Find any errors? Please send them back, I want to keep them!

Allgemeines

Definitionen

- $\mathbb{N}^0 = \{0, 1, 2, \ldots\}$
- $\mathbb{N}^{\setminus\{0\}} = \{1, 2, \ldots\}$

Alphabet: endliche Menge von Zeichen $\Sigma = \{a, b\}$

Wort Zeichenfolge $w = abab \in \Sigma^*$

Σ^***:** Menge der Wörter

Σ^+**:** $\Sigma^* \setminus \{\varepsilon\}$ $\{\} \neq \varepsilon \neq \{\varepsilon\}$

Operator $\circ$**:** Konkatenation $\varepsilon \circ w = w, w \circ w = ww = w^2$

$w^n = \underbrace{www \ldots w}_{n-mal}$ $n \in \mathbb{N}^0$

$|w_1|$**:** Wortlänge $|w| = 5$

$|w|_\sigma, \sigma \in \Sigma$ Anzahl σ in Σ $|w|_a = 3$

Seiten A, B Mengen (Sprachen): $A = \{\varepsilon, a, bba, bbabb, B = \{a, aa, aaa\}$

$|A|$ Mächtigkeit $=$ Anzahl der Elemente

Sprache: Teilmenge von Σ^*

Vereinigung $A \cup B = \{x \in \Sigma^* | x \in A \vee x \in B\}$

Schnitt $A \cap B = \{c \in \Sigma^* | x \in A \wedge x \in B\}$

Komplement $\overline{A} = \{x \in \Sigma^* | x \notin A\} = \Sigma^* \setminus A$

Produkt $AB = \{xy \in \Sigma^* | x \in A \wedge y \in B\}$

De Morgan Regeln

$$\overline{A \cup B} = \overline{\overline{A} \cap \overline{B}}$$
$$\overline{A \cap B} = \overline{\overline{A} \cup \overline{B}}$$

Klassen

Sei $\mathcal{C}$ eine Klasse von Sprachen. $\mathcal{C}$ heisst $\left|\begin{array}{c}\text{vereinigungs-}\\ \text{schnitt-}\\ \text{komplement-}\\ \text{produkt-}\end{array}\right|$ abgeschlossen $\Leftrightarrow A \in \mathcal{C} \wedge B \in \mathcal{C}$ $\left|\begin{array}{c} A \cup B \in \mathcal{C} \\ A \cap B \in \mathcal{C} \\ \overline{A} \in \mathcal{C} \\ AB \in \mathcal{C} \end{array}\right|$.

A ist abgeschlossen gegen Vereinigung und Komplement $\Leftrightarrow$ A ist abgeschlossen gegen Schnitt und Komplement.

$$
\begin{aligned}
A^0 &= \{\varepsilon\} \\
A^1 &= A \\
A^{n+q} &= A^n A \\
A^i A^j &= A^{i+j} \\
\left(A^i\right)^j &= A^{i \cdot j} \\
A^* &= \bigcup_{n \geq 0} A^n && = A^0 \cup A^1 \cup A^2 \cup \dots \\
A^+ &= \bigcup_{n > 0} A^n && = \quad A^1 \cup A^2 \cup \dots \\
&= \bigcap_{n \geq 0} A^n && = A^0 \cap A^1 \cap A^2 \cap \dots \\
\text{Potenzmenge} &= 2^A = \{B | B \subseteq A\} \\
M_a &= \{x \in \Sigma^* | x = n^a\} \\
M_2 &= \{1, 4, 9, 16, \dots\} \\
M_3 &= \{1, 8, 27, 64, \dots\} \\
M_4 &= \{1, 16, 81 \dots\} \\
\bigcup_{i=2}^{4} M_i &= \{1, 4, 8, 9, 16, \dots\} \\
\bigcap_{i=2}^{4} M_i &= \{1^1 2, 2^1 2, 3^1 2, \dots\} \\
M &= \{1, 2, 3\} \\
2^M &= \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}
\end{aligned}
$$

Kreuzprodukt:
Seien A_i Mengen,

$$
A_1 \times A_2 \times A_3 \times \dots \times A_n = \{(a_1, a_2, a_3, \dots, a_n) | a_i \in A_i, i \in \mathbb{N}^n\}
$$

$$
\chi_i^n \leq 1 A_i = \begin{cases} \emptyset & i < 1 \\ A_1 & n = 1 \\ \chi_{i=1}^{n-1} A_i \times A_n & n > 1 \end{cases}
$$

Relationen

Relationen, $\tau_{A_i} \subseteq \chi_{i=1}^{n} A_i$ setzen Elemente von Mengen zueinander in Beziehung.
Bsp:

$$\{a, b, \dots, z\}\tau\{1, 2, \dots, 26\} = \{(a, 1), (b, 2), \dots, (z, 26)\}$$
$$\mathbb{N}^0 \leq \mathbb{N}^0 = \{(a, b)|a + c = b,\ a, b, c \in \mathbb{N}^0\}$$

Eigenschaften von Relationen auf gleichen Mengen A:

reflexiv: $\forall a \in A : a\tau a$

irreflexiv: $\forall a \in A : \overline{a\tau a}$

symmetrisch: $\forall a, b \in A : a\tau b \Rightarrow b\tau a$

antisymmetrisch: $\forall a, b \in A : (a\tau b \wedge b\tau a) \Rightarrow a = b$ (antisymmetrisch $\Rightarrow$ reflexiv)

asymmetrisch: $\forall a, b \in A : a\tau b \Rightarrow \overline{b\tau a}$

transitiv: $\forall a, b, c \in A : (a\tau b \wedge b\tau c) \Rightarrow a\tau c$

äquivalent:

Ordnungsrelationen Typ „$\leq$“: reflexiv, transitiv, antisymmetrisch

Ordnungsrelationen Typ „$<$“: irreflexiv, transitiv, antisymmetrisch

τ^+ transitive Hülle

τ^* reflexive, transitive Hülle

Satz: τ^* ist die kleinste, reflexive und transitive Relation (Hülle), die τ selbst umfasst.

