

# 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

Maschinenmodell für Typ-3 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

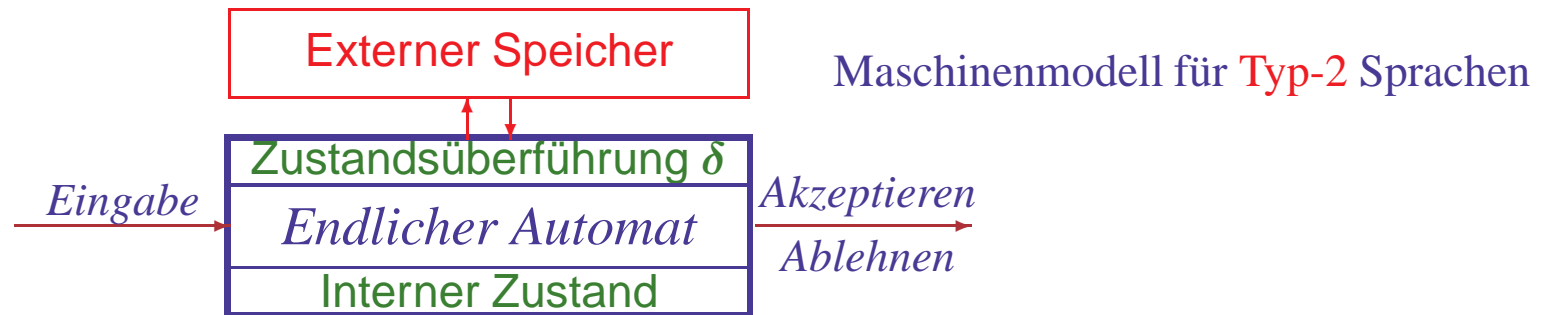
# EIN MASCHINENMODELL FÜR TYP-2 SPRACHEN

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

# 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 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**

## **Analysiere das Verhalten von Linksableitungen**

### Analysiere das Verhalten von Linksableitungen

- **Links** von der aktuellen Variablen  $A$  stehen **nur** erzeugte **Terminalsymbole**
  - Entspricht den **schon verarbeiteten Eingabesymbolen** des Automaten

### 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**

### 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**
  - Automaten muß **zuletzt erzeugte Information zuerst abarbeiten**



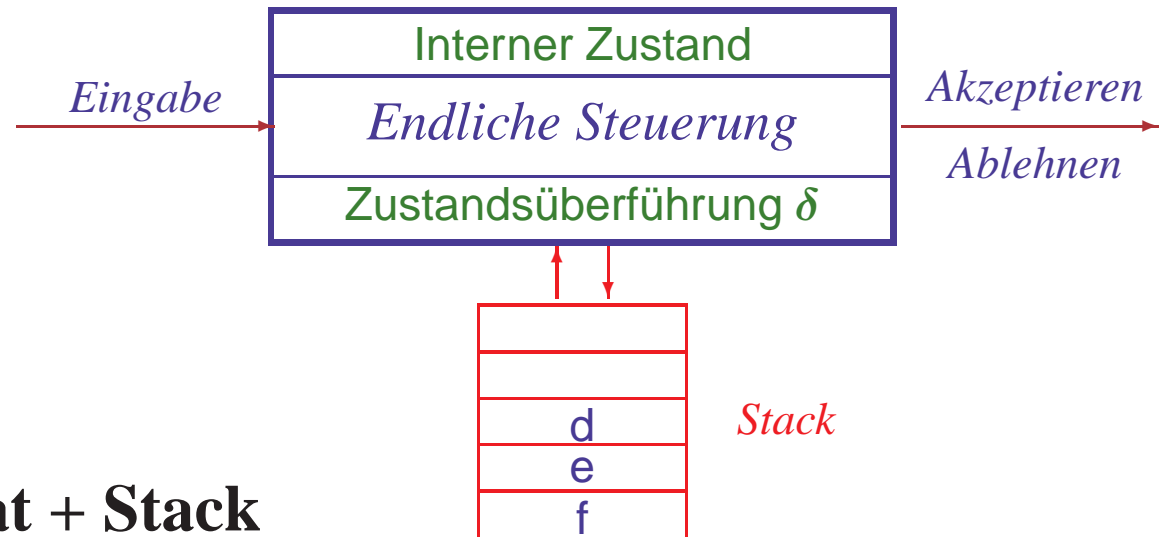
### Analysiere das Verhalten von Linksableitungen

- **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**
  - Automaten muß **zuletzt erzeugte Information zuerst abarbeiten**



**Speicher des Automaten sollte ein Stack sein**

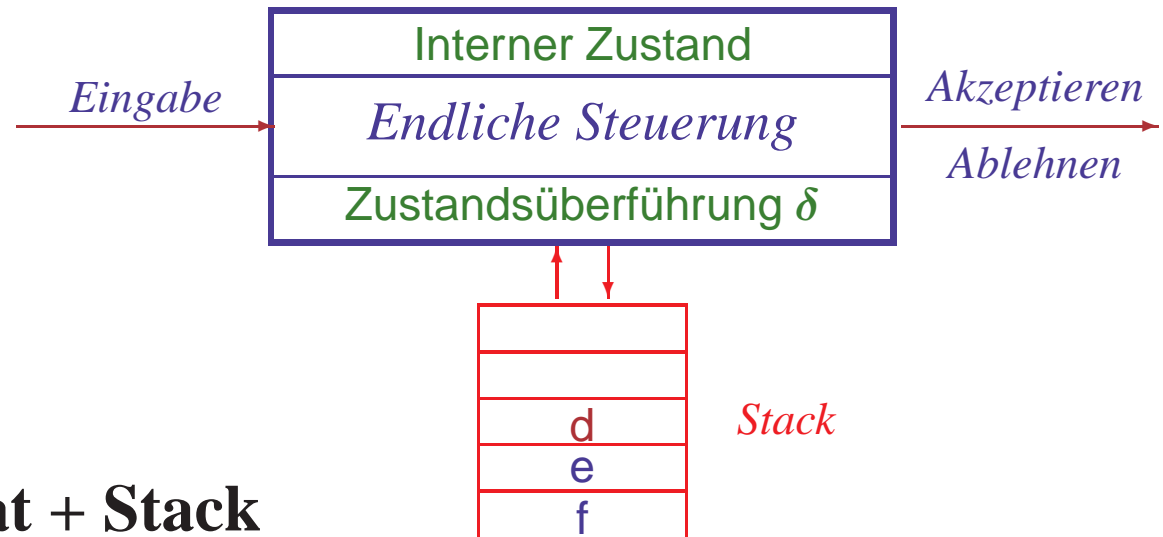
# PUSHDOWN-AUTOMATEN INTUITIV



- **Endlicher Automat + Stack**

- Endliche Steuerung liest Eingabesymbole

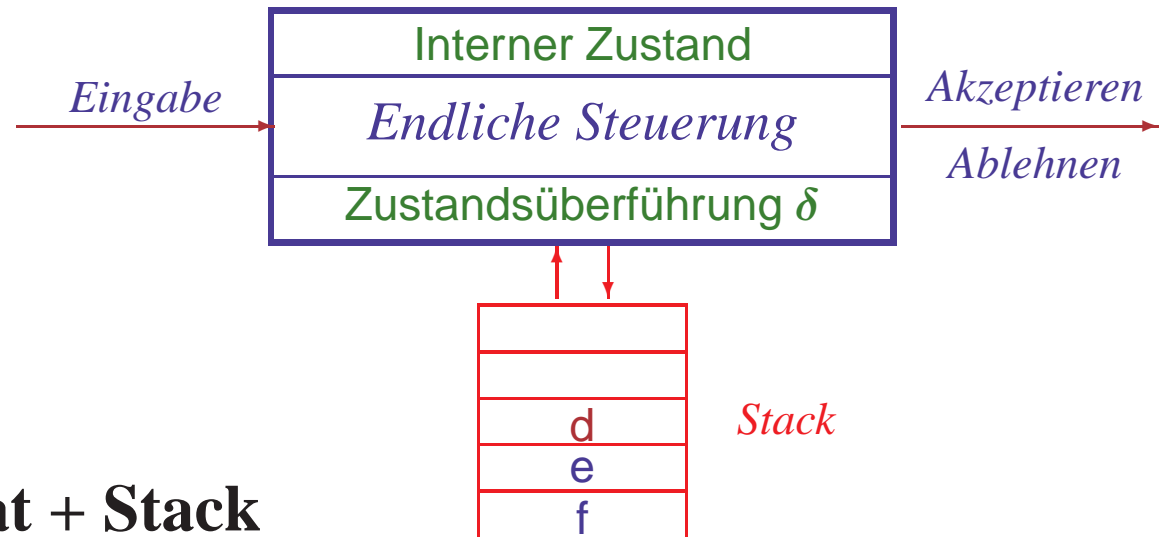
# PUSHDOWN-AUTOMATEN INTUITIV



- **Endlicher Automat + Stack**

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

# PUSHDOWN-AUTOMATEN INTUITIV



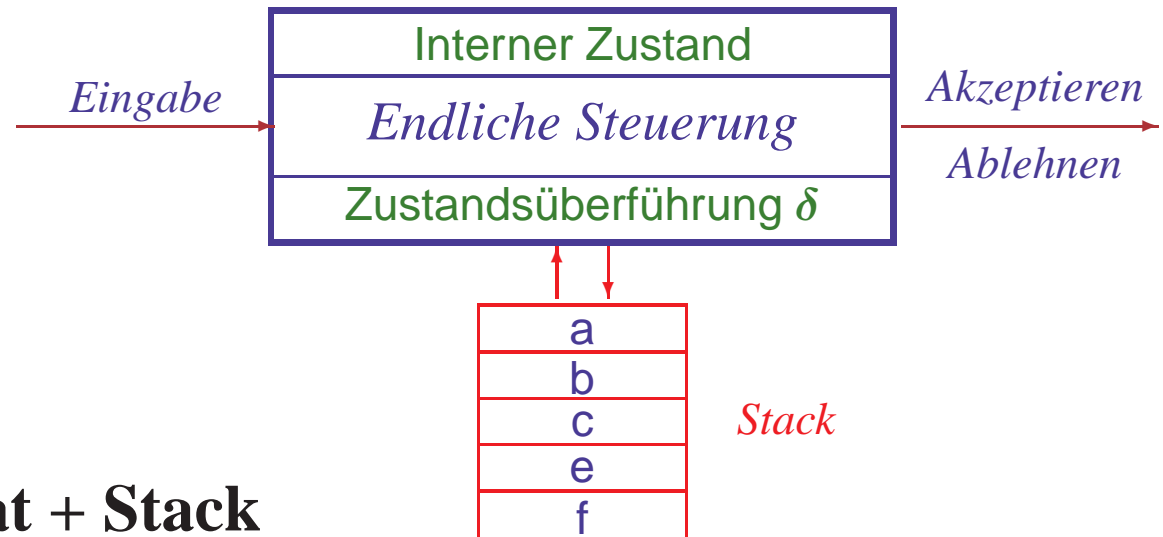
- **Endlicher Automat + Stack**

- 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

# PUSHDOWN-AUTOMATEN INTUITIV



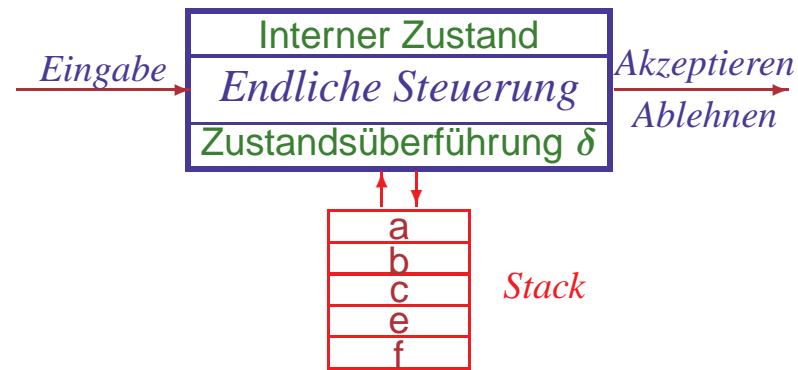
- **Endlicher Automat + Stack**

- 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  $\epsilon$ -Übergänge möglich

