Computational Logic

Einführung in die Theoretische Informatik

Woche 8

Harald Zankl

Institut für Informatik @ UIBK
Wintersemester 2014/2015

Definition

Eine Grammatik G ist ein Quadrupel $G = (V, \Sigma, R, S)$, wobei

1. V eine endliche Menge von Variablen (oder Nichtterminale)
2. Σ ein Alphabet, die Terminale, $V \cap \Sigma = \varnothing$
3. R eine endliche Menge von Regeln
4. $S \in V$ das Startsymbol von G

Definition

Eine Grammatik G ist ein Quadrupel $G = (V, \Sigma, R, S)$, wobei

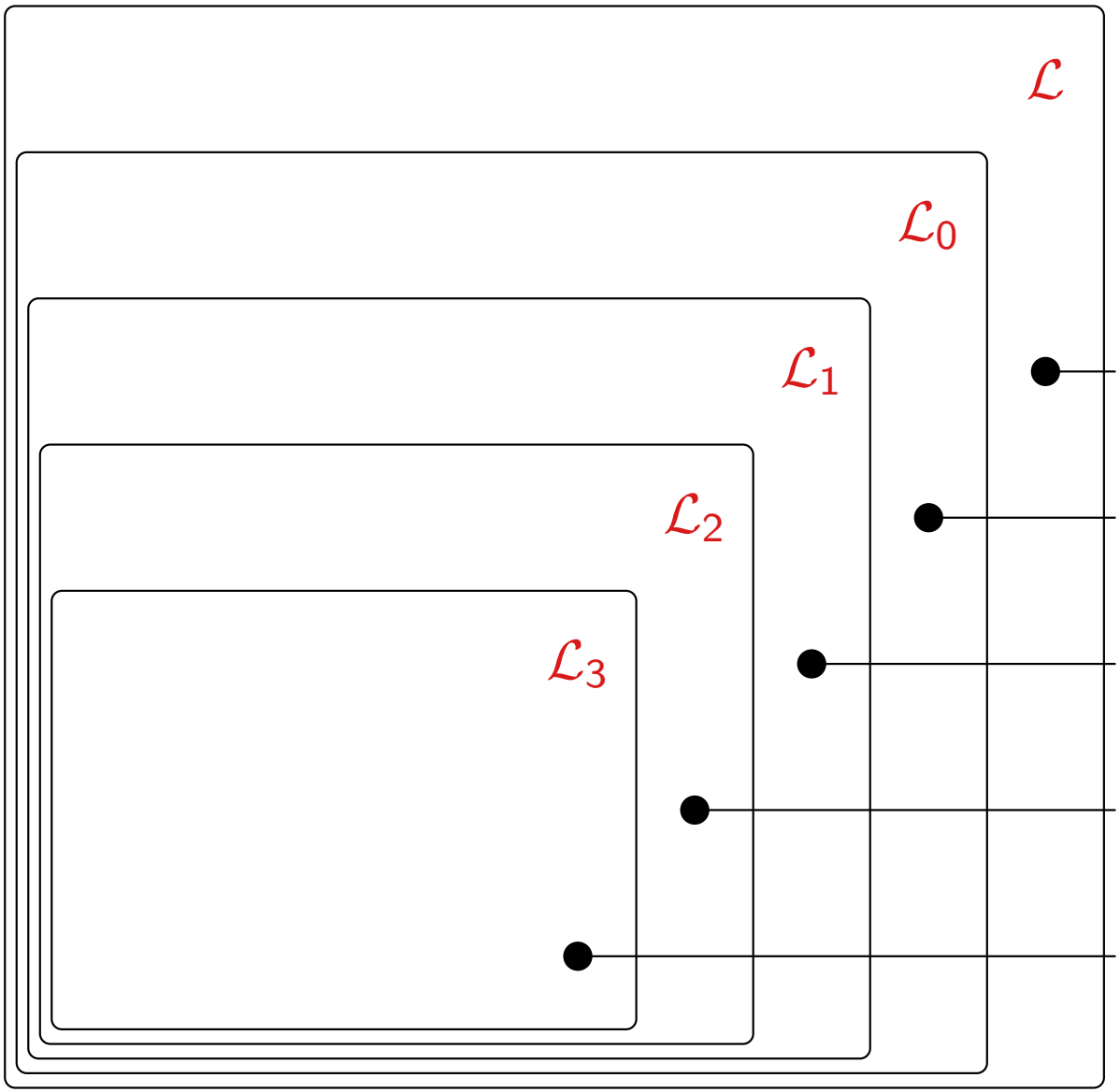
1. V eine endliche Menge von Variablen (oder Nichtterminale)
2. Σ ein Alphabet, die Terminale, $V \cap \Sigma = \varnothing$
3. R eine endliche Menge von Regeln
4. $S \in V$ das Startsymbol von G

Definition

Eine formale Sprache L heißt

- regulär (vom Typ 3)
  wenn $\exists$ rechtslineare Grammatik G mit $L = L(G)$
- kontextfrei (vom Typ 2)
  wenn $\exists$ kontextfreie Grammatik G mit $L = L(G)$
- kontextsensitiv (vom Typ 1)
  wenn $\exists$ kontextsensitive Grammatik G mit $L = L(G)$
- rekursiv aufzählbar (vom Typ 0) wenn $\exists$ Grammatik G mit $L = L(G)$

Chomsky-Hierarchie



$\{(E, s, t) \mid E \not\vdash s \approx t\}$

$\{(E, s, t) \mid E \vdash s \approx t\}$

$\mathrm{L}(G_3) = \{\mathrm{a}^n\mathrm{b}^n\mathrm{c}^n \mid n \geqslant 1\}$

$\mathrm{L}(G_2) =$ „Klammerausdrücke“

$\mathrm{L}(G_1) = \{0, 1\}^+$

Inhalte der Lehrveranstaltung

Einführung in die Logik

Syntax & Semantik der Aussagenlogik, Formales Beweisen, Konjunktive und Disjunktive Normalformen

Einführung in die Algebra

Boolesche Algebra, Universelle Algebra, Logische Schaltkreise

Einführung in die Theorie der Formalen Sprachen

Grammatiken und Formale Sprachen, Reguläre Sprachen, Kontextfreie Sprachen

Einführung in die Berechenbarkeitstheorie

Algorithmisch unlösbare Probleme, Turing Maschinen, Registermaschinen

Einführung in die Programmverifikation

Prinzipien der Analyse von Programmen, Verifikation nach Hoare, Verschlüsselung und Sicherheit

Deterministische endliche Automaten

Reguläre Sprachen können durch

- ... rechtslineare Grammatiken erzeugt werden
- ... deterministische endliche Automaten akzeptiert werden

Deterministische endliche Automaten

Reguläre Sprachen können durch

- ... rechtslineare Grammatiken erzeugt werden
- ... deterministische endliche Automaten akzeptiert werden

Definition (Deterministischer endlicher Automat (kurz: DEA))

Ein DEA ist ein 5-Tupel $A = (Q, \Sigma, \delta, s, F)$ sodass

Deterministische endliche Automaten

Reguläre Sprachen können durch

- ... rechtslineare Grammatiken erzeugt werden
- ... deterministische endliche Automaten akzeptiert werden

Definition (Deterministischer endlicher Automat (kurz: DEA))

Ein DEA ist ein 5-Tupel $A = (Q, \Sigma, \delta, s, F)$ sodass

1. Q eine endliche Menge von Zuständen

Deterministische endliche Automaten

Reguläre Sprachen können durch

- ... rechtslineare Grammatiken erzeugt werden
- ... deterministische endliche Automaten akzeptiert werden

Definition (Deterministischer endlicher Automat (kurz: DEA))

Ein DEA ist ein 5-Tupel $A = (Q, \Sigma, \delta, s, F)$ sodass

1. Q eine endliche Menge von Zuständen
2. Σ eine endliche Menge von Eingabesymbolen (Σ wird auch Eingabealphabet genannt)

Deterministische endliche Automaten

Reguläre Sprachen können durch

- ... rechtslineare Grammatiken erzeugt werden
- ... deterministische endliche Automaten akzeptiert werden

Definition (Deterministischer endlicher Automat (kurz: DEA))

Ein DEA ist ein 5-Tupel $A = (Q, \Sigma, \delta, s, F)$ sodass

1. Q eine endliche Menge von Zuständen
2. Σ eine endliche Menge von Eingabesymbolen (Σ wird auch Eingabealphabet genannt)
3. $\delta\colon Q \times \Sigma \to Q$ die Übergangsfunktion

Deterministische endliche Automaten

Reguläre Sprachen können durch

- ... rechtslineare Grammatiken erzeugt werden
- ... deterministische endliche Automaten akzeptiert werden

Definition (Deterministischer endlicher Automat (kurz: DEA))

Ein DEA ist ein 5-Tupel $A = (Q, \Sigma, \delta, s, F)$ sodass

1. Q eine endliche Menge von Zuständen
2. Σ eine endliche Menge von Eingabesymbolen (Σ wird auch Eingabealphabet genannt)
3. $\delta\colon Q \times \Sigma \to Q$ die Übergangsfunktion
4. $s \in Q$ der Startzustand