$geg\ A = \{1, 2, 3, 4, 5\}$

$$R, S \subseteq A \times A$$
$$R = \{(1, 2), (2, 3), (2, 5), (3, 4), (5, 4)\}$$
$$S = \{(1, 3), (3, 5), (5, 1)\}$$

$ges\ R^+, S^+, R^*, S^*$

$$R^+ = R \cup \{(1, 3), (1, 4), (2, 4), (1, 5)\}$$
$$R^* = R^+ \cup \{(1, 1), (2, 2), (3, 3), (5, 5)\}$$
$$S^+ = S^+ \cup \{(1, 5), (1, 1), (3, 1), (3, 3), (4, 5), (5, 3)\}$$
$$S^* = S^* \cup \{(2, 2), (4, 4)\}$$

Bsp

a „ist blutsverwandt“ auf Menge der Personen

b „x ist Teiler von y“, d.h. $\exists z$ mit $x \cdot z = y\ x, y, z \in \mathbb{Z}$

c $R \subseteq \mathbb{Z} \times \mathbb{Z}$ mit $R = \{(x, y)|xy > 0\}$

d „w ist Präfix von v“, d.h. $\exists u$ mit $v = wu,\ \ u, v, w \in \Sigma^*,\ \Sigma$ sei Menge

	refl.	irrefl.	symm.	antiymm.	asymm.	trans.	äquiv.	„<“	„≤“
a	✔	✘	✔	✘	✘	✘	✘	✘	✘
b	✔	✘	✘	✘	✘	✔	✘	✘	✘
c	✘	✘	✔	✘	✘	✔	✘	✘	✘
d	✔	✘	✘	✔	✘	✔	✘	✔	✘

Äquivalenzklasse: Sei $R \subseteq A \times A$ eine ÄR, dann heissen für $a \in A$ die Mengen $[a]_R := \{b \in A | aRb\}$ Äquivalenzklassen von R.

Partitionierung: Menge aller ÄK $A/_R := \{[a]_R | a \in A\}$

Index Anzahl der ÄK $|A/_R|$
Bsp $R = \{(x, y) | x \mod 2 = y \mod 2\}$ partitioniert $\mathbb{N}^0$ in 2 ÄK:

- $[0]_a = \{0, 2, 3, \dots\}$
- $[1]_a = \{1, 3, 5, \dots\}$

Automatentheorie und formale Sprachen

Grammatik

Eine Grammatik ist ein 4-Tupel $G = (V, \Sigma, P, S)$

V - Variblen

Σ - Terminalalphabet

P - Regeln/Produktionen

S - Startvariable

$|V| < \infty$
$|\Sigma| < \infty$
$|P| < \infty$

Chomsky-Hirarchie

Typ 0: (Phrasenstrukturgrammatik) - keine Einschränkungen

Typ 1: (kontextsensitiv) - $(w_1 \to w_2) \Rightarrow (|w_1| \leq |w_2|)$ (Wort wird nicht kürzer)

Typ 2: (kontextfrei) - $(w_1 \to w_2 \Rightarrow (w_1 \in V)$ w_1 ist einzelne Variable

Typ 3: (regulär) - $w_2 \in \Sigma \cup \Sigma V$ „rechte Seiten" von Regeln Terminalsymbol oder Terminalsymbole gefolgt von Variablen

Alle Sprachen der Typen 1,2 und 3 sind *entscheidbar*.

ε-Sonderregelung (Zulassen des leeren Wortes ε in Typ 1,2 oder 3)

- Regel hinzufügen: $S \to \varepsilon$
- Verhindern von S auf rechter Seite von Regeln: Regel mit „$\to S$" ersetzen durch „$\to S'$"
- Zulassen von $A \to \varepsilon$ (verändert Sprache nicht)
  Algorithmus:
  1. Zerlege $V \to V_1, V_2$, $(A \Rightarrow^* \varepsilon) \in V_1$ und $V_1 \cap V_2 = \emptyset$.
  2. Entferne alle $A \to \varepsilon$, füge für $(B \to xAy)$ $(B \to xy)$ hinzu.

Wortproblem (Gehört ein Wort zu einer Sprache?)

$(\exists \text{Algorithmus})[(\text{Algo terminiert in endl. Zeit} \land (Algoentscheidet(x \in \mathcal{L}(G)) \lor (x \notin \mathcal{L}(G)))]$
$\Rightarrow$ das Wortproblem ist für Typ 1,2 und 3 entscheidbar (aber NP-hart für Typ 1)

Syntaxbäume

Wurzel: S
Für $i = 1, 2, \ldots, n$ $A \to z \in P \Rightarrow |z|$ viele Söhne $\to$ „weitere Kette“

Linksableitung: Variable am weitesten links wird abgeleitet.