# PUSHDOWN-AUTOMATEN – MATHEMATISCH PRÄZISIERT



Ein **Pushdown-Automat (PDA, Kellerautomat)**

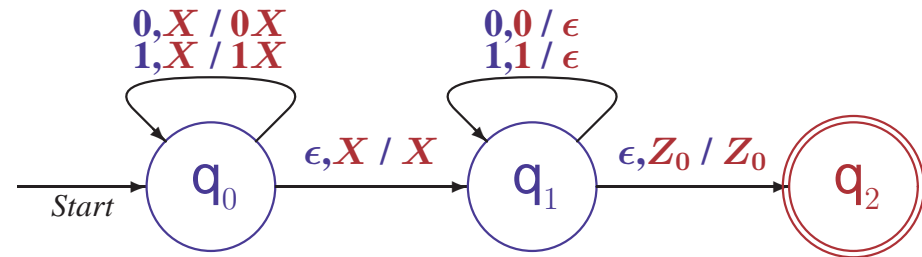
ist ein 7-Tupel  $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  mit

- $Q$  nichtleere endliche **Zustandsmenge**
- $\Sigma$  endliches **Eingabealphabet**
- $\Gamma$  endliches **Stackalphabet**
- $\delta: Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow \mathcal{P}_e(Q \times \Gamma^*)$  **Überföhrungsfunktion** (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

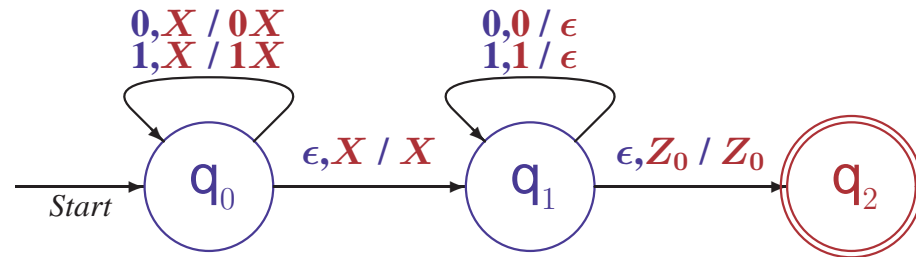
## • Übergangsdiagramme



- 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**  $q \xrightarrow{a, X / \alpha} p$  (mehrere Beschriftungen derselben Kante möglich)
- $q_0$  wird durch einen mit *Start* beschrifteten Pfeil angezeigt
- Endzustände in  $F$  werden durch **doppelte Kreise** gekennzeichnet
- $\Sigma$  und  $\Gamma$  implizit durch Diagramm bestimmt, **Initialsymbol** heißt  $Z_0$

# 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**  $q \xrightarrow{a, X/\alpha} p$  (mehrere Beschriftungen derselben Kante möglich)
- $q_0$  wird durch einen mit *Start* beschrifteten Pfeil angezeigt
- Endzustände in  $F$  werden durch **doppelte Kreise** gekennzeichnet
- $\Sigma$  und  $\Gamma$  implizit durch Diagramm bestimmt, **Initialsymbol** heißt  $Z_0$

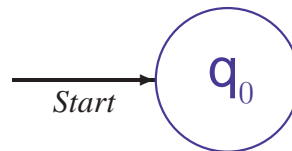
## • Übergangstabellen

- Tabellarische Darstellung der Funktion  $\delta$
- Kennzeichnung von  $q_0$  durch einen Pfeil
- Kennzeichnung von  $F$  durch Sterne
- $\Sigma$ ,  $\Gamma$  und  $Q$  implizit durch die Tabelle bestimmt
- **Wildcardvariablen** für  $a \in \Sigma \cup \{\epsilon\}$ ,  $X \in \Gamma$  erlaubt

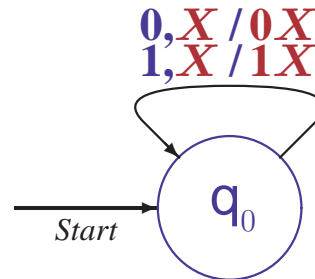
	$Q$	$\Sigma + \epsilon$	$\Gamma$	Resultat
→	$q_0$	0	$X$	$q_0, 0X$
→	$q_0$	1	$X$	$q_0, 1X$
→	$q_0$	$\epsilon$	$X$	$q_1, X$
	$q_1$	0	0	$q_1, \epsilon$
	$q_1$	1	1	$q_1, \epsilon$
	$q_1$	$\epsilon$	$Z_0$	$q_2, Z_0$
*	$q_2$			



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

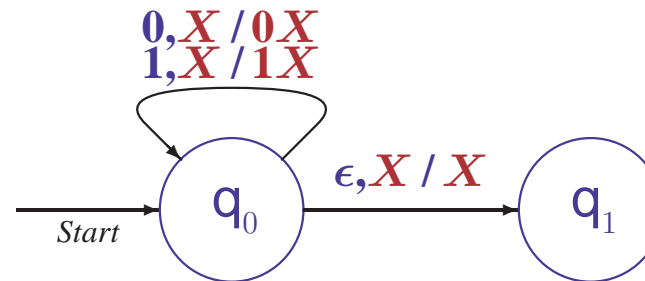


# PUSHDOWN-AUTOMAT FÜR $\{ww^R \mid w \in \{0,1\}^*\}$



- **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  $a \in \{0,1\}$ ,  $X \in \Gamma$

# 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

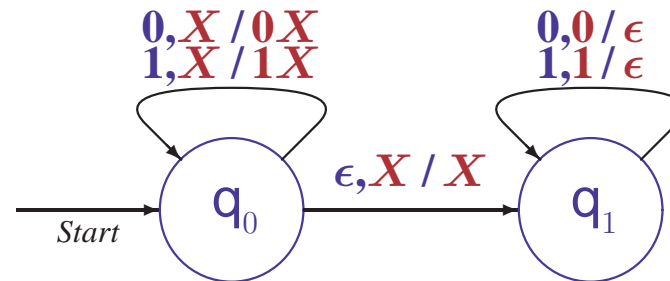
- $\delta(q_0, a, X) = \{(q_0, aX)\}$  für  $a \in \{0, 1\}$ ,  $X \in \Gamma$

- **Spontaner Wechsel “in der Mitte”**

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

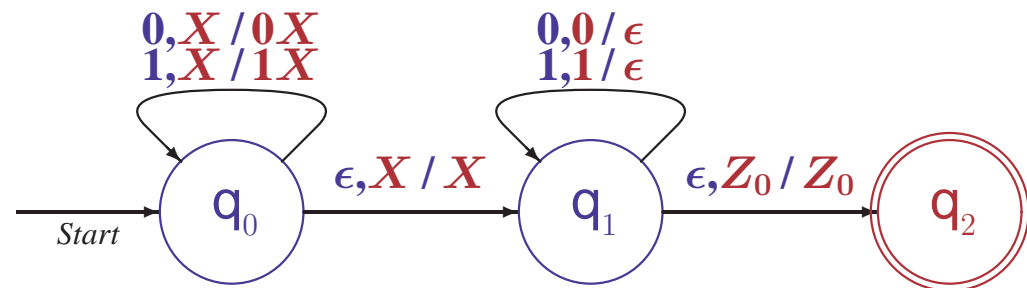
(nichtdeterministischer  $\epsilon$ -Übergang)

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



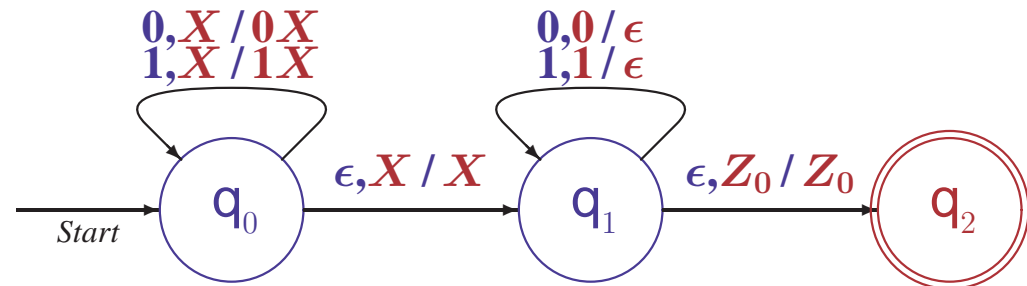
- **Speichere  $w$  in  $q_0$** 
  - Es wird je ein Symbol gelesen und auf den Stack gelegt
    - $\delta(q_0, a, X) = \{(q_0, aX)\}$  für  $a \in \{0,1\}, X \in \Gamma$
- **Spontaner Wechsel “in der Mitte”**
  - $\delta(q_0, \epsilon, X) = \{(q_1, X)\}$  für  $X \in \Gamma$  (nichtdeterministischer  $\epsilon$ -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  $a \in \{0,1\}$

# 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  $\epsilon$ -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  $\epsilon$ -Übergang)

# 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  $\epsilon$ -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  $\epsilon$ -Übergang)

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

## Generalisiere Konzept der Konfigurationsübergänge

- **Erweitere Begriff der Konfiguration**

- Aktueller Zustand, Inhalt des Stacks und unverarbeitete Eingabe zählt
- Formal dargestellt als Tripel  $K = (q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*$

## Generalisiere Konzept der Konfigurationsübergänge

- **Erweitere Begriff der Konfiguration**

- Aktueller Zustand, Inhalt des Stacks und unverarbeitete Eingabe zählt
- Formal dargestellt als Tripel  $K = (q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*$

- **Modifiziere Konfigurationsübergangsrelation  $\vdash^*$**

- Wechsel zwischen Konfigurationen durch Abarbeitung von Wörtern

- $(q, aw, X\beta) \vdash (p, w, \alpha\beta)$ , falls  $(p, \alpha) \in \delta(q, a, X)$

- $(q, w, X\beta) \vdash (p, w, \alpha\beta)$ , 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  $\alpha$  ersetzt, der Rest bleibt stehen)

- $K_1 \vdash^* K_2$ , falls  $K_1 = K_2$  oder

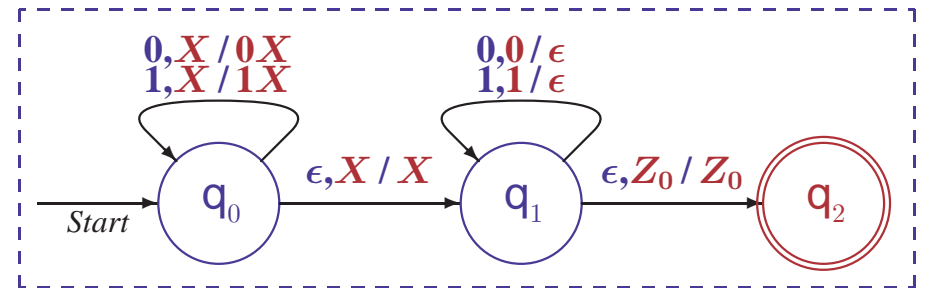
es gibt eine Konfiguration  $K$  mit  $K_1 \vdash K$  und  $K \vdash^* K_2$



