

Kontextsensitive und Typ 0-Sprachen

Prof. Dr. David Sabel

LFE Theoretische Informatik



Letzte Änderung der Folien: 29. Juni 2019

Erinnerung: Typ 1- und Typ 0-Sprachen

- Typ 1: $|\ell| \leq |r|$ für alle Produktionen $\ell \rightarrow r$
- Typ 1-Grammatik = kontextsensitive Grammatik
- aber (im Gegensatz zu Typ 2): $\ell \in (\Sigma \cup V)^+$
- Typ 0: alles erlaubt
- In manchen Büchern werden unsere Typ 1-Grammatik auch **monotone Grammatiken** genannt
- In manchen Büchern wird für kontextsensitive Grammatiken gefordert: Produktionen von der Form $\alpha_1 A \alpha_2 \rightarrow \alpha_1 \alpha_3 \alpha_2$ mit $\alpha_3 \neq \varepsilon$

Inhalt

- Typ 1- und Typ 0-Sprachen
- Maschinenmodelle, die dazu passen
- Turingmaschinen und linear beschränkte Turingmaschinen
- LBA-Probleme

Kuroda-Normalform für Typ 1-Sprachen

(benannt nach dem japanischen Linguisten Sige-Yuki Kuroda)

Definition

Eine Typ 1-Grammatik $G = (V, \Sigma, P, S)$ ist in **Kuroda-Normalform**, falls alle Produktionen in P einer der folgenden vier Formen entsprechen:

$$A \rightarrow a \quad A \rightarrow B \quad A \rightarrow BC \quad AB \rightarrow CD$$

wobei $a \in \Sigma$ und $A, B, C, D \in V$.

Bemerkung: Die Kuroda-Normalform „erweitert“ kontextfreie Grammatiken um Regeln der Form $AB \rightarrow CD$.

Satz

Sei L eine kontextsensitive Sprache mit $\epsilon \notin L$. Dann gibt es eine Grammatik in Kuroda-Normalform, die L erzeugt.

Beweis: Algorithmus 10 (nächste Folie) bewerkstelligt dies.

Eingabe: Eine Typ 1-Grammatik $G = (V, \Sigma, P, S)$ mit $\epsilon \notin L(G)$
Ausgabe: Eine Typ 1-Grammatik in Kuroda-Normalform die $L(G)$ erzeugt.
Beginn

```

für alle  $a \in \Sigma$  tue
  /* Führe neue Variable  $A_a$  für  $a$  ein, und ersetze Vorkommen von  $a$  durch das Nichtterminal */
   $G := (V \cup \{A_a\}, \Sigma, \{\ell[A_a/a] \rightarrow r[A_a/a] \mid \ell \rightarrow r \in P\}, S)$ ;
  /* Nun sind alle Regeln von der Form  $A \rightarrow a$  oder  $A_1 \cdots A_m \rightarrow B_1 \cdots B_n$  mit  $A_i, B_j \in V$  */
für alle  $A \rightarrow B_1 \cdots B_n \in P$  mit  $n > 2$  tue
  Seien  $C_1, \dots, C_{n-2}$  neue Variablen;
   $V := V \cup \{C_1, \dots, C_{n-2}\}$ ;
  /* Ersetze in  $P$  die Produktion  $A \rightarrow B_1 \cdots B_n$  durch neue Regeln */
   $P := (P \setminus \{A \rightarrow B_1 \cdots B_n\})$ 
     $\cup \{A \rightarrow B_1 C_1\} \cup \{C_i \rightarrow B_{i+1} C_{i+1} \mid i = 1, \dots, n-3\} \cup \{C_{n-2} \rightarrow B_{n-1} B_n\}$ ;
für alle  $A_1 \cdots A_m \rightarrow B_1 \cdots B_n \in P$  mit  $m > 2$  oder  $n > 2$  tue
  Seien  $D_2, \dots, D_{n-1}$  neue Variablen;
   $V := V \cup \{D_2, \dots, D_{n-1}\}$ ;
  /* Ersetze in  $P$  die Produktion  $A_1 \cdots A_m \rightarrow B_1 \cdots B_n$  durch neue Regeln */
   $P := (P \setminus \{A_1 \cdots A_m \rightarrow B_1 \cdots B_n\})$ 
     $\cup \{A_1 A_2 \rightarrow B_1 D_2\} \cup \{D_i A_{i+1} \rightarrow B_i D_{i+1} \mid i = 2, \dots, m-1\}$ 
     $\cup \{D_i \rightarrow B_i D_{i+1} \mid i = m, \dots, n-2\} \cup \{D_{n-1} \rightarrow B_{n-1} B_n\}$ 
  Gebe die so entstandene Grammatik aus;
  
```

Algorithmus 10: Herstellung der Kuroda-Normalform

Eingabe: Eine Typ 1-Grammatik $G = (V, \Sigma, P, S)$ mit $\epsilon \notin L(G)$
Ausgabe: Eine Typ 1-Grammatik in Kuroda-Normalform die $L(G)$ erzeugt.
Beginn

```

für alle  $a \in \Sigma$  tue
  /* Führe neue Variable  $A_a$  für  $a$  ein, und ersetze Vorkommen von  $a$  durch das Nichtterminal */
   $G := (V \cup \{A_a\}, \Sigma, \{\ell[A_a/a] \rightarrow r[A_a/a] \mid \ell \rightarrow r \in P\}, S)$ ;
  /* Nun sind alle Regeln von der Form  $A \rightarrow a$  oder  $A_1 \cdots A_m \rightarrow B_1 \cdots B_n$  mit  $A_i, B_j \in V$  */
für alle  $A \rightarrow B_1 \cdots B_n \in P$  mit  $n > 2$  tue
  Seien  $C_1, \dots, C_{n-2}$  neue Variablen;
   $V := V \cup \{C_1, \dots, C_{n-2}\}$ ;
  /* Ersetze in  $P$  die Produktion  $A \rightarrow B_1 \cdots B_n$  durch neue Regeln */
   $P := (P \setminus \{A \rightarrow B_1 \cdots B_n\})$ 
     $\cup \{A \rightarrow B_1 C_1\} \cup \{C_i \rightarrow B_{i+1} C_{i+1} \mid i = 1, \dots, n-3\} \cup$ 
für alle  $A_1 \cdots A_m \rightarrow B_1 \cdots B_n \in P$  mit  $m > 2$  oder  $n > 2$  tue
  Seien  $D_2, \dots, D_{n-1}$  neue Variablen;
   $V := V \cup \{D_2, \dots, D_{n-1}\}$ ;
  /* Ersetze in  $P$  die Produktion  $A_1 \cdots A_m \rightarrow B_1 \cdots B_n$  durch neue Regeln */
   $P := (P \setminus \{A_1 \cdots A_m \rightarrow B_1 \cdots B_n\})$ 
     $\cup \{A_1 A_2 \rightarrow B_1 D_2\} \cup \{D_i A_{i+1} \rightarrow B_i D_{i+1} \mid i = 2, \dots, m-1\}$ 
     $\cup \{D_i \rightarrow B_i D_{i+1} \mid i = m, \dots, n-2\} \cup \{D_{n-1} \rightarrow B_{n-1} B_n\}$ 
  Gebe die so entstandene Grammatik aus;
  
```

Entfernt alle $a \in \Sigma$ aus den Regeln bis auf neue $A \rightarrow a$ -Regeln

Effekt:
 Ableitung vorher:
 $x A_1 \cdots A_m y$
 $\Rightarrow x B_1 \cdots B_n y$
 Ableitung nachher:
 $x A_1 A_2 \cdots A_m y$
 $\Rightarrow x B_1 D_2 A_3 \cdots A_m y$
 $\Rightarrow x B_1 B_2 D_3 A_4 \cdots A_m y$
 $\Rightarrow \dots$
 $\Rightarrow x B_1 \cdots B_{m-2} D_{m-1} A_m y$
 $\Rightarrow x B_1 \cdots B_{m-1} D_m y$
 $\Rightarrow x B_1 \cdots B_{m-1} B_m D_{m+1} y$
 $\Rightarrow \dots$
 $\Rightarrow x B_1 \cdots B_{n-2} D_{n-1} y$
 $\Rightarrow x B_1 \cdots B_{n-2} B_{n-1} B_n y$

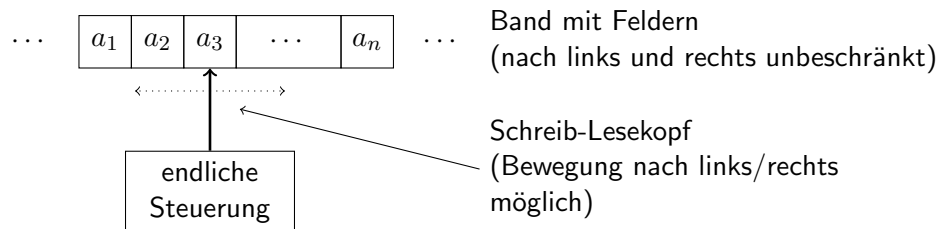
Motivation für Turingmaschinen

Einschränkungen der Kellerautomaten

- PDAs erkennen genau die CFLs, daher müssen Automaten für Typ 1- und Typ 2-Sprachen „mehr können“
- Wesentliche Beschränkung bei PDAs: Zugriff auf Speicher nur von oben möglich
- Z.B. kann man $\{a^i b^i c^i \mid i \in \mathbb{N}_{>0}\}$ nicht mit PDA erkennen, da man die Anzahl i
 - ... beim Lesen der a 's im Keller speichert;
 - ... beim Lesen der b 's vergleichen muss und das geht nur durch sukzessives Entnehmen aus dem Keller;
 - beim Lesen der c 's nicht mehr hat!

Mit beliebigem Lesen des Speichers wäre es kein Problem, $a^i b^i c^i$ zu erkennen.

Turingmaschine:Illustration



Zustandswechsel, informell

DTM: Ein Eintrag $\delta(z, a) = (z', b, x)$ bedeutet:

Falls die TM im Zustand z ist und das Zeichen a an der aktuellen Position des Schreib-Lesekopfs ist, dann

- Wechsele in Zustand z'
- Ersetze a durch b auf dem Band
- Falls $x = L$: Verschiebe den Schreib-Lesekopf ein Position nach links
- Falls $x = R$: Verschiebe den Schreib-Lesekopf ein Position nach rechts
- Falls $x = N$: Lasse Schreib-Lesekopf unverändert (Neutral)

NTM: $\delta(z, a)$ ist eine Menge solcher möglichen Schritte, und die NTM macht in einem Lauf irgendeinen davon (nichtdeterministisch)

Formale Definition der Turingmaschine

Definition (Turingmaschine)

Eine **Turingmaschine (TM)** ist ein 7-Tupel $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ mit

- Z ist eine endliche Menge von **Zuständen**,
- Σ ist das (endliche) **Eingabealphabet**,
- $\Gamma \supset \Sigma$ ist das (endliche) **Bandalphabet**,
- δ ist die **Zustandsüberföhrungsfunktion**
 - **deterministische TM (DTM):**
 $\delta : Z \times \Gamma \rightarrow Z \times \Gamma \times \{L, R, N\}$,
 - **nichtdeterministische TM (NTM):**
 $\delta : Z \times \Gamma \rightarrow \mathcal{P}(Z \times \Gamma \times \{L, R, N\})$
- $z_0 \in Z$ ist der **Startzustand**,
- $\square \in \Gamma \setminus \Sigma$ ist das **Blank-Symbol**
- $E \subseteq Z$ ist die Menge der **Endzustände**.

Konfigurationen

Definition (Konfiguration einer Turingmaschine)

Eine Konfiguration einer Turingmaschine ist ein Wort $k \in \Gamma^* Z \Gamma^*$

D.h. eine Konfiguration ist ein Wort $wz w'$, sodass:

- die TM ist im Zustand z ,
- auf dem Band steht $\dots \square \square w w' \square \square \dots$ und
- der Schreib-Lesekopf steht auf dem ersten Symbol von w'

Definition (Startkonfiguration einer TM)

Für ein Eingabewort w ist die Startkonfiguration einer TM $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ das Wort $z_0 w$.

Im Spezialfall $w = \varepsilon$ ist die Startkonfiguration $z_0 \square$

D.h. am Anfang steht der Kopf auf dem ersten Symbol der Eingabe.

Transitionsrelation einer TM

Definition (Transitionsrelation für Konfigurationen einer TM)

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ eine TM. Die Relation \vdash_M ist definiert durch (wobei $\delta(z, a) = (z', c, x)$ für eine NTM $(z', c, x) \in \delta(z, a)$ meint):

- $b_1 \cdots b_m z a_1 \cdots a_n \vdash_M b_1 \cdots b_m z' c a_2 \cdots a_n$,
wenn $\delta(z, a_1) = (z', c, N)$, $m \geq 0, n \geq 1, z \notin E$
- $b_1 \cdots b_m z a_1 \cdots a_n \vdash_M b_1 \cdots b_{m-1} z' b_m c a_2 \cdots a_n$,
wenn $\delta(z, a_1) = (z', c, L)$, $m \geq 1, n \geq 1, z \notin E$
- $b_1 \cdots b_m z a_1 \cdots a_n \vdash_M b_1 \cdots b_m c z' a_2 \cdots a_n$,
wenn $\delta(z, a_1) = (z', c, R)$, $m \geq 0, n \geq 2, z \notin E$
- $b_1 \cdots b_m z a_1 \vdash_M b_1 \cdots b_m c z' \square$,
wenn $\delta(z, a_1) = (z', c, R)$ und $m \geq 0, z \notin E$
- $z a_1 \cdots a_n \vdash_M z' \square c a_2 \cdots a_n$,
wenn $\delta(z, a_1) = (z', c, L)$ und $n \geq 1, z \notin E$

Akzeptierte Sprache einer TM

Definition (Akzeptierte Sprache einer TM)

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ eine TM.

Die von M **akzeptierte Sprache** $L(M)$ ist definiert als

$$L(M) := \{w \in \Sigma^* \mid \exists u, v \in \Gamma^*, z \in E : z_0 w \vdash_M^* u z v\}$$

Triviale Beispiele:

- Für Turingmaschinen der Form $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ mit $z_0 \in E$ gilt $L(M) = \Sigma^*$, denn diese Turingmaschinen akzeptieren jede Eingabe sofort.
- Für Turingmaschinen der Form $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, \emptyset)$ gilt $L(M) = \emptyset$, denn sie akzeptieren nie.

Transitionsrelation einer TM (2)

Weitere Notation dazu:

- \vdash_M^i : die i -fache Anwendung von \vdash_M
- \vdash_M^* die reflexiv-transitive Hülle von \vdash_M
- Wenn M klar ist, schreiben wir nur \vdash, \vdash^i , bzw. \vdash^* .

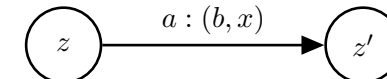
Bemerkung:

Wir nehmen an, dass die TM anhält, sobald sie einen Endzustand erreicht.

(Schöning-Buch erlaubt weiterrechnen)

Notation als Zustandsgraph

- Darstellung analog zu DFA / NFA / PDA
- Für $(z', b, x) \in \delta(z, a)$ zeichnen wir

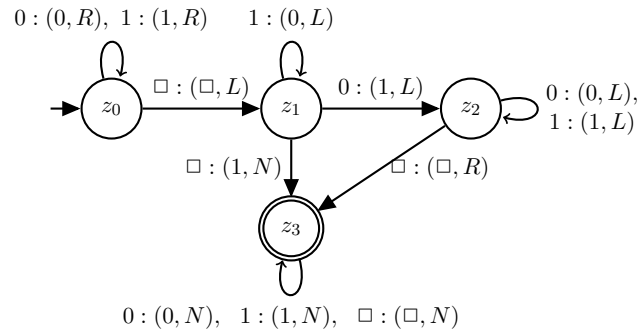


Beispiel (aus Schöning-Buch)

TM $M = (\{z_0, z_1, z_2, z_3\}, \{0, 1\}, \{0, 1, \square\}, \delta, z_0, \square, \{z_3\})$ mit

$\delta(z_0, 0) = (z_0, 0, R)$ $\delta(z_0, 1) = (z_0, 1, R)$ $\delta(z_0, \square) = (z_1, \square, L)$
 $\delta(z_1, 0) = (z_2, 1, L)$ $\delta(z_1, 1) = (z_1, 0, L)$ $\delta(z_1, \square) = (z_3, 1, N)$
 $\delta(z_2, 0) = (z_2, 0, L)$ $\delta(z_2, 1) = (z_2, 1, L)$ $\delta(z_2, \square) = (z_3, \square, R)$
 $\delta(z_3, 0) = (z_3, 0, N)$ $\delta(z_3, 1) = (z_3, 1, N)$ $\delta(z_3, \square) = (z_3, \square, N)$

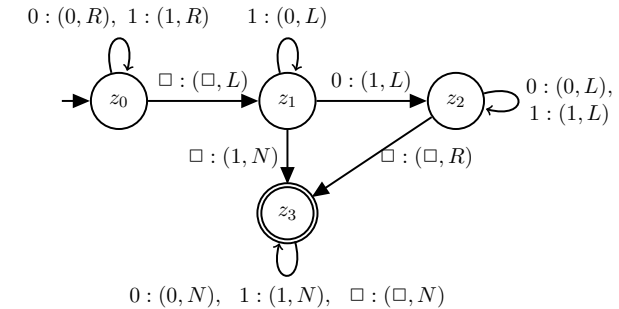
Zustandsgraph:



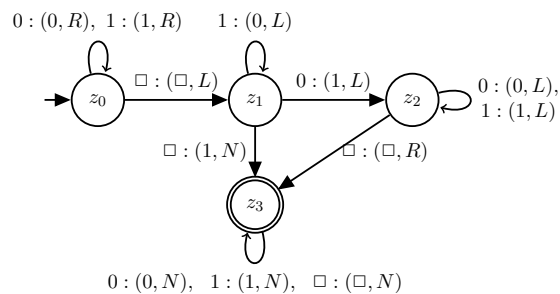
Beispiel (Forts.)

TM interpretiert Eingabe $w \in \{0, 1\}^*$ als Binärzahl und addiert 1:

- In z_0 wird das rechte Ende gesucht, dann in z_1 gewechselt
- In z_1 wird versucht 1 zur aktuellen Ziffer hinzu zu addieren:
Gelingt das ohne Übertrag, dann in z_2
Bei Übertrag: Weitermachen in z_1 und +1 zur nächsten Ziffer links
- In z_2 : bis zum Anfang links laufen, dann in z_3 .
- In z_3 wird akzeptiert.



Beispiellauf



$z_0 0011$
 $\vdash 0z_0 011$
 $\vdash 00z_0 11$
 $\vdash 001z_0 1$
 $\vdash 0011z_0 \square$
 $\vdash 001z_1 1 \square$
 $\vdash 00z_1 10 \square$
 $\vdash 0z_1 000 \square$
 $\vdash z_2 0100 \square$
 $\vdash z_2 0100 \square$
 $\vdash z_2 \square 0100 \square$
 $\vdash \square z_3 0100 \square$

LBAs: Spezielle Turingmaschinen

Ideen und Notationen:

- **Linear beschränkte Turingmaschinen:**
Schreib-Lesekopf darf den **Bereich der Eingabe** auf dem Band **nicht verlassen**
- Zum Erkennen des Endes:
Letztes Symbol der Eingabe wird markiert
- Kopie des Alphabets:
Für Alphabet $\Sigma = \{a_1, \dots, a_n\}$ bezeichne $\widehat{\Sigma} = \{\widehat{a}_1, \dots, \widehat{a}_n\}$.
- Eingabe bei LBAs: Statt $a_1 \cdots a_m$ nun $a_1 \cdots a_{m-1} \widehat{a}_m$
- TM arbeitet auf $\Sigma' = \Sigma \cup \widehat{\Sigma}$
- Linkes Ende muss die Maschine selbst markieren!

Definition (LBA)

Eine NTM $M = (Z, \Sigma \cup \widehat{\Sigma}, \Gamma, \delta, z_0, \square, E)$ heißt **linear beschränkt** (LBA, linear bounded automaton), wenn für alle $a_1 \cdots a_m \in \Sigma^+$ und **alle Konfigurationen** uzv mit $z_0 a_1 \cdots a_{m-1} \widehat{a}_m \vdash_M^* uzv$ gilt: $|uv| \leq m$.

Die **akzeptierte Sprache eines LBA** M ist

$$L(M) := \left\{ a_1 \cdots a_m \in \Sigma^* \mid \begin{array}{l} z_0 a_1 \cdots a_{m-1} \widehat{a}_m \vdash_M^* uzv, \\ \text{wobei } u, v \in \Gamma^* \text{ und } z \in E \end{array} \right\}$$

Beachte: LBAs sind NTMs

Kontextsensitive Sprachen durch LBAs erkennbar (2)

- Für Produktionen $A \rightarrow a$, $A \rightarrow B$, $AB \rightarrow CD$ kann man direkt ersetzen, für den Fall $A \rightarrow BC$ wird BC durch $\square A$ ersetzt und dann alle Zeichen von links um eins nach rechts verschoben.
- Akzeptiere, wenn Startsymbol S alleine auf dem Band steht.
- Nichtdeterminismus: Welche Produktion wird rückwärts angewendet und für welches Vorkommen einer rechten Seite.

Suche nach einer rechter Seite r :

- Beginne links an der Eingabe und laufe diese durch.
- Speichere im aktuellen Zustand: Symbol links vom Schreib-Lesekopf
- Entscheide mit dem aktuellen Symbol, ob es passende Produktion gibt (da rechte Seiten von Produktionen in Kuroda-Normalform aus maximal 2 Zeichen bestehen, genügt dies).

Satz

Jede kontextsensitive Sprache wird von einem LBA erkannt.

Beweis:

- Sprache sei als $G = (V, \Sigma, P, S)$ in Kuroda-Normalform gegeben.
- Konstruiere TM mit Bandalphabet $((\Sigma \cup V) \cup \widehat{\Sigma \cup V} \cup \square) \subseteq \Gamma$
- Zur einfacheren Illustration unterscheiden wir nicht zwischen a und \widehat{a} und schreiben a auch für den letzten Buchstaben der Eingabe, aber: Wir gehen davon aus, dass der LBA entsprechend programmiert ist, die notwendigen Ersetzungen zu machen.
- Die TM versucht nichtdeterministisch für $w \in \Sigma^*$ das Startsymbol S der Grammatik rückwärts herzuleiten, durch rückwärts Anwenden der Produktionen $\ell \rightarrow r \in P$: ersetze Vorkommen von r durch ℓ
- ...

Kontextsensitive Sprachen durch LBAs erkennbar (3)

Ersetzung r durch ℓ :

- Für $A \rightarrow a$ und $A \rightarrow B$ wird das aktuelle Symbol durch A ersetzt, anschließend wird der nächste Schritt gestartet (d.h. es gibt einen Zustand, der den Schreib-Lesekopf nach links fährt).
- Für $AB \rightarrow CD$, wird B geschrieben und der Kopf nach links wechseln, dann A geschrieben und der nächste Schritt gestartet.
- Für $A \rightarrow BC$ schreibe A und wechsele nach links, schreibe \square , fahre ganz nach links und starte Prozedur zum Verschieben der Zeichen nach rechts, solange bis die Lücke geschlossen ist.

Verschieben nach rechts:

- Zustand speichert das linkeste Symbol
- Laufen nach rechts: aktuelles Symbol mit dem gespeicherten vertauschen
- Vertauschen beenden nachdem \square mit einem anderen Symbol vertauscht wird.

Die TM ist ein LBA:

- Da für alle Produktionen $\ell \rightarrow r \in P$ gilt: $|\ell| \leq |r|$, werden nur Teilworte r durch gleichlange oder kürzere Teilworte ℓ ersetzt
- TM kommt mit dem Platz der Eingabe aus □

LBAs erkennen kontextsensitive Sprachen

Satz

Sei M ein LBA. Dann ist $L(M)$ eine kontextsensitive Sprache.

Beweis:

- Sei $M = (Z, \Sigma \cup \hat{\Sigma}, \Gamma, \delta, z_0, \square, E)$
- Wir konstruieren eine Typ 1-Grammatik G mit $L(G) = L(M)$
- Idee für die Grammatik:
 - 1 Erzeuge beliebiges $w \in \Sigma^*$ und Startkonfiguration von M für w
 - 2 Simuliere LBA zum Prüfen, ob $w \in L(M)$
 - 3 Wenn LBA akzeptiert erzeuge w endgültig
- Variablen der Grammatik:
 - Neue Variablen S und A
 - Variablen der Form $\langle \begin{smallmatrix} u \\ v \end{smallmatrix} \rangle$ wobei $u \in \Sigma$ und $v \in \Gamma \cup \{Z\}$

Obere Komponenten ergeben Wort w , untere Komponenten ergeben TM-Konfiguration.

Bemerkung 1:

- Konstruktion funktioniert auch für Grammatiken nicht in Kuroda-Normalform, ist aber komplizierter:
- Speichere im Zustand $q - 1$ Zeichen, wobei q die Länge der längsten rechten Seite
- Funktioniert immer noch mit endlich vielen Zuständen und als LBA

Bemerkung 2:

- Konstruktion funktioniert auch für Typ 0-Grammatiken: Platz allerdings dann unbeschränkt (kein LBA!)

Satz

Jede Typ i -Sprache (für $i=0,1,2,3$) wird von einer nichtdeterministischen Turing-Maschine akzeptiert.

LBAs erkennen kontextsensitive Sprachen (2)

- Regeln zur Erzeugung von $w \in \Sigma^*$ Startkonfiguration zw :

$$P_1 := \left\{ S \rightarrow A \left\langle \begin{smallmatrix} a \\ \hat{a} \end{smallmatrix} \right\rangle \mid a \in \Sigma \right\} \cup \left\{ A \rightarrow A \left\langle \begin{smallmatrix} a \\ a \end{smallmatrix} \right\rangle \mid a \in \Sigma \right\} \cup \left\{ A \rightarrow \left\langle \begin{smallmatrix} a \\ z_0 a \end{smallmatrix} \right\rangle \mid a \in \Sigma \right\}$$
- Worte $w \in L(M)$ mit $|w| < 2$ können dadurch nicht erzeugt werden, daher direkt alle Worte aus $L(M)$ der Länge < 2 erzeugen:

$$P_0 = \{ S \rightarrow w \mid |w| < 2, w \in L(M) \}$$

- Für $a_1 \cdots a_n \in \Sigma^*$ mit $n > 1$ gilt: $S \Rightarrow_{P_1} A \left\langle \begin{smallmatrix} a_n \\ \hat{a}_n \end{smallmatrix} \right\rangle \Rightarrow_{P_1}^* \left\langle \begin{smallmatrix} a_1 \\ z_0 a_1 \end{smallmatrix} \right\rangle \left\langle \begin{smallmatrix} a_2 \\ a_2 \end{smallmatrix} \right\rangle \cdots \left\langle \begin{smallmatrix} a_n \\ \hat{a}_n \end{smallmatrix} \right\rangle$
- Regelsatz P_2 simuliert M auf den unteren Komponenten. Wir bilden:

$$P_2 := \left\{ \left\langle \begin{smallmatrix} a \\ u \end{smallmatrix} \right\rangle \left\langle \begin{smallmatrix} b \\ v \end{smallmatrix} \right\rangle \rightarrow \left\langle \begin{smallmatrix} a \\ u' \end{smallmatrix} \right\rangle \left\langle \begin{smallmatrix} b \\ v' \end{smallmatrix} \right\rangle \mid a, b \in \Sigma \text{ und } uv \rightarrow u'v' \in P_2^{\text{unten}} \right\} \\ \cup \left\{ \left\langle \begin{smallmatrix} a \\ u \end{smallmatrix} \right\rangle \rightarrow \left\langle \begin{smallmatrix} a \\ u' \end{smallmatrix} \right\rangle \mid a \in \Sigma \text{ und } u \rightarrow u' \in P_2^{\text{unten}} (\text{mit } u, u' \in \Gamma \cup \{Z\}) \right\}$$

wobei wir P_2^{unten} noch definieren.

LBA's erkennen kontextsensitive Sprachen (3)

$$P_2^{\text{unten}} := \{cza \rightarrow z'cb \mid \text{für alle } c \in \Gamma \text{ und } (z', b, L) \in \delta(z, a)\} \\ \cup \{zac \rightarrow bz'c \mid \text{für alle } c \in \Gamma \text{ und } (z', b, R) \in \delta(z, a)\} \\ \cup \{za \rightarrow zb \mid \text{für alle } c \in \Gamma \text{ und } (z', b, N) \in \delta(z, a)\}$$

Es gilt:

- $wzw' \vdash_M^* uz'u'$ g.d.w. $wzw' \Rightarrow_{P_2^{\text{unten}}}^* uz'u'$
- Dabei: Darstellung von z, z' in der Ableitung immer verbunden mit einem Zeichen aus Γ

P_3 : Nach Akzeptieren des LBA, erstelle aus Tupelfolgen das Wort $a_1 \cdots a_n$

$$P_3 := \left\{ \left\langle \begin{matrix} b \\ za \end{matrix} \right\rangle \rightarrow b \mid z \in E, a \in \Gamma, b \in \Sigma \right\} \cup \left\{ \left\langle \begin{matrix} b \\ a \end{matrix} \right\rangle \rightarrow b \mid a \in \Gamma, b, b' \in \Sigma \right\}$$

$$\text{Es gilt } \left\langle \begin{matrix} a_1 \\ b_1 \end{matrix} \right\rangle \cdots \left\langle \begin{matrix} a_m \\ b_m \end{matrix} \right\rangle \left\langle \begin{matrix} a_{m+1} \\ zb_{m+1} \end{matrix} \right\rangle \left\langle \begin{matrix} a_{m+2} \\ b_{m+2} \end{matrix} \right\rangle \cdots \left\langle \begin{matrix} a_n \\ b_n \end{matrix} \right\rangle \Rightarrow_{P_3}^* a_1 \cdots a_n.$$

Satz von Kuroda

Theorem (Satz von Kuroda)

Kontextsensitive Sprachen werden genau von den LBA's erkannt.

LBA's erkennen kontextsensitive Sprachen (4)

- Sei $G = \left(\{S, A\} \cup \left\{ \left\langle \begin{matrix} u \\ v \end{matrix} \right\rangle \mid u \in \Sigma, v \in \Gamma \cup (Z\Gamma) \right\}, \Sigma, P_0 \cup P_1 \cup P_2 \cup P_3, S \right)$.
- Dann gilt für alle $w \in \Sigma^*$: $S \Rightarrow_G^* w$ genau dann, wenn $w \in L(M)$.
- Des weiteren gilt, dass G eine kontextsensitive Grammatik ist, da es keine verkürzenden Regeln gibt. \square

Typ 0-Sprachen

Die Konstruktion der Typ 1-Grammatik aus einem LBA kann für beliebige NTMs angepasst werden:

- Zusätzliche Tupel $\left\langle \begin{matrix} \$ \\ c \end{matrix} \right\rangle$ für $c \in \Gamma \cup Z\Gamma$ und $\$$ ein neues Symbol.
- Darstellung von Konfiguration, die länger als das Eingabewort sind: $\left\langle \begin{matrix} a_1 \\ c_1 \end{matrix} \right\rangle \cdots \left\langle \begin{matrix} a_n \\ c_n \end{matrix} \right\rangle \left\langle \begin{matrix} \$ \\ c_{n+1} \end{matrix} \right\rangle \cdots \left\langle \begin{matrix} \$ \\ z_i c_m \end{matrix} \right\rangle \left\langle \begin{matrix} \$ \\ c_r \end{matrix} \right\rangle$
- Regelsatz P_3 enthält Regeln $\left\langle \begin{matrix} \$ \\ c_i \end{matrix} \right\rangle \rightarrow \varepsilon$ (nicht kontextsensitiv!)

Satz

Die durch (allgemeine) nichtdeterministischen Turingmaschinen akzeptierten Sprachen sind genau die Typ 0-Sprachen.

- Nichtdeterministische Turingmaschinen können durch deterministische Turingmaschinen simuliert werden: Probiere alle Berechnungsmöglichkeiten der NTM nacheinander durch
- Daher gilt der letzte Satz auch für DTM
- Unterschied zwischen NTMs und DTMs kommt erst zum Tragen, wenn wir das Laufzeitverhalten betrachten (s. Kapitel zur Komplexitätstheorie)

Satz von Immerman und Szelepcsényi

Beweisskizze:

- Sei $G = (V, \Sigma, P, S)$ eine Typ 1-Grammatik mit $L(G) = L$.
- Konstruiere LBA M für $\bar{L} = \Sigma^* \setminus L$
- Sei $w \in \Sigma^*$. M berechnet die exakte Anzahl $a \in \mathbb{N}$ der von S aus erzeugbaren Satzformen der Länge $n \leq |w|$
- $a \leq (|V| + |\Sigma| + 1)^n$ und kann daher in $(k + 1) \cdot n$ -Bits dargestellt werden: Die passen auf das Band von M , wenn man Symbole für je $k + 1$ -Bitblöcke hat
- Anschließend: Zähle alle Satzformen u der Länge $\leq |n|$ aus $(V \cup \Sigma^*)$ auf (außer w selbst) und prüfe ob $S \Rightarrow_G^* u$ gilt
- Dabei wird ein Zähler mitgeführt, der hochgezählt wird, wenn Ableitung möglich ist
- Wenn der Zähler die Zahl a erreicht, dann akzeptiert M : Es wurden alle ableitbaren Worte der Länge $\leq n$ aufgezählt, w war nicht dabei. Also $w \notin L$ und damit $w \in \bar{L}$.

1. LBA-Problem

Erkennen deterministische LBAs die selben Sprachen wie nichtdeterministische LBAs?

Bis heute ungeklärt!

2. LBA-Problem

Sind die kontextsensitiven Sprachen abgeschlossen unter Komplementbildung?

Formuliert 1964 von Kuroda, 1987 gelöst von Neil Immerman als auch Róbert Szelepcsényi
Überraschenderweise positiv:

Theorem (Satz von Immerman und Szelepcsényi)

Die kontextsensitiven Sprachen sind abgeschlossen unter Komplementbildung.

Satz von Immerman und Szelepcsényi (2)

Berechnung der Zahl a :

- Sei $a(m, n)$ die Zahl der Satzformen, die in höchstens m Schritten aus S erzeugbar sind und deren Länge n nicht überschreitet:
$$a(m, n) = |\{w \in (V \cup \Sigma)^* \mid |w| \leq n, S \Rightarrow^{\leq m} w\}|.$$
- Wenn wir $a(i, n)$ für $i = 0, 1, 2, \dots$ berechnen, muss irgendwann $a(i, n) = a(i + 1, n)$ gelten, dann haben wir a gefunden.

Berechnung von $a(m, n)$

- Starte mit $a(0, n) = |\{S\}| = 1$
- Berechne $result = a(m + 1, n)$ durch Eingabe von $a(m, n)$
- Initial: $result = 0$.
- Äußere Schleife zählt alle Satzformen u bis zur Länge n auf
- Innere Schleife zählt nochmal alle Satzformen v bis zur Länge n auf.
- Vor Beginn der inneren Schleife: $count = 0$
- In der inneren Schleife: prüfe nichtdeterministisch, ob $S \Rightarrow^{\leq m} v$
Wenn ja $count = count + 1$;
Wenn $v = u$ oder $v \Rightarrow u$ gilt, $result = result + 1$
- Nach Ablauf der inneren Schleife, prüfe ob $count = a(m, n)$ gilt.
Wenn nein, dann verwerfe diese nichtdeterministische Berechnung
Wenn ja, dann war dies die richtige nichtdeterministische Berechnung
und es wurde für alle in $\leq m$ -Schritten aus S herleitbaren
Satzformen v geprüft, ob durch Verlängern mit $=$ oder \Rightarrow eine der
Satzformen u herleitbar ist d. h. $result$ enthält den Wert $a(m + 1, n)$.

Zusammenfassung

- Typ 1-Sprachen: Kuroda-Normalform
- Turingmaschinen (DTM und NTM)
- linear beschränkte NTMs (LBAs)
- Satz von Kuroda: LBAs erkennen genau die Typ 1-Sprachen
- TMs erkennen genau die Typ 0-Sprachen
- LBA-Probleme