Rechtsableitung: Variable am weitesten rechts wird abgeleitet.

mehrdeutige Grammatik: für ein x verschiedene Syntaxbäume möglich

- Mehrdeutigkeit kann oft beseitigt werden.
- Ist dies nicht möglich $\Rightarrow$ *inhärent mehrdeutig*

Backus-Naur-Form Bnf (Typ 2 Grammatiken)

Metaregeln für selbe linke Seite

$$\left.\begin{array}{ccc} A & \to & \beta_1 \\ A & \to & \beta_2 \\ & \vdots & \\ A & \to & \beta_3 \end{array}\right\} A \to \beta_1 | \beta_2 | \ldots \beta_n$$

erweiterte Backus-Naur-Form Ebnf

$$A \to \alpha[\beta]\gamma \Rightarrow \left\{\begin{array}{ccc} A & \to & \alpha\gamma \\ A & \to & \alpha\beta\gamma \end{array}\right.$$

$$A \to \alpha\{\beta\}\gamma \Rightarrow \left\{\begin{array}{ccc} A & \to & \alpha\gamma \\ A & \to & \alpha B\gamma \\ B & \to & \beta \\ B & \to & \beta B \end{array}\right.$$

Reguläre Sprachen

Endliche (deterministische) Automaten DFA

$M = (Z, \Sigma, \delta, z_0, E)$

Σ Eingabealphabet

z_0 Startzustand

$z \cap \Sigma = \emptyset$
$\to$ Zustandsgraphen

Z Menge der Zustände

$E \subseteq Z$ Menge der Endzustände

$\delta : z \times E \to z$
Überführungsfunktion

$|z| < \infty, \ |\Sigma| < \infty$

akzeptierte Sprache

Die von M akzeptierte Sprache ist:

$$T(M) = \{x \in \Sigma^* | \hat{\delta}(z, e)\}$$

wobei

$$\hat{\delta}(z, \varepsilon) = z$$
$$\hat{\delta}(z, ax) = \hat{\delta}\left(\delta(z, a), x\right)$$

Jede durch Endliche Automaten erkennbare Sprache ist Regulär (Typ 3).
Jede Reguläre Sprache ist durch einen Endlichen Autoamten erkennbar.

Nichtdeterministische Automaten

Ein *nichtdeterministischer, endlicher Automat* (NFA) wird spezifiziert durch ein 5-Tupel:

$M = (Z, \Sigma, \delta, S, E)$	Z Menge der Zustände
Σ Eingabealphabet	$E \subseteq Z$ Menge der Endzustände
$S \subseteq Z$ Menge der Startzustände	$\delta : Z \times E \to \mathcal{P}(z)$ Überführungsfunktion
$z \cap \Sigma = \emptyset$	$\lvert z\rvert < \infty,\ \lvert\Sigma\rvert < \infty$

Jede durch einen NFA akzeptierbare Sprache ist auch durch einen DFA akzeptierbar.

Für jede Reguläre Grammatik G gibt es einen NFA M mit $L(G) = T(M)$.

regulärer Ausdruck

$\emptyset$ ist regulärer Ausdruck.

ε ist regulärer Audruck.

$a \in \Sigma$ ist regulärer Audruck.

$\alpha\beta, (\alpha|\beta), (\alpha)^*$ sind reguläre Ausdrücke, wenn α, β reguläre Ausrdücke sind.

$\gamma := \emptyset$	$\Rightarrow L(\gamma) = \emptyset$	$\gamma := (\alpha\vert\beta)$	$\Rightarrow L(\gamma) = L(\alpha) \cup L(\beta)$
$\gamma := \varepsilon$	$\Rightarrow L(\gamma) = \{\varepsilon\}$	$\gamma := (\alpha)^*$	$\Rightarrow L(\gamma) = L(\gamma)^*$
$\gamma := \alpha$	$\Rightarrow L(\gamma) = \{\alpha\}$	$\gamma := \alpha\alpha = \alpha^2$	
$\gamma := \alpha\beta$	$\Rightarrow L(\gamma) = L\{\alpha\}L\{\beta\}$		

Die Menge der durch Reguläre Ausdrücke beschreibbaren Sprachen ist genau die Menge der Regulären Sprachen.

Pumping-Lemma Schleifenlemma, Iterationslemma, Lemma von Bar-Hillel, $uvw-$Theorem

Sei L eine Reguläre Sprache. Dann gibt es eine Zahl n so, dass sich alle Wörter $x \in L$ mit $|x| \geq n$ zerlegen lasen in $x = uwv$, so dass folgende Eigenschaften erfüllt sind:

- $|v| \geq 1$
- $|uv| \leq n$
- $\forall i \in \{0, 1, \dots\}$ gilt: $uv^iw \in L$

→ Zum Erkennen von nicht regulären Sprachen (geht nicht bei allen).

→ v^i-Schleifen

Anwendung: Annehmen, L sei regulär ⇒Wenn Widerspruch, L nicht regulär.

Äquivalenzrelationen und Minimalautomaten

xR_Ly gdw für alle $z \in \Sigma^* : xz \in L$

Eine Sprache L ist genau dann Regulär, wenn der Index von R_L endlich ist:

$$L \text{ regulär } \Leftrightarrow Index(R_L) < \infty$$

$Index$: Anzahl erzeugbarer Äquivalenzklassen

Äquivalenzklassenautomat ≡ Minimalautomat (nur DFA!)
Minimalautomat → Minimalautomat mit min Zustandszahl

Algorithmus Minimalautomat $\mathcal{O}(n^2)$

Eingabe: DFA, alle Zustände erreichbar. **Ausgabe:** zu verschmelzende Zustände

1. Tabelle mit Zustandspaaren $\{z, z'\}, z \neq z'$
2. Markiere Paare $\{z, z'\}$ mit $z \in E$ und $z' \notin E$ und umgekehrt
3. unmarkierte Felder: Teste, ob $\underbrace{\{\delta(z, a), \delta(z', a)\}}_{\text{1 hop!}}$ markiert ist, wenn ja, markiere $\{z, z'\}$
4. Punkt 3 widerholen, bis keine Änderung mehr in der Tabelle passiert.
5. Verschmelze unmarkierte Paare.

