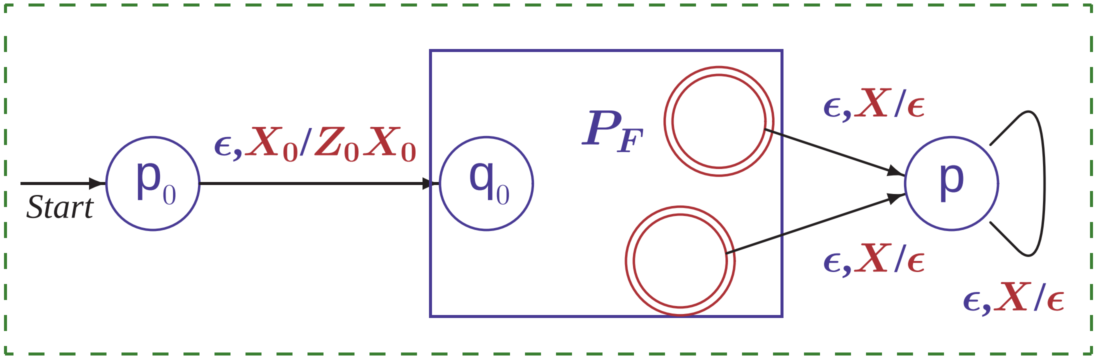
  $\delta_F(q,\epsilon,X_0) = \{(p_f,\epsilon)\}$



Transformation von L_F– in L_ϵ–Automaten

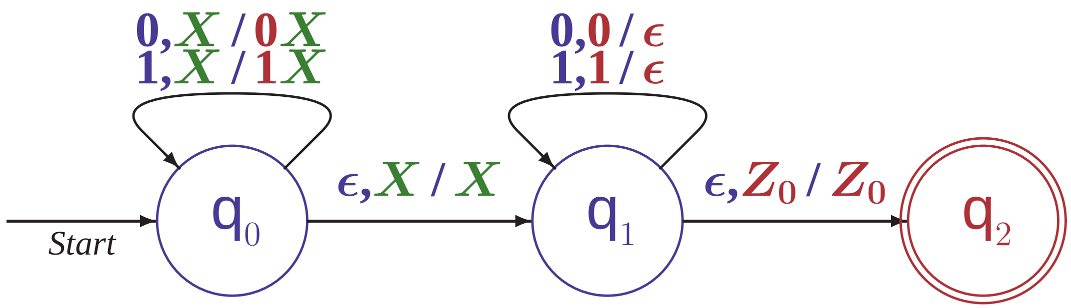
Zu jedem PDA $P_F = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ kann ein PDA P_ϵ konstruiert werden mit $L_F(P_F) = L_\epsilon(P_\epsilon)$

- **Im Endzustand leere den Stack**
  - Neuer Stacklösch-Zustand p, in den 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
- $P_\epsilon = (Q \cup \{p_0, p\}, \Sigma, \Gamma \cup \{X_0\}, \delta_\epsilon, p_0, X_0, \emptyset)$
  - $\delta_\epsilon(q,a,X) = \delta(q,a,X)$ für alle $q \in Q$, $X \in \Gamma$
  - $\delta_\epsilon(p_0,\epsilon,X_0) = \{(q_0,Z_0X_0)\}$
  - $\delta_\epsilon(q,\epsilon,X) = \{(p,\epsilon)\}$ für alle $q \in F$
  - $\delta_\epsilon(p,\epsilon,X) = \{(p,\epsilon)\}$ für alle $X \in \Gamma \cup \{X_0\}$



