

## Sprachen & Automaten

- formale Sprachen zur Formulierung von Wissen
  - Programmiersprachen*
  - Logik*
  - Teilmengen natürlicher Sprachen*
  - ...
- Beschreibung formaler Sprachen durch Grammatiken
- Beschreibung und Verarbeitung formaler Sprachen durch Automaten, d.h. abstrakte Maschinen
- Klassifizierung formaler Sprachen
  - Typen von Sprachen*  $\leftrightarrow$
  - Typen von Grammatiken*  $\leftrightarrow$
  - Typen von Automaten*
- wichtige Typen formaler Sprachen und Automaten
  - reguläre Sprachen*  $\leftrightarrow$  *endliche Automaten*
  - kontextfreie Sprachen*  $\leftrightarrow$  *Kellerautomaten*

## Grundlagen

- Alphabet A
  - endliche, nichtleere Menge von Buchstaben
- Wort
  - endliche Aneinanderreihung (Konkatenation) von Elementen aus A
- leeres Wort
  - $\varepsilon$
- Menge aller Wörter  $A^*$ 
  - Menge aller endlichen Folgen von Elementen aus A
  - $A^* = \{a_1 a_2 \dots a_n \mid a_i \in A\} \cup \{\varepsilon\}$
  - $A^+ = A^* - \{\varepsilon\}$
- Beispiel
  - $A = \{a, b\}$
  - $A^* = \{\varepsilon, a, b, aa, ab, bb, aaa, aab, abb, \dots\}$
- Konkatenation
  - zweistellige Operation " $\cdot$ ":  $A^* \times A^* \rightarrow A^*$
  - $u, v \in A^*$ ,  $u \cdot v$  wird  $uv$  geschrieben
- algebraische Struktur  $(A^*, \cdot)$  ist Halbgruppe (Monoid)
  - $u, v \in A^* \Rightarrow u \cdot v = uv \in A^*$  (Abgeschlossenheit)
  - $(u \cdot v) \cdot w = u \cdot (v \cdot w) = uvw$  (Assoziativität)
  - $\varepsilon \cdot u = u \cdot \varepsilon = u$  (neutrales Element)

## Grundlagen

- Abkürzung
  - $u^n = uu\dots u$  (n-mal)
  - $u^0 = \varepsilon$
  - Beispiele:  $a^3 = aaa$ ,  $(ab)^3 = ababab$ ,  $(abc)^0 = \varepsilon$
- Länge eines Wortes  $w$ 
  - $|w|$
  - $|\varepsilon| = 0$ ,  $|a_1 a_2 \dots a_n| = n$  für  $a_i \in A$
  - $|uv| = |u| + |v|$ ,  $|u^n| = n \cdot |u|$
  - Beispiel:  $|abc| = 3$

## Sprachen

- Sprachen L über einem Alphabet A
  - $L \subseteq A^*$
- $L_1$  und  $L_2$  Sprachen über A
  - $L_1 \cup L_2 = \{u \mid u \in L_1 \text{ oder } u \in L_2\}$  (Vereinigung)
  - $L_1 \cap L_2 = \{u \mid u \in L_1 \text{ und } u \in L_2\}$  (Durchschnitt)
  - $L_1^c = A^* - L_1$  (Komplement)
  - $L_1 L_2 = \{uv \mid u \in L_1 \text{ und } v \in L_2\}$  (Produkt)
- Sprache L
  - $L^0 = \{\varepsilon\}$
  - $L^1 = L$
  - $L^{n+1} = LL^n$
  - $L^n L^m = L^{n+m}$
  - $(L^n)^m = L^{nm}$

Kleene-Sternoperation

$$L^* = \bigcup_{n \geq 0} L^n = \{u_1 u_2 \dots \mid u_i \in L\} \cup \{\varepsilon\}$$

$$L^+ = \bigcup_{n \geq 1} L^n$$

## Sprachen

- Beispiel

$$A = \{ (, ), +, -, *, /, X \}$$

(X steht für irgendeine Konstante oder Variable)

Sprache  $ARIM \subseteq A^*$  der arithmetischen Ausdrücke

X	∈ ARIM
X/(X+X) +X*X	∈ ARIM
X+*X	∉ ARIM

- Probleme

Sprachen sind i. A. unendliche Objekte; um damit umgehen zu können, brauchen wir endliche Beschreibungen.

Welche Worte gehören zur Sprache, welche nicht?

Diese beiden Fragen werden durch Grammatiken oder (abstrakte) Automaten beantwortet.

## Grammatiken

- (Ausschnitt aus einer) Grammatik für ARIM

$E \rightarrow T$	(Expression ist Term)
$E \rightarrow E + T$	(Expression ist Expression + Term)
$T \rightarrow F$	(Term ist Faktor)
$T \rightarrow T * F$	(Term ist Term * Faktor)
$F \rightarrow X$	(Faktor ist Konstante oder Variable)
$F \rightarrow (E)$	(Faktor ist Expression in Klammern)

- Grammatiken sind Regeln der Form

"linke Seite" → "rechte Seite"

Bedeutung der Regel: die rechte Seite kann die linke Seite ersetzen

(Abkürzungen: LHS für linke Seite, RHS für rechte Seite)

- Regeln enthalten Nichtterminalsymbole (z.B. E, T, F) und Terminalsymbole (z.B. X)

- Regeln werden solange angewendet, bis das abgeleitete Wort nur noch aus Terminalsymbolen besteht

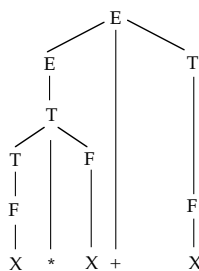
- jedes so abgeleitete Wort gehört zu der von der Grammatik definierten (erzeugten) Sprache

- Beispiel

$$E \Rightarrow E + T \Rightarrow T + T \Rightarrow T * F + T \Rightarrow F * F + T \Rightarrow X * F + T \Rightarrow X * X + T \Rightarrow X * X + F \Rightarrow X * X + X$$

## Grammatiken

- Ableitung kann als Syntaxbaum dargestellt werden



- bei der Ableitung wurde immer das am weitesten links stehende Nichtterminal ersetzt (Linksableitung)

