Definition

Alphabet

Definition

Menge der endlichen Folgen

Definition

Wort

Definition

Induktiv w^n definieren

Definition

y,w sind Wörter über $\sum$. Dann heißt y:

Definition

Sprachen

Definition

Präfix

Definition

Infix

Definition

Suffix

Definition

formale Sprachen

Für eine Menge X ist X* die Menge der endlichen Folgen über X.
Beispiel: Elemente von $a, b, c, d*$: $(a, b, c), ()$

Ein Alphabet ist eine endliche nichtleere Menge. Üblicherweise heißen Alphabete hier $\sum, \Gamma, \Delta$. Ist $\sum$ Alphabet, so nennen wir die Elemente oft Buchstaben und die Elemente von $\sum *$ auch Wörter über $\sum$ (auch String/Zeichenkette).

$$w^n = \begin{cases} \epsilon \quad \text{falls } n = 0 \\ w * w^{n-1} \quad \text{falls } n > 0 \end{cases}$$

Sind $u = (a_1, a_2, ...a_n)$ und $v = (b_1, b_2, ..., b_n)$ Wörter, so ist $u * v$ das Wort $(a_1, a_2, ...a_n, b_1, b_2, ..., b_n)$; es wird als Verkettung/Konkatenation von u und v bezeichnet. An Stelle von $u * v$ schreibt man auch uv.

f: Menge der möglichen Eingaben $\rightarrow$ Menge der möglichen Ausgaben
Spezialfall $A = 0, 1$ heißt Entscheidungsproblem. Sie ist gegeben durch die Menge der Eingaben.

- Präfix/Anfangsstück von w, wenn es $z \in \sum^*$ gibt mit $yz = w$
- Infix/Faktor von w, wenn es $x, z \in \sum^*$ gibt mit $xyz = w$
- Suffix/Endstück von w, wenn es $x \in \sum^*$ gibt mit $xy = w$

Seien y,w Wörter über $\sum$. Dann heißt Infix/Faktor von w, wenn es $x, z \in \sum *$ gibt mit $xyz = w$.

Seien y,w Wörter über $\sum$. Dann heißt Präfix/Anfangsstück von w, wenn es $z \in \sum *$ gibt mit $yz = w$.

Sei $\sum$ ein Alphabet. Teilmengen von $\sum *$ werden formale Sprachen über $\sum$ genannt.
Eine Menge L ist eine formale Sprache wenn es ein Alphabet $\sum$ gibt, so dass L formale Sprache über $\sum$ ist (d.h. $L \subseteq \sum *$).

Seien y,w Wörter über $\sum$. Dann heißt Suffix/Endstück von w, wenn es $x \in \sum *$ gibt mit $xy = w$.

Definition

Verkettung von Sprachen

Definition

Kleene Abschluss

Definition

Prioritätsregeln für Operationen auf Sprachen

Definition

Grammatik

Definition

Ableitung einer Grammatik

Definition

Wort ist Satzform

Definition

erzeugte Sprache

Definition

Chomsky-0

Definition

Chomsky-1

Definition

Chomsky-2

Sei L eine Sprache. Dann ist $L* = \bigcup_{n\geq 0} L^n$ der Kleene-Abschluss oder die Kleene-Iteration von L. Weiter ist $L^+ = \bigcup_{n\geq 0} L^n$
$(L^+ = L * L = L^* * L)$

Sind L_1 und L_2 Sprachen, so heißt die Sprache $L_1L_2 = \{w|\exists w_1 \in L_1, w_2 \in L_2 : w = w_1w_2\}$ (auch $L_1 * L_2$) die Konkatenation oder Verkettung von L_1 und L_2.

Grammatiken sind ein Mittel um alle syntaktisch korrekten Sätze einer Sprache zu erzeugen.
Eine Grammatik G ist ein 4-Tupel $G = (V, \sum, P, S)$ das folgende Bedingungen erfüllt

- V ist eine endliche Menge von Nicht-Terminalen oder Variablen
- $\sum$ ist ein Alphabet (Menge der Terminale) mit $V \cap \sum = \varnothing$,d.h. kein Zeichen ist gleichzeitig Terminal und Nicht-Terminal
- $P \subseteq (V \cup \sum)^+ \times (v \cup \sum)^*$ ist eine endliche Menge von Regeln oder Produktionen (Produktionsmenge)
- $S \in V$ ist das Startsymbol/ die Startvariable oder das Axiom

Jede Grammatik hat nur endlich viele Regeln!

- Potenz/Iteration binden stärker als Konkatenation
- Konkatenation stärker als Vereinigung/Durchschnitt/Differenz

Ein Wort $w \in (V \cup \sum)^*$ heißt Satzform, wenn es eine Ableitung gibt, deren letztes Wort w ist.

Sei $G = (V, \sum, P, S)$ eine Grammatik. Eine Ableitung ist eine endliche Folge von Wörtern $w_0, w_1, w_2, ..., w_n$ mit $w_0 \Rightarrow w_1 \Rightarrow w_2 \Rightarrow ... \Rightarrow w_n$.

Jede Grammatik ist vom Typ 0 (Semi-Thue-System) und wird auch als rekursiv-aufzählbar bezeichnet.

Die Sprache $L(G) = w \in \sum^* |S \Rightarrow_G^* w$ aller Satzformen aus $\sum^*$ heißt von G erzeugte Sprache.

Eine Regel $(l \rightarrow r)$ heißt kontext-frei wenn $l \in V$ und $r \in (V \cup \sum)^*$ gilt. Eine Grammatik ist vom Typ 2, falls sie nur kontext-freie Regeln enthält

Eine Regel heißt kontext-sensitiv, wenn es Wörter $u, v, w \in (V \cup \sum)^*, |v| > 0$ und ein Nichtterminal $A \in V$ gibt mit $l = uAw$ und $r = uvw$. Eine Grammatik ist vom Typ 1 (oder kontext-sensitiv) falls

- alle Regeln aus P kontext-sensitiv sind
- $(S \rightarrow \epsilon) \in P$ die einzige nicht kontext-sensitive Regel in P ist und S auf keiner rechten Seite einer Regel aus P vorkommt

Definition

Chomsky-3

Beweise

Es gibt einen Algorithmus, der als Eingabe eine Typ-1-Grammatik G und ein Wort w bekommst und nach endlicher Zeit entscheidet ob $w \in L(G)$ gilt.

Definition

Deterministische endliche Automaten

Definition

DFA mit Funktion $\hat{\delta}$

Definition

von einem DFA akzeptierte Sprache

Definition

Wann ist eine Sprache regulär?

Definition

ein nichtdeterministischer endlicher Automat M

Definition

Zu einem gegebenen NFA M definieren wir die Funktion $\hat{\delta} : P(Z) \times \sum^* \to P(Z)$

Definition

die von einem NFA M akzeptierte Sprache ist

Satz

Sei $\sum$ ein Alphabet und $L \subseteq \sum^*$ eine Sprache. Dann sind äquivalent

1. $w = \epsilon$: Da G vom Typ 1 ist, gilt $w \in L(G)$ genau dann wenn $(S \to \epsilon) \in P$. Dies kannn ein Algorithmus entscheiden

2. $|w| \geq 1$: Definiere einen gerichteten Graphen (W,E) wie folgt

   - Knoten sind die nichtleeren Wörter über $V \cup \sum$ der Länge $\geq |w|$ (insbes. $S, w \in W$)
   - $(u, v) \in E$ genau dann wenn $u \Rightarrow_G v$

   da kontext-sensitiv ist, gilt $1 = |u_0| \geq |u_1| \geq |u_2| \geq ... \geq |u_n| = |w|$, also $u_i \in W$ f.a. $1 \geq i \geq n$. Also existiert Pfad von S nach w im Graphen (W , E), womit die Behauptung bewiesen ist.

Eine Regl ist rechtslinear, wenn $l \in V$ und $r \in \sum V \cup \epsilon$ gilt. Eine Grammatik ist vom Typ 3 wenn sie nur rechtslineare Regeln enthält.

