Karlsruher Institut für Technologie

Grundbegriffe der Informatik

Inoffizielle Zusammenfassung der Vorlesung im WS24/25 von Torsten Ueckerdt

Inhaltsverzeichnis

1 Mengen, Alphabete, Abbildungen

1.1 Mengen

- "Behälter"mit Öbjekten"
- Menge kann Objekt enthalten oder nicht
- Beispiel: Menge mit Zahlen $1, 2, 3 : M = \{1, 2, 3\} = \{3, 2, 1\} = \{1, 1, 3, 2, 3, 3, 3, 2, 1, 2\}$

Kardinalität

- Anzahl der Elemente in einer Menge (Schreibe: $|A|$ oder $\#A$)
- $|A| \in \mathbb{N}_0 \cup \{\infty\}$

1.1.1 Notation

Pünktchen

- ohne explizite Definition ($\mathbb{N}_+ = \{1, 2, 3, 4, \cdots\}$ und $\mathbb{N}_0 = \{0, 1, 2, 3, \cdots\}$
- Achtung: Gefahr von Missverständnissen

Set Comprehension

- Sei $P(x)$ eine Aussage, welche für jedes Objekt x wahr oder falsch ist
- $\{x \in M | P(x)\}$ enthält genau die $x \in M$, für die $P(x)$ wahr ist
- alternativ geht auch $\{x | P(x)\}$
- nur harmlose Aussagen erlaubt, nicht bspw. $A = \{x : x \notin A\}$

1.1.2 Teilmengen

- es seien A und B zwei Mengen
- A Teilmenge von B und B Obermenge von A
  - $A \subseteq B$ oder $B \subseteq B$
- $A = B$ wenn $A \subseteq B \land B \subseteq A$
- A **echte Teilmenge** von B, wenn $A \subseteq B$, aber $A \neq B$ (schreibe $A \subset B$)

1.1.3 Mengenoperationen

- **Vereinigung** $A \cup B = \{x | x \in A \lor x \in B\}$
- **Durchschnitt** $A \cap B = \{x | x \in A \lor x \in B\}$
  - Eine Menge A ist ***disjunkt*** zu einer Menge B wenn gilt $A \cap B = \emptyset$
- **Mengendifferenz** $A \setminus B = \{x \in A | x \notin B\}$
- **Karthesisches Produkt** $A \times B = \{(a, b) | a \in A \land b \in B\}$
  - $M^2 = M \times M, M^3 = M \times M \times M$
  - Paare $(x, y) \neq (y, x)$
- **Potenzmenge** $2^M = \mathfrak{B}(M) = \mathcal{P}(M) = \{A | A \subseteq M\}$
- **Indikatorfunktion**
$$\chi_A = \begin{cases} 1, \text{falls } x \in A \\ 0, \text{falls } x \notin A \end{cases}$$
- **"Große"Vereinigung** $\cup_{i \in I} M_i = \{x | \exists i \in I : x \in M_i\}$
- **"Großer"Durchschnitt** $\cap_{i \in I} M_i = \{x | \forall i \in I : x \in M_i\}$

1.1.4 Mengengesetze

Seien A, B, C Mengen. Dann gilt

- $A \cup A = A$ und $A \cap A = A$ (**Idempotenzgesetz**)
- $A \cup B = B \cup A$ und $A \cap B = B \cap A$ (**Kommutativgesetz**)
- $(A \cup B) \cup C = A \cup (B \cup C)$ und $(A \cap B) \cap C = A \cap (B \cap C)$ (**Assoziativgesetz**)
- $(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$ und $(A \cap B) \cup C = (A \cup C) \cap (B \cup C)$ (**Distributivgesetz**)

1.2 Alphabete

Ein Alphabet ist eine nichtleere endliche Menge von Zeichen oder Symbolen. Zeichen sind elementare Bausteine für Inschriften.

Beispiele:

- $A = \{|\}$, $A = \{0, 1\}$, $\cdots$
- $A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$
- ASCII
- Unicode $\approx$ 100.000 Zeichen

1.3 Relationen und Abbildungen

1.3.1 Relationen

- sind Paare in Beziehung stehender Elemente
- Unicode: Angabe aller Paare (a, n), für die $n \in \mathbb{N}_0$ der Code Point von $a \in A_U$ ist (bspw. $(A, 65)$ $(\alpha, 945)$, ...)
- Relation R: Teilmenge $R \subseteq A \times B$
  - *binäre Relation* von A und B
  - $(a, b) \in R$ (gelesen: "*a steht in Relation R zu b*")
  - Schreibe auch aRb statt $(a, b) \in R$

1.3.2 Eigenschaften von Relationen

Sei $R \subseteq A \times B$ eine Relation.

- $\forall a \in A \exists b \in B : (a, b) \in R$ (R ist **linkstotal**)

  Sprich: "*Jedes Element aus A hat mind. einen Partner in B*"
- $\forall a_1, a_2 \in A, b \in B : [(a_1, b) \in R \land (a_2, b) \in R] \implies a_1 = a_2$ (R ist **linkseindeutig**)

  Sprich: "*Jedes Element aus B hat höchstens einen Partner in A*"
- $\forall b \in B \exists a \in A : (a, b) \in R$ (R ist **rechtstotal**)

  Sprich: "*Jedes Element aus B hat mind. einen Partner in A*"

- $\forall a \in A\ b_1, b_2 \in B : [(a, b_1) \in R \wedge (a, b_2) \in R] \implies b_1 = b_2$ (R ist **rechtseindeutig**)
  Sprich: "*Jedes Element aus A hat höchstens einen Partner in B*"

1.3.3 Abbildungen

- sind spezielle Relationen die linkstotal und rechtseindeutig sind
- zu ihnen gehören
  - der **Definitionsbereich** A und
  - der **Zielbereich** B
  - die **Abbildungsvorschrift**
  - analog für Relationen
- Schreibweise:
  - $R : A \to B$
  - $(a, b) \in R$ schreibt man auch als $R(a) = b$
  - b heißt der Funktionswert an der Stelle a
- nicht linkstotale, aber rechtstotale Relationen heiße partielle Funktionen
- eine Abbildung, die linkseindeutig ist, heißt **injektiv**
- eine Abbildung, die rechtstotal ist, heißt **surjektiv**
- eine Abbildung heißt **bijektiv**, wenn sie sowohl injektiv als auch surjektiv ist

Beispiele

- $f : \mathbb{N}_0 \to \mathbb{N}_0, n \mapsto \begin{cases} \frac{n}{2}, & \text{falls } n \text{ gerade} \\ 3n + 1, & \text{falls } n \text{ ungerade} \end{cases}$
- seien A, B Mengen: $B^A := \{f | f \text{ ist Funktion } f : A \to B\}$

2 Wörter

2.1 Wörter

Ein Wort ist eine Liste, eine Aneinanderreihung an Zeichen, d.h. eine Zeichenkette. Es existiert eine klare Reihenfolge der Zeichen.

