Fragen

Thomas Schimper

<2017-10-23 Mo>

1 Was ist eine Sprache?

Eine Menge Σ heißt Alphabet, falls gilt: $|\Sigma|$ und $\Sigma \neq \emptyset$. Eine Sprache L über Σ ist eine Teilmenge des von der Menge Σ frei erzeugten Monoids ($L \subseteq \Sigma^*$).

2 Reguläre Sprache

Eine Sprache $L \subseteq \Sigma^*$ heißt regulär $\Leftrightarrow$ es gibt ein endliches Monoid $(M, \circ, e)$ einen Monoidmorphismus $h : (\Sigma^*, \cdot, \lambda) \to (M, \circ, e)$, sowie eine endliche Menge $F \subseteq M$ gibt mit:

$$L = \{w \in \Sigma^* | h(w) \in F\} \tag{1}$$

3 DEA & NEA

3.1 DEA

$$A = (Q, \Sigma, \delta, q_0, F) \tag{2}$$
$$Q : \text{endliche Menge von Zuständen} \tag{3}$$
$$\Sigma : \text{Eingabezeichen} \tag{4}$$
$$\delta : Q \times \Sigma \to Q \text{ totale Überführungsfunktion} \tag{5}$$
$$q_0 \in Q : \text{Anfangszustand} \tag{6}$$
$$F \subseteq Q : \text{Endzustände} \tag{7}$$

3.1.1 Konfigurationen

Ein Element aus $C = Q \times \Sigma^*$ heißt Konfiguration von A. Definiere eine Binärrelation $\vdash_A$ auf C durch

$$(q, w) \vdash_A (q', w') \Leftrightarrow a \in \Sigma : w = aw' \text{ und } q' = \delta(q, a) \tag{8}$$

- $\vdash_A^n$ beschreibt n Schritte von A"
- $L(A) = \{w \in \Sigma^* | \exists q \in F : (q_0, w) \vdash_A^* (q, \lambda)\}$

Es sei $A = (Q, \Sigma, \delta, q_0, F)$ ein DEA

$$\hat{\delta} : Q \times \Sigma^* \to Q, (q, w) \mapsto \begin{cases} q, & w = \lambda \\ \hat{\delta}(\delta(q, a), w'), & w = aw', a \in \Sigma \end{cases} \tag{9}$$

3.2 NEA

Ein nichtdeterministischer Endlicher Automat oder NEA wird beschrieben durch ein Quintupel

$$A = (Q, \Sigma, \delta, Q_0, F) \tag{10}$$
$$Q : \text{ endliche Menge von Zuständen} \tag{11}$$
$$\Sigma : \text{ endliche Menge von Eingabezeichen} \tag{12}$$
$$\delta \subseteq Q \times \Sigma \times Q : \text{ Überführungsrelation} \tag{13}$$
$$Q_0 \subseteq Q : \text{ Anfangszustände} \tag{14}$$
$$F \subseteq Q : \text{ Endzustände} \tag{15}$$

3.2.1 Konfigurationen

Ein Element aus $C = Q \times \Sigma^*$ heißt Konfiguration von A. Definiere eine Binärrelation $\vdash_A$ auf C durch

$$(q, w) \vdash_A (q', w') \Leftrightarrow a \in \Sigma : w = aw' \text{ und } \delta \ni (q, a, q') \tag{16}$$

$\vdash_A^n$ n Schritte von A

$$L(A) = \{w \in \Sigma^* | q_0 \in Q_0, q \in F : (q_0, w) \vdash_A^* (q, \lambda)\} \tag{17}$$

4 Schlingen-Lemma

Es sei $A = (Q, \Sigma, \delta, q_0, F)$ ein DEA. Es sei $q \in Q$ und sei $X = \{a \in \Sigma : (q, a) \vdash_A (q, \lambda)\}$. Dann gilt: $X^* \subseteq \{w \in \Sigma^* | (q, w) \vdash_A^* (q, \lambda)\}$

5 Pumping-Lemma

Zu jeder regulären Sprache L gibt es eine Zahl $n > 0$, sodass jedes Wort $w \in L$ mit $\ell(w) \geq n$ als Konkatenation $w = xyz$ dargestellt werden kann mit geeigneten x, y, z mit folgenden Eigenschaften:

1. $\ell(y) > 0$
2. $\ell(xy) \leq n$
3. $\forall i \geq 0 : xy^i z \in L$

Wichtig: Die Umkehrung gilt nicht!

