Fragenkatalog Prüfungsfragen Prof. Dr. Verbeek

Dieser Fragenkatalog entstand während meiner Vorbereitung zur Bachelor Prüfung in Theoretischer Informatik (1657/1658) 2008. Hier sind ausschließlich Prüfungsfragen des Prof. Dr. Verbeek aus den Kursen 1653/4 gesammelt. Der Fragenkatalog enthält nicht die Fragen aus irgendwelchen Prüfungsprotokollen zu 1657/8.

Ich habe versucht, möglichst fehlerfrei zu arbeiten, kann jedoch nicht ausschließen, dass sich doch ein Fehler eingeschlichen hat. Deswegen kann ich auch keine Garantie für Korrektheit übernehmen.

Bitte immer mit den KEs lernen. Fragenkatalog pauken alleine ist zu WENIG!

Written by Jörg Jacob 2008 - studium (at) snuablo (punkt) com

Das Copyright für die Antworten liegt, soweit ohne explizite Hervorhebung wörtlich aus den Kurseinheiten zitiert wird (was um der Präzision willen oft nicht zu vermeiden war), grundsätzlich bei der FernUniversität in Hagen.

1. **Wie ist die Berechenbarkeit arithmetischer Funktionen definiert?**

   Mittels Registermaschinen. Eine Funktion $f :\subseteq \mathbb{N}^k \to \mathbb{N}$ heißt berechenbar, wenn eine Registermaschine M existiert mit $f = f_M$.

2. **Gibt es für die Berechenbarkeit noch andere Ansätze als die über Registermaschinen?**

   Ja, primitiv-rekursive Funktionen, WHILE-berechenbarkeit, LOOP-berechenbarkeit, μ-rekursive Funktionen.

   Nach Rechenzeitsatz existiert zu einer primitiv-rekursiven Funktion $f : \mathbb{N}^k \to \mathbb{N}$ eine Registermaschine M, eine Konstante $c \in \mathbb{N}$ und eine Zahl n, so dass $f = f_M$. Somit ist eine Funktion primitiv-rekursiv, gdw. sie mit einer RM berechnet werden kann.

   Berechenbare Zahlenfunktionen lassen sich auch über die WHILE Programme definieren. Zu jeden WHILE-Programm P gibt es ein Registermaschinen- Flussdiagramm F mit $\tau(P) = f_F$ und umgekehrt. Damit ist eine Funktion $f :\subseteq \mathbb{N}^k \to \mathbb{N}$ WHILE-bb., gdw. sie Registermaschinen-bb. ist.

   Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ ist LOOP-bb., gdw. sie primitiv-rekursiv ist.

   Die μ-rekursiven Funktionen sind genau die bb. Zahlenfunktionen.

3. **Wie lautet die Definintion von ES-, SZ- und GS-Funktion?**

   $KON := Q \times D$ ist die Menge der Konfigurationen.

   $ES :\subseteq KON \to KON$:

$$ES(q,d) := \begin{cases} (q', f(d)) & \text{falls } \sigma(q) = (f, q') \text{ und } d \in Def(f) \\ (q', d) & \text{falls } \sigma(q) = (S, q', q'') \text{ und } d \in S \\ (q'', d) & \text{falls } \sigma(q) = (S, q', q'') \text{ und } d \notin S \\ div & \text{sonst} \end{cases}$$

   $SZ :\subseteq KON \to \mathbb{N}$:

$$SZ(q,d) := \begin{cases} \text{div, falls für kein } i \in \mathbb{N}\ ES^i(q,d) \text{ eine Endkonfig. ist} \\ \text{dasjenige } i \text{ so dass } ES^i(q,d) \text{ eine Endkonfig. ist, sonst} \end{cases}$$

   $GS :\subseteq KON \to KON$:

$$GS(q,d) := \begin{cases} ES^{SZ(q,d)}(q,d) & \text{falls } (q,d) \in Def(SZ) \\ div & \text{sonst.} \end{cases}$$

4. **Wie ist die Bandmaschine definiert?**

   Sei Σ ein Alphabet und sei $k \in \mathbb{N}$. Eine k-stellige Bandmaschine über Σ ist eine Maschine

$$M = (F, (\Sigma^*)^k, \Sigma^*, EC^{(k)}, AC)$$

mit einer Datenmenge D_Γ, so dass (1)-(4) gelten:

(1) $D_\Gamma = \{\beta \in \Gamma^{\mathbb{Z}} | \beta(i) \neq B$ nur für endliche viele $i \in \mathbb{Z}\}$, wobei Γ ein Alphabet mit $\Sigma \subseteq \Gamma$ und $B \in \Gamma \backslash \Sigma$ ist (das Band- oder Arbeitsalphabet von M).

