Grundlagen der Informatik Zusammenfassung

Propositional Logic

- $A \wedge B$ ($\wedge$ → math. und)
- $A \vee B$ ($\vee$ → math. oder / 1 or both)
- $\neg A$ ($\neg$ → math. Negation)
- $A \Leftrightarrow B$ (A is True, if B is True)
- $A \Rightarrow B$ (: if A is True, B is True; if A is False, B can be true)

Mengen

Leere Menge:

$\emptyset$, $\{\}$

Zahlenmengen

- $\mathbb{N}$ → positiv Integers
- $\mathbb{Z}$ → positiv & negative Integers
- $\mathbb{Q}$ → floats
- $\mathbb{R}$ → reell

$A \subseteq B$ (A ist Teil von B)

$A \cup B$ (A oder B → Vereinigung)

$A \cap B$ (A und B → Gemeinsames)

$A \setminus B$ (A ohne B → Schnitt)

Beweise

direkter Beweis:

Beginnt mit Hypothese → Logische Schritte führen zur Schlussfolgerung

konstruktiver Beweis:

wir konstruieren eine Aussage, welche sich von der zu Beweisenden Aussage ableiten lässt und zeigen mit direktem Beweis, dass diese zutrifft.

Kontraposition / indirekter Beweis:

wir Beweisen, dass das Gegenteil stimmt. → Start-Aussage ist falsch

Beweis durch Widerspruch:

wir probieren das Gegenteil zu beweisen, was jedoch zu einem Widerspruch führt → also stimmt die Start-Aussage.

Beweis durch Induktion:

Induktionsanfang: wir zeigen, dass die Aussage für eine gegebene Zahl stimmt (meistens 1)

Induktionsschritt: $n \rightarrow (n+1)$

wir zeigen, dass die Aussage nun auch für $n+1$ gilt.

Setze auf der einen Seite der Rechnung $(n+1)$ für n ein und auf der anderen Seite z.B addiere $(n+1)$ zum ganzen Term

Tupel

5 - Tupel:

Q - Zustände des Automaten

Σ - alphabet des Automaten (inputs)

δ - $Q \times \Sigma \to Q$ Funktionen (pro Zustand für alle Inputs)

q_0 - Startzustand

F - Akzeptanzzustände

6 - Tupel:

Q, Σ - Zustände, Alphabet

Γ - Kelleralphabet

δ - $Q \times \Sigma \times \Gamma \to P(Q \times \Gamma)$ Übergangsfunktion

S, F - Startzustand, Endzustände

7 - Tupel:

Q - Zustände der TM

Σ - Inputalphabet ($\sqcup \notin \Sigma$)

Γ - Bandalphabet, s.d. $\Sigma \subseteq \Gamma$ und $\sqcup \in \Gamma$

δ - $Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ Übergangsfunktion
links rechts

S - Startzustand

q_{accept} - Akzeptanzzustand

q_{reject} - Verwerfzustand

Pumping Lemma - reguläre Sprachen

Falls eine Sprache (A) regulär ist, kann man Strings aus dieser Sprache "aufpumpen"

Es gibt also eine Pump-Zahl P sodass jeder String s ($|s| \geq p$) in 3-Teile geteilt werden kann.

Für die 3-Teile gelten folgende Regeln:

$s = xyz$

i) $\forall i \geq 0 \quad xy^i z \in A$

ii) $|y| > 0$

iii) $|xy| \leq p$

Beweis durch Widerspruch

Nimm den kleinst möglichen String der Sprache und setze die Exponenten mit p

Zerlege nun diesen String in 3-Teile sodass die Länge des 1. & 2. - Teiles kleiner gleich p sind. und die Länge des 2. grösser ist wie 0.

Mache nun ein Beispiel z.B $i = 2$ und zeige, dass der resultierende String die Regeln nun nicht mehr erfüllt.

Pumping Lemma kontextfreie Sprachen

Falls eine Sprache A kontextfrei ist, kann man Strings aus dieser Sprache "aufpumpen"

Es gibt also eine Pump-Zahl P sodass jeder String s ($|s| \geq p$) in 5-Teile geteilt werden kann.

Für die 5-Teile gelten folgende Regeln:

$s = uvxyz$

i) $\forall i \geq 0 \quad uv^ixy^iz \in A$

ii) $|vx| > 0$

iii) $|vxy| \leq p$

$\Rightarrow$ ablauf gleich wie für reguläre Sprachen.

Chompsky Normalform

Alle Regeln der kontextfreien Grammatik haben die Form:

- $A \rightarrow BC$
- $A \rightarrow a$
- $S \rightarrow \varepsilon$

Konvertierungs-Schritte:

1. erstelle neue Regel $S_0 \rightarrow S$
2. lösche alle Regeln der Form $A \rightarrow \varepsilon$
   und ersetze alle Vorkommnisse von A (auf der Rechten Seite) mit ε als zusätzliche Option.
3. lösche alle Regeln der Form $A \rightarrow B$ und erstelle neue Regeln $A \rightarrow u$, falls $B \rightarrow u$ bereits existiert.
4. erstelle neue Regeln, sodass alle anderen Regeln nur noch die erlaubten Formen hat.