Abschlusseigenschaften

Reguläre Sprachen sind abgeschlossen unter:

- Vereinigung
- Schnitt
- Komplement
- Produkt
- Stern

Entscheidbarkeit

Wortproblem: Linearer Aufwand bei bekanntem DFA

Leerheitsproblem: Entscheidbar (DFA, kein Weg $S \to E$)

Endlichkeitsproblem: $T(M) = \infty \Leftrightarrow S \to \circlearrowright \to E$

Schnittproblem: → Leerheitsproblem

Äquivalenzproblem DFA → Minimalautomat → Isomorphie checken → $\mathcal{O}(n)$

$$L_1 = L_2 \Leftrightarrow \left(L_1 \cap \overline{L_2}\right) \cup \left(L_2 \cap L_1\right) = \emptyset$$

→ Leerheitsproblem → NP-hart

Kontextfreie Sprachen

korrekt geklammerte Ausdrücke

Normalformen

Chromsky-Normalform: (CNF) Alle Regeln haben die Form $A \to BC$ oder $A \to a$

> Zu jeder Kontextfreien Grammatik G mit $\varepsilon \notin L(G)$ gibt es eine Chromsky-Normalform-Grammatik G' mit $L(G) = L(G')$.

Greibach-Normalform: (GNF) Alle Regeln haben die Form $A \to aB_1B_2 \dots B_k$ $(K \geq 0)$

Erweiterung der Regulären Sprachen, hier nur $k = 0$ und $k = 1$

> Zu jeder Kontextfreien Grammatik G mit $\varepsilon \notin L(G)$ gibt es eine Greibach-Normalform-Grammatik G' mit $L(G) = L(G')$.

Pumpinglemma

> Sei L eine kontextfreie Sprache. Dann gibt es eine Zahl $n \in \mathbb{N}$, so adss sich alle Wörter $z \in L$ mit $|z| \geq n$ zerlegen lassen in $z = uvwxy$ mit folgenden Eigenschaften:
>
> - $|vx| \geq 1$
> - $|vwx| \leq n$
> - $\forall i \geq 0$ gilt $uv^iwx^iy \in L$
>
> Jede kontextfreie Sprache über einem einelementigen Alphabet ist bereits regulär.

Abschlusseigenschaften

Die Kontextfreien Sprachen sind...

... abgeschlossen unter	... nicht abgeschlossen unter
• Vereinigung	• Schnitt
• Produkt	• Komplement
• Stern	

Der CYK-Algorithmus

Data: $x = a_1 a_2 \dots a_n$
initialisation;
for $i := 1$ **to** n $_{(j=1)}$ **do**
    $T[i,1] := \{A \in V | A \rightarrow a_i \in P\}$
end
for $j := 2$ **to** n $_{(j>1)}$ **do**
    for $i := 1$ **to** $n+1-j$ **do**
        $T[i,j] := \emptyset$ **for** $k := 1$ **to** $j-1$ **do**
            $T[i,j] := T[i,j] \cup \{A \in V | A \rightarrow BC \in P \wedge B \in T[i,k] \wedge C \in T[i+k, j-k]\}$
        end
    end
end
if $S \in T[1,n]$ **then**
    „x liegt in $L(G)$“
else
    „x liegt nicht in $L(G)$“
end

Algorithm 1: CYK

Kellerautomaten (PDA)

Ein *(nichtdeterministischer) Kellerautomat* (*pushdown automaton*) wird angegeben durch ein 6-Tupel: $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$

- Z - endliche Zustandsmenge
- Σ - Eingabealphabet
- Γ - Kelleralphabet
- $\delta : Z \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow \mathcal{P}_e(Z \times \Gamma^*$ - die Überfühungsfunktion
- $z_0 \in Z$ Startzustand
- $\# \in \Gamma$ unterstes Kellerzeichen

Um Schreibarbeit zu sparen, schreibt man statt $(z', x) \in \delta(z, a, A)$ einfach $zaA \rightarrow z'x$.

Eine Sprache L ist kontextfrei genau dann, wenn L von einem nichtdeterministischen Kellerautomaten erkannt wird.

Deterministisch kontextfreie Sprachen

Ein Kellerautomat M heisst *deterministisch*, falls für alle $z \in Z, a \in \Sigma, A \in \Gamma$ gilt: $|\delta(z,a,A)| + |\delta(z, \varepsilon, A| \leq 1$.
Hinzu kommt, dass deterministisch kontextfreie Kellerautomaten *per Endzustand* akzeptieren, nicht per leerem Keller.

Eine Sprache heisst *deterministisch kontextfrei*, falls sie von einem deterministischen Kellerautomaten erkannt wird.

Abgeschlossenheit

Deterministisch kontextfreie Sprachen sind. . .

. . . abgeschlossen unter:

- Komplement

. . . nicht abgeschlossen unter:

- Schnitt
- Vereinigung

Zudem ist der *Schnitt* einer deterministisch kontextfreien Sprache mit einer regulären Sprache wieder deterministisch kontextfrei.

Entscheidbarkeit

Wortproblem CYK

Leerheitsproblem CNF $\rightarrow$

1. Markiere Variablen, die auf Terminale ableiten $A \rightarrow a$
2. markiere sukzessive alle Variablen, die auf diese führen $A \rightarrow BC$ (B, C bereits markiert)
3. Wenn die Startvariable nicht markiert ist, ist die Sprache leer.

Endlichkeitsproblem Pumping Lemma

Äquivalenzproblem

Kontextsensitive und Typ-0-Sprachen

Kuroda-Normalform

Vergleichbar mit *Chromsky-Normalform*

Eine Typ 1-Grammatik ist in *Kuroda-Normalform*, falls alle Regeln eine der 4 Formen haben:

$$A \rightarrow a, A \rightarrow B, A \rightarrow BC, AB \rightarrow CD$$

hierbei stehen A, B, C, D für Variablen und a für ein Terminalsymbol.