6 Reguläre Ausdrücke

1. $\emptyset$ und a sind RA (über Σ) für jedes $a \in \Sigma$
2. Ist R ein RA (über Σ), so auch $(R)*$.
3. Sind R_1 und R_2 RAs (über Σ), so auch $R_1 R_2$ und $(R_1 \cup R_2)$
4. $L(\emptyset) = \emptyset; L(a) = \{a\}$
5. Ist R ein RA, setzte $L((R)*) = (L(R))^*$
6. Sind R_1 und R_2 RA, setze $L(R_1 R_2) = L(R_1) \cdot L(R_2)$ und $L((R_1 \cup R_2)) = L(R_1) \cup L(R_2)$
7. Jede RA-Sprache ist regulär
8. Jede reguläre Sprache ist durch einen RA beschreibbar

7 Potenzen in Kleene-Sternen

Ist $(M, \circ, e)$ ein Monoid, so können wir induktiv die n -te Potenz eines Elementes $x \in M$ rekursiv festlegen durch : $x^0 = e$ sowie $x^{n+1} = x^n \circ x$ für $n \in \mathbb{N}$ Dann kann man $A^+ = \bigcup_{n \geq 1} A^n$ und $A^* = \bigcup_{n \geq 0} A^n$ Somit ist auch L^+ und L^* (*Kleene-Stern*) für $L \subseteq \Sigma^*$ definiert.

8 Myhill-Nerode

Für jede Sprache $L \subseteq \Sigma^*$ ist $\equiv_L$ eine ÄR

- $\equiv_L$ heißt auch Myhill-Nerode Äquivalenz.
- Eine Sprache $L \subseteq \Sigma^*$ ist genau dann regulär, wenn es nur endlich viele ÄK bezüglich $\equiv_L$

9 Hilfsbegriffe

- U heißt Teilwort von $x \in \Sigma^* \Leftrightarrow x \in \Sigma^*\{u\}\Sigma^*$
- u heißt Präfix von $x \in \Sigma^* \Leftrightarrow x \in \{u\}\Sigma^*$
- u heißt Suffix oder Endwort von $x \in \Sigma^* \Leftrightarrow x \in \Sigma^* u$
- Ein Teilwort/Präfix/Suffix u von x heißt echt gdw. $\ell(u) < \ell(x)$
- Ein echtes Teilwort u von x, das sowohl Präfix als auch Suffix von x ist, heißt Rand (der Breite $\ell(u)$) von x

10 kfG

- Σ ist das Terminalalphabet
- N ist das Nonterminalalphabet mit $N \cap \Sigma = \emptyset$
- $R \subset N \times (\Sigma \cup N)^*$ ist das Alphabet der Regeln
- $S \in N$ ist das Startsymbol oder Anfangszeichen

Ein Wort über dem Gesamtalphabet $(\Sigma \cup N)$ heißt auch Satzform

11 Doner, Thatcher, Wright

Es sei $L \subseteq \Sigma^*$ eine Sprache. L ist kontextfrei gdw. $L = yield(B(A))$ für einen endlichen Baumautomaten A.

12 kfG in CNF

ein Quadrupel $G = (\Sigma, N, R, S)$:

- Σ ist das Terminalalphabet
- N ist das Nonterminalalphabet
- $R \subset (N \times N^2) \cup (N \times \Sigma)$ ist das Alphabet der Regeln oder Produktionen.
- $S \in N$ ist das Startsymbol oder Anfangszeichen

13 CNF

Zu jeder kfg G gibt es eine kfG G' in Chomsky-Normalform mit $L(G) \setminus \{\lambda\} = L(G')$.

14 CYK

Ist eine kfG G in CNF fixiert, so lässt sich die Frage "$w \in L(G)$in einer Zeit beantworten, die sich durch ein kubisches Polynom in $\ell(w)$ abschätzen lässt.

15 Binärer Wurzelbaum

ist gegeben durch ein Tripel $B = (V, \phi, r)$ mit ausgezeichneter Wurzel $r \in V$ und einer Vater-Abbildung $\phi : V \setminus \{r\} \to V$ mit der Eigenschaft

$$\forall v \in V : \#\{u \in V | \phi(u) = v\} \leq 2 \tag{18}$$

15.0.2 Lemma

Der Ableitungsbaum eines jeden von einer kontextfreien Grammatik in CNF akzeptierten Wortes kann als binärer Wurzelbaum aufgefasst werden.

15.0.3 Höhe des Baums

ist gegeben durch

$$h(B) = max_{v \in V \setminus \{r\}}\{k \in \mathbb{N} | \phi^k(v) = r\} \tag{19}$$

16 Pumping-Lemma für KF

Zu jeder kfS L gibt es eine Konstante $n > 0$, sodass jedes Wort $w \in L$ mit $\ell(w) \geq n$ als Konkatenation $w = uvxyz$ dargestellt werden kann mit geeigneten u, v, x, y, z mit folgenden Eigenschaften

- $\ell(v) > 0$ oder $\ell(y) > 0$
- $\ell(vxy) \leq n$
- $\forall i \geq 0 : uv^i xy^i z \in L$