Ein Wort über dem Alphabet A ist eine Abbildung

$$w : [n] \to A \text{ mit } [n] := \{i \in \mathbb{N}_0 | 1 \leq i \wedge i \leq n\}$$

- Länge eines Wortes: $|w|$
- Menge der Wörter der Länge n über Alphabet A: A^n
- Menge aller Wörter über Alphabet A:

$$\begin{aligned} A^* &= A^0 \cup A^1 \cup A^2 \cup \cdots \\ &= \bigcup_{i \in \mathbb{N}_0} A^i = \{w | \exists i \in \mathbb{N}_0 : w \in A^i\} \end{aligned}$$

  Formalistisch ist A^* die Menge aller Abbildungen $w : [n] \to A$ mit $n \in \mathbb{N}_0$

Beispiel: $w = hallo$, formal $w : [5] \to \{a, h, l, o\}$ mit

$$w(1) = h, w(2) = a, w(3) = l, w(4) = l, w(5) = o$$

2.2 Das leere Wort

- besteht aus 0 Symbolen, schreibe ε
- formal ist es eine Abbildung $\varepsilon : [0] \to A$ also $\varepsilon : \emptyset \to A$

  $\Longrightarrow$ Relation ε ist linkstotal und rechtseindeutig
- ist das neutrale Element von Wörtern bzgl. Konkatenation (2.3)

  Für jedes Alphabet A gilt $\forall w \in A^* : w \cdot \varepsilon = w = \varepsilon \cdot w$

2.3 Konkatenation

2.3.1 Konkatenation von Wörtern

Anschaulich werden hier Wörter hintereinander geschrieben.

- (Definition) Seien $w_1 : [m] \to A_1$ und $w_2 : [n] \to A_2$ zwei Wörter. Dann ist die Konkatenation definiert als

$$w_1 \cdot w_2 : [m+n] \to A_1 \cup A_2$$
$$i \mapsto \begin{cases} w_1(i), & \text{falls } 1 \leq i \leq m \\ w_2(i-m), & \text{falls } m < i \leq m+n \end{cases}$$

- nicht kommutativ, da

$$BAUM \cdot STAMM = BAUMSTAMM \neq STAMMBAUM = STAMM \cdot BAUM$$

- aber assoziativ, es gilt: $(w_1 \cdot w_2) \cdot w_3 = w_1 \cdot (w_2 \cdot w_3)$ (für jedes Alphabet A und alle $w_1, w_2, w_3 \in A^*$
- für jedes Alphabet A, jedes $w \in A^*$ und jedes $n \in \mathbb{N}_0$ gilt: $|w^n| = n \cdot |w|$ (**Länge von Wortpotenzen**)

2.3.2 Iterierte Konkatenation

- bei Potenzen von Wörtern wird Potenzschreibweise verwendet (bspw. $w^3 = w \cdot w \cdot w$)
- eine Definition wäre $w^n = \underbrace{w \cdot w \cdot \ldots \cdot w}_{n\text{-mal}}$ (schlecht, da Pünktchen verwendet werden!)
- deswegen induktive Definition

$$w^0 = \varepsilon$$
$$\forall n \in \mathbb{N}_+ : w^n = w^{n-1} \cdot w$$

3 Aussagenlogik

3.1 Aussagen

- korrekte Aussagen sind nur wohldefiniert, Kausalität irrelevant!
- (**Zweiwertigkeit**) Jede Aussage entweder wahr oder falsch
- (**Extensionalität**) Wahrheitswert zusammengesetzter Aussagen durch Wahrheitswerte der Teilaussagen eindeutig festgelegt

Operationen

- $\neg P$: "Nicht P"(**Negation**)
- $P \wedge Q$: "P und Q"(**Konjunktion**, logisches Und)
- $P \vee Q$: "P oder Q"(**Disjunktion**, logisches Oder)
- $P \rightarrow Q$: "P impliziert Q" $\iff$ "Wenn P, dann Q"(**Implikation**, logische Folgerung)

3.2 Alphabet der Aussagenlogik

- Aussagevariablen sind $P_0, P_1, P_2, \ldots$

  $\text{Var}_{AL} \subset \{P_i | i \in \mathbb{N}_0\}$
- kurz P, Q, R, S

3.2.1 Aussagenlogische Konnektive

- $\neg$ bindet am stärksten
- $\wedge$ bindet am zweitstärksten
- $\vee$ bindet am drittstärksten
- $\rightarrow$ bindet am viertstärksten
- $\leftrightarrow$ bindet am schwächsten

3.2.2 Formale Syntax

- Alphabet: $A_{AL} = \{(,), \neg, \wedge, \vee, \rightarrow\} \cup \mathrm{Var}_{AL}$

  bspw. $(P_0 \vee P_1) \rightarrow P_0$, aber auch $v))P_{42}\neg($

- Funktionen für Konnektive

$$\begin{aligned} f_\neg &: A_{AL}^* \rightarrow A_{AL}^* & G &\mapsto (\neg G) \\ f_\wedge &: A_{AL}^* \times A_{AL}^* \rightarrow A_{AL}^* & (G, H) &\mapsto (G \wedge H) \\ f_\vee &: A_{AL}^* \times A_{AL}^* \rightarrow A_{AL}^* & (G, H) &\mapsto (G \vee H) \\ f_\rightarrow &: A_{AL}^* \times A_{AL}^* \rightarrow A_{AL}^* & (G, H) &\mapsto (G \rightarrow H) \end{aligned}$$

- induktive Definition syntaktisch korrekter Formeln

$$\begin{aligned} M_0 &= \mathrm{Var}_{AL} \\ \forall n \in \mathbb{N}_+ : M_n &= M_{n-1} \cup f_\neg(M_{n-1}) \\ &\quad \cup f_\wedge(M_{n-1} \times M_{n-1}) \\ &\quad \cup f_\vee(M_{n-1} \times M_{n-1}) \\ &\quad \cup f_\rightarrow(M_{n-1} \times M_{n-1}) \end{aligned}$$

- alle Formeln zusammen $\mathrm{For}_{AL} = \bigcup_{i \in \mathbb{N}_0} M_i$
- Abkürzung $(G \leftrightarrow H) \iff ((G \rightarrow H) \wedge (H \rightarrow G))$

3.3 Boolsche Funktionen

- Menge der Wahrheitswerte $\mathbb{B} = \{\mathbf{w}, \mathbf{f}\}$
- boolsche Funktion ist eine Abbildung $f : \mathbb{B}^k \rightarrow \mathbb{B}$

$b_\neg(x)$	$\neg x$	Negation
$b_\wedge(x, y)$	$x \wedge y$	Konjunktion (Und)
$b_\vee(x, y)$	$x \vee y$	Disjunktion (Oder)
$b_\rightarrow(x, y)$	$x \implies y$	Implikation

- Wahrheitswerte der obigen boolschen Funktionen

x_1	x_2	$b_\neg(x_1)$	$b_\wedge(x_1, x_2)$	$b_\vee(x_1, x_2)$	$b_\rightarrow(x_1, x_2)$
f	**f**	**w**	**f**	**f**	**w**
f	**w**	**w**	**f**	**w**	**w**
w	**f**	**f**	**f**	**w**	**f**
w	**w**	**f**	**w**	**w**	**w**

- Anzahl der boolschen Funktionen $|\mathbb{B}^{\mathbb{B}^k}| = |\mathbb{B}|^{|\mathbb{B}^k|} = 2^{(2^k)}$