Für jede Typ-1 Grammatik G mit $\varepsilon \notin L(G)$ gibt es eine Grammatik G' in Kuroda-Normalform mit $L(G) = L(G')$.

Turingmaschine

Turingmaschinen sind grundsätzlich nichtdeterministisch.

Eine *Turingmaschine* ist gegeben durch ein 7-Tupel $M = (Z, \Sigma, \Gamma, \delta, z_0, \Box, E)$

- Z endliche Zustandsmenge
- Σ Eingabealphabet
- $\Gamma \supset \Sigma$ Arbeitsalphabet
- $\delta : Z \times \Gamma \to Z \times \Gamma \times \{L, R, N\}$ im deterministischen Fall (bzw. $\delta : Z \times \Gamma \to \mathcal{P}(Z \times \Gamma \times \{L, R, N\})$ im nichtdeterministischen Fall) die Überführungsfunktion
- $z_0 \in Z$ der Startzustand
- $\Box \in \Gamma - \Sigma$ das Blank
- $E \subseteq Z$ Menge der Endzustände

Dabei bedeutet $\delta(z, a) = (z', b, x)$: Wenn sich M im Zustand z befindet und unter dem Schreib-Lesekopf das Zeichen a steht, so geht M im nächsten Schritt in den Zustand z' über, schreibt (auf den Platz von a) b auf das Band und führt danach die Kopfbewegeung $x \in \{L, R, N\}$ aus.

Eine *Konfiguration* einer Turingmaschine ist ein Wort $k \in \Gamma^* Z \Gamma^*$.

Konfiguration = Momentaufnahme $k = \alpha z \beta$ mit z aktueller Zustand.

Definieren auf Menge der Konfigurationen zweistellige Relation $\vdash$ $\vdash$ 1 Schritt
$\vdash^*$ mehrere Schritte

$$a_1 \dots a_m z b_1 \dots b_n \vdash \begin{cases} a_1 \dots a_m z' c b_2 \dots b_n, & \delta(z, b_1) = (z', c, N), m \geq 0, n \geq 1 \\ a_1 \dots a_m c z' b_2 \dots b_n, & \delta(z, b_1) = (z', c, R), m \geq 0, n \geq 2 \\ a_1 \dots a_m z' c b_2 \dots b_n, & \delta(z, b_1) = (z', c, L), m \geq 1, n \geq 1 \end{cases}$$

Die von einer Turingmascheine M *akzeptierte Sprache* ist wie folgt definiert:

$$T(M) = \{x \in \Sigma^* | z_0 x \vdash^* \alpha z \beta; \alpha, \beta \in \Gamma^*; z \in E\}$$

linear beschränkte Turingmaschine (LBA)

Eine nichteterministische Turingmaschine heisst *linear beschränkt*, wenn für alle $a_1 a_2 \dots a_{n-1} a_n \in \Gamma^+$ und alle Konfigurationen $\alpha z \beta$ mit $z_0 a_1 a_2 \dots a_{n-1} \hat{a_n} \vdash^* \alpha z \beta$ gilt $|\alpha\beta| = n$. Die von einer linear beschränkten Turingmaschine m *akzeptierte Sprache* ist wie folgt definiert.

$$T(M) = \{a_1 a_2 \dots a_{n-1} \in \Sigma^* | z_0 a_1 a_2 \dots a_{n-1} \hat{a_n} \vdash^* \alpha z \beta,\ \alpha, \beta \in \Gamma^*;\ z, \in E\}$$

Die von linear beschränkten, nichtdeterministischen Turingmaschine (LBAs) akzeptierten Sprachen sind genau die kontextsensitiven (Typ 1) Sprachen.

Die durch algemeine Turingmaschinen akzeptierbaren Sprachen sind genau die TYP 0-Sprachen.

Eine *Mehrband-Turingmaschine* ist eine Maschine mit $k \geq 1$ Bändern und k Schreib-Leseköpfen. Sie kann daher auf jedem Band unabhängig agieren.

Zu jeder Mehrbandturingmaschine M gibt es eine (Einband-)Turingmaschine M' mit $T(M) = T(M')$, bzw so, dass M' dieselbe Funktion berechnet wie M.

Überblick

Beschreibungsmittel

Typ 3	reguläre Grammatik DFA NFA regulärer Ausdruck
Deterministisch kontextfrei	deterministische Kellerautomaten (DPDA)
Typ 2	kontextfreie Gramatik Kellerautomat PDA
Typ 1	kontextsensitive Grammatik linear beschränkter Automat LBA
Typ 0	Typ 0-Grammatik Turingmaschine TM

Determinismus und Nichtdeterminismus

nichtdet. Automat	determ. Automat	äquivalent?
NFA	DFA	✔
PDA	DPDA	✘
LBA	DLBA	? LBA-Problem
TM	DTM	✔

Abschlusseigenschaften

	Schnitt	Vereinigung	Komplement	Produkt	Stern
Typ 3	✔	✔	✔	✔	✔
Det. kf.	✘	✘	✔	✘	✘
Typ 2	✘	✔	✘	✔	✔
Typ 1	✔	✔	✔	✔	✔
Typ 0	✔	✔	✘	✔	✔

Entscheidbarkeit

	Wort-	Leerheits-	Äquivalenz-	Schnitt-
Typ 3	✔	✔	✔	✔
Det. kf.	✔	✔	✔	✘
Typ 2	✔	✔	✘	✘
Typ 1	✔	✘	✘	✘
Typ 0	✘	✘	✘	✘

Wortproblem (Komplexität)

Typ 3 (DFA GEGEBEN)	lineare Komplexität
Det.kf.	lineare Komplexität
Typ 2 (CNF GEGEBEN)	$\mathcal{O}(n^3)$
Typ 1	exponentielle Komplexität, NP-hart
Typ 0	unlösbar

Berechenbarkeitstheorie

Churchse These
Die durch die formale Definition der *Turing-Berechenbarkeit* (äquivalent: WHILE-*Berechenbarkeit,* GOTO-*Berechenbarkeit,* μ*-Rekursivität*) erfasste Klasse von Funktionen stimmt genau mit der im intuitiven Sinne berechenbaren Funktionen überein.