17 Chomsky-Hierarchie

Grammatik	Regeln	Sprachen	Entscheidbarkeit	Automanten	Abgeschlossenheit
Typ-0 Beliebig formale Grammatik	$\alpha \to B$	rekursiv aufzählbar	-	Turingmaschine	$\circ, \cap \cup, *$
Typ-1 Kontext-sensitive Grammatik	$\alpha A \beta \to \alpha \gamma \beta$	kontextsensitiv	Wortproblem	linear platz-beschränkte nichtdeterministische Turingmaschine	$\complement, \circ, \cap, \cup, *$
Typ-2 Kontextfreie Grammatik	$A \to \gamma$	kontextfrei	Wortproblem, Leerheitsproblem, Endlichkeitsproblem	nichtdeterministischer Kellerautomat	$\circ, \cup, *$
Typ-3 Reguläre Grammatik	$A \to aB$	regulär	Wortproblem, Leerheitsproblem, Endlichkeitsproblem, Äquivalenzproblem	Endlicher Automat	$\complement, \circ, \cap, \cup, *$

17.0.4 Satz

$REG \subsetneq KF \subsetneq KS \subsetneq RA$

18 Turing Machine

Eine Turingmaschine ist durch ein 7-Tupel beschrieben:

$$TM = (S, E, A, \delta, s_0, \square, F) \tag{20}$$

Dabei bedeuten:

- $S = \{s_0, s_1, \cdots, s_n\}$ ist die Menge der Zustände
- $E = \{e_1, e_2, \cdots, e_n\}$ ist das endliche Eingabealphabet
- $A = \{a_0, a_1, \cdots, a_m\}$ das endliche Arbeitsalphabet (auch Bandalphabet genannt), es sei dabei $E \subset A$.
- s_0 der Startzustand
- $a_0 = \square$ das Blank-Symbol, das zwar dem Arbeitsalphabet , aber nicht dem Eingabealphabet angehört
- $F \subseteq S$ die Menge der Endzustände
- δ sei die Überführungsfunktion/-relation

$$\delta : (S \setminus F) \times A \to S \times A \times \{L, R, N\} \quad \text{deterministischer Fall} \tag{21}$$
$$\delta \subseteq ((S \setminus F) \times A) \times (S \times A \times \{L, R, N\}) \quad \text{nichtdeterministische Fall} \tag{22}$$

18.0.5 Konfiguration

Eine Konfiguration einer Turingmaschine $TM = (S, E, A, \delta, s_0, \square, F)$ ist ein Tripel (u, s, v) aus $A^* \times S \times A^+$:

- uv ist aktuelle Bandinschrift
- s ist der aktuelle Zustand
- Schreib-Lesekopf über erstem Zeichen von v, daher $v \neq \varepsilon$
- Start der Maschine: $v \in E^* \cup \{\square\}$ (Eingabe), $s = s_0, u = \varepsilon$

18.0.6 Initialkonfiguraiton, Finalkonfiguration, akzeptierte Sprache

- Anfangskonfiguration beim Start der TM mit Eingabe $w \in E^*$ ist $s_0 w$ (bzw. $s_0\square$, falls $w = e$)
- Endkonfiguration sind alle Konfigurationen $us_f v$ mit $s_f \in F$. Hier kan die Berechnung nicht fortgesetzt werden.
- Weiter ist

$$L(TM) := \{w \in E^* | s_o w \vdash^* us_f v, s_f \in F, u, v \in A^*\} \tag{23}$$

die von der Turingmaschine akzeptierte Sprache

19 These von Church

Turingmaschinen können 'alles', was überhaupt jemals von Computern"gemacht werden kann.

20 LBA

Eine TM heißt linear beschränkter Automat (`LBA`), wenn sie keine Blankzeichen überschreiben darf.

21 Kellerautomat

Ein Kellerautomat (Pushdown Automaton (PDA)) ist ein Sextupel

$$A = (Q, \Sigma, \Gamma, q_0, \Delta, F) \tag{24}$$

- Q ist das Zustandsalphabet
- Σ ist das Eingabealphabet
- Γ ist das Kelleralphabet
- $q_0 \in Q$ ist der Startzustand
- $\Delta \subset (Q \times (\Sigma \cup \{\lambda\} \times \Gamma^*) \times (Q \times \Gamma^*)$ ist die endliche Übergangsrelation
- $F \subseteq Q$ ist die Endzustandsmenge

22 Compiler-Aufbau

- Scanner: lexikalische Analyse
- Parser: syntaktische Analyse
- semantische Analyse
- Codegenerierung
- Optimierung

23 Greibach-NF

Eine kfG $G = (\Sigma, N, R, S)$ ist in Greibach-Normalform gdw.

$$R \subseteq (N \times \Sigma(N \setminus \{S\})^*) \cup (S \times \{\lambda\}) \tag{25}$$

Jede kontextfreie Sprache besitzt eine sie erzeugende kfG in Greibach-NF.