Zu einem gegebenen DFA definieren wir die Funktion $\hat{\delta} : Z \times \sum^* \to Z$ induktiv wie folgt, wobei $z \in Z$, $w \in \sum^+$ und $a \in \sum$:

- $\hat{\delta}(z, \epsilon) = z$
- $\hat{\delta}(z, aw) = \hat{\delta}(\delta(z, a), w)$

Der Zustand $\hat{\delta}(z, w)$ ergibt sich indem man vom Zustand z aus dem Pfad folgt der mit w beschriftet ist.

ein deterministischer endlicher Automat M ist ein 5-Tupel $M = (Z, \sum, z_0, \delta, E)$

- Z eine endliche Menge von Zuständen
- $\sum$ das Eingabealphabet (mit $Z \cap \sum = \emptyset$)
- $z_0 \in Z$ der Start/Anfangszustand (max Einer)
- $\delta : Z \times \sum \to Z$ die Übergangsfunktion
- $E \subseteq Z$ die Menge der Endzustände

Abkürzung: DFA (deterministic finite automaton)

Eine Sprache $L \supseteq \sum^*$ ist regulär, wenn es einen DFA mit $L(M) = L$ gibt (bzw. wird von einem DFA akzeptiert). Jede reguläre Sprache ist rechtslinear.

die von einem DFA akzeptierte Sprache ist: $L(M) = w \in \sum^* | \hat{\delta}(z_0, w) \in E$
Mit anderen Worten: Ein Wort w wird genau dann akzeptiert, wenn derjenige Pfad, der im Anfangszustand beginnt und dessen Übergänge mit den Zeichen von w markiert sind, in einem Endzustand endet.

induktiv wie folgt, woebei $Y \subseteq Z$, $w \in \sum^*$ und $a \in \sum$: $\hat{\delta}(Y, \epsilon) = Y$, $\hat{\delta}(Y, aw) = del\hat{t}a(\bigcup \delta(z, a), w)$

ist ein 5-Tupel $M = (Z, \sum, S, \delta, E)$ mit - Z ist eine endliche Menge von Zuständen - $\sum$ ist das Eingabealphabet - $S \subseteq Z$ die Menge der Startzustände (können mehrere sein) - $\delta : Z \times \sum \to P(Z)$ ist die (Menge der) Überführungs/Übergangsfunktion - $E \subseteq Z$ die Menge der Endzustände

¿ 1. L ist regulär (d.h. von einem DFA akzeptiert) ¿ 2. L wird von einem NFA akzeptiert ¿ 3. L ist rechtslinear (d.h. von einer Typ-3 Grammatik erzeugt)

$L(M) = w \in \sum^* | \hat{\delta}(S, w) \cap E \neq \emptyset$ (Das Wort wird akzeptiert wenn es mindestens einen Pfad vom anfangs in den endzustand gibt)

Definition

Gegeben sei eine Klasse K und ein n-stelliger Operator $\otimes : K^n \to K$.

Satz

Wenn $L \subseteq \sum^*$ eine reguläre Sprache ist,

Satz

Wenn L_1 und L_2 reguläre Sprachen sind,

Satz

Wenn L_1 und L_2 reguläre Sprachen sind,

Satz

Wenn L_1 und L_2 reguläre Sprachen sind,

Satz

Wenn L eine reguläre Sprache ist,

Satz

Wenn L eine reguläre Sprache ist,

Definition

Die Menge $Reg(\sum)$ der regulären Ausdrücke über dem Alphabet $\sum$

Definition

Für einen regulären Ausdruck $\alpha \in Reg(\sum)$ ist die Sprache $L(\alpha) \subseteq \sum^*$

Satz

Für jedes Alphabet $\sum$ ist die Menge $P(\sum^*) = L|L$Sprache über $\sum$ überabzählbar

dann ist auch $\sum^* \setminus L$ regulär

Man sagt, eine Klasse $K' \subseteq K$ ist unter $\otimes$ abgeschlossen, wenn für beliebige Elemente $k_1, k_2, ..., k_n \in K'$ gilt $\otimes(k_1, k_2, ..., k_n) \in K'$

dann ist auch $L_1 \cap L_2$ regulär.

dann ist auch $L_1 \cup L_2$ regulär.

dann ist auch L^+ regulär

dann ist auch $L_1 L_2$ regulär

ist die kleinste Menge mit folgenden Eigenschaften:

- $\varnothing \in Reg(\sum), \lambda \in Reg(\sum), \sum \subseteq Reg(\sum)$
- Wenn $\alpha, \beta \in Reg(\sum)$, dann auch $(\alpha * \beta), (\alpha + \beta), (\alpha^*) \in Reg(\sum)$

dann ist auch L^* regulär.

, d.h. es gibt keine bijektive Funktion $F : \mathbb{N} \to P(\sum^*)$.

induktiv definiert

$$L(\alpha) = \begin{cases} \varnothing & \text{falls } alpha = \not{O} \\ \epsilon & \text{falls } \alpha = \lambda \\ a & \text{falls } \alpha = a \in \sum \\ L(\beta) \cup L(\gamma) & \text{falls } \alpha = (\beta + \gamma) \\ L(\beta) L(\gamma) & \text{falls } \alpha = (\beta * \gamma) \\ (L(\beta))^* & \text{falls } \alpha = (\beta^*) \end{cases}$$

Pumping Lemma

Definition

Myhill-Nerode-Äquivalenz

Definition

Für eine Sprache L und ein Wort $x \in \sum^*$ ist $[x]_L = \{y \in \sum^* | xR_Ly\}$

Satz

Satz von Myhill-Nerode

Definition

Ein DFA M heißt reduziert,

Definition

Sei M ein DFA. Zwei Zustände $z, z' \in Z$ heißen erkennungsäquivalent

Definition

Sei M ein DFA. Dann ist $M' = (Z_{\equiv}, \sum, [z_0], \sigma', E')$ mit

Definition

Homomorphismus

Satz

surjektiver Homomorphismus

Satz

Seien M_1 und M_2 reduzierte DFAs mit $L(M_1) = L(M_2)$. Sei M_1' der Quotient von M bzgl $\equiv$

Wenn L eine reguläre Sprache ist, dann gibt es $n \leq 1$ derart, dass für alle $x \in L$ mit $|x| \geq n$ gilt: es gibt Wörter $u, v, w \in \sum^*$ mit: 1. $x = uvw$ 2. $|uv| \leq n$ 3. $|v| \geq 1$ 4. $uv^i w \in L$ für alle $i \geq 0$ Dieses Lemma spricht nicht über Automaten, sondern nur über die Eigenschaften der Sprache. Es ist geeignet, Aussagen über Nicht-Regularität zu machen. Dabei ist es aber nur eine notwendige Bedingung. Es kann nicht genutzt werden, um die Regularität einer Sprache L zu zeigen.

Für eine Sprache $L \subseteq \sum^*$ definieren wir eine binäre Relation $R_L \subseteq \sum^* \times \sum^*$ wie folgt: Für alle $x, y \in \sum^*$ setze $(x, y) \in R_L$ genau dann, wenn $\forall z \in \sum^* : (xy \in L \leftrightarrow yz \in L)$ gilt. Wir schreiben hierfür auch xR_Ly.

Sei L eine Sprache. L ist regulär $\leftrightarrow index(R_L) < \infty$ (d.h. nur wenn die Myhill-Nerode-Äquivalenz endliche Klassen hat)

die Äquivalenzklasse von x. Ist L klar, so schreiben wir einfacher $[x]$.

(in Zeichen $z \equiv z'$) wenn für jedes Wort $w \in \sum^*$ gilt: $\hat{\sigma}(z, w) \in E \leftrightarrow \hat{\sigma}(z', w) \in E$

wenn es für jeden Zustand $z \in Z$ ein Wort $x_z \in \sum^*$ gibt mit $\hat{\sigma}(l, x_z) = z$

Seien M_i DFAs (für $i \in \{1, 2\}$) und $f : Z_1 \rightarrow Z_2$ eine Funktion. Dann ist f ein Homomorphismus von M_1 auf M_2, falls gilt:

- $f(l_1) = l_2$
- $f(\sigma_1(z, a)) = \sigma_2(f(z), a)$ für alle $z \in Z_1$ und $a \in \sum$
- $z \in E_1 \leftrightarrow f(z) \in E_2$ für alle $z \in Z_1$ (bildet Endzustände aufeinander ab)

- $\sigma'([z], a) = [\sigma(z, a)]$ für $z \in Z$ und $a \in \sum$ und
- $E' = \{[z] | z \in E\}$

der Quotient von M bzgl $\equiv$

- M_2 hat wenigstens so viele Zustände wie M_1'
- Hat M_2 genauso viele Zustände wie M_1', so sind M_2 und M_1' bis auf Umbennenung der Zustände identisch (sie sind Isomorph)

Folgerung: Seien M_1 und M_2 reduzierte DFAs mit $L(M_1) = L(M_2)$. Seien M_1' und M_2' die Quotienten bzgl $\equiv$. Dann sind M_1' und M_2' isomorph, d.h. für jede reguläre Sprache gibt es (bis auf Umbenennung der Zustände) genau einen minimalen DFA
Um den minimalen DFA zu erhalten bildet man den Quotienten eines beliebigen zur Sprache passenden DFA.

