Deterministischer endlicher Automat

CIRCUITVALUE

Nichtdeterministische Turingmaschine

Spezielles Halteproblem $\mathsf{HALT}^{\varepsilon}_{\mathsf{TM}}$

inkl. Aussagen zu Entscheidbarkeit

$$\mathsf{CIRCUITVALUE} = \Big\{ \langle C, s \rangle : \begin{array}{l} C \text{ ist Schaltkreis} \\ \text{mit einem Ausgang,} \\ \text{und } s \text{ ist ein Eingabevektor,} \\ \text{für den } C \text{ am Ausgang} \\ \text{den Wert 1 liefert} \end{array} \Big\}$$

Ein deterministischer endlicher Automat ist ein 5-Tupel

$$(Q, \Sigma, \delta, q_0, F) \quad ,$$

wobei

- Q eine endliche, nichtleere Menge von Zuständen,
- Σ ein Alphabet,
- $\delta\colon Q \times \Sigma \to Q$ die Zustandsübergangsfunktion,
- $q_0 \in Q$ der Startzustand
- $F \subseteq Q$ die Menge der akzeptierenden Zustände

sind.

Man beachte, dass es keine akzeptierenden Zustände geben muss, also $F = \varnothing$ gelten kann.

$$\mathsf{HALT}^{\varepsilon}_{\mathsf{TM}} = \Big\{ \langle M \rangle : \begin{array}{l} M \text{ ist eine Turingmaschine,} \\ \text{die bei Eingabe } \varepsilon \text{ hält} \end{array} \Big\}$$

$\mathsf{HALT}^{\varepsilon}_{\mathsf{TM}}$ ist nicht entscheidbar. Man beweist dies, indem man vom als unentscheidbar bekannten Halteproblem für Turingmaschinen reduziert.

Eine nichtdeterministische Turingmaschine ist ein 7-Tupel

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}) \quad ,$$

wobei

- Q eine endliche, nichtleere Menge von Zuständen,
- Σ das Eingabealphabet mit $\sqcup \notin \Sigma$,
- Γ das Band- oder Arbeitsalphabet mit $\Sigma \cup \{\sqcup\} \subseteq \Gamma$,
- $\delta\colon Q \setminus \{q_{acc}, q_{rej}\} \times \Gamma \to \mathfrak{P}(Q \times \Gamma \times \{L, R\})$ die Zustandsübergangsfunktion,
- $q_0 \in Q$ der Startzustand,
- $q_{acc} \in Q$ der akzeptierende Zustand,
- $q_{rej} \in Q \setminus \{q_{acc}\}$ der verwerfende Zustand

sind.

Kellerautomat

Turing-berechenbare Funktion

- Eine Satzform ist eine Folge von Terminalsymbolen und dem Leerzeichen, d.h. ist $G = (N, T, P, S)$ eine Grammatik und α eine Satzform über G, so gilt $\alpha \in (T \cup \{\sqcup\})^*$.
- Satzformen enthalten Variablen (d.h. Nichtterminalsymbole) und Terminalsymbole.
- Satzformen sind nicht leer.

Pumping-Lemma für reguläre Sprachen

6 *Antwort*

Sei Σ ein Alphabet und $f\colon \Sigma^* \to \Sigma^*$ eine Funktion. Wir nennen f Turing-berechenbar, wenn es eine Turingmaschine M gibt, die zu jedem $w \in \Sigma^*$ in einer akzeptierenden Konfiguration hält und dann ausschließlich $f(w)$ auf dem Band stehen hat.

5 *Antwort*

Ein Kellerautomat ist ein 8-Tupel

$$(Q,\ \Sigma,\ \$,\ \perp,\ \Gamma,\ \delta,\ q_0,\ F) \quad ,$$

wobei

- Q eine endliche, nichtleere Menge von Zuständen,
- Σ das Eingabealphabet, laut Skript endlich!
- Γ das Kelleralphabet, laut Skript endlich!
- $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}$, $\Sigma_{\$,\varepsilon} := \Sigma \cup \{\varepsilon,\ \$\}$ und $\Gamma_\varepsilon := \Gamma \cup \{\varepsilon\}$,
- $\$ \notin \Sigma$ das Wortendesymbol,
- $\perp \in \Gamma$ das Kellerbodensymbol,
- $\delta\colon Q \times \Sigma_{\$,\varepsilon} \times \Gamma_\varepsilon \to \mathfrak{P}(Q \times \Gamma_\varepsilon)$ die Zustandsübergangsfunktion,
- q_0 der Startzustand,
- $F \subseteq Q$ die Menge der akzeptierenden Zustände

sind.

Man beachte, dass es keine akzeptierenden Zustände geben muss, also $F = \varnothing$ gelten kann.

8 *Antwort*

Sei Σ ein Alphabet, und sei $L \subseteq \Sigma^*$ eine reguläre Sprache. Dann gibt es eine Zahl $n \in \mathbb{N}_{\geqslant 1}$ derart, dass für jedes Wort $z \in L$ mit $|z| \geqslant n$ Wörter $u,\ v,\ w \in \Sigma^*$ existieren mit

- $z = u\,v\,w$,
- $|u\,v| \leqslant n$,
- $|v| \geqslant 1$,
- $\forall\, i \in \mathbb{N}_0 \quad u\,v^i\,w \in L$.

7 *Antwort*

- falsch
- wahr
- falsch

Pumping-Lemma für kontextfreie Sprachen

2-SAT

inkl. Aussagen zu Entscheidbarkeit

$\mathsf{MEET}_{\mathsf{CFG}}$

inkl. Aussagen zu Entscheidbarkeit

CYK-Algorithmus

$$\text{2-SAT} = \{ \langle \varphi \rangle : \varphi \text{ ist eine erfüllbare Formel in KNF, deren jede Klausel aus genau zwei Literalen besteht} \}$$

2-SAT ist entscheidbar. Man beweist dies, indem man zu φ einen Graphen (V, E) konstruiert, dessen Knotenmenge alle negierten und nichtnegierten Literale aus φ enthält und dessen Kantenmenge für jede Klausel $x \vee y$ (x, y seien Literale) die Kanten $(\neg x, y)$ und $(x, \neg y)$ enthält. Falls es ein Literal ℓ mit $(\ell, \neg \ell) \in E$ gibt, verwirf die Eingabe; sonst akzeptiere sie.

Sei Σ ein Alphabet, und sei $L \subseteq \Sigma^*$ eine kontextfreie Sprache. Dann gibt es eine Zahl $n \in \mathbb{N}_{\geqslant 1}$ derart, dass für jedes Wort $z \in L$ mit $|z| \geqslant n$ Wörter $u, v, w, x, y \in \Sigma^*$ existieren mit

- $z = u\,v\,w\,x\,y$,
- $|v\,w\,x| \leqslant n$,
- $|v\,x| \geqslant 1$,
- $\forall\, i \in \mathbb{N}_0 \quad u\,v^i\,w\,x^i\,y \in L$.

Hier fehlt noch was!

$$\text{MEET}_{\text{CFG}} = \{ \langle G_1, G_2 \rangle : G_1, G_2 \text{ sind kontextfreie Grammatiken, und es gilt } L(G_1) \cap L(G_2) \neq \varnothing \}$$

MEET_{CFG} ist Turing-erkennbar, aber nicht Turing-entscheidbar. Man beweist Letzteres, indem man vom als unentscheidbar bekannten Postschen Korrespondenzproblem reduziert.

Generalisierter nichtdeterministischer endlicher Automat (GNEA)

a.k.a. NEA mit Worttransitionen

3-COLOR

k-knotenfärbbarer Graph

Komplexitätsklasse P

$$\mathsf{3\text{-}COLOR} \;=\; \{\; \langle G \rangle \;:\; G \text{ ist ein 3-knotenfärbbarer Graph} \;\}$$

Ein GNEA ist ein 5-Tupel

$$(Q,\ \Sigma,\ \delta,\ q_{\text{start}},\ q_{\text{acc}}) \quad ,$$

wobei

- Q eine endliche, nichtleere Menge von Zuständen,
- Σ ein Alphabet,
- $\mathcal{R}_\Sigma$ die Menge aller regulären Ausdrücke über Σ,
- $\delta\colon Q \setminus \{q_{\text{acc}}\} \times \mathcal{R}_\Sigma \to Q \setminus \{q_{\text{start}}\}$ die Zustandsübergangsfunktion,
- $q_{\text{start}} \in Q$ der Startzustand,
- $q_{\text{acc}} \in Q \setminus \{q_{\text{start}}\}$ der akzeptierende Zustand

sind.

$$\mathsf{P} = \mathsf{DTIME}(\mathcal{P}),$$

wobei $\mathcal{P}$ die Menge aller Polynomfunktionen mit einer Variablen und nichtnegativen Koeffizienten ist.

Die Komplexitätsklasse P enthält genau die Entscheidungsprobleme, die von einer deterministischen Turingmaschine in Polynomialzeit gelöst werden können.

Sei $k \in \mathbb{N}_0$.

Ein ungerichteter Graph $G = (V,\ E)$ heißt k-knotenfärbbar, falls es eine gültige k-Färbung der Knoten von G gibt, d.h. eine Abbildung $f\colon V \to \{1,\ \ldots,\ k\}$ mit

$$\forall\, v,\ w \in V: \quad v \neq w \ \wedge\ \{v,\ w\} \in E \implies f(v) \neq f(w) \quad .$$

Ein gerichteter Graph $G = (V,\ E)$ heißt k-knotenfärbbar, falls es eine gültige k-Färbung der Knoten von G gibt, d.h. eine Abbildung $f\colon V \to \{1,\ \ldots,\ k\}$ mit

$$\forall\, v,\ w \in V: \quad (v,\ w) \in E \implies f(v) \neq f(w) \quad .$$

Entscheidungsproblem

Turing-entscheidbare Sprache

A_{TM}

inkl. Aussagen zu Entscheidbarkeit

Turing-erkennbare Sprache

18 *Antwort*

Sei Σ ein Alphabet. Eine Sprache L heißt Turing-entscheidbar, falls es eine Turingmaschine M gibt, die auf jeder Eingabe $w \in \Sigma^*$ hält und für die gilt:

- Ist $w \in L$, so hält M in einer akzeptierenden Konfiguration.
- Ist $w \notin L$, so hält M in einer verwerfenden Konfiguration.

17 *Antwort*

Ein Entscheidungsproblem ist ein Problem, für das nur die Antworten „ja“ und „nein“ in Frage kommen.

20 *Antwort*

Sei Σ ein Alphabet. Eine Sprache L heißt Turing-erkennbar, falls es eine Turingmaschine M so gibt, dass für alle $w \in \Sigma^*$ gilt:

- Ist $w \in L$, so hält M in einer akzeptierenden Konfiguration.
- Ist $w \notin L$, so braucht M nicht zu halten; aber wenn doch, so hält M in einer verwerfenden Konfiguration.

19 *Antwort*

$$\mathsf{A}_{\mathsf{TM}} = \{ \langle M, w \rangle : \begin{array}{l} M \text{ ist eine Turingmaschine} \\ \text{und } M \text{ akzeptiert } w \end{array} \}$$

A_{TM} ist Turing-erkennbar, aber nicht Turing-entscheidbar. Man beweist Ersteres, indem man eine Turingmaschine angibt, die alle positiven Instanzen erkennt, und Letzteres durch Diagonalisierung.

Man vergleiche dazu
http://www.youtube.com/watch?v=92WHN-pAFCs

Letzter Zugriff am 18.11.2016

Nichtabgeschlossenheit der kontextfreien Sprachen unter Schnitt

inkl. Beweisidee

Stille Transition

Abschlusseigenschaften der regulären Sprachen

inkl. Beweisidee

Umwandlung einer kfG G in eine kfG G' ohne Startsymbol auf rechter Regelseite

Sei $(Q, \Sigma, \delta, q_0, F)$ ein nichtdeterministischer endlicher Automat. Eine stille Transition ist ein Zustandsübergang, der sich ohne Lesen eines Eingabezeichens ereignet.

Formal: Wir definieren für jeden Zustand $q \in Q$ die Menge $E(q)$ der stillen Transitionen, indem wir für alle $q' \in Q$ festlegen:

$$\begin{aligned} & q' \in E(q) \;:\Longleftrightarrow \\ & \quad \Big(\exists\, n \in \mathbb{N} : \exists\, r_0, \ldots, r_n \in Q : \\ & \qquad r_0 = q \;\wedge\; r_n = q' \;\wedge\; \bigwedge_{i=0}^{n-1} \delta(r_i, \varepsilon) = r_{i+1}\Big) \;\vee\; (q' = q)\,. \end{aligned}$$

Für eine Menge $R \subseteq Q$ von Zuständen definieren wir

$$E(R) := \bigcup_{q \in R} E(q) \quad .$$

Sind L_1 und L_2 kontextfreie Sprachen, so muss ihr Durchschnitt $L_1 \cap L_2$ nicht kontextfrei sein.

Man beweist dies, indem man geeignete Sprachen miteinander schneidet.

Die kontextfreien Sprachen

$$\begin{aligned} L_1 &:= \left\{a^n\, b^n\, c^k \,:\, n, k \in \mathbb{N}_0\right\} \quad \text{und} \\ L_2 &:= \left\{a^k\, b^n\, c^n \,:\, n, k \in \mathbb{N}_0\right\} \end{aligned}$$

etwa schneiden sich in der Sprache

$$L_1 \cap L_2 \;=\; \{a^n\, b^n\, c^n \,:\, n \in \mathbb{N}_0\} \quad ,$$

die nicht kontextfrei ist.

Zu jeder kontextfreien Grammatik G gibt es eine kontextfreie Grammatik G' derart, dass in keiner Regel von G' das Startsymbol von G auf der rechten Seite steht, und für die außerdem $L(G) = L(G')$ gilt.