Eine kfG $G = (\Sigma, N, R, S)$ mit $R \subseteq (N \times \Sigma(N \cup \Sigma)^*)$ heißt simpel oder s-Grammatik gdw.

$$\forall A \in N \forall a \in \Sigma : |\{\beta \in (N \cup \Sigma)^* | A \to a\beta \in R\}| \leq 1 \tag{26}$$

23.0.7 Lemma

Linksparser für s-Grammatiken arbeiten deterministisch

24 Berechenbarkeit

24.1 Berechenbarkeit

- Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ ist berechenbar, wenn es einen Algorithmus (mechanisches Rechenverfahren) gibt, der f berechnet, also nach Eingabe von $(n_1, \cdots, n_k) \in \mathbb{N}^k$ in endlich vielen Schritten terminiert mit Ausgabe $f(n_1, \cdots, n_k)$

24.2 Churches These

Die durch den formalen Begriff der *Turingberechenbarkeit* (`WHILE,GOTO,µ-Rekursivität`) erfasste Klasse von Funktionen stammt genau mit der Klasse der intuitiv berechenbaren Funktionen überein.

24.3 Turing-Berechenbarkeit

- Der Berechenbarkeitsbegriff sollte mit Hilfe einer sehr einfachen Rechenmaschine beschrieben werden (**Turing-Maschine**) erfasst werden. - Eine (nichtdeterministische) Turingmaschine (TM) ist ein 7-Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, E)$, wobei Q eine endliche Menge von Zuständen, Σ eine endliche Menge von Eingabesymbolen, $\Gamma \subset \Sigma$ endliche Menge von Bandsymbolen

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\} \text{ deterministische} \quad (27)$$

$$\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R,N\}} \text{ nichtdeterministische} \quad (28)$$

$q_0 \in Q$ Startzustand,$\square \in \Gamma - \Sigma$ leeres Bandsymbolen $E \subseteq Q$ endliche Menge von Endzuständen

24.3.1 Konfiguration

Eine Konfiguration einer TM M ist ein Wort $k \in \Gamma^* Q \Gamma^*$: $k = \alpha q \beta$ bedeutet:

- $\alpha\beta$ ist der nichtleere Teil der Bandinschrift
- q ist der Zustand TM.
- Das erste Zeichen von β ist das Zentrum des Lese-/Schreibkopfes

24.3.2 formale Definition

Eine Funktion $f : \mathbb{N}^* \to \mathbb{N}$ heißt *Turing-berechenbar*, falls es eine (deterministische) TM M gibt, so dass für alle $n_1, \cdots, n_k, m \in \mathbb{N}$ gilt:

$$f(n_1, \cdots, n_k) = m \Leftrightarrow q_0 bin(n_1) \# bin(n_2) \# \cdots \# bin(n_k) \vdash^* q_e bin(m) \quad (29)$$

Eine Funktion $f : \Sigma^* \to \Sigma$ heißt Turing-berechenbar, falls es eine (deterministische) TM M gibt, so dass für alle $x, y \in \Sigma^*$ gilt:

$$f(x) = y \Leftrightarrow q_0 x \vdash^* q_e y \quad (30)$$

Damit ist ausgedrückt, dass im Falle $f(x) = undef$ M eine unendliche Schleife geht.

24.3.3 Mehrbanddefinition

Zu jeder Mehrband-TM M gibt es eine 1-Band TM M' die dieselbe Funktion berechnet wie M.

1. formaler

$$\Gamma' := (\Gamma \cup \{*\})^{2k} \tag{31}$$

M' simuliert M wie folgt: Gestartet mit Eingabe $x_1, \cdots, x_n \in \Gamma^*$ erzeugt M' die Darstellung der Startkonfiguration von M in Spuren-Darstellung.

- Sei M eine 1-Band TM: $M(i,k); i \leq n$ bezeichne die k -Band TM, die aus M dadurch entsteht, dass alle Aktionen auf Band i ablaufen.

24.4 `LOOP-`, `WHILE-`, und `GOTO-` Berechenbarkeit

24.4.1 `LOOP`

- Variablen: $x_0, x_1, \cdots$
- Konstanten: 0, 1, 2
- Trennsymbole: ;, $:=$
- Operationszeichen: $+$,-
- Schlüsselwörter `LOOP`, `DO`, `END`
- Wertzuweisung:

  $x_i := x_j + c$ $x_i := x_j - c$ ist ein `LOOP`-Programm
- Sind P_1, P_2 `LOOP`-Programme, dann auch $P_1; P_2$
- ist P ein `LOOP`-Programm, x_1 Variable, dann auch `LOOP` x_1 `DO` P `END`;

1. formale Definition Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ heißt `LOOP`-berechenbar, falls es ein `LOOP`-Programm P gibt, das gestartet mit $n_1, \cdots, n_k$ in den Variable $x_1, \cdots, x_k$ mit dem Wert $f(n_1, \cdots, n_k)$ in x_0 stoppt.

   (a) Bemerkung
   - Alle `LOOP`-berechenbaren Funktionen sind total definiert.
   - Erweiterung der `LOOP`-Programme durch `WHILE`-Schleife $\Rightarrow$ `WHILE`-Programme

24.4.2 WHILE

1. formale Definition Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ heißt WHILE-berechenbar, falls es ein WHILE-Programm P gibt, das gestartet mit $n_1, \cdots, n_k$ in $x_1, \cdots, x_k$ (0 sonst) mit dem Wert $f(n_1, \cdots, n_k)$ in x_0 stoppt. Sonst stoppt P nicht.