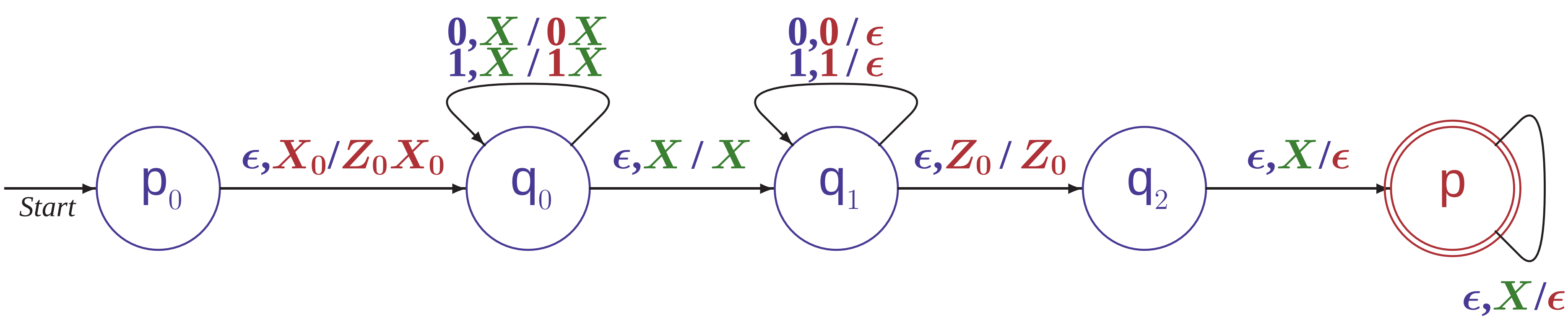
Korrektheitsbeweis durch Detailanalyse

UMWANDLUNG EINES L_F-PDA IN EINEN L_ϵ-PDA



- $P_F = (\{q_0, q_1, q_2\}, \{0,1\}, \{0,1,Z_0\}, \delta, q_0, Z_0, \{q_2\})$
  mit δ wie oben erkennt $\{ww^R \mid w \in \{0,1\}^*\}$ mit Endzustand

- **Äquivalenter PDA P_ϵ mit leerem Stack ist**
  $(\{p_0, q_0, q_1, q_2, p\}, \{0,1\}, \{0,1,Z_0,X_0\}, \delta_\epsilon, p_0, X_0, \{p\})$



Sind PDAs wirklich 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 verarbeitet 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 Terminalteilwörter 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 bei Abarbeitung des Stacksymbols X mit δ Zwischenwörter 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 im Stack durch rechte Seite β einer Regel $A \to \beta$ ersetzt
  - $a \in T$ wird vom Stack entfernt, wenn es als Eingabe erscheint, um im Stack die nächsten Variable einer Linksableitung freizulegen
- **Generierter PDA $P = (\{q\}, T, V \cup T, \delta, q, S, \emptyset)$**
  - $\delta(q,\epsilon,A) = \{(q,\beta) \mid A \to \beta \in P_G\}$ für alle $A \in V$
  - $\delta(q,a,a) = \{(q,\epsilon)\}$ für alle $a \in T$
- **Korrektheit: $L(G) = L_\epsilon(P)$** (Detailbeweis im Anhang)

Zeige: ($\subseteq$) Wenn $S = x_1A_1\alpha_1 \ldots \longrightarrow_L x_mA_m\alpha_m \longrightarrow_L w \in T^*$ dann gibt es für alle i ein y_i mit $w = x_iy_i$ und $(q,w,S) \vdash^* (q,y_i,A_i\alpha_i)$

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

Umwandlung einer Grammatik in einen PDA

- $G_7 = (\{E, I\},\ \{a, b, c, 0, 1, +, *, (,)\},\ P_G,\ E)$
  mit $P_G = \{\ E \rightarrow I \mid E{+}E \mid E{*}E \mid (E)$
  $I \rightarrow a \mid b \mid c \mid Ia \mid Ib \mid Ic \mid I0 \mid I1\ \}$