3.4 Semantik aussagenlogischer Formeln

Im folgenden sei V eine Menge von Aussagevariablen

- Interpretation $I : V \to \mathbb{B}$
- $\mathbb{B}^V$ als Menge aller Interpretationen
- definiere die **Auswertung** $val_I(F)$ für jede aussagenlogische Formel F

$$val_I : \text{For}_{AL} \to \mathbb{B} = \{\mathbf{w}, \mathbf{f}\}$$

- I ist **Modell** einer Formel G, wenn $val_i(G) = w$
- I ist **Modell** einer Formelmenge Γ, wenn I Modell jeder Formel $G \in \Gamma$ ist
- schreibe $\Gamma \models G$ (jedes Modell von Γ auch ein Modell von G)
- schreibe $\models G$ statt $\emptyset \models G$ für Tautologien (G für *alle* Interpretationen wahr
- G **erfüllbar** $\iff \exists I \in \mathbb{B}^V : val_I(G) = w$
- zwei Formeln G, H heißen **äquivalent**, wenn für jede Interpretation I gilt:

$$val_I(G) = val_I(H)$$

  Schreibe $G \equiv H$

3.5 Beweisbarkeit im Aussagenkalkül

3.5.1 Das Aussagenkalkül

Kalkül allgemein

- Alphabet A
- syntaktisch korrekte Formeln $For \subseteq A^*$
- **Axiome** $Ax \subseteq For$
- **Schlussregeln** $R \subseteq For_{AL}^k$

Aussagenkalkül für die Aussagenlogik

- Alphabet A_{AL}
- syntaktisch korrekte Formeln $For_{AL} \subseteq A_{AL}^*$
- **Axiome** $Ax_{AL} \subseteq For_{AL}$

  Axiome sind Formeln die gegeben sind.

- Schlussregel **Modus Ponens** $MP \subseteq For_{AL}^3$

  $MP = \{(G \rightarrow H, G, H) | G, H \in For_{AL}\}$. "Mit $G \rightarrow H$ und G bekommen wir auch H"

Ableitungen

- nutzen Prämissen, Axiome und Schlussregeln
- wichtiger Bestandteil von Beweisen
- endliche Folge $(G1, \ldots, G_n)$ von Formeln mit
  - $G_n = G$ (irgendwo, kann auch weiter vorne sein)
  - Jedes G_i entweder Axiom, Prämisse oder Bestandteil einer Schlussregeln

    $(G_{i_1}, G_{i_2}, G_i) \in MP$ mit $i_1, i_2 < i$
- schreibe $\Gamma \vdash G$

3.5.2 Beweise im Aussagenkalkül

- formal eine Ableitung aus $\Gamma = \emptyset$
- schreibe $\vdash G$
- G heißt **Theorem** des Aussagenkalküls

Beispiel eines Beweises im Aussagenkalkül

Beispiel eines Beweises: $(\neg P \rightarrow P) \rightarrow P$

$$
\begin{array}{ll}
G_1 (\neg P \rightarrow \neg P) \rightarrow ((\neg P \rightarrow P) \rightarrow P) & Ax_{AL3} \\
G_2 \neg P \rightarrow \neg P & \\
G_3 (\neg P \rightarrow P) \rightarrow P & MP(1,2)
\end{array}
$$

4 Induktion

4.1 Vollständige Induktion

Ein Beweisprinzip, welches auf einer fundamentalen Eigenschaft der natürlichen Zahlen beruht.

Allgemeines Muster eines Beweises (M Menge, A Aussage)

1. (**Induktionsanfang**, IA) zeigen, dass für das kleinste Element n_0 aus M gilt: $A(n_0) = w$
2. (**Induktionsvoraussetzung**, IV) nötige Voraussetzung, welche als wahr angenommen wird damit IS gilt. Meistens sie so aufgebaut:

   Sei festes $n \in M$ derart, dass $A(n-1) = w$
3. (**Induktionsschritt**, IS) folgern, dass folgendes gilt

   $[\forall n \in M : A(n-1) = w] \implies A(n) = w$

   Sprich: "*zeige A_n ist wahr, falls A_{n-1} wahr ist für ein beliebiges $n \in \mathbb{N}_+$*"

Beispiel: $A_n = [\forall A \forall w \in A^* : |w^n| = n \cdot |w|]$ Induktionsanfang ($n = 0$)

$$\forall A \forall w \in A^* : |w^0| = |\varepsilon| = 0 = 0 \cdot |w|$$

Induktionsschritt ($n \in \mathbb{N}_+$)

$$\begin{aligned} &\forall A \forall w \in A^* : \\ &\qquad |w^n| = |w^{n-1} \cdot w| = |w^{n-1}| + |w| \\ &\qquad \overset{A_{n-1}}{=} (n-1) \cdot |w| + |w| = n \cdot |w| \end{aligned}$$

4.2 Varianten vollständiger Induktion

- Induktionsanfang an anderer Stelle (z.B. $n_0 = 1$ statt $n_0 = 0$)
- mehrere Induktionsanfänge

  z.B IA für $n \leq 2$: A_0, A_1, A_2 sind wahr. Dann der IS für $n \geq 3$

- starke Induktion. Hier werden im Induktionsschritt *alle* "früheren Aussagen"verwendet (nicht nur von $A(n-1)$)

  Also $A(n)$ ist wahr, falls $A(k)$ für alle $k < n$ wahr ist

- geschachtelte Induktion. Nützlich für Funktionen, welche auf Matrizen (bspw. $A^{p \times q}$) definiert sind

  Für $A_{m,n}$ gibt es äußere Induktion über m und innere Induktion über n

5 Formale Sprachen

5.1 Formale Sprachen

Eine formale Sprache L über einem Alphabet A ist eine Teilmenge aller Wörter über A, also $L \subseteq A^*$

- Syntax, Dinge die formal korrekt sind (auf syntaktischer Ebene korrekt)
- Semantik, die Bedeutung syntaktisch korrekter Dinge

Beispiel $A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, -\}$ ist die formale Sprache der Dezimaldarstellungen ganzer Zahlen

- $1, -22 \in A$
- $2 - 33 - -21 \notin A$

5.2 Produkte und Potenzen formaler Sprachen

5.2.1 Produkte

Seien L_1, L_2 formale Sprachen. Dann ist das Produkt der beiden Sprachen definiert als

$$L_1 \cdot L_2 := \{w_1 w_2 | w_1 \in L_1, w_2 \in L_2\}$$

Das Produkt formaler Spachen ist assoziativ (aufgrund der Assoziativität der Konkatenation)

$$L_1 \cdot L_2 \cdot L_3 = \{w_1 w_2 w_3 | w_1 \in L_1, w_2 \in L_2, w_3 \in L_3\}$$

Beispiel: Java Deklarationen (fast). Seien $S = \{\text{int}, \text{double}, \text{char}\}, B = \{\text{a}, \ldots, \text{z}\}, Z = \{0, \ldots, 9\}$ Sprachen

$$L := S \cdot \{\sqcup\} \cdot B \cdot (B \cup Z \cup \{\varepsilon\}) \cdot \{;\}$$

- $\text{int} \sqcup \text{x2;} \in L$
- $\text{double} \sqcup \text{w;} \in L$
- leider aber nicht $\text{char} \sqcup \sqcup \text{hugo} \sqcup \text{;} \notin L$

Das neutrale Element ε verändert eine Sprache nicht $L \cdot \{\varepsilon\} = L = \{\varepsilon\} \cdot L$

5.2.2 Potenzen

Die Potenz n einer Sprache L ist induktiv definiert (wie bei Wörtern)

$$L^0 = \{\varepsilon\}$$
$$\forall n \in \mathbb{N}_+ : L^n = L \cdot L^{n-1}$$

- man kann Tupel als Wörter auffassen
- dann sind die Definitionen von Potenzen von Alphabeten und Sprachen "dasselbe"
- es gilt $L^* = A^*$

5.3 Konkatenationsabschluss formaler Sprachen

- Konkatenationsabschluss L^* von L (**kleenesche Hülle**)

$$L^* = \bigcup_{i \in \mathbb{N}_0} L^i$$

- ε-freier Konkatenationsabschluss L^+ von L (**positiver Abschluss**)

$$L^+ = \bigcup_{i \in \mathbb{N}_+} L^i$$

- $L^* = L^0 \cup L^+$

Beispiel:

$$L = \{\mathrm{a}\}^* \cup \{\mathrm{b}\}^* = \{\varepsilon, \mathrm{a}, \mathrm{aa}, \ldots, \mathrm{b}, \mathrm{bb}, \ldots\}$$

Dann enthält L^* z.B.