Seien M_i reduzierte DFAs mit $L(M_1) = L(M_2)$. Sei weiter M_2' der Quotient von M_2 bzgl $\equiv$. Dann existiert ein surjektiver Homomorphismus von M_1 auf M_2' - die Abbildung f ist surjektiv (auf M_2). Und damit ist $M_2 < M_1$ - die Abbildung f ist ein Homomorphismus

Satz

Markierungsalgorithmus

Algorithmus Minimalautomat

Satz

Minimierungsalgorithmus

Wortproblem

Leerheitsproblem

Endlichkeitsproblem

Schnittproblem

Inklusionsproblem

Äquivalenzproblem

Kontextfreie Sprachen

Eingabe: reduzierter DFA M
Ausgabe: Menge der Paare erkennungsäquivalenter Zustände 1. Stelle eine Tabelle aller ungeordneten Zustandspaare $\{z, z'\}$ mit $z \neq z'$ auf 2. Markiere alle Paare $\{z, z'\}$ mit $z \in E$ und $z' \notin E$ 3. Markiere ein beliebiges unmarkiertes Paar $\{z, z'\}$, für das es ein $a \in \sum$ gibt, sodass $\{\sigma(z, a), \sigma(z', a)\}$ bereits markiert ist (falls möglich) 4. Wiederhole den vorherigen Schritt, bis sich keine Änderung in der Tabelle mehr ergibt

Für einen reduzierten DFA M wird ein Paar $z, z' \subseteq Z$ mit $z \neq z'$ genau dann durch den Markierungsalgorithmus markiert werden, wenn $z \not\equiv z'$

Gilt $w \in L$ für eine gegebene reguläre Sprache L und $w \in \sum^*$?
Eingabe: DFA M und $w \in \sum^*$
Verfahren: Verfolge die Zustandsübergänge von M, die durch die Symbole $a_1, ..., a_n$ vorgegeben sind.

Für einen gegebenen reduzierten DFA M markiert der Minimierungsalgorithmus ein $\{z, z'\}(z, z' \in Z, z \neq z')$ genau dann, wenn $z \not\equiv z'$

Ist eine gegebene reguläre Sprache L endlich?
Eingabe: NFA M
Verfahren: Sei $G = (Z, \rightarrow)$ wieder der gerichtete Graph mit $z \rightarrow z' \leftrightarrow \exists a \in \sum : z' \in \sigma(z, a)$. Dann gilt L(M) ist genau dann unendlich, wenn es $z \in Z, z_0 \in S$ und $z_1 \in E$ gibt mit $z_0 \rightarrow^* z \rightarrow^+ z \rightarrow^* z_1$. D.h. z liegt auf einem Zyklus, ist von einem Startzustand aus erreichbar und von z kann ein Endzustand erreicht werden. Dies kann wieder mit dem Algorithmus von Dijkstra entschieden werden.

Gilt $L = \varnothing$ für eine gegebene reguläre Sprache L?
Eingabe: NFA M
Verfahren: Sei $G = (Z, \rightarrow)$ der gerichtete Graph mit $z \rightarrow z' \leftrightarrow \exists a \in \sum : z' \in \sigma(z, a)$. Dann gilt $L(M) \neq \varnothing$ genau dann, wenn es in dem Graphen G einen Pfad von einem Knoten aus S zu einem Knoten aus E gibt. Dies kann zB mit dem Algorithmus von Dijkstra entschieden werden.

Gilt $L_1 \subseteq L_2$ für gegebene reguläre L_1, L_2?
Eingabe: NFAs M_1 und M_2
Verfahren: Aus M_1 und M_2 kann ein NFA M mit $L(M) = L(\bar{M}_2) \cap L(M_1)$ konstruieren. Es gilt $L(M_1) \subseteq L(M_2)$ genau dann, wenn $L(M) = \varnothing$.

Gilt $L_1 \cap L_2 = \varnothing$ für gegebene reguläre L_1, L_2?
Eingabe: NFAs M_1 und M_2
Verfahren: Konstruiere aus M_1 und M_2 einen NFA M mit $L(M) = L(M_1) \cap L(M_2)$. Teste ob $L(M) = \varnothing$

bei Kontext-freien Grammatiken haben alle Produktionen die Form $A \rightarrow w$ mit $A \in V$ und $w \in (V \cup \sum)^*$.

Gilt $L_1 = L_2$ für gegebene reguläre L_1, L_2?
Eingabe: NFAs M_1 und M_2
Verfahren 1: es gilt $L(M_1) = L(M_2)$ genau dann, wenn $L(M_1) \subseteq L(M_2)$ und $L(M_2) \subseteq L(M_1)$.
Verfahren 2: bestimme zu $M_i (i \in \{1, 2\})$ den äquivalenten minimalen DFA N_i. Dann gilt $L(M_1) = L(M_2)$ genau dann, wenn N_1 und N_2 isomorph sind (d.h. sie können durch Umbennenung der Zustände ineinander überführt werden).

Definition

Ableitungsbaum

Definition

Linksableitung

Definition

kontextfreie Grammatik

Definition

Chomsky Normalform

Satz

Zu jeder kontextfreien Grammatik gibt es eine Grammatik G' in Chomsky Normalform mit

Der Cocke-Younger-Kasami- oder CYK-Algorithmus

Definition

Ein Kellerautomat

Definition

Ein Konfiguration eines PDA

Definition

Seien $\gamma \in \Gamma^*, A_1 B_1, ..., B_k \in \Gamma, w, w' \in \sum^*$ **und** $z, z' \in Z$**. Dann gilt** $(z, w, A\gamma) \rightarrow (z', w', B_1 ... B_{k\gamma})$ **genau dann, wenn**

Definition

Sei M ein PDA. Dann ist die von M akzeptierte Sprache:

Eine Ableitung heißt Linksableitung wenn in jedem Schritt das am weitesten links stehende Nichtterminal ersetzt wird.

Sei G eine kontext-freie Grammatik und $X \in V \cup \sum$. Ein X-Ableitungsbaum ist ein gerichteter, geordneter Baum T mit Wurzel, dessen Knoten mit Elementen von $V \cup \sum \cup \{\epsilon\}$ beschriftet sind, wobei:

- die Wurzel mit X beschriftet ist
- Knoten v mit $a \in \sum \cup \{\epsilon\}$ beschriftet $\Rightarrow$ v ist ein Blatt
- Knoten v mit $A \in V$ beschriftet und kein Blatt $\Rightarrow$
  - es gibt eine Produktion $A \to X_1...X_r$ mit $X_1...X_r \in \sum \cup V$ $(r \geq 1)$ sodass die Nachfolgerknoten von v mit $X_1, X_2, ..., X_r$ beschriftet sind
  - oder es gibt Produktion $A \to \epsilon$ und v hat genau einen Nachfolger; dieser ist mit ϵ beschriftet
- Das Blattwort $\alpha(T)$ des X-Ableitungsbaumes T erhält man, indem man die Beschriftungen der Blätter von links nach rechts betrachtet. Ein Ableitungsbaum ist ein S-Ableitungsbaum.
- ein X-Ableitungsbaum ist vollständig, wenn seine Blätter mit Elementen von $\sum \cup \{\epsilon\}$ beschriftet sind.

Eine kontextfreie Grammatik g ist in Chomsky Normalform, falls

- alle Produktionen von G die Form $A \to AB$ oder $A \to a$ haben
- oder alle Produktionen von G die Form $A \to BC$ oder $A \to a$ oder $S \to \epsilon$ haben und S nie auf der rechten Seite einer Produktion vorkommt.

Eine Kontextfreie Grammatik G heißt mehrdeutig, wenn es zwei verschiedene vollständige Ableitungsbäume T und T' gibt mit $\alpha(T) = \alpha(T')$. Sonst heißt G eindeutig, d.h. G ist eindeutig wenn jedes Wort $w \in L(G)$ genau eine Ableitung besitzt. Eine Kontextfreie Sprache heißt inhärent mehrdeutig, wenn jede kontextfreie Grammatik mit $L = L(G)$ mehrdeutig ist.