# ABARBEITUNG DES PALINDROM PDA

## Verarbeitung von 1111

$(q_0, 1111, Z_0)$



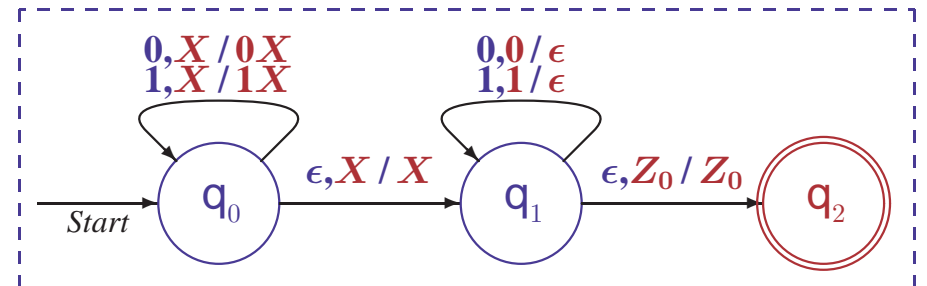
# ABARBEITUNG DES PALINDROM PDA

## Verarbeitung von 1111

$(q_0 \mathbf{1111}, Z_0)$



$(q_0 \mathbf{111}, \mathbf{1}Z_0)$



# ABARBEITUNG DES PALINDROM PDA

## Verarbeitung von 1111

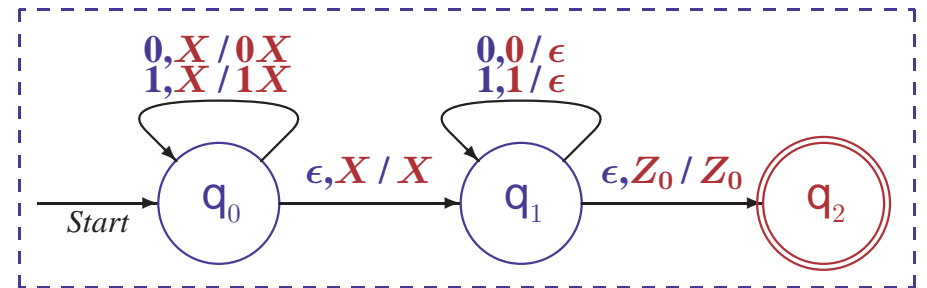
$(q_0 \mathbf{1111}, Z_0)$



$(q_0 \mathbf{111}, \mathbf{1}Z_0)$

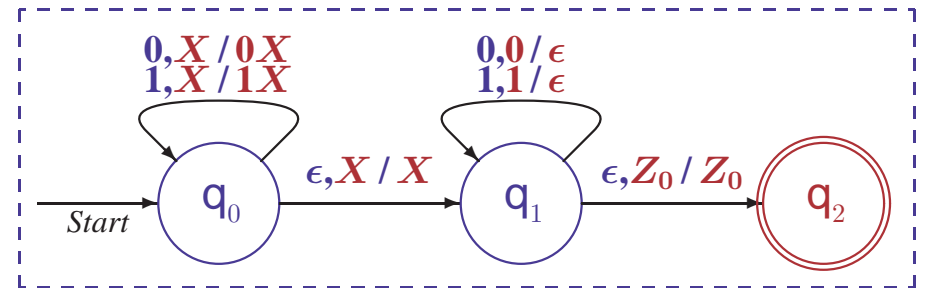
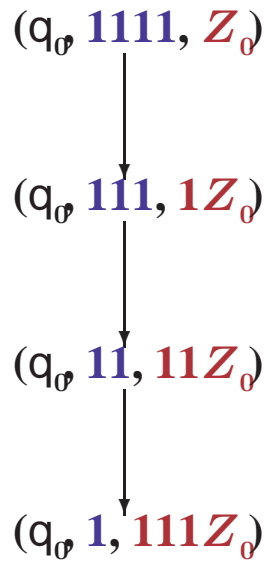


$(q_0 \mathbf{11}, \mathbf{11}Z_0)$



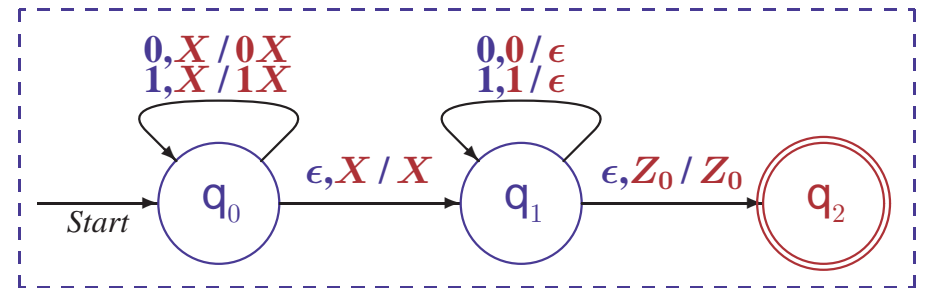
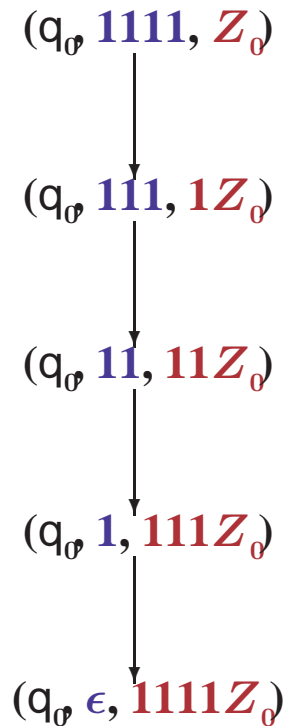
# ABARBEITUNG DES PALINDROM PDA

## Verarbeitung von 1111



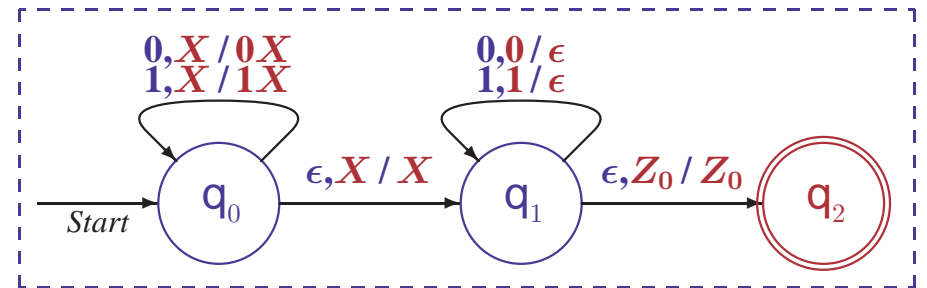
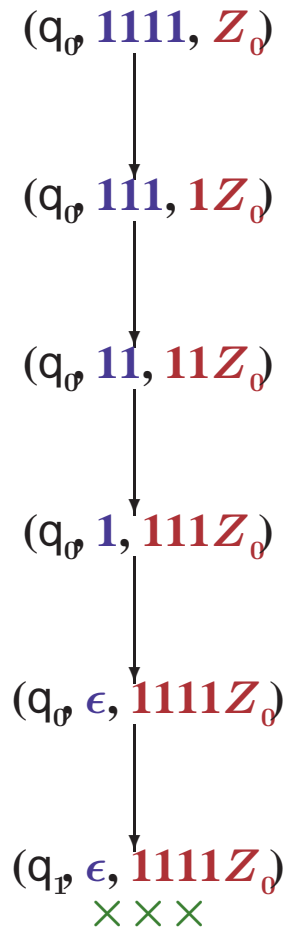
# ABARBEITUNG DES PALINDROM PDA

## Verarbeitung von 1111



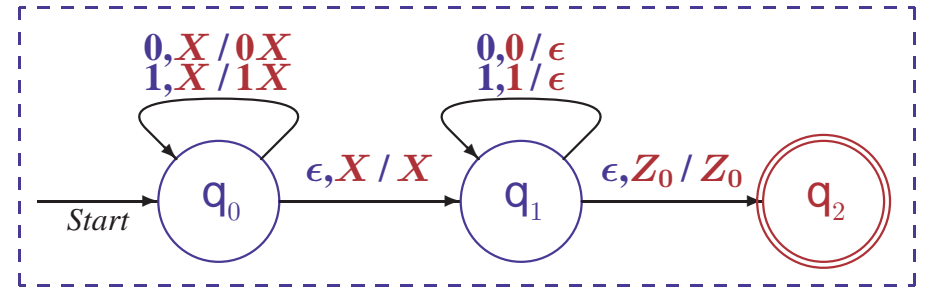
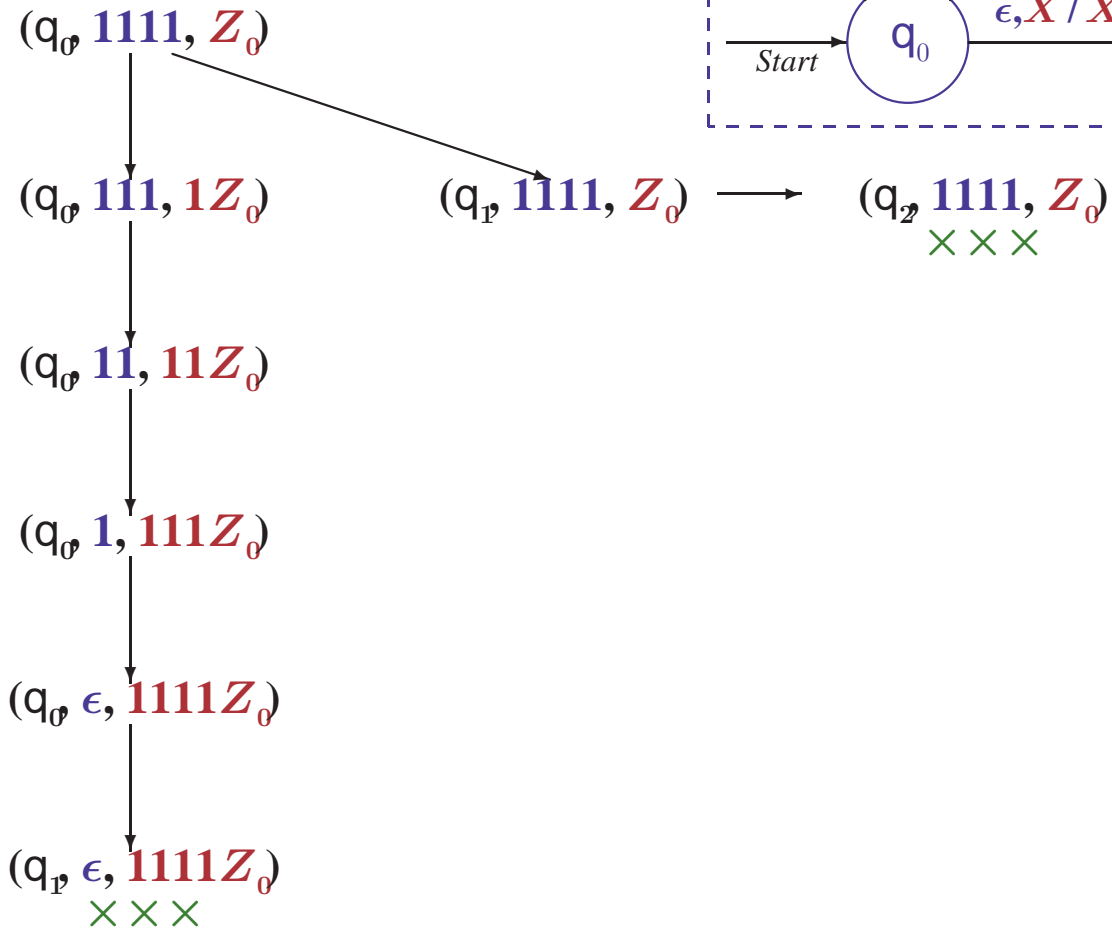
# ABARBEITUNG DES PALINDROM PDA