   (a) Bemerkung
   - TM können WHILE-Programme simulieren. D.h. jede WHILE-berechenbare Funktion ist auch TM-berechenbar.

24.4.3 GOTO

1. formale Definition Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ heißt GOTO-berechenbar, falls es ein GOTO-Programm P gibt, das gestartet mit $n_1, \cdots, n_k$ in $x_1, \cdots, x_k$ (0 sonst) mit dem Wert $f(n_1, \cdots, n_k)$ in x_0 stoppt. Sonst stoppt P nicht

   (a) Bemerkung
   - Jedes WHILE-Programm kann durch ein GOTO-Programm simuliert werden.
   - Jedes GOTO-Programm kann durch ein WHILE-Programm (mit nur einer WHILE-Schleife) simuliert werden.
   - Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ ist GOTO-berechenbar, falls f WHILE-berechenbar ist.
   - Eine Funktion ist TM-berechenbar $\Leftrightarrow$ GOTO-berechenbar $\Leftrightarrow$ WHILE-berechenbar

24.5 Primitiv rekursive und μ-rekursive Funktionen

24.5.1 Definition

Die Klasse der primitiv rekursiven Funktione ist induktiv wie folgt definiert:

1. Alle konstanten Funktionen sind primitiv rekursive
2. Projektionsabbildungen sind primitiv rekursive, d.h $pr_i : \mathbb{N}^k \to \mathbb{N}$
   $pr_i(n_1, \cdots, n_k) = n_i$
3. Die Nachfolgerfunktion ist primitiv rekursiv, d.h. $s : \mathbb{N}^k \to \mathbb{N}$
   $i \mapsto i + 1$
4. Jede Funktion, die durch Einsetzung (Komposition) aus primitiv rekursiven Funktionen entsteht ist primitiv rekursiv.

5. Jede Funktion, die durch primitive Rekursion aus primitiv rekursiven Funktionen entsteht, ist primitiv rekursiv.

1. Bemerkung

   - primitiv rekursive Funktionen sind offenbar berechenbar
   - primitiv rekursive Funktionen sind total berechenbar
   - Es gilt nicht : primitiv rekursive = total und berechenbar

24.5.2 Zusammenhang zur LOOP-berechenbaren Funktionen

Die Klasse der primitiv rekursiven Funktionen stimmt mit der Klasse der LOOP-berechenbaren Funktionen überein.

24.5.3 µ-Operator

Sei $f : \mathbb{N}^{k+1} \to \mathbb{N}$ Durch Anwendung des µ-Operators entsteht aus f die Funktion $g : \mathbb{N}^K \to \mathbb{N}$ mit $g(x_1, \cdots, x_k) = min\{n : f(n, x_1, \cdots, x_k) = 0$ und für alle $m < n$ ist $f(m, x_1, \cdots, x_k) def.\}$ mit min $\emptyset = 0$ undefiniert.

Durch die Anwendung des µ-Operators können partielle Funktionen entstehen.

24.5.4 formale Definition

Die Klasse der µ-rekursiven Funktionen ist die kleinste Klasse von Funktionen, die die Basisfunktionen (konstante Funktionen, Projektionen, Nachfolgerfunktion) enthält und abgeschlossen ist bzgl. der primitiven Rekursion und der Anwendung des µ-Operators. Die Klasse des µ-rekursiven Funktionen stimmt genau mit der Klasse der WHILE- (GOTO- , TM-) berechenbaren Funktionen überein.

24.5.5 Kleene

Für jede n -stellige µ-rekursive Funktion f gibt es zwei $n + 1$ -stellige, primitiv rekursive Funktionen p und q, so dass sich f darstellen lässt als

$$f(x_1, \cdots, x_n) = p(x_1, \cdots, x_n, q(x_1, \cdots, x_n)). \tag{32}$$

24.6 Ackermann-Funktion

24.6.1 Definition

Die Ackermann-Funktion ist eine Funktion, die intuitive berechenbar (`WHILE` berechenbar), aber nicht primitiv rekursive (`LOOP` berechenbar) ist.

$$ack(0, y) = y + 1 \tag{33}$$
$$ack(x, 0) = ack(x - 1, 1) \tag{34}$$
$$ack(x, y) = ack\,(x - 1, ack\,(x, y - 1)) \tag{35}$$
$$ack(x, y) = ack\,\underbrace{(x - 1, ack\,(x - 1, \cdots ack\,(x - 1, 1) \cdots))}_{y-mal} \tag{36}$$

24.6.2 Lemma

Für jedes `LOOP`-Programm P gibt es eine Konstante k mit

$$f_p(n) < a(k, n) \text{ für alle } n \geq k \tag{37}$$

24.6.3 Eigenschaften

- Die Ackermann-Funktion a ist nicht `LOOP`-berechenbar.
- Die Ackermann-Funktion ist `WHILE`-berechenbar.