(2) $EC^{(k)} : (\Sigma^*)^k \to D$ ist definiert durch

$$EC^{(k)}(x_1, \ldots, x_k) := (\epsilon, B, x_1 B x_2 \ldots B x_k).$$

(3) $AC(u, a, v) :=$ das längste Suffix y von u mit $y \in \Sigma^*$.

(4) Jeder Funktionsbefehl des Flussdiagramms hat die Form "L", "R" oder "a" ($a \in \Gamma$) und jeder Testbefehl die Form "a?" ($a \in \Gamma$).

5. **Welche Funktionen werden von einer Bandmaschine berechnet?**

   Mit einer Bandmaschine werden Wortfunktionen berechnet. Dabei ist ein Wort über Σ eine endliche Folge von Elementen aus Σ.

6. **Wie hängen Zahlen- und Wortfunktionen zusammen?**

   Über eine Standardnummerierung.

   Es sei Σ ein Alphabet, $n := card(\Sigma) \geq 1$, mit einer Ordnungsfunktion $a : \{1, \ldots, n\} \to \Sigma$. Es sei $\sigma : \Sigma^* \to \mathbb{N}$ definiert durch

$$\sigma(\epsilon) := 0$$
$$\sigma(a_{i_k} a_{i_{k-1}} \ldots a_{i_1} a_{i_0}) := i_k n^k + i_{k-1} n^{k-1} + \ldots + i_1 n + i_0$$

   für alle $k \in \mathbb{N}, i_0, \ldots, i_k \in \{1, \ldots, n\}$. Dann heißt $\nu_\Sigma : \mathbb{N} \to \Sigma^*$, definiert durch $\nu_\Sigma := \sigma^{-1}$, eine Standardnummerierung von Σ^*.

7. **ν_Σ ist bijektiv. φ auch? Warum nicht?**

   Weil $\nu_P : \mathbb{N} \to BP$ nicht bijektiv ist und φ u.a. aus dieser Funktion zusammengesetzt ist.

$$\nu_P(i) = \begin{cases} \nu_\Omega(i) & \text{falls } \nu_\Omega(i) \in BP \\ \text{“(:B,,)”} & \text{sonst} \end{cases}$$

   Wobei ν_Ω die aus der Ordnungsfunktion $a : 1, \ldots, 8 \to \Omega$ abgleitete Standardnummerierung von Ω^* ist.

   $(a_1|a_2|\ldots|a_8) := (1|B|(|)| : |, |R|L)$

   $BP := \{z_1 z_2 \ldots z_k | k \geq 1$ und $z_1 \ldots z_k \in BB\}$ ist die Menge der Bandprogramme.

   $BB := \{\text{“}(1^m : \beta, 1^n)\text{''} | \beta \in \{B, 1, R, L\}; m, n \in \mathbb{N}\} \cup \{\text{“}(1^m : \beta, 1^k, 1^n)\text{''} | \beta \in \{B, 1\}; k, m, n \in \mathbb{N}\}$

   Diese Funktion ist nicht linkseindeutig (aus $x \rho y$ und $x' \rho y$ folgt nicht $x = x'$, da $(: B, ,)$ aus mehreren Nummern erzeugt werden kann. Also nicht bijektiv, damit auch φ nicht bijektiv.

8. **Wenn nun ν_P bijektiv wäre, könnte man dann φ bijektiv machen?**

   Korrektheits/Äquivalenzproblem:

   $d :\subseteq \mathbb{N} \to \mathbb{N}$, $d(i) = div$ für alle $i \in \mathbb{N}$.

