

19. KONTEXTFREIE SPRACHEN II

Abschlusseigenschaften, Maschinencharakterisierung, Komplexität

ABSCHLUSSEIGENSCHAFTEN

ABSCHLUSSEIGENSCHAFTEN VON KF

KF ist abgeschlossen gegen

- Vereinigung
- Verkettung
- Iteration (*-Operator)
- homomorphe Bilder

KF ist nicht abgeschlossen gegen

- Durchschnitt
- Komplement

ABSCHLUSS GEGEN VEREINIGUNG, VERKETTUNG UND ITERATION

Dies zeigt man wie bei den kontext-sensitiven Sprachen.

ABSCHLUSS GEGEN HOMOMORPHE BILDER

Zu zeigen: $L \in \text{KF}$, $h : \Sigma^* \rightarrow T^*$ Homomorphismus $\Rightarrow h(L) \in \text{KF}$

Für $G = (N, \Sigma, P, S)$ mit $L(G) = L$ in Chomsky-Normalform erzeugt $G' = (N, T, P', S)$ die Sprache $h(L)$, wobei P' aus P durch Ersetzung der Substitutionsregeln wie folgt entsteht:

$$X \rightarrow a \longrightarrow X \rightarrow h(a)$$

FEHLENDER ABSCHLUSS GEGEN DURCHSCHNITT

Wie wir bereits gezeigt haben, gilt:

- $\{0^m 1^m 0^n : m, n \geq 0\}, \{0^m 1^n 0^n : m, n \geq 0\} \in \text{KF}$
- $\{0^n 1^n 0^n : n \geq 0\} = \{0^m 1^m 0^n : m, n \geq 0\} \cap \{0^m 1^n 0^n : m, n \geq 0\} \notin \text{KF}$

FEHLENDER ABSCHLUSS GEGEN KOMPLEMENT

- $\{0^n 1^n 0^n : n \geq 0\} \notin \text{KF}$ (s.o.)
- $\overline{\{0^n 1^n 0^n : n \geq 0\}} \in \text{KF}$ (Übung)

ABSCHLUSSEIGENSCHAFTEN VON LIN

LIN ist abgeschlossen gegen

- Vereinigung
- homomorphe Bilder

KF ist nicht abgeschlossen gegen

- Durchschnitt
- Komplement
- Verkettung
- Iteration (*-Operator)

ABSCHLUSS GEGEN VEREINIGUNG UND HOMOMORPHE BILDER

Dies zeigt man wie bei den kontextfreien Sprachen.

FEHLENDER ABSCHLUSS GEGEN DURCHSCHNITT UND KOMPLEMENT

Man kann dieselben Gegenbeispiele wie bei KF benutzen

FEHLENDER ABSCHLUSS GEGEN VERKETTUNG UND ITERATION

- $L = \{0^n 1^n : n \geq 0\} \in \text{LIN}$ (bereits gezeigt)
- $LL \notin \text{LIN}$ (bereits gezeigt) und $L^* \notin \text{LIN}$ (zeigt man analog)

ÜBERSICHT ABSCHLUSSEIGENSCHAFTEN

	\cup	\cap	Komplement	Verkettung	Iteration	hom. Bild
CH	+	+	-	+	+	+
REK	+	+	+	+	+	-
KS	+	+	+	+	+	-
KF	+	-	-	+	+	+
LIN	+	-	-	-	-	+

MASCHINENCHARAKTERISIERUNG

KELLERAUTOMATEN (PUSH DOWN AUTOMATA)

Ein Kellerautomat M zur Erkennung einer Sprache $L = L(M)$ über dem Eingabealphabet Σ besitzt als Speicher einen Stapel (Stack, Keller). Dieser kann ein Wort w über dem sog. Kellularphabet $\Gamma = \Gamma_- \cup \{b_0\}$ aufnehmen. Dieses beginnt stets mit dem ausgezeichneten Bottomelement b_0 gefolgt von einem möglicherweise leeren Wort v , das b_0 nicht enthält: $w = b_0v$, $v \in \Gamma_-^*$. Der Automat kann dabei immer nur auf das rechte Wortende von w zugreifen (*last-in-first-out*-Speicher).