Sei G kontextfreie Grammatik. Gesucht ist ein Algorithmus mit dessen Hilfe wir entscheiden können, ob ein gegebenes Wort zu L(G) gehört.

$$L(G) = L(G')$$

ist ein Tripel $k \in Z \times \sum^* \times \Gamma^*$ - $z \in Z$ ist der aktuelle Zustand - $w \in \sum$ ist der noch zu lesende Teil der Eingabe - $\gamma \in \Gamma^*$ ist der aktuelle Kellerinhalt. Dabei steht das oberste Kellerzeichen ganz links Übergänge zwischen Konfigurationen ergeben sich aus der Überführungsfunktion δ

M ist ein 6-Tupel $M = (Z, \sum, \Gamma, z_0, \delta, \#)$, wobei

- Z die endliche Menge der Zustände
- $\sum$ das Eingabealphabet
- Γ das Kelleralphabet
- $z_o \in Z$ der Startzustand
- $\delta : Z \times (\sum \cup \{\epsilon\}) \times \Gamma \to P_\epsilon Z \times \Gamma^*)$ die Überführungsfunktion

$L(M) = \{x \in \sum^* |$es gibt $z \in Z$mit$(z_0, x, \#)[...]^*(z, \epsilon, \epsilon)\}$

es $a \in \sum \cup \{\epsilon\}$ gibt mit $w = aw'$ und $(z', B_1...B_k) \in \delta(z, a, A)$

Definition

Sei M ein PDA. Dann ist die von M akzeptierte Sprache

Definition

eine kontextfreie Grammatik G ist in Greibach Normalform

Satz

aus einer kontextfreien Grammatik G kann eine kontextfreie Grammatik G' in Greibach Normalform berechnet werden mit

Satz

Sei L eine Sprache. Dann sind äquivalent

Definition

PDAs mit Endzuständen

Definition

Sei M ein PDAE. Die von M akzeptierte Sprache ist

Definition

ein deterministischer Kellerautomat oder DPDA ist ein PDAE M,

Definition

eine Sprache L ist deterministisch kontextfrei,

Satz

Ist $L \subseteq \sum^*$ deterministisch kontextfrei,

Satz

falls alle Produktionen aus P folgende Form haben: $A \to aB_1B_2...B_k$, mit $k \in \mathbb{N}$, $A, B_1, ..., B_k \in V$ und $a \in \sum$ Die Greibach Normalform garantiert, dass bei jedem Ableitungsschritt genau ein Alphabetsymbol entsteht.

$L(M) = \{x \in \sum^* \mid$ es gibt $z \in Z$ mit $(z_0, x, \#) \vdash^* (z, \epsilon, \epsilon)\}$

- L ist kontextfrei
- es gibt einen PDA M mit $L(M) = L$
- es gibt einen PDA M mit nur einem Zustand und $L(M) = L$. Gilt $\epsilon \notin L$, so sind diese Aussagen äquivalent zu
- es gibt einen PDA M mit nur einem Zustand und ohne eine ϵ-Transitionen, so dass $L(M) = L$ gilt

$L(G') = L(G)\ \{\epsilon\}$.
¿ Jede kontextfreie Sprache L ist Sprache eines PDA M mit nur einem Zustand. Gilt $\epsilon \notin L$, so werden keine ϵ-Transitionen benötigt
¿ Ist M ein PDA, so ist L(M) kontextfrei

$L(M) = \{w \in \sum^* |$es gibt $e \in E$ und $\gamma \in \Gamma^*$ mit $(\iota, w, \#) \vdash^* (e, \epsilon, \gamma)\}$

Ein Kellerautomat mit Endzuständen oder PDAE ist ein 7-Tupel M, wobei $(Z, \sum, \Gamma, \iota, \delta, \#)$ ein PDa und $E \subseteq Z$ eine Menge von Endzuständen ist

wenn es einen deterministischen Kellerautomaten M gibt mit $L(M) = L$

so dass für alle $z \in Z, a \in \sum, A \in \Gamma$ gilt: $|\delta(z, a, A)| + |\delta(z, \epsilon, A)| \leq 1$.

aus einem DPDA M kann ein DPDA M' berechnet werden mit $L/M') = \sum^* \setminus L(M)$

so auch $\sum^* \setminus L$

Definition

das Lemma von Ogden (William Ogden)

Definition

Wortproblem für eine kontextfreie Sprache L

Definition

Uniformes Wortproblem für kontextfreie Sprachen

Definition

Leerheitsproblem für kontextfreie Sprachen

Definition

Endlichkeitsproblem für kontextfreie Sprachen

Definition

Intuitiver Berechenbarkeitsbegriff

Definition

Ein Loop-Programm ist von der Form

Definition

Definition

Definition

Gegeben $w \in \sum^*$. Gilt $w \in L$? Ist die kontextfreie Sprache L durch eine kontextfreie Grammatik in Chomsky-Normalform gegeben, so kann das Wortproblem mit dem CYK-Algorithmus in Zeit $O(|w|^3)$ gelöst werden. Ist L durch einen deterministischen PDA gegeben, so kann das Wortproblem für L sogar in Zeit $O(n)$ gelöst werden.

Wenn L eine kontextfreie Sprache ist, dann gibt es $n \geq 1$ derart, dass für alle $z \in L$, in denen n Positionen markiert sind, gilt: es gibt Wörter $u, v, w, x, y \in \sum^*$ mit

- $z = uvwxy$
- v oder x enthält wenigstens eine der Markierungen oder
- $uv^iwx^iy \in L$ für alle $i \geq 0$

Gegeben eine kontextfreie Grammatik $G = (V, \sum, P, S)$. Gilt $L(G) = \varnothing$ Lösung: Sei $W = \{A \in V | \exists w \in \sum^* : A \Rightarrow_G^* w\}$ die Menge aller produktiven Nichtterminale. Dann gilt $L(G) \neq \varnothing \leftrightarrow S \in W$. Berechnung von W: $W_0 := \{A \in V | \exists w \in \sum^* : (A \rightarrow w) \in P\}$

Gegeben kontextfreie Grammatik G und Wort $w \in \sum^*$. Gilt $w \in L(G)$? Lösung:

- berechne kontextfreie Grammatik G' in Chomsky Normalform mit $L(G) = L(G')$
- Wende CYK-Algorithmus auf die Frage $w \in L(G')$ an

Eine Funktion $f : \mathbb{N}^k \rightarrow \mathbb{N}$ ist intuitiv berechenbar, wenn es einen Algorithmus gibt, der f berechnet, d.h.

- das Verfahren erhält $(n_1, ..., n_k)$ als Eingabe,
- terminiert nach endlich vielen Schritten
- und gibt $f(n_1, ..., n_k)$ aus.

Gegeben eine kontextfreie Grammatik G. Ist $L(G)$ endlich? O.E. können wir annehmen, daß G in Chomsky-Normalform ist. Wir definieren einen Graphen (W, E) auf der Menge der produktiven Nichtterminale mit folgender Kantenrelation: $E = \{(A, B) \in W \times W | \exists C \in W : (A \rightarrow BC) \in P$ oder $(A \rightarrow CB) \in P\}$ Beobachtung: $(A, B) \in E$ gilt genau dann, wenn es einen vollständigen A-Ableitungsbaum gibt, so daß B ein Kind der Wurzel beschriftet.

Die modifizierte Subtraktion $\div$ ist definiert durch $\div : \mathbb{N}^2 \rightarrow \mathbb{N} : (m, n) \rightarrow max(0, m - n)$

- $x_i := c, x_i := x_j + c, x_i := x_j \div c$ mit $c \in \{0, 1\}$ und i, j (Wertzuweisung) oder
- $P_1; P_2$, wobei P_1 und P_2 Loop-Programme sind (sequentielle Komposition) oder
- loop x_i do P end, wobei P ein Loop-Programm ist und i_1.

Eine Funktion $f : \mathbb{N}^k \rightarrow \mathbb{N}$ (mit $k \geq 0$) heißt loop-berechenbar, falls es ein $l \geq k$ und ein Loop-Programm P, in dem höchstens die Variablen $\forall n_1, ..., n_k \in \mathbb{N} : f(n_1, ..., n_k) = \pi_1^l([[P]]_l(n_1, ..., n_k, 0, ..., 0))$. Loop-Vermutung: Eine Funktion $\mathbb{N}^k \rightarrow \mathbb{N}$ mit $k \geq 0$ ist genau dann intuitiv berechenbar, wenn sie loop-berechenbar ist.

Für jedes Loop-Programm P, in dem keine Variable x_i mit $i > k$ vorkommt, definieren wir zunächst eine Funktion $[[P]]_k : \mathbb{N}^k \rightarrow \mathbb{N}^k$ durch Induktion über den Aufbau von P

Definition

Definition

Hilberts Vermutung (1926)

Definition

Die primitiv rekursiven Funktionen sind induktiv wie folgt definiert

Definition

Definition

Definition

Satz

Die Ackermann Funktion ist nicht berechenbar

Definition

Ein While Programm ist von der Form

Definition

Definition

Eine Funktion $\mathbb{N}^k \to \mathbb{N}$ mit $k \geq 0$ ist genau dann intuitiv berechenbar, wenn sie primitiv rekursiv ist.

Seien $k \geq 0, \mathbb{N}^k \to \mathbb{N}$ und $h : \mathbb{N}^{k+2}$. Die Funktion $f : \mathbb{N}^{k+1} \to \mathbb{N}$ mit $f(0, n_2, ..., n_{k+2}) = g(n_2, ..., n_{k+1})$ und $f(m+1, n_2, ..., n_{k+1}) = h(f(m, n_2, ..., n_{k+1}), m, n_2, ..., n_{k+1})$ ensteht aus g und h mittels Rekursion.