- $\mathrm{aa} \cdot \varepsilon \cdot \mathrm{aaaa} \cdot \mathrm{b}$
- $\mathrm{bbb} \cdot \varepsilon \cdot \mathrm{a}$

Achtung bei L^+ "ε-freier" Konkatenationsabschluss

- wenn $\varepsilon \in L$, dann ist auch $\varepsilon \in L^+$!
- ebenfalls gilt $\emptyset^* = \{\varepsilon\}$

6 Zahlendarstellungen und Kodierungen

6.1 Von Wörtern zu Zahlen und zurück

6.1.1 Division mit Rest

Sei $x \in \mathbb{N}_0$ und $y \in \mathbb{N}_+$

- x **div** $y \in \mathbb{N}_0$ (ganzzahlige Division von x durch y)
- x **mod** $y \in \mathbb{N}_0$ (Rest der ganzzahligen Division von x durch y)

$$0 \leq x \bmod y < y$$

- es gilt: $\forall x \in \mathbb{N}_0, y \in \mathbb{N}_+$

$$x = y \cdot (x \operatorname{div} y) + (x \bmod y)$$

6.1.2 k-äre Darstellung von Zahlen

Sei $k \in \mathbb{N}_0$ mit $k \geq 2$ und Z_k Alphabet mit k Ziffern ($x_{k-1} \cdots x_0 \in Z_k^*$)

- Konvertierung von Wort zu Zahl: $\mathrm{Num}_k : Z_k^* \to \mathbb{N}_0$ (linksinvers zu Repr_k)

$$\mathrm{Num}_k(w) = \begin{cases} \varepsilon, & \text{falls } w = 0 \\ k \cdot \mathrm{Num}_k(w_1) + \mathrm{Num}_k(w_2), & \text{falls } w \neq \varepsilon \text{ mit } w_1 \in Z_k^*, w_2 \in Z_k \end{cases}$$

- Konvertierung von Zahl zu Wort: $\mathrm{Repr}_k : \mathbb{N}_0 \to Z_k^+$

$$\mathrm{Repr}_k(i) = \begin{cases} \text{i}, & \text{falls } n < k \\ \mathrm{Repr}_k(i \operatorname{div} k) \cdot \mathrm{Repr}_k(i \bmod k) & \text{falls } n \geq k \end{cases}$$

  Es gilt für alle $n, k \in \mathbb{N}_0$ ($k \geq 2$): $\mathrm{Num}_l(\mathrm{Repr}_k(n)) = n$

- *binäre* Darstellung für $k = 2$
- *ternäre* Darstellung für $k = 3$

6.2 Von einem Alphabet zum anderen

6.2.1 Übersetzungn allgemein

- sind Abbildungen $t : L_A \longrightarrow L_B$ (L_A,L_B Sprachen)
- von einer Zahlendarstellung in eine andere $\text{Trans}_{n,m} : Z_n^* \longrightarrow Z_m^*$

$$\text{Trans}_{n,m} = \text{Repr}_m \circ \text{Repr}_n$$

- wichtig für bspw. Verschlüsselung, Kompression, Fehlererkennung und Fehlerkorrektur
- wenn t injektiv, kann man von $t(w) \in L_B$ eindeutig zu $w \in L_A$
- injektive Übersetzungen heißen **Kodierungen**
  - für $w \in L_A$ ist $t(w) \in L_B$ das zugehörige **Codewort**
  - das Bild von t heißt der **Code**

6.2.2 Homomorphismen

- eine Abbildung $h : A^* \longrightarrow B^*$, mit A,B Alphabete, heißt **Homomorphismus**, wenn $\forall w_1, w_2 \in A^*$:

$$h(w_1 w_2) = h(w_1)h(w_2)$$

- es gilt $h(\varepsilon) = \varepsilon$ für alle Homomorphismen
- h heißt ε-**frei**, wenn $\forall x \in A : h(x) \neq \varepsilon$

Beispiel: Sei h Homomorphismus, $h(\text{a}) = 10$ und $h(\text{b}) = 001$

$$\begin{aligned} h(\text{bab}) &= h(\text{ba}) \cdot h(\text{b}) \\ &= h(\text{b}) \cdot h(\text{a}) \cdot h(\text{b}) \\ &= 001 \cdot 10 \cdot 001 \\ &= 00110001 \end{aligned}$$

Der induzierte Homomorphismus

Seien A,B Alphabete, $f : A \longrightarrow B^*$ Abbildung. Dann ist der durch f induzierte Homomorphismus $f^{**} A^* \longrightarrow B^*$

$$f^{**}(w) = \begin{cases} \varepsilon, & \text{falls } w = \varepsilon \\ f^{**}(w')f(x), & \text{falls } w = w'x \text{ mit } w' \in A^*, x \in A \end{cases}$$

Beispiel: $f(0) = \text{aba}, f(1) = \text{bb}$

$$\begin{aligned} f^{**}(0010) &= f^{**}(001)f(0) = f^{**}(001) \cdot \text{aba} \\ &= f^{**}(00)f(1) \cdot \text{aba} = f^{**}(00) \cdot \text{bb} \cdot \text{aba} \\ &= \dots \\ &= \text{aba} \cdot \text{aba} \cdot \text{bb} \cdot \text{aba} = \text{abaababbaba} \end{aligned}$$

6.2.3 Präfixfreie Codes

Im folgenden sei A Alphabet und $w \in A^*$ Wort.