   1. Für jedes $f \in P^{(1)} \backslash \{d\}$ sind die Mengen $M_f := \{i | \varphi_i = f\}$ und $\mathbb{N} \backslash M_f = \{i | \varphi_i \neq f\}$ nicht r.a.

   2. Die Mengen $\ddot{A}q := \{(i,j) | \varphi_i = \varphi_j\}$ und $\mathbb{N}^2 \backslash \ddot{A}q = \{i,j\} | \varphi_i \neq \varphi_j\}$ sind nicht r.a.

9. **Was ist eine rekursive Funktion?**

   Dies sind die berechenbaren Funktionen. Dabei gibt es *partiell-rekursive* und *total-rekursive* Funktionen.

   Für $k \in \mathbb{N}$ sei

$$
\begin{aligned}
P^{(k)} &:= \{f :\subseteq \mathbb{N}^k \to \mathbb{N} | \text{ f bb.}\} \\
R^{(k)} &:= \{f : \mathbb{N}^k \to \mathbb{N} | \text{ f bb.}\} \\
P &:= \bigcup_{k \in \mathbb{N}} P^{(k)} \\
R &:= \bigcup_{k \in \mathbb{N}} R^{(k)}
\end{aligned}
$$

10. **Wann ist eine Menge rekursiv?**

    Eine Teilmenge $A \subseteq \mathbb{N}^k$ ($k \in \mathbb{N}$) heißt rekursiv (entscheidbar), gdw. die Funktion $cf_A : \mathbb{N}^k \to \mathbb{N}$ bb. ist.

$$
cf_A(x) := \begin{cases} 1 & \text{falls } x \in A \\ 0 & \text{sonst} \end{cases}
$$

    $A \subseteq \mathbb{N}^k$ ist rekursiv, gdw. es eine totale bb. Funktion $f \in R^{(k)}$ gibt mit $A = f^{-1}\{0\}$.

11. **Wann ist eine Menge r.a.?**

    Eine Teilmenge $A \subseteq \mathbb{N}$ heißt r.a., gdw. die Funktion $df_A : \mathbb{N} \to \mathbb{N}$ bb. ist.

$$
df_A(x) := \begin{cases} 1 & \text{falls } x \in A \\ div & \text{sonst} \end{cases}
$$

    Eine Teilmenge $A \subseteq \mathbb{N}^k$ heißt r.a., gdw. es eine partielle bb. Funktion $f :\subseteq \mathbb{N}^k \to \mathbb{N}$ gibt mit $A = Def(f)$.

12. **Wie sind die μ-rekursiven Funktionen definiert?**

    Eine Funktion $f :\subseteq \mathbb{N}^k \to \mathbb{N}$ heißt μ-rekursiv, gdw. sie sich in endlich vielen Schritten aus der Menge $Gr := \{\tilde{0}, Z, S\} \cup \{pr_i^{(k)} | i, k \in \mathbb{N}, 1 \leq i \leq k\}$ durch wiederholtes Anwenden der Operatoren Sub, Prk und $\tilde{\mu}$ erzeugen lässt.

13. **Warum erhält man bei den μ-rekursiven Funktion die gleichen Funktionen wie bei den RM?**

    Die μ-rekursiven Funktionen sind genau die berechenbaren Zahlenfunktionen. Diese wurden genau über die Registermaschinen definiert (eine Funktion ist bb., wenn eine RM mit $f_M = f$ exisitert).

14. **Genügt es, den $\tilde{\mu}$-Operator auf totale Funktionen anzuwenden?**

    Ja. Nach der Definition des $\tilde{\mu}$-Operators, wird dieser auf totale Funktionen angewendet:

    Es sei $h : \mathbb{N}^{k+1} \to \mathbb{N}$ eine totale Funktion. Dann sei die Funktion $\tilde{\mu}(h) :\subseteq \mathbb{N}^k \to \mathbb{N}$ definiert durch

    $$\tilde{\mu}(h)(\overline{x}) = \mu y[h(\overline{x}, y) = 0]$$

    für alle $\overline{x} \in \mathbb{N}^k$. Man sagt, $\tilde{\mu}$ entsteht aus h durch μ-Rekursion.