Turing-Berechenbarkeit

Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ heisst *Turingberechenbar*, falls es eine (deterministische) Turingmaschine M gibt, so dass für alle $n_1, \ldots, n_k, m \in \mathbb{N}$ gilt:

$$f(n_1, \ldots n_k) = m$$

genau dann, wenn

$$z_0 bin(n_1)\#bin(n_2)\# \ldots \#bin(n_k) \vdash^* \square \ldots \square z_e bin(m) \square \ldots \square$$

wobei $z_e \in E$, $bin(n)$ ist die binäre Darstellung der Zahl $n \in \mathbb{N}$.

Eine Funktion $f : \Sigma^* \to \Sigma^*$ heisst *Turingberechenbar*, falls es eine (deterministische) Turingmaschine M gibt, so dass für alle $x, y \in \Sigma^*$ gilt:

$$f(x) = y$$

genau dann wenn

$$z_0 x \vdash^* \square \ldots \square z_e \square \ldots \square$$

wobei $z_e \in E$

Man beachte, dass beide Definitionen die Turingmaschine in eine Endlosschleife übergehen kann, wenn $f(x) = undefiniert$.

LOOP-, WHILE- und GOTO-Berechenbarkeit

LOOP-Programme

Variablen: $x_0\ x_1\ x_2\ \ldots$

Konstanten: $0\ 1\ 2\ \ldots$

Trennsymbole: ; :=

Operationszeichen $+$ $-$

Schlüsselwörter: LOOP, DO, END

Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ heisst LOOP-berechenbar, falls es ein LOOP-Programm P gibt, dass f in dem Sinne berechnet, dass P, gestartet mit $n_1, \ldots, n_k$ in den Variablen $x_1, \ldots, x_k$ (und 0 in den restlichen Variablen) stoppt mit dem Wert $f(n_1, \ldots, n_k)$ in der Variablen x_0.

WHILE-Programme

Wir erweitern LOOP-Programme durch das Konzept der WHILE-Schleife:

$$WHILE x_i \neq 0 DO P END$$

Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ heisst WHILE-berechenbar, falls es ein WHILE-Programm gibt P gibt, das f in dem Sinne berechnet, dass P, gestartet mit $n_1, \ldots, n_k$ in den Variablen $x_1, \ldots, x_k$ (und 0 in den restlichen Fällen) stoppt mit dem Wert $f(n_1, \ldots, n_k)$ in der Variablen x_0, sofern $f(n_1, \ldots, n_k)$ definiert ist, ansonsten stoppt P nicht.

Turingmaschinen können WHILE-Programme simulieren. Dass heisst, jede WHILE-berechenbare Funktion ist auch Turing-berechenbar.

GOTO-Programme

GOTO-Programme bestehen aus Sequenzen von Anweisungen A_i, die jeweils durch eine Marke M_i eingeleitet werden:

$$M_1 : A_1;\ M_2 : A_2;\ \ldots;\ M_k : A_k$$

Als mögliche Anweisungen A_i sind zugelassen:

Wertzuweisungen: $x_i := x_j \pm c$

unbedingter Sprung: GOTO M_i

bedingter Sprung: IF $x_i = c$ THEN GOTO M_j

Stopanweisung HALT

Jedes WHILE-Programm kann durch ein GOTO-Programm simuliert werden. Das heisst, jede WHILE-berechenbare Funktion ist auch GOTO-berechenbar.

Jedes GOTO-Programm kann durch ein WHILE-Programm (mit nur einer WHILE-Schleife) simuliert werden. Also ist jede GOTO-berechenbare Funktion auch WHILE-berechenbar

Kleensche Normalform für WHILE-Programme
Jede WHILE-berechenbare Funktion kann durch ein WHILE-Programm mit nur einer WHILE-Schleife berechnet werden.

GOTO-Programme können Turingmaschinen simulieren. Also ist jede Turingberechenbare Funktion auch GOTO-berechenbar.

$$\begin{tikzcd}
\textsc{Goto} \arrow[r, leftrightarrow] \arrow[rr, leftrightarrow, bend left] & \textsc{While} \arrow[r, leftrightarrow] & \textsc{Tm} \\
& \textsc{Loop} \arrow[u] &
\end{tikzcd}$$

Primitiv und μ-rekursive Funktionen

Prüfungsrelevant?