Seien $f, g : \mathbb{N}^{k+1} \to \mathbb{N}$ Funktionen mit

- $$g(m, \bar{n}) = \begin{cases} 1 & \text{falls } \exists i \leq m : f(i, \bar{n} \geq 1) \\ 0 & \text{sonst} \end{cases}$$
- für alle $\bar{n} \in \mathbb{N}^k$. Wir sagen, g geht durch den beschränkten Existenzwuantor aus f hervor.

- Alle konstanten Funktionen der Form $k_c : \mathbb{N}^0 \to \mathbb{N} : () \to c$ (für ein festes $c \in \mathbb{N}$) sind primitiv rekursiv.
- Alle Projektionen der Form $\pi_i^k : \mathbb{N}^k \to \mathbb{N} : (n_1, ..., n_k) \to n_i$ (mit $1 \geq i \geq k$) sind primitiv rekursiv.
- Die Nachfolgerfunktion $s : \mathbb{N} \to \mathbb{N} : n \to n + 1$ ist primitiv rekursiv.
- Wenn $f : \mathbb{N}^k \to \mathbb{N}$ und $g_1 1, ..., g_k : \mathbb{N}^l \to \mathbb{N}$ (mit $k, l \geq 0$) primitiv rekursiv sind, dann ist auch die Funktion $f(g_1, ..., g_k) : \mathbb{N}^l \to \mathbb{N}$ primitiv rekursiv (Substitution).
- Sind $g : \mathbb{N}^k \to \mathbb{N}$ und $h : \mathbb{N}^{k+2} \to \mathbb{N}$ primitiv rekursiv (mit $k \geq 0$) und entsteht $f : \mathbb{N}^{k+1} \to \mathbb{N}$ aus g und h mittels Rekursion, so ist auch f primitiv rekursiv (Rekursion).

Sei P Loop-Programm mit Variablen $x_1, x_2, ..., x_n$. Für Anfangswerte (n_i) seien (n_i') die Werte der Variablen bei Programmende.

$$f_p : \mathbb{N} \to \mathbb{N} : n \to max\{\sum_{1 \leq i \leq l} n_i' | \sum_{1 \leq i \leq l} n_i \leq n\}$$

Die Funktion $ack : \mathbb{N}^2 \to \mathbb{N}$ mit $ack(x, y,) = ack_x(y)$ heißt Ackermann Funktion

- $x_i = c; x_i = x_j + c; x_i = x_j - c$ mit $c \in \{0, 1\}$ und $i, j \geq 1$ (Wertzuweisung) oder
- $P_1; P_2$, wobei P_1 und P_2 bereits While Programme sind (sequentielle Komposition) oder
- while $x_i \neq 0$ do P end, wobei P ein While Programm ist und $i \geq 1$.

Beweis indirekt: Angenommen P wäre Loop-Programm, das ack berechnet. Nach Beschränkungslemma existiert $k \in \mathbb{N}$ mit $f_p(m) < ack_k(m)$, damit $ack_k(k) \leq f_p(2k) < ack_k(2k)$ im Widerspruch zum Monotonielemma.

wie bei Loop Programmen definieren wir zunächst für jedes While Programm P in dem keine Variable x_i mit $i > k$ vorkommt induktiv eine partielle Abbildung $[[P]]_k : \mathbb{N}^k \to \mathbb{N}^k$. Hierfür sei $\bar{n} \in \mathbb{N}^k$

- $[[x_i = c]]_k(n_1, ..., n_k) = (m_1, ..., m_k)$ genau dann, wenn $m_i = c$ und $m_l = n_l$ für $l \neq i$
- $[[x_i = x_j \pm c]]_k(n_1, ..., n_k) = (m_1, ..., m_k)$ genau dann, wenn $m_i = n_j \pm c$ und $m_l = n_l$ für $l \neq i$
- $[[P_1; P_2]]_k(\bar{n})$ ist genau dann definiert, wenn $\bar{m} = [[P_1]]_k(\bar{n}) \in \mathbb{N}^k$ und $[[P_2]]_k(\bar{m})$ definiert sind. In diesem Falle gilt $[[P_1; P_2]]_k(\bar{n}) = [[P_2]]_k([[P_1]]_k(\bar{n}))$, sonst undefiniert.

Seien $r \in \mathbb{N}$ und $D \subseteq \mathbb{N}^r$. Eine Funktion $f : D \to \mathbb{N}$ heißt partielle Funktion von $\mathbb{N}^r$ nach $\mathbb{N}$. Wir schreiben hierfür $f : \mathbb{N}^r \to \mathbb{N}$.

Definition

Definition

Gödels Vermutung

Definition

μ-rekursive Funktion

Definition

Die Klasse der μ-rekursiven Funktionen ist rekursiv definiert

Definition

Ein GoTo Programm

Definition

Definition

Definition

Eine partielle Funktion $f : \mathbb{N}^k \to \mathbb{N}$ heißt Goto berechenbar,

Definition

Definition

Eine Turingmaschine (TM)

Eine partielle Funktion $\mathbb{N}^k \to \mathbb{N}$ ist gneau dann intuitiv berechenbar, wenn sie μ-rekursiv ist.

- Alle konstanten Funktionen $k_m : \mathbb{N}^0 \to \mathbb{N} : () \to m$, alle Projektionen $\pi_i^k : \mathbb{N}^k \to \mathbb{N} : (n_1, ..., n_k) \to n_i$ und die Nachfolgerfunktion $s : \mathbb{N} \to \mathbb{N} : n \to n + 1$ sind μ-rekursiv.
- Sind $f : \mathbb{N}^k \to \mathbb{N}$ und $g_1, ..., g_k : \mathbb{N}^r \to \mathbb{N}$ μ-rekursiv, so auch $F : \mathbb{N}^r \to \mathbb{N}$ mit $F(n) = f(g_1(\bar{n}), ..., g_k(\bar{n}))$ (wobei $F(n)$ genau dann definiert ist, wenn $g_i(n)$ für alle i definiert ist und wenn f auf diesen Werten definiert ist).
- Jede partielle Funktion f , die durch Rekursion aus μ-rekursiven Funktionen entsteht, ist μ-rekursiv.
- Ist f μ-rekursiv, so auch μf.

Sei $P = A_1; A_2; ...; A_m$ ein Goto Programm, in dem keine Variable x_i mit $i > k$ vorkommt. Eine Konfiguration von P ist ein $(k + 1)$-Tupel $(n_1, n_2, ..., n_k, p) \in \mathbb{N}^k \times \{0, 1, ..., m\}$, wobei n_i die Belegung der Variablen x_i und p den Wert des Programmzählers beschreibt.

falls es ein $l \geq k$ und ein Goto Programm P, in dem keine Variable x_i mit $i > l$ vorkommt, gibt, sodass für alle $\bar{n} \in \mathbb{N}^k$ gilt:

- $f(n)$ definiert $\leftrightarrow [[P]]_l(\bar{n}, 0, ..., 0)$ definiert
- Falls $f(\bar{n})$ definiert ist, gilt $f(\bar{n}) = \pi_1^l([[P]]_l(\bar{n}, 0, ..., 0))$

ist ein 7-Tupel $M = (Z, \sum, \Phi, \delta, z_o, \square, E)$, wobei

- $\sum$ das Eingabealphabet
- Φ mit $\Phi \supseteq \sum$ und $\Phi \cap Z \neq 0$ das Arbeits- oder Bandalphabet,
- $z_0 \in Z$ der Startzustand,
- $\delta : Z \times \Phi \to (Z \times \Phi \times \{L, N, R\})$ die Überführungsfunktion
- $\square \in \Phi / \sum$ das Leerzeichen oder Blank und
- $E \subseteq Z$ die Menge der Endzustände ist

Eine partielle Funktion $f : \mathbb{N}^k \to \mathbb{N}$ heißt while Berechenbar, falls es ein $l \geq k$ und ein While Programm P, in dem höchstens die Variablen $x_1, ..., x_l$ vorkommen, gibt, sodass für alle $n_1, ..., n_k \in \mathbb{N}$ gilt:

- $f(n_1, ..., n_k)$ definiert $\leftrightarrow [[P]]_l(n_1, ..., n_k, 0, ..., 0)$ definiert
- Falls $f(n_1, ..., n_k)$ definiert ist, gilt $f(n_1, ..., n_k) = \pi_1^l([[P]]_l(n_1, ..., n_k, 0, ..., 0))$.