- ein Wort $a \in A^*$ heißt **Präfix** von w, wenn

$$\exists b \in A^* : a \cdot b = w$$

- ein Wort $b \in A^*$ heißt **Suffix** von w, wenn

$$\exists a \in A^* : a \cdot b = w$$

- ε ist immer sowohl Präfix als auch Suffix von w
- ein Homomorphismus $h : A^* \to B^*$ heißt **präfixfrei**, wenn $\forall x_1, x_2 \in A$ mit $x_1 \neq x_2$: $h(x_1)$ kein Präfix von $h(x_2)$
- ein präfixfreier Homomorphismus ist immer injektiv, aber h im allgemeinen nicht surjektiv

6.3 Huffman-Kodierung

- liefert kürzest mögliche präfixfreie Codes (”ideale Kodierung”)
- Bestandteil von z.B. zip, gzip, png, …

6.3.1 Algorithmus zur Berechnung von Huffman-Codes

- Anzahl an Vorkommen von x in w: $|w|_x$

$$|w|_x = \begin{cases} 0, & \text{falls } w = \varepsilon \\ 1 + |w'|_x, & \text{falls } w = w'x \text{ mit } w' \in A^* \\ |w'|_x, & \text{falls } w = w'y \text{ mit } w' \in A^*, y \neq x \end{cases}$$

- erfolgt in zwei Phasen (1. Konstruktion eines ”Baumes”, 2. Ablesen des Codes aus dem Baum)

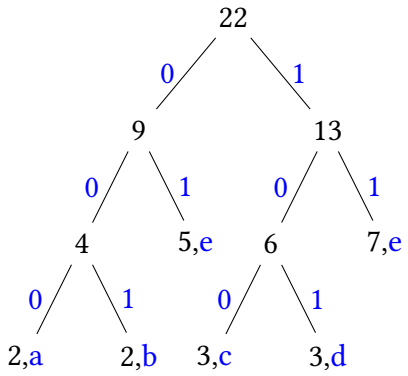
Für Schritt $i \in \mathbb{N}_+$ sei M_i die Menge der Symbole mit ihren Häufigkeiten.

$$M_1 := \{(|w|_x, \{x\})\} \ (x \in A)$$
$$M_{i+1} := (M_i \setminus \{(k_1, X_1), (k_2, X_2)\} \cup \{(k_1 + k_2, X_1 \cup X_2)\})$$

Wobei $(k_1, X_1), (k_2, X_2)$ die Elemente aus M_i mit den kleinsten Häufigkeiten sind

- der Baum ist gewurzelt (Knoten $(|w|, A)$)
- Blätter des Baumes sind Zeichen $x \in A$
- jeder innere Knoten hat genau zwei Kinder

Beispiel: w = afebfecaffdeddccefbeff



- dieser entstehende Homomorphismus ist präfixfrei!
- Kodierung erfolgt durch Ablesen der Kodierungen (gehe auf kürzestem Weg von Wurzel zu jedem Blatt für x)
- Dekodierung erfolgt analog
- beim obigen Beispiel ist $h(c) = 100$

7 Kontextfreie Grammatiken

7.1 Kontextfreie Grammatiken

- Grammatik $G = (\Sigma, V, S, R)$
- Σ Alphabet der **Terminalsymbole**
- V Alphabet der **Nichtterminalsymbole** bzw. Variablen
- es gilt $\Sigma \cap V = \emptyset$
- $S \in V$ ist das **Startsymbol**
- $R \subseteq V \times (\Sigma \cup V)^*$ endliche Menge an **Ableitungsregeln** (Schreibe: $A \to \beta$)

  Sprich: *man kann Symbol X ersetzen durch Wort w*

Beispiel: $G = (\{a, b\}, \{X\}, X, \{X \to \varepsilon, X \to aXb\})$

- Ersetzung aus $aXabXX$ wird $aaXbabXX$

7.2 Ableitungen

Sei $X \to w$ eine Ableitungsregel

- linke Seite X ist immer eine Variable
- Terminalsymbole können nicht ersetzt werden
- Ableitungen sind **kontextfrei** (Ersetzung immer überall möglich, auch unabhängig vom Kontext)
- die Ableitung $\to$ ist binäre Relation ($\to \subseteq (\Sigma \cup V)^* \times (\Sigma \cup V)^*$
- im allgemeinen, keine Eigenschaften (nicht linkstotal, rechtstotal, linkseindeutig und rechtseindeutig)

Ableitungsfolgen

Für alle $u, v \in (\Sigma \cup V)^*$ sind die Ableitungsfolgen "$\overset{i}{\rightarrow}$" und "$\overset{*}{\rightarrow}$" definiert als

$$u \overset{0}{\rightarrow} v \text{ wenn } u = v$$
$$\forall i \in \mathbb{N}_+ : u \overset{i}{\rightarrow} v \text{ wenn } \exists w \in (\Sigma \cup V)^* : u \rightarrow w \overset{i-1}{\rightarrow} v$$
$$u \overset{*}{\rightarrow} v \text{ wenn } \exists i \in \mathbb{N}_0 : u \overset{i}{\rightarrow} v$$

Beispiel: $G = (\{a, b\}, \{X\}, X, \{X \rightarrow \varepsilon, X \rightarrow aXb\})$

- $X \rightarrow aXb \rightarrow aaXbb \rightarrow aabb$
- $X \overset{2}{\rightarrow} aaXbb$
- $X \overset{4}{\rightarrow} aaabbb$
- $abb \overset{0}{\rightarrow} abb$

Kompakte Notation

- statt $\{X \rightarrow w_1, X \rightarrow w_2, X \rightarrow w_3, X \rightarrow w_4, X \rightarrow w_5\}$, schreibe
$$\{X \rightarrow w_1 | w_2 | w_3 | w_4 | w_5\}$$
- im vorherigen Beispiel: $G = (\{a, b\}, \{X\}, X, R)$ mit
$$R = \{X \rightarrow aXb | \varepsilon\}$$

7.3 Erzeugte formale Sprache

Die von $G = (\Sigma, V, S, R)$ erzeugte formale Sprache $L(G)$ ist definiert als

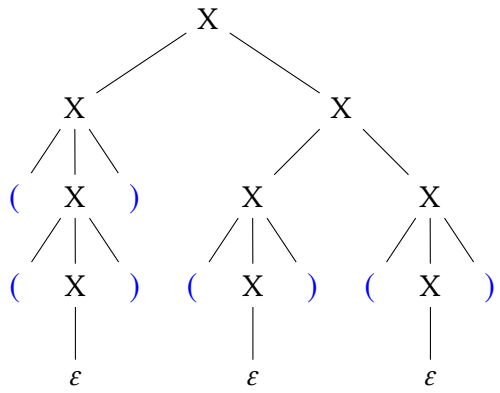
$$L(G) = \{w \in \Sigma^* | S \overset{*}{\rightarrow} w\}$$

- formale Sprachen von einer Grammtik erzeugt, heißen **kontexfrei**
- $L(G)$ erhält nur Wörter aus Σ^*
- die Ableitung $S \rightarrow \ldots \rightarrow w$ für $w \in L(G)$ enthält aber Variablen

Beispiel: $G = (\{a, b\}, \{X\}, X, \{X \rightarrow \varepsilon, X \rightarrow aXb\})$

- für alle $i \in \mathbb{N}_0$ gilt: $X \overset{*}{\rightarrow} a^i b^i$, also $\{a^i b^i | i \in \mathbb{N}_0\} \subseteq L(G)$
- Umgekehrt zeigt man: für jedes $i \in \mathbb{N}_0$: wenn $X \overset{i+1}{\rightarrow} w$, dann $w = a^i b^i$ oder $w = a^{i+1} X b^{i+1}$
  es gilt also $L(G) \subseteq \{a^i b^i | i \in \mathbb{N}_0\}$
- $\Longrightarrow L(G) = \{a^i b^i | i \in \mathbb{N}_0\}$

7.4 Ableitungsbäume



- sind übersichtlicher als schrittweise Ableitungen
- beginne mit Startsymbol als Wurzel
- Ableitung $X \rightarrow w$: von erstztem X zu jedem Symbol von w eine separate Kante nach unten

Beispiel: **Syntax aussagenlogischer Formeln**

- $G = (\Sigma, \{X\}, X, R)$ mit $\Sigma = \text{Var}_{AL} \cup \{(,), \neg, \wedge, \vee, \rightarrow\}$
- Regelmenge R

$$R = \{X \rightarrow \text{P}_i | \text{P}_i \in \text{Var}_{AL}\}$$
$$\cup \{X \rightarrow (\neg X) | (X \neg X) | (X \vee X) | (X \rightarrow X)\}$$