Ist eine Grammatik in Chompsky Normalform, dann kann ein String w in genau $2|w|-1$ Schritten erzeugt werden, solange $|w| \neq 0$.

Kellerautomat

⇒ Automat mit Stack

Notation für Übergänge:

| a, b → c |

Bei Input a wird b im Stack durch c ersetzt, wenn b zuobest ist.

wenn b = ε wird c zum Stack hinzugefügt

wenn c = ε wird b aus dem Stack gelöscht

Turing - Maschiene

⇒ Automat mit ∞ Speicher und verschiebbaen lese/schreib - kopf

Speicher ist ein Band, das nur nach rechts ∞ ist.

Notation für Übergänge:

| a → b, {R, L} |

Bei Input a wird b auf das Tape geschrieben und der Lesekopf geht 1 nach rechts/links.

wenn b = ␣ wird der Wert nicht geändert

Empty string bei TM ist ␣

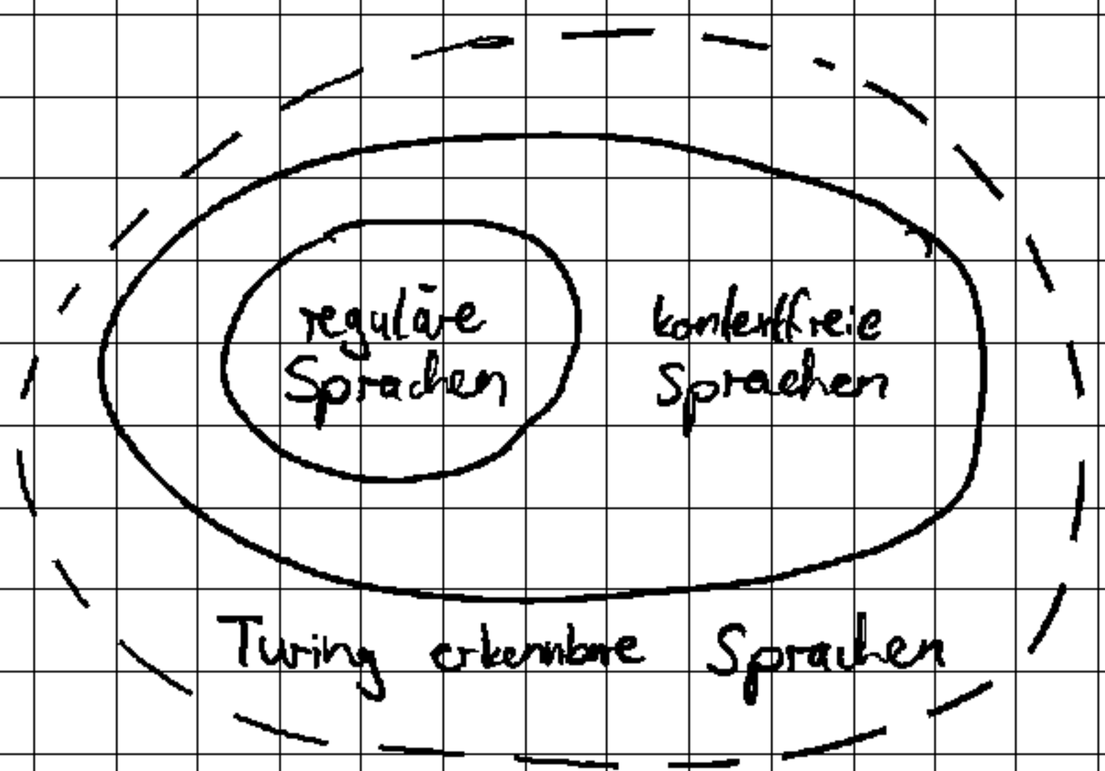
0, 1 → {R, L} — Bei Input 1 oder 0 gehe nach R/L schreibe nichts

Sprachen

- reguläre Sprachen $\equiv$ endliche Automaten (deterministisch / nicht deterministisch)
- kontextfreie Sprachen $\equiv$ Kellerautomaten (nicht-deterministisch)
- Turing erkennbare Sprachen $\equiv$ Turingmaschienen (det. / nicht-det)

Entscheidungsprobleme - {Ja / Nein}, {0 / 1}, {wahr / falsch}

↳ formale Sprachen erkennen - Ja / Nein (String enthalten)



Für eine Reguläre Sprache existiert auch ein regulärer Ausdruck

Turing - entscheidbar vs -erkennbar

entscheidbar:

Es existiert eine TM, die jeden String der Sprache akzeptiert und jeden anderen verwirft (läuft nie ∞ weiter)

erkennbar:

Es existiert eine TM, die jeden String der Sprache akzeptiert.

