Der Satz von Kuroda und LBA-Probleme

Prof. Dr. Jasmin Blanchette

Lehr- und Forschungseinheit für
Theoretische Informatik

Stand: 13. Juni 2023
Folien ursprünglich von PD Dr. David Sabel

Satz von Kuroda

Theorem (Satz von Kuroda)

Kontextsensitive Sprachen werden genau von den LBAs erkannt.

Der Beweis erfolgt in zwei Teilen:

- Jede kontextsensitive Sprache wird von einem LBA erkannt.
- LBAs erkennen kontextsensitive Sprachen.

Kontextsensitive Sprachen durch LBAs erkennbar (1)

Satz

Jede kontextsensitive Sprache wird von einem LBA erkannt.

Kontextsensitive Sprachen durch LBAs erkennbar (1)

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 $(\Sigma \cup V) \cup \widehat{\Sigma \cup V} \cup \square \subseteq \Gamma$.
- Der Einfachheit halber 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 $\ell \to r \in P$: Ersetze Vorkommen von r durch ℓ.

Kontextsensitive Sprachen durch LBAs erkennbar (2)

- ...
- Für Produktionen $A \to a$, $A \to B$ und $AB \to CD$ kann man direkt ersetzen, für den Fall $A \to BC$ 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?

Kontextsensitive Sprachen durch LBAs erkennbar (2)

- ...
- Für Produktionen $A \to a$, $A \to B$ und $AB \to CD$ kann man direkt ersetzen, für den Fall $A \to BC$ 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.
- Stelle mit dem aktuellen Symbol fest, ob es passende Produktion gibt (da rechte Seiten von Produktionen in Kuroda-Normalform aus maximal 2 Zeichen bestehen, genügt dies).

Kontextsensitive Sprachen durch LBAs erkennbar (3)

Ersetzung r durch ℓ:

- Für $A \to a$ und $A \to B$ 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 $AB \to CD$, wird B geschrieben und der Kopf nach links wechseln, dann A geschrieben und der nächste Schritt gestartet.
- Für $A \to BC$ 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.

Kontextsensitive Sprachen durch LBAs erkennbar (4)

Die TM ist ein LBA:

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

Bemerkungen und Typ 0-Grammatiken

Bemerkung 1:

- Die Idee der Konstruktion funktioniert auch für Typ 1-Grammatiken nicht in Kuroda-Normalform, ist aber komplizierter:
  Speichere im Zustand $q - 1$ Zeichen, wobei q die Länge der längsten rechten Seite.

Bemerkung 2:

- Die Idee der Konstruktion funktioniert auch für Typ 0-Grammatiken: Platz allerdings dann unbeschränkt (kein LBA).

Bemerkungen und Typ 0-Grammatiken

Bemerkung 1:

- Die Idee der Konstruktion funktioniert auch für Typ 1-Grammatiken nicht in Kuroda-Normalform, ist aber komplizierter:
  Speichere im Zustand $q - 1$ Zeichen, wobei q die Länge der längsten rechten Seite.

Bemerkung 2:

- Die Idee der 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 Turingmaschine akzeptiert.

LBAs erkennen kontextsensitive Sprachen

Satz

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

LBAs erkennen kontextsensitive Sprachen

Satz

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

Beweis:

- Sei $M = (Z, \Sigma \cup \widehat{\Sigma}, \Gamma, \delta, z_0, \square, E)$.
- 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:
  - Variablen S und A
  - Variablen der Form $\left\langle \begin{matrix} u \\ v \end{matrix} \right\rangle$ wobei $u \in \Sigma$ und $v \in \Gamma \cup Z\Gamma$

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

LBAs erkennen kontextsensitive Sprachen (2)

1. Erzeuge beliebiges $w \in \Sigma^*$ und Startkonfiguration von M für w.

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

$$P_1 := \left\{ S \to A \left\langle \begin{matrix} a \\ \widehat{a} \end{matrix} \right\rangle \mid a \in \Sigma \right\} \cup \left\{ A \to A \left\langle \begin{matrix} a \\ a \end{matrix} \right\rangle \mid a \in \Sigma \right\} \cup \left\{ A \to \left\langle \begin{matrix} a \\ z_0 a \end{matrix} \right\rangle \mid a \in \Sigma \right\}$$

- Wörter $w \in L(M)$ mit $|w| < 2$ können dadurch nicht erzeugt werden, daher direkt alle Wörter aus $L(M)$ der Länge < 2 erzeugen:

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

- Für $a_1 \cdots a_n \in \Sigma^*$ mit $n > 1$ gilt: $S \Rightarrow_{P_1} A \left\langle \begin{matrix} a_n \\ \widehat{a}_n \end{matrix} \right\rangle \Rightarrow^*_{P_1} \left\langle \begin{matrix} a_1 \\ z_0 a_1 \end{matrix} \right\rangle \left\langle \begin{matrix} a_2 \\ a_2 \end{matrix} \right\rangle \cdots \left\langle \begin{matrix} a_n \\ \widehat{a}_n \end{matrix} \right\rangle$.

LBAs erkennen kontextsensitive Sprachen (3)

2. Simuliere LBA zum Prüfen, ob $w \in L(M)$.

- Regelmenge P_2 simuliert M auf **den unteren** Komponenten. Wir bilden

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

wobei

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

Es gilt: $wzw' \vdash_M^* uz'u'$ g.d.w. $wzw' \Rightarrow_{P_2^{\text{unten}}}^* uz'u'$

LBAs erkennen kontextsensitive Sprachen (4)

3. Wenn LBA akzeptiert, erzeuge w endgültig.

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

$$P_3 := \left\{ \left\langle \begin{matrix} b \\ za \end{matrix} \right\rangle \to b \;\middle|\; z \in E, a \in \Gamma, b \in \Sigma \right\} \cup \left\{ \left\langle \begin{matrix} b \\ a \end{matrix} \right\rangle \to b \;\middle|\; a \in \Gamma, b \in \Sigma \right\}$$

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

LBAs erkennen kontextsensitive Sprachen (5)

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

Typ 0-Sprachen

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

- 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:

$$\left\langle \begin{matrix} a_1 \\ c_1 \end{matrix} \right\rangle \dots \left\langle \begin{matrix} a_n \\ c_n \end{matrix} \right\rangle \left\langle \begin{matrix} \$ \\ c_{n+1} \end{matrix} \right\rangle \dots \left\langle \begin{matrix} \$ \\ Z_i c_m \end{matrix} \right\rangle \left\langle \begin{matrix} \$ \\ c_r \end{matrix} \right\rangle$$

- Regelmenge P_3 enthält Regeln $\left\langle \begin{matrix} \$ \\ c_i \end{matrix} \right\rangle \to \varepsilon$ (**nicht Typ 1**)

Satz

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

LBA-Probleme

1. LBA-Problem

Erkennen deterministische LBAs dieselben Sprachen wie nichtdeterministische LBAs?

Bis heute ungeklärt

LBA-Probleme

1. LBA-Problem

Erkennen deterministische LBAs dieselben Sprachen wie nichtdeterministische LBAs?

Bis heute ungeklärt

2. LBA-Problem

Sind die kontextsensitiven Sprachen abgeschlossen unter Komplementbildung?

Formuliert 1964 von Kuroda, 1988 gelöst von
Neil Immerman als auch Róbert Szelepcsényi (Gödel-Preis 1995)
Überraschenderweise positiv:

Theorem (Satz von Immerman und Szelepcsényi)

Die kontextsensitiven Sprachen sind abgeschlossen unter Komplementbildung.

Lösung zum 2. LBA-Problem (1)

Naiver Beweis:

- Sei $G = (V, \Sigma, P, S)$ eine Typ 1-Grammatik mit $L(G) = L$.
- Konstruiere LBA M für $\overline{L} = \Sigma^* \setminus L$.
  - Sei $w \in \Sigma^*$.
  - Prüfe, ob $S \Rightarrow_G^* w$ gilt.
  - Wenn ja, dann verwirf.
  - Wenn nein, dann akzeptiere.

Lösung zum 2. LBA-Problem (1)

Naiver Beweis:

- Sei $G = (V, \Sigma, P, S)$ eine Typ 1-Grammatik mit $L(G) = L$.
- Konstruiere LBA M für $\overline{L} = \Sigma^* \setminus L$.
  - Sei $w \in \Sigma^*$.
  - Prüfe, ob $S \Rightarrow_G^* w$ gilt.
  - Wenn ja, dann verwirf.
  - Wenn nein, dann akzeptiere.

Problem: Die Berechnung von $S \Rightarrow_G^* w$ ist **nichtdeterministisch**.
Der Test $S \Rightarrow_G^* w$ kann fehlschlagen, auch wenn $S \Rightarrow_G^* w$ gilt.

Lösung zum 2. LBA-Problem (2)

Beweisskizze:

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

Lösung zum 2. LBA-Problem (3)

Berechnung der Zahl A:

- Sei A_m die Zahl der Satzformen, die in höchstens m Schritten aus S erzeugbar sind und deren Länge n nicht überschreitet:

$$A_m = \left|\{w \in (V \cup \Sigma)^* \mid |w| \leq n, S \Rightarrow^{\leq m} w\}\right|$$

- Wenn wir A_i für $i = 0, 1, 2, \ldots$ berechnen, muss irgendwann $A_i = A_{i+1}$ gelten, dann haben wir A gefunden.

Berechnung von A_m

Starte mit $A_0 = |\{S\}| = 1$.
Berechne A_{m+1} durch Eingabe von A_m.

1. Initial: $A_{m+1} := 0$.
2. Äußere Schleife zählt alle Satzformen u bis zur Länge n auf:
   - 2.1 $count := 0$.
   - 2.2 Innere Schleife zählt nochmal alle Satzformen v bis zur Länge n auf:
     - 2.2.1 Prüfe nichtdeterministisch, ob $S \Rightarrow^{\leq m} v$.
     - 2.2.2 Wenn ja, $count := count + 1$.
     - 2.2.3 Wenn $v = u$ oder $v \Rightarrow u$ gilt, $A_{m+1} := A_{m+1} + 1$ und setze die Iteration von u fort.
   - 2.3 Wenn $count \neq A_m$, dann verwirf diese nichtdeterministische Berechnung.
   - 2.4 Wenn $count = A_m$, dann war dies eine richtige nichtdeterministische Berechnung und es wurde für alle in $\leq m$ Schritten aus S ableitbaren Satzformen v geprüft, ob durch Verlängern mit $=$ oder $\Rightarrow$ eine der Satzformen u ableitbar ist.
3. Gib A_{m+1} aus. □