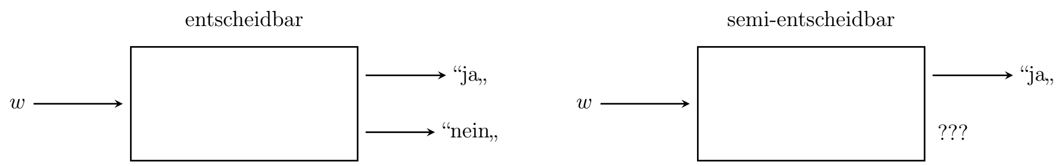
Halteproblem, Unentscheidbarkeit, Reduzierbarkeit

Eine Menge $A \subseteq \Sigma^*$ heisst *entscheidbar*, falls die *charakteristische Funktion* von A, nämlich $\chi_A : \Sigma^* \to \{0, 1\}$, berechenbar ist. Hierbei gilt für alle $w \in \Sigma^*$:

$$\chi_A(w) = \begin{cases} 1, & w \in A \\ 0, & w \notin A \end{cases}$$

Eine Menge $A \subseteq \Sigma^*$ heisst *semi-entscheidbar*, falls die *„halbe“ charakteristische Funktion* von A, nämlich $\chi_A : \Sigma^* \to \{0, 1\}$, berechenbar ist. Hierbei gilt für alle $w \in \Sigma^*$:

$$\chi_A(w) = \begin{cases} 1, & w \in A \\ undefiniert, & w \notin A \end{cases}$$



Eine Sprache A ist genau dann entscheidbar, wenn sowohl A, als auch $\overline{A}$ semi-entscheidbar sind.

Eine Sprache ist genau dann semi-entscheidbar, wenn sie rekursiv aufzählbar ist.

Äquivalente Aussagen:

- A ist rekursiv aufzählbar.
- A ist semi-entscheidbar.
- A ist vom Typ 0.
- $A = T(M)$ für Turingmaschine M

- χ'_A ist (Turing-, WHILE-, GOTO-) berechenbar.
- A ist Definitionsbereich einer berechenbaren Funktion.
- A ist Wertebereich einer berechenbaren Funktion.

Halteproblem

Turingmaschinen sind als Wort schreibbar. Dazu nummerieren wir Elemente von Γ und Z durch. Hierbei sei festgelegt, welche Nummern die Symbole $\Box, 0, 1, \#$ sowie Start- und Endzustände erhalten.

$$\Gamma = \{a_0, a_1, \ldots, a_k\}$$
$$Z = \{z_0, z_1, \ldots, z_k\}$$

Jeder δ-Regel ordnen wir ein Wort zu

$$\delta(z_i, a_j) = (z_{i'}, a_{j'}, y) \Rightarrow w_{i,j,i',j',y} = \#\#bin(i)\#bin(j)\#bin(i')\#bin(j')\#bin(m)$$

wobei

$$m = \begin{cases} 0, & y = R \\ 1, & y = L \\ 2, & y = N \end{cases}$$

Somit erhalten wir einen Code für die Turingmaschine über dem Alphabet $\{0, 1, \#\}$. Sei $\hat{M}$ eine beliebige, feste Turingmaschine. Dann können wir für jedes Wort $w \in \{0,1\}^*$ festlegen, dass M_w eine Turingmaschine bezeichnet:

$$M_w = \begin{cases} M, & \text{falls } w \text{ Codewort von } M \text{ ist} \\ \hat{M}, & \text{sonst} \end{cases}$$

Unter dem *speziellen Halteproblem* oder *Selbstanwendbarkeitsproblem* verstehen wir die Sprache

$$K = \{w \in \{0,1\}^* | M_w \text{ angesetzt auf } w \text{ hält (im Endzustand)}\}$$

Das spezielle Halteproblem ist nicht entscheidbar.

Reduktion

Seien $A \subseteq \Sigma^*$ und $B \subseteq \Gamma^*$ Sprachen. Dann heisst A *auf* B *reduzierbar* - symbolisch mit $A \leq B$ bezeichnet - falls es eine totale und berechenbare Funktion $f : \Sigma^* \to \Gamma^*$ gibt, so dass für alle $x \in \Sigma^*$ gilt:

$$x \in A \Longleftrightarrow f(x) \in B$$

Falls $A \leq B$ und B entscheidbar (bzw. semi-entscheidbar), dann ist auch A entscheidbar (bzw-semi-entscheidbar).

Das (allgemeine) Halteproblem ist die Sprache

$$H = \{w\#x | M_w \text{ angesetzt auf } w \text{ hält}\}$$

Das allgemeine Halteproblem H ist nicht entscheidbar.
$K \leq H$, da K Spezialfall von H

Das Halteproblem auf leerem Band ist die Sprache

$$H = \{w\#x | M_w \text{ angesetzt auf } w \text{ hält}\}$$

Das Halteproblem auf leerem Band H_0 ist nicht entscheidbar.
$H \leq H_0$, da Turingmaschine einfach das nichtleere Band von H auf das leere Band von H_0 schreiben kann.

Satz von Rice
Sei $\mathcal{R}$ die Klasse aller Turing-berechenbaren Funktionen. Sei $\mathcal{S}$ eine *beliebige* Teilmenge hiervon (ausgenommen $\mathcal{S} = \emptyset$ und $\mathcal{S} = \mathcal{R}$). Dann ist die Sprache

$$C(\mathcal{S} = \{w | \text{ die von } M_w \text{ berechnete Funktion liegt in } \mathcal{S}\}$$

unentscheidbar.

Das Postsche Korrespondenzproblem (PCP)

gegeben: Eine endliche Folge von Wortpaaren $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$, wobei $x_i, y_i \in \Sigma^+$.

gefragt: Gibt es eine Folge von Indizes $i_1, i_2, \ldots, i_n \in \{1, 2, \ldots, k\}, n \geq 1$, mit $x_{i_1}, x_{i_2}, \ldots, x_{i_n} = y_{i_1}, y_{i_2}, \ldots, y_{i_n}$?

Das PCP ist semi-entscheidbar Immer längere Indexfolgen daraufhin untersuchen, ob sie Lösung sind..

modifiziertes Postsches Korrespondenzproblem (MPCP)

gegeben: wie PCP

gefragt: Gibt es eine Lösung $i_1, i_2, \ldots, i_n$ mit $i_1 = 1$

MPCP $\leq$ PCP und $H \leq$ MPCP

Das Postsche Korrespondenzproblem ist untentscheidbar.