24.7 Entscheidbarkeit und Semi-Entscheidbarkeit

24.7.1 Definition

$A \subseteq \Sigma^*$ heißt entscheidbar, falls die charakteristische Funktion

$$\chi_A : \Sigma^* \to \{0, 1\} \tag{38}$$
$$\chi_A(\omega) = \begin{cases} 1 & \text{falls } \omega \in A \\ 0 & \text{sonst} \end{cases} \tag{39}$$

berechenbar ist. $A \subseteq \Sigma^*$ heißt semi-entscheidbar, falls die charakteristische Funktion

$$\chi_A^{'}(\omega) = \begin{cases} 1 & \text{falls } \omega \in A \\ \text{undefiniert} & sonst. \end{cases} \tag{40}$$

berechenbar ist.

1. Man kann an Stelle von $A \subseteq \Sigma^*$ auch $A \subseteq \mathbb{N}$ betrachten.

2. Das Entscheidungsproblem für A ist die Frage nach einem stoppenden Algorithmus mit $\omega \to (ja, nein)$

3. Das Semi-Entscheidungsproblem für A ist die Frage nach einem Algorithmus mit $\omega \to (ja, ???)$. Hat der Algorithmus noch nicht gestoppt, dann ist unklar ob $\omega \in A$ oder nicht.

24.7.2 Satz

- A ist entscheidbar $\Leftrightarrow$ sowohl A als auch $\bar{A}$ sind semi-entscheidbar.
- $A \subseteq \Sigma^*$ heißt rekursive aufzählbar, falls $A = \emptyset$ oder es eine totale und berechenbare Funktion $f : \mathbb{N} \to \Sigma^*$ gibt mit

$$A = \{f(0), f(1), f(2), \cdots\} \tag{41}$$

"f zählt A auf"

- Eine Sprache ist rekursive aufzählbar, genau dann wenn sie semi-entscheidbar ist.
- Eine Sprache A ist entscheidbar, genau dann wenn A und $\bar{A}$ rekursive aufzählbar sind.
- A heißt abzählbar, falls $A = \emptyset$ oder es gibt eine totale Funktion f

$$A = \{f(0), f(1), f(2), \cdots\} \tag{42}$$

- A ist rekursive aufzählbar, falls A durch eine totale rekursive Funktion abzählbar ist.

24.8 Das Halte-Problem und die Reduzierbarkeit

24.8.1 Definition

Die folgende Sprache

$$K = \{\omega \in \{0,1\}^* | M_\omega \text{ angesetzt auf } \omega \text{ hält } \} \tag{43}$$

heißt spezielles Halte-Problem

24.8.2 Eigenschaften

1. Das spezielle Halte-Problem ist nicht entscheidbar.
2. Seien $A, B \subseteq \Sigma^*$. A heißt auf B reduzierbar ($A \leq B$), falls es eine totale berechenbare Funktion. $f : \Sigma^* \to \Sigma^*$ gibt mit:
$$\forall x \in \Sigma^* x \in A \Leftrightarrow f(x) \in B \tag{44}$$
3. Gilt $A \leq B$ und ist B entscheidbar, so ist auch A entscheidbar.
4. Gilt $A \leq B$ und ist B semientscheidbar, so ist auch A semientscheidbar.
5. $A \leq B$ und A ist nicht entscheidbar $\Rightarrow B$ ist nicht entscheidbar
6. Die Sprache $H = \{\omega \# x | M_\omega$ angesetzt auf x hält $\}$ heißt (allgemeines) Halte-Problem
7. Das Halte-Problem ist nicht entscheidbar
8. Die Sprache $H_0 = \{\omega | M_\omega$ angesetzt auf leeren Band hält$\}$ heißt Halte-Problem auf leeren Band.
9. Das Halte-Problem auf dem leeren Band H_0 ist nicht entscheidbar.

24.8.3 Satz von Rice

Sei $\mathcal{R}$ die Klasse aller TM-berechenbaren Funktionen. Sei $\mathcal{S} \subset \mathcal{R}, \mathcal{S} \neq \emptyset$, Dann ist die Sprache

$$\mathcal{C}(\mathcal{S}) := \{\omega | \text{ die von } M_\omega \text{ berechnete Funktion liegt in } \mathcal{S}\} \tag{45}$$

unentscheidbar.

