graph.tex

Richtig oder falsch: Jeder Graph $G = (V, E)$ mit $|V| > |E|$ ist ein Baum. $\boxed{\textbf{wichtig}}$

6 P

Zeichnen Sie einen zusammenhängenden Graphen, der sowohl einen Hamiltonkreis als auch eine 3–Färbung besitzt. **4 P**

(Geben Sie den Hamiltonkreis und die Färbung an.)

---

4 P

Zeichnen Sie einen zusammenhängenden Graphen, der weder einen Hamiltonkreis besitzt noch 3–färbbar ist.

(Begründen Sie die Eigenschaften.)

---

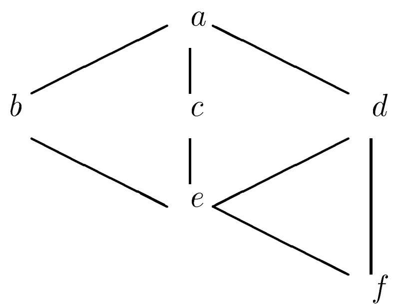
6 P

Zeigen Sie, daß jeder Baum 2–färbbar ist.

---

verband.tex

Sie sehen das Hasse–Diagramm einer Halbordnung $\leq$ auf $M = \{a, b, c, d, e, f\}$: **8 P**



Ist die Relation $\leq$

1. eine lineare Ordnung,
2. eine transitive Relation,
3. ein Verband,
4. eine wohlfundierte Relation,
5. ein distributiver Verband,
6. eine symmetrische Relation?

Sie müssen dazu jeweils Aussagen der Form $\forall x, y, \ldots \in M : P(x, y \ldots)$ überprüfen. Nennen Sie diese Aussagen! Fall sie zutreffen, geben Sie zwei Beispiele $(x_1, y_1, \ldots), (x_2, y_2, \ldots)$; falls nicht, wenigstens ein Gegenbeispiel.

sprache.tex **4 P**

Wir betrachten über dem Alphabet $\Sigma = \{0, 1\}$ eine Sprache L mit der Eigenschaft

$$L = 0 \cdot (\Sigma^* \setminus L) \cdot 1$$

Finden Sie ein Wort aus Σ^5, das zu L, und eines, das nicht zu L gehört.

Beschreiben Sie die Sprache L. Ist sie regulär? kontextfrei? **schwer**

---

8 P

Zeigen Sie, daß für $\Sigma = \{0, 1\}$ die Sprache

$$L = 0 \cdot (00)^* (\epsilon + 1\Sigma^*) \tag{1}$$

diese Gleichung erfüllt:

$$L = 0 \cdot (\Sigma^* \setminus L)^* \tag{2}$$

Geben Sie je zwei Wörter aus L und $\Sigma^* \setminus L$ an.

Bestimmen Sie aus Gleichung 1 einen regulären Ausdruck für $\Sigma^* \setminus L$.

Setzen Sie diesen dann in Gleichung 2 ein und verifizieren Sie die Übereinstimmung beider Seiten.

---

regular.tex **8 P** **wichtig**

Beweis oder Gegenbeispiel: Für reguläre Sprachen $A, B \subseteq \Sigma^*$ gilt

$$(A \cup B)^* = A^* \cup B^* \tag{3}$$

$$(A^* \cup B^*)^* = (A \cup B)^* \tag{4}$$

---

ackermann.tex **8 P**

Vervollständigen Sie die Definition der Ackermann–Funktion:

$$\begin{aligned} A(0, y) &= \boxed{} \\ A(x+1, 0) &= A(x, 1) \\ A(x+1, y+1) &= A(\boxed{}, A(\boxed{}, y)) \end{aligned}$$

Hinweis: $A(1, y) = y + 2, A(2, y) = 2y + 3$.

Leiten Sie einen geschlossenen Ausdruck für $A(3, y)$ ab.

Die Funktion A ist total und berechenbar, aber wächst so schnell, daß sie zwei Funktionenklassen trennt (sie ist in der einen, aber nicht in der anderen). Welche?

Man kann eine Umkehrung der Ackermann–Funktion ansetzen mit **schwer**

$$a(n) = \max\{x \mid A(x, x) \leq n + 1\}$$

Diese Funktion wächst ganz langsam, und gerät dadurch auch in die andere Funktionenklasse. Begründen Sie!

Hinweis. Finden Sie eine geeignete Schranke für die Anzahl der Rechenschritte bei $A(x, y)$.

pcp.tex

8 P

Gegeben sind ein Alphabet Σ sowie eine Folge von n Paaren von nichtleeren Wörtern $((u_1, v_1), \ldots, (u_n, v_n))$ mit $u_i, v_i \in \Sigma^+$

Welche der beiden Fragen ist als "Postsches Korrespondenzproblem" (PCP) bekannt?

1. $\exists m > 0, a_1, \ldots, a_m, b_1, \ldots, b_m$ mit $u_{a_1} \cdot \ldots \cdot u_{a_m} = v_{b_1} \cdot \ldots \cdot v_{b_m}$
2. $\exists m > 0, a_1, \ldots, a_m$ mit $u_{a_1} \cdot \ldots \cdot u_{a_m} = v_{a_1} \cdot \ldots \cdot v_{a_m}$

Das PCP ist bekanntlich nicht entscheidbar, die andere Frage läßt sich unter Benutzung regulärer Sprachen beantworten. Wie?

Begründen Sie, daß das PCP über Alphabet $\{0, 1\}$ genauso schwer ist wie das PCP über einem beliebigen größeren Alphabet.

Beschreiben Sie, wie beliebige PCP für Alphabete mit $|\Sigma| = 1$ entschieden werden können. Geben Sie je ein Beispiel für ein lösbares und ein unlösbares PCP dieser Art.