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

VON PUSHDOWN-AUTOMATEN ZU GRAMMATIKEN

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

- **Simuliere Abarbeitung eines Symbols vom Stack**
  - Verarbeite Variablen der Form “(q, X, p)” mit impliziter Bedeutung “*Entfernen von X kann von Zustand q zu Zustand p führen*”
  - Entfernen von X kann zuerst ein $Y_1..Y_m$ auf- und dann abbauen
  - Beginne mit Erzeugung von Z_0 und zeige, daß Z_0 entfernt werden kann
- **Generiere $G = (\{S\} \cup Q \times \Gamma \times Q,\ \Sigma,\ P_G,\ S)$** mit
  - $S \rightarrow (q_0, Z_0, q) \in P_G$ für alle $q \in Q$
  - $(q, X, p) \rightarrow a\,(q', Y_1, q_1)...(q_{m-1}, Y_m, p) \in P_G$, für beliebige Kombinationen $q_1, .., q_{m-1} \in Q$, falls $(q', Y_1..Y_m) \in \delta(q, a, X)$
    $(q, X, p) \rightarrow a \in P_G$, falls $(p, \epsilon) \in \delta(q, a, X)$
- **Korrektheitsbeweis $L_\epsilon(P) = L(G)$**
  - Zeige: $(q, X, p) \xrightarrow{*} w \in \Sigma^*$ genau dann, wenn $(q, w, X) \vdash^* (p, \epsilon, \epsilon)$
    $\subseteq$: Induktion über Länge der PDA Berechnung
    $\supseteq$: Induktion über Länge der Ableitung (viele Details)

Umwandlung eines PDA in eine Grammatik

- **Gegeben** $P = (\{q_0, q_1\}, \{a, b, c\}, \{A, B, Z_0\}, \delta, q_0, Z_0, \{q_1\})$

mit $\delta(q_0, a, X) = \{(q_0, AX)\}$ für $X \in \Gamma$

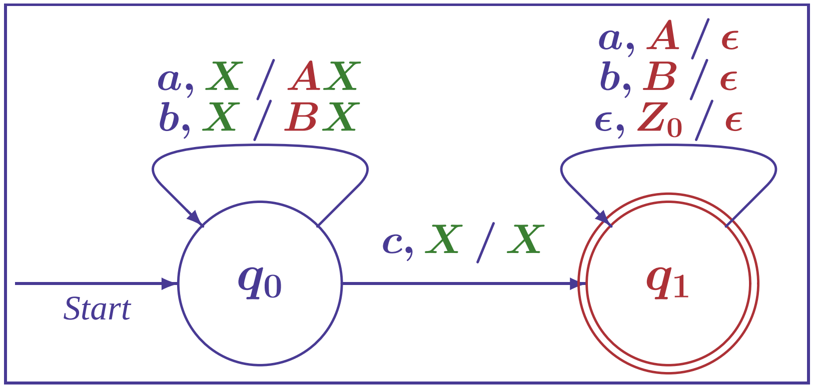
$\delta(q_0, b, X) = \{(q_0, BX)\}$ für $X \in \Gamma$

$\delta(q_0, c, X) = \{(q_1, X)\}$ für $X \in \Gamma$

$\delta(q_1, a, A) = \{(q_1, \epsilon)\}$

$\delta(q_1, b, B) = \{(q_1, \epsilon)\}$

$\delta(q_1, \epsilon, Z_0) = \{(q_1, \epsilon)\}$



- **Generiere** $G = (V_G, \{a, b, c\}, P_G, S)$

mit $V_G = \{S,\ (q_0, A, q_0),\ (q_0, B, q_0),\ (q_0, Z_0, q_0),\ (q_0, A, q_1), .., \ (q_1, Z_0, q_1)\}$

mit $P_G = \quad S \to (q_0, Z_0, q_0) \mid (q_0, Z_0, q_1)$

$$(q_0, X, q_0) \to a(q_0, A, q_0)(q_0, X, q_0) \mid b(q_0, B, q_0)(q_0, X, q_0) \mid c(q_1, X, q_0)$$
$$\mid a(q_0, A, q_1)(q_1, X, q_0) \mid b(q_0, B, q_1)(q_1, X, q_0)$$

$$(q_0, X, q_1) \to a(q_0, A, q_0)(q_0, X, q_1) \mid b(q_0, B, q_0)(q_0, X, q_1) \mid c(q_1, X, q_1)$$
$$\mid a(q_0, A, q_1)(q_1, X, q_1) \mid b(q_0, B, q_1)(q_1, X, q_1)$$