Das Postsche Korrespondenzproblem ist bereits untentscheidbar, wenn man sich auf das Eingabealphabet $\{0, 1\}$ beschränkt.

Unentscheidbare Grammatik-Probleme

nicht Prüfungsrelevant

Der Gödelsche Satz

nicht Prüfungsrelevant

Komplexitätstheorie

Komplexitätsklasse und P-NP-Problem

Sei $f : \mathbb{N} \to \mathbb{N}$ eine Funktion. Die Klasse $TIME(f(n))$ besteht aus allen Sprachen A, für die es eine deterministische Mehrbandturingmaschine M gibt mit $A = T(M)$ und $time_M(x) \leq f(|x|)$. Hierbei bedeutet $time_M : \Sigma^* \to \mathbb{N}$ die Anzahl der Rechenschritte von M bei Eingabe x.

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

$$p(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0, a_i \in \mathbb{N}, k \in \mathbb{N}$$

Die *Komplexitätsklasse* P ist wie folgt definiert:

$$\begin{aligned} \mathsf{P} &= \{A| \text{ es gibt eine Turingmaschine } M \text{ und ein Polynom } p \text{ mit } T(M) = A \text{ und } time_M(x) \leq p(|x|)\} \\ &= \bigcup_{p \text{ Polynom}} TIME(p(n)) \end{aligned}$$

Die Klasse P (ebenso wie größere Komplexitätsklassen wie $TIME(2^n)$ oder $TIME\left(2^{2^{\cdot^{\cdot^{\cdot^{2^2}}}}}\Big\}n\text{-mal}\right)$) sind immer noch in der Klasse der primitiv rekursiv bzw LOOP-berechenbaren Sprachen enthalten

Für nichtdeterministische Turingmaschinen M sei

$$ntime_M(x) = \begin{cases} min \text{ Länge einer akzeptierenden Rechnung von } M \text{ auf } x, & x \in T(M) \\ 0, & x \notin T(M) \end{cases}$$

Sei $f : \mathbb{N} \to \mathbb{N}$ eine Funktion. Die Klasse $NTIME(f(n))$ besteht aus allen Sprachen A, für die es eine *nichtdeterministische* Mehrband-Turingmaschine M gibt mit $A = T(M)$ und $ntime_M(x) \leq f(|x|)$.
Weiter definieren wir

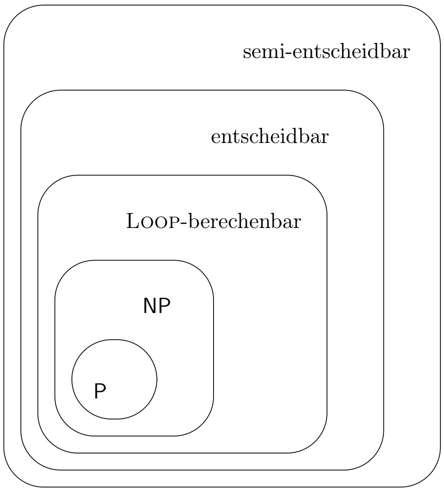
$$\mathsf{NP} = \bigcup_{p \text{ Polynom}} NTIME(p(n))$$

P-NP-Problem

bekannt: $\mathsf{P} \subseteq \mathsf{NP}$

erhofft: $\mathsf{P} = \mathsf{NP}$

vermutet: $\mathsf{P} \neq \mathsf{NP}$



NP-Vollständigkeit

Seien $A \subseteq \Sigma^*$ und $B \subseteq \Gamma^*$ Sprachen. Dann heisst A *auf B polynomial reduzierbar* - symbolisch mit $A \leq_p B$ bezeichnet - falls es eine totale und mit polynomieller Komplexität berechenbare Funktion $f : \Sigma^* \to \Gamma^*$ gibt, so dass für alle $x \in \Sigma^*$ gilt:

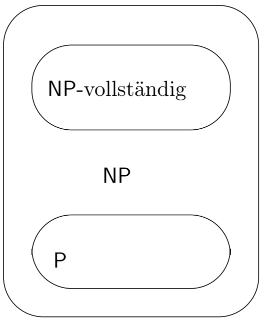
$$x \in A \Longleftrightarrow f(x) \in B$$

Falls $A \leq_p B$ und $B \in \mathsf{P}$ (bzw. $B \in \mathsf{NP}$), so ist auch $A \in \mathsf{P}$ (bzw. $A \in \mathsf{NP}$)

Eine Sprache A heisst *NP-hart*, falls für alle Sprachen $L \in \mathsf{NP}$ gilt: $L \leq_p A$.

Eine Sprache A heisst *NP-vollständig*, falls A NP-hart ist und $A \in \mathsf{NP}$ gilt.

Sei A NP vollständig. Dann gilt $A \in P \Leftrightarrow \mathsf{P} = \mathsf{NP}$. falls es einnmal so wäre...



NP-vollständige Probleme

Das *Erfüllbarkeitsproblem der Aussagenlogik*, kurz SAT, ist das Folgende:

gegeben: eine Formel F der Aussagenlogik

gefragt: Ist F erfüllbar, d.h. gibt es eine Belegung der Variablen mit Konstanten $\in \{0, 1\}$, so dass F den Wert 1 erhält?

$$\textsc{Sat} = \{code(F) \in \Sigma^* | F \text{ ist eine erfüllbare Formel der Aussagenlogik}\}$$

$$\Sigma = \{(,), \neg, \wedge, \vee, \times, 0, 1\}$$

Das Erfüllbarkeitsproblem derAussagenlogik, Sat , ist NP-vollständig.