Die Klasse der regulären Sprachen ist abgeschlossen unter

- Vereinigung Produktautomat
- Durchschnitt
- Kleene-Stern NEA basteln
- Konkatenation

Außerdem noch unter

- Komplement
- Differenzbildung de Morgan
- Spiegelung
- „Abrunden“ $\lfloor\cdot\rfloor$ (Serie 8, Aufgabe 2)
- Shuffle-Operator $\sqcup\!\sqcup$ (Serie 7, Aufgabe 4)

Sie ist *nicht* abgeschlossen unter

- Teilmengenbildung

- Die Menge der 0/1-Folgen, also die Menge

$$\{0,\ 1\}^{\mathbb{N}} \quad = \quad \{f\ :\ f \text{ ist eine Funktion von } \mathbb{N} \text{ nach } \{0,\ 1\}\}$$

  ist überabzählbar.

- Die Menge der Turingmaschinen mit Bandalphabet $\Gamma = \{a,\ b,\ c,\ \sqcup\}$ ist überabzählbar.

Transformation von NEA zu GNEA

Satz von Cook und Levin

PARTY

INDEPENDENT-SET

Sei $N = (Q, \Sigma, \delta, q_0, F)$ ein nichtdeterministischer endlicher Automat. Wir erstellen einen generalisierten nichtdeterministischen endlichen Automaten $G := (Q_G, \Sigma, \delta_G, q_{start}, q_{acc})$ über N wie folgt:

- $Q_G := Q \cup \{q_{start}, q_{acc}\}$ sei die Menge der Zustände,
- $\mathcal{R}_\Sigma$ sei die Menge aller regulären Ausdrücke über Σ,
- Die Zustandsübergangsfunktion $\delta_G \colon Q \times Q \to \mathcal{R}_\Sigma$ sei für alle $p, q \in Q$ definiert durch

$$
\begin{aligned}
\delta_G(p, q) &:= \varepsilon && \text{falls } p = q_{start} \text{ und } q = q_0 \\
\delta_G(p, q) &:= \varepsilon && \text{falls } p \in F \text{ und } q = q_{acc} \\
\delta_G(p, q) &:= \bigcup_{q \in \delta(p, a)} \{a\} && \text{falls es } a \in \Sigma \text{ mit } q \in \delta(p, a) \text{ gibt} \\
\delta_G(p, q) &:= \varnothing && \text{in allen anderen Fällen}
\end{aligned}
$$

- wahr
- falsch

$$
\begin{aligned}
\mathsf{PARTY} = \{ \langle G, k \rangle \ : \ & k \text{ ist eine natürliche Zahl, und} \\
& G \text{ ist ein ungerichteter Graph,} \\
& \text{der eine unabhängige Menge} \\
& \text{der Mächtigkeit } k \text{ enthält} \}
\end{aligned}
$$

Eine alternative Bezeichnung für PARTY ist INDEPENDENT-SET.

Es gilt $\mathsf{SAT} \in \mathsf{NPC}$.

Man beweist dies, indem man $\mathsf{SAT} \in \mathsf{NP}$ zeigt (durch Angabe einer Turingmaschine, die nichtdeterministisch eine Belegung für die betrachtete Formel „rät“ und diese anschließend in Polynomialzeit verifiziert) und danach jedes Problem $A \in \mathsf{NP}$ auf SAT reduziert.

PATH

Charakterisierung Turing-entscheidbarer Sprachen mittels Turing-Erkennbarkeit

- Das Pumping-Lemma für reguläre Sprachen kann benutzt werden, um die Regularität einer Sprache zu zeigen.
- Das Pumping-Lemma für reguläre Sprachen kann benutzt werden, um ein Wort sinnvoll zu unterteilen.
- Das Pumping-Lemma für reguläre Sprachen kann zur Erkennung der Sprachstruktur benutzt werden.

Reguläre Operatoren

30 *Antwort*

Eine Sprache ist genau dann Turing-entscheidbar,
wenn sie Turing-erkennbar und ihr Komplement Turing-erkennbar ist.

29 *Antwort*

$$\mathsf{PATH} = \Big\{ \langle G, s, t \rangle : \begin{array}{l} G \text{ ist ein endlicher, gerichteter Graph,} \\ \text{und } s, t \text{ sind Knoten in } G, \\ \text{zwischen denen ein gerichteter Pfad} \\ \text{in } G \text{ existiert} \end{array} \Big\}$$

32 *Antwort*

Sei Σ ein Alphabet und seien $L_1, L_2 \subseteq \Sigma^*$ Sprachen. Wir definieren die folgenden drei Operatoren

- Vereinigung

$$L_1 \cup L_2 = \{w \in \Sigma^* : w \in L_1 \vee w \in L_2\}$$

- Konkatenation

$$L_1 \circ L_2 = \{uv \in \Sigma^* : u \in L_1 \wedge v \in L_2\}$$

- Kleene-Stern

$$L_1^* = \left\{w^{(0)} w^{(1)} \dots w^{(k)} \in \Sigma^* : k \in \mathbb{N}_0 \wedge \forall i \in [k]_0 : w^{(i)} \in L_1\right\}$$

und nennen sie die regulären Operatoren.

31 *Antwort*

- falsch
- wahr
- falsch

Neumann-artige Reihe für eine reguläre Sprache

ungerichteter Graph

gerichteter Graph

Unabhängige Menge

34 *Antwort*

Seien E und V disjunkte Mengen. Ist E eine Teilmenge der Menge aller 2-elementigen Teilmengen von V, so heißt das Paar (V, E) ein ungerichteter Graph (auf V). Die Elemente von V heißen dann Knoten oder Ecken, und die Elemente von E heißen Kanten.

33 *Antwort*

Man äußere sich zur Sinnhaftigkeit des Folgenden:

Frage:

Welche Sprache L erfüllt L = ε+aL?

Antwort: Wir subtrahieren aL und klammern L aus. Dann erhalten wir:

(ε - a)L = ε

Nach Division durch ε - a steht dann da:

$$L = \frac{\varepsilon}{\varepsilon - a}$$

Auf der rechten Seite erkennen wir eine Geometrische Reihe, und so bekommen wir die Antwort:

L = ε + a + aa + aaa + aaaa + ...

https://matheplanet.com/matheplanet/nuke/html/viewtopic.php?topic=84&post_id=1059835
https://de.wikipedia.org/w/index.php?title=Neumann-Reihe&oldid=199355591
Letzter Zugriff am 02.05.2020

36 *Antwort*

Sei V eine nichtleere Menge, und sei $G = (V, E)$ ein ungerichteter Graph. Sei U eine Teilmenge von V. Wenn zwischen je zwei verschiedenen Knoten in U keine Kante verläuft, also

$$\forall u, v \in U \qquad u \neq v \implies \{u, v\} \notin E$$

gilt, so heißt U eine unabhängige Menge in G.

35 *Antwort*

Seien E und V disjunkte Mengen. Ist E eine Teilmenge von $V \times V$, so heißt das Paar (V, E) ein gerichteter Graph (auf V). Die Elemente von V heißen dann Knoten oder Ecken, und die Elemente von E heißen Kanten.

Potenzmengenkonstruktion

Produktautomat

CLIQUE

inkl. Aussagen zu Entscheidbarkeit

Gödelisierung

Seien $\mathcal{A}_0 = (Q_0,\ \Sigma,\ \delta_0,\ q_0,\ F_0)$ und $\mathcal{A}_1 = (Q_1,\ \Sigma,\ \delta_1,\ q_1,\ F_1)$ deterministische endliche Automaten. Wir definieren den Produktautomaten $\mathcal{A} = (Q,\ \Sigma,\ \delta,\ (q_0,\ q_1),\ F)$ wie folgt:

- $Q := Q_0 \times Q_1$,
- $\delta\colon Q \times \Sigma \to Q$ mit

$$\delta((q,\ q'),\ a) := (\delta_0(q,\ a),\ \delta_1(q',\ a))$$
$$\text{für alle } (q,\ q') \in Q \text{ und alle } a \in \Sigma,$$

- $F := (F_0 \times Q_1) \cup (Q_0 \times F_1)$

Es gilt dann $L(\mathcal{A}) = L(\mathcal{A}_0) \cup L(\mathcal{A}_1)$.

Sei Σ ein Alphabet. Eine Abbildung $G\colon \Sigma^* \to \mathbb{N}_0$ heißt Gödelisierung von Σ^*, falls gilt:

- G ist injektiv,
- G ist Turing-berechenbar,
- $G(\Sigma^*)$ ist Turing-entscheidbar,
- Die auf das Bild von G eingeschränkte Umkehrfunktion $G^{-1}\colon G(\Sigma^*) \to \Sigma^*$ ist Turing-berechenbar.

Die Potenzmengenkonstruktion ist ein Verfahren, um zu einem gegebenen nichtdeterministischen endlichen Automaten

$$\mathcal{N} = (Q_{\mathcal{N}},\ \Sigma,\ \delta_{\mathcal{N}},\ q_{0\mathcal{N}},\ F_{\mathcal{N}})$$

einen äquivalenten deterministischen endlichen Automaten

$$\mathcal{A} = (Q_{\mathcal{A}},\ \Sigma,\ \delta_{\mathcal{A}},\ q_{0\mathcal{A}},\ F_{\mathcal{A}})$$

zu konstruieren: Wir setzen

- $Q_{\mathcal{A}} := \mathfrak{P}(Q_{\mathcal{N}})$,
- $\delta_{\mathcal{A}}(R,\ a) := E\left(\bigcup_{q \in R} \delta_{\mathcal{N}}(q,\ a)\right)$ für alle $R \in Q_{\mathcal{A}}$,
- $q_{0\mathcal{A}} := E(\{q_{0\mathcal{N}}\})$,
- $F_{\mathcal{A}} := \{R \in Q_{\mathcal{A}} \ :\ R \cap F_{\mathcal{N}} \neq \varnothing\}$.

$$\begin{aligned}\mathsf{CLIQUE} \ = \ \{\ \langle G,\ k\rangle \ :\ & k \text{ ist eine natürliche Zahl, und}\\ & G \text{ ist ein ungerichteter Graph,}\\ & \text{der eine Clique}\\ & \text{der Größe } k \text{ enthält}\ \}\end{aligned}$$

CLIQUE ist Turing-entscheidbar. Man beweist dies, indem man $\mathsf{CLIQUE} \in \mathsf{NPC}$ zeigt.

erkannte Sprache eines GNEA

3-SAT

Spezielles Halteproblem $\mathsf{HALT}^{\varepsilon}_{\mathsf{TM}}$
inkl. Aussagen zu Entscheidbarkeit

k-Band-Turingmaschine

$$\mathsf{3\text{-}SAT} = \{ \langle \varphi \rangle : \varphi \text{ ist eine erfüllbare Formel in KNF, deren jede Klausel aus genau drei Literalen besteht} \}$$

Sei $G = (Q_G, \Sigma, \delta_G, q_{start}, q_{acc})$ ein generalisierter nichtdeterministischer Automat, und sei $w \in \Sigma^*$.
Es akzeptiert G das Wort w genau dann, wenn

$$w \in L(\delta_G(q_{start}, q_{acc}) \circ \delta_G(q_1, q_2) \circ \ldots \circ \delta_G(q_k, q_{acc}))$$

für ein geeignetes $k \in \mathbb{N}$ und einen Pfad $q_{start} q_1 q_2 \ldots q_{acc}$ im Transitionsgraphen in G.

Die erkannte Sprache von G wird durch

$$L(G) := \{w \in \Sigma^* : G \text{ akzeptiert } w\}$$

definiert.

Sei $k \in \mathbb{N}$.
Eine k-Band-Turingmaschine ist ein 7-Tupel

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}) \quad ,$$

wobei

- Q eine endliche, nichtleere Menge von Zuständen,
- Σ das Eingabealphabet mit $\sqcup \notin \Sigma$,
- Γ das Band- oder Arbeitsalphabet mit $\Sigma \cup \{\sqcup\} \subseteq \Gamma$,
- $\delta\colon Q \setminus \{q_{acc}, q_{rej}\} \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k$ die Zustandsübergangsfunktion,
- $q_0 \in Q$ der Startzustand,
- $q_{acc} \in Q$ der akzeptierende Zustand,
- $q_{rej} \in Q \setminus \{q_{acc}\}$ der verwerfende Zustand

sind.

$$\mathsf{HALT_{TM}} = \{ \langle M, w \rangle : M \text{ ist eine Turingmaschine, die bei Eingabe } w \text{ hält} \}$$

$\mathsf{HALT_{TM}}$ ist nicht entscheidbar. Man beweist dies, indem man vom als unentscheidbar bekannten allgemeinen Wortproblem für Turingmaschinen reduziert.

Berechnung im Kellerautomaten
Akzeptierende Berechnung im Kellerautomaten

Worttransitionsfunktion für deterministische endliche Automaten

Zusammenhang (Erkennbarkeit)

reguläre Sprache

$\leftrightarrow$

nichtdeterministischer endlicher Automat

Sinn und Zweck des Produktautomaten

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein deterministischer endlicher Automat. Wir definieren die Worttransitionsfunktion $\widehat{\delta}\colon Q \times \Sigma^* \to Q$ für alle $q \in Q$ und alle $w \in \Sigma^*$ durch

$$\begin{aligned} \widehat{\delta}(q, w) &:= q && \text{falls } w = \varepsilon \\ \widehat{\delta}(q, w) &:= \delta\left(\widehat{\delta}(q, v), a\right) && \text{falls } w = va \text{ für ein } a \in \Sigma \text{ und ein } v \in \Sigma^* \end{aligned}$$

Sei $\mathcal{M} = (Q, \Sigma, \$, \perp, \Gamma, \delta, q_0, F)$ ein Kellerautomat, und sei $w \in \Sigma^*$. Wir nennen eine endliche, nichtleere Folge

$$(q_0, w^{(0)}, s_0) \dots (q_m, w^{(m)}, s_m)$$

von Konfigurationen für ein passendes $m \in \mathbb{N}_0$ eine Berechnung in $\mathcal{M}$, wenn gilt:

- q_0 ist Anfangszustand von $\mathcal{M}$,
- $s_0 = \perp$,
- $w^{(0)} = w\$$,
- $w^{(m)} = \varepsilon$,
- Für alle $i \in [m-1]_0$ ist $(q_{i+1}, w^{(i+1)}, s_{i+1})$ eine Folgekonfiguration von $(q_i, w^{(i)}, s_i)$.