$(q_1, A, q_1) \to a, \quad (q_1, B, q_1) \to b, \quad (q_1, Z_0, q_1) \to \epsilon$

Keine Regel für (q_1, X, q_0), *da* $\delta(q_1, x, X) = \{(q_1, \epsilon)\}$ *für alle* x, X

G kann optimiert werden: streiche überflüssige Variablen, vereinfache Variablennamen

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

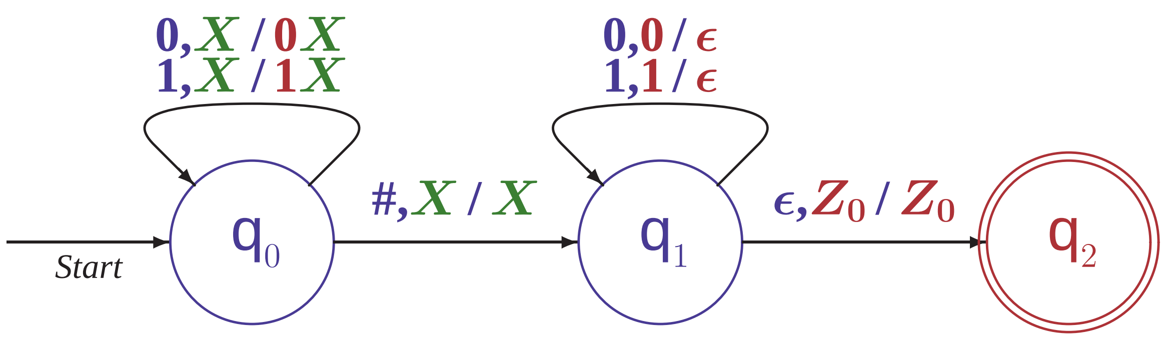
Deterministische Pushdown-Automaten präzisiert

- **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**
  - $\delta{:}Q\times(\Sigma\cup\{\epsilon\})\times\Gamma \to Q\times\Gamma^*$ **Überführungsfunktion**
    $\delta(q,\epsilon,X)$ nur definiert, wenn $\delta(q,a,X)$ für alle $a\in\Sigma$ undefiniert
  - $q_0\in Q$ **Startzustand**
  - $Z_0\in\Gamma$ **Initialsymbol des Stacks**
  - $F\subseteq Q$ Menge von **akzeptierenden** (End-)**Zuständen**

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

L_F-DPDAs sind mächtiger als DEAs

- $\mathcal{L}_3 = L(DEA) \subseteq \mathcal{L}_{DPDA}^F$
  - Jeder DEA ist ein spezieller DPDA, der mit Endzustand akzeptiert
- $L = \{w\#w^R \mid w \in \{0,1\}^*\} \in \mathcal{L}_{DPDA}^F - \mathcal{L}_3$
  - **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 $\delta(q,a,Z_0)$ für alle $a \in \Sigma$ undefiniert ist

L_ϵ-DPDAs sind zu ausdrucksschwach

- $\mathcal{L}^\epsilon_{DPDA} \subseteq \mathcal{L}^F_{DPDA}$

  – Transformation von L_ϵ- in L_F-PDAs erhält Bedingung an DPDAs

- **L_ϵ-DPDAs erkennen nur präfixfreie Sprachen**

  "*kein Wort aus $L_\epsilon(P)$ ist echtes Präfix eines anderen Wortes aus $L_\epsilon(P)$*"

  Intuitiv: Wenn der Stack einmal leer ist, kann P nicht weiterarbeiten

  – Ist $u \in L_\epsilon(P)$ dann stoppt P nach Verarbeitung von u mit leerem Stack

  – Eine echte Erweiterung von u wird von P nicht komplett abgearbeitet und damit nicht akzeptiert