8 Prädikatenlogik

- vieles kann in Aussagenlogik nicht dargestellt werden $\implies$ Prädikatenlogik
- bekannte Elemente (aus Aussagenlogik): Variablen, logische Verknüpfungen, Klammern
- neue Elemente: Quantoren, Relationen, Funktionen, Konstanten

8.1 Syntax prädikatenlogischer Formeln

- Prädikatenlogik besteht aus drei syntaktischen Einheiten
- **Terme**: aus Variablensymbole, Funktionssymbole
- **atomare Formeln**: aus Terme (mithilfe von Relationssymbolen)
- **prädikatenlogische Formeln**: aus atomaren Formeln unter Verwendung aussagenlogischer Konnektive und Quantoren

8.1.1 Relationen und Funktionen

- Objekte bekommen Werte einer **Definitionsmenge** D zugewiesen
- Funktionen $f : D^k \to D$, $ar(f) = k$ heißt **Stelligkeit** von f (engl. arity)
- Konkrete Beispiele für Funktionen
  - $f(x_1, x_2) = x_1 + x_2$ (Addition $a + b$)
  - $f(x_1, x_2, x_3) = x_1^{x_3} + x_2^{x_3}$ ($a^n + b^n$)
  - $f(x_1, x_2) = x_1 x_2$ (Konkatenation uv)
  - $f() = c$ (**Konstante** c)
- Relationen $R \subseteq D^k = \{(x_1, \ldots, x_k) | x_1, \ldots, x_k \in D\}$, $ar(R) = k$

8.1.2 Signatur

- Alphabet Var_{PL} sind **Variablensymbole** (x_i, endlich viele $i \in \mathbb{N}_0$, kurz $\text{x}, \text{y}, \text{z}$)
- Alphabet Fun_{PL} sind **Funktionssymbole** (jede $\text{f}_i \in \text{Fun}_{PL}$ hat **Stelligkeit** $ar(\text{f}_i) \in \mathbb{N}_0$, kurz $\text{f}, \text{g}, \text{h}$)
- Alphabet Rel_{PL} sind **Relationssymbole/Prädikatensymbole** (jedes $\text{R}_i \in \text{Rel}_{PL}$ hat **Stelligkeit** $ar(\text{R}_i) \in \mathbb{N}_0$)
- eine Signatur ist definiert als $S = (\text{Var}_{PL}, \text{Fun}_{PL}, \text{Rel}_{PL})$
- **Alphabet für Prädikatenlogik**

$$A_S = \{\neg, \wedge, \vee, \rightarrow, (, ,,), \forall, \exists\} \cup \text{Var}_{PL} \cup \text{Fun}_{PL} \cup \text{Rel}_{PL}$$

Terme

- Menge Ter_S der Terme
- $\forall \text{x} \in \text{Var}_{PL} : \text{x} \in \text{Ter}_S$ (jede Variable ist ein Term)
- $\forall \text{f} \in \text{Fun}_{PL}, t_1, \ldots, t_{ar(\text{f})} \in \text{Ter}_S$, dann ist

$$\text{f}(t_1, \ldots, t_{ar(\text{f})}) \in \text{Ter}_S$$

Atmorare Formeln

Menge AtFor_S der atmoaren Formeln. Wenn $\text{R} \in \text{Rel}_{PL}, t_1, \ldots, t_{ar(\text{R})} \in \text{Ter}_S$, dann ist

$$\text{R}(t_1, \ldots, t_{ar(\text{R})}) \in \text{AtFor}_S$$

Formeln

- Menge For_S der Formeln
- $\forall A \in \text{AtFor}_S : A \in \text{For}_S$ (jede atomare Formel ist eine Formel)
- $\forall \text{x} \in \text{Var}_{PL}, A \in \text{For}_S : (\forall \text{x} A), (\exists \text{x} A) \in \text{For}_S$
- $\forall A, B \in \text{For}_S$, dann ist

$$\neg A, (A \wedge B), (A \vee B), (A \rightarrow B) \in \text{For}_S$$

Beispiel: ”Satz des Euklid”

$$\forall x \exists y : (y \geq x) \wedge y\text{Primzahl}$$

- Signatur $S = (\text{Var}_{PL}, \text{Fun}_{PL}, \text{Rel}_{PL})$

$$\text{Var}_{PL} = \{\text{x}, \text{y}\}, \text{Fun}_{PL} = \emptyset, \text{Rel}_{PL} = \{\text{R}, \text{S}\}, ar(\text{R}) = 2, ar(\text{S}) = 1$$

- Formel $F = \forall \text{x} \exists \text{y}\ \text{R}(\text{y}, \text{x}) \wedge \text{S}(\text{y})$

8.2 Semantik prädikatenlogischer Formeln

8.2.1 Auswertung

Interpretationen Sei $S = (\mathrm{Var}_{PL}, \mathrm{Fun}_{PL}, \mathrm{Rel}_{PL})$ Signatur.

- (D, I) ist **Interpretation** von S
- $D \neq \emptyset$ heißt **Universium** (engl. *domain*)
- um zu schauen, ob Formel F wahr oder falsch, wird **Variablenbelegung** $\beta : \mathrm{Var}_{PL}$ benötigt

  Beispiel: $\beta(\mathrm{x}) = 3$ und $\beta(\mathrm{y}) = 42$

Auswertungsfunktion – Terme

Sei $S = (\mathrm{Var}_{PL}, \mathrm{Fun}_{PL}, \mathrm{Rel}_{PL})$ Signatur, (D, I) Interpretation von S, $\beta : \mathrm{Var}_{PL} \to D$ Variablenbelegung.

Definiere die Auswertung von Termen $\mathrm{tval}_{D,I,\beta} : \mathrm{Ter}_S \to D$

- für Variablen $\mathrm{x} \in \mathrm{Var}_{PL}$

$$\mathrm{tval}_{D,I,\beta}(\mathrm{x}) = \beta(\mathrm{x})$$

- für zusammengesetzte Terme $\mathrm{f}(t_1, \ldots, t_k)$ $(k = ar(\mathrm{f}))$

$$\mathrm{tval}_{D,I,\beta}(\mathrm{f}(t_1, \ldots, t_k)) = I(\mathrm{f})(\mathrm{tval}_{D,I,\beta}(t_1), \ldots, \mathrm{tval}_{D,I,\beta}(t_k))$$

Auswertungsfunktion – Formeln

Sei $S = (\mathrm{Var}_{PL}, \mathrm{Fun}_{PL}, \mathrm{Rel}_{PL})$ Signatur, (D, I) Interpretation von S, $\beta : \mathrm{Var}_{PL} \to D$ Variablenbelegung.