Deterministische endliche Automaten

Reguläre Sprachen können durch

- ... rechtslineare Grammatiken erzeugt werden
- ... deterministische endliche Automaten akzeptiert werden

Definition (Deterministischer endlicher Automat (kurz: DEA))

Ein DEA ist ein 5-Tupel $A = (Q, \Sigma, \delta, s, F)$ sodass

1. Q eine endliche Menge von Zuständen
2. Σ eine endliche Menge von Eingabesymbolen (Σ wird auch Eingabealphabet genannt)
3. $\delta\colon Q \times \Sigma \to Q$ die Übergangsfunktion
4. $s \in Q$ der Startzustand
5. $F \subseteq Q$ eine endliche Menge von akzeptierenden Zuständen

Deterministische endliche Automaten

Reguläre Sprachen können durch

- ... rechtslineare Grammatiken erzeugt werden
- ... deterministische endliche Automaten akzeptiert werden

Definition (Deterministischer endlicher Automat (kurz: DEA))

Ein DEA ist ein 5-Tupel $A = (Q, \Sigma, \delta, s, F)$ sodass

1. Q eine endliche Menge von Zuständen
2. Σ eine endliche Menge von Eingabesymbolen (Σ wird auch Eingabealphabet genannt)
3. $\delta\colon Q \times \Sigma \to Q$ die Übergangsfunktion
4. $s \in Q$ der Startzustand
5. $F \subseteq Q$ eine endliche Menge von akzeptierenden Zuständen

Zu beachten: δ muss für alle möglichen Argumente definiert sein

Zustandstabelle

	$a_1 \in \Sigma$	$a_2 \in \Sigma$	$\cdots$
$q_1 \in Q$	$\delta(q_1, a_1)$	$\delta(q_1, a_2)$	$\cdots$
$q_2 \in Q$	$\delta(q_2, a_1)$	$\ddots$	
$\vdots$	$\vdots$		

Zustandstabelle

	$a_1 \in \Sigma$	$a_2 \in \Sigma$	$\cdots$
$q_1 \in Q$	$\delta(q_1, a_1)$	$\delta(q_1, a_2)$	$\cdots$
$q_2 \in Q$	$\delta(q_2, a_1)$	$\ddots$	
$\vdots$	$\vdots$		

Zustandsgraph

Sei $A = (Q, \Sigma, \delta, s, F)$ ein DEA. Der Zustandsgraph von A hat

1. Ecken Q,
2. eine Kante von p nach q (beschriftet mit a), wenn $\delta(p, a) = q$.

Zustandstabelle

	$a_1 \in \Sigma$	$a_2 \in \Sigma$	$\cdots$
$q_1 \in Q$	$\delta(q_1, a_1)$	$\delta(q_1, a_2)$	$\cdots$
$q_2 \in Q$	$\delta(q_2, a_1)$	$\ddots$	
$\vdots$	$\vdots$		

Zustandsgraph

Sei $A = (Q, \Sigma, \delta, s, F)$ ein DEA. Der Zustandsgraph von A hat

1. Ecken Q,
2. eine Kante von p nach q (beschriftet mit a), wenn $\delta(p, a) = q$.

Konvention

- Den Startzustand markiert man mit einem Pfeil; die akzeptierenden Zustände werden mit einem doppelten Kreis gekennzeichnet

Beispiel (DEA)

DEA $A = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$
mit δ gegeben durch Zustandstabelle

	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_0	q_0

Beispiel (DEA)

DEA A

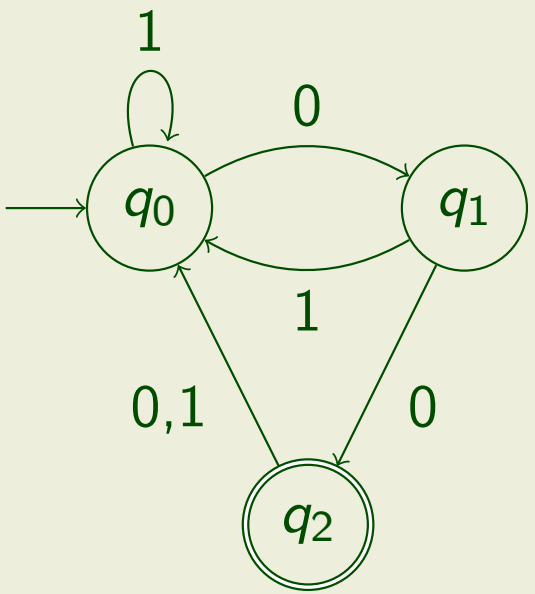
	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_0
q_2*	q_0	q_0

Beispiel (DEA)

DEA A

	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_0
q_2*	q_0	q_0

Zustandsgraph von A

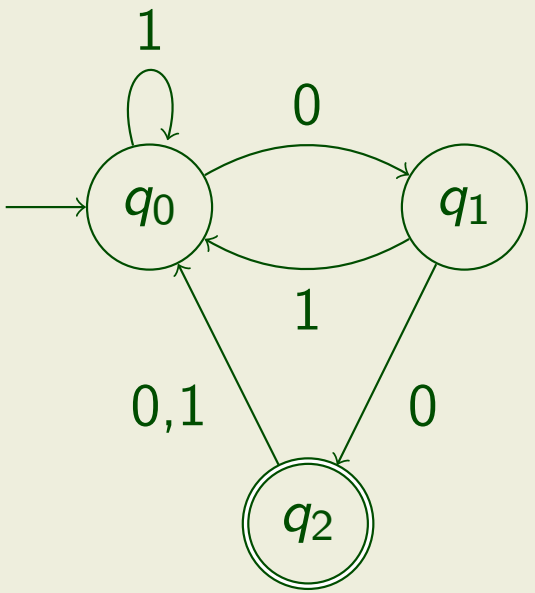


Beispiel (DEA)

DEA A

	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_0
q_2*	q_0	q_0

Zustandsgraph von A



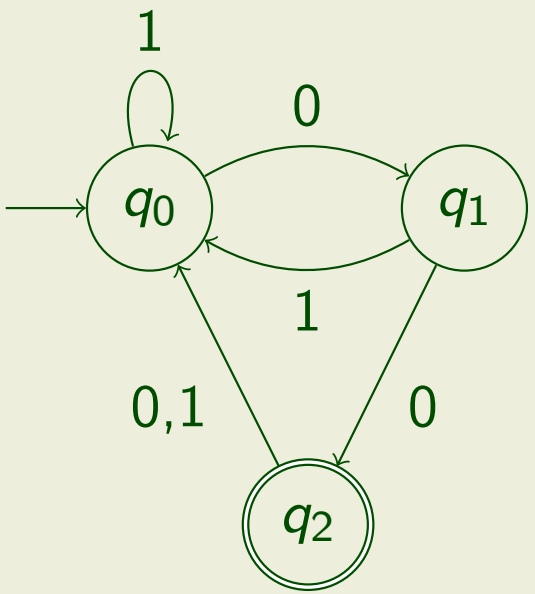
- $\epsilon \notin \mathrm{L}(A)$

Beispiel (DEA)

DEA A

	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_0
q_2*	q_0	q_0

Zustandsgraph von A



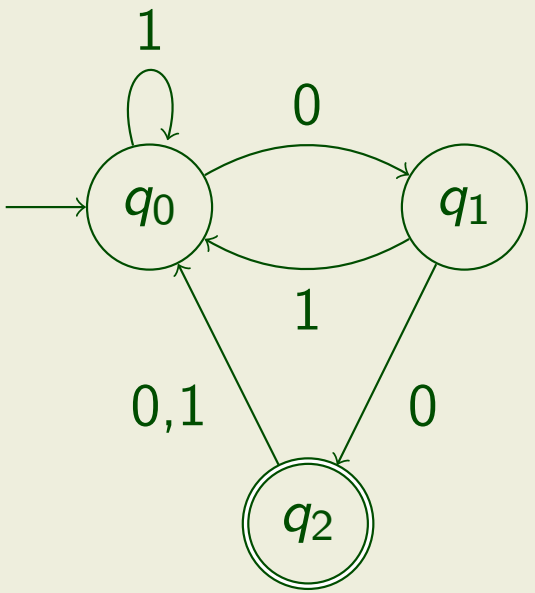
- $\epsilon \notin \mathrm{L}(A)$
- $0 \notin \mathrm{L}(A)$

Beispiel (DEA)

DEA A

	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_0
q_2*	q_0	q_0

Zustandsgraph von A



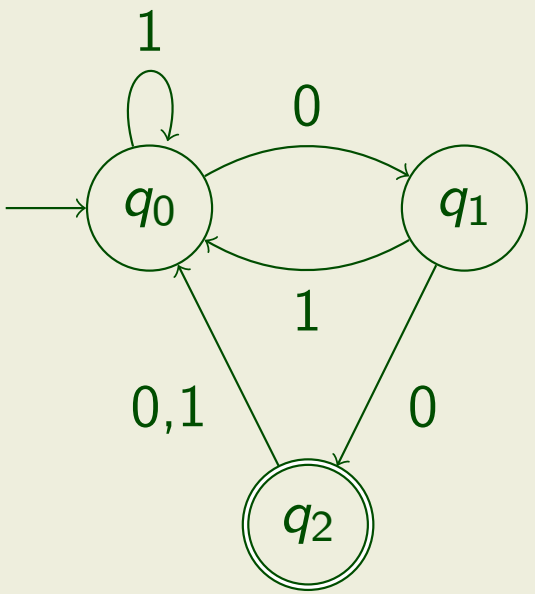
- $\epsilon \notin \mathrm{L}(A)$
- $0 \notin \mathrm{L}(A)$
- $00 \in \mathrm{L}(A)$