24.9 Das Postsche Korrespondenz-Problem

24.9.1 Definition

Das nachfolgend beschriebene Problem heißt Postsches Korrespondez-Problem (PCP)

$gegeben$:
Eine endliche Folge $(x_1, y_1), \cdots, (x_k, y_k)$ von Wortpaaren mit $x_i, y_i \in A^+$ (A endliche Alphabet)

$gefragt$:
Gibt es eine Folge von Indizes $i_1, \cdots, i_n \in \{1, \cdots, k\}, n \geq 1,$ mit x_{i_1}

24.9.2 Eigenschaften

- Das Postsche Korrespondenz-Problem ist nicht entscheidbar
- $MPCP \leq PCP$
- $H \leq MPCP$
- PCP bleibt unentscheidbar, falls für das Alphabet A gilt: $A = \{0, 1\}$. ("01-PCP")
- Sei PCP_k die Variante des PCP s deren Eingabe aus genau k Wortpaaren besteht.
  - PCP_k ist unentscheidbar für $k \geq 9$
  - PCP_k ist entscheidbar für $k \leq 2$
- Das Halte-Problem H für TM ist semi-entscheidbar.

24.10 Universelle TM

24.10.1 Definition

Nachobiger Folgerung gibt es eine TM U, die sich bei Eingabe von $\omega\#x$ so verhält wie M_ω bei Eingabe von x. (Zunächst nur im Bezug auf Halten, bzw. Nicht-Halten).

24.10.2 Eigenschaften

- Das Schnittproblem für kontextfreie Sprachen (G_1, G_2 kontextfrei Sprachen gilt: $L(G_1) \cup L(G_2) = \emptyset$?) ist unentscheidbar.
- Das Schnittproblem für deterministische kontextfreie Sprachen ist unentscheidbar.
- Das Äquivalenzproblem für kontextfreie Sprachen ist unentscheidbar.
  - Das Äquivalenzproblem ist für folgende Probleme unentscheidbar:
    * nichtdeterministische Kellerautomaten
    * BNF
    * EBNF
    * Syntaxdiagramme
    * LBA

    - kontextsensitiven Grammatiken
    - TM

- Das Leerheitsproblem für kontextsensitive Sprachen ist unentscheidbar.

24.11 Der Gödelsche Unvollständigkeitssatz

24.11.1 Definition

Arithmetische Terme sind induktiv wie folgt definiert:

- Jedes $n \in \mathbb{N}$ und jede Variabel $x_i, i \in \mathbb{N}$ ist ein Term
- sind t_1, t_2 Terme, so ist auch $(t_1 + t_2)$ und $(t_1 \cdot t_2)$ Terme.

Arithmetische Formeln sind wie folgt definiert:

- Sind t_1, t_2 Terme, dann ist $t_1 = t_2$ eine Formel.
- Sind F, G Formeln, dann auch $\neg F, F \vee G$ und $F \wedge G$
- Ist x eine Variable und F eine Formeln, dann sind $\exists x F$ und $\forall x F$ Formeln.
- Eube Variable heißt gebunden, wenn sie in Wirkungsbereich eines Quantoren steht. Sonst heißt eine Variable frei.
- Eine Funktion $f:\mathbb{N}^k \to \mathbb{N}$ ist arithmetisch repräsentierbar, falls es eine arithmetische Formel $F(x_1, \cdots, x_k, y)$ gibt, so dass für alle $n_1, \cdots, n_k, m \in \mathbb{N}$ gilt:

$$f(n_1, \cdots, n_k) = m \Leftrightarrow F(x_1/n_1, , x_k/n_k, y/m) \text{ ist wahr.} \tag{46}$$

- Jede `WHILE`-berechenbare Funktion ist arithmetisch repräsentierbar.
- Die Menge der arithmetischen Formeln ist nicht rekursiv aufzählbar.

24.11.2 Zusatz

Jedes Beweissystem für WA ist notwendigerweise unvollständig (d.h es bleiben stets Formeln, die nicht beweisbar sind.)

24.12 Beweissystem

24.12.1 Definition

Ein Beweissystem für eine Menge $A \subseteq \Gamma^*$ ist ein Paar (B, F) mit folgenden Eigenschaften:

1. $B \subseteq \Gamma^*$ ist entscheidbar.
2. $F : B \to A$ ist total und berechenbar.

25 Komplexität

25.0.2 `TIME`-Komplexität

Die Komplexitätsklasse `TIME` $(f(n))$, wobei $f(n)$ nach oben durch `LOOP`-Programme beschränkt weden kann, ist enthalten in der Klasse der primitiv rekursiven Sprachen (bzw. der `LOOP`-berechenbaren Sprachen)

1. Korollar =TIME=$(n^k (k \in \mathbb{N}), \texttt{TIME}(2^n), \texttt{TIME}(2^{2^{\cdots^2}})$ enthalten nur primitiv rekursiven Mengen.

25.0.3 Die Komplexitätsklassen P und NP

1. Definition Ein Polynom $p : \mathbb{N} \to \mathbb{N}$ ist eine Funktion der Form

$$p(n) = a_k n^k + \cdots + a_1 n + a_0 \qquad a_i \in \mathbb{N}, k \in \mathbb{N} \tag{47}$$

2. Komplexitätsklasse P

$$P = \bigcup_{P \text{ Polynom}} TIME(p(n)) \tag{48}$$

$$= \{A : TMM \text{ mit } L(M) = A \text{ und } time_M(x) \leq p(|x|) \text{ für ein Polynom } p\} \tag{49}$$

$$\Rightarrow P = \bigcup_{k \geq 1} TIME(O(n^k)) \tag{50}$$

- $P =$ Klasse der effizienten Algorithmen (d.h. der praktisch realisierbaren Algorithmen)
- Algorithmen mit exponentieller Laufzeit sind nicht effizient.