    Nebenbei:

    $$\mu x[Q(x)] := \begin{cases} min\ M & \text{falls } M := \{k \in \mathbb{N} | Q(k)\} \neq \emptyset \\ div & \text{sonst.} \end{cases}$$

15. **Gibt es Mengen, die rekursiv-aufzählbar (beweisbar), aber nicht rekursiv (entscheidbar) sind?**

    Ja. Grundsätzlich ist eine r.a.-Menge $A \subseteq \mathbb{N}^k$ auch rekursiv gdw. $\mathbb{N}^k \backslash A$ r.a. ist. Umgekehrt ist eine r.a. Menge $A \subseteq \mathbb{N}^k$ nicht rekursiv, wenn die Menge $\mathbb{N}^k \backslash A$ nicht r.a. ist.

    Als Beispiel sei hier das Selbstanwendbarkeitsproblem genannt. Die Menge $K_\varphi := \{i \in \mathbb{N} | \varphi_i(i) \text{ existiert}\}$ ist r.a., aber nicht rekursiv. Das Komplement $\mathbb{N}^k \backslash K_\varphi$ ist nicht r.a.

16. **Beschreiben Sie das Selbstanwendbarkeitsproblem**

    Es geht grundsätzlich darum, ob die i-te Bandmaschine bei der Eingabe von i hält oder nicht. Die Menge K_φ ist also nicht rekursiv.

17. **Beweis, dass K_φ r.a. ist, aber nicht rekursiv**

    vgl. auch S. 161, KE6

    Es sei $h :\subseteq \mathbb{N} \to \mathbb{N}$ definiert durch $h(i) := u_\varphi(i, i)$ für alle $i \in \mathbb{N}$. Dann ist h berechenbar, und es gilt $K_\varphi = Def(h)$. Also ist auch K_φ r.a.

    (nach Definition der r.a. Mengen: Eine Menge $A \subseteq \mathbb{N}^k$ heißt r.a., gdw. es eine partielle bb. Funktion $f :\subseteq \mathbb{N}^k \to \mathbb{N}$ gibt mit $A = Def(f)$.)

    Nun muss gezeigt werden, dass $\mathbb{N} \backslash K_\varphi$ nicht r.a. ist. Beweis durch Diagonalisierung über die Folge $(Def(\varphi_0), Def(\varphi_1), \ldots)$ aller r.a. Teilmengen von $\mathbb{N}$:

    Sei $i \in \mathbb{N}$ eine beliebige natürliche Zahl. Dann gilt nach Definition von K_φ

    $$i \in \mathbb{N} \backslash K_\varphi \Longleftrightarrow i \notin K_\varphi \Longleftrightarrow i \notin Def(\varphi_i)$$

Wegen $A \neq B$ gdw. $(\exists k \in \mathbb{N})(k \in A \iff k \notin B)$ folgt daraus: $\mathbb{N} \backslash K_\varphi \neq Def(\varphi_i)$. Die Mengen $\mathbb{N} \backslash K_\varphi$ und $Def(\varphi_i)$ unterscheiden sich an der Stelle i.

Da $i \in \mathbb{N}$ beliebig gewählt war, gilt damit

$$\mathbb{N} \backslash K_\varphi \neq Def(\varphi_0), \mathbb{N} \backslash K_\varphi \neq Def(\varphi_1), \mathbb{N} \backslash K_\varphi \neq Def(\varphi_1), \ldots$$

Da die Liste $(Def(\varphi_0), Def(\varphi_1), \ldots)$ alle r.a. Teilmengen von $\mathbb{N}$ enthält, kann $\mathbb{N} \backslash K_\varphi$ nicht r.a. sein. Damit ist K_φ nicht rekursiv.

18. **Beschreiben Sie das utm-Theorem**

    Die Funktion $u_\varphi :\subseteq \mathbb{N}^2 \to \mathbb{N}$ sei definiert durch

    $$u_\varphi(i, x) := \varphi_i(x) \text{ für alle } i, x \in \mathbb{N}$$

    Dann ist u_φ bb. u_φ nennt man die universelle Funktion von φ.