- $\mathcal{L}_3 \not\subseteq \mathcal{L}^\epsilon_{DPDA}$

  – $\{0\}^* \notin \mathcal{L}^\epsilon_{DPDA}$, da $\{0\}^*$ nicht präfixfrei ist

- **L_ϵ-DPDAs und L_F-DPDAs sind nicht äquivalent**

  – Aus $\{0\}^* \notin \mathcal{L}^\epsilon_{DPDA}$ folgt auch $\mathcal{L}^F_{DPDA} \not\subseteq \mathcal{L}^\epsilon_{DPDA}$

$\mathcal{L}^F_{DPDA}$ IST ECHTE TEILKLASSE VON $\mathcal{L}_2$

- $\boldsymbol{\mathcal{L}^F_{DPDA} \subseteq \mathcal{L}_2}$
  – Jeder DPDA ist ein spezieller PDA

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

  Intuitiv: DPDA P kann die Mitte eines Wortes nicht identifizieren

  – P muß $0^n 110^n$ akzeptieren, Stack wird beim Vergleich geleert (großes n)

  – P muß auch akzeptieren, wenn noch einmal $0^n 110^n$ folgt

  P darf nicht akzeptieren, wenn stattdessen $0^m 110^m$ $(m \neq n)$ folgt

  – Aber P kann nicht unterscheiden, da n nicht mehr gespeichert ist

  Technisches Argument: Abschlußeigenschaften + Pumping Lemma für $\mathcal{L}_2$

- **L_F-DPDAs bilden eine eigene wichtige Sprachklasse**
  – Deterministisch kontextfreie Sprachen

Deterministische Sprachen sind eindeutig HMU §6.4.4

- **$\mathcal{L}^{\epsilon}_{DPDA}$-Sprachen sind eindeutig**
  - Standardumwandlung von L_ϵ-(D)PDA P liefert Typ-2 Grammatik
  - Konfigurationsübergangsfolgen bestimmen Linksableitungen eindeutig
  - Für jede Eingabe w gibt es in P nur einen Konfigurationsübergang
  - Erzeugte Grammatik ist eindeutig
- **$\mathcal{L}^{F}_{DPDA}$-Sprachen sind eindeutig**
  - Mache Sprache von L_F-DPDA P präfixfrei durch Anhängen von \$
  - Präfixfreie Sprachen in $\mathcal{L}^{F}_{DPDA}$ sind $\mathcal{L}^{\epsilon}_{DPDA}$-Sprachen
  - Damit hat die modifizierte Sprache eine eindeutige Grammatik G
  - G' mit zusätzlicher Regel $\$\rightarrow\epsilon$ (und Variable \$) erzeugt Sprache von P
  - Linksableitungen in G' sind “identisch” zu denen in G, also eindeutig
- **Nicht jede eindeutige Typ-2 Sprache ist deterministisch**
  - $\{ww^R \mid w\in\{0,1\}^*\}$ ist eindeutig, aber nicht deterministisch erkennbar

Pushdown-Automaten – Zusammenfassung

- **Maschinenmodell für kontextfreie Sprachen**
  - Nichtdeterministischer endlicher Automat mit Stack und ϵ-Übergängen
  - Erkennung von Wörtern 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

ANHANG

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 im Stack durch rechte Seite β einer Regel $A\to\beta$ ersetzt
  - $a\in T$ wird vom Stack entfernt, wenn es als Eingabe erscheint, um im Stack die nächsten Variable einer Linksableitung freizulegen

- **Generierter PDA $P=(\{q\},T,V\cup T,\delta,q,S,\emptyset)$**
  - $\delta(q,\epsilon,A)=\{(q,\beta)\mid A\to\beta\in P_G\}$ für alle $A\in V$
  - $\delta(q,a,a)=\{(q,\epsilon)\}$ für alle $a\in T$

- **Korrektheit: zeige $L(G)=L_\epsilon(P)$** (nächste 2 Folien)