- P könnte auch als `WHILE`-Programm mit logarithmischen Kostenmaß definiert werden.
- \$P$\subseteq TIME(2^n)\subseteq$\Primitiv rekursive Sprache\
- P enthält alle Probleme, für die sich in polynomieller Zeit ein Beweis finden lässt.

3. Komplexitätsklasse NP Sei M eine nichtdeterministische Mehrband-TM

$$time_M(x) = \begin{cases} min & \{ \text{Länge einer akzeptierenden Berechnung von } M \text{ auf } x\}, x \in L(M) \\ 0, & x \notin L(M) \end{cases} \tag{51}$$

$$f : \mathbb{N} \to \mathbb{N} \tag{52}$$

$$NTIME(f(n)) = \{A : NTMM \text{ mit } L(M) = A \text{ und } time_M(x) \leq f(|x|) \forall x \in \Sigma^*\} \tag{53}$$

$$NP := \bigcup_{pPolynom} NTIMEp(n) \tag{54}$$

$$= \bigcup_{k \geq 1} TIME(O(n^k)) \tag{55}$$

Offenbar gilt: $P \subseteq NP$. Die Umkehrung ist unklar.

4. NP -Vollständig

   (a) Definition
   Seien $A, B \subseteq \Sigma^*$. A heißt polynomiell auf B reduzierbar ($A \leq_p B$), falls es eine totale und mit polynomialer Komplexität berechenbare Funktion $f : \Sigma^* \to \Sigma^*$ gibt, so dass für alle $x \in \Sigma^*$ gilt:

$$x \in A \Leftrightarrow f(x) \in B \tag{56}$$

   (b) Lemma
   - $\leq_p$ ist transitiv
   - $A \leq_p B, B \in P \to A \in P$
   - $A \leq_p B, B \in NP \to A \in NP$
   - Eine Sprache A heißt $NP-hart$, falls für alle Sprachen $L \in NP$ gilt: $L \leq_p A$.
   - Eine Sprache A heißt NP -vollständig, falls A NP -hart ist und ein $A \in NP$ gilt.

- NP -vollständige Sprachen sind ßchwereSSprachen in NP.
- Sei A NP -vollständig. Dann gilt:

$$A \in P \Leftrightarrow P = NP \tag{57}$$

5. Erfüllbarkeitsproblem der Aussagenlogik `SAT` Das folgende Problem heißt SAT:
   - Eine Formel F der Aussagenlogik mit n Variablen, $n \in \mathbb{N}$
     - Ist F erfüllbar? (d.h. Belegung $a \in \{0,1\}^n$ mit $F(a) = 1$?)
6. Theorem von Cook Das Erfüllbarkeitsproblem der Aussagenlogik `SAT` ist NP -vollständig.
7. weitere NP -vollständige Probleme
   (a) Definition von `3SAT` Boolsche Formel F in KNF mit höchstens 3 Literalen pro Klausel.
   (b) Eigenschaft `3SAT` ist NP -vollständig.
8. `CLIQUE`
   (a) Definition Ein ungerichteter Graph $G = (V, E), k \in \mathbb{N}$
   - Besitzt G eine Clique der Größe k? Wobei Clique ein vollständiger Teilgraph $G' = (V', E')$ ist, mit $(u, v) \in E \quad \forall u, v \in V', u \neq v$

   (b) Satz `CLIQUE` ist NP -vollständig
9. Hamilton-Kreis
   (a) Definition Ein Hamilton-Kreis ist eine Permutation der Knotenindizies $(\nu_{\pi(1)}, \cdots, \nu_{\pi(n)}$, so dass $(\nu_{\pi(i)}, \nu_{\pi(i+1)}) \in E \forall i = 1, \cdots, n-1$ und $(\nu_{\pi(n)}, \nu_{\pi(1)}) \in E$
   (b) Satz
   - Ein gerichteter Hamilton-Kreis ist NP -vollständig
   - F ist erfüllbar $\Leftrightarrow G$ hat Hamilton-Kreis.
   - Ein ungerichteter Hamilton-Kreis ist NP -vollständig.
   - Eulerkreis ist in P Königsberger Brückenproblem.

10. Traveling Salesperson $n \times n$ Matrix $(M_{i,j})$ von "Entfernungen" zwischen n Städten. Gesucht Permutation ("Rundreise") mit

$$\Sigma_{i=1}^{n-1} M_{\pi(i),\pi(i+1)} + M_{\pi(n),\pi(1)} \leq k \tag{58}$$

    (a) Satz Das Traveling Salesperson Problem ist NP -vollständig.