Wir nennen eine Berechnung akzeptierend, wenn $q_m \in F$ gilt.

Seien $\mathcal{A}_0 = (Q_0, \Sigma, \delta_0, q_0, F_0)$ und $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ deterministische endliche Automaten und $\mathcal{A} = (Q, \Sigma, \delta, (q_0, q_1), F_1)$ der wie in Definition 2.11 konstruierte Produktautomat. Dann gilt

$$L(\mathcal{A}) = L(\mathcal{A}_0) \cup L(\mathcal{A}_1) \quad ,$$

der Produktautomat erkennt also die Vereinigung der durch die Einzelautomaten erkannten Sprachen.

Eine Sprache L ist genau dann regulär, wenn es einen nichtdeterministischen endlichen Automaten $\mathcal{N}$ gibt, der sie erkennt, d.h. für den $L = L(\mathcal{N})$ gilt.

HORN-SAT

inkl. Aussagen zu Entscheidbarkeit

normierte Turingmaschine

Gleichheitsrelation auf Zuständen eines DEA

inkl. was für eine Art von Relation ist das?

VERTEX-COVER

Eine Turingmaschine $(Q,\ \Sigma,\ \Gamma,\ \delta,\ q_0,\ q_{acc},\ q_{rej})$ heißt normiert, wenn gilt

- $Q = \{0, \ldots, n\}$ für ein $n \in \mathbb{N}_{>2}$,
- $q_0 = 0$,
- $\Sigma = \{0,\ 1\}$,
- $\Gamma = \{0,\ 1,\ 2\}$,
- $q_{acc} = 1$,
- $q_{rej} = 2$.

Bei einer normierten Turingmaschine kodieren wir das $\sqcup$-Symbol auf dem Arbeitsband mit 2, sowie die Kopfbewegung L durch 0 und R durch 1.

$$\begin{aligned}\textsf{HORN-SAT} \quad = \quad \Big\{ \quad \langle \varphi \rangle \quad : \quad & \varphi \text{ ist eine erfüllbare Formel in KNF,} \\ & \text{deren jede Klausel höchstens ein} \\ & \text{nichtnegiertes Literal besitzt } \Big\}\end{aligned}$$

HORN-SAT ist entscheidbar. Man beweist dies, indem...

Hier fehlt noch was!

$$\begin{aligned}\textsf{VERTEX-COVER} \quad = \quad \Big\{ \quad \langle G,\ k \rangle \quad : \quad & k \text{ ist eine natürliche Zahl, und} \\ & G \text{ ist ein Graph, in dem es} \\ & \text{eine Knotenüberdeckung} \\ & \text{der Mächtigkeit } k \text{ gibt } \Big\}\end{aligned}$$

Sei $(Q,\ \Sigma,\ \delta,\ q_0,\ F)$ ein deterministischer endlicher Automat. Wir definieren die Relation $\approx\ \subseteq Q \times Q$, indem wir für alle $p,\ q \in Q$ festlegen

$$p \approx q \ :\Longleftrightarrow \quad \forall x \in \Sigma^* \quad \Big(\widehat{\delta}(p,\ x) \in F \iff \widehat{\delta}(q,\ x) \in F\Big) \quad .$$

Dies ist eine Äquivalenzrelation.

Knotenüberdeckung

RELPRIME

inkl. Aussagen zu Entscheidbarkeit

Abschluss der regulären Sprachen unter Kleene-Stern: Automatenkonstruktion

Redundanz zusätzlicher Bänder

$$\mathsf{RELPRIME} = \{ \langle m, n \rangle : m \text{ und } n \text{ sind teilerfremde natürliche Zahlen} \}$$

RELPRIME ist entscheidbar. Man beweist dies, indem man sich des euklidischen Algorithmus erinnert.

Sei $G = (V, E)$ ein Graph. Sei $U \subseteq V$ eine Teilmenge der Knoten von G.

Es heißt U eine Knotenüberdeckung von G, falls jede Kante von G mit mindestens einem Knoten aus U verbunden ist.

Sei $k \in \mathbb{N}_{\geqslant 1}$.

Jede k-Band-Turingmaschine kann durch eine 1-Band-Turingmaschine simuliert werden.

Sei $\mathcal{N}_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ ein nichtdeterministischer endlicher Automat.
Für den nichtdeterministischen endlichen Automaten
$\mathcal{N} := (Q, \Sigma, \delta, q_0, F)$ mit

- $Q = Q_1 \dot\cup \{q_0\}$,
- $\delta\colon Q \times \Sigma \to Q$ mit

$$\begin{aligned}
\delta(q, a) &:= \{q_1\} && \text{falls } q = q_0 \text{ und } a = \varepsilon \\
\delta(q, a) &:= \varnothing && \text{falls } q = q_0 \text{ und } a \neq \varepsilon \\
\delta(q, a) &:= \delta_1(q, a) && \text{falls } q \in Q_1 \setminus F_1 \ \vee\ (q \in F_1 \ \wedge\ a \neq \varepsilon) \\
\delta(q, a) &:= \delta_1(q, a) \cup \{F_1\} && \text{falls } q \in F_1 \text{ und } a = \varepsilon
\end{aligned}$$

  für alle $(q, a) \in Q \times \Sigma$,
- $F := F_1 \cup \{q_0\}$

gilt $L(\mathcal{N}) = L(\mathcal{N}_1)^*$.

Laufzeitfunktion

erkannte Sprache eines Kellerautomaten

kontextfreie Grammatik

HAMPATH

Sei $\mathcal{A} = (Q,\ \Sigma,\ \$,\ \perp,\ \Gamma,\ \delta,\ q_0,\ F)$ ein Kellerautomat.

Wir definieren die durch den Kellerautomaten $\mathcal{A}$ erkannte Sprache durch

$$L(\mathcal{A}) \quad = \quad \Big\{ \quad w \in \Sigma^* \quad : \quad \text{es gibt eine akzeptierende Berechnung von } \mathcal{A} \text{ auf } w \Big\}$$

Sei $\mathcal{M}$ eine deterministische Turingmaschine.

Die Laufzeitfunktion zu $\mathcal{M}$ ist eine Funktion $f_{\mathcal{M}} \colon \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ mit

$$f_{\mathcal{M}}(n) \quad = \quad \max \Big\{ \quad \ell \quad : \quad \ell \text{ ist Länge einer Berechnung eines Wortes } w \in \Sigma^* \text{ auf } \mathcal{M}, \text{ für das } |w| = n \text{ gilt} \Big\}$$

$$\mathsf{HAMPATH} \quad = \quad \Big\{ \quad \langle G,\ s,\ t \rangle \quad : \quad G \text{ ist ein Graph, } s \text{ und } t \text{ sind Knoten in } G\text{, und es gibt in } G \text{ einen Hamiltonpfad mit Startknoten } s \text{ und Endknoten } t \Big\}$$

Eine kontextfreie Grammatik ist ein 4-Tupel

$$(N,\ T,\ P,\ S) \quad ,$$

wobei

- N eine endliche, nichtleere Menge von Nichtterminalsymbolen,
- T eine endliche Menge von Terminalsymbolen,
- $P \subseteq N \times (N \cup T)^*$ eine endliche Menge von Produktionen,
- $S \in N$ das Startsymbol

sind. Dass N, T und P endlich sind, fordern wir, damit die Umwandlung in Chomsky-Normalform in endlicher Zeit möglich ist.

Merkmal einer kontextfreien Grammatik ist also, dass in jeder Regel auf der linken Seite genau ein Nichtterminalsymbol (und sonst nichts, insbesondere also auch kein Terminalsymbol) steht, aus dem eine beliebig lange Folge aus Nichtterminal- und Terminalsymbolen abgeleitet wird.

reguläre Sprache

Abschluss der regulären Sprachen unter Vereinigung: Automatenkonstruktion

Nichtdeterministischer endlicher Automat

$\mathsf{EQ}_{\mathsf{CFG}}$

inkl. Aussagen zu Entscheidbarkeit

Seien $\mathcal{N}_1 = (Q_1,\ \Sigma,\ \delta_1,\ q_1,\ F_1)$ und $\mathcal{N}_2 = (Q_2,\ \Sigma,\ \delta_2,\ q_2,\ F_2)$ nichtdeterministische endliche Automaten.
Für den nichtdeterministischen endlichen Automaten
$\mathcal{N} := (Q,\ \Sigma,\ \delta,\ q_0,\ F)$ mit

- $Q = Q_1 \mathbin{\dot\cup} Q_2 \mathbin{\dot\cup} \{q_0\}$,
- $\delta\colon Q \times \Sigma \to Q$ mit

$$\begin{aligned}
\delta(q,\ a) &:= \{q_1,\ q_2\} && \text{falls } q = q_0 \text{ und } a = \varepsilon \\
\delta(q,\ a) &:= \varnothing && \text{falls } q = q_0 \text{ und } a \in \Sigma \\
\delta(q,\ a) &:= \delta_1(q,\ a) && \text{falls } q \in Q_1 \\
\delta(q,\ a) &:= \delta_2(q,\ a) && \text{falls } q \in Q_2
\end{aligned}$$

  für alle $(q,\ a) \in Q \times \Sigma$,
- $F := F_1 \cup F_2$

gilt $L(\mathcal{N}) = L(\mathcal{N}_1) \cup L(\mathcal{N}_2)$.

Eine Sprache heißt regulär, wenn es eine reguläre Grammatik gibt, die sie erzeugt.

$$\mathsf{EQ}_{\mathsf{CFG}} = \Big\{ \langle G_1,\ G_2 \rangle : G_1,\ G_2 \text{ sind kontextfreie Grammatiken, und es gilt } L(G_1) = L(G_2) \Big\}$$

$\mathsf{EQ}_{\mathsf{CFG}}$ ist nicht Turing-entscheidbar. Man beweist dies, indem man vom als unentscheidbar nachzuweisenden Problem $\mathsf{SUB}_{\mathsf{CFG}}$ reduziert.

Ein nichtdeterministischer endlicher Automat ist ein 5-Tupel

$$(Q,\ \Sigma,\ \delta,\ q_0,\ F) \quad ,$$

wobei

- Q eine endliche, nichtleere Menge von Zuständen,
- Σ ein Alphabet, laut Skript endlich!
- $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}$,
- $\delta\colon Q \times \Sigma_\varepsilon \to \mathfrak{P}(Q)$ die Zustandsübergangsfunktion,
- $q_0 \in Q$ der Startzustand
- $F \subseteq Q$ die Menge der akzeptierenden Zustände

sind.

Man beachte, dass es keine akzeptierenden Zustände geben muss, also $F = \varnothing$ gelten kann.

Satz 65

Erhalt der Aussagekraft nach Kodierung

Definition 66

Äquivalenzklassenautomat

Definition 67

Platzbedarfsfunktion einer deterministischen Turingmaschine

Definition 68

Platzbedarfsfunktion einer nichtdeterministischen Turingmaschine

Sei $\mathcal{A} = (Q,\ \Sigma,\ \delta,\ q_0,\ F)$ ein deterministischer endlicher Automat.

Wir definieren den Äquivalenzklassenautomaten $\mathcal{A}/_{\approx}$ durch

$$\mathcal{A}/_{\approx} = (Q',\ \Sigma,\ \delta',\ q_0',\ F') \quad ,$$

wobei

- $Q' := \{[p] \ :\ p \in Q\}$ die Menge der Zustände,
- $\delta' \colon Q' \times \Sigma \to Q'$ die Zustandsübergangsfunktion mit

$$\delta'([p],\ a) := [\delta(p,\ a)] \quad \text{für alle } p \in Q \text{ und alle } a \in \Sigma,$$

- $q_0' := [q_0]$ der Startzustand,
- $F' := \{[p] \ :\ p \in F\}$ die Menge der akzeptierenden Zustände

sind.

Sei Σ ein Alphabet und sei $\varphi \colon \Sigma \to \{0,\ 1\}$ eine Kodierungsfunktion.
Sei $L \subseteq \Sigma^*$. Dann gelten:

- L ist Turing-entscheidbar $\iff$ $\varphi(L)$ ist Turing-entscheidbar
- L ist Turing-erkennbar $\iff$ $\varphi(L)$ ist Turing-erkennbar

Sei $\mathcal{M}$ eine nichtdeterministische Turingmaschine.

Die Platzbedarfsfunktion zu $\mathcal{M}$ ist eine Funktion $S_{\mathcal{M}} \colon \mathbb{N} \to \mathbb{N}$ mit

$$\begin{aligned} S_{\mathcal{M}}(n) \quad = \quad \max\Big\{ \quad \ell \quad : \quad & \ell \text{ ist Anzahl der Bandzellen,} \\ & \text{die bei einer längsten Berechnung} \\ & \text{eines Wortes } w \in \Sigma^* \text{ auf } \mathcal{M}, \\ & \text{für das } |w| = n \text{ gilt,} \\ & \text{beschrieben werden} \Big\} \end{aligned}$$

Sei $\mathcal{M}$ eine deterministische Turingmaschine.

Die Platzbedarfsfunktion zu $\mathcal{M}$ ist eine Funktion $S_{\mathcal{M}} \colon \mathbb{N} \to \mathbb{N}$ mit

$$\begin{aligned} S_{\mathcal{M}}(n) \quad = \quad \max\Big\{ \quad \ell \quad : \quad & \ell \text{ ist Anzahl der Bandzellen,} \\ & \text{die bei der Berechnung eines Wortes} \\ & w \in \Sigma^* \text{ auf } \mathcal{M}, \text{ für das } |w| = n \text{ gilt,} \\ & \text{beschrieben werden} \Big\} \end{aligned}$$

SAT

inkl. Aussagen zu Entscheidbarkeit

Reduzierung von Sprachen

Polynomialzeitreduktion von Sprachen

Zusammenhang (Erkennbarkeit)

kontextfreie Sprache

$\leftrightarrow$

Kellerautomat

Seien Σ, Δ Alphabete und $L_1 \subseteq \Sigma^*$ und $L_2 \subseteq \Delta^*$ Sprachen.