Beispiel (DEA)

DEA A

	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_0
q_2*	q_0	q_0

Zustandsgraph von A



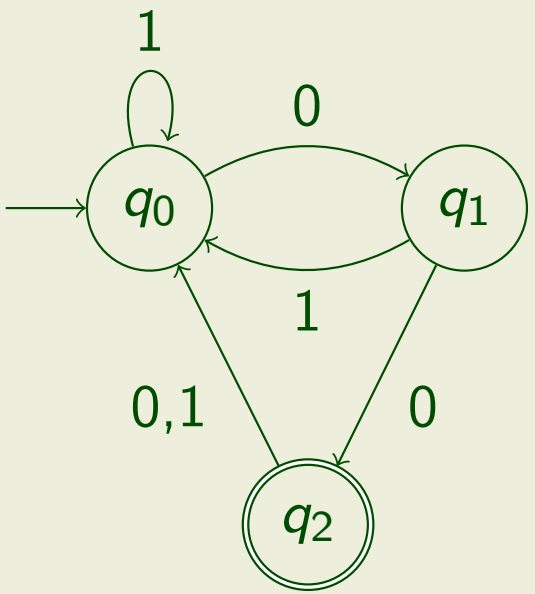
- $\epsilon \notin \mathrm{L}(A)$
- $0 \notin \mathrm{L}(A)$
- $00 \in \mathrm{L}(A)$
- $000 \notin \mathrm{L}(A)$

Beispiel (DEA)

DEA A

	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_0
q_2*	q_0	q_0

Zustandsgraph von A



- $\epsilon \notin \mathrm{L}(A)$
- $0 \notin \mathrm{L}(A)$
- $00 \in \mathrm{L}(A)$
- $000 \notin \mathrm{L}(A)$
- $101100 \in \mathrm{L}(A)$

Definition (erweiterte Übergangsfunktion)

Sei δ eine Übergangsfunktion. Wir definieren die erweiterte Übergangsfunktion $\hat{\delta}\colon Q \times \Sigma^* \to Q$ induktiv:

Definition (erweiterte Übergangsfunktion)

Sei δ eine Übergangsfunktion. Wir definieren die erweiterte Übergangsfunktion $\hat{\delta}\colon Q \times \Sigma^* \to Q$ induktiv:

$$\hat{\delta}(q, \epsilon) := q$$

Definition (erweiterte Übergangsfunktion)

Sei δ eine Übergangsfunktion. Wir definieren die erweiterte Übergangsfunktion $\hat{\delta}\colon Q \times \Sigma^* \to Q$ induktiv:

$$\hat{\delta}(q, \epsilon) := q$$

$$\hat{\delta}(q, xa) := \delta(\hat{\delta}(q, x), a) \qquad x \in \Sigma^*,\ a \in \Sigma$$

Definition (erweiterte Übergangsfunktion)

Sei δ eine Übergangsfunktion. Wir definieren die erweiterte Übergangsfunktion $\hat{\delta}\colon Q \times \Sigma^* \to Q$ induktiv:

$$\hat{\delta}(q, \epsilon) := q$$
$$\hat{\delta}(q, xa) := \delta(\hat{\delta}(q, x), a) \qquad x \in \Sigma^*,\ a \in \Sigma$$

Definition

Sei $A = (Q, \Sigma, \delta, s, F)$ ein DEA. Die von A akzeptierte Sprache ist

$$L(A) := \{x \in \Sigma^* \mid \hat{\delta}(s, x) \in F\}$$

Definition (erweiterte Übergangsfunktion)

Sei δ eine Übergangsfunktion. Wir definieren die erweiterte Übergangsfunktion $\hat{\delta}\colon Q \times \Sigma^* \to Q$ induktiv:

$$\hat{\delta}(q, \epsilon) := q$$
$$\hat{\delta}(q, xa) := \delta(\hat{\delta}(q, x), a) \qquad x \in \Sigma^*,\ a \in \Sigma$$

Definition

Sei $A = (Q, \Sigma, \delta, s, F)$ ein DEA. Die von A akzeptierte Sprache ist

$$\mathrm{L}(A) := \{x \in \Sigma^* \mid \hat{\delta}(s, x) \in F\}$$

Satz

Für jeden DEA A ist $\mathrm{L}(A)$ regulär. Umgekehrt existiert zu jeder regulären Sprache L ein DEA A, sodass $L = \mathrm{L}(A)$.

Beispiel

Definiere DEA A, der alle aus Nullen und Einsen bestehenden Wörter akzeptiert, welche die Folge 01 enthalten

Beispiel

Definiere DEA A, der alle aus Nullen und Einsen bestehenden Wörter akzeptiert, welche die Folge 01 enthalten

$$L = \{x01y \mid x, y \text{ sind beliebige Wörter aus Nullen und Einsen}\}$$

Beispiel

Definiere DEA A, der alle aus Nullen und Einsen bestehenden Wörter akzeptiert, welche die Folge 01 enthalten

$$L = \{x01y \mid x, y \text{ sind beliebige Wörter aus Nullen und Einsen}\}$$

Wir machen die folgende Hilfsüberlegung:

Beispiel

Definiere DEA A, der alle aus Nullen und Einsen bestehenden Wörter akzeptiert, welche die Folge 01 enthalten

$$L = \{x01y \mid x, y \text{ sind beliebige Wörter aus Nullen und Einsen}\}$$

Wir machen die folgende Hilfsüberlegung:

q_0	A hat Sequenz 01 noch nicht gefunden
	A wechselt in Zustand q_1, sobald 0 gelesen wird
	Sonst verharrt A in Zustand q_0

Beispiel

Definiere DEA A, der alle aus Nullen und Einsen bestehenden Wörter akzeptiert, welche die Folge 01 enthalten

$$L = \{x01y \mid x, y \text{ sind beliebige Wörter aus Nullen und Einsen}\}$$

Wir machen die folgende Hilfsüberlegung:

q_0	A hat Sequenz 01 noch nicht gefunden
	A wechselt in Zustand q_1, sobald 0 gelesen wird
	Sonst verharrt A in Zustand q_0
q_1	A hat Sequenz 0 gelesen
	A wechselt in Zustand q_2, sobald 1 gelesen wird
	Sonst verharrt A in Zustand q_1

Beispiel

Definiere DEA A, der alle aus Nullen und Einsen bestehenden Wörter akzeptiert, welche die Folge 01 enthalten

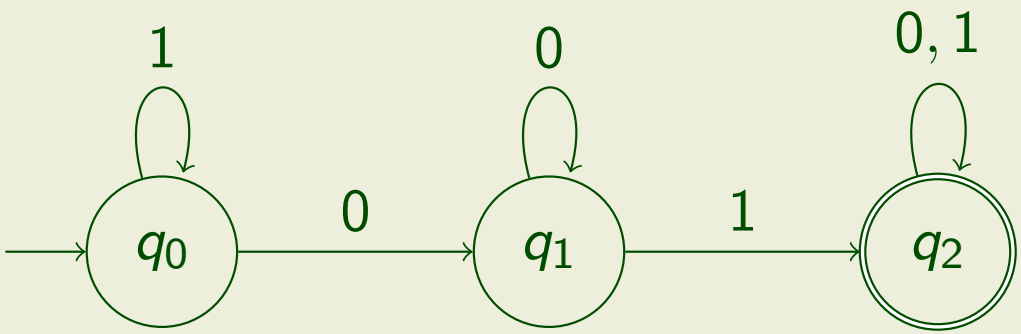
$$L = \{x01y \mid x, y \text{ sind beliebige Wörter aus Nullen und Einsen}\}$$

Wir machen die folgende Hilfsüberlegung:

q_0	A hat Sequenz 01 noch nicht gefunden A wechselt in Zustand q_1, sobald 0 gelesen wird Sonst verharrt A in Zustand q_0
q_1	A hat Sequenz 0 gelesen A wechselt in Zustand q_2, sobald 1 gelesen wird Sonst verharrt A in Zustand q_1
q_2	A hat Sequenz 01 gelesen A akzeptiert jede weitere Eingabe

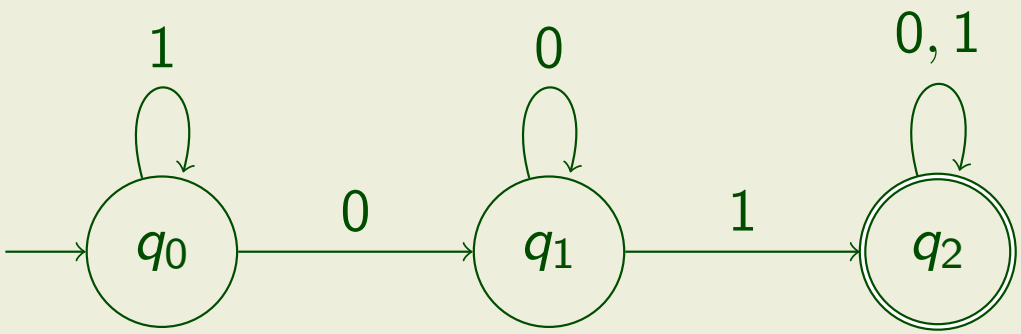
Beispiel (Fortsetzung)

Der Automat kann durch den Zustandsgraphen angegeben werden:



Beispiel (Fortsetzung)

Der Automat kann durch den Zustandsgraphen angegeben werden:



Es ist auch eine Darstellung durch die Zustandstabelle denkbar, wobei der Startzustand durch einen Pfeil gekennzeichnet ist und die akzeptierenden Zustände durch $*$.