Korrektheitsbeweis im Detail: $L(G) \subseteq L_\epsilon(P)$

Wenn $S = x_1 A_1 \alpha_1 .. \longrightarrow_L x_m A_m \alpha_m \longrightarrow_L w \in T^*$ $(x_i \in T^*, A_i \in V)$ **dann gibt es für alle i ein y_i mit $w{=}x_i y_i$ und** $(q, w, S) \vdash^* (q, y_i, A_i \alpha_i)$

- **Beweis durch Induktion über $i \leq m$**
- **Basisfall $i = 1$:** $S = x_1 A_1 \alpha_1 \xrightarrow{*} w$
  - Es folgt $S = A_1$ und $x_1{=}\alpha_1{=}\epsilon$, also muß $y_1{=}w$ gewählt werden
  - $(q, w, S) \vdash^* (q, w, S)$ gilt mit 0 Konfigurationsübergängen
- **Induktionsschritt:** $S .. \longrightarrow_L x_i A_i \alpha_i \longrightarrow_L x_{i+1} A_{i+1} \alpha_{i+1} \xrightarrow{*} w$
  - $x_i A_i \alpha_i \longrightarrow_L x_{i+1} A_{i+1} \alpha_{i+1}$ verlangt $A_i \rightarrow \beta_i \in P_G$ für ein β_i,
    wobei $x_i \beta_i \alpha_i = x_i z A_{i+1} \alpha_{i+1}$ für ein $z \in T^*$ und $x_{i+1} = x_i z \sqsubseteq w$.
  - Per Konstruktion gilt dann $(q,\beta_i) \in \delta(q,\epsilon,A_i)$ und mit der Induktionsannahme folgt $(q, w, S) \vdash^* (q, y_i, A_i \alpha_i) \vdash (q, y_i, z A_{i+1} \alpha_{i+1})$
  - Wegen $x_{i+1} = x_i z \sqsubseteq w$ und $w = x_i y_i$ kann y_i zerlegt werden in $z y_{i+1}$ und der PDA arbeitet z ab: $(q, y_i, z A_{i+1} \alpha_{i+1}) \vdash^* (q, y_{i+1}, A_{i+1} \alpha_{i+1})$
- **Schlußfolgerung:** $S = x_1 A_1 \alpha_1 .. \longrightarrow_L x_m A_m \alpha_m \longrightarrow_L w \in T^*$
  - Wegen $w \in T^*$ folgt $A_m \rightarrow \beta_m \in P_G$ für ein $\beta_m \in T^*$ und $w = x_m \beta_m \alpha_m$
  - Also $(q, w, S) \vdash^* (q, \beta_m \alpha_m, A_m \alpha_m) \vdash (q, \beta_m \alpha_m, \beta_m \alpha_m) \vdash^* (q, \epsilon, \epsilon)$,
    d.h. $w \in L_\epsilon(P)$

Korrektheitsbeweis im Detail: $L(G) \supseteq L_\epsilon(P)$

Für alle $X \in V$ gilt: 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 $X \to \epsilon \in P_G$ 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 $X \to Y_1..Y_k \in P_G$ haben $(Y_i \in V \cup T)$
  - Dann gibt eine Zerlegung $w = w_1..w_k$ mit $(q, w_1 w_2..w_k, Y_1 Y_2..Y_k) \vdash^* (q, w_2..w_k, Y_2..Y_k) \vdash^* (q, \epsilon, \epsilon)$
  - Es folgt $(q, w_i w_{i+1}..w_k, Y_i) \vdash^* (q, w_{i+1}..w_k, \epsilon)$ also $(q, w_i, Y_i) \vdash^* (q, \epsilon, \epsilon)$
  - Per Induktionsannahme folgt $Y_i \xrightarrow{*} w_i$ für alle i (für $Y_i \in T$ ist $Y_i = w_i$) also $X \longrightarrow Y_1..Y_k \xrightarrow{*} w_1..w_k = w$
- **Es folgt $L_\epsilon(P) = \{w \mid (q, w, S) \vdash^* (q, \epsilon, \epsilon)\} \subseteq \{w \mid S \xrightarrow{*} w\} = L(G)$**