Wir nennen L_1 auf L_2 reduzierbar, wenn es eine Turing-berechenbare Funktion $f\colon \Sigma^* \to \Delta^*$ gibt, so dass für alle $w \in \Sigma^*$ die Äquivalenz

$$w \in L_1 \iff f(w) \in L_2$$

gilt.

Ist L_1 auf L_2 reduzierbar, so schreiben wir $L_1 \leqslant L_2$ mittels f.

$$\mathsf{SAT} \;=\; \{\; \langle\varphi\rangle \;:\; \varphi \text{ ist eine erfüllbare Formel in KNF} \;\}$$

SAT ist entscheidbar. Man beweist dies, indem man eine nichtdeterministische Turingmaschine angibt, die eine erfüllende Belegung „rät“ und diese anschließend verifiziert.

Eine Sprache L ist genau dann kontextfrei, wenn es einen Kellerautomaten $\mathcal{N}$ gibt, der sie erkennt, d.h. für den $L = L(\mathcal{N})$ gilt.

Seien Σ, Δ Alphabete und $L_1 \subseteq \Sigma^*$ und $L_2 \subseteq \Delta^*$ Sprachen.

Wir nennen L_1 in Polynomialzeit auf L_2 reduzierbar, wenn es eine Turing-berechenbare Funktion $f\colon \Sigma^* \to \Delta^*$ gibt, deren Zeitbedarf polynomiell in der Eingabelänge ist, so dass für alle $w \in \Sigma^*$ die Äquivalenz

$$w \in L_1 \iff f(w) \in L_2$$

gilt.

Ist L_1 in Polynomialzeit auf L_2 reduzierbar, so schreiben wir $L_1 \leqslant_p L_2$ mittels f.

Konfigurationen einer Turingmaschine

Abschluss der regulären Sprachen unter Konkatenation: Automatenkonstruktion

- In jedem deterministischen endlichen Automaten kann ε zum Alphabet hinzugefügt werden, ohne die erkannte Sprache zu verändern.
- In jedem nichtdeterministischen endlichen Automaten gibt es ε-Übergänge.
- In jedem nichtdeterministischen endlichen Automaten kann die Zustandsübergangsfunktion δ durch die Zustandsübergangsrelation

  $$\Delta := \{(q,\ a,\ q') \in Q \times (\Sigma \cup \{\varepsilon\}) \times Q\ :\ \delta(q,\ a) = q'\}$$

  ersetzt werden, ohne die erkannte Sprache zu verändern.

- Es genügt in der Definition eines deterministischen endlichen Automaten zu fordern, dass die Zustandsübergangsrelation $\delta \subseteq Q \times \Sigma \times Q$ eindeutig ist, d.h. sie muss nicht total sein.
- Es gibt einen deterministischen endlichen Automaten $(Q,\ \Sigma,\ \delta,\ q_0,\ \varnothing)$.
- Es gibt einen deterministischen endlichen Automaten $(\varnothing,\ \Sigma,\ \delta,\ q_0,\ F)$.

Seien $\mathcal{N}_1 = (Q_1,\ \Sigma,\ \delta_1,\ q_1,\ F_1)$ und $\mathcal{N}_2 = (Q_2,\ \Sigma,\ \delta_2,\ q_2,\ F_2)$ nichtdeterministische endliche Automaten.
Für den nichtdeterministischen endlichen Automaten
$\mathcal{N} := (Q,\ \Sigma,\ \delta,\ q_1,\ F_2)$ mit

- $Q = Q_1 \mathbin{\dot\cup} Q_2$,
- $\delta\colon Q \times \Sigma \to Q$ mit

$$\begin{aligned}
\delta(q,\ a) &:= \delta_2(q,\ a) && \text{falls } q \in Q_2 \\
\delta(q,\ a) &:= \delta_1(q,\ a) && \text{falls } q \in Q_1 \text{ und } (a \neq \varepsilon \text{ oder } q \notin F_1) \\
\delta(q,\ a) &:= \delta_1(q,\ a) \cup \{q_2\} && \text{falls } q \in F_1 \text{ und } a = \varepsilon
\end{aligned}$$

für alle $(q,\ a) \in Q \times \Sigma$

gilt $L(\mathcal{N}) = L(\mathcal{N}_1) \circ L(\mathcal{N}_2)$.

Sei $\mathcal{M} = (Q,\ \Sigma,\ \Gamma,\ \delta,\ q_0,\ q_{acc},\ q_{rej})$ eine Turingmaschine. Wir definieren

- Für eine Eingabe $w \in \Sigma^*$ ist $q_0 w$ eine Anfangskonfiguration, falls $w \neq \varepsilon$ und $q_{0\sqcup}$, falls $w = \varepsilon$.
- Sind $u,\ v \in \Gamma^+$, so nennen wir $u q_{acc} v$ und $u q_{rej} v$ Haltekonfigurationen.
- Zu einer Konfiguration $u q a v$ mit $q \in Q \setminus \{q_{acc},\ q_{rej}\}$, $q' \in Q$ und $a,\ b \in \Gamma$ sowie $u,\ v \in \Gamma^*$ gibt es folgende Möglichkeiten einer Folgekonfiguration:

$$\begin{aligned}
&u b q' v && \text{falls } \delta(q,\ a) = (q',\ b,\ R) \text{ und } v \neq \varepsilon \\
&u b q'_{\sqcup} && \text{falls } \delta(q,\ a) = (q',\ b,\ R) \text{ und } v \neq \varepsilon \\
&u' q' c b v && \text{falls } \delta(q,\ a) = (q',\ b,\ L) \text{ und } u = u'c \\
& && \text{für ein } u' \in \Gamma^* \text{ und ein } c \in \Gamma \\
&q' b v && \text{falls } \delta(q,\ a) = (q',\ b,\ L) \text{ und } u = \varepsilon
\end{aligned}$$

- falsch
- wahr
- falsch

- falsch
- falsch
- wahr prüfen!

Zusammenhang

kontextfreie Grammatik

$\leftrightarrow$

kontextfreie Grammatik
in Chomsky-Normalform

Zusammenhang

kontextfreie Grammatik

$\leftrightarrow$

kontextfreie Grammatik
ohne ε- und Kettenregel

Entscheidbarkeit von PKP

Zusammenhang

MPKP

$\leftrightarrow$

PKP

78 *Antwort*

Zu jeder kontextfreien Grammatik $G = (N, T, P, S)$ gibt es eine kontextfreie Grammatik G', die weder Produktionen der Form $A \to \varepsilon$ noch solche der Form $A \to B$ enthält (A, B seien Nichtterminalsymbole) und für die

$$L(G) \setminus \{\varepsilon\} = L(G')$$

gilt.

77 *Antwort*

Zu jeder kontextfreien Grammatik G gibt es eine kontextfreie Grammatik G' in Chomsky-Normalform, für die

$$L(G) \setminus \{\varepsilon\} = L(G')$$

gilt.

80 *Antwort*

MPKP ist auf PKP reduzierbar.

79 *Antwort*

PKP ist nicht Turing-entscheidbar.

Worttransitionsfunktion für nichtdeterministische endliche Automaten

Eigenschaften der Worttransitionsfunktion für nichtdeterministische endliche Automaten

NP-Vollständigkeit

NP-Schwere

Sei $\mathcal{N} = (Q,\ \Sigma,\ \delta,\ q_0,\ F)$ ein nichtdeterministischer endlicher Automat, sei I eine Indexmenge, und sei $\widehat{\delta}$ die Worttransitionsfunktion für $\mathcal{N}$. Sei $\{A_i : i \in I\} \in \mathfrak{P}(Q)$, sowie $A \in \mathfrak{P}(Q)$. Weiter seien $u,\ v,\ w \in \Sigma^*$. Dann gelten

$$\widehat{\delta}\left(\bigcup_{i \in I} A_i,\ w\right) = \bigcup_{i \in I} \widehat{\delta}(A_i,\ w)$$

und

$$\widehat{\delta}(A,\ uv) = \widehat{\delta}\left(\widehat{\delta}(A,\ u),\ v\right) \quad .$$

Sei $\mathcal{N} = (Q,\ \Sigma,\ \delta,\ q_0,\ F)$ ein nichtdeterministischer endlicher Automat.

Wir definieren die Worttransitionsfunktion $\widehat{\delta}\colon \mathfrak{P}(Q) \times \Sigma^* \to \mathfrak{P}(Q)$ durch

$$\begin{aligned} \widehat{\delta}(R,\ w) &:= R && \text{falls } w = \varepsilon \\ \widehat{\delta}(R,\ w) &:= \bigcup_{q \in \widehat{\delta}(R,\ v)} \delta(q,\ a) && \text{falls } w = va \text{ für ein } a \in \Sigma \text{ und ein } v \in \Sigma^* \end{aligned}$$

für alle $R \in \mathfrak{P}(Q)$ und alle $w \in \Sigma^*$.

Sei Σ ein Alphabet. Eine Sprache $L_2 \subseteq \Sigma^*$ heißt NP-schwer, wenn $L_1 \leqslant_p L_2$ für alle $L_1 \in \mathsf{NP}$ gilt.

Sei Σ ein Alphabet. Eine Sprache $L_2 \subseteq \Sigma^*$ heißt NP-vollständig, wenn

- $L_2 \in \mathsf{NP}$
- Für alle $L_1 \in \mathsf{NP}$ gilt $L_1 \leqslant_p L_2$

Definition 85

Äquivalenzrelation auf Klassen von Zuständen eines deterministischen endlichen Automaten

Satz 86

Verifizierung in polynomieller Zeit

Definition 87

polynomial zeitbeschränkte Turingmaschine

Menge von polynomial zeitbeschränkten Turingmaschinen

Definition 88

Notation von kodierten Objekten

Sei $L \in \mathsf{NP}$ eine Sprache.

Dann wird L durch einen Verifikator in polynomieller Zeit verifiziert.

Sei $\mathcal{A} = (Q,\ \Sigma,\ \delta,\ q_0,\ F)$ ein deterministischer endlicher Automat und $\mathcal{A}/_{\approx} = (Q',\ \Sigma,\ \delta',\ q_0',\ F')$ sein Äquivalenzklassenautomat.

Wir definieren die Äquivalenz $\sim$ von zwei $\approx$-Klassen, indem wir für alle $p,\ q \in Q$ festlegen

$$[p] \sim [q] \ :\Longleftrightarrow \quad \forall w \in \Sigma^* \ \big(\delta'([p], w) \in F \iff \delta'([q], w) \in F\big) \quad .$$

Seien $O_1,\ O_2$ beliebige Objekte.

Mit $\langle\cdot\rangle$ bezeichnen wir eine beliebige, aber feste Kodierungsfunktion, die Objekte nach $\{0,\ 1\}^*$ abbildet. Dann ist $\langle O_1\rangle$ die Kodierung des Objekts O_1.

Zur Vereinfachung der Schreibweise vereinbaren wir außerdem

$$\langle O_1,\ O_2\rangle := \langle (O_1,\ O_2)\rangle$$

als Kodierung des Paars $(O_1,\ O_2)$.

Sei $f\colon \mathbb{N} \to \mathbb{N}$ eine Funktion, also $f \in \mathbb{N}^{\mathbb{N}}$. Wir definieren

$$\mathsf{DTIME}(f) \ = \ \Big\{ \ L(M) \ : \ \begin{array}{l} \text{es gibt eine Turingmaschine } M \text{ so,} \\ \text{dass } L(M) \text{ Turing-entscheidbar ist und} \\ \mathcal{O}(f) = f_M \text{ gilt.} \ \Big\} \end{array}$$

Sei $\mathcal{F}$ eine Teilmenge von $\mathbb{N}^{\mathbb{N}}$. Wir definieren

$$\mathsf{DTIME}(\mathcal{F}) := \bigcup_{f \in \mathcal{F}} \mathsf{DTIME}(f) \quad .$$

Akzeptanz eines Wortes durch eine Turingmaschine

Ablehnen eines Wortes durch eine Turingmaschine

Zusammenhang

nichtdeterministischer endlicher Automat

$\leftrightarrow$

regulärer Ausdruck

Charakterisierung regulärer Sprachen

Vergleich der Ausdruckskraft von nichtdeterministischen und deterministischen endlichen Automaten

inkl. Beweisidee

Sei $\mathcal{M} = (Q,\ \Sigma,\ \Gamma,\ \delta,\ q_0,\ q_{acc},\ q_{rej})$ eine Turingmaschine. Sei $w \in \Sigma^*$.

Die Turingmaschine $\mathcal{M}$ lehnt das Wort w genau dann ab, wenn es ein $m \in \mathbb{N}_0$ so gibt, dass die Folge

$$K_0,\ K_1,\ \ldots,\ K_m$$

der beim Verarbeiten von w durchlaufenen Konfigurationen endlich ist und K_0 eine Anfangskonfiguration ist und für jedes $i \in \mathbb{N}_{<m}$ die Konfiguration K_{i+1} eine Folgekonfiguration von K_i ist und K_m eine verwerfende Konfiguration ist.

Sei $\mathcal{M} = (Q,\ \Sigma,\ \Gamma,\ \delta,\ q_0,\ q_{acc},\ q_{rej})$ eine Turingmaschine. Sei $w \in \Sigma^*$.

Die Turingmaschine $\mathcal{M}$ akzeptiert das Wort w genau dann, wenn es ein $m \in \mathbb{N}_0$ so gibt, dass die Folge

$$K_0,\ K_1,\ \ldots,\ K_m$$

der beim Verarbeiten von w durchlaufenen Konfigurationen endlich ist und K_0 eine Anfangskonfiguration ist und für jedes $i \in \mathbb{N}_{<m}$ die Konfiguration K_{i+1} eine Folgekonfiguration von K_i ist und K_m eine akzeptierende Konfiguration ist.

Zu jedem nichtdeterministischen endlichen Automaten $\mathcal{N}$ existiert ein äquivalenter deterministischer endlicher Automat $\mathcal{A}$.

Nichtdeterministische und deterministische endliche Automaten besitzen dieselbe Ausdruckskraft.