In einem Schritt liest M den nächsten noch nicht gelesenen Buchstaben a der Eingabe x und den letzten Buchstaben c des Kellerwortes w (und löscht beide von der Eingabe bzw. im Keller). In Abhängigkeit von den gelesenen Buchstaben und dem aktuellen (Programm-)Zustand fügt M ein (möglicherweise leeres) Wort $u \in \Gamma_-^*$ an das Kellerwort hinzu (war $c = b_0$ so wird ein Wort $u = b_0u'$ mit $u' \in \Gamma_-^*$ hinzugefügt). Alternativ kann M auch einen spontanen oder λ -Übergang durchführen, bei dem kein weiterer Buchstabe der Eingabe gelesen wird.

Wir erlauben hier auch *nichtdeterministische* Maschinen, d.h. in einzelnen Schritten mag die Maschine aus verschiedenen möglichen Übergängen einen Übergang auswählen. Z.B. erlauben wir der Maschine wahlweise den nächsten Buchstaben einzulesen oder einen λ -Übergang zu machen.

FORMALE DEFINITION

Ein (nichtdeterministischer) Kellerautomat (Push-Down-Automat; (N)PDA) M wird durch

$$M = (\Sigma, \Gamma, b_0, S, s_0, \delta, F)$$

definiert wobei

- Σ Eingabealphabet
- Γ Kelleralphabet
- $b_0 \in \Gamma$ das Bottomelement
- S endliche Menge der (Programm-)Zustände
- s_0 Startzustand
- δ endliche Übergangsrelation (Programm)
 $\delta \subseteq S \times \Sigma \cup \{\lambda\} \times \Gamma \times \Gamma^* \times S$
- $F \subseteq S$ Menge der (akzeptierenden) Endzustände

Lässt sich δ als partielle Funktion

$$\delta : S \times \Sigma \cup \{\lambda\} \times \Gamma \rightarrow \Gamma^* \times S$$

schreiben, so ist M *deterministisch*.

ARBEITSWEISE VON M

Eine M -Konfiguration ist ein Tripel (s, y, v) , wobei s der aktuelle Zustand, y der noch nicht gelesene Teil der Eingabe und v das aktuelle Kellerwort ist.

Gilt $(s, y, v) = (s, ay', v'c)$, so lässt sich die Instruktion (s, a, c, u, s') anwenden und überführt (s, y, v) in $(s', y', v'u)$.

Jede mögliche Rechnung bei Eingabe x beginnt mit der Konfiguration (s_0, x, b_0) und ist ansonsten wie üblich definiert.

Die Akzeptanz von x wird auf zwei unterschiedliche Arten definiert:

- M kann x vollständig einlesen und befindet sich nach dem Einlesen von x in einem Endzustand (= Akzeptanz via Endzustände).
- M kann x vollständig einlesen und nach dem Einlesen von x ist der Keller leer, d.h. enthält nur das Bottomelement. (= Akzeptanz via leerem Keller; hier kann man auf die Endzustände verzichten).

(Im nichtdeterministischen Fall ist dies so zu lesen, dass dies für eine mögliche Rechnung gilt.)

BEISPIEL 1. Ein Kellerautomat M zur Erkennung der Sprache $L = \{0^n 1^n : n \geq 0\}$ via leerem Keller kommt mit folgenden Instruktionen aus:

S	$\Sigma \cup \{\lambda\}$	Γ	Γ^*	S
s_0	0	c	$c0$	s_0
s_0	1	0	λ	s_1
s_1	1	0	λ	s_1

M liest im Startzustand s_0 den ersten aus Nullen bestehenden Wortteil und speichert die gelesenen Nullen im Keller. Kommt M zu der ersten 1 geht M in den Zustand s_1 und entfernt für jede gelesene 1 eine 0 aus dem Keller. Hat die Eingabe x nicht die Gestalt $x = 0^m 1^n$, so kann M die Eingabe nicht komplett lesen, verwirft also. Dasselbe gilt für $x = 0^m 1^n$ mit $m < n$. Im Falle von $m \geq n$ wird die Eingabe komplett gelesen. Der Keller ist danach genau dann leer, wenn $m = n$.

Der Automat ist deterministisch.

Folgende Variante M' von M erkennt $L = \{0^n 1^n : n \geq 0\}$ via Endzustände, wobei $F = \{s_0, s_3\}$:

S	$\Sigma \cup \{\lambda\}$	Γ	Γ^*	S
s_0	0	c	$c0$	s_1
s_1	0	c	$c0$	s_1
s_1	1	0	λ	s_2
s_2	1	0	λ	s_2
s_2	λ	b_0	b_0	s_3

Der Automat M' ist nichtdeterministisch, da er im Zustand s_2 die Wahl zwischen einem eigentlichen und einem λ -Übergang hat.