19. **Beweis, dass $u_\varphi(i)$ bb. ist**

    Vorweg: Die Funktion $h : \mathbb{N}^3 \to \mathbb{N}$ ist bb.

    $$h(i, n, t) = \begin{cases} 1 + \varphi_i(n) & \text{falls } \Phi_i(n) \text{ existiert und } \Phi_i(n) \leq t \\ 0 & \text{sonst.} \end{cases}$$

    Folgendes Flussdiagramm einer verallgemeinerten Registermaschine berechnet dann die Funktion u_φ

    $$\begin{aligned} &1 : R_0 := h(R_1, R_2, R_3), 2; \\ &2 : R_0 = 0, 3, 4; \\ &3 : R_3 := R_3 + 1, 1; \\ &4 : R_0 := R_0 - 1, 5; \\ &5 \text{ HALT} \end{aligned}$$

    Dieses Programm sucht die kleinste Zahl t, für die $h(i, x, t) \neq 0$ gilt. Vom ermittelten Wert muss dann nur noch 1 abgezogen zu werden.

20. **Beschreiben Sie das smn-Theorem**

    Es sei $f \in P^{(2)}$ eine beliebige zweistellige bb. Funktion. Dann gibt es eine totalrekursive Funktion $r \in R^{(1)}$ mit

    $$f(i, j) = \varphi_{r(i)}(j) \text{ für alle } i, j \in \mathbb{N}.$$

    Dabei lässt sich die Funktion $f \in P^{(2)}$ als universelle Funktion einer Programmiersprache $\psi : \mathbb{N} \to F(F \subseteq P^{(1)})$ auffassen: $\psi(i)(x) := f(i, x)$. Nach obigen Theorem gilt dann $\psi(i) = \varphi(r(i))$ für alle $i \in \mathbb{N}$. Die bb. Funktion r übersetzt jedes Programm der Programmiersprache ψ in ein äquivalentes der Programmiersprache φ.

21. **Erläutern Sie die Standardnummerierung φ**

φ ist die Standardnummerierung von $P^{(1)}$. Dabei ist diese wie folgt aufgebaut:

$\varphi : \mathbb{N} \xrightarrow{\nu_P} BP \xrightarrow{\nu_M} BM \xrightarrow{\xi} P^{(1)}$

$$\begin{aligned} BP &: \text{ Bandprogramme} \\ BM &: \text{ normierte Bandmaschine} \\ P^{(1)} &: \{f :\subseteq \mathbb{N} \to \mathbb{N}, \text{f bb.}\} \end{aligned}$$

$\nu_P : \mathbb{N} \to BP$ Nummer zu Bandprogramm

$\nu_M : BP \to BM$: Liefert BM zu Bandprogramm

$\xi : BM \to P^{(1)}$ $\xi(M) := \iota^{-1} f_M \iota,$ wobei $\iota : \mathbb{N} \to \{1\}^*, \iota(i) = 1^i$.

22. **Defintion der Band-/Zeitkomplexitätsklassen**

Sei $f : \mathbb{N} \to \mathbb{N}$ eine (totale) Funktion, Σ ein Alphabet.

$$\begin{aligned} ZEIT_\Sigma(f) &:= \{L_M | M \text{ TM über } \Sigma, \tilde{t}_M \in O(f)\} \\ FZEIT_\Sigma(f) &:= \{f_M | M \text{ TM über } \Sigma, \tilde{t}_M \in O(f)\} \\ BAND_\Sigma(f) &:= \{L_M | M \text{ TM über } \Sigma, \tilde{s}_M \in O(f)\} \\ FBAND_\Sigma(f) &:= \{f_M | M \text{ TM über } \Sigma, \tilde{s}_M \in O(f)\} \\ ZEIT(f) &:= \bigcup_\Sigma ZEIT_\Sigma(f) \\ FZEIT(f) &:= \bigcup_\Sigma FZEIT_\Sigma(f) \\ BAND(f) &:= \bigcup_\Sigma BAND_\Sigma(f) \\ FBAND(f) &:= \bigcup_\Sigma FBAND_\Sigma(f) \end{aligned}$$