Sei $f : \mathbb{N}^{k+1} \to \mathbb{N}$ eine partielle Funktion Dann ist $\mu f : \mathbb{N}^k \to \mathbb{N}$ definiert durch $\mu f(n_1, ..., n_k) = min\{m | f(m, n_1, ..., n_k) = 0$ und $\forall x < m : f(x, n_1, ..., n_k)$ definiert $\}$. Dabei ist min $\varnothing$ undefiniert. Wir sagen, dass die Funktion μf aus f durch den μ-Operator hervorgeht.

ist eine endliche nichtleere Datei $P = A_1; A_2; ...; A_m$ von Anweisungen A_i der folgenden Form:

- $x_i = c, x_i = x_j + c, x_i = x_j - c$ mit $c \in \{0, 1\}$ und $i, j \geq 1$
- goto l mit $0 \leq l \leq m$ (unbedingter Sprung)
- if $x_i = 0$ then l mit $i \geq 1$ und $0 \leq l \leq m$ (bedingter Sprung)

$[[P]]_k(\bar{n})$ ist definiert, falls es $\bar{n'} \in \mathbb{N}^k$ gibt mit $(\bar{n}, 1) \vdash_P^* (\bar{n'}, 0)$. In diesem Fall gilt $[[P]]_k(\bar{n}) = \bar{n'}$

Seien $P = A_1; A_2; ...; A_m$; ein GoTo Programm und $(\bar{n}, p), (\bar{n'}, p')$ zwei Konfigurationen. Wir setzen $(\bar{n}, p) \vdash_P (\bar{n'}, p')$, falls $p > 0$ und eine der folgenden Bedingungen gilt:

- $A_p = (x_i = c), n'_i = c, n'_l = n_l$ für $l \;/\!= i$ und $p' = p + 1$
- $A_p = (x_i = x_j + c), n'_i = n_j + c, n'_l = n_l$ für $l \not= i$ und $p' = p + 1$
- $A_p = (x_i = x_j - c), n'_i = n_j - c, n'_l = n_l$ für $l \not= i$ und $p' = p + 1$
- $A_p = (gotol), \bar{n'} = \bar{n}$ und $p' = l$

Definition

Definition

Definition

Definition

Definition

Definition

Mehrband Tunringmaschine

Satz

Satz

Zu jeder Mehrband Turingmaschine M gibt es

Satz

Definition

Eine Sprache $L \subseteq \sum^*$ hei entscheidbar,

Sei $M = (Z, \sum, \Phi, \delta, z_o, \Box, E)$ eine TM und k eine Konfiguration. Dann heißt k Haltekonfiguration falls für alle Konfigurationen k' gilt: $k \vdash_M k' \Rightarrow k = k'$ (d.h. ist $k = uzav$, so gilt $\delta(z, a) = (z, a, N)$). Die Haltekonfiguration k ist akzeptierend, wenn zusätzlich $k \in \Box^* E \sum^* \Box^*$ gilt.

Eine Konfiguration einer Turingmaschine ist ein Wort $k \in \Phi^* Z \Phi^+$ Bedeutung: k=uzv

- $u \in \Phi^*$ ist Abschnitt des Bandes vor Kopfposition der bereits besucht wurde
- $z \in Z$ ost aktueller Zustand
- $c \in \Phi^+$ ist Abschnitt des Bandes ab Kopfposition, der Besicht wurde oder im Bereich des Eingabewortes liegt.

Eine partielle Funktion $f : \sum^* \to \sum^*$ heißt Turing berechenbar, wenn es eine TM M gibt mti $g_M = f$.

Sei $M = (Z, \sum, \Phi, \delta, z_o, \Box, E)$ eine TM. Die von M berechnete partielle Funktion $f_M : \sum^* \to \sum^*$ erfüllt f+r alle $x, y \in \sum^* : f_M(x) = y \leftrightarrow \exists z_e \in E, i, j, \in \mathbb{N} :$ $z_0 x \Box \vdash_M^* \Box^i z_e y \Box^j$ und $\Box^i z_e y \Box^j$ ist Haltekonfiguration.

- Eine Mehrband-Turingmaschine besitzt $k (k \geq 1)$ Bänder mit k unabhängigen Köpfen, aber nur eine Steuereinheit.
- Aussehen der Übergangsfunktion: $\delta : Z \times \Phi^k \to (Z \times \Phi^k \times \{L, N, R\}^k)$ (ein Zustand, k Bandsymbole, k Bewegungen)
- Die Ein- und Ausgabe stehen jeweils auf dem ersten Band. Zu Beginn und am Ende (in einer akzeptierenden Haltekonfiguration) sind die restlichen Bänder leer.

Sei $f : \mathbb{N}^k \to \mathbb{N}$ eine partielle Funktion. Definiere eine partielle Funktion $F : \{0, 1, \#\}^* \to \{0, 1, \#\}^*$ durch

$$F(w) = \begin{cases} bin(f(n_1, \dots, n_k)) & \text{falls } w = bin(n_1)\#bin(n_2)\# \dots \#bin(n_k) \text{ und} \\ \text{undefiniert} & textsonst \end{cases}$$

Dann heißt f Turing berechenbar, wenn F Turing berechenbar ist. ¿ (Für $n \in \mathbb{N}$ sei $bin(n)$ die Binärdarstellung der Zahl n)

Sei $g : \sum^* \to \sum^*$ eine Turing-berechenbare partielle Funktion. Dann wird g von einer TM M berechnet, für die gilt: $\forall x \in \sum^* \forall k$ Haltekonfiguration: $z_o x \Box \vdash_M^* k \Rightarrow k \in \Box^* E \sum^* \Box^*$.

eine (Einband) Turingmaschine M' die diesselbe Funktion löst Beweis:

- Simulation mittels Einband-Turingmaschine durch Erweiterung des Alphabets: Wir fassen die übereinanderliegenden Bandeinträge zu einem Feld zusammen und markieren die Kopfpositionen auf jedem Band durch $*$. Neues Bandalphabet: $\Phi' = \sum \uplus \{\Box\} \uplus (\Phi \times \{*, \diamond\})^k$
- Alphabetsymbol der Form $(a, *, b, \diamond, c, *, ...) \in (\Phi \times \{*, \diamond\})^k$ bedeutet: 1. und 3. Kopd anwesen ($*$ Kopf anwesend, $\diamond$ Kopf nicht anwesend)

falls die charakteristische Funktion von L, d.h. die Funktion $\chi_L : \sum^* \to \{0, 1\}$ mit

$$\chi_L(w == \begin{cases} 1 & \text{falls } w \in L \\ 0 & \text{falls } w \notin L \end{cases}$$

berechenbar ist. Eine Sprache die nicht entscheidbar ist, heißt unentscheidbar.

Sind $f : \mathbb{N}^k \to \mathbb{N}$ und $g_1, g_2, \dots, g_k : \mathbb{N}^l \to \mathbb{N}$ Turing berechenbar, so auch die partielle Funktion $f(g_1, g_2, \dots, g_k) : \mathbb{N}^l \to \mathbb{N}$

Definition

Definition

das allgemeine Halteproblem ist die Sprache

das spezielle Halteproblem ist die Sprache

Satz

Definition

Satz

Satz

Satz

Definition

Satz

Satz

Sei $L \subseteq \sum^*$ eine nichtleere Sprache. Dann sind äquivalent

$K = \{w \in L_{TM} | M_w \text{ angesetzt auf w hält}\}$

$H = \{w\#x | w \in L_{TM}, x \in \{0,1\}^*, M_w \text{ angesetzt auf x hält}\}$

Seien $A \subseteq \sum^*, B \subseteq \Phi^*$. Eine Reduktion von A auf B ist eine totale und berechenbare Funktion $f : \sum^* \to \Phi^*$, so dass für alle $w \in \sum^*$ gilt: $w \in A \leftrightarrow f(x) \in B$. A heißt auf B reduzierbar (in Zeichen $A \leq B$), falls es eine Reduktion von A auf B gibt.

Das spezielle Halteproblem ist unentscheidbar

Das Halteproblem auf leerem Band ist unentscheidbar

Das allgemeine Halteproblem ist unentscheidbar

Eine Sprache $L \subseteq \sum^*$ heißt semi-entscheidbar, falls die "halbe" charakteristische Funktion von L, d.h. die partielle Funktion $X'_L : \sum^* \to \{1\}$ mit $x'_L = \begin{cases} 1 & \text{falls } w \in L \\ undef. & \text{falls } w \notin L \end{cases}$ berechenbar ist.

Satz von Rice: Sei R die Klasse aller Turing-berechenbaren Funktionen $\{0,1\}^* \to \{0,1\}^*$, Ω die nirgendwo definierte Funktion und sei $S \subseteq R$ mit $\Omega \in S$ und $\neq R$. Dann ist die Sprache $C(S) = \{w \in L_{TM} | \phi_w \in S\}$ unentscheidbar.