## Grammatiken

- (Phrasenstruktur-) Grammatik G ist ein Quadrupel

$$G = (N, T, S, R)$$

N endliche Menge von Nichtterminalsymbolen

T endliche Menge von Terminalsymbolen

$$N \cap T = \emptyset$$

$S \in N$  ist das Startsymbol

R ist eine endliche Menge von Regeln (Produktionen)

$$R \subseteq (N \cup T)^+ \times (N \cup T)^*$$

- seien  $u = xyz$ ,  $v = xy'z$  mit  $x, z \in (N \cup T)^*$

Relation  $u \rightarrow_G v$  (u geht unter G unmittelbar in v über) falls

$y \rightarrow y'$  eine Regel in R ist

- G erzeugt (definiert) die Sprache

$$L(G) = \{ w \in T^* \mid S \rightarrow_G^* w \}$$

$\rightarrow_G^*$  ist die reflexiv transitive Hülle von  $\rightarrow_G$

- Grammatiken heißen äquivalent, wenn sie die gleiche Sprache erzeugen

## Grammatiken

- Folge von Wörtern ( $w_0, w_1, \dots, w_n$ ) mit  $w_0 = S$  und  $w_n \in T^*$  und  $w_0 \Rightarrow w_1 \Rightarrow \dots \Rightarrow w_n$  heisst (maximale) Ableitung von  $w_n$
- Anwendung einer Regel  $y \rightarrow y'$ , die aus dem Wort  $xyz$  das Wort  $xy'z$  macht, heisst linksmaximal (rechtsmaximal), wenn es keine Regel gibt, in der  $y$  weiter links (rechts) auftritt, als in der Zerlegung  $xyz$
- Linksableitung (Rechtsableitung) ist eine maximale Ableitung, deren Regelanwendungen alle linksmaximal (rechtsmaximal) sind
- Beispiel einer Grammatik  $G = (N, T, S, R)$   
 $N = \{S, B, C\}$   
 $T = \{a, b, c\}$   
 $R = \{S \rightarrow aSBC, S \rightarrow aBC, CB \rightarrow BC, aB \rightarrow ab, bB \rightarrow bb, bC \rightarrow bc, cC \rightarrow cc\}$

$S \Rightarrow aSBC \Rightarrow aaBCBC \Rightarrow aabCBC \Rightarrow aabBCC \Rightarrow aabbCC$   
 $\Rightarrow aabbcc \Rightarrow aabbcc = a^2b^2c^2$

$L(G) = \{a^n b^n c^n \mid n \geq 1\}$

Wieso?

## Chomsky Hierarchie

- Noam Chomsky hat Grammatiken – und die damit assoziierten Sprachen – in 4 Typen eingeteilt und es gilt  $\text{Typ } 3 \subset \text{Typ } 2 \subset \text{Typ } 1 \subset \text{Typ } 0$
- Typ 0: Phrasenstrukturgrammatiken  
Regeln  $LHS \rightarrow RHS$   
es gibt bezüglich LHS und RHS keine Einschränkungen
- Typ 1: kontextsensitive Grammatiken  
für alle Regeln  $LHS \rightarrow RHS$  gilt  $|LHS| \leq |RHS|$

Beispiel:

$R = \{S \rightarrow aSBC, S \rightarrow aBC, CB \rightarrow BC, aB \rightarrow ab, bB \rightarrow bb, bC \rightarrow bc, cC \rightarrow cc\}$

man beachte, dass die Nichtterminale  $B$  und  $C$  in LHS je nach Kontext durch ein anderes RHS ersetzt werden

erzeugte Sprache  $\{a^n b^n c^n \mid n \geq 1\}$  ist kontextsensitiv

## Chomsky Hierarchie

- Typ 2: kontextfreie Grammatiken  
für alle Regeln  $LHS \rightarrow RHS$  gilt  $|LHS| \leq |RHS|$  **und zusätzlich**  $LHS \in N$   
kontextfreie Regel  $N \rightarrow RHS$ : Nichtterminal  $N$  kann bedingungslos – in jedem Kontext – durch RHS ersetzt werden  
  
Beispiel:  
 $R = \{S \rightarrow ab, S \rightarrow aSb\}$   
erzeugte Sprache  $\{a^n b^n \mid n \geq 1\}$  ist kontextfrei  
  
Beispiel:  
Sprache ARIM der arithmetischen Ausdrücke ist kontextfrei
- Typ 3: reguläre Grammatiken  
für alle Regeln  $LHS \rightarrow RHS$  gilt  $|LHS| \leq |RHS|$  und  $LHS \in N$  **und zusätzlich**: RHS ist entweder ein einzelnes Terminalzeichen, oder ein Terminalzeichen gefolgt von einem Nichtterminalzeichen, oder ein Nichtterminalzeichen gefolgt von einem Terminalzeichen  
  
Beispiel:  
 $R = \{S \rightarrow a, S \rightarrow aS\}$  oder alternativ  $R = \{S \rightarrow a, S \rightarrow Sa\}$   
erzeugte Sprache  $\{a^n \mid n \geq 1\}$  ist regulär

## Chomsky Hierarchie

- Sprache  $L \subseteq T^*$  heisst vom Typ  $X$ , wenn eine Grammatik vom Typ  $X$  – **aber** keine vom Typ  $X+1$  – die Sprache  $L$  erzeugen kann  
  
(Eine Sprache vom Typ  $X$  kann natürlich auch durch Grammatiken vom Typ  $X-1$  erzeugt werden.)
- alle Sprachen vom Typ 1, 2, 3 sind entscheidbar, d.h. es gibt einen Algorithmus, der bei der Eingabe einer Grammatik  $G$  und eines Wortes  $w$  nach endlicher Zeit feststellt, ob  $w \in L(G)$  oder nicht
- Sprachen vom Typ 0 werden auch rekursiv aufzählbar genannt; sie sind semi-entscheidbar
- Typ 0 Grammatiken sind endliche Objekte, d.h. die Menge aller Typ 0 Grammatiken ist abzählbar, d.h. hat die Kardinalität von  $\mathbb{N}$ ; da jeder Typ 0 Sprache mindestens eine Typ 0 Grammatik zugeordnet werden kann, ist die Menge aller Typ 0 Sprachen ebenfalls abzählbar
- Menge aller Sprachen ist überabzählbar, hat die Kardinalität von  $\mathbb{R}$  – schon Potenzmenge von  $\{0, 1\}^*$  ist überabzählbar
- es gibt also Sprachen, die nicht durch Grammatiken beschrieben werden können

## Chomsky Hierarchie

Typ	Sprache	Grammatik	Maschine	Sprachzugehörigkeit
3	regulär	linkslinear rechtslinear  linksregulär rechtsregulär	endlicher Automat	entscheidbar
2	kontextfrei	kontextfrei  Chomsky  Greibach  reduziert	Keller-automat	entscheidbar
1	kontextsensitiv	Kontextsensitiv  Kuroda	linear beschränkte Turing-Maschine	entscheidbar
0	rekursiv aufzählbar	allgemein  separiert  normal	Turing-Maschine	semi-entscheidbar

## Reguläre Sprachen (Typ 3)

- reguläre Sprachen werden durch lineare und reguläre Grammatiken beschrieben
- linkslineare Grammatik
 

Jede Regel hat links genau ein Nichtterminal und rechts entweder genau ein Terminal oder genau ein Nichtterminal gefolgt von genau einem Terminal.

$$N \rightarrow T \quad N \rightarrow NT$$
- rechtslineare Grammatik
 

Jede Regel hat links genau ein Nichtterminal und rechts entweder genau ein Terminal oder genau ein Terminal gefolgt von genau einem Nichtterminal.

$$N \rightarrow T \quad N \rightarrow TN$$
- linksreguläre Grammatik
 

Jede Regel hat links genau ein Nichtterminal und rechts entweder genau ein Terminal oder genau ein Nichtterminal gefolgt von genau einem Terminal oder das leere Wort  $\epsilon$ .

$$N \rightarrow T \quad N \rightarrow NT \quad N \rightarrow \epsilon$$
- rechtsreguläre Grammatik
 

Jede Regel hat links genau ein Nichtterminal und rechts entweder genau ein Terminal oder genau ein Terminal gefolgt von genau einem Nichtterminal oder das leere Wort  $\epsilon$ .

$$N \rightarrow T \quad N \rightarrow TN \quad N \rightarrow \epsilon$$

## Chomsky Hierarchie

- in der praktischen Informatik spielen vor allem die regulären und die kontextfreien Sprachen eine Rolle
- viele Probleme sind allerdings kontextsensitiv; man versucht dann, mit kontextfreien Grammatiken zu arbeiten und die Kontextbedingungen durch nicht-grammatische Zusatzbedingungen zu behandeln

Beispiele:

Für Sprachen wie Pascal oder Modula werden kontextfreie Grammatiken verwendet, obwohl Typprüfungen, Kontrolle von Parametern in Prozeduraufrufen, Verwendung von vorher deklarierten Objekten etc. eigentlich kontextsensitiv sind; diese Kontrollen werden durch zusätzliche Algorithmen erledigt.

Bei der Verarbeitung natürlicher Sprache durch den Computer führt die Numerusunterscheidung – Singular oder Plural – zu Kontextsensitivität; man verwendet trotzdem kontextfreie Regeln, z. B. je einen Satz von Regeln für Singular und einen für Plural. Sogenannte Definite Clause Grammatiken erlauben eine kompaktere Darstellung.

## Reguläre Sprachen (Typ 3)

- für eine reguläre oder Typ 3 Sprache  $L \subseteq T^*$  sind die folgenden Aussagen äquivalent
  - $L \setminus \{\epsilon\}$  wird durch eine linkslineare Grammatik beschrieben.
  - $L \setminus \{\epsilon\}$  wird durch eine rechtslineare Grammatik beschrieben.
  - $L$  wird durch eine linksreguläre Grammatik beschrieben.
  - $L$  wird durch eine rechtsreguläre Grammatik beschrieben.
- Pumping Lemma
 

Eine Version des sogenannten Pumping Lemma wird verwendet, um nachzuweisen, dass Sprachen – z.B.  $L = \{a^n b^n \mid n \geq 1\}$  – **nicht** regulär sind.
- reguläre Sprachen sind unter Vereinigung, Durchschnitt, Konkatenation, Sternbildung, Komplement und Spiegelung abgeschlossen

## Automaten

- Automaten sind abstrakte Maschinen
- konkrete Maschinen in Hardware und Software – z.B. Biletautomaten, Computer, Menusteuerung eines Computers, Compiler – sind oft Realisierungen von Automaten
- Automaten haben endlich viele Zustände; es gibt ausgezeichnete Anfangs- und Endzustände
- zwischen Zuständen kann es Übergänge geben, die durch Eingaben – z.B. das Einlesen eines Buchstaben oder eines Wortes – ausgelöst werden
- liest der Automat ein Wort, dann geht er in einen neuen Zustand über, der vom vorherigen Zustand und vom eingelesenen Wort abhängt
- deterministischer Automat: Folgezustand ist eindeutig bestimmt
- nichtdeterministischer Automat: es kann mehrere Folgezustände geben
- ausserdem kann ein Automat einen Speicher haben
- Automaten werden in Klassen eingeteilt, die die Chomsky Hierarchie der Sprachen widerspiegelt
- jedem Sprachtyp kann eindeutig eine Automatenklasse zugeordnet werden und umgekehrt

## Endliche Automaten

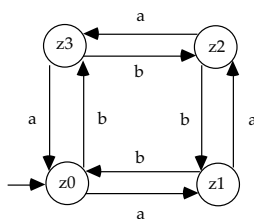
- endlicher Automat  $M$  ist ein Quadrupel  $M = (Z, A, z_0, t)$ 
  - $Z$  endliche Menge von Zuständen
  - $A$  endliches Eingabealphabet
  - $z_0 \in Z$  Anfangszustand
  - $t$  Zustandsübergangsfunktion  $Z \times A \rightarrow P(Z)$
- $t(z,a)$  ist die Menge aller Zustände, die der Automat  $M$  einnehmen kann, wenn er im Zustand  $z \in Z$  den Buchstaben  $a \in A$  liest
- wenn  $t(z,a)$  für jedes Paar  $(z,a)$  genau ein Element enthält, dann nennt man den endlichen Automaten **deterministisch**; in allen anderen Fällen, speziell wenn  $t(z,a)$  mehr als ein Element enthält, **nichtdeterministisch**
- Beispiel für deterministischen endlichen Automaten
  - $Z = \{z_0, z_1, z_2, z_3\}$
  - $A = \{a,b\}$
  - $t(z_0,a) = \{z_1\}$                        $t(z_2,a) = \{z_3\}$
  - $t(z_0,b) = \{z_3\}$                        $t(z_2,b) = \{z_1\}$
  - $t(z_1,a) = \{z_2\}$                        $t(z_3,a) = \{z_0\}$
  - $t(z_1,b) = \{z_0\}$                        $t(z_3,b) = \{z_2\}$

## Endliche Automaten

- Automatentafel des gleichen deterministischen endlichen Automaten

	a	b
$z_0$	$z_1$	$z_3$
$z_1$	$z_2$	$z_0$
$z_2$	$z_3$	$z_1$
$z_3$	$z_0$	$z_2$

- graphische Darstellung des gleichen deterministischen endlichen Automaten



## Endliche Automaten

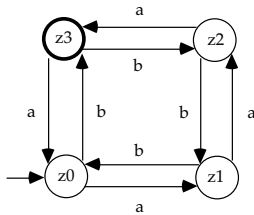
- akzeptierender endlicher Automat  $M$  ist ein Quintupel  $M = (Z, A, z_0, t, E)$ 
  - $(Z, A, z_0, t)$  endlicher Automat
  - $E \subseteq Z$  endliche Menge von Endzuständen
- Wort aus Eingabebuchstaben wird vom Automaten akzeptiert, wenn es zu diesem Wort eine Folge von Zustandsübergängen gibt, die im Anfangszustand beginnt und in einem Endzustand endet
- zu jedem akzeptierenden Automaten  $M$  gibt es eine Sprache  $L(M) \subseteq A^*$ , die  $M$  akzeptiert
- sei  $t^*$  als Funktion  $Z \times A^* \rightarrow P(Z)$  definiert durch
  - $t^*(z,\epsilon) = \{z\}$                                        $z \in Z$
  - $t^*(z,ax) = t^*(t(z,a),x)$                        $a \in A, x \in A^*$
 dann akzeptiert  $M$  die Sprache
 
$$L(M) = \{x \in A^* \mid t^*(z_0, x) \in E\}$$
- Jede durch einen (nicht-) deterministischen endlichen Automaten akzeptierte Sprache ist regulär (Typ 3).
- Jede reguläre Sprache wird durch einen (nicht-) deterministischen endlichen Automaten akzeptiert.

## Endliche Automaten

- Beispiel eines deterministischen akzeptierenden endlichen Automaten

$$\begin{aligned}
 Z &= \{z_0, z_1, z_2, z_3\} \\
 A &= \{a, b\} \\
 E &= \{z_3\} \\
 t(z_0, a) &= \{z_1\} & t(z_2, a) &= \{z_3\} \\
 t(z_0, b) &= \{z_3\} & t(z_2, b) &= \{z_1\} \\
 t(z_1, a) &= \{z_2\} & t(z_3, a) &= \{z_0\} \\
 t(z_1, b) &= \{z_0\} & t(z_3, b) &= \{z_2\}
 \end{aligned}$$

- graphische Darstellung des Automaten



- Beispielautomat landet nach den Worten b, bab, aaa, abb, bbbbbb, ... im Endzustand  $z_3$ , er "akzeptiert" diese Worte

## Reguläre Sprachen

- Konstruktion eines deterministischen endlichen Automaten  $M = (Z, A, S_M, t, E)$ , der die durch eine rechtslineare Grammatik  $G = (N, T, S_G, R)$  erzeugte reguläre Sprache  $L(G)$  akzeptiert

$$Z = N \cup \{\text{accepted}\} \cup \{\text{not accepted}\}$$

$$A = T$$

$$S_M = S_G$$

$$E = \{\text{accepted}\}$$

Grammatikregel von der Art  $N_1 \rightarrow TN_2$  definiert den Übergang  $t(N_1, T) = N_2$

Grammatikregel von der Art  $N_1 \rightarrow T$  definiert den Übergang  $t(N_1, T) = \text{accepted}$

für alle anderen Paare gilt  $t(N, T) = \text{not accepted}$

- Beispiel:

Grammatik  $S \rightarrow aS \mid aA, A \rightarrow bA \mid b$  über Alphabet  $\{a, b\}$

$Z = \{S, A, \text{accepted}, \text{not accepted}\}$

$S \rightarrow aS$  definiert  $t(S, a) = \{S\}$

$S \rightarrow aA$  definiert  $t(S, a) = \{A\}$

$A \rightarrow bA$  definiert  $t(A, b) = \{A\}$

$A \rightarrow b$  definiert  $t(A, b) = \{\text{accepted}\}$

$t(S, b) = t(A, a) = \{\text{not accepted}\}$

- analog konstruiert man eine reguläre Grammatik der von einem endlichen Automaten akzeptierten Sprache

## Minimale endliche Automaten

- zwei endliche Automaten heißen äquivalent, wenn sie dieselbe Sprache akzeptieren
- Konstruktion des einfachsten äquivalenten Automaten, der eine gegebene Sprache akzeptiert

*Determinierung:*

nichtdeterministischer endlicher Automat wird durch einen äquivalenten deterministischen ersetzt

*Vereinfachung:*

Elimination unerreichbarer Zustände

*Reduzierung:*

Elimination von funktionsgleichen Teilautomaten

- Determinierung: zu jedem nichtdeterministischen endlichen Automaten  $M = (Z, A, z_0, t, E)$  gibt es einen äquivalenten deterministischen  $M' = (Z', A', z_0', t', E')$ , der Potenzautomat genannt wird.

$$Z' = P(Z) \setminus \{\emptyset\}$$

$$A' = A$$

$$z_0' = \{z_0\}$$

$$t'(X) = \bigcup_{z \in X} t(z) \text{ mit } X \subseteq Z$$

$$E' = \{X \subseteq Z \mid X \cap E \neq \emptyset\}$$

## Minimale endliche Automaten

- Beispiel für Determinierung

	a	b	c
$z_1$	$z_1, z_3$	$z_1, z_2$	$z_3$
$z_2$	$z_1, z_2$	$z_3$	$z_1$
$z_3$	$z_2, z_3$	$z_2$	$z_2, z_3$

	a	b	c
$\{z_1\}$	$\{z_1, z_3\}$	$\{z_1, z_2\}$	$\{z_3\}$
$\{z_2\}$	$\{z_1, z_2\}$	$\{z_3\}$	$\{z_1\}$
$\{z_3\}$	$\{z_2, z_3\}$	$\{z_2\}$	$\{z_2, z_3\}$
$\{z_1, z_2\}$	$\{z_1, z_2, z_3\}$	$\{z_1, z_2, z_3\}$	$\{z_1, z_3\}$
$\{z_1, z_3\}$	$\{z_1, z_2, z_3\}$	$\{z_1, z_2\}$	$\{z_2, z_3\}$
$\{z_2, z_3\}$	$\{z_1, z_2, z_3\}$	$\{z_2, z_3\}$	$\{z_1, z_2, z_3\}$
$\{z_1, z_2, z_3\}$	$\{z_1, z_2, z_3\}$	$\{z_1, z_2, z_3\}$	$\{z_1, z_2, z_3\}$

## Minimale endliche Automaten

- Vereinfachung: zu jedem endlichen Automaten gibt es einen äquivalenten, aus dem die nicht erreichbaren Zustände entfernt wurden
- Beispiel:

	a	b	c
$z_0$	$z_1$	$z_1$	$z_2$
$z_1$	$z_2$	$z_2$	$z_2$
$z_2$	$z_0$	$z_0$	$z_0$
$z_3$	$z_1$	$z_2$	$z_3$

- Zustand  $z_3$  ist nicht erreichbar, d.h. es gibt keinen Übergang von einem der anderen Zustände  $z_0$  (=Anfangszustand),  $z_1, z_2$
- Streichen der letzten Zeile führt wieder zu einem sinnvollen Automaten

	a	b	c
$z_0$	$z_1$	$z_1$	$z_2$
$z_1$	$z_2$	$z_2$	$z_2$
$z_2$	$z_0$	$z_0$	$z_0$

## Minimale endliche Automaten

- Beispiel für Reduzierung

	a	b
$z_1$	$z_2$	$z_3$
$z_2$	$z_2$	$z_4$
$z_3$	$z_3$	$z_5$
$z_4$	$z_2$	$z_7$
$z_5$	$z_6$	$z_3$
$z_6$	$z_6$	$z_6$
$z_7$	$z_7$	$z_4$

- man erhält folgende Äquivalenzklassen  $K_1 = \{z_1\}, K_2 = \{z_2, z_4, z_7\}, K_3 = \{z_3, z_5, z_6\}$  und den reduzierten Automaten

	a	b
$K_1$	$K_2$	$K_3$
$K_2$	$K_2$	$K_2$
$K_3$	$K_3$	$K_3$

## Minimale endliche Automaten

- zwei Zustände  $z_1, z_2 \in Z$  eines endlichen Automaten  $(Z, A, z_0, t, E)$  heißen äquivalent, wenn die Automaten  $(Z, A, z_1, t, E)$  und  $(Z, A, z_2, t, E)$  äquivalent sind, d.h. die Teilautomaten, die in  $z_1$  bzw.  $z_2$  starten, akzeptieren die gleiche Sprache
- ein endlicher Automat heißt reduziert, wenn er keine äquivalenten Zustände enthält
- Zu jedem deterministischen endlichen Automaten gibt es einen äquivalenten, der reduziert ist und Quotientenautomat genannt wird. Die Konstruktion des Quotientenautomaten geschieht über die rekursive Definition von Äquivalenzklassen von Zuständen (Details in Cap).

## Minimale endliche Automaten

- Minimalautomaten: bis auf Isomorphie – d.h. Umbenennung der Zustände – gibt es zu jedem endlichen Automaten genau einen äquivalenten deterministischen Automaten, der einfach und reduziert ist. Der Minimalautomat kann durch Determinierung, Vereinfachung und Reduzierung bestimmt werden. Dieses Verfahren kann automatisiert werden.

## Endliche Automaten mit Ausgabefunktionen

- endliche Automaten können um Ausgabefunktionen zur Berechnung von Werten erweitert werden
- Moore Automaten  $M = (Z, E, z_0, t, A, w)$  haben eine Ausgabefunktion, die vom Zustand abhängt
 

$Z$	endliche Menge von Zuständen
$E$	endliches Eingabealphabet
$z_0 \in Z$	Anfangszustand
$t$	Zustandsübergangsfunktion $Z \times E \rightarrow Z$
$A$	endliches Ausgabealphabet
$w$	Ausgabefunktion: $Z \rightarrow A$
- Mealy Automaten  $M = (Z, E, z_0, t, A, w)$  haben eine Ausgabefunktion, die vom Zustand und vom zuletzt gelesenen Eingabewert abhängt
 

$Z$	endliche Menge von Zuständen
$E$	endliches Eingabealphabet
$z_0 \in Z$	Anfangszustand
$t$	Zustandsübergangsfunktion $Z \times E \rightarrow Z$
$A$	endliches Ausgabealphabet
$w$	Ausgabefunktion: $Z \times E \rightarrow A$
- Moore und Mealy Automaten wandeln eine Eingabe in eine gleich lange Ausgabe um
- beide Automaten sind gleichwertig

## Kontextfreie Sprachen (Typ 2)

- kontextfreie Sprachen werden durch kontextfreie, Chomsky Normalform, Greibach Normalform und reduzierte Grammatiken beschrieben
- kontextfreie Grammatik
  - Jede Regel hat links genau ein Nichtterminal.
  - $N \rightarrow R$
- Chomsky Normalform
  - Jede Regel hat links genau ein Nichtterminal und rechts entweder genau ein Terminal oder genau zwei Nichtterminale.
  - $N \rightarrow T \quad N \rightarrow NN$
- Greibach Normalform
  - Jede Regel hat links genau ein Nichtterminal und rechts entweder genau ein Terminal oder genau ein Terminal gefolgt von einem oder mehr Nichtterminalen.
  - $N \rightarrow T \quad N \rightarrow TN \quad N \rightarrow TNN \dots$
- reduzierte Grammatik
  - Jede Regel hat links genau ein Nichtterminal; wenn es sich nicht um das Startsymbol handelt, stehen rechts nur Terminale; jedes Nichtterminal taucht rechts in einer Regel auf, in der links das Startsymbol steht.

## Kontextfreie Sprachen (Typ 2)

- für eine kontextfreie oder Typ 2 Sprache  $L \subseteq T^*$  sind die folgenden Aussagen äquivalent
  - $L$  wird durch eine kontextfreie Grammatik beschrieben.
  - $L \setminus \{\epsilon\}$  wird durch eine Grammatik in Chomsky Normalform beschrieben.
  - $L \setminus \{\epsilon\}$  wird durch eine Grammatik in Greibach Normalform beschrieben.
  - $L$  wird durch eine reduzierte Grammatik beschrieben.
- Pumping Lemma
  - Eine Version des Pumping Lemma wird verwendet, um nachzuweisen, dass Sprachen – z.B.  $L = \{a^n b^n c^n \mid n \geq 1\}$  – **nicht** kontextfrei sind.
- kontextfreie Sprachen sind unter Vereinigung, Konkatenation und Sternbildung abgeschlossen

## Kellerautomaten

- Kellerspeicher (stack): abstrakter Datentyp über endlichem Kellularphabet  $K$ .
  - Keller wird durch Worte  $K^*$  repräsentiert.
  - Operationen
 

push:	$K^* \times K^* \rightarrow K^*$	push(w,k) = wk
pop:	$K^* \setminus \{\epsilon\} \rightarrow K^*$	pop( $k_0 k_1 \dots k_n$ ) = $k_1 \dots k_n$
top:	$K^* \setminus \{\epsilon\} \rightarrow K$	top( $k_0 k_1 \dots k_n$ ) = $k_0$
- Kellerautomat  $M = (Z, A, K, z_0, \#, E, t)$  ist ein endlicher Automat mit einem Kellerspeicher.
 

$Z$	endliche Menge von Zuständen
$A$	endliches Eingabealphabet
$K$	endliches Kellularphabet
$z_0 \in Z$	Anfangszustand
$\# \in K$	Anfangskellerzeichen
$E \subseteq Z$	endliche Menge von Endzuständen
$t$	Zustandsübergangsfunktion
	$Z \times (A \cup \{\epsilon\}) \times K \rightarrow P(Z \times K^*)$



## Kellerautomaten

- Anfang: Zustand  $z_0$ , Keller enthält nur #
- später: Zustand  $z$ , oberstes Kellerzeichen  $k$ , nächstes Eingabezeichen  $a$   
*Übergang mit Lesen eines Eingabezeichens:*  
 Menge  $t(z,a,k)$  ist nicht leer: Kellerautomat wählt ein Element  $(z', k')$  aus dieser Menge, geht in Zustand  $z'$ , entfernt  $k$  durch pop vom Keller, schreibt mit push  $k'$  auf den Keller  
*spontaner Übergang ohne Lesen eines Eingabezeichens:*  
 Menge  $t(z,\epsilon,k)$  ist nicht leer: Kellerautomat wählt ein Element  $(z', k')$  aus dieser Menge, geht in Zustand  $z'$ , entfernt  $k$  durch pop vom Keller, schreibt mit push  $k'$  auf den Keller
- deterministischer Kellerautomat: in jeder Situation gibt es nur genau einen Übergang  
 ..  
 entweder Übergang mit Lesen oder Übergang ohne Lesen  
 ..  
 Menge der möglichen Übergänge enthält nur ein Element

## Kellerautomaten

- Kellerautomat hält an, wenn ein Endzustand erreicht oder der Keller leer oder das ganze Eingabewort gelesen worden ist
- nichtleeres Wort aus Eingabebuchstaben wird vom Kellerautomaten akzeptiert, wenn er das ganze Wort liest und dann hält; es gibt drei gleichwertige Formen der Akzeptanz:  
 akzeptiert: Endzustand erreicht und Keller leer  
 zustandsakzeptiert: Endzustand erreicht  
 kellerakzeptiert: Keller leer (am praktischsten)
- sei  $k \in Z \times A^* \times K^*$  eine Konfiguration eines Kellerautomaten  $M$ , dann bewirkt eine Anwendung der Zustandsübergangsfunktion  $t$  einen Übergang von  $k$  zu  $k'$ , ausgedrückt als Relation  $k \Rightarrow k'$ . Sei  $\Rightarrow^*$  die reflexiv transitive Hülle von  $\Rightarrow$ .
- Kellerautomat  $M$  akzeptiert dann die Sprache  

$$L(M) = \{x \in A^* \mid (z_0, x, \#) \Rightarrow^* (z, \epsilon, \epsilon) \text{ für } z \in Z\}$$
- Jede durch einen nichtdeterministischen Kellerautomaten akzeptierte Sprache ist kontextfrei (Typ 2); jede kontextfreie Sprache wird durch einen nicht-deterministischen Kellerautomaten akzeptiert.

## Kellerautomaten

- Beispiel: Palindrom, geschachtelte Ausdrücke  
 Kellerautomat für  $L = \{a_1 a_2 \dots a_n \$ a_n \dots a_2 a_1 \mid a_i \in \{a,b\}\}$   
 $M = (\{z_0, z_1\}, \{a,b,\$, \#, A, B\}, z_0, \#, t)$   
 (Endzustand wird nicht gebraucht)  
 ..  
 Zustandsübergangsfunktion  $t$  als Übergänge zwischen Tupeln  $(Z, A, K) \rightarrow (Z', K')$  geschrieben  

$$\begin{array}{lll} z_0 a \# \rightarrow z_0 A \# & z_0 a A \rightarrow z_0 A A & z_0 a B \rightarrow z_0 A B \\ z_0 b \# \rightarrow z_0 B \# & z_0 b A \rightarrow z_0 B A & z_0 b B \rightarrow z_0 B B \\ z_0 \$ \# \rightarrow z_1 \# & z_0 \$ A \rightarrow z_1 A & z_0 \$ B \rightarrow z_1 B \\ z_1 a A \rightarrow z_1 \epsilon & z_1 b B \rightarrow z_1 \epsilon & z_1 \epsilon \# \rightarrow z_1 \epsilon \end{array}$$
  
 Wort  $ba\$ab \in L(M)$ , denn  
 $(z_0, ba\$ab, \#) \Rightarrow (z_0, a\$ab, B\#) \Rightarrow (z_0, \$ab, AB\#) \Rightarrow (z_1, ab, AB\#) \Rightarrow (z_1, b, B\#) \Rightarrow (z_1, \epsilon, \#) \Rightarrow (z_1, \epsilon, \epsilon)$

## Kellerautomaten

- Sprache der Palindrome über dem Alphabet  $\{a,b,\$\}$  wird durch die folgende kontextfreie Grammatik erzeugt  

$$S \rightarrow aSa \mid bSb \mid \$$$
  
 Beweis durch vollständige Induktion
- Wiederholung: Vollständige Induktion  
 $A: N_0 \rightarrow \{W, F\}$  eine Eigenschaft natürlicher Zahlen  
 $A(n_0)$  gilt (Induktionsverankerung)  
 Für beliebiges  $n$  gilt: falls  $A(n)$  gilt, dann gilt auch  $A(n+k)$  (Induktionsschritt)  
 dann gilt  $\forall n \in N_0 (A(n_0 + n^*k))$
- Ist  $w$  ein Palindrom, dann kann es durch die obige Grammatik erzeugt werden.  
 $A(n)$  sei "jedes Palindrom der Länge  $n$  kann durch die Grammatik erzeugt werden"  
 $A(1)$  gilt  
 Zu zeigen:  $\forall n \in N (A(n) \Rightarrow A(n+2))$   
 Sei  $w$  Palindrom der Länge  $n+2$ , dann muss es die Form  $ara$  oder  $brb$  haben, wobei  $r$  ein Palindrom der Länge  $n$  sein muss. Das kann aber laut Induktionsvoraussetzung  $A(n)$  erzeugt werden, also kann auch mit Hilfe der ersten beiden Grammatikregeln ein Palindrom der Länge  $n+2$  erzeugt werden.  
 Also kann jedes Palindrom der Länge  $\geq 1$  erzeugt werden.

## Kellerautomaten

- Wird  $w$  durch die obige Grammatik erzeugt, dann ist es ein Palindrom.

$A(n)$  sei "jede Ableitung der Länge  $n$  oder kürzer enthält nur Palindrome"

$A(1)$  gilt

Zu zeigen:  $\forall n \in \mathbb{N} (A(n) \Rightarrow A(n+1))$

Sei  $S \Rightarrow S_1 \dots \Rightarrow S_n \Rightarrow S_{n+1}$  eine Ableitung der Länge  $n+1$ . Laut Induktionsvoraussetzung  $A(n)$  enthält die Ableitung der Länge  $n \Rightarrow S_1 \dots \Rightarrow S_n$  nur Palindrome. Da auf  $S_n$  eine Grammatikregel anwendbar ist, muss es ein  $S$  enthalten. Dieses  $S$  muss in der Mitte stehen, da  $S_n$  ein Palindrom ist. Die Grammatikregeln ersetzen  $S$  durch  $aSa$ ,  $bSb$  oder  $\$$ . D.h.  $S_{n+1}$  ist wieder ein Palindrom.

D.h. jedes durch die Grammatik erzeugte Wort der Länge  $n \geq 1$  ist ein Palindrom.

## Kellerautomaten

- Kellerautomat  $M$  heisst deterministisch, wenn für alle  $z \in Z$ ,  $a \in A$  und  $k \in K$  gilt  $|t(z,a,k)| + |t(z,\epsilon,k)| \leq 1$  und der Kellerautomat zustandsakzeptiert
- deterministische Kellerautomaten definieren die deterministisch kontextfreien Sprachen, eine echte Teilmenge der kontextfreien Sprachen
- Sprache  $L = \{a_1 \dots a_n \$ a_n \dots a_1 \mid a_i \in A^*\}$  ist deterministisch kontextfrei
- Sprache  $L = \{a_1 \dots a_n a_n \dots a_1 \mid a_i \in A^*\}$  ist kontextfrei, aber nicht deterministisch kontextfrei
- deterministisch kontextfreie Sprachen werden auch LR(k) Sprachen genannt und spielen im Compilerbau eine grosse Rolle

## Syntaxbäume

- einer Ableitung eines Wortes  $w$  in einer Typ 2 oder 3 Grammatik  $G$  kann ein Syntaxbaum (Ableitungsbaum) zugeordnet werden

sei  $w \in L(G)$  und  $S = w_0 \Rightarrow w_1 \Rightarrow \dots \Rightarrow w_n = w$  eine Ableitung von  $w$

$S$  bildet die Wurzel des Syntaxbaums

falls im  $i$ -ten Ableitungsschritt  $w_{i-1} \Rightarrow w_i$  das Nichtterminal  $N$  durch ein Wort  $x$  ersetzt wird, dann erhält der Knoten  $N$  des Syntaxbaums  $|x|$  Töchter, die mit den einzelnen Zeichen von  $x$  beschriftet werden

Blätter des Baumes sind die Symbole von  $w$

## Beispiel

- Beispielgrammatik für ARIM

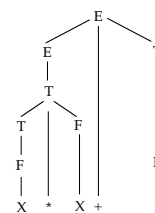
$E \rightarrow T$	(Expression ist Term)
$E \rightarrow E + T$	(Expression ist Expression + Term)
$T \rightarrow F$	(Term ist Faktor)
$T \rightarrow T * F$	(Term ist Term * Faktor)
$F \rightarrow X$	(Faktor ist Konstante oder Variable)
$F \rightarrow (E)$	(Faktor ist Expression in Klammern)

- Beispielableitungen

$$E \Rightarrow E + T \Rightarrow T + T \Rightarrow T * F + T \Rightarrow F * F + T \Rightarrow X * F + T \Rightarrow X * X + T \Rightarrow X * X + F \Rightarrow X * X + X$$

$$E \Rightarrow E + T \Rightarrow E + F \Rightarrow E + X \Rightarrow T + X \Rightarrow T * F + X \Rightarrow T * X + X \Rightarrow F * X + X \Rightarrow X * X + X$$

haben den gleichen Syntaxbaum



einmal links und einmal rechts abgeleitet

## Syntaxbäume

- verschiedenen Ableitungen kann der gleiche Syntaxbaum zugeordnet werden, der dann verschieden traversiert wird
- es gilt:  $w \in L(G) \Leftrightarrow$  es gibt eine Ableitung von  $w$ 
  - $\Leftrightarrow$  es gibt einen Syntaxbaum mit  $w$  an den Blättern
  - $\Leftrightarrow$  es gibt eine Linksableitung von  $w$
  - $\Leftrightarrow$  es gibt eine Rechtsableitung von  $w$

## Mehrdeutige Grammatiken

- eine Grammatik heisst mehrdeutig, wenn es für dasselbe Wort  $w$  verschieden strukturierte Syntaxbaume gibt; andernfalls heisst die Grammatik eindeutig

- Beispiel
  - $S \rightarrow aB$
  - $S \rightarrow Ac$
  - $A \rightarrow ab$
  - $B \rightarrow bc$

für das Wort  $abc$  gibt es zwei verschiedene Ableitungen mit verschiedenen Syntaxbäumen

$$S \Rightarrow Ac \Rightarrow abc \quad S \Rightarrow aB \Rightarrow abc$$

Mehrdeutigkeit kann in diesem Fall beseitigt werden, da eine eindeutige Grammatik existiert, die dieselbe Sprache generiert

$$S \rightarrow abc$$

- Sprache  $L$  heisst inhärent mehrdeutig, wenn jede Grammatik  $G$  mit  $L = L(G)$  mehrdeutig ist

$$\text{Beispiel } L = \{a^i b^j c^k \mid i=j \text{ oder } j=k\}$$

## (Extended) Backus Naur Form

- Backus und Naur entwickelten die Extended Backus Naur Form EBNF zur kompakten Darstellung von Typ 2 Grammatiken. (Eine einfachere Version ist als BNF bekannt.)
- Zusammenfassung von Regeln mit gleicher linker Seite
  - $A \rightarrow R_1$
  - $A \rightarrow R_2$
  - ...
  - $A \rightarrow R_n$
 zu
 
$$A \rightarrow R_1 \mid R_2 \mid \dots \mid R_n$$
- Zusammenfassung von Regeln mit optionalen Elementen (ein Wort  $O$  kann, aber muss nicht, auftauchen)
  - $A \rightarrow R_1 R_2$
  - $A \rightarrow R_1 O R_2$
 zu
 
$$A \rightarrow R_1 [O] R_2$$

## (Extended) Backus Naur Form

- Zusammenfassung von Regeln mit wiederholten Elementen (ein Wort  $W$  kann beliebig oft, auch 0 Mal, auftauchen)
  - $A \rightarrow R_1 R_2$
  - $A \rightarrow R_1 W R_2$
  - $W_R \rightarrow W$
  - $W_R \rightarrow W W_R$
 zu
 
$$A \rightarrow R_1 \{W\} R_2$$
- (E)BNF und kontextfreie Grammatiken sind gleichwertig, d.h. durch (E)BNF werden genau die kontextfreien Sprachen dargestellt

## Kontextsensitive Sprachen (Typ 1)

- kontextsensitive Sprachen werden durch kontextsensitive Grammatiken oder durch Kuroda Normalform Grammatiken beschrieben
- kontextsensitive Grammatik
  - keine Regel ist verkürzend, d.h. für jede Regel
  - $LHS \rightarrow RHS$  gilt  $|LHS| \leq |RHS|$
  - $RHS \neq \epsilon$
- Kuroda Normalform
  - jede Regel hat eine der vier Formen
  - $N \rightarrow T$     $N \rightarrow N$     $N \rightarrow NN$     $NN \rightarrow NN$
- kontextsensitive Sprachen sind unter Vereinigung, Durchschnitt, Konkatenation, Komplement und Sternbildung abgeschlossen

## Rekursiv aufzählbare Sprachen (Typ 0)

- rekursiv aufzählbare Sprachen werden durch allgemeine, separierte oder normale Grammatiken beschrieben
- allgemeine Grammatik
  - beliebige Regeln
- separierte Grammatik
  - jede Regel hat eine der Formen
  - $N \rightarrow T$
  - $N \rightarrow \epsilon$
  - $N_1 N_2 \dots N_n \rightarrow \epsilon$
  - $N_1 N_2 \dots N_n \rightarrow N_{n+1} N_{n+2} \dots N_{n+m}$
- normale Grammatik
  - jede Regel hat eine der Formen
  - Reduktionsregel:  $N \rightarrow \epsilon$
  - Terminierungsregel:  $N \rightarrow T$
  - Expansionsregel:  $N \rightarrow NN$
  - Doppelsubstitution:  $NN \rightarrow NN$

## Turingmaschinen

- Turingmaschinen sind endliche Automaten mit einem beliebig grossen, sequentiell zugreifbaren Speicher
- Turings Vorstellung: endlicher Automat mit einem unendlichen Speicherband, das aus einzelnen Feldern besteht, und einem Lese- und Schreibkopf, der sich auf dem Band bewegen kann. Die Felder enthalten Buchstaben des Bandalphabets. Zeichen, die sich unter dem Kopf befinden, können gelesen und verändert werden. Der Kopf kann sich um ein Feld nach rechts oder links bewegen oder an der gleichen Stelle bleiben.
- Turingmaschine  $M$  wird formal durch ein 7-Tupel beschrieben  $M=(Z, A, B, t, z_0, \#, E)$ 
  - $Z$  endliche Menge von Zuständen
  - $A$  endliches Eingabealphabet
  - $B$  endliches Bandalphabet,  $A \subseteq B$
  - $z_0 \in Z$  Anfangszustand
  - $\# \in B-A$  Leerzeichen
  - $E \subseteq Z$  endliche Menge von Endzuständen
  - $t$  Zustandsübergangsfunktion
  - $Z \times B \rightarrow P(Z \times B \times (L, R, N))$
  - $L$ =Bewegung nach links,  $R$ =Bewegung nach rechts,  $N$ =keine Bewegung

## Turingmaschinen

- Befindet sich  $M$  im Zustand  $z$  und ist unter dem Kopf der Buchstabe  $a$ , dann geht  $M$  im nächsten Schritt in einen Zustand  $z'$  über, schreibt anstelle von  $a$  einen Buchstaben  $b$  auf das Band und führt dann eine Bewegung  $x \in (L, R, N)$  aus, formal
  - $t(z,a)$  ist ein Element der Menge  $(z',b,x)$
- Konfiguration der Turingmaschine ist  $k \in B^* \times Z \times B^*$ 
  - $k=b_1 z b_2$
  - $b_1$  schon besuchter Teil des Bandes
  - $z$  aktueller Zustand
  - $b_2$  Rest des Bandes, Kopf steht auf erstem Zeichen von  $b_2$
- Anfangskonfiguration  $z_0 x$ : auf dem Band steht die Eingabe  $x \in A$  und die Turingmaschine ist im Zustand  $z_0$
- später: Konfiguration  $a_1 \dots a_m z b_1 \dots b_n$ 
  - Übergang ohne Bewegung des Kopfes:
    - $t(z, b_1) = (z', c, N)$ , neue Konfiguration  $a_1 \dots a_m z' c b_2 \dots b_n$
  - Übergang mit Bewegung nach rechts:
    - $t(z, b_1) = (z', c, R)$ , neue Konfiguration  $a_1 \dots a_m c z' b_2 \dots b_n$
  - Übergang mit Bewegung nach links:
    - $t(z, b_1) = (z', c, L)$ , neue Konfiguration  $a_1 \dots a_{m+1} z' a_m c b_2 \dots b_n$

## Turingmaschinen

- Spezialfälle

$n=1$  und Übergang mit Bewegung nach rechts:

$t(z, b_1) = (z', c, R)$ , neue Konfiguration  $a_1 \dots a_n c z' \#$

$m=0$  und Übergang mit Bewegung nach links:

$t(z, b_1) = (z', c, L)$ , neue Konfiguration  $z' \# c b_2 \dots b_n$

- Beispiel: Turingmaschine, die Eingabe  $x \in \{0,1\}^*$  als binäre Zahl interpretiert und eine 1 dazu addiert.

$M = (\{z_0, z_1, z_2, z_3\}, \{0,1\}, \{0, 1, \#\}, t, z_0, \#, \{z_3\})$

$t(z_0, 0) = (z_0, 0, R)$

$t(z_0, 1) = (z_0, 1, R)$

$t(z_0, \#) = (z_1, \#, L)$

$t(z_1, 0) = (z_1, 1, L)$

$t(z_1, 1) = (z_1, 0, L)$

$t(z_1, \#) = (z_2, 1, N)$

$t(z_2, 0) = (z_2, 0, L)$

$t(z_2, 1) = (z_2, 1, L)$

$t(z_2, \#) = (z_3, \#, R)$

startet M mit 101 auf dem Band, dann hält sie mit 110 auf dem Band und dem Kopf auf dem ersten Zeichen

$z_0 101 \Rightarrow z_0 101 \Rightarrow z_0 101 \Rightarrow z_0 101 z_0 \# \Rightarrow z_0 101 z_0 \# \Rightarrow z_0 101 z_0 \#$   
 $\Rightarrow z_2 110 \# \Rightarrow z_2 \# 110 \# \Rightarrow \# z_3 110 \#$

## Linear beschränkte Turingmaschinen

- Linear beschränkte Turingmaschinen verlassen nie den Teil des Bandes, auf dem die Eingabe steht.

- Eine nichtdeterministische Turingmaschine heißt linear beschränkt, wenn für alle  $a_1 \dots a_{n-1} a_n \in A^*$ , die rechts durch ein Kontrollzeichen  $c$  begrenzt sind und alle Konfigurationen  $b_1 z b_2$  mit  $z_0 a_1 \dots a_{n-1} a_n c \Rightarrow^* b_1 z b_2$  gilt  $|b_1 b_2| = n$ .

- Linear beschränkte Turingmaschine M akzeptiert Sprache

$L(M) = \{a_1 \dots a_{n-1} a_n \in A^* \mid z_0 a_1 \dots a_{n-1} a_n c \Rightarrow^* b_1 z b_2 \text{ für } b_1, b_2 \in B \text{ und } z \in E\}$

- Die durch linear beschränkte, nichtdeterministische Turingmaschinen akzeptierten Sprachen sind genau die kontextsensitiven Sprachen (Typ 1).