	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
q_2*	q_2	q_2

Abschlusseigenschaften von regulären Sprachen

Satz

Seien L, M reguläre Sprachen (über dem Alphabet Σ), dann gilt

1. *Die Vereinigung $L \cup M$ ist regulär*
2. *Das Komplement $\sim L$ ist regulär*
3. *Der Schnitt $L \cap M$ ist regulär*
4. *Die Mengendifferenz $L \setminus M$ ist regulär*
5. *L^* ist regulär*

Anwendungen von regulären Sprachen

- Software zum Entwurf und Testen von digitalen Schaltkreisen

Anwendungen von regulären Sprachen

- Software zum Entwurf und Testen von digitalen Schaltkreisen
- Softwarebausteine eines Compilers, etwa in der lexikalischen Analyse:
  1. lexikalische Scanner („Lexer“) wird mit endlichen Automaten implementiert
  2. Der lexikalische Scanner dient zur Aufteilung des Eingabetextes in logische Einheiten, wie Bezeichner oder Schlüsselwörter

Anwendungen von regulären Sprachen

- Software zum Entwurf und Testen von digitalen Schaltkreisen
- Softwarebausteine eines Compilers, etwa in der lexikalischen Analyse:
  1. lexikalische Scanner („Lexer“) wird mit endlichen Automaten implementiert
  2. Der lexikalische Scanner dient zur Aufteilung des Eingabetextes in logische Einheiten, wie Bezeichner oder Schlüsselwörter
- Software zum Durchsuchen umfangreicher Texte

Anwendungen von regulären Sprachen

- Software zum Entwurf und Testen von digitalen Schaltkreisen
- Softwarebausteine eines Compilers, etwa in der lexikalischen Analyse:
  1. lexikalische Scanner („Lexer“) wird mit endlichen Automaten implementiert
  2. Der lexikalische Scanner dient zur Aufteilung des Eingabetextes in logische Einheiten, wie Bezeichner oder Schlüsselwörter
- Software zum Durchsuchen umfangreicher Texte
- Software zur Verifizierung aller Arten von Systemen, die eine endliche Anzahl verschiedener Zustände besitzen

Anwendungen von regulären Sprachen

- Software zum Entwurf und Testen von digitalen Schaltkreisen
- Softwarebausteine eines Compilers, etwa in der lexikalischen Analyse:
  1. lexikalische Scanner („Lexer“) wird mit endlichen Automaten implementiert
  2. Der lexikalische Scanner dient zur Aufteilung des Eingabetextes in logische Einheiten, wie Bezeichner oder Schlüsselwörter
- Software zum Durchsuchen umfangreicher Texte
- Software zur Verifizierung aller Arten von Systemen, die eine endliche Anzahl verschiedener Zustände besitzen
- Softwarebausteine eines Computerspiels:
  1. Kontrolle von Spielfiguren kann mit Hilfe eines endlichen Automaten implementiert werden
  2. erlaubt eine bessere Modularisierung des Codes

Anwendungen von regulären Sprachen

- Software zum Entwurf und Testen von digitalen Schaltkreisen
- Softwarebausteine eines Compilers, etwa in der lexikalischen Analyse:
  1. lexikalische Scanner („Lexer“) wird mit endlichen Automaten implementiert
  2. Der lexikalische Scanner dient zur Aufteilung des Eingabetextes in logische Einheiten, wie Bezeichner oder Schlüsselwörter
- Software zum Durchsuchen umfangreicher Texte
- Software zur Verifizierung aller Arten von Systemen, die eine endliche Anzahl verschiedener Zustände besitzen
- Softwarebausteine eines Computerspiels:
  1. Kontrolle von Spielfiguren kann mit Hilfe eines endlichen Automaten implementiert werden
  2. erlaubt eine bessere Modularisierung des Codes

Beispiel

Wir untersuchen Protokolle, die den Gebrauch elektronischen „Geldes“ ermöglichen; dabei handeln: der Kunde, die Bank und das Geschäft:

Beispiel

Wir untersuchen Protokolle, die den Gebrauch elektronischen „Geldes“ ermöglichen; dabei handeln: der Kunde, die Bank und das Geschäft:

- Der Kunde kann zahlen

Beispiel

Wir untersuchen Protokolle, die den Gebrauch elektronischen „Geldes“ ermöglichen; dabei handeln: der Kunde, die Bank und das Geschäft:

- Der Kunde kann zahlen
- Der Kunde kann das Geld löschen

Beispiel

Wir untersuchen Protokolle, die den Gebrauch elektronischen „Geldes“ ermöglichen; dabei handeln: der Kunde, die Bank und das Geschäft:

- Der Kunde kann zahlen
- Der Kunde kann das Geld löschen
- Das Geschäft kann dem Kunden Waren zusenden

Beispiel

Wir untersuchen Protokolle, die den Gebrauch elektronischen „Geldes“ ermöglichen; dabei handeln: der Kunde, die Bank und das Geschäft:

- Der Kunde kann zahlen
- Der Kunde kann das Geld löschen
- Das Geschäft kann dem Kunden Waren zusenden
- Das Geschäft kann Geld einlösen

Beispiel

Wir untersuchen Protokolle, die den Gebrauch elektronischen „Geldes“ ermöglichen; dabei handeln: der Kunde, die Bank und das Geschäft:

- Der Kunde kann zahlen
- Der Kunde kann das Geld löschen
- Das Geschäft kann dem Kunden Waren zusenden
- Das Geschäft kann Geld einlösen
- Die Bank kann Geld überweisen

Beispiel

Wir untersuchen Protokolle, die den Gebrauch elektronischen „Geldes“ ermöglichen; dabei handeln: der Kunde, die Bank und das Geschäft:

- Der Kunde kann zahlen
- Der Kunde kann das Geld löschen
- Das Geschäft kann dem Kunden Waren zusenden
- Das Geschäft kann Geld einlösen
- Die Bank kann Geld überweisen

Wir treffen die folgenden Grundannahmen:

- Der Kunde ist unverantwortlich
- Das Geschäft ist verantwortlich, aber gutgläubig
- Die Bank ist strikt

Beispiel

Wir untersuchen Protokolle, die den Gebrauch elektronischen „Geldes“ ermöglichen; dabei handeln: der Kunde, die Bank und das Geschäft:

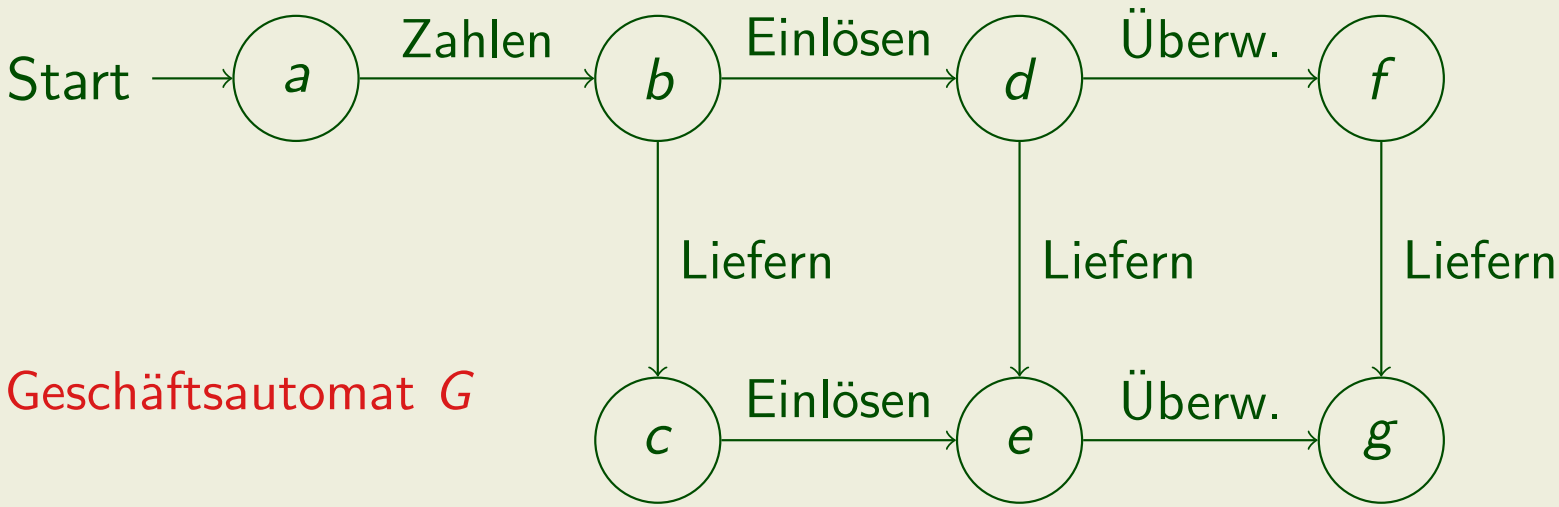
- Der Kunde kann zahlen
- Der Kunde kann das Geld löschen
- Das Geschäft kann dem Kunden Waren zusenden
- Das Geschäft kann Geld einlösen
- Die Bank kann Geld überweisen