- L ist semi-entscheidbar
- L wird von einer Turing-Maschine akzeptiert
- L ist vom Typ 0 (d.h. von einer Grammatik erzeugt)
- L ist Bild einer berechenbaren partiellen Funktion $\sum^* \to \sum^*$
- L ist Bild einer berechenbaren totalen Funktion $\sum^* \to \sum^*$
- L ist rekursiv aufzählbar
- L ist Definitionsbereich einer berechenbaren

Ein Problem $L \subseteq \sum^*$ ist gneua dann entscheidbar, wenn sowohl L als auch $\bar{L} = \sum^* \setminus L$ semi-entscheidbar sind. 1. $w \in L$, dann existiert $t \in \mathbb{N}$, so dass M_L nach t Schritten terminiert. Wegen $w \notin \bar{L}$ terminiert $M_{\bar{L}}$ niemals. 2. $w \notin L$, dann existiert $t \in \mathbb{N}$, so dass $M_{\bar{L}}$ nach t Schritten terminiert. Wegen $w \notin L$ terminiert M_L niemals.
Dieses letzte Argument heißt mitunter "Schwalbenschwanz-Argument".

Definition

Definition

Definition

Satz

Satz

Satz

Satz

Definition

Satz

Satz

Eine Sprache $L \subseteq \sum^*$ heißt rekursiv aufzählbar, falls $L \notin \varnothing$ oder es eine totale und berechenbare Funktion $f : \mathbb{N} \to \sum^*$ gibt mit $L = \{f(n) | n \in \mathbb{N}\} = \{f(0), f(1), f(2), ...\}$.

Sei M eine Turing Maschine. DIe von M akzeptierte Sprache ist $L(M) = \{w \in \sum^* |$es gibt akzept. Haltekonf. mit $z_0 w \Box \vdash_M^* k\}$.

Es gibt eine universelle Turing Maschine Beweis: eine Turing Maschine mit drei Bändern. - 1.Band: Kode w der zu simulierenden Turing Maschine M_w - 2.Band: aktueller Zustand der zu simulierenden Turing Maschine M_w - 3.Band: augenblicklicher Bandinhalt der Turing Maschine M_w 1. Initialisierung: auf 1.Band steht w000x mit $w \in L_{TM}$. Kopiere x auf 3.Band und lösche 000x auf erstem, schreibe 010 auf 2.Band 2. Simulation: stehen auf 2.Band $01^{i+1}0$ und auf 3. an Kopfposition j, so suche auf 1.Band Anweisung $(z_{i'}, a_{j'}, y) = \delta(z_i, a_j)$ und schreibe $01^{i'+1}0$ auf 2.Band; ersetzte j an Kopfposition auf 3.Band durch j'; bewege 3.Kopf entsprechend y nah rechts, links oder aber auch nicht. 3. Aufräumen: bei Erreichen einer akzeptierenden Haltekonfiguration auf 3.Band

Eine Turing Maschine U heißt universelle Turing Maschine, wenn sie die folgende partielle Funktion berechnet. $\{0,1\}^* \to \{0,1\}^*$ ¿

$$y \to \begin{cases} \phi_w(x) & \text{falls } y = w000x, w \in L_{TM}, x \in \{0,1\}^* \\ undef. & \text{sonst} \end{cases}$$

- U hält bei Eingabe $w000x$ genau dann, wenn M_w bei Eingabe x hält - U akzeptiert $w000x$ genau dann, wenn M_w das Wort x akzeptiert

es gibt eine Grammatik G, deren Wortproblem $L(G)$ unentscheidbar ist.

Folgerung: es gibt eine Typ-0 Sprache, die nicht vom Typ 1 ist.

das spezielle Halteproblem $K = \{w \in L_{TM} | M_w$ angesetzt auf w hält$\}$ ist semi-entscheidbar.

¿ 1. Ein Korrespondezsystem ist eine endliche Folge von Paaren $K = ((x_1, y_1), (x_2, y_2), ..., (x_k, y_k))$ mit $x_i, y_i \in \sum^+$ für alle $1 \leq i \leq k$ (dabei ist $\sum$ ein beliebiges Alphabet) ¿ 2. Eine Lösung von K ist eine endliche Folge von Indizes $i_1, i_2, ..., i_n \in \{1, 2, ..., k\}$ mit $n \geq 1$ und $x_{i1} x_{i2} ... x_{in} = y_{i1} y_{i2} ... y_{in}$. ¿ 3. MPCP ("modifiziertes PCP") ist die Menge der Korrespondezsysteme, die eine Lösung mit $i_1 = 1$ besitzen ¿ 4. PCP ist die Menge der Korrespondenzsysteme, die eine Lösung besitzen

das allgemeine Wortproblem $A = \{(G, w) |$ G ist Grammatik mit $w \in L(G)\}$ ist unentscheidbar.

PCP ist semi-entscheidbar.

Satz (Emil Post, 1947): PCP ist unentscheidbar. (T. Neary 2015: 5 Paare reichen hierfür.)

Satz

Satz

Satz

Definition

Church-Turing These

Definition

Unentscheidbarkeit

Definition

Intuitiver Effizienzbegriff

Definition

Deterministische Zeitklassen

Definition

REACH

Definition

Euler-Kreise

Satz

Satz (Euler 1707-1783, 1736)

Satz (Stearns 1967): Das Regularitätsproblem für DPDAs $Reg_{DPDA} = \{P | \text{P DPDA mit L(P) regulär}\}$ ist entscheidbar.

Das Regularitätsproblem für PDAs $Reg_{PDA} = \{P | \text{P PDA mit L(P) regulär}\}$ ist nicht semi-entscheidbar.

Die Funktionen, die durch Turingmaschinen bzw. While/Goto-Programme berechnet werden können, sind genau die intuitiv berechenbaren Funktionen.

Das Schnittproblem für DPDAs $Schn_{DPDA} = \{(P_1, P_2) | P_1, P_2 \text{ DPDAs mit } L(P_1) \cap L(P_2) = \varnothing\}$ ist nicht semi-entscheidbar.

Das Wortproblem einer Sprache L ist effizient entscheidbar, wenn es einen Algorithmus gibt, der die Antwort auf die Frage "Gehört das Wort w zu L?" mit geringen Ressourcen (Zeit, Speicherplatz) bestimmt. "mit geringen Ressourcen" heißt hier, daß die benötigten Ressourcen nur moderat mit der Eingabelänge $|w|$ wachsen.

Probleme, die nicht durch Turing-Maschinen gelöst werden können, sind damit prinzipiell unlösbar (wenn auch u.U. semi-entscheidbar). Beispiele:

- die verschiedenen Versionen des Halteproblems
- Posts Korrespondenzproblem
- das Schnitt- und verwandte Probleme über kontextfreie Sprachen

ist die Menge der gerichteten Graphen mit zwei ausgezeichneten Knoten s und t, in denen es einen Pfad von s nach t gibt. REACH ist in P. (Beweis: z.B. mit Dijkstras Algorithmus)

Sei $f : \mathbb{N} \to \mathbb{N}$ eine monotone Funktion. Die Klasse $TIME(f)$ besteht aus allen Sprachen L, für die es eine Turingmaschine M gibt mit:

- M berechnet die charakteristische Funktion von L.
- Für jede Eingabe $w \in \sum^*$ erreicht M von der Startkonfiguration $z_0 w \Box$ aus nach höchstens $f(|w|)$ Rechenschritten eine akzeptierende Haltekonfiguration (und gibt 0 oder 1 aus, je nachdem ob $w \notin L$ oder $w \in L$ gilt).

Ein Graph (V, E) enthält einen Eulerkreis genau dann, wenn er höchstens eine Zusammenhangskomponente mit > 1 Knoten enthält und jeder Knoten geraden Grad hat (d.h. jeder Knoten hat eine gerade Anzahl von Nachbarn). Folgerung: EC ist in P, denn die genannten Bedingungen lassen sich in polynomieller Zeit prüfen. Die erweiterte Church-Turing These: P umfaßt die Klasse der effizient lösbaren Probleme.

EC ist die Menge der ungerichteten Graphen, die einen Eulerkreis (d.h. einen Kreis, der jede Kante genau einmal durchläuft) enthalten.