Definiere die Auswertung von Formeln $\mathrm{val}_{D,I,\beta} : \mathrm{For}_S \to \mathbb{B}$

- für atomare Formeln $A \in \mathrm{AtFor}_S$

  $A = \mathrm{R}(t_1, \ldots, t_k)$ ($\mathrm{R} \in \mathrm{Rel}_{PL}$ mit $ar(\mathrm{R}) = k$ und Terme $t_1, \ldots, t_k$)

$$\mathrm{val}_{D,I,\beta}(\mathrm{R}(t_1, \ldots, t_k)) = \begin{cases} w, & \text{falls } (\mathrm{tval}_{D,I,\beta}(t_1), \ldots, \mathrm{tval}_{D,I,\beta}(t_k)) \in I(\mathrm{R}) \\ f, & \text{falls } (\mathrm{tval}_{D,I,\beta}(t_1), \ldots, \mathrm{tval}_{D,I,\beta}(t_k)) \notin I(\mathrm{R}) \end{cases}$$

- für zusammengesetzte Formel $F \in \mathrm{For}_S$

  $F = \neg A, (A \wedge B), (A \vee B), (A \to B)$ $(A, B \in \mathrm{For}_S$

  Analog wie in Aussagenlogik, z.b. $\mathrm{val}_{D,I,\beta}(A \wedge B) = \mathrm{val}_{D,I,\beta}(A) \wedge \mathrm{val}_{D,I,\beta}(B)$

- für zusammengesetzte Formel $F \in \text{For}_S$

  $F = (\forall \text{x} A), (\exists \text{x} A)$ $(A \in \text{For}_S)$

  neue Variablenbelegung wird benötigt $\beta^d_{\text{x}} : \text{Var}_{PL} \to D$ (Überschreibung der Belegung von x in β durch $d \in D$

$$\beta^d_{\text{x}}(\text{y}) = \begin{cases} \beta(\text{y}), & \text{falls y} \neq \text{x} \\ d, & \text{falls y} = \text{x} \end{cases}$$

$$\text{val}_{D,I,\beta}(\forall \text{x} A) = \begin{cases} w, & \text{falls } \textbf{für jedes } d \in D : \text{val}_{D,I,\beta^d_{\text{x}}}(A) = w \\ f, & \text{sonst} \end{cases}$$

$$\text{val}_{D,I,\beta}(\exists \text{x} A) = \begin{cases} w, & \text{falls } \textbf{für mindestens ein } d \in D : \text{val}_{D,I,\beta^d_{\text{x}}}(A) = w \\ f, & \text{sonst} \end{cases}$$

Beispiel einer Formalisierung in Prädikatenlogik

"Wenn es jeden Tag eine GBI-Vorlesung gibt, dann gibt es auch am Mittwoch eine GBI-Vorlesung."

- $S = (\{\text{x}, \text{m}\}, \{\text{G}\}, \emptyset)$
- $F = (\forall \text{x}\ \text{G}(\text{x})) \to \text{G}(\text{m})$
- Interpretation: $D = \{Mo, Di, Mi, Do, Fr, Sa, So\}$

  $I(\text{G}) = \{d \in D | \text{Es gibt eine GBI-Vorlesung am } d\}$
- Auswertung: $\beta : \text{Var}_{PL} \to D, \beta(\text{x}) = Sa, \beta(\text{m}) = Mi$. Dann gilt

$$\text{val}_{D,I,\beta}((\forall \text{x}\ \text{G}(\text{x})) \to \text{G}(\text{m})) = w$$

  Dies gilt immer!

8.2.2 Modelle

- (D, I) ist **Modell** für $G \in \text{For}_S$, wenn (D, I) Interpretation für G und $\forall \beta : \text{val}_{D,I,\beta}(G) = w$
- (D, I) ist **Modell** für $\Gamma \subseteq \text{For}_S$, wenn (D, I) Modell $\forall G \in \Gamma$
- schreibe $\Gamma \models G$ (jedes Modell von Γ auch Modell von G
- schreibe $H \models G$, wenn $\{H\} \models G$
- schreibe $\models G$ statt $\emptyset \models G$ wenn G allgemeingültig (Tautologie)
- zwei Formeln $G, H \in \text{For}_S$ sind **logisch äquivalent**, wenn für jede Interpretation (D, I) von S und jede Variablenbelegung β gilt: $\text{val}_{D,I,\beta}(G) = \text{val}_{D,I,\beta}(H)$

9 Algorithmen

9.1 Der Algorithmus

- endliche Beschreibung
- Abfolgen von Schritten: elementare Anweisungen + endlich viele Schritte
- endliche Eingabe $\rightsquigarrow$ endliche Ausgabe
- **Determinismus** nächste elementare Anweisung eindeutig festgelegt
- Ablauf klar *nachvollziehbar*

Adjazenzmatrix

- $|V| \times |V|$ Matrix A_G indiziert nach Knoten
- $A_G[u,v] = \begin{cases} 1, & \text{falls } uv \in E \\ 1, & \text{falls } uv \notin E \end{cases}$

9.2 $\mathcal{O}$-Notation

- Laufzeit = Anzahl ausgeführter Anweisungen
- Bestimmung durch Abschätzungen der Laufzeit
- verschiedene Laufzeiten: best case, average case, worst case (eigentlich oft worst-case Abschätzungen)

Seien $f, g : \mathbb{N}_+ \to \mathbb{N}_+$ Funktionen

- f wächst **höchstens so schnell** wie g, wenn

  $$\exists c > 0 \exists n_0 \in \mathbb{N}_+ \forall n \geq n_0 : f(n) \leq c \cdot g(n)$$

  Schreibe: $f(n) \in \mathcal{O}(g(n))$
- f wächst **mindestens so schnell** wie g, wenn

  $$\exists c > 0 \exists n_0 \in \mathbb{N}_+ \forall n \geq n_0 : f(n) \geq c \cdot g(n)$$

  Schreibe: $f(n) \in \Omega(g(n))$

- f wächst **genauso schnell** wie g, wenn

$$f(n) \in O(g(n)) \text{ und } f(n) \in \Omega(g(n))$$

  Schreibe: $f(n) \in \Theta(g(n))$

Rechenregeln und Gesetze

- $a \cdot f(n) \in \Theta(f(n))$ (**Konstante Faktoren**)
- $n^a \in O(n^b) \iff a \leq b$ (**Monome**)
- $[f_1(n) \in O(g_1(n)) \land f_2(n) \in O(g_2(n))]$
  $\implies f_1(n) \cdot f_2(n) \in O(g_1(n) \cdot g_2(n))$ (**Produkte**)
- $[f(n) \in O(g(n)) \land g(n) \in O(h(n))] \implies f(n) \in O(h(n))$ (**Transitivität**)

Klassen von Funktionen Sei $f : \mathbb{N}_+ \to \mathbb{N}_+$ Funktion

- $O(f) = \{g : \mathbb{N}_+ \to \mathbb{N}_+ | \exists c > 0 \exists n_0 \in \mathbb{N}_+ \forall n \geq n_0 : g(n) \leq c \cdot f(n)\}$
- $\Omega(f) = \{g : \mathbb{N}_+ \to \mathbb{N}_+ | \exists c > 0 \exists n_0 \in \mathbb{N}_+ \forall n \geq n_0 : g(n) \geq c \cdot f(n)\}$
- $\Theta(f) = O(f) \cap \Omega(f)$

10 Graphen

10.1 Graphen

10.1.1 Definition

Ein **Graph** $G = (V, E)$ besteht aus

- endliche Knotenmenge V
- Kantenmenge $E \subseteq \binom{V}{2} = \{\{u, v\} | u, v \in V, u \neq v\}$

  $e \in E$ ist paar aus Knoten u, v. Schreibe $e = uv$

10.1.2 Arten von Graphen

- **Vollständiger Graph** K_n $(n \in \mathbb{N}_+)$

  $V(K_n) = \{v_1, \ldots, v_n\}, E(K_n) = \binom{V}{2}$
- **Pfad** P_n $(n \in \mathbb{N}_+)$

  $V(P_n) = \{v_1, \ldots, v_n\}, E(P_n) = \{v_i v_{i+1} | i \in [n-1]\}$

  Die Länge von P_n ist $n - 1$.
- **Kreis** C_n $(n \in \mathbb{N}_+, n \geq 3)$

  $V(C_n) = \{v_1, \ldots, v_n\}, E(C_n) = E(P_n) \cup \{v_1 v_n\}$

  Die Länge von C_n ist n.
- **Komplementgraph** $\overline{G} = (V', E')$

  $V' = V, E' = \{uv \in \binom{V}{2} \mid uv \notin E\}$
- **Kantengraph** $L(G) = (V', E')$

  $V' = E, E' = \{e_1 e_2 \in \binom{E}{2} \mid |e_1 \cap e_2| = 1\}$

Teilgraphen

Seien $G = (V_G, E_G), H = (V_H, E_H)$ Graphen.

$$H \text{ ist Teilgraph von } G \iff V_H \subseteq V_G \wedge E_H \subseteq E_G$$

Schreibe: $H \subseteq G$

spezielle Teilgraphen:

- $H \subseteq G$ vollständig $\implies V_H$ heißt **Clique** von G
- $H \subseteq G$ Pfad $\implies H$ **Pfad** in G
- $H \subseteq G$ Kreis $\implies H$ **Kreis** in G

10.1.3 Eigenschaften

Sei $G = (V, E)$ Graph

- **Cliquenzahl** von G ist $\omega(G) = \max\{\,|A| : A \subseteq V \text{ Clique}\,\}$
- G **zusammenhängend**, wenn zwischen allen zwei Knoten ein Pfad in G existiert
- G **kreisfrei**, wenn G keinen Kreis enthält
- inklusionsmaximale, zusammenhängende Teilgraphen von G: **Komponenten** / **Zusammenhangskomponenten**

10.2 Bäume und Wälder

Sei $G = (V, E)$ Graph mit $n = |V|$ Knoten

- G ist **Baum**, wenn zusammenhängend und kreisfrei (hat $n - 1$ Kanten)
- Bäume sind maximal kreisfrei und minimal zusammenhängend
- zwischen zwei Knoten in G gibt exakt einen Pfad
- jeder Teilgraph von G hat einen Knoten mit höchstens einer inzidenten Kante
- G ist **Wald**, wenn jede Komponente von G ein Baum ist

Weitere Notation von Bäumen

- **Wurzel** ist ausgezeichneter Knoten, also beliebiger Knoten mit Namen "Wurzel"
- **Blätter** sind Knoten mit ≤ 1 Nachbarn
- innere Knoten sind Knoten mit ≥ 2 Nachbarn
- **Kinder** von Knoten v sind Nachbarn mit größerem Abstand zur Wurzel

10.3 Bipartite Graphen

Sei $G = (V, E)$ Graph und $\chi(G) = \min\{k \in \mathbb{N}_+ : G \text{ hat eine k-Färbung}\}$

G ist bipartit $\iff \chi(G) \leq 2$

- bipartite Graphen ëinfachünd gut verstanden

10.4 Euler-Touren

Euler-Tour ist ein Pfad, um jede Kante einer Folge von Knoten **genau einmal** abzulaufen

Knotengrad

Die Anzahl der zu v inzidenten Kanten in G ist definiert als

$$deg(v) = |N(v)|$$

Es gilt für $G = (V, E)$ Graph mit $deg(v) \neq 0$ für jeden Knoten

$$\begin{aligned} & G \text{ hat eine geschlossene Euler-Tour} \\ \iff & G \text{ ist zusammenhängend und } \forall v \in V : deg(v) \mod 2 = 0 \end{aligned}$$

11 Endliche Automaten

11.1 Endliche Automaten

Ein Endlicher Automat $\mathcal{A} = (Q, q_0, \Sigma, \delta, F)$ besteht aus

- Q **Zustandsmenge** (endlich)
- $q_0 \in Q$ **Startzustand**
- Σ **Eingabealphabet**
- $\delta : Q \times \Sigma \to Q$ **Zustandsübergangsfunktion**
- $F \subseteq Q$ **akzeptierte Zustände**
- erkannte bzw. akzeptierte **formale Sprache**

$$L(\mathcal{A}) = \{ w \in \Sigma^* | \delta_*(q_0, w) \in F \}$$

Zustandsübergangsfunnktionen Sei $w \in \Sigma^*$ das eingegebene Wort in einen Endlichen Automaten

- $\delta_* : Q \times \Sigma^* \to Q$ **am Ende erreichter Zustand**

$$\delta_*(q, \varepsilon) = q$$
$$\forall w \in \Sigma^* \forall x \in \Sigma : \delta_*(q, wx) = \delta(\delta_*(q, w), x)$$

- $\delta_{**} : Q \times \Sigma^* \to Q^+$ **Folge der durchlaufenen Zustände**

$$\delta_{**}(q, \varepsilon) = q$$
$$\forall w \in \Sigma^* \forall x \in \Sigma : \delta_{**}(q, wx) = \delta_{**}(q, w) \cdot \delta_*(q, wx)$$

Akzeptanz Sei $w \in \Sigma^*$ eingegebenes Wort in einen EA

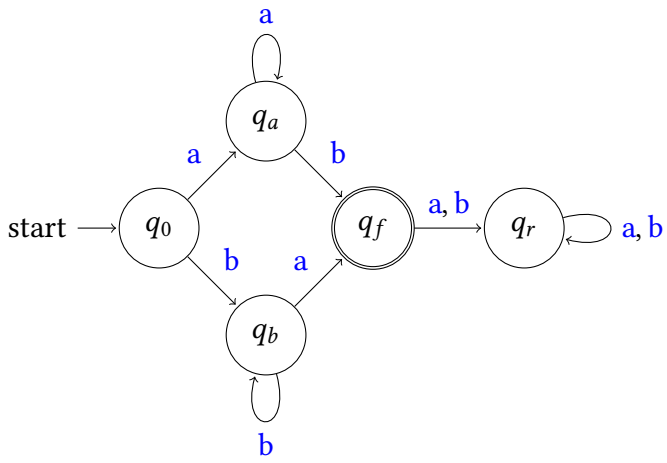
- w wird **akzeptiert**, falls $\delta_*(q_0, w) \in F$
- w wird **abgelehnt**, falls $\delta_*(q_0, w) \notin F$

11.2 Beispiel eines Endlichen Automaten

Sei $\mathcal{A} = (Q, q_0, \Sigma, \delta, F)$ Endlicher Automat mit

- $Q = \{ q_0, q_a, q_b, q_f, q_r \}$
- $\Sigma = \{ \mathrm{a}, \mathrm{b} \}$
- $F = \{ q_f \}$

Dieser kann mithilfe eines Graphen visualisiert werden



- $\delta_*(q_0, \mathrm{aaaba}) = q_r$
- $\delta_{**}(q_0, \mathrm{aaaba}) = q_0 q_a q_a q_a q_f q_r$
- $\delta_*(q_0, \mathrm{aaaba}) = q_r \notin F$ wird **abgelehnt**
- $\delta_*(q_0, \mathrm{aaab}) = q_f \in F$ wird **akzeptiert**