Man beweist die eine Richtung, indem man ausgehend von $\mathcal{N}$ die Potenzmengenkonstruktion durchführt. Die andere Richtung ist nach einem Blick auf die Definitionen klar.

Sei Σ ein Alphabet und $L \subseteq \Sigma^*$ eine Sprache.

L ist genau dann regulär, wenn es einen nichtdeterministischen endlichen Automaten $\mathcal{A}$ gibt, für den $L = L(\mathcal{A})$ gilt; und Letzteres ist genau dann der Fall, wenn es einen regulären Ausdruck $\mathcal{R}$ gibt, für den $L = L(\mathcal{R})$ gilt.

Sei Σ ein Alphabet.

Man bestimme den Wahrheitsgehalt der folgenden Aussagen.

- $L(\varnothing^*) = \varnothing$
- $L(\varnothing^*) = \{\varepsilon\}$
- $L(\varnothing^*) = \Sigma$
- $L(\varnothing^*) = \Sigma^*$
- $L(\varnothing^*)$ ist nicht definiert

Alphabet, Buchstabe, Symbol

Wort über einem Alphabet, Sprache

Länge eines Wortes, leeres Wort ε

Seien $f, g \in (\mathbb{R}_{>0})^{\mathbb{N}_0}$. Dann gilt:

$\log(f) \in \mathcal{O}(\log(g)) \quad \Longrightarrow \quad f \in \mathcal{O}(g)$.

Abschlusseigenschaft von $\leqslant_p$

94 *Antwort*

Ein Alphabet ist eine nichtleere Menge Σ mit $\varepsilon \notin \Sigma$, deren Elemente als Buchstaben oder Symbole bezeichnet werden.

Ist Σ ein Alphabet und $w \in \Sigma^k$ für ein geeignetes $k \in \mathbb{N}_0$, so heißt w ein Wort über Σ, und k heißt seine Länge, geschrieben $|w|$. Das Wort der Länge 0 heißt leeres Wort und wird mit ε bezeichnet.

Wo nötig, benennen wir einen Buchstaben eines Wortes $w \in \Sigma^k$ mittels der Bezeichnung $w^{(i)}$ für ein geeignetes $i \in [k-1]_0$.

Die Menge Σ^* aller Wörter über Σ ist durch

$$\Sigma^* = \bigcup_{k \in \mathbb{N}_0} \Sigma^k$$

definiert.

Jede Teilmenge von Σ^* heißt eine Sprache über Σ.

93 *Antwort*

- falsch
- wahr
- falsch
- falsch
- falsch

96 *Antwort*

Seien Σ, Δ Alphabete, und seien $L_1 \subseteq \Sigma^*$ und $L_2 \subseteq \Delta^*$ Sprachen. Dann gilt die Implikation

$$L_2 \in \mathsf{P} \wedge L_1 \leqslant_p L_2 \implies L_1 \in \mathsf{P} \quad .$$

Man beweist dies, indem man einen Entscheider M für L_2 und eine in Polynomialzeit berechenbare Transformationsfunktion f von L_1 auf L_2 angibt. Dann entscheidet eine Turingmaschine $\mathcal{N}$ die Sprache L_1 in polynomieller Laufzeit wie folgt:

- Berechne $f(w)$ für ein $w \in \Sigma^*$.
- Lasse M auf $f(w)$ laufen und gib die Ausgabe von M aus.

95 *Antwort*

falsch

https://cs.stackexchange.com/questions/117577/which-grows-faster-factorial-or-double-exponentiation#comment248585_117583
Letzter Zugriff am 27.11.2019

Asymptotische obere Schranke

$$\mathcal{O}(g) = f$$

$$f \in \mathcal{O}(g)$$

Zusammenhang

Aufzählbarkeit

$\leftrightarrow$

Turing-Erkennbarkeit

Berechnung einer Turingmaschine

kontextfreie Sprache

Eine Sprache L ist Turing-erkennbar genau dann, wenn sie aufzählbar ist.

Man beweist die eine Richtung, indem man eine aufzählende Turingmaschine E für L laufen lässt und jedes aufgezählte Wort w_i mit einem vorgegebenen Wort w vergleicht.

Die andere Richtung beweist man, indem man aus einer L erkennenden Turingmaschine $\mathcal{M}$ eine L aufzählende Turingmaschine E konstruiert. Dazu lässt man $\mathcal{M}$ für alle $i \in \mathbb{N}_{>0}$ auf den ersten i Elementen einer beliebigen, aber festen Abzählung von Σ^* jeweils i Schritte laufen und gibt w_j auf dem Ausgabeband aus, falls es ein $j \in \mathbb{N}$ so gibt, dass w_j von $\mathcal{M}$ akzeptiert wird.

Eine Sprache heißt kontextfrei, wenn es eine kontextfreie Grammatik gibt, die sie erzeugt.

Seien $f,\ g\colon \mathbb{N}_0 \to \mathbb{R}_{>0}$ Funktionen. Wir definieren

$$\mathcal{O}(g) = f \ :\Longleftrightarrow \quad \exists c \in \mathbb{R}_{>0}\ \exists n_0 \in \mathbb{N}\ \forall n \in \mathbb{N}_{\geqslant n_0} \quad f(n) \leqslant c \cdot g(n)$$

und nennen g dann eine asymptotische obere Schranke von f.

Eine „ungefährlichere“ Schreibweise hierfür ist

$$f \in \mathcal{O}(g) \quad ,$$

denn das Gleichheitszeichen suggeriert eine nicht bestehende Symmetrie. TODO: saubere mengentheoretische Definition schreiben, etwa so:

$$\mathcal{O}(g) = \left\{ f \in (\mathbb{R}_{>0})^{\mathbb{N}_0} : \exists c \in \mathbb{R}_{>0}\ \exists n_0 \in \mathbb{N}\ \forall n \in \mathbb{N}_{\geqslant n_0}\ \ f(n) \leqslant c \cdot g(n) \right\}$$

https://de.wikipedia.org/w/index.php?title=Landau-Symbole&oldid=162680389

Letzter Zugriff am 03.04.2017

Sei $\mathcal{M} = (Q,\ \Sigma,\ \Gamma,\ \delta,\ q_0,\ q_{acc},\ q_{rej})$ eine Turingmaschine. Sei $w \in \Sigma^*$.

Eine Berechnung von $\mathcal{M}$ auf w ist eine (nicht notwendigerweise endliche) Folge

$$K_0,\ K_1,\ \ldots$$

von Konfigurationen, für die gilt, dass K_0 eine Anfangskonfiguration ist und für jedes $i \in \mathbb{N}_0$ die Konfiguration K_{i+1}, sofern existent, eine Folgekonfiguration von K_i ist.

von einer Grammatik erzeugte Sprache

(Chomsky-)Grammatik

Zusammenhang

kontextfreie Sprache

$\leftrightarrow$

deterministischer endlicher Automat

Zusammenhang

reguläre Ausdrücke

$\leftrightarrow$

reguläre Sprachen

inkl. Beweisidee

Eine Grammatik ist ein 4-Tupel

$$(N,\ T,\ P,\ S) \quad ,$$

wobei

- N eine nichtleere, endliche Menge von Nichtterminalsymbolen,
- T eine nichtleere, endliche Menge von Terminalsymbolen mit $N \cap T = \varnothing$ und $\varepsilon \notin N \cup T$,
- P eine nichtleere, endliche Menge von Produktionen oder Regeln mit

  $$P \subseteq \{\ (p,\ q)\ :\ p \in (N \cup T)^* \circ N \circ (N \cup T)^* \ \wedge\ q \in (N \cup T)^*\ \} \quad ,$$

  die meist in der Form $p \to q$ geschrieben werden,
- $S \in N$ das Startsymbol

sind.

nach Volker Claus / Andreas Schwill (1988): Duden „Informatik". Ein Sachlexikon für Studium und Praxis. Mannheim (Dudenverlag), S. 250, sowie
https://en.wikipedia.org/w/index.php?title=Formal_grammar&oldid=759905094#The_syntax_of_grammars
Letzter Zugriff am 30.03.2017

Sei $G = (N,\ T,\ P,\ S)$ eine Grammatik. Dann ist die von G erzeugte Sprache $L(G)$ definiert durch

$$L(G) = \{w \in T^* \ :\ S \Longrightarrow_G^* w\} \quad .$$

Sei L eine Sprache.

Es ist L genau dann regulär, wenn es einen regulären Ausdruck R gibt, für den $L = L(R)$ gilt.

Man beweist die eine Richtung, indem man zu einem gegebenen regulären Ausdruck R einen nichtdeterministischen endlichen Automaten $\mathcal{A}$ mit $L = L(\mathcal{A})$ konstruiert, und die andere Richtung, indem man zu einer gegebenen regulären Sprache L einen generalisierten nichtdeterministischen Automaten G mit $L = L(G)$ konstruiert, aus welchem man einen regulären Ausdruck R mit $L = L(R)$ gewinnt.

Zu jedem deterministischen endlichen Automaten $\mathcal{A}$ existiert eine kontextfreie Grammatik G mit $L(G) = L(\mathcal{A})$.

Es gibt kontextfreie Sprachen, die von keinem deterministischen endlichen Automaten erkannt werden können. Standardbeispiel hierfür ist die Sprache $\{a^n\, b^n\ :\ n \in \mathbb{N}_0\} \subseteq \{a,\ b\}^*$.

Es seien

$$\mathcal{N}_1 = (Q_1,\ \Sigma_1,\ \delta_1,\ q_1,\ F_1)$$

und

$$\mathcal{N}_2 = (Q_2,\ \Sigma_2,\ \delta_2,\ q_2,\ F_2)$$

äquivalente endliche Automaten – entweder beide deterministisch oder beide nichtdeterministisch.

Man bestimme den Wahrheitsgehalt der folgenden Aussagen.

- Es gilt $\Sigma_1 = \Sigma_2$.
- Es gilt $|Q_1| = |Q_2|$.
- Es gilt $Q_1 = Q_2$.

Polynomiell zeitberechenbare Funktion

PKP

Postsches Korrespondenzproblem

MPKP

Modifiziertes Postsches Korrespondenzproblem

Minimierung eines deterministischen endlichen Automaten

Seien Σ, Δ Alphabete, und sei $f\colon \Sigma^* \to \Delta^*$ eine Funktion.

Wir nennen f polynomiell zeitberechenbar, wenn es eine Turingmaschine M gibt, die polynomielle Laufzeit besitzt und beim Lesen einer Eingabe $w \in \Sigma^*$ mit dem ausschließlichen Bandinhalt $f(w)$ hält.

- falsch
- falsch
- falsch

Sei $(Q, \Sigma, \delta, q_0, F)$ ein deterministischer endlicher Automat. Der folgende Algorithmus berechnet die Relation $\approx$ aus Definition 7.21:

(1) Notiere alle Paare $(p, q) \in Q \times Q$ als unmarkiert.

(2) Falls $p \in F$ und $q \notin F$ oder umgekehrt, so markiere (p, q).

(3) Falls (p, q) nicht markiert ist, jedoch $(\delta(p, a), \delta(q, a))$ für ein $a \in \Sigma$ markiert ist, dann markiere auch (p, q).

(4) Wiederhole (3), bis keine Änderung mehr auftritt.

Es gilt nun für alle $p, q \in Q$ die Äquivalenz $p \approx q$ genau dann, wenn das Paar (p, q) nicht markiert ist.

Sei Σ ein Alphabet und $k \in \mathbb{N}_0$. Eine Instanz P des Postschen Korrespondenzproblems ist für zwei gegebene $(k+1)$-Tupel $(a^{(0)}, \ldots, a^{(k)})$ und $(b^{(0)}, \ldots, b^{(k)})$ von Wörtern definiert durch

$$P = \left\{ \left[\frac{a^{(0)}}{b^{(0)}}\right], \left[\frac{a^{(1)}}{b^{(1)}}\right], \ldots, \left[\frac{a^{(k)}}{b^{(k)}}\right] \right\} \quad .$$

Eine Lösung zu einer Instanz P ist eine Folge $(i_0, \ldots, i_l)$ für ein $l \in \mathbb{N}_0$ mit

$$a^{(i_0)} \ldots a^{(i_l)} = b^{(i_0)} \ldots b^{(i_l)} \quad .$$

Wir definieren das Postsche Korrespondenzproblem durch

$$\mathsf{PKP} = \{\langle P \rangle \ : \ P \text{ ist eine Instanz des PKP und hat eine Lösung}\} \quad .$$

Eine Lösung zu einer Instanz P heißt speziell, wenn sie mit dem Index 0 beginnt, also wenn $i_0 = 0$ gilt. Wir sprechen dann von einer Instanz des modifizierten Postschen Korrespondenzproblems MPKP, das durch

$$\mathsf{MPKP} = \{\langle P \rangle \ : \ P \text{ ist eine Instanz des PKP und hat eine spezielle Lösung}\}$$

definiert ist.

Speicherbedarf(skomplexitätsklassen) von Turingmaschinen

Zusammenhang

SAT

$\leftrightarrow$

P

CIRCUIT-SAT

Verifikator

polynomieller Verifikator

polynomielle Verifizierbarkeit

Aus $\mathsf{SAT} \in \mathsf{P}$ folgt $\mathsf{P} = \mathsf{NP}$.

Sei $f\colon \mathbb{N} \to \mathbb{R}_{>0}$ eine Funktion. Wir definieren

$$\begin{aligned}
\mathsf{SPACE}(f) &= \Big\{\ L \ :\ \text{L wird in } \mathcal{O}(f) \text{ Speicherbedarf von} \\
&\qquad \text{einer deterministischen Turingmaschine} \\
&\qquad \text{entschieden} \Big\} \\
\mathsf{NSPACE}(f) &= \Big\{\ L \ :\ \text{L wird in } \mathcal{O}(f) \text{ Speicherbedarf von} \\
&\qquad \text{einer nichtdeterministischen} \\
&\qquad \text{Turingmaschine entschieden} \Big\}
\end{aligned}$$

sowie, wenn $\mathcal{P}$ die Menge aller Polynomfunktionen mit einer Variablen und nichtnegativen Koeffizienten ist,

$$\mathsf{PSPACE} := \bigcup_{p \in \mathcal{P}} \mathsf{SPACE}(p)\,, \qquad \mathsf{NPSPACE} := \bigcup_{p \in \mathcal{P}} \mathsf{NSPACE}(p)\,.$$

Sei $\mathcal{V} = (Q,\ \Sigma,\ \Gamma,\ \delta,\ q_0,\ q_{\text{acc}},\ q_{\text{rej}})$ eine Turingmaschine. Sei L eine Sprache.