Wir treffen die folgenden Grundannahmen:

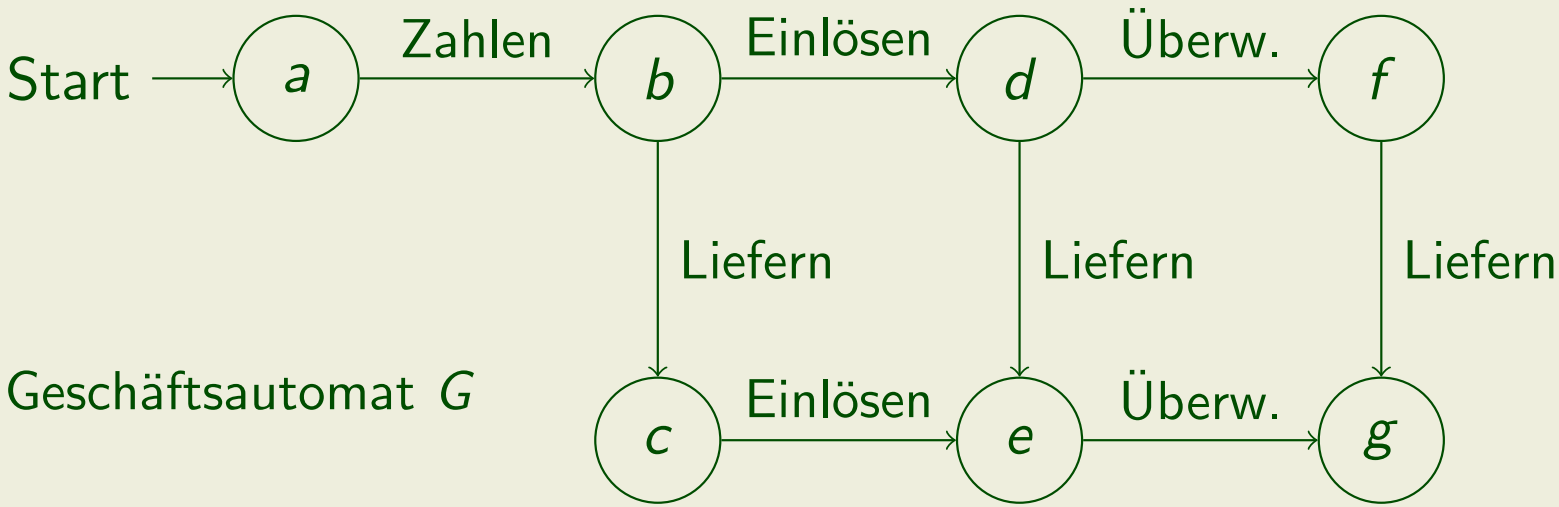
- Der Kunde ist unverantwortlich
- Das Geschäft ist verantwortlich, aber gutgläubig
- Die Bank ist strikt

wir betrachten die Handlungen der Akteure als „extern“; die Sequenz der Handlungen ist wichtig, nicht wer sie initiiert

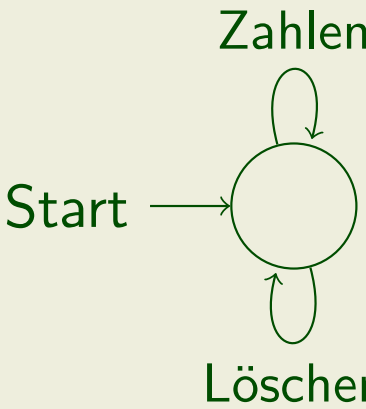
Beispiel (Fortsetzung)



Beispiel (Fortsetzung)

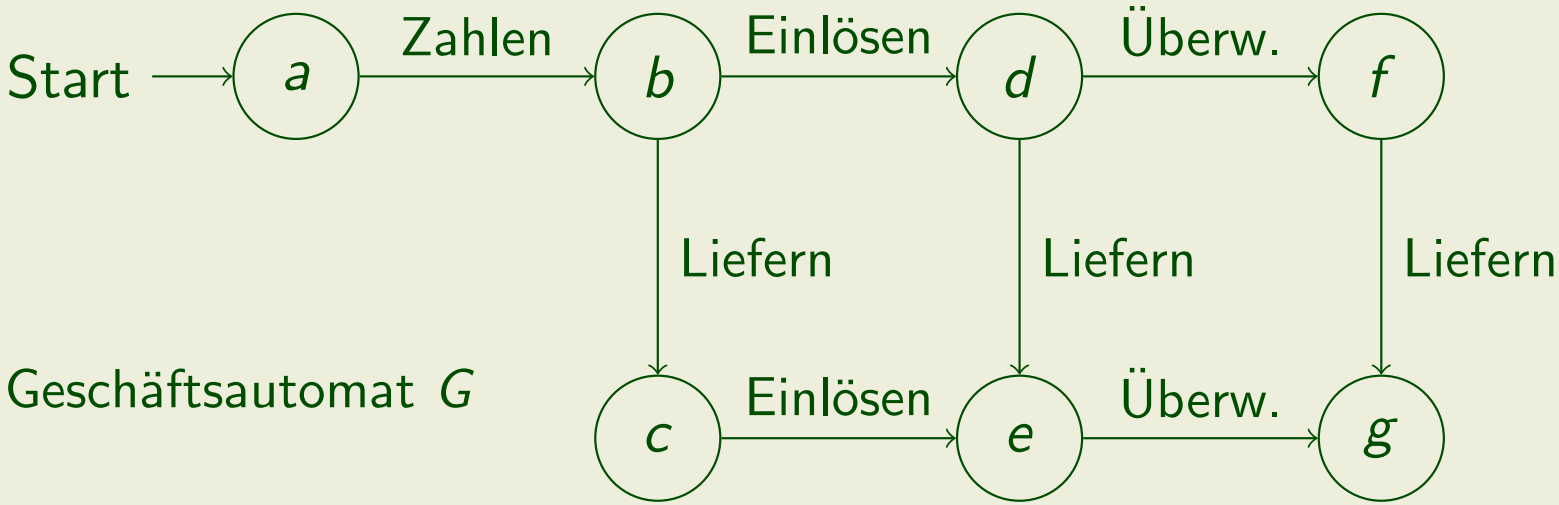


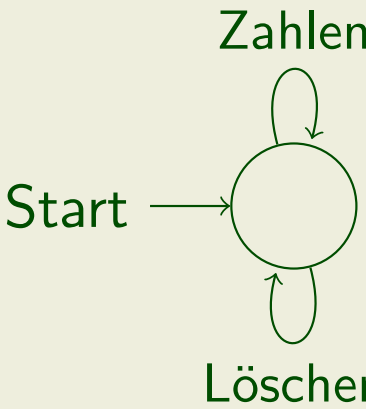
Geschäftsautomat G



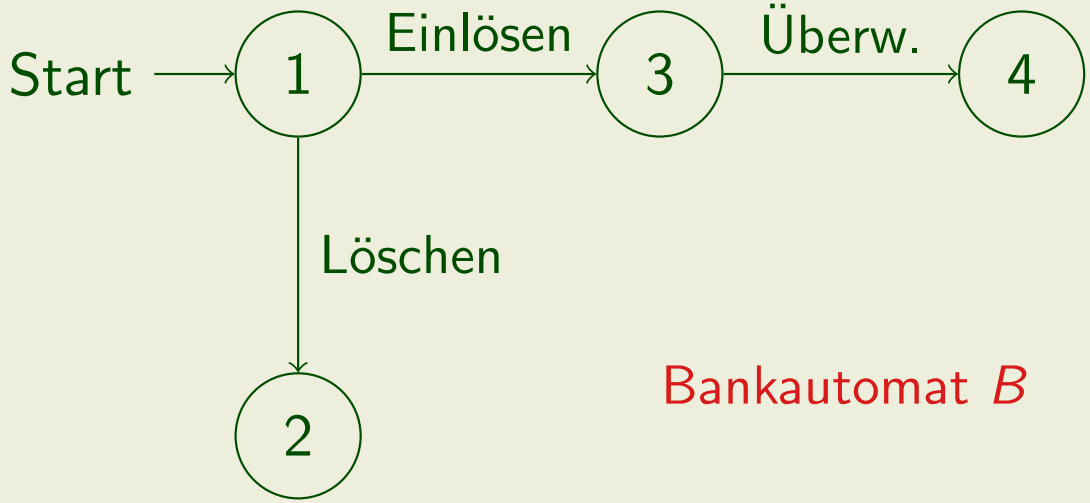
Kundenautomat K

Beispiel (Fortsetzung)