$ZEIT(f)$ ist die Menge der in $O(f)$ Zeit entscheidbaren Sprachen. $FZEIT(f)$ die Menge der in $O(f)$ entscheidbaren Funktionen (entsprechend für Band).

$$\begin{aligned} \tilde{t}_M(n) &:= max\{t_M(x) | x \in \Sigma^n\} \\ \tilde{s}_M(n) &:= max\{s_M(x) | x \in \Sigma^n\} \end{aligned}$$

23. **Was muss erfüllt sein, damit zwei Funktionen unterschiedliche Zeit-/Bandkomplexitätsklassen haben?**

Vorweg: band- und zeitkontruierbar:

$f : \mathbb{N} \to \mathbb{N}$ heißt bandkonstruierbar, falls es eine TM M gibt, so dass

$$\forall n. f_M(0^n) = 0^{f(n)}$$
$$\tilde{s}_M \in O(f).$$

$f : \mathbb{N} \to \mathbb{N}$ heißt zeitkonstruierbar, falls es eine TM M gibt, so dass

$$\forall n. f_M(0^n) = 0^{f(n)}$$
$$\tilde{t}_M \in O(f).$$

Zeitseparationssatz: Seien $f, g : \mathbb{N} \to \mathbb{N}$, g zeitkonstruierbar, $n \in O(g)$, und $g \notin O(f(n) \cdot log f(n))$.

Dann ist $ZEIT(g) \nsubseteq ZEIT(f)$.

Zeithierarchiesatz: Sei $g : \mathbb{N} \to \mathbb{N}$ zeitkonstruierbar, $n \in O(g)$, $f : \mathbb{N} \to \mathbb{N}$, $f \in O(g)$, $g \notin O(f(n) \cdot log f(n))$.

Dann ist $ZEIT(f) \subsetneq ZEIT(g)$.

Bandseparationssatz: Sei $g : \mathbb{N} \to \mathbb{N}$ bandkonstruierbar, $log \in O(g)$, $f : \mathbb{N} \to \mathbb{N}$.

Dann gilt $BAND(g) \nsubseteq BAND(f) \Leftrightarrow g \in O(f)$.

Bandhierarchiesatz: Sei $log \in O(g)$, g bandkonstruierbar, $f \in O(g)$, $g \notin O(f)$.

Dann ist $BAND(f) \subsetneq BAND(g)$

24. **Welche Sprache wird von einer kontrollband-Turingmaschine erkannt?**

    Die von einer KTM M erkannte Sprache $L_M \subseteq \Sigma^*$ ist definiert durch

    $$L_M = \{x \in \Sigma^* | \exists y \in \Delta^*. \tau_M(x, y) = 1\}$$

    Wobei Δ ein Hilfsalphabet für die Kontrolleingabe (=Beweis) y ist ($\$, \not\epsilon \notin \Delta$). Die KTM schreibt mittels EC die Kontrolleingabe y auf Band 0, Band 1 enthält die zu beweisende Eingabe. $\tau_M :\subseteq \Sigma^* \times \Delta^* \to \{1, 2\}$:

    $$\tau_M(x, y) = \begin{cases} 1 & \text{falls } \exists d \in D.GS(q_0, ((\epsilon, \not\epsilon, y\$), (\epsilon, \not\epsilon, x\$), (\epsilon, B, \epsilon), \ldots)) = (q_+, d) \\ 2 & \text{falls } \exists d \in D.GS(q_0, ((\epsilon, \not\epsilon, y\$), (\epsilon, \not\epsilon, x\$), (\epsilon, B, \epsilon), \ldots)) = (q, d), q \neq q_+ \\ div & \text{sonst} \end{cases}$$

    Endet die KTM mit 1 ist der Beweis erbracht, dass $x \in L$. Falls die Maschine anders endet, ist nur erbracht, das y kein Beweis für $x \in L$ ist. Keine Aussage über das eigentliche Problem.