Wir nennen $\mathcal{V}$ einen Verifikator für L, falls es für alle $w \in L$ ein $c \in \Sigma^*$ so gibt, dass $\mathcal{V}$ die Kodierung $\langle w,\ c\rangle$ akzeptiert.

Ein Verifikator heißt polynomieller Verifikator, falls seine Laufzeit für jede Eingabe $w \in \Sigma^*$ polynomiell in $|w|$ ist. Dann heißt L polynomiell verifizierbar.

$$\begin{aligned}
\mathsf{CIRCUIT\text{-}SAT} &= \Big\{\ \langle C\rangle \ :\ C \text{ ist Schaltkreis} \\
&\qquad \text{mit einem Ausgang,} \\
&\qquad \text{und es gibt einen Eingabevektor,} \\
&\qquad \text{für den C am Ausgang} \\
&\qquad \text{den Wert 1 liefert} \Big\}
\end{aligned}$$

polynomial zeitbeschränkte
nichtdeterministische Turingmaschine

Menge von polynomial zeitbeschränkten
nichtdeterministischen Turingmaschinen

Zusammenhang

nichtdeterministische Turingmaschine

$\leftrightarrow$

deterministische Turingmaschine

Redundanz des Nichtdeterminismus

Abschluss der kontextfreien Sprachen unter
Vereinigung:
Automatenkonstruktion

Ableitungsrelation in einer Grammatik

Wenn eine Sprache L von einer nichtdeterministischen Turingmaschine erkannt wird, so wird L auch von einer deterministischen Turingmaschine erkannt.

Sei $f\colon \mathbb{N} \to \mathbb{N}$ eine Funktion, also $f \in \mathbb{N}^{\mathbb{N}}$. Wir definieren

$$\begin{aligned} \mathsf{NTIME}(f) \quad = \quad \Big\{ \quad L(M) \quad : \quad & \text{es gibt eine nichtdeterministische} \\ & \text{Turingmaschine } M \text{ so, dass } L(M) \\ & \text{Turing-entscheidbar ist} \\ & \text{und } \mathcal{O}(f) = f_M \text{ gilt.} \Big\} \end{aligned}$$

Sei $\mathcal{F}$ eine Teilmenge von $\mathbb{N}^{\mathbb{N}}$. Wir definieren

$$\mathsf{NTIME}(\mathcal{F}) := \bigcup_{f \in \mathcal{F}} \mathsf{NTIME}(f) \quad .$$

Sei $G = (N,\ T,\ P,\ S)$ eine Grammatik mit $P \subseteq N \times (N \cup T)^*$. Seien $u,\ v,\ w \in (N \cup T)^*$ Satzformen, und sei $A \in N$ ein Nichtterminalsymbol.

Wir definieren einen Ableitungsschritt durch die folgende Äquivalenz:

$$uAv \Longrightarrow_G uwv \quad :\Longleftrightarrow \quad (A \to w) \in P \quad .$$

Weiter definieren wir die reflexiv-transitive Hülle von $\Longrightarrow_G$ durch

$$\begin{aligned} uAv \Longrightarrow_G^* uwv \quad :\Longleftrightarrow \quad A = w \ \vee \ \Big(& \exists\, n \in \mathbb{N}_0 \ \ \exists\, A_0, \ldots, A_n \in (N \cup T)^* : \\ & \forall\, i \in [n-1]_0 : (A_i \Longrightarrow_G A_{i+1}) \\ & \wedge\, (A_n \Longrightarrow_G w) \Big) \end{aligned}$$

Die reflexiv-transitive Hülle unserer Ableitungsrelation besagt somit, dass es eine endliche Folge von Ableitungen in den Produktionen gibt, mittels derer w aus A ableitbar ist.

Die Klasse der kontextfreien Sprachen ist unter Vereinigung abgeschlossen.

Seien

$$\mathcal{A}_1 = (Q_1,\ \Sigma,\ \$,\ \bot_1,\ \Gamma_1,\ \delta_1,\ q_1,\ F_1)$$

und

$$\mathcal{A}_2 = (Q_2,\ \Sigma,\ \$,\ \bot_2,\ \Gamma_2,\ \delta_2,\ q_2,\ F_2)$$

Kellerautomaten.

Dann gilt für den Kellerautomaten $\mathcal{A} = (Q,\ \Sigma,\ \$,\ \bot,\ \Gamma,\ \delta,\ q_0,\ F)$ mit

- $Q := Q_1 \cup Q_2 \cup \{q_0\}$ mit $q_0 \notin Q_1 \cup Q_2$,
- $\Gamma := \Gamma_1 \cup \Gamma_2 \cup \{\bot\}$ mit $\bot \notin \Gamma_1 \cup \Gamma_2$,
- $\delta := \delta_1 \cup \delta_2 \cup \{(q_0,\ \varepsilon,\ \bot,\ q_1,\ \bot_1),\ (q_0,\ \varepsilon,\ \bot,\ q_2,\ \bot_2)\}$,
- $F := F_1 \cup F_2$

die Gleichheit $L(\mathcal{A}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.

Satzform

Nichtdeterministische Berechnung

Berechnung eines deterministischen endlichen Automaten

akzeptierende Berechnung eines deterministischen endlichen Automaten

Chomsky-Normalform

Sei $\mathcal{N} = (Q,\ \Sigma,\ \delta,\ q_0,\ F)$ ein nichtdeterministischer endlicher Automat, sei $n \in \mathbb{N}_{>0}$, und sei $w = w_0 \dots w_{n-1} \in \Sigma^*$ ein Wort.

Eine nichtdeterministische Berechnung von $\mathcal{N}$ auf w ist eine Folge $r_0 \dots r_n$ von Zuständen mit den Eigenschaften

- Es gilt $w = w_0 \dots w_{m-1}$ für ein $m \in \mathbb{N}_0$, und für alle $i \in [m-1]_0$ gilt $w_i \in \Sigma \cup \{\varepsilon\}$,
- $r_0 = q_0$,
- Für alle $i \in [m-1]_0$ gilt $r_{i+1} \in \delta(r_i,\ w_i)$.

Sei $G = (N,\ T,\ P,\ S)$ eine Grammatik mit $P \subseteq N \times (N \cup T)^*$.

Eine Satzform über G ist ein Element aus $(N \cup T)^*$.

Sei $G = (N,\ T,\ P,\ S)$ eine kontextfreie Grammatik.

G ist in Chomsky-Normalform, wenn auf der rechten Seite einer jeden Regel genau ein Terminalsymbol oder genau zwei vom Startsymbol verschiedene Nichtterminalsymbole stehen.

Um das leere Wort erzeugen zu können, erlauben wir zusätzlich die Regel $S \to \varepsilon$.

Sei $\mathcal{A} = (Q,\ \Sigma,\ \delta,\ q_0,\ F)$ ein deterministischer endlicher Automat, sei $n \in \mathbb{N}_{>0}$, und sei $w = w_0 \dots w_{n-1} \in \Sigma^*$ ein Wort.

Eine Berechnung von $\mathcal{A}$ auf w ist eine Folge $r_0 \dots r_n$ von Zuständen mit den Eigenschaften

- $r_0 = q_0$,
- Für alle $i \in [n-1]_0$ gilt $\delta(r_i,\ w_i) = r_{i+1}$.

Gilt zusätzlich $r_n \in F$, so akzeptiert $\mathcal{A}$ das Wort w, und wir nennen die Berechnung akzeptierend.

Logarithmisch beschränkte Speicherbedarfskomplexitätsklassen

aufzählende Turingmaschine

aufzählbare Sprache

Vereinfachte Form der Chomsky-Hierarchie:

Alternative Bezeichnungen der Sprachklassen

122 *Antwort*

Eine aufzählende Turingmaschine ist ein 7-Tupel

$$(Q,\ \Sigma,\ \Gamma,\ \delta,\ \#,\ q_0,\ q_{\text{print}}) \quad ,$$

wobei

- Q eine endliche, nichtleere Menge von Zuständen,
- Σ das Ausgabealphabet mit $\# \in \Sigma$,
- Γ das Band- oder Arbeitsalphabet mit $(\Sigma \setminus \{\#\}) \subseteq \Gamma$,
- δ die Zustandsübergangsfunktion (wahlweise wie bei der deterministischen oder wie bei der nichtdeterministischen Turingmaschine),
- $\#$ das Trennsymbol des Ausgabebandes,
- $q_0 \in Q$ der Startzustand,
- $q_{\text{print}} \in Q$ der akzeptierende Zustand

sind. Eine aufzählende Turingmaschine schreibt nach und nach alle Wörter der Sprache, durch # getrennt, auf das Band, wobei ein Wort ohne Weiteres mehrfach geschrieben werden kann.

121 *Antwort*

Sei $\mathcal{L} = \{n \mapsto c \cdot \log(n) + d : c \in \mathbb{N}_{>0},\ d \in \mathbb{N}_0\}$ eine Menge logarithmisch beschränkter Funktionen. Wir definieren

$$\mathsf{L} := \bigcup_{p \in \mathcal{L}} \mathsf{SPACE}(p) \quad ,$$

$$\mathsf{NL} := \bigcup_{p \in \mathcal{L}} \mathsf{NSPACE}(p) \quad .$$

Es ist L die Klasse der Sprachen, die mit logarithmisch beschränktem Platzbedarf von einer deterministischen Turingmaschine entschieden werden können, und NL die Klasse der Sprachen, die mit logarithmisch beschränktem Platzbedarf von einer nichtdeterministischen Turingmaschine entschieden werden können.

124 *Antwort*

reguläre Sprache	Typ-3-Sprache
kontextfreie Sprache	Typ-2-Sprache kontextunabhängige Sprache
kontextsensitive Sprache	Typ-1-Sprache
Turing-erkennbare Sprache	Typ-0-Sprache (positiv) semi-entscheidbare Sprache rekursiv aufzählbare Sprache aufzählbare Sprache

123 *Antwort*

Eine Sprache heißt aufzählbar, wenn sie durch eine aufzählende Turingmaschine erzeugt werden kann.

- Für jedes Alphabet Σ und für jede Sprache $L \subseteq \Sigma^*$ gilt, dass wenigstens eine der Sprachen L, $\overline{L}$ Turing-erkennbar ist.

Linksableitung

Rechtsableitung

Semantik von durch reguläre Ausdrücke induzierten Sprachen

Äquivalenz von Automaten

Sei $G = (N, T, P, S)$ eine Grammatik mit $P \subseteq N \times (N \cup T)^*$.
Seien $\alpha, \beta, \gamma \in (N \cup T)^*$ Satzformen, und sei $A \in N$ ein Nichtterminalsymbol.

Ist $\alpha \in T^*$, so ist $\alpha A \gamma \Longrightarrow_G \alpha\beta\gamma$ eine Linksableitung. Wir schreiben dann $\alpha A \gamma \Longrightarrow_G^L \alpha\beta\gamma$.

Ist $\gamma \in T^*$, so ist $\alpha A \gamma \Longrightarrow_G \alpha\beta\gamma$ eine Rechtsableitung. Wir schreiben dann $\alpha A \gamma \Longrightarrow_G^R \alpha\beta\gamma$.

Bei einer Linksableitung wird also das am weitesten links stehende Nichtterminalsymbol ersetzt, bei einer Rechtsableitung das am weitesten rechts stehende Nichtterminalsymbol.

- falsch

Seien $\mathcal{A}_1$ und $\mathcal{A}_2$ endliche Automaten.

Wir nennen $\mathcal{A}_1$ und $\mathcal{A}_2$ äquivalent, wenn $L(\mathcal{A}_1) = L(\mathcal{A}_2)$ gilt, wenn sie also dieselbe Sprache erkennen.

Sei Σ ein Alphabet. Wir legen induktiv fest:

- Induktionsanfang
  - Für alle $a \in \Sigma$ ist $L(a) = \{a\}$.
  - Es ist $L(\varepsilon) = \{\varepsilon\}$.
  - Es ist $L(\varnothing) = \varnothing$.
- Induktionsschritt
  - Für reguläre Ausdrücke R_0 und R_1 ist $L((R_0 \cup R_1)) = L(R_0) \cup L(R_1)$.
  - Für reguläre Ausdrücke R_0 und R_1 ist $L((R_0 \circ R_1)) = L(R_0) \circ L(R_1)$.
  - Für den regulären Ausdruck R definieren wir $L((R^*)) = L(R)^*$.

Äquivalenz von Grammatiken

- Die durch deterministische endliche Automaten erkennbaren Sprachen sind genau die regulären Sprachen.
- Ersetzt man „deterministische endliche Automaten“ durch „nichtdeterministische endliche Automaten“, so verliert die Aussage ihre Gültigkeit.

Kodierung einer Turingmaschine

Deterministische Entscheidung in exponentieller Zeit

- wahr
- falsch

Seien G_1 und G_2 Grammatiken.

Wir nennen G_1 und G_2 äquivalent, wenn $L(G_1) = L(G_2)$ gilt, wenn sie also dieselbe Sprache erzeugen.

Für jede Sprache $L \in \mathsf{NP}$ existiert ein Polynom p so, dass

$$L \in \mathsf{DTIME}\left(2^{p(n)}\right)$$

für alle $n \in \mathbb{N}$ gilt.