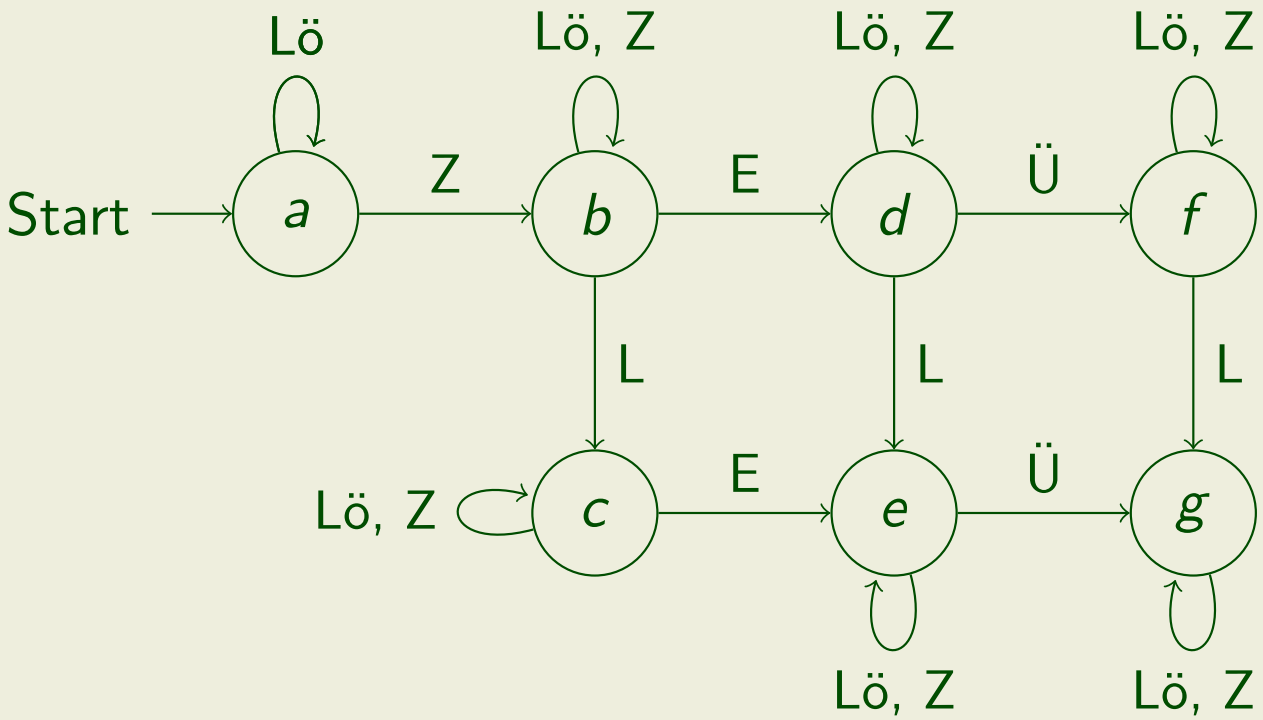


Kundenautomat K

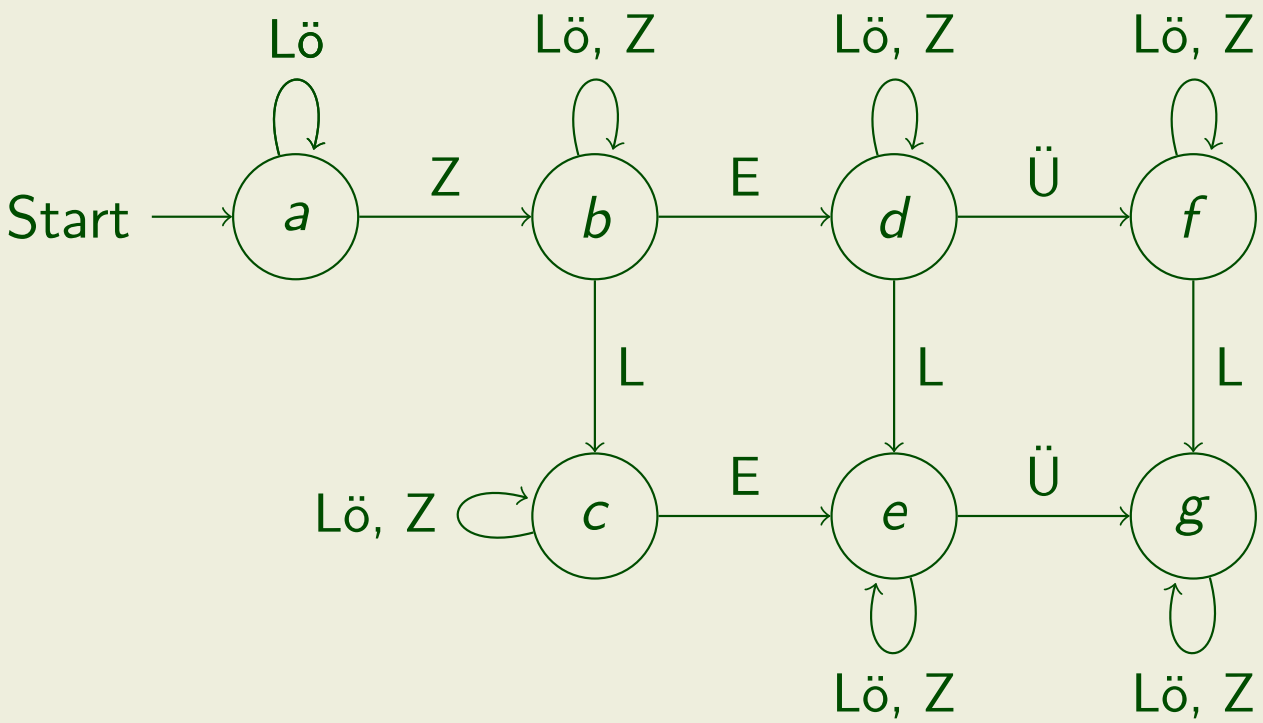


Bankautomat B

Beispiel (Fortsetzung)



Beispiel (Fortsetzung)