## Verarbeitung von 1111



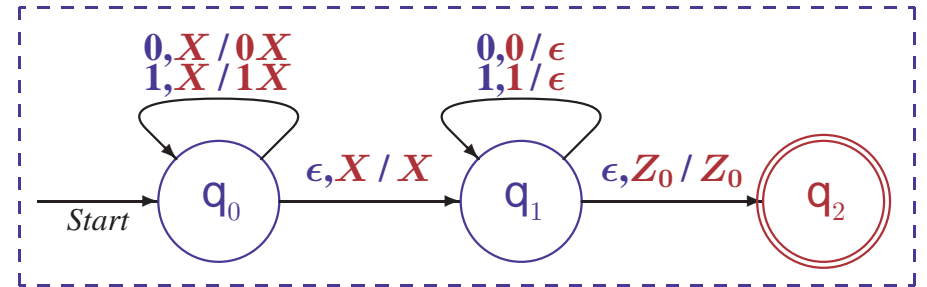
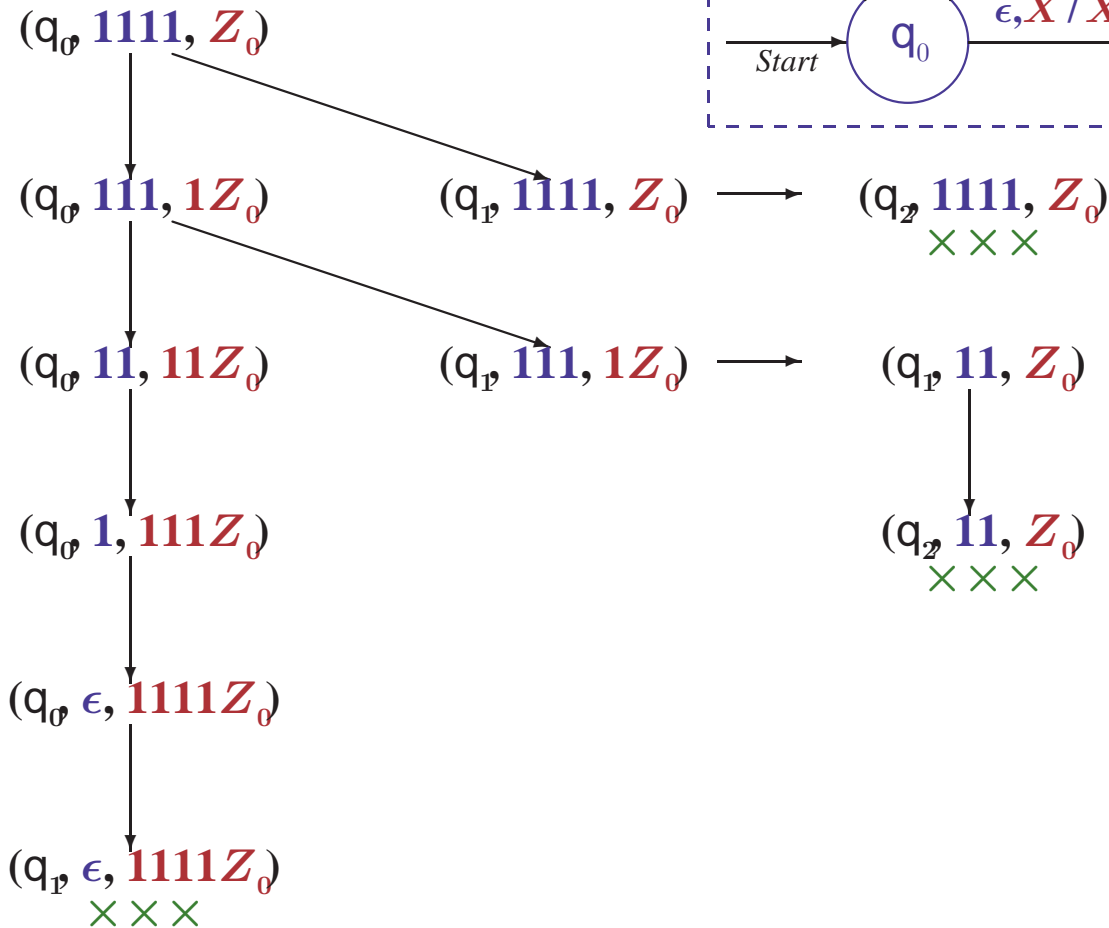
# ABARBEITUNG DES PALINDROM PDA

## Verarbeitung von 1111



# ABARBEITUNG DES PALINDROM PDA

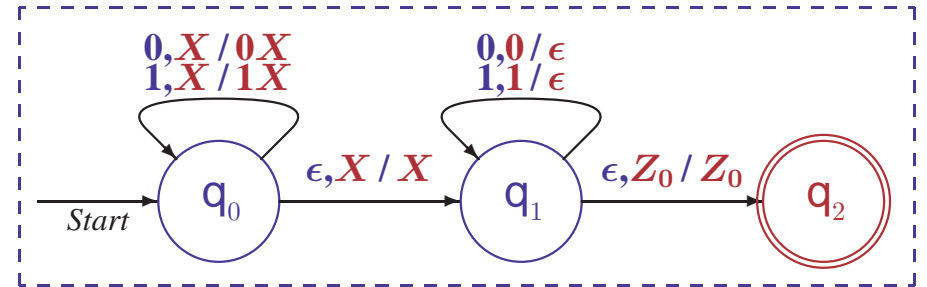
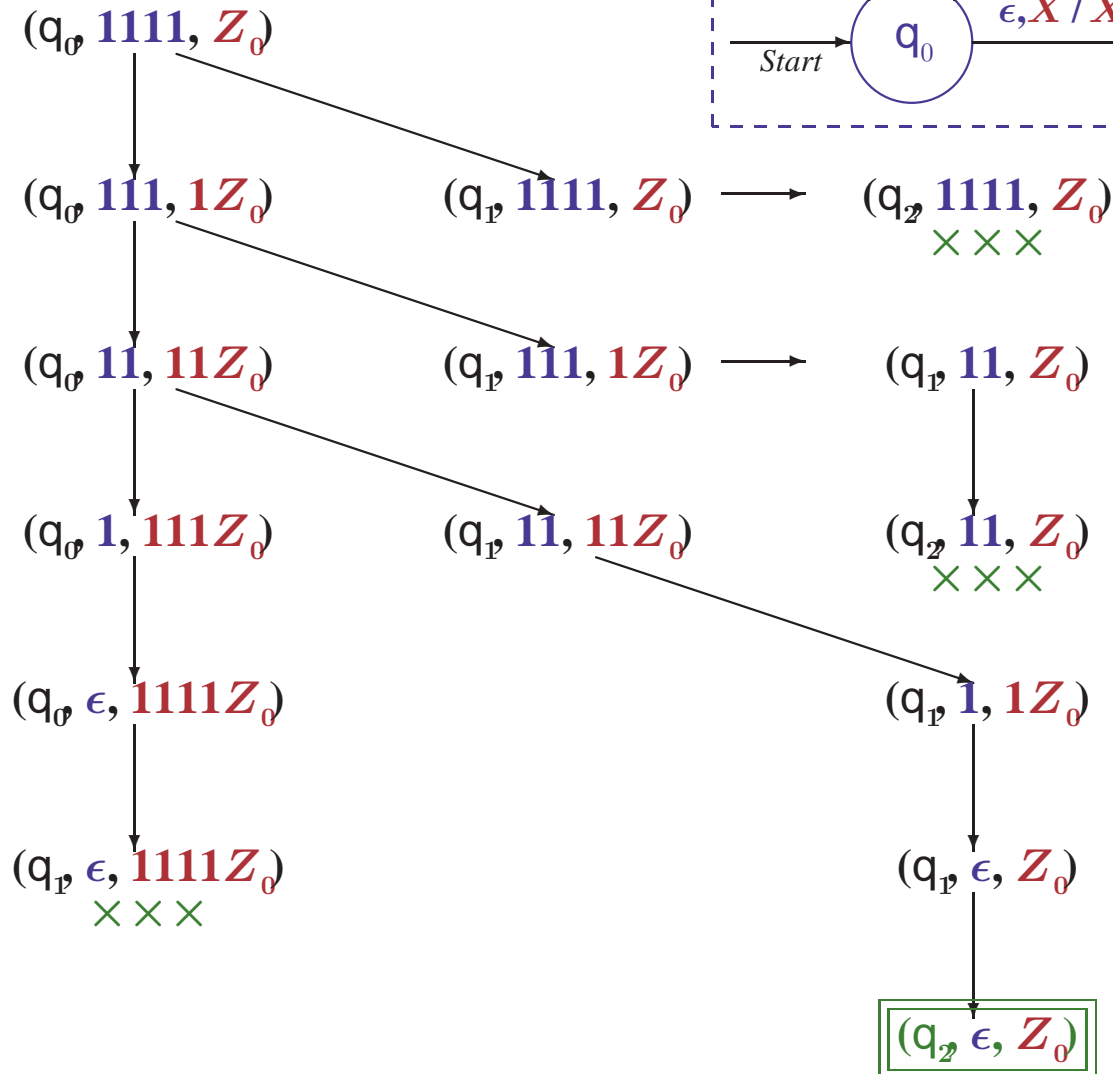
## Verarbeitung von 1111





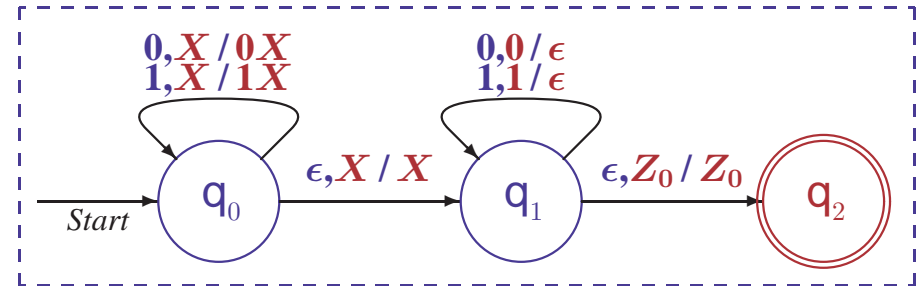
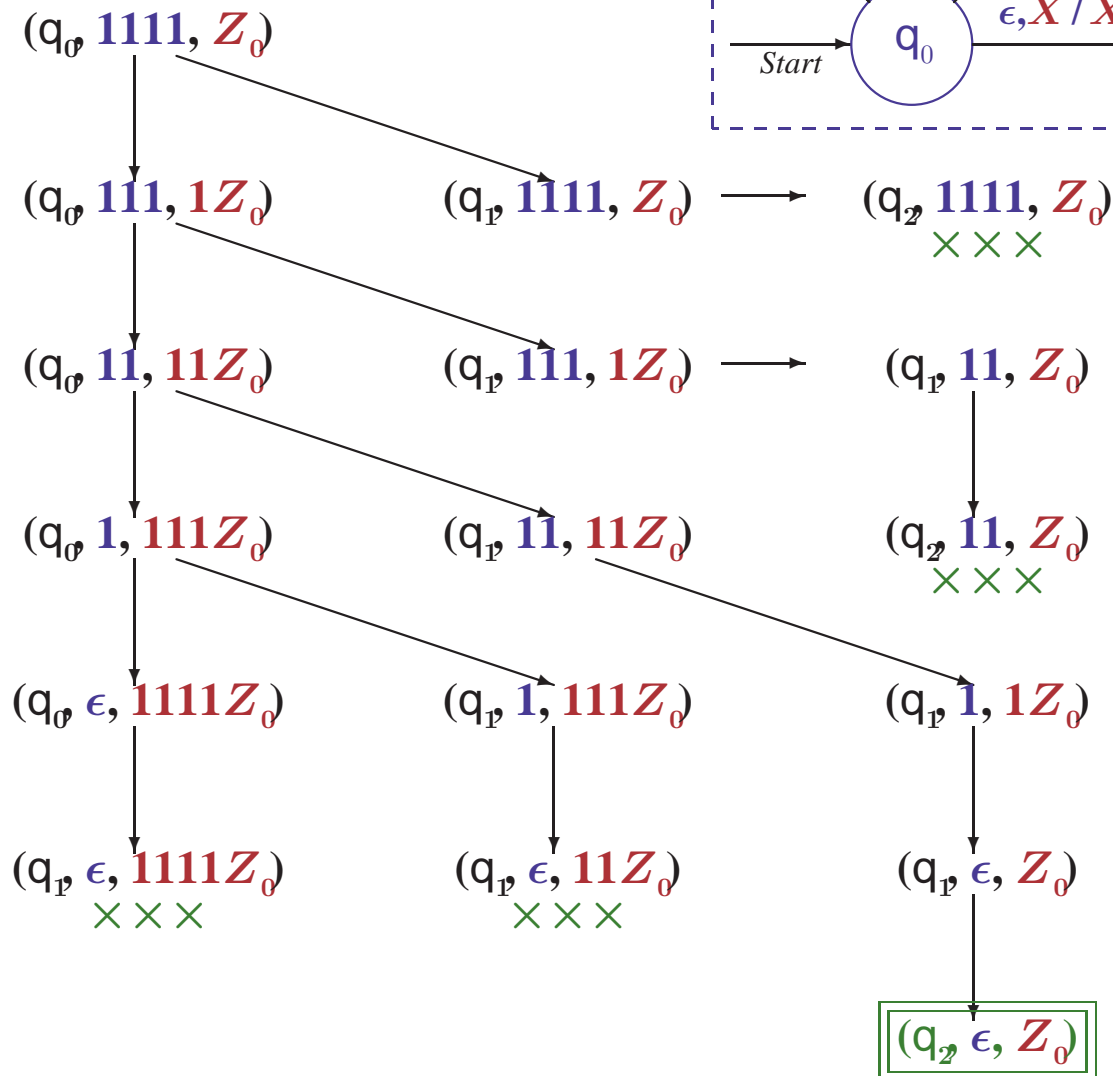
# ABARBEITUNG DES PALINDROM PDA