Sei $\mathcal{M} = (Q,\ \Sigma,\ \Gamma,\ \delta,\ q_0,\ q_{acc},\ q_{rej})$ eine normierte Turingmaschine, und seien

$$b_0, \ldots, b_r \ \in \Big\{ \ (q,\ a,\ p,\ b,\ X) \ : \ q, p \in Q \ \wedge \ a, b \in \Gamma \ \wedge \ X \in \{\mathsf{L}, \mathsf{R}\} \\ \wedge\ \delta(q,\ a) = (p,\ b,\ X) \ \Big\}$$

für ein passendes $r \in \mathbb{N}$, wobei die b_i aufsteigend gemäß $(q,\ a)$ sortiert sind. Wir definieren für jedes $i \in \mathbb{N}_{<r}$ die Kodierung eines Transitionsschrittes durch

$$\langle b_i \rangle = \langle (q,\ a,\ p,\ b,\ X) \rangle = 0^{q+1}\ 1\ 0^{a+1}\ 1\ 0^{p+1}\ 1\ 0^{b+1}\ 1\ 0^{X+1}$$

und nennen

$$\langle \mathcal{M} \rangle = 111\ 0^{|Q|+1}\ 111\ \langle b_0 \rangle\ 11\ \langle b_1 \rangle\ 11\ \ldots\ 11\ \langle b_r \rangle\ 111$$

die Binärkodierung der Turingmaschine $\mathcal{M}$.

Church-Turing-These

Deterministische Turingmaschine

Satz von Savitch

- Es gilt $\mathsf{PSPACE} = \mathsf{NPSPACE}$.
- Es gilt $\mathsf{P} \subseteq \mathsf{NP}$.
- Es gilt $\mathsf{P} = \mathsf{NP}$.

134 *Antwort*

Eine deterministische Turingmaschine ist ein 7-Tupel

$$(Q,\ \Sigma,\ \Gamma,\ \delta,\ q_0,\ q_{acc},\ q_{rej}) \quad ,$$

wobei

- Q eine endliche, nichtleere Menge von Zuständen,
- Σ das Eingabealphabet mit $\sqcup \notin \Sigma$,
- Γ das Band- oder Arbeitsalphabet mit $\Sigma \cup \{\sqcup\} \subseteq \Gamma$,
- $\delta\colon Q \setminus \{q_{acc},\ q_{rej}\} \times \Gamma \to Q \times \Gamma \times \{\mathsf{L},\ \mathsf{R}\}$ die Zustandsübergangsfunktion,
- $q_0 \in Q$ der Startzustand,
- $q_{acc} \in Q$ der akzeptierende Zustand,
- $q_{rej} \in Q \setminus \{q_{acc}\}$ der verwerfende Zustand

sind.

133 *Antwort*

Der intuitive Algorithmusbegriff stimmt mit dem formalen Begriff der Turingmaschine überein;

der intuitive Entscheidbarkeitsbegriff stimmt mit dem formalen Begriff der Turing-Entscheidbarkeit überein;

der intuitive Berechenbarkeitsbegriff stimmt mit dem formalen Begriff der Turing-Berechenbarkeit überein.

136 *Antwort*

- wahr
- wahr
- unbekannt

135 *Antwort*

Für jede Funktion $f\colon \mathbb{N} \to \mathbb{R}_{>0}$, die $f(n) \geqslant n$ für alle $n \in \mathbb{N}$ erfüllt, gilt

$$\mathsf{NSPACE}(f(n)) \subseteq \mathsf{SPACE}(f(n)^2) \quad .$$

Komplexitätsklasse NP

regulärer Ausdruck

erkannte Sprache eines nichtdeterministischen endlichen Automaten

erkannte Sprache eines deterministischen endlichen Automaten

Sei Σ ein Alphabet. Wir legen induktiv fest:

I.A. Für alle $a \in \Sigma$ ist a ein regulärer Ausdruck.

I.A. Es ist ε ein regulärer Ausdruck.

I.A. Es ist $\varnothing$ ein regulärer Ausdruck.

I.S. Für reguläre Ausdrücke R_0 und R_1 ist $(R_0 \cup R_1)$ ein regulärer Ausdruck.

I.S. Für reguläre Ausdrücke R_0 und R_1 ist $(R_0 \circ R_1)$ ein regulärer Ausdruck.

I.S. Falls R ein regulärer Ausdruck ist, so auch (R^*).

Andere Schreibweisen:

- Weglassen der äußeren Klammern bei eindeutiger Lesbarkeit
- $R_0 \mid R_1$ für $R_0 \cup R_1$
- $R_0 \cdot R_1$ für $R_0 \circ R_1$, noch kürzer $R_0 R_1$

Sei $\mathcal{P}$ die Menge aller Polynomfunktionen mit einer Variablen und nichtnegativen Koeffizienten.

Dann definieren wir die Komplexitätsklasse der polynomiell zeitbeschränkten nichtdeterministischen Turingmaschinen durch

$$\mathsf{NP} = \mathsf{NTIME}(\mathcal{P}) \quad .$$

Die Komplexitätsklasse NP enthält genau die Entscheidungsprobleme, die von einer nichtdeterministischen Turingmaschine in Polynomialzeit verifiziert werden können.

Sei $\mathcal{A} = (Q,\ \Sigma,\ \delta,\ q_0,\ F)$ ein deterministischer endlicher Automat. Die von $\mathcal{A}$ erkannte Sprache ist

$$L(\mathcal{A}) \quad = \quad \Big\{ \quad w \in \Sigma^* \quad : \quad \text{es gibt eine akzeptierende Berechnung in } \mathcal{A} \text{ für } w \Big\} \quad .$$

Sei $\mathcal{N} = (Q,\ \Sigma,\ \delta,\ q_0,\ F)$ ein nichtdeterministischer endlicher Automat. Die von $\mathcal{N}$ erkannte Sprache ist

$$L(\mathcal{N}) \quad = \quad \Big\{ \quad w \in \Sigma^* \quad : \quad \text{es gibt eine akzeptierende Berechnung in } \mathcal{N} \text{ für } w \Big\} \quad .$$

Turingmaschine mit reduziertem Bandalphabet

Zusammenhang

Reduzierbarkeit

$\leftrightarrow$

Turing-Erkennbarkeit/-Entscheidbarkeit

Eigenschaften der Reduktion

Eigenschaften von NP-vollständigen Sprachen

Überführen einer kontextfreien Grammatik in Chomsky-Normalform

Seien $\Sigma,\ \Delta$ Alphabete und $L_1 \subseteq \Sigma^*$ sowie $L_2 \subseteq \Delta^*$ Sprachen.

- Ist L_2 Turing-entscheidbar und gilt $L_1 \leqslant L_2$,
  so ist auch L_1 Turing-entscheidbar.
- Ist L_2 Turing-erkennbar und gilt $L_1 \leqslant L_2$,
  so ist auch L_1 Turing-erkennbar.
- Ist L_1 nicht Turing-entscheidbar und gilt $L_1 \leqslant L_2$,
  so ist auch L_2 nicht Turing-entscheidbar.
- Ist L_1 Co-Turing-erkennbar und gilt $L_1 \leqslant L_2$,
  so ist auch L_2 Co-Turing-erkennbar.

Sei $\mathcal{M}_1 = (Q_1,\ \Sigma,\ \Gamma_1,\ \delta_1,\ q_1,\ q_{acc1},\ q_{rej_1})$ mit $\Sigma = \{0,\ 1\}$.

Dann gibt es eine Turingmaschine $\mathcal{M}_2 = (Q_2,\ \Sigma,\ \Gamma_2,\ \delta_2,\ q_2,\ q_{acc2},\ q_{rej_2})$ so, dass $\Gamma_2 = \{0,\ 1,\ 2\}$ und $L(\mathcal{M}_1) = L(\mathcal{M}_2)$ gelten.

Sei $G = (N,\ T,\ P,\ S)$ eine kontextfreie Grammatik. Transformiere sie wie folgt:

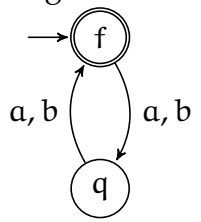
(1) Bestimme die Menge U aller Nichtterminale, die in keiner Ableitungsfolge erscheinen können (etwa weil sie auf keiner rechten Regelseite vorkommen). Ersetze N durch $N \setminus U$ und lösche aus P alle Regeln, in denen Nichtterminale aus U vorkommen.
(2) Bestimme die Menge U' aller Nichtterminale, die auf mindestens einer rechten Regelseite, aber auf keiner linken Regelseite erscheinen. Ersetze N durch $N \setminus U'$ und lösche aus P alle Regeln, in denen Nichtterminale aus U' vorkommen.
(3) Sofern S auf einer rechten Regelseite steht, füge ein neues Nichtterminal S' zu N hinzu und ergänze P um die Regel $S' \to S$.
(4) Entferne alle Produktionen mit rechter Seite ε (jedoch behalte $S \to \varepsilon$ im Fall $\varepsilon \in L(G)$). Dazu bestimme alle Nichtterminale X, aus denen sich das leere Wort ableiten lässt und für jede Produktion $Y \to uXv$ mit $u,\ v \in T^*$ und $uv \neq \varepsilon$ füge die Produktion $Y \to uv$ zu P hinzu. Wiederhole dies, bis keine Änderung mehr auftritt.
(5) Entferne Zyklen von Kettenproduktionen, also Zyklen der Form $X_0 \to X_1,\ X_1 \to X_2,\ \ldots,\ X_k \to X_0$ für ein passendes $k \in \mathbb{N}_0$. Ersetze außerdem jedes Vorkommen eines Nichtterminals X_i mit $i \leqslant k$ durch S, falls $S = X_j$ für ein $j \in [k]_0$, und sonst durch X_0.
(6) Entferne alle Kettenproduktionen der Form $A \to B$ mit $A,\ B \in N$. Dazu ersetze B durch die aus B direkt ableitbaren rechten Seiten. Wiederhole dies, bis keine Änderung mehr auftritt.
(7) Führe für jedes Terminal t ein neues Nichtterminal V_t ein, ersetze jedes Vorkommen von t in den Produktionen durch V_t, und füge neue Produktionen $V_t \to t$ hinzu.
(8) Spalte Produktionen, deren rechte Seite zu lang ist, mittels neu eingeführter Nichtterminale auf. Beispiel: aus $S \to ASB$ mache $S \to AZ,\ Z \to SB$.

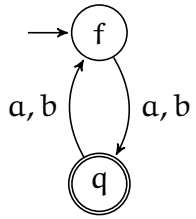
http://www.iti.fh-flensburg.de/lang/theor/{normalisierung,chomsky-normalform}.htm
https://wwwmath.uni-muenster.de/logik/Personen/Schulz/BETHWIKI/index.php/Chomsky_Normalform
Letzter Zugriff am 29.03.2017

Seien $\Sigma,\ \Delta$ Alphabete und $L_1 \subseteq \Sigma^*$ sowie $L_2 \subseteq \Delta^*$ Sprachen, und es gelte $L_1 \in \mathsf{NPC}$. Dann gelten

- $L_1 \in \mathsf{P} \implies \mathsf{P} = \mathsf{NP}$,
- $L_2 \in \mathsf{NP} \wedge L_1 \leqslant_p L_2 \implies L_2 \in \mathsf{NPC}$.

Sei $\Sigma := \{a,\ b\}$. Es seien $\mathcal{A}$ der linke und $\mathcal{B}$ der rechte der hier dargestellten deterministischen endlichen Automaten.





Dann gelten:

- $L(\mathcal{A}) = L(\mathcal{B})$
- $L(\mathcal{A}) \cap L(\mathcal{B}) \neq \varnothing$
- $L(\mathcal{A}) \supseteq \{w \in \Sigma^* \ :\ |w|_a = |w|_b\}$
- $L(\mathcal{A}) = \{w \in \Sigma^* \ :\ |w|_a = |w|_b\}$
- $L(\mathcal{A}) = \{w \in \Sigma^* \ :\ |w|_a \equiv_2 |w|_b\}$
- $L(\mathcal{B}) = L\left((a|b) \cdot ((a|b) \cdot (a|b))^*\right)$

Es sei $L := \{a^p \ :\ p \text{ ist eine Primzahl}\} \subseteq \{a\}^*$. Dann gelten

- L ist regulär.
- L ist nicht kontextfrei.
- L ist Turing-erkennbar.
- L ist nicht Turing-entscheidbar.

SUBSET-SUM

- Jede kontextfreie Sprache ist endlich.
- Jede endliche Sprache ist regulär.
- Jede reguläre Sprache ist kontextfrei.
- Es gibt eine nichtentscheidbare Sprache, die kontextfrei ist.
- Für jedes Alphabet Σ ist Σ^* kontextfrei.
- Jede kontextfreie Sprache ist Turing-erkennbar.
- Für jede Turing-entscheidbare Sprache L ist $\overline{L}$ Turing-entscheidbar.
- Für jede Turing-erkennbare Sprache L ist $\overline{L}$ Turing-entscheidbar.
- Für jede Turing-erkennbare Sprache L ist $\overline{L}$ Turing-erkennbar.

146 *Antwort*

- falsch
- wahr
- wahr
- falsch

145 *Antwort*

- falsch
- falsch
- wahr
- falsch
- wahr
- wahr

148 *Antwort*

- falsch
- wahr
- wahr
- falsch
- wahr
- wahr
- wahr
- falsch
- falsch

147 *Antwort*

$$\textsf{SUBSET-SUM} = \Big\{ \langle S, t \rangle : S \text{ ist eine endliche Menge natürlicher Zahlen, } t \text{ ist eine natürliche Zahl, und es gibt eine Teilmenge } U \text{ von } S \text{ mit } \sum_{u \in U} u = t \Big\}$$

- Jedes NP-vollständige Problem ist in NP.
- Jedes NP-schwere Problem ist in NP.

Sei Σ ein abzählbares Alphabet.

Sei

$$\mathcal{C} := \left\{ f \in (\Sigma^*)^{\Sigma^*} \ : \ f \text{ ist Turing-berechenbar} \right\} \quad .$$

$\mathcal{C}$ ist abzählbar, und es gelte $\mathcal{C} = \{f_i \ : \ i \in \mathbb{N}_0\}$.