Zahlen. . . Z

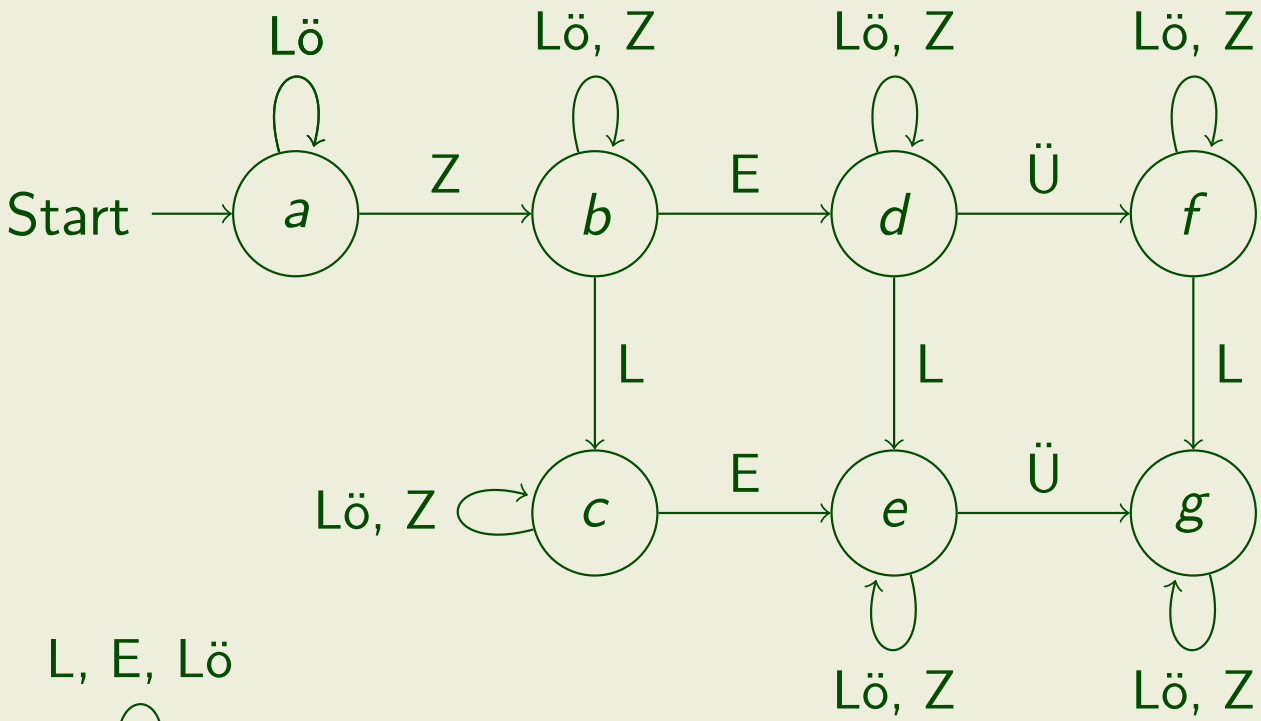
Einlösen. . . E

Löschen. . . Lö

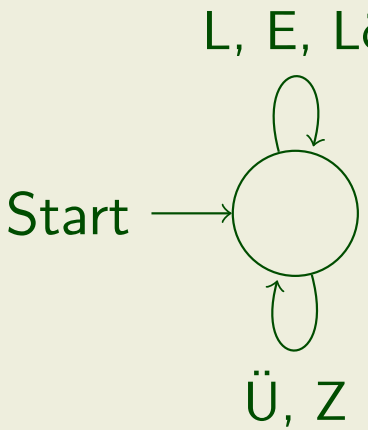
Liefern. . . L

Überw. . . . Ü

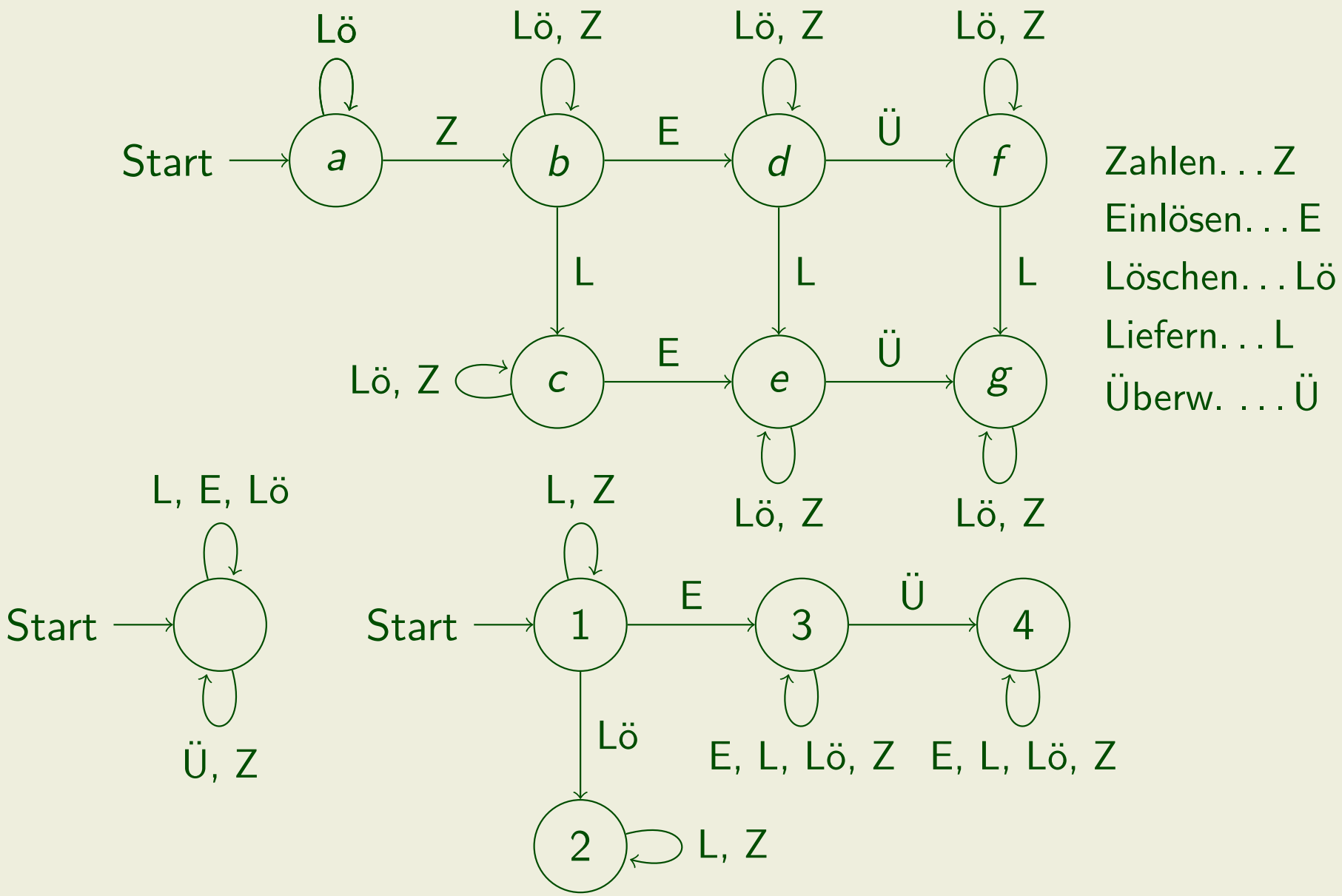
Beispiel (Fortsetzung)



Zahlen. . . Z
Einlösen. . . E
Löschen. . . Lö
Liefern. . . L
Überw. . . . Ü



Beispiel (Fortsetzung)



Beispiel (Fortsetzung)

Wir definieren den Produktautomaten $B \times G$ aus B und G:

Beispiel (Fortsetzung)

Wir definieren den Produktautomaten $B \times G$ aus B und G:

1. Die Zustände dieses Automaten sind:

$$(i, x) \quad \text{wobei } i \in \{1, 2, 3, 4\} \text{ und } x \in \{a, b, c, d, e, f, g\}$$

Beispiel (Fortsetzung)

Wir definieren den Produktautomaten $B \times G$ aus B und G:

1. Die Zustände dieses Automaten sind:

   $$(i, x) \quad \text{wobei } i \in \{1, 2, 3, 4\} \text{ und } x \in \{a, b, c, d, e, f, g\}$$

2. Die Übergänge werden durch paralleles Ausführen von B und G definiert:

   $$\text{wenn } i \xrightarrow{\text{Aktion}} i' \text{ und } x \xrightarrow{\text{Aktion}} x' \quad \text{dann} \quad (i, x) \xrightarrow{\text{Aktion}} (i', x')$$

Beispiel (Fortsetzung)

Wir definieren den Produktautomaten $B \times G$ aus B und G:

1. Die Zustände dieses Automaten sind:

   $$(i, x) \quad \text{wobei } i \in \{1, 2, 3, 4\} \text{ und } x \in \{a, b, c, d, e, f, g\}$$

2. Die Übergänge werden durch paralleles Ausführen von B und G definiert:

   $$\text{wenn } i \xrightarrow{\text{Aktion}} i' \text{ und } x \xrightarrow{\text{Aktion}} x' \quad \text{dann} \quad (i, x) \xrightarrow{\text{Aktion}} (i', x')$$

Betrachte B und G:

Beispiel (Fortsetzung)

Wir definieren den Produktautomaten $B \times G$ aus B und G:

1. Die Zustände dieses Automaten sind:

   $$(i, x) \quad \text{wobei } i \in \{1, 2, 3, 4\} \text{ und } x \in \{a, b, c, d, e, f, g\}$$

2. Die Übergänge werden durch paralleles Ausführen von B und G definiert:

   $$\text{wenn } i \xrightarrow{\text{Aktion}} i' \text{ und } x \xrightarrow{\text{Aktion}} x' \quad \text{dann} \quad (i, x) \xrightarrow{\text{Aktion}} (i', x')$$

Betrachte B und G:

- in B gilt, dass aus Zustand 1 durch die Aktion „Einlösen“ Zustand 3 wird, konzise: $1 \xrightarrow{\text{Einlösen}} 3$

Beispiel (Fortsetzung)

Wir definieren den Produktautomaten $B \times G$ aus B und G:

1. Die Zustände dieses Automaten sind:

   $$(i, x) \quad \text{wobei } i \in \{1, 2, 3, 4\} \text{ und } x \in \{a, b, c, d, e, f, g\}$$

2. Die Übergänge werden durch paralleles Ausführen von B und G definiert:

   $$\text{wenn } i \xrightarrow{\text{Aktion}} i' \text{ und } x \xrightarrow{\text{Aktion}} x' \quad \text{dann} \quad (i, x) \xrightarrow{\text{Aktion}} (i', x')$$

Betrachte B und G:

- in B gilt, dass aus Zustand 1 durch die Aktion „Einlösen“ Zustand 3 wird, konzise: $1 \xrightarrow{\text{Einlösen}} 3$
- in G gilt, dass aus Zustand b mit Aktion „Einlösen“ d wird, konzise: $b \xrightarrow{\text{Einlösen}} d$

Beispiel (Fortsetzung)

Wir definieren den Produktautomaten $B \times G$ aus B und G:

1. Die Zustände dieses Automaten sind:

$$(i, x) \quad \text{wobei } i \in \{1, 2, 3, 4\} \text{ und } x \in \{a, b, c, d, e, f, g\}$$

2. Die Übergänge werden durch paralleles Ausführen von B und G definiert:

$$\text{wenn } i \xrightarrow{\text{Aktion}} i' \text{ und } x \xrightarrow{\text{Aktion}} x' \quad \text{dann} \quad (i, x) \xrightarrow{\text{Aktion}} (i', x')$$

Betrachte B und G:

- in B gilt, dass aus Zustand 1 durch die Aktion „Einlösen“ Zustand 3 wird, konzise: $1 \xrightarrow{\text{Einlösen}} 3$
- in G gilt, dass aus Zustand b mit Aktion „Einlösen“ d wird, konzise: $b \xrightarrow{\text{Einlösen}} d$
- also gilt in $B \times G$, dass aus Zustand $(1, b)$ mit Aktion „Einlösen“ Zustand $(3, d)$ wird

Beispiel (Fortsetzung)