25. **Komplexitätsklassen**

Deterministisch:

$$\begin{aligned} P &:= \bigcup_{k\in\mathbb{N}} ZEIT(n^k) \\ LOGSPACE &:= BAND(log) \\ PSPACE &:= \bigcup_{k\in\mathbb{N}} BAND(n^k) \end{aligned}$$

Nichtdeterministisch:

$$\begin{aligned} NP &:= \bigcup_{k\in\mathbb{N}} NZEIT(n^k) \\ NLOGSPACE &:= NBAND(log) \\ NPSPACE &:= \bigcup_{k\in\mathbb{N}} NBAND(n^k) \end{aligned}$$

$$\begin{aligned} NZEIT(f) &:= \{L_M | M \text{ KTM }, \tilde{t}_M \in O(f)\} \\ NBAND(f) &:= \{L_M | M \text{ KTM }, \tilde{s}_M \in O(f)\} \end{aligned}$$

26. **Was bedeutet NP-vollständig?**

    Vorweg: Reduzierbarkeit

    $A, B \in \mathbb{N}$, $A \leq B \Leftrightarrow \exists f \in R^{(1)} \wedge x \in A \Leftrightarrow f(x) \in B$. Daraus lässt sich folgern:

    $$\begin{aligned} B \text{ r.a } &\Rightarrow A \text{ r.a.} \\ B \text{ entscheidbar } &\Rightarrow A \text{ entscheidbar} \end{aligned}$$

    Vorweg: Polynomiell reduzierbar

    L heißt polynomiell reduzierbar auf M ($L \leq_{pol} M$), gdw.

    $\exists f \in$ FP.$\forall x \in \Sigma^*.x \in L \Leftrightarrow f(x) \in M$.

    M heißt **NP-hart**, gdw. $\forall L \in$ NP.$L \leq_{pol} M$

    M heißt **NP-vollständig**, gdw. M NP-hart und $M \in NP$.

    Die Folge daraus: Löst man für ein NP-vollständiges M die P-NP Frage, sind alle anderen NP-vollständigen Probleme auch gelöst.

27. **Was ist das P-NP Problem und warum ist dieses interessant?**

    Grundsätzlich gilt: $P \subseteq NP \subseteq EXP$ und $P \subsetneq EXP$.

    Man weiß bis heute nicht, ob $P = NP$ gilt oder nicht. Da viele natürliche Probleme in NP liegen und NP-vollständig sind, wäre eine ist es naürlich interessant, was NP nun ist...

28. **Beweisidee, das SAT NP-vollständig ist**

Zwei Schritte notwendig: erstmal, das SAT in NP liegt, dann, dass SAT NP-hart ist.

NP:

(1) Prüfe, ob $x \in \Pi^* \cap Bild(\gamma_{WFF})$.

(2) Bestimme $N = max\{n|$ in x kommt das Teilwort $|^n$ vor$\}$

(3) Rate $\beta(1), \beta(2), \ldots, \beta(N)$

(4) Ersetze jedes maximale Teilwort $|^m$ in x durch $\beta(m)$; y sei das resultierende Wort.

(5) Solange $y \notin \{0,1\}$, ersetze Teilwort $\neg 0$ durch 1 und $\neg 1$ durch 0.

$$(a \vee b) \text{ durch } \begin{cases} 0 & \text{falls } a = b = 0 \\ 1 & \text{sonst} \end{cases}$$

$$(a \wedge b) \text{ durch } \begin{cases} 1 & \text{falls } a = b = 1 \\ 0 & \text{sonst} \end{cases}$$

(6) Falls $y = 1$ akzeptiere, sonst verwerfe.

Durch Induktion über den Aufbau von WFF zeigt man leicht, dass nach maximal $lg(x)$ Ersetzungsschritten (5) mit $y \in \{0,1\}$ endet. Ebenfalls durch Induktion zeigt man, dass bei Ersetzen von y durch y' in (5) stets gilt: $val(y, \beta) = val(y', \beta)$. In (5) wird also der Wert der in (3), (4) belegten Formel $\gamma_{WFF}^{-1}(x)$ berechnet. Die Berechnung kann in $O(n^2)$ ausgeführt werden. Damit gilt

$$\begin{aligned} x \in SAT &\Rightarrow \exists \beta . val(\gamma_{WFF}^{-1}(x), \beta) = 1 \\ &\Rightarrow \text{Falls M in (3) genau } \beta(1), \ldots, \beta(N) \text{ rät, wird x akzeptiert} \\