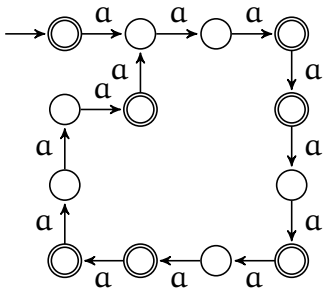
Ferner sei Σ^* durch $\Sigma^* = \{w_j \ : \ j \in \mathbb{N}_0\}$ abgezählt.

Für ein festes $a \in \Sigma$ sei $\widetilde{f}_a \colon \Sigma^* \to \Sigma^*$ definiert durch

$$\widetilde{f}_a(w) := f_i(w_i)a \quad \text{für alle } i \in \mathbb{N}_0.$$

Dann ist $\widetilde{f}_a$ Turing-berechenbar.

Sei $\Sigma := \{a\}$. Es sei $\mathcal{A}$ der hier dargestellte deterministische endliche Automat.



Man bestimme die Anzahl der Zustände eines minimalen nichtdeterministischen endlichen Automaten, der genau $L(\mathcal{A})$ erkennt.

E_{CFG}

inkl. Aussagen zu Entscheidbarkeit

150 *Antwort*

falsch

nach Volker Claus / Andreas Schwill (1988): Duden „Informatik“. Ein Sachlexikon für Studium und Praxis. Mannheim (Dudenverlag), S. 171

149 *Antwort*

- wahr
- falsch

152 *Antwort*

$$\mathsf{E}_{\mathsf{CFG}} = \{ \langle G \rangle : G \text{ ist eine kontextfreie Grammatik, und es gilt } L(G) = \varnothing \}$$

$\mathsf{E}_{\mathsf{CFG}}$ ist Turing-entscheidbar. Man beweist dies, indem...

Hier fehlt noch was!

151 *Antwort*

8

Volker Claus / Andreas Schwill (1988): Duden „Informatik“. Ein Sachlexikon für Studium und Praxis. Mannheim (Dudenverlag), S. 208

$\mathsf{E}_{\mathsf{DEA}}$

inkl. Aussagen zu Entscheidbarkeit

Spezielles Halteproblem $\mathsf{HALT}^{\Sigma}_{\mathsf{TM}}$

inkl. Aussagen zu Entscheidbarkeit

Sei Σ ein Alphabet.

Man bestimme den Wahrheitsgehalt der folgenden Aussagen.

- $L(\varnothing) = \varnothing$
- $L(\varepsilon) = \varepsilon$
- $L(\varepsilon) = \{\varepsilon\}$
- $L(\{\varepsilon\}) = \varnothing$
- $L(\{\varepsilon\})$ ist nicht definiert

Man zeige, dass die Sprache

$$\{z \in \{a, b\}^* : |z|_a \geqslant 1 \text{ und } |z|_a \text{ teilt } |z|_b\}$$

nicht regulär ist.

$$\mathsf{HALT}^{\Sigma}_{\mathsf{TM}} \;=\; \Big\{\; \langle M\rangle \;:\; M \text{ ist eine Turingmaschine, die bei jeder Eingabe hält} \;\Big\}$$

$\mathsf{HALT}^{\Sigma}_{\mathsf{TM}}$ ist nicht entscheidbar. Man beweist dies, indem man von $\mathsf{HALT}^{\varepsilon}_{\mathsf{TM}}$ reduziert.

$$\mathsf{E}_{\mathsf{DEA}} \;=\; \Big\{\; \langle \mathcal{A}\rangle \;:\; \mathcal{A} \text{ ist ein deterministischer endlicher Automat, und es gilt } L(\mathcal{A}) = \varnothing \;\Big\}$$

$\mathsf{E}_{\mathsf{DEA}}$ ist Turing-entscheidbar. Man beweist dies, indem man $\mathcal{A}$ minimiert und dem minimalen Automaten ansieht, ob die von ihm erkannte Sprache leer ist oder nicht.

Sei $\Sigma := \{a, b\}$, sei $L := \{z \in \Sigma^* \,:\, |z|_a \geqslant 1 \text{ und } |z|_a \text{ teilt } |z|_b\}$.

Behauptung: L ist nicht regulär.

Beweis: Wir nehmen an, L wäre regulär. Nach Pumping-Lemma gibt es dann $n \in \mathbb{N}_{\geqslant 1}$ so, dass für jedes Wort $z \in L$ mit $|z| \geqslant n$ Wörter $u,\ v,\ w \in \Sigma^*$ existieren mit $z = u\,v\,w$ und

$$(1)\ |u\,v| \leqslant n \quad\wedge\quad (2)\ |v| \geqslant 1 \quad\wedge\quad (3)\ \forall\, i \in \mathbb{N}_0 \quad u\,v^i\,w \in L \quad .$$

Sei $n \in \mathbb{N}_{\geqslant 1}$ derart gewählt. Für das Wort $z := a^n\,b^n \in \{a, b\}^*$ gilt dann $|z|_a = |z|_b = n$, und wegen $|z| = 2\,n \geqslant n$ und $n \mid n$ gelten $z \in L$ und $|z| \geqslant n$, also gibt es Wörter $u,\ v,\ w \in \Sigma^*$ mit $z = u\,v\,w$ und (1), (2), (3). Seien $u,\ v,\ w \in \Sigma^*$ derart gewählt. Wegen (1) gilt dann $u\,v \in L(a^*)$ und damit $u \in L(a^*)$ und $v \in L(a^*)$. Also gibt es $k,\ l,\ r,\ s \in \mathbb{N}_0$ mit $u = a^k$ und $l \geqslant 1$ und $v = a^l$ und $s \geqslant 1$ und $w = a^r\,b^s$. Seien $k,\ l,\ r,\ s \in \mathbb{N}_0$ derart. Nun haben wir also

$$z = u\,v\,w = a^k\,a^l\,a^r\,b^s \quad \text{und} \quad k + l + r = s \quad .$$

Aus $v \in L(a^*)$ folgt $v^i \in L(a^*)$ für alle $i \in \mathbb{N}_{>0}$, Aufpumpen des mittleren Teils erhöht also lediglich die Anzahl der a's, und die Anzahl der b's bleibt gleich. Dies formalisieren wir, uns an $l \geqslant 1$ erinnernd, wie folgt:

$$\big|u\,v^2\,w\big|_a \;=\; k + 2\,l + r \;>\; k + l + r \;=\; s \;=\; \big|u\,v\,w\big|_b \;=\; \big|u\,v^2\,w\big|_b \quad ,$$

also ist $\big|u\,v^2\,w\big|_a$ kein Teiler von $\big|u\,v^2\,w\big|_b$, und damit ist $u\,v^2\,w \notin L$, Widerspruch zu (3).

Also ist L nicht regulär. $\square$

- wahr
- falsch
- wahr
- falsch
- wahr

Clique

PRIMES

inkl. Aussagen zu Entscheidbarkeit

Komplexitätsklasse co-P

Komplexitätsklasse co-NP

$$\mathsf{PRIMES} \;=\; \{\; \langle n \rangle \;:\; n \in \mathbb{N}, \text{ und } n \text{ ist eine Primzahl} \;\}$$

PRIMES ist Turing-entscheidbar. Der einfachste Algorithmus, der dies leistet, ist die Probedivision. Wird sie in der Variante benutzt, dass durch jede ungerade Zahl $i \leqslant \lceil \sqrt{n} \rceil$ dividiert wird, wobei n die zu testende Zahl sei, so ist ihre Laufzeit exponentiell in der Anzahl der Ziffern von n. Jedoch gilt sogar

$$\mathsf{PRIMES} \text{ is in } \mathsf{P} \quad ,$$

wie Manindra Agrawal, Neeraj Kayal und Nitin Saxena im Jahr 2002 zeigten.

https://de.wikipedia.org/w/index.php?title=AKS-Primzahltest&oldid=162838445, letzter Zugriff am 03.04.2017
https://en.wikipedia.org/w/index.php?title=Trial_division&oldid=911650300, letzter Zugriff am 30.09.2019

Sei V eine nichtleere Menge, und sei $G = (V, E)$ ein ungerichteter Graph. Sei U eine Teilmenge von V. Wenn zwischen je zwei verschiedenen Knoten in U eine Kante verläuft, also

$$\forall u, v \in U \qquad u \neq v \implies \{u, v\} \in E$$

gilt, so heißt U eine Clique in G.

$$\mathsf{co\text{-}NP} = \left\{ L \,:\, \overline{L} \in \mathsf{NP} \right\}$$

Die Komplexitätsklasse co-NP enthält genau jedes Entscheidungsproblem, dessen Komplement von einer nichtdeterministischen Turingmaschine in Polynomialzeit verifiziert werden kann.

$$\mathsf{co\text{-}P} = \left\{ L \,:\, \overline{L} \in \mathsf{P} \right\}$$

Die Komplexitätsklasse co-P enthält genau jedes Entscheidungsproblem, dessen Komplement von einer deterministischen Turingmaschine in Polynomialzeit gelöst werden kann.

- Sind $f, g\colon \mathbb{N} \to \mathbb{R}_{>0}$ definiert durch

  $$f\colon n \mapsto 2^{n+3} \quad \text{und} \quad g\colon n \mapsto 2^n \quad \text{jeweils für alle } n \in \mathbb{N} \quad ,$$

  so gelten $f \in \mathcal{O}(g)$ und $g \in \mathcal{O}(f)$.
- Für alle $f, g, h \in (\mathbb{R}_{>0})^{\mathbb{N}_0}$ gilt: Aus $f \in \mathcal{O}(g)$ und $g \in \mathcal{O}(h)$ folgt $f \in \mathcal{O}(h)$.
- Für alle $f, g \in (\mathbb{R}_{>0})^{\mathbb{N}_0}$ gilt: Aus $f \in \mathcal{O}(g)$ und $g \in \mathcal{O}(f)$ folgt $f = g$.
- Für jede durch $T(1) = 1$ und für alle $n \in \mathbb{N}_{>1}$ durch

  $$T(n) = 2\,T\left(\left\lfloor \tfrac{n}{2} \right\rfloor\right) + \mathcal{O}(n \cdot \log(n))$$

  definierte Rekurrenz gilt $T(n) = \mathcal{O}(n \cdot \log(n))$.

- Es gibt eine kontextfreie Grammatik $(\{S\}, \varnothing, \varnothing, S)$.
- Die Sprache des wiederholenden Wortes

  $$\left\{u^2 : u \in \{a, b\}^*\right\} \subseteq \{a, b\}^*$$

  ist nicht kontextfrei.
- Die Sprache $\{a^n\, b^n : n \in \mathbb{N}\} \subseteq \{a, b\}^*$ ist kontextfrei.
- Sind L_1, L_2 Sprachen mit $L_1 \subseteq L_2$ und ist L_2 kontextfrei, so auch L_1.

Prinzip des Nichtdeterminismus

Seien $f, g \in (\mathbb{R}_{>0})^{\mathbb{N}_0}$. Dann gilt:

Gibt es $n_0 \in \mathbb{N}$ so, dass $f(n) \neq 0$ für alle $n \in \mathbb{N}_{\geqslant n_0}$ gilt, und ist die Folge $\left(\dfrac{g(n)}{f(n)}\right)_{n \in \mathbb{N}_{\geqslant n_0}}$ konvergent, so gilt $f \in \mathcal{O}(g)$.

- wahr
- wahr
- wahr
- falsch

- wahr
- wahr
- falsch
- falsch

Barbara Langfeld, Uni Kiel, Mathematik B für Informatikstudierende, Sommersemester 2013, Hausaufgabe 1.2
https://de.wikipedia.org/w/index.php?title=Landau-Symbole&oldid=162680389
http://cstheory.stackexchange.com/a/3637
Letzter Zugriff am 03.04.2017

falsch

http://www.inf.fu-berlin.de/lehre/SS10/infb/o-notation.pdf
Letzter Zugriff am 03.04.2017

Eine nichtdeterministische Turingmaschine rät genau einmal, und zwar richtig.

Mitja Kulczynski, pers. comm., 17.03.2017

Seien $f, g \in (\mathbb{R}_{>0})^{\mathbb{N}_0}$. Dann gilt:

$f \in \mathcal{O}(g) \quad \Longrightarrow \quad \log(f) \in \mathcal{O}(\log(g))$.

WHILE-Programm

Variablenbezeichner BEZ

Monoid

Wortmonoid

Die Menge BEZ der Variablenbezeichner ist die kleinste (im Inklusionssinne) Menge, die den Bezeichner `x` und mit einem Variablenbezeichner u auch die Bezeichner $u_0, \ldots, u_9$ enthält.

Die Menge WHILE der *WHILE*-Programme ist die kleinste (im Inklusionssinne) Menge mit den folgenden Eigenschaften:

- Sind $u, v \in$ BEZ, so gilt `u := v+1` $\in$ WHILE.
- Sind $u, v \in$ BEZ, so gilt `u := v-1` $\in$ WHILE.
- Sind $u, v \in$ WHILE, so auch `u ; v`.
- Sind $u, v \in$ WHILE und $w \in$ BEZ, so gilt `if␣w > 0␣then␣u␣else␣v␣fi` $\in$ WHILE.
- Sind $u \in$ WHILE und $w \in$ BEZ, so gilt `while␣w > 0␣do␣u␣endwhile` $\in$ WHILE.

falsch

https://cs.stackexchange.com/q/116343
Letzter Zugriff am 27.10.2019

Ist M eine nichtleere Menge und $\circ\colon M \times M \to M$ eine Verknüpfung auf M, die das Assoziativgesetz

$$\forall\, a, b, c \in M \quad (a \circ b) \circ c = a \circ (b \circ c)$$

erfüllt und für die es ein neutrales Element $e \in M$ gibt, für das also

$$\forall\, a \in M \quad a \circ e = a \;\wedge\; e \circ a = a$$

gilt, so heißt das Tripel $(M, \circ, e)$ ein Monoid.

Ist Σ ein Alphabet, so ist Σ^* zusammen mit der Konkatenation $\circ$ von Wörtern und dem leeren Wort ε, also

$$(\Sigma^*, \circ, \varepsilon) \quad ,$$

ein Monoid, das Wortmonoid über Σ.