Wir definieren den Produktautomaten $B \times G$ aus B und G:

1. Die Zustände dieses Automaten sind:

   $$(i, x) \quad \text{wobei } i \in \{1, 2, 3, 4\} \text{ und } x \in \{a, b, c, d, e, f, g\}$$

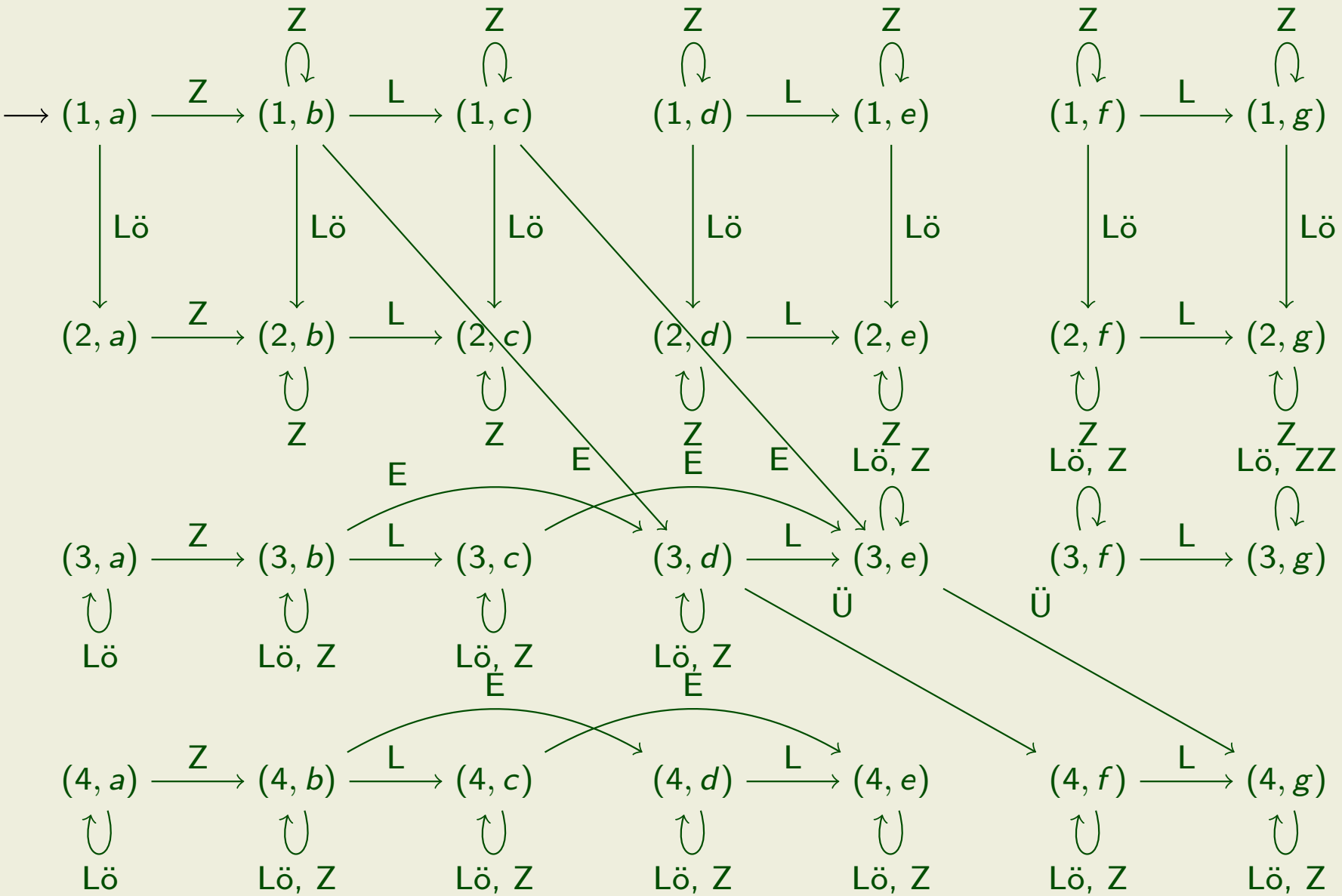
2. Die Übergänge werden durch paralleles Ausführen von B und G definiert:

   $$\text{wenn } i \xrightarrow{\text{Aktion}} i' \text{ und } x \xrightarrow{\text{Aktion}} x' \quad \text{dann} \quad (i, x) \xrightarrow{\text{Aktion}} (i', x')$$

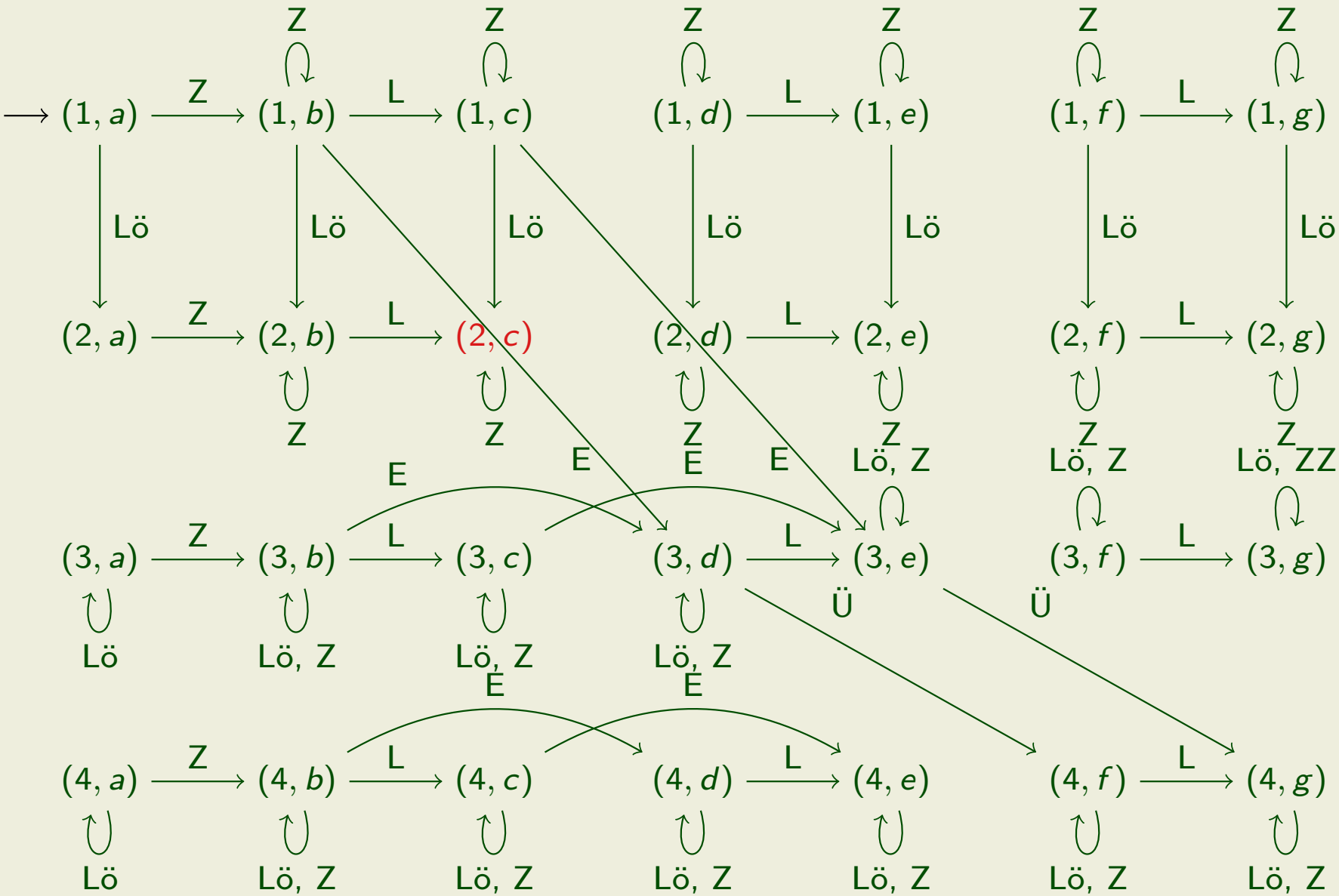
Betrachte B und G:

- in B gilt, dass aus Zustand 1 durch die Aktion „Einlösen“ Zustand 3 wird, konzise: $1 \xrightarrow{\text{Einlösen}} 3$
- in G gilt, dass aus Zustand b mit Aktion „Einlösen“ d wird, konzise: $b \xrightarrow{\text{Einlösen}} d$
- also gilt in $B \times G$, dass aus Zustand $(1, b)$ mit Aktion „Einlösen“ Zustand $(3, d)$ wird, konzise: $(1, b) \xrightarrow{\text{Einlösen}} (3, d)$

Beispiel (Fortsetzung)



Beispiel (Fortsetzung)



Beispiel (Fortsetzung)

Als Schlussfolgerung ergibt sich, dass das Protokoll nicht sicher ist:

> *Der Automat $B \times G$ kann in den Zustand $(2, c)$ gelangen, in welchem die Waren geschickt wurden und trotzdem nie eine Überweisung an das Geschäft erfolgen wird*