Beweis durch Diagonalisierung

wir wollen zeigen, dass eine Menge A nicht - abzählbar ist.

Also vergleichen wir die Menge A mit der Menge $\mathbb{N}$

Das heißt, wir maden eine numerierte liste von Strings aus der Menge A die alle unterschiedlich sind.

Diese Liste könnte nun ∞ weitergeführt werden.

Jedoch können wir nun immer einen String finden, die nirgends auf dieser unendlichen Liste ist, indem wir die Diagonale der Liste nehmen und jedes korrespondierende element ändern. $\Rightarrow$ wir bekommen einen String der sich von dem 1., 2., 3. ... String unterscheidet.

Also ein Element, welches nicht auf unserer ∞-Liste ist.

$\Rightarrow$ Dementsprechend ist A nicht abzählbar.

1	1	0	0	0	0	0	0	...
2	1	1	0	0	0	0	0	
3	1	1	1	0	0	0	0	
4	1	1	1	1	0	0	0	
5	1	1	1	1	1	0	0	
	⋮				⋱	⋱		

$x = \underbrace{0\,0\,0\,0\,0\,0\,\ldots}$

nicht auf der Liste.

Zeitkomplexität

Laufzeiten:

$O(1)$ - konstante Laufzeit
$O(\log n)$ - logarithmische Laufzeit
$O(n)$ - lineare Laufzeit
$O(n^2)$ - quadratische Laufzeit
$O(n^k)$ - polynomielle Laufzeit (wenn $k \in \mathbb{N}$)
$O(a^n)$ - exponentielle Laufzeit

Berechnen von Big-O

wie schauen wir, ob eine Laufzeit Teil einer Anderen ist:

$$\underbrace{n^2 \cdot \log(n)}_{f(n)} \subseteq \underbrace{O(n^2)}_{O(g(n))}$$

wenn $\lim_{n \to \infty} \frac{f(n)}{g(n)} \neq \infty$, dann ist $f(n)$ Teil von $g(n)$

Berechnen von Little-O

$$\underbrace{n}_{f(n)} = \underbrace{o(2n)}_{o(g(n))}$$

wenn $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$, dann ist die Aussage richtig.

Laufzeiten von verschachtelten abhängigen loops:

loops können als $\sum$ geschrieben werden diese $\sum$ kann umgeformt werden zu $\sum q^k = \frac{1}{1-q}$ (geometrische Reihe)

P vs NP

P - polynomielle Zeit

Falls Sprache $A \in P$

$\Rightarrow$ es existiert eine TM, die A in max polynomieller Zeit entscheiden kann.

NP - Nichtdeterministische polynomielle Zeit

Falls Sprache $A \in NP$

$\Rightarrow$ es existiert keine TM, die A schneller als mit exponentieller Zeit entscheiden kann.

Negation

$\neg (A \Rightarrow B) = A \wedge \neg B$

$\neg (\forall x) = \exists x$

$\neg (\exists x) = \forall x$

$\neg (=) = \neq$

$\neg (A \wedge B) = \neg A \vee \neg B$

$\neg (A \leq B) = A > B$

Regel von 'hopital

$$\lim_{x \to c} \frac{f(x)}{g(x)} = \lim_{x \to c} \frac{f'(x)}{g'(x)}$$

wird angewendet wenn $f(x)$ und $g(x)$ gegen 0 oder ∞ gehen.

z.B $\dfrac{\overbrace{\log_e(n^2)}^{\ln(n^2)}}{\log_2(n)} = \dfrac{\frac{2}{n}}{\frac{1}{\ln(2) \cdot n}} = \dfrac{2}{n} \cdot \dfrac{\ln(2) \cdot n}{1} = \dfrac{2n \ln(2)}{n}$

$$\boxed{(\sqrt{n^3})' = \frac{3\sqrt{n}}{2} \quad | \quad (\sqrt{n})' = \frac{1\sqrt{n}}{2} \quad | \quad \frac{\sqrt{x}}{x} = \frac{x^{\frac{1}{2}}}{x^1} = \frac{1}{\sqrt{x}}}$$

Bsp. verschachtelte loops

for $i = 1 \dots m$

for $j = 1 \dots \frac{n}{2^i}$

$\rightarrow$ Anzahl loops $= \sum_{i=1}^{m} \frac{n}{2^i}$

$$\Rightarrow \lim_{m \to \infty} \sum_{i=1}^{m} \frac{n}{2^i} = \lim_{m \to \infty} n \sum_{i=1}^{m} \frac{1}{2^i} = \lim_{m \to \infty} n \boxed{\sum_{i=1}^{m} \left(\frac{1}{2}\right)^i}$$

$$= \frac{1}{1 - \frac{1}{2}}$$

$$= n \cdot \frac{1}{\frac{1}{2}} = n \cdot \frac{1}{1} \cdot \frac{2}{1} = 2n$$