## Verarbeitung von 1111



# 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

## 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

## 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

## 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**

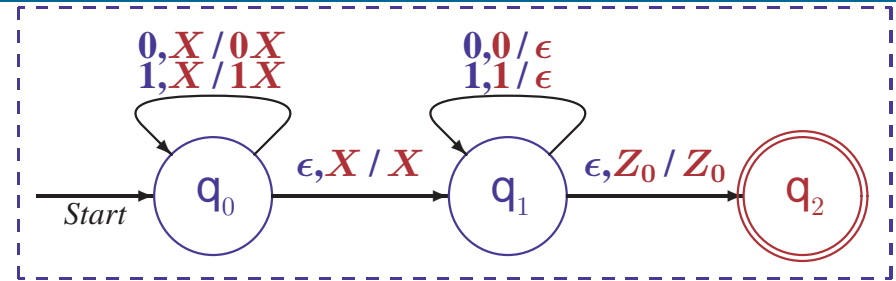
- Sprachen können für konkrete PDAs sehr verschieden ausfallen

- **Beide Definitionen sind gleichmächtig**

- PDA kann passend zur anderen Definition umgewandelt werden

# DIE BEIDEN SPRACHEN DES PALINDROMAUTOMATEN

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

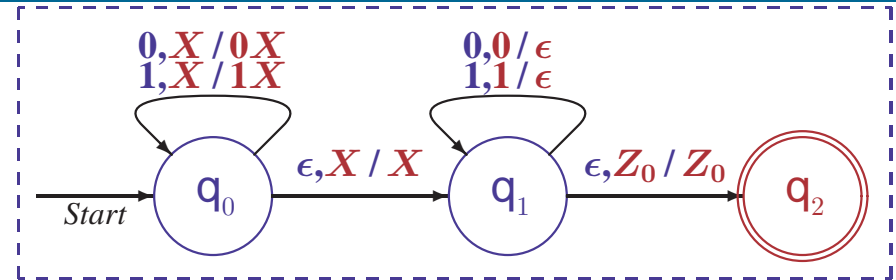


# DIE BEIDEN SPRACHEN DES PALINDROMAUTOMATEN

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

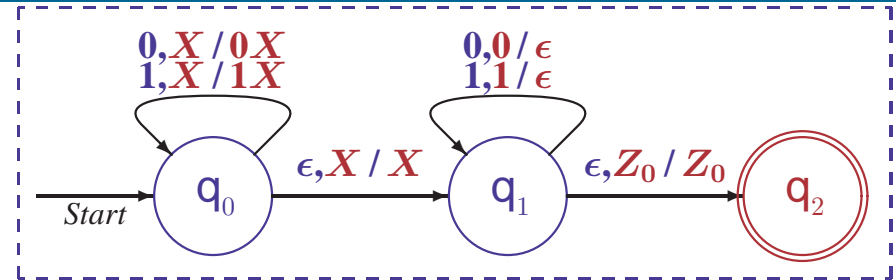
$\supseteq$ : Durch strukturelle Induktion zeige, daß für jedes Wort  $w$  gilt

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





# 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

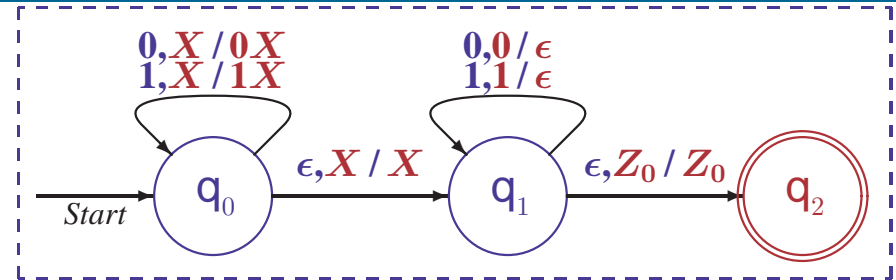
(Details in HMU §6.2.1)

$$\begin{aligned} \text{Wenn } (q_0, x_1..x_n, \alpha') \vdash^* (q_0, x_2..x_n, x_1\alpha') \vdash^* (q_1, x_i..x_n, \beta x_1\alpha') \\ \vdash^* (q_1, x_n, x_1\alpha') \quad \vdash^* (q_1, \epsilon, \alpha') \text{ für } \alpha', \beta \in \Gamma^*, \end{aligned}$$

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

und  $x_1 = x_n$  und per Induktion  $x_2..x_{n-1} = vv^R$  für ein  $v \in \{0, 1\}^*$

# 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)

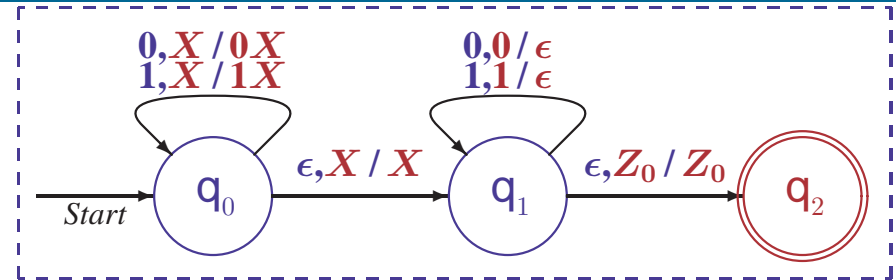
$$\begin{aligned} \text{Wenn } (q_0, x_1..x_n, \alpha') \vdash^* (q_0, x_2..x_n, x_1\alpha') \vdash^* (q_1, x_i..x_n, \beta x_1\alpha') \\ \vdash^* (q_1, x_n, x_1\alpha') \quad \vdash^* (q_1, \epsilon, \alpha') \text{ für } \alpha', \beta \in \Gamma^*, \end{aligned}$$

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

und  $x_1 = x_n$  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

# 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)

$$\begin{aligned} \text{Wenn } (q_0, x_1..x_n, \alpha') \vdash^* (q_0, x_2..x_n, x_1\alpha') \vdash^* (q_1, x_i..x_n, \beta x_1\alpha') \\ \vdash^* (q_1, x_n, x_1\alpha') \quad \vdash^* (q_1, \epsilon, \alpha') \text{ für } \alpha', \beta \in \Gamma^*, \end{aligned}$$

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

und  $x_1 = x_n$  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**  $\alpha$  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  $(q, ay, X\beta) \vdash (p, y, \gamma\beta)$

## 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  $\alpha$  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  $\gamma$  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

## Konstruiere PDA für korrekte Klammersausdrücke

- **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

# 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**

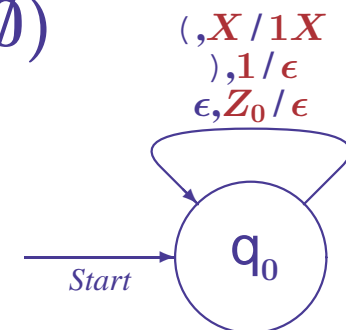
- Jedes ( erhöht die Anzahl, jedes ) erniedrigt sie
- ) ist nicht erlaubt, wenn der Stackboden erreicht ist
- Am Ende des Wortes wird der Stackboden entfernt

- **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 $L_{\epsilon}$ - 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  $L_{\epsilon}(P_{\epsilon}) = L_F(P_F)$**

## TRANSFORMATION VON $L_\epsilon$ - 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  $P_\epsilon$  auf Stack
  - Neuer Endzustand  $p_f$ , in den bei “leerem” Stack gewechselt wird

# TRANSFORMATION VON $L_\epsilon$ - IN $L_F$ -AUTOMATEN

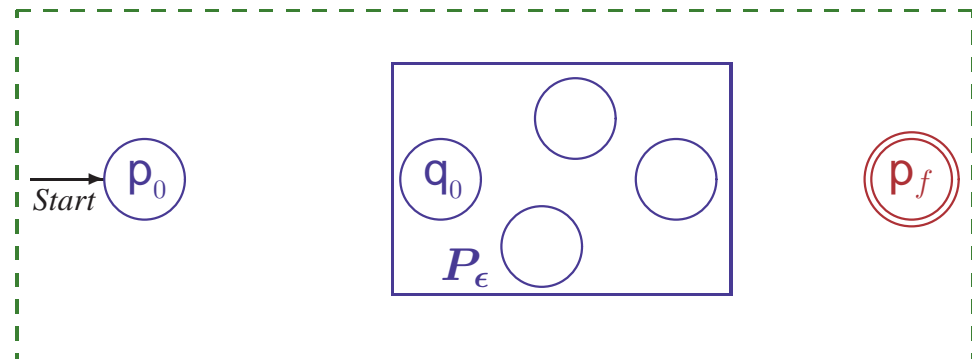
Zu jedem PDA  $P_\epsilon = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset)$  kann ein PDA  $P_F$  konstruiert werden mit  $L_\epsilon(P_\epsilon) = L_F(P_F)$

- **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  $P_\epsilon$  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  $q \in Q, X \in \Gamma$



# TRANSFORMATION VON $L_\epsilon$ - IN $L_F$ -AUTOMATEN

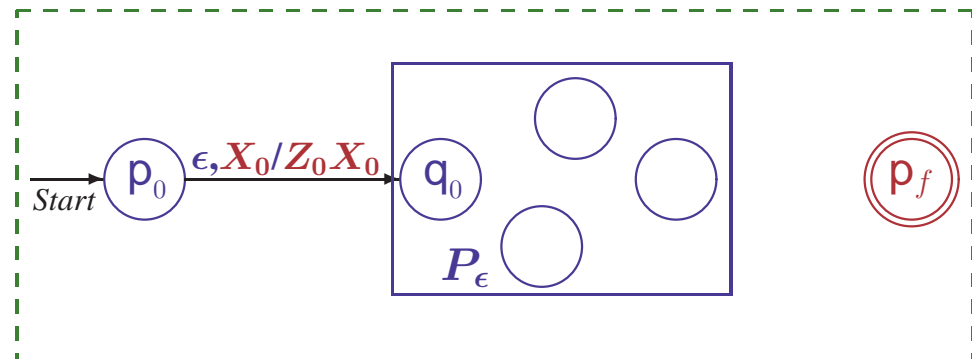
Zu jedem PDA  $P_\epsilon = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset)$  kann ein PDA  $P_F$  konstruiert werden mit  $L_\epsilon(P_\epsilon) = L_F(P_F)$

- **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  $P_\epsilon$  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  $q \in Q, X \in \Gamma$
- $\delta_F(p_0, \epsilon, X_0) = \{(q_0, Z_0 X_0)\}$



# TRANSFORMATION VON $L_{\epsilon}$ - IN $L_F$ -AUTOMATEN

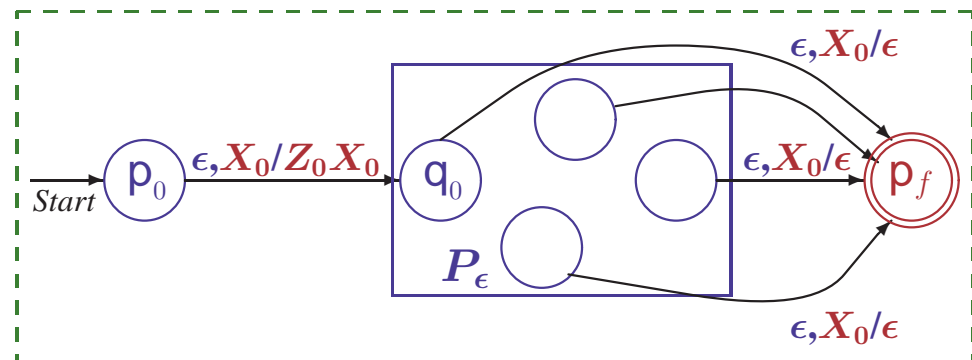
Zu jedem PDA  $P_{\epsilon} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset)$  kann ein PDA  $P_F$  konstruiert werden mit  $L_{\epsilon}(P_{\epsilon}) = L_F(P_F)$

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



Korrektheitsbeweis durch Detailanalyse

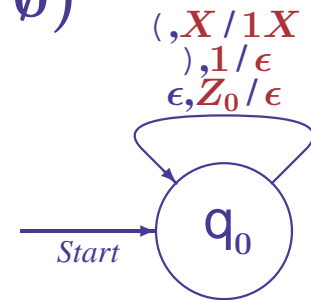
# UMWANDLUNG EINES $L_\epsilon$ -PDA IN EINEN $L_F$ -PDA

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

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

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

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



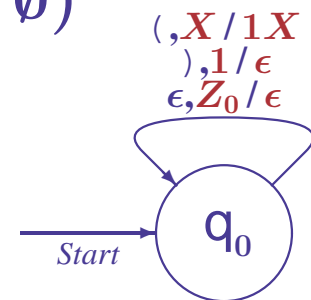
# UMWANDLUNG EINES $L_\epsilon$ -PDA IN EINEN $L_F$ -PDA