Definition

Deterministische Platzklassen

Definition

Definition

Definition

SAT

Definition

Hamilton-Kreise HC

Definition

3-Färbbarkeit 3C

Definition

Sei M NTM. Die von M akzeptierte Sprache ist

Satz

Determinisierbarkeit von NTM

Definition

Nichtdeterministische Zeitklassen

Definition

Nichtdeterministische Platzklassen

Satz

Satz von Kuroda (1964)

$$PSPACE = \bigcup_{f \in Poly} SPACE(f)$$

$$EXPSPACE = \bigcup_{f \in Poly} SPACE(2^f)$$

$$2EXPSPACE = \bigcup_{f \in Poly} SPACE(2^{2^f})...$$

Sei $f : \mathbb{N} \to \mathbb{N}$ eine monotone Funktion. Die Klasse $SPACE(f)$ besteht aus allen Sprachen L, für die es eine Turingmaschine M gibt mit:

- M berechnet die charakteristische Funktion von L.
- Für jede Eingabe $w \in \sum^*$ hat jede von der Startkonfiguration $z_0 w \Box$ aus erreichbare Konfiguration höchstens die Länge $f(|w|)$.

ist die Menge der ungerichteten Graphen, die einen Hamiltonkreis (d.h. einen Kreis, der jeden Knoten genau einmal besucht) enthalten.

ist die Menge der erfüllbaren aussagenlogischen Formeln.
Beobachtung: SAT 2 PSPACE

$L(M) = \{w \in \sum^* \mid$ es gibt akzept. Haltekonf. k mit $z_0 w \Box \vdash_M^* k\}$.

: 3C ist die Menge der ungerichteten Graphen, deren Knoten sich mit drei Farben färben lassen, so daß benachbarte Knoten unterschiedliche Farben haben.
Beobachtung: $3C \in PSPACE$

Sei $f : \mathbb{N} \to \mathbb{N}$ eine monotone Funktion. Die Klasse $NTIME(f)$ besteht aus allen Sprachen L, für die es eine nichtdeterministische Turingmaschine M gibt mit:

- M akzeptiert L.
- Für jede Eingabe $w \in \sum^*$ hält M auf jeden Fall nach $f(|w|)$ vielen Schritten.

Definition

- $$NP = \bigcup_{f \in Poly} NTIME(f)$$
- $$NEXPTIME = \bigcup_{f \in Poly} NTIME(2^f)$$
- $$2NEXPTIME = \bigcup_{f \in Poly} NTIME(2^{2^f})...$$

Lemma: $NP \subseteq PSPACE, NEXPTIME \subseteq EXPSPACE, 2NEXPTIME \subseteq 2EXPSPACE...$

Zu jeder nichtdeterministischen Turingmaschine gibt es eine Turingmaschine, die dieselbe Sprache akzeptiert.

Sei L eine Sprache. Dann sind äquivalent

- 1. L ist kontextsensitiv (d.h. vom Typ 1)
- 2. $L \in NSPACE(n)$

Sei $f : \mathbb{N} \to \mathbb{N}$ eine monotone Funktion. Die Klasse $NSPACE(f)$ besteht aus allen Sprachen L, für die es eine nichtdeterministische Turingmaschine M gibt mit: - M akzeptiert L. - Für jede Eingabe $w \in \sum^*$ folgt $|k| \le f(|w|)$ aus $z_0 w \Box \vdash_M^* k$.

Satz

Satz von Savitch (1970)

Definition

NP-Vollständigkeit

Satz

SAT Vollständigkeit

Definition

3-SAT

Satz

3-SAT vollständigkeit

Satz

SAT und 3-SAT vollständigkeit

Satz

ist eine Formel ϕ in KNF mit höchstens zwei Literalen pro Klausel erfüllbar?

Satz

Erfüllbarkeitsprobleme vollständigkeit

Definition

kC

Satz

3C vollständigkeit

Eine Sprache B ist NP-hart, falls für alle $A \in NP$ gilt: $A \leq_P B$ (A ist mindestens so schwer wie jedes Problem in NP). Eine Sprache ist NP-vollständig, falls sie zu NP gehört und NP-hart ist.

Für jede super-lineare monotone Funktion $f : \mathbb{N} \to \mathbb{N}$ gilt $NSPACE(f(n)) \subseteq SPACE((f(n))^2)$. Damit haben wir die folgende Struktur der Komplexitätsklassen:

1. P
2. NP
3. PSPACE = NPSPACE
4. EXPTIME
5. NEXPTIME
6. EXPSPACE = NEXPSPACE

3-SAT ist die Menge der erfüllbaren aussagenlogischen Formeln in konjunktiver Normalform mit höchstens drei Literalen pro Klausel.

Stephen Cook & Leonid Levin: SAT ist NP-vollständig.

Die Probleme SAT und 3-SAT sind NP-vollständig.

Das Problem 3-SAT ist NP-vollständig.

- Die Erfüllbarkeitsprobleme SAT und 3-SAT sind NP-vollständig.
- Das Erfüllbarkeitsproblem 2-SAT ist in P.

Es ist in Polynomialzeit entscheidbar, ob eine Formel ϕ in KNF mit höchstens zwei Literalen pro Klausel erfüllbar ist. Beweisidee: konstruieren gerichteten Graphen G:

- Für jede atomare Formel x aus ϕ gibt es die Knoten x und $\neg x$.
- Für jede Klausel $\alpha \vee \beta$ in ϕ gibt es Kanten $\sim \alpha \to \beta$ und $\sim \beta \to \alpha$, wobei $\sim x = \neg x$ und $\sim \neg x = x$ gelte.

Das Problem 3C ist NP-vollständig.

kC ist die Menge der ungerichteten Graphen, die sich mit k Farben färben lassen.
Ein Graph ist genau dann 2-färbbar, wenn er bipartit ist. Das Problem 2C ist also in P.

DHC - Gerichteter Hamiltonkreis

Definition

DHC

Satz

DHV vollständigkeit

HC - Ungerichteter Hamiltonkreis

Definition

HC

Satz

HC vollständigkeit

TSP - Travelling Salesman

Satz

Das Problem TSP

Church-Turing These

Unentscheidbarkeit

DHC ist die Menge der gerichteten Graphen, die einen Hamiltonkreis enthalten.

- EINGABE: ein gerichteter Graph $G = (V, E)$ mit Knotenmenge V und Kantenmenge $E \supseteq V \times V$.
- FRAGE: Besitzt der Graph G einen Hamiltonkreis, d.h. kann man den Graphen so durchlaufen, dass jeder Knoten genau einmal besucht wird?

- EINGABE: ein ungerichteter Graph $G = (V, E)$ mit Knotenmenge V und Kantenmenge $E \supseteq \binom{V}{2} = \{X \subseteq V || X| = 2\}$.
- FRAGE: Besitzt der Graph G einen Hamiltonkreis, d.h. kann man den Graphen so durchlaufen, dass jeder Knoten genau einmal besucht wird?

Das Problem DHC ist NP-vollständig.

as Problem HC ist NP-vollständig.

ist die Menge der ungerichteten Graphen, die einen Hamiltonkreis enthalten.

ist NP-vollständig.

- Beweis: $TSP \in NP$ ist einfach zu sehen, da eine Indexfolge geraten und in polynomieller Zeit überprüft werden kann, ob sie die Bedingungen erfüllt.
- Für die NP-Härte zeigen wir $HC \leq_P TSP$: Sei $G = (V, E)$ ein ungerichteter Graph, o.E. $V = \{1, ..., n\}$. Wir konstruieren dazu folgende Matrix: $M_{i,j} = \begin{cases} 1 & \text{falls } \{i,j\} \in E \\ 2 & \text{falls } \notin E \end{cases}$
- Außerdem setzen wir $d = n$.

- EINGABE: eine $n \times n$-Matrix $M = (M_{i,j})$ von Entfernungen zwischen n Städten und eine Zahl d.
- FRAGE: Gibt es eine Tour durch alle Städte, die maximal die Länge d hat? Das heißt, gibt es eine Indexfolge $i_1, ..., i_m$, so dass gilt:
- $\{i_1, ..., i_m\} = \{1, ..., n\}$ (jede Stadt kommt vor)
- $M_{i_1,i_2} + M_{i_2,i_3} + ... + M_{i_{m-1},i_m} + M_{i_m,i_1} \leq d$ (die Länge Tour ist höchstens d)

Probleme, die nicht durch Turing-Maschinen gelöst werden können, sind damit prinzipiell unlösbar (wenn auch u.U. semi-entscheidbar).

Die Church-Turing These besagt, dass die Funktionen, die durch Turingmaschinen bzw. While-/Goto-Programme berechnet werden können, genau die intuitiv berechenbaren Funktionen sind.

Probleme, die durch Turing-Maschinen nicht in Polynomialzeit gelöst werden können, sind damit prinzipiell nicht effizient lösbar.

Die erweiterte Church-Turing These besagt, dass die Funktionen, die durch Turingmaschine bzw. While-/Goto-Programme in Polynomialzeit berechnet werden können, genau die intuitiv und effizient berechenbaren Funktionen sind.