Beschreiben Sie, wie PCP mit nur einem Paar ($n = 1$), aber auf beliebigem Alphabet, entschieden werden können. Geben Sie je ein Beispiel für ein lösbares und ein unlösbares PCP dieser Art.

---

beaver.tex

8 P

Wir betrachten die Menge B_n aller (deterministischen, Einband–, Einkopf–)Turingmaschinen T mit Arbeitsalphabet $\Sigma = \{0, 1\}$ und genau n Zuständen, deren Rechnung terminiert, wenn sie auf dem leeren Band beginnt.

Teil A. Beschreiben Sie das Programm einer Maschine D_n mit n Zuständen, die nach Termination ein Wort von wenigstens $3n$ Zeichen "1" auf dem Band hinterläßt. (Keine Übergangstabelle nötig, sondern Idee erklären und Zustandszahl mit Ausgabelänge vergleichen.) (Hinweis: denken Sie an arithmetische Rechnungen mit Unär– oder Binärzahlen).

Teil B. Die Funktion $b : \mathbb{N} \to \mathbb{N}$ ist erklärt als

$$b(n) \quad = \quad \max_{T \in B_n} \{\text{Anzahl der 1, die nach Termination von } T \text{ auf dem Band stehen}\}$$

Diese Funktion b ist offenbar total (überall definiert) und streng monoton wachsend.

Zeigen Sie, daß die Funktion b nicht berechenbar ist.

Hinweis.

Falls es doch einen Algorithmus zur Berechnung von b gibt, dann kann dieser auch von einer Turingmaschine M mit Arbeitsalphabet $\{0, 1\}$ ausgeführt werden, die die Eingabe n als Unärzahl liest, und die Antwort $b(n)$ als Unärzahl schreibt. Die Maschine M hat m Zustände.

Betrachten Sie eine Maschine M', die auf dem leeren Band beginnt, dann D_m aufruft (also $3m$ Einsen schreibt) und danach M ausführt. Wieviel Zustände hat M'? Wieviel Einsen schreibt M'? Leiten Sie daraus einen Widerspruch ab.

`aufzahl.tex`

8 P **wichtig** **schwer**

Zeigen Sie, daß die Menge aller Paare von kontextfreien Grammatiken (G_1, G_2), für die $L(G_1) \neq L(G_2)$ gilt, rekursiv aufzählbar ist.

Zeigen Sie, daß diese Menge nicht entscheidbar ist. **Hinweis.** Simulieren Sie das PCP.

`hierarch.tex`

8 P **wichtig**

Sie kennen diese Inklusionen (wobei REC die Familie der rekursiv entscheidbaren Sprachen ist, und RE die Familie der rekursiv aufzählbaren Sprachen.)

$$\mathsf{Chomsky3} \subseteq \mathsf{Chomsky2} \subseteq \mathsf{Chomsky1} \subseteq \mathsf{Chomsky0};\ \mathsf{P} \subseteq \mathsf{NP};\ \mathsf{REC} \subseteq \mathsf{RE}$$

Welche weiteren Relationen bestehen zwischen den genannten Sprachfamilien?

Welche Inklusionen sind echt?

Gibt es Sprachen oberhalb von $\mathsf{Chomsky0}$?

`complexity.tex`

8 P **wichtig**

In der Berechenbarkeits–Theorie haben wir deterministische Turingmaschinen untersucht. Diese sind umständlich zu programmieren, und niemand würde so etwas tatsächlich als Prozessor in einem PC verwenden.

Warum erhalten wir daraus trotzdem Aussagen, die auch für tatsächlich praktisch benutzte Prozessoren und Programmiersprachen gelten? Nennen Sie eine solche Aussage. (Hinweis: Halteproblem?)

Warum läßt sich diese Aussage für Turingmaschinen leichter beweisen als für realer Programmiersprachen (z. B. C oder Java)? (Hinweis: vergleichen Sie den Aufwand, in dem jeweiligen Modell eine universelle Maschine zu schreiben).

---

8 P **wichtig**

Eine NP–Maschine ist eine reine Gedankenkostruktion: man kann so etwas, wegen des Nichtdeterminismus, nicht wirklich bauen, sondern höchstens simulieren.

Warum haben Aussagen über NP–Maschinen (bzw. über Sprachen, die durch NP–Maschinen akzeptiert werden können) trotzdem große praktische Bedeutung? (Benutzen Sie die Begriffe NP–vollständig und Reduktion.)

crossword.tex

8 P

wichtig

Ordnen Sie diese Begriffe aus der Theoretischen und Praktischen Informatik einander zu.

- primitiv–rekursive Funktion
- partiell–rekursive Funktion
- allgemein–rekursive Funktion
- universelle Turingmaschine
- Hauptnumerierung
- These von Church und Turing
- Polynomialzeit–Algorithmus
- NP–vollständiges Problem
- rekursiv unentscheidbares Problem
- Lambda–Kalkül
- (Umformen von) prädikatenlogischen Ausdrücken mit Quantoren
- Fixpunkte von Numerierungen, $s-n-m$–Theorem

- for–Programm
- Interpreter (z. B. für LISP, geschrieben in LISP)
- Compiler (z. B. von C nach Assembler)
- Programmier–Aufgabe ohne Lösung
- Programm, das seinen eigenen Quelltext druckt
- Programmtexte mit Blöcken und lokalen Deklarationen
- Auswahl zwischen verschiedenen Programmiersprachen
- Programmier–Aufgabe ohne effiziente Lösung
- effizientes Programm
- while–Programm
- Programm, das bei jeder Eingabe nach endlicher Zeit hält
- Funktionales Programmieren