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

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

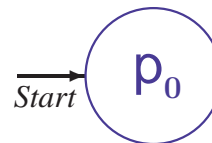
$\delta(q, \cdot, 1) = \{(q, \epsilon)\}$

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



- Äquivalenter PDA  $P_F$  mit Endzuständen ist

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



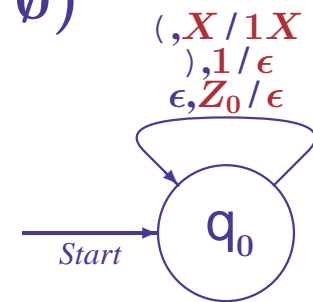
# UMWANDLUNG EINES $L_\epsilon$ -PDA IN EINEN $L_F$ -PDA

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

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

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

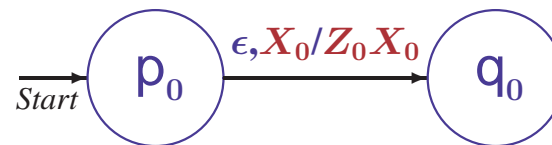
$\delta(q, (\epsilon, Z_0) = \{(q, \epsilon)\}$



- Äquivalenter PDA  $P_F$  mit Endzuständen ist

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

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





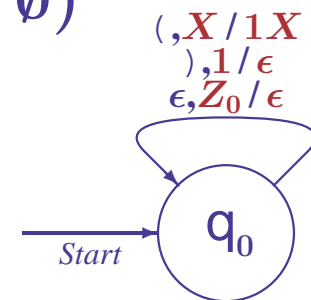
# UMWANDLUNG EINES $L_\epsilon$ -PDA IN EINEN $L_F$ -PDA

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

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

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

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



- Äquivalenter PDA  $P_F$  mit Endzuständen ist

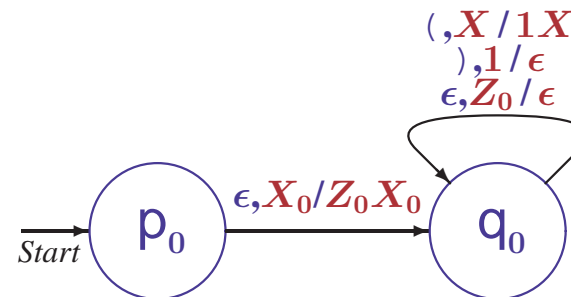
$$(\{p_0, q, p_f\}, \{(\cdot)\}, \{X_0, Z_0, 1\}, \delta_F, p_0, X_0, \{p_f\})$$

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

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

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

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



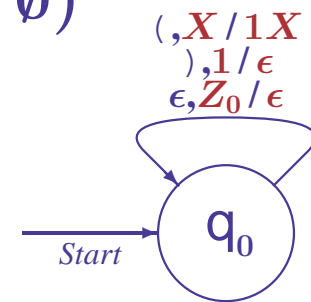
# UMWANDLUNG EINES $L_\epsilon$ -PDA IN EINEN $L_F$ -PDA

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

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

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

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



- Äquivalenter PDA  $P_F$  mit Endzuständen ist

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

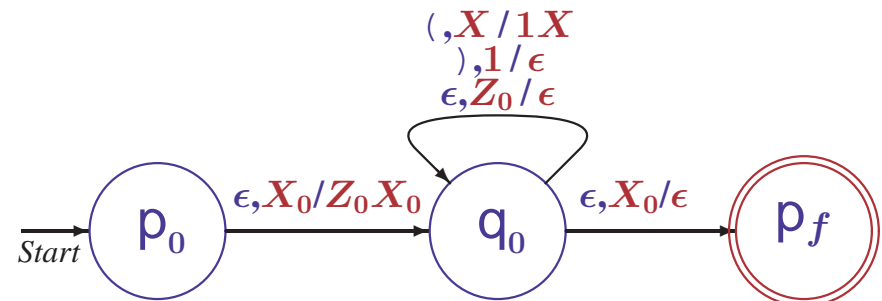
mit  $\delta_F(p_0, (\epsilon, X_0) = \{(q, Z_0X_0)\}$

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

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

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

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



## TRANSFORMATION VON $L_F$ - IN $L_\epsilon$ -AUTOMATEN

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

## TRANSFORMATION VON $L_F$ - IN $L_\epsilon$ -AUTOMATEN

Zu jedem PDA  $P_F = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  kann ein PDA  $P_\epsilon$  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  $P_\epsilon$  verhindert irrtümliches Leeren des Stacks
- Neuer Anfangszustand  $p_0$  für  $P_\epsilon$  schreibt Initialsymbol von  $P_F$  auf Stack

# TRANSFORMATION VON $L_F$ - IN $L_\epsilon$ -AUTOMATEN

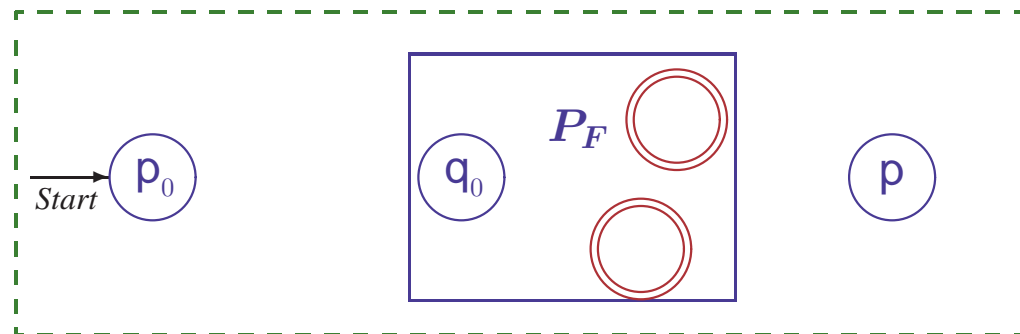
Zu jedem PDA  $P_F = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  kann ein PDA  $P_\epsilon$  konstruiert werden mit  $L_F(P_F) = L_\epsilon(P_\epsilon)$

- **Im Endzustand leere den Stack**

- Neuer Stacklösch-Zustand  $p$ , in den von Endzuständen gewechselt wird
- Neues Initialsymbol  $X_0$  für  $P_\epsilon$  verhindert irrtümliches Leeren des Stacks
- Neuer Anfangszustand  $p_0$  für  $P_\epsilon$  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  $q \in Q, X \in \Gamma$



# TRANSFORMATION VON $L_F$ - IN $L_\epsilon$ -AUTOMATEN

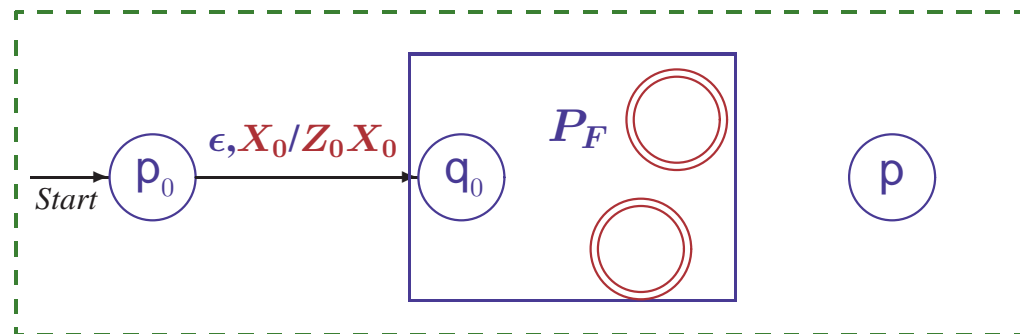
Zu jedem PDA  $P_F = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  kann ein PDA  $P_\epsilon$  konstruiert werden mit  $L_F(P_F) = L_\epsilon(P_\epsilon)$

- **Im Endzustand leere den Stack**

- Neuer Stacklösch-Zustand  $p$ , in den von Endzuständen gewechselt wird
- Neues Initialsymbol  $X_0$  für  $P_\epsilon$  verhindert irrtümliches Leeren des Stacks
- Neuer Anfangszustand  $p_0$  für  $P_\epsilon$  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  $q \in Q, X \in \Gamma$
- $\delta_\epsilon(p_0, \epsilon, X_0) = \{(q_0, Z_0 X_0)\}$



# TRANSFORMATION VON $L_F$ - IN $L_\epsilon$ -AUTOMATEN

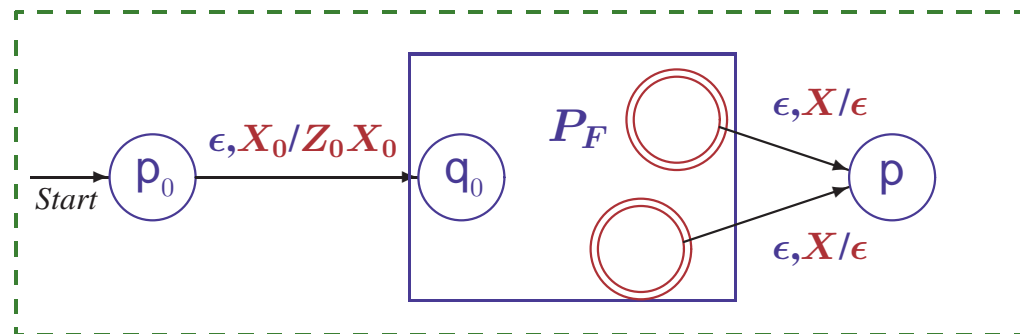
Zu jedem PDA  $P_F = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  kann ein PDA  $P_\epsilon$  konstruiert werden mit  $L_F(P_F) = L_\epsilon(P_\epsilon)$

- **Im Endzustand leere den Stack**

- Neuer Stacklösch-Zustand  $p$ , in den von Endzuständen gewechselt wird
- Neues Initialsymbol  $X_0$  für  $P_\epsilon$  verhindert irrtümliches Leeren des Stacks
- Neuer Anfangszustand  $p_0$  für  $P_\epsilon$  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  $q \in Q, X \in \Gamma$
- $\delta_\epsilon(p_0, \epsilon, X_0) = \{(q_0, Z_0 X_0)\}$
- $\delta_\epsilon(q, \epsilon, X) = \{(p, \epsilon)\}$   
für alle  $q \in F$



# TRANSFORMATION VON $L_F$ - IN $L_\epsilon$ -AUTOMATEN

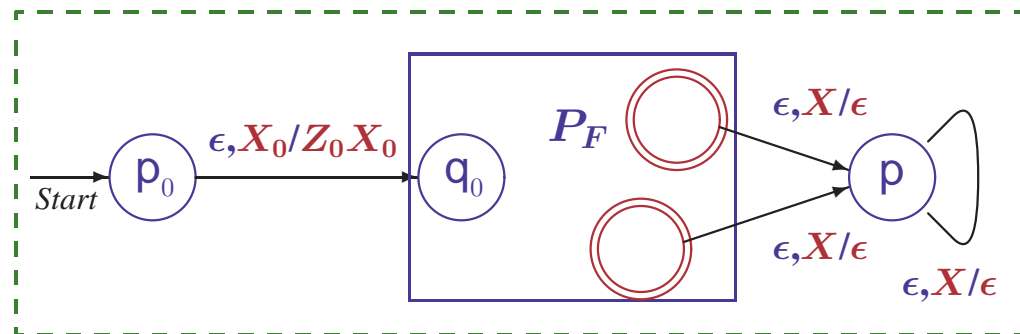
Zu jedem PDA  $P_F = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  kann ein PDA  $P_\epsilon$  konstruiert werden mit  $L_F(P_F) = L_\epsilon(P_\epsilon)$

- **Im Endzustand leere den Stack**

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

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

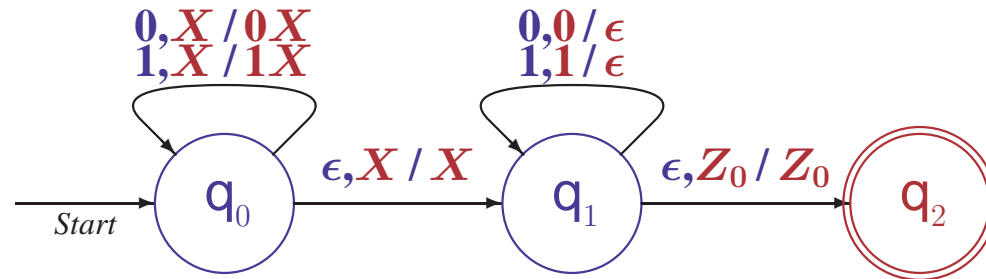
- $\delta_\epsilon(q, a, X) = \delta(q, a, X)$   
für alle  $q \in Q, X \in \Gamma$
- $\delta_\epsilon(p_0, \epsilon, X_0) = \{(q_0, Z_0 X_0)\}$
- $\delta_\epsilon(q, \epsilon, X) = \{(p, \epsilon)\}$   
für alle  $q \in F$
- $\delta_\epsilon(p, \epsilon, X) = \{(p, \epsilon)\}$   
für alle  $X \in \Gamma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



Korrektheitsbeweis durch Detailanalyse

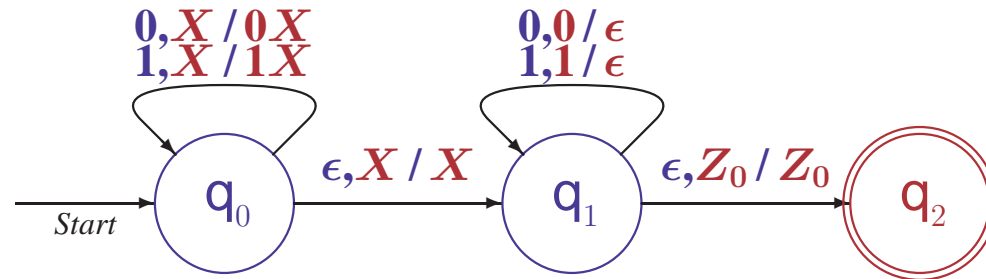


# UMWANDLUNG EINES $L_F$ -PDA IN EINEN $L_\epsilon$ -PDA



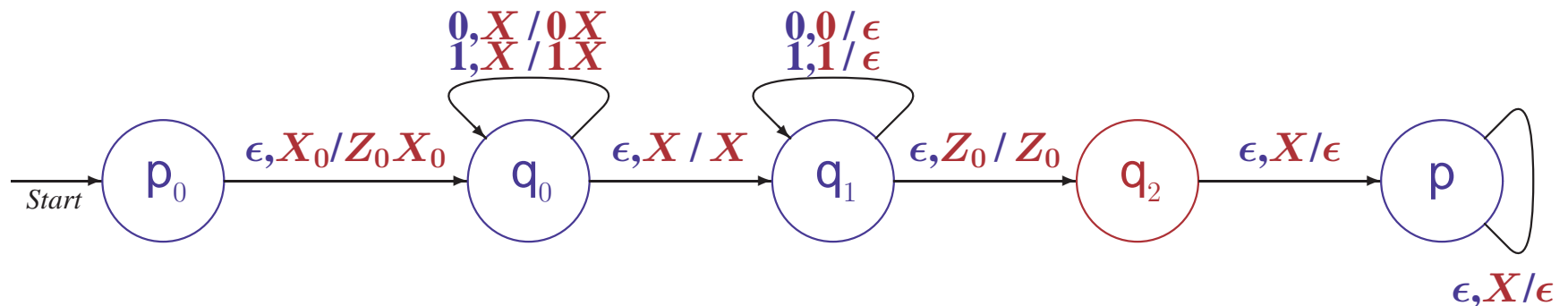
- $P_F = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, Z_0\}, \delta, q_0, Z_0, \{q_2\})$   
mit  $\delta$  wie oben erkennt  $\{ww^R \mid w \in \{0, 1\}^*\}$  mit Endzustand

# UMWANDLUNG EINES $L_F$ -PDA IN EINEN $L_\epsilon$ -PDA



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

- Äquivalenter PDA  $P_\epsilon$  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?

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

## 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

# 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

# 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  $\delta$  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  $L(G)=L_\epsilon(P)$**

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



**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  $\beta$  einer Regel  $A \rightarrow \beta$  ersetzt
  - $a \in T$  wird vom Stack entfernt, wenn es als Eingabe erscheint, um im Stack die nächsten Variable einer Linksableitung freizulegen
- **Generierter PDA  $P = (\{q\}, T, V \cup T, \delta, q, S, \emptyset)$** 
  - $\delta(q, \epsilon, A) = \{(q, \beta) \mid A \rightarrow \beta \in P_G\}$  für alle  $A \in V$
  - $\delta(q, a, a) = \{(q, \epsilon)\}$  für alle  $a \in T$

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

- **Generierter PDA  $P = (\{q\}, T, V \cup T, \delta, q, S, \emptyset)$**

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

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

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

( $\supseteq$ ) 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  $P_G = \{ E \rightarrow I \mid E+E \mid E*E \mid (E)$   
 $I \rightarrow a \mid b \mid c \mid Ia \mid Ib \mid Ic \mid I0 \mid I1 \}$

## 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  $T = \{a, b, 0, 1, +, *, (, )\}$

## 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  $T = \{a, b, 0, 1, +, *, (, )\}$

–  $\delta(q, \epsilon, E) = \{(q, I), (q, E+E), (q, E*E), (q, (E))\}$

–  $\delta(q, \epsilon, I) = \{(q, a), (q, b), (q, c), (q, Ia), (q, Ib), (q, Ic), (q, I0), (q, I1)\}$

## 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  $T = \{a, b, 0, 1, +, *, (, )\}$

–  $\delta(q, \epsilon, E) = \{(q, I), (q, E+E), (q, E*E), (q, (E))\}$

–  $\delta(q, \epsilon, I) = \{(q, a), (q, b), (q, c), (q, Ia), (q, Ib), (q, Ic), (q, I0), (q, I1)\}$

–  $\delta(q, a, a) = \{(q, \epsilon)\}$       –  $\delta(q, +, +) = \{(q, \epsilon)\}$

–  $\delta(q, b, b) = \{(q, \epsilon)\}$       –  $\delta(q, *, *) = \{(q, \epsilon)\}$

–  $\delta(q, c, c) = \{(q, \epsilon)\}$       –  $\delta(q, (, ( ) = \{(q, \epsilon)\}$

–  $\delta(q, 0, 0) = \{(q, \epsilon)\}$       –  $\delta(q, ), ) = \{(q, \epsilon)\}$

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

## KORREKTHEITSBEWEIF IM DETAIL: $L(G) \subseteq L_\epsilon(P)$

Wenn  $S = x_1 A_1 \alpha_1 \dots \xrightarrow{L} x_m A_m \alpha_m \xrightarrow{L} w \in T^*$  ( $x_i \in T^*, A_i \in V$ ) 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  $i \leq m$

## KORREKTHEITSBEWWEIS IM DETAIL: $L(G) \subseteq L_\epsilon(P)$

Wenn  $S = x_1 A_1 \alpha_1 \dots \xrightarrow{L} x_m A_m \alpha_m \xrightarrow{L} w \in T^*$  ( $x_i \in T^*, A_i \in V$ ) 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



## KORREKTHEITSBEWEIF IM DETAIL: $L(G) \subseteq L_\epsilon(P)$

**Wenn  $S = x_1 A_1 \alpha_1 \dots \xrightarrow{L} x_m A_m \alpha_m \xrightarrow{L} w \in T^*$  ( $x_i \in T^*, A_i \in V$ ) 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:  $S \dots \xrightarrow{L} x_i A_i \alpha_i \xrightarrow{L} x_{i+1} A_{i+1} \alpha_{i+1} \xrightarrow{*} w$** 
  - $x_i A_i \alpha_i \xrightarrow{L} x_{i+1} A_{i+1} \alpha_{i+1}$  verlangt  $A_i \rightarrow \beta_i \in P_G$  für ein  $\beta_i$ , 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:  $(q, y_i, z A_{i+1} \alpha_{i+1}) \vdash^* (q, y_{i+1}, A_{i+1} \alpha_{i+1})$

## KORREKTHEITSBEWEIF IM DETAIL: $L(G) \subseteq L_\epsilon(P)$

**Wenn  $S = x_1 A_1 \alpha_1 \dots \xrightarrow{L} x_m A_m \alpha_m \xrightarrow{L} w \in T^*$  ( $x_i \in T^*, A_i \in V$ ) 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:  $S \dots \xrightarrow{L} x_i A_i \alpha_i \xrightarrow{L} x_{i+1} A_{i+1} \alpha_{i+1} \xrightarrow{*} w$** 
  - $x_i A_i \alpha_i \xrightarrow{L} x_{i+1} A_{i+1} \alpha_{i+1}$  verlangt  $A_i \rightarrow \beta_i \in P_G$  für ein  $\beta_i$ , 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:  $S = x_1 A_1 \alpha_1 \dots \xrightarrow{L} x_m A_m \alpha_m \xrightarrow{L} w \in T^*$** 
  - 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

## 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  $X \rightarrow \epsilon \in P_G$  und  $w = \epsilon$ , also  $X \xrightarrow{*} w$

# KORREKTHEITSBEWEIF 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  $X \rightarrow \epsilon \in P_G$  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  $X \rightarrow Y_1..Y_k \in P_G$  haben ( $Y_i \in V \cup T$ )
  - Dann gibt eine Zerlegung  $w = w_1..w_k$  mit  $(q, w_1w_2..w_k, Y_1Y_2..Y_k) \vdash^* (q, w_2..w_k, Y_2..Y_k) \vdash^* (q, \epsilon, \epsilon)$
  - Es folgt  $(q, w_iw_{i+1}..w_k, Y_i) \vdash^* (q, w_{i+1}..w_k, \epsilon)$  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  $X \rightarrow Y_1..Y_k \xrightarrow{*} w_1..w_k = w$

# KORREKTHEITSBEWEIF 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  $X \rightarrow \epsilon \in P_G$  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  $X \rightarrow Y_1..Y_k \in P_G$  haben ( $Y_i \in V \cup T$ )
  - Dann gibt eine Zerlegung  $w = w_1..w_k$  mit  $(q, w_1w_2..w_k, Y_1Y_2..Y_k) \vdash^* (q, w_2..w_k, Y_2..Y_k) \vdash^* (q, \epsilon, \epsilon)$
  - Es folgt  $(q, w_iw_{i+1}..w_k, Y_i) \vdash^* (q, w_{i+1}..w_k, \epsilon)$  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  $X \rightarrow Y_1..Y_k \xrightarrow{*} w_1..w_k = w$
- **Es folgt  $L_\epsilon(P) = \{w \mid (q, w, S) \vdash^* (q, \epsilon, \epsilon)\} \subseteq \{w \mid S \xrightarrow{*} w\} = L(G)$**

## 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  $L_\epsilon(P) = L(G)$**

## 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



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

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

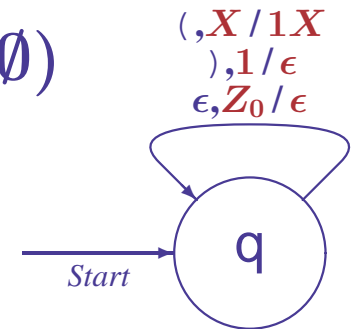
# UMWANDLUNG EINES PDA IN EINE GRAMMATIK

- Gegeben  $P_1 = (\{q\}, \{(\,)\}, \{Z_0, 1\}, \delta, q, Z_0, \emptyset)$

mit  $\delta(q, (\, X) = \{(q, 1X)\}$  für  $X \in \{Z_0, 1\}$

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

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



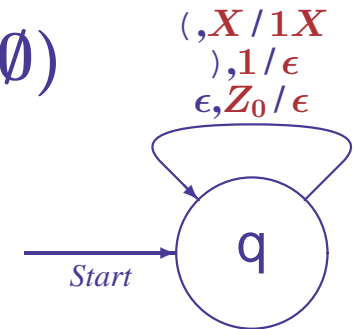
# UMWANDLUNG EINES PDA IN EINE GRAMMATIK

- Gegeben  $P_1 = (\{q\}, \{(\,)\}, \{Z_0, 1\}, \delta, q, Z_0, \emptyset)$

mit  $\delta(q, (\, X) = \{(q, 1X)\}$  für  $X \in \{Z_0, 1\}$

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

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



- Generiere  $G = (\{(\,)\}, \{S, (q, Z_0, q), (q, 1, q)\}, P_G, S)$

mit  $P_G = S \rightarrow (q, Z_0, q)$

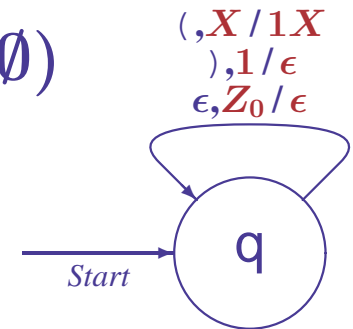
# UMWANDLUNG EINES PDA IN EINE GRAMMATIK

- Gegeben  $P_1 = (\{q\}, \{(\,)\}, \{Z_0, 1\}, \delta, q, Z_0, \emptyset)$

mit  $\delta(q, (\, X) = \{(q, 1X)\}$  für  $X \in \{Z_0, 1\}$

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

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



- Generiere  $G = (\{(\,)\}, \{S, (q, Z_0, q), (q, 1, q)\}, P_G, S)$

mit  $P_G = S \rightarrow (q, Z_0, q)$

$$(q, Z_0, q) \rightarrow ((q, 1, q)(q, Z_0, q)$$

$$(q, Z_0, q) \rightarrow \epsilon$$

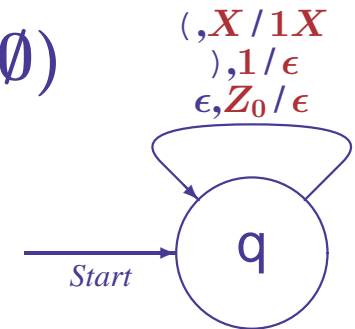
# UMWANDLUNG EINES PDA IN EINE GRAMMATIK

- Gegeben  $P_1 = (\{q\}, \{(\,)\}, \{Z_0, 1\}, \delta, q, Z_0, \emptyset)$

mit  $\delta(q, (\, X) = \{(q, 1X)\}$  für  $X \in \{Z_0, 1\}$

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

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



- Generiere  $G = (\{(\,)\}, \{S, (q, Z_0, q), (q, 1, q)\}, P_G, S)$

mit  $P_G = S \rightarrow (q, Z_0, q)$

$$(q, Z_0, q) \rightarrow ((q, 1, q)(q, Z_0, q)$$

$$(q, Z_0, q) \rightarrow \epsilon$$

$$(q, 1, q) \rightarrow ((q, 1, q)(q, 1, q)$$

$$(q, 1, q) \rightarrow )$$

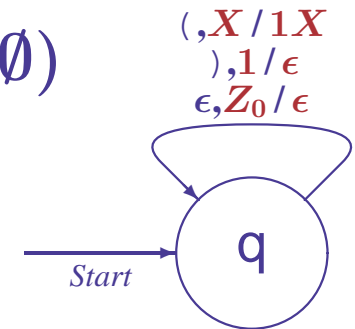
# UMWANDLUNG EINES PDA IN EINE GRAMMATIK

- Gegeben  $P_1 = (\{q\}, \{(\,)\}, \{Z_0, 1\}, \delta, q, Z_0, \emptyset)$

mit  $\delta(q, (\, X) = \{(q, 1X)\}$  für  $X \in \{Z_0, 1\}$

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

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



- Generiere  $G = (\{(\,)\}, \{S, (q, Z_0, q), (q, 1, q)\}, P_G, S)$

mit  $P_G = S \rightarrow (q, Z_0, q)$

$$(q, Z_0, q) \rightarrow ((q, 1, q)(q, Z_0, q)$$

$$(q, Z_0, q) \rightarrow \epsilon$$

$$(q, 1, q) \rightarrow ((q, 1, q)(q, 1, q)$$

$$(q, 1, q) \rightarrow )$$

Wähle Kurzschreibweise  $A/B$  für Hilfssymbole  $(q, Z_0, q)$  bzw.  $(q, 1, q)$ :

$$G = (\{(\,)\}, \{S, A, B\}, P, S)$$

mit  $P = \{S \rightarrow A, A \rightarrow (BA, A \rightarrow \epsilon, B \rightarrow (BB, B \rightarrow )\}$

# BRAUCHEN WIR NICHTDETERMINISTISCHE AUTOMATEN?

- **Grammatiken sind nichtdeterministisch**
  - Nichtdeterministische Automaten sind das “natürliche” Gegenstück
    - Grammatikregeln führen zu mengenwertiger Überföhrungsfunktion
  - “Wirkliche” Automaten müssen deterministisch sein



# BRAUCHEN WIR NICHTDETERMINISTISCHE AUTOMATEN?

- **Grammatiken sind nichtdeterministisch**
  - Nichtdeterministische Automaten sind das “natürliche” Gegenstück
    - Grammatikregeln führen zu mengenwertiger Überföhrungsfunktion
  - “Wirkliche” Automaten müssen deterministisch sein
- **Typ-3 Sprachen haben deterministische Modelle**
  - NEAs können in äquivalente DEAs umgewandelt werden
  - Teilmengenkonstruktion kann Automaten exponentiell vergrößern

# BRAUCHEN WIR NICHTDETERMINISTISCHE AUTOMATEN?

- **Grammatiken sind nichtdeterministisch**
  - Nichtdeterministische Automaten sind das “natürliche” Gegenstück
    - Grammatikregeln führen zu mengenwertiger Überföhrungsfunktion
  - “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?**
  - Überföhrungsfunktion  $\delta: Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$  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**
  - $\Sigma$  endliches **Eingabealphabet**
  - $\Gamma$  endliches **Stackalphabet**
  - $\delta: Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$  **Überföhrungsfunktion**  
 $\delta(q, \epsilon, X)$  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**

# 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**
- $\Sigma$  endliches **Eingabealphabet**
- $\Gamma$  endliches **Stackalphabet**
- $\delta: Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$  **Überföhrungsfunktion**  
 $\delta(q, \epsilon, X)$  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**

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

DPDAs SIND NICHT MÄCHTIG GENUG

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

## DPDAs SIND NICHT MÄCHTIG GENUG

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

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

# DPDAS SIND NICHT MÄCHTIG GENUG

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

- $L(DPDA) \subseteq \mathcal{L}_2$

- Jeder DPDA ist ein spezieller PDA

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

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

- Wenn  $0^n 1 10^n$  (großes  $n$ ) gelesen ist, ist Stack durchs Zählen geleert

- Wenn noch einmal  $0^n 1 10^n$  gelesen wird, muß  $P$  akzeptieren

- Wenn stattdessen  $0^m 1 10^m$  ( $m \neq n$ ) kommt, darf  $P$  nicht akzeptieren

- Aber die Information über  $n$  ist nicht mehr gespeichert

# DPDAS SIND NICHT MÄCHTIG GENUG

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

- $L(DPDA) \subseteq \mathcal{L}_2$

- Jeder DPDA ist ein spezieller PDA

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

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

- Wenn  $0^n 1 10^n$  (großes  $n$ ) gelesen ist, ist Stack durchs Zählen geleert

- Wenn noch einmal  $0^n 1 10^n$  gelesen wird, muß  $P$  akzeptieren

- Wenn stattdessen  $0^m 1 10^m$  ( $m \neq n$ ) kommt, darf  $P$  nicht akzeptieren

- Aber die Information über  $n$  ist nicht mehr gespeichert

Technisches Argument:  $\{ww^R \mid w \in \{0, 1\}^*\}$  ist keine **präfixfreie Sprache**

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



- **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  $L_\epsilon$  – DPDA kann deterministisch gemacht werden

# DETERMINISTISCHE SPRACHEN SIND EINDEUTIG

- **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  $L_\epsilon$  – DPDA kann deterministisch gemacht werden

- **Nicht jede eindeutige Typ-2 Sprache ist deterministisch**

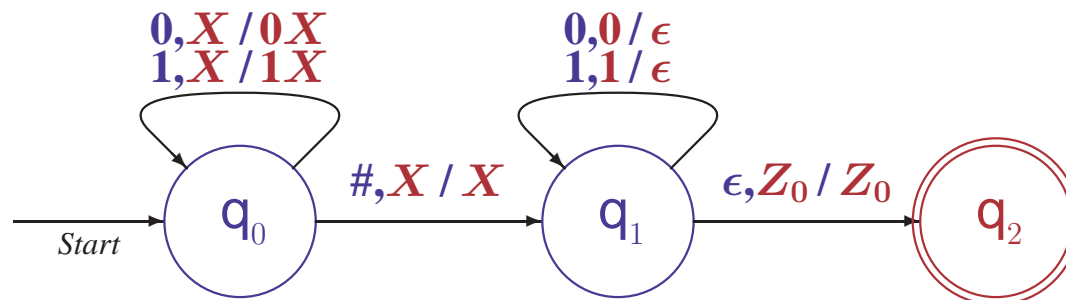
- $\{ww^R \mid w \in \{0, 1\}^*\}$  ist eindeutig, aber nicht deterministisch erkennbar

## DPDAs SIND MÄCHTIGER ALS ENDLICHE AUTOMATEN

- $\mathcal{L}_3 = L(DEA) \subseteq L_F(DPDA)$ 
  - Jeder DEA ist ein spezieller DPDA, der mit Endzustand akzeptiert

# DPDAs SIND MÄCHTIGER ALS ENDLICHE AUTOMATEN

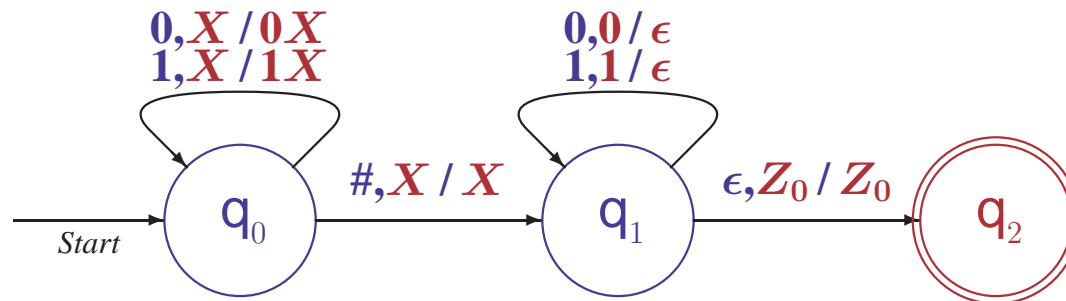
- $\mathcal{L}_3 = L(DEA) \subseteq L_F(DPDA)$ 
  - Jeder DEA ist ein spezieller DPDA, der mit Endzustand akzeptiert
- $L = \{w\#w^R \mid w \in \{0, 1\}^*\} \in L_F(DPDA) - L(DEA)$ 
  - $L$  ist nicht regulär
    - Beweis durch Pumping Lemma, analog zu  $\{ww^R \mid w \in \{0, 1\}^*\}$
  - $L = L_F(P)$  für folgenden DPDA  $P$



- $P$  ist deterministisch, da  $\epsilon$ -Übergang in  $q_1$  genau bei Stacksymbol  $Z_0$

# DPDAs SIND MÄCHTIGER ALS ENDLICHE AUTOMATEN

- $\mathcal{L}_3 = L(DEA) \subseteq L_F(DPDA)$ 
  - Jeder DEA ist ein spezieller DPDA, der mit Endzustand akzeptiert
- $L = \{w\#w^R \mid w \in \{0, 1\}^*\} \in L_F(DPDA) - L(DEA)$ 
  - $L$  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  $\epsilon$ -Übergang in  $q_1$  genau bei Stacksymbol  $Z_0$
- $\{0\}^* \notin L_\epsilon(DPDA)$ 
  - $\{0\}^*$  ist nicht präfixfrei

- **Maschinenmodell für kontextfreie Sprachen**
  - Nichtdeterministischer endlicher Automat mit Stack und  $\epsilon$ -Übergängen
  - Erkennung von Wörtern durch Endzustand oder leeren Stack
  - Erkennungsmodelle sind ineinander transformierbar
- **Verhaltensanalyse durch Konfigurationsübergänge**
  - Konfigurationen beschreiben ‘Gesamtzustand’ von Pushdown-Automaten
  - Konfigurationsübergänge verallgemeinern Überföhrungsfunktionen
- **Äquivalent zu kontextfreien Grammatiken**
  - Umwandlung von Konfigurationsübergängen in Regeln und umgekehrt
- **Deterministische PDAs sind weniger mächtig**
  - DPDAs erkennen nur eindeutige Typ-2 Sprachen
  - $L_\epsilon$ -DPDAs können nicht einmal alle regulären Sprachen erkennen