BEISPIEL 2. Ein Kellerautomat M zur Erkennung der Sprache $L = \{ww^R : w \in \{0,1\}^*\}$ via leerem Keller kommt mit folgenden Instruktionen aus:

S	$\Sigma \cup \{\lambda\}$	Γ	Γ^*	S
s_0	i	c	ci	s_0
s_0	i	i	λ	s_1
s_1	i	i	λ	s_1

M speichert im Startzustand s_0 die gelesenen Bits $i = 0,1$ im Keller, rät die Wortmitte, und vergleicht dann im Zustand s_1 das Restwort mit dem (durch die last-in-first-out-Speicherung gespiegelt) gespeicherten Anfangsteil.

M ist nichtdeterministisch.

Man kann zeigen, dass die Sprache L von keinem deterministischen Kellerautomaten erkannt wird.

SATZ. Folgende Aussagen sind äquivalent:

- L kontextfrei
- $L \cup \{\lambda\}$ wird von einem nd. Kellerautomaten via leerem Keller erkannt.
- L wird von einem nd. Kellerautomaten via Endzuständen erkannt.

Beweis s. Vorlesung *Formale Sprachen*.

Im deterministischen Fall, fallen Akzeptanz via leerem Keller und Akzeptanz via Endzuständen nicht zusammen, sondern ersteres Akzeptanzkriterium ist mächtiger.

Eine Sprache heisst *deterministisch kontextfrei*, wenn sie von einem det. Kellerautomat via leerem Keller erkannt wird.

Beispiele:

- $L = \{0^n 1^n : n \geq 0\}$ ist deterministisch kontextfrei.
- $L = \{ww^R : w \in \{0, 1\}^*\}$ ist kontextfrei aber nicht det. kontextfrei.

KOMPLEXITÄT UND ENTSCHEIDBARKEIT

KOMPLEXITÄT KONTEXTFREIER SPRACHEN

SATZ VON COCKE, KASAMI UND YOUNGER. Das Wortproblem für kontextfreie Grammatiken $G = (N, T, P, S)$ in Chomsky-Normalform ist deterministisch in Zeit n^3 lösbar. D.h.

$$\{(G, w) : w \in L(G)\} \in \text{DTIME}(n^3)$$

D.h. insbesondere, dass es zu jeder kf. Grammatik G in Chomsky-Normalform eine Mehrband-TM M mit $L(G) = L(M)$ gibt, deren Laufzeit kubisch beschränkt ist:

$$\forall x \in T^* \quad (\text{time}_M(x) \leq |x|^3)$$

Im Gegensatz zu *KS* (wo diese Frage offen ist) können wir als das Wortproblem für eine gegebene kontextfreie Grammatik *tatsächlich* entscheiden.

SYNTAXANALYSE

In der Praxis genügt es meist nicht, für ein Wort x zu entscheiden, ob es von einer kontextfreien Grammatik G erzeugt wird, sondern man möchte im positiven Fall eine Herleitung erhalten (und damit Information über die Struktur von x) bzw. eine qualifizierte Fehlermeldung. Diese Aufgabe bezeichnet man als *Syntaxanalyse*. Bedeutend ist diese vor allem bei der Übersetzung von Programmiersprachen (Compilerbau).

Hier hat man sehr effiziente Analyseverfahren für spezielle kontextfreie Grammatiken entwickelt, die einen Herleitungsbaum Top-Down bzw. Bottom-Up erzeugen (LL(k)- bzw. LR(k)-Grammatiken).

Diese Thematik wird ausführlich in der Vorlesung FORMALE SPRACHEN und COMPILERBAU behandelt.

ENTSCHEIDBARKEITSFragen FÜR KF

Folgende Probleme sind für KF entscheidbar:

- Wortproblem ($x \in L(G)$?)
- Leerheitsproblem ($L(G) = \emptyset$?)
- (Un-)Endlichkeitsproblem ($L(G)$ (un)endlich ?)

(Dies haben wir bereits gezeigt.)

Folgende Probleme sind für KF nicht entscheidbar:

- Totalitätsproblem ($L(G) = T^*$?)
- Durchschnittsproblem ($L(G_1) \cap L(G_2)$ kontextfrei?)
- Äquivalenzproblem ($L(G_1) = L(G_2)$)

(Dies wird in der Vorlesung FORMALE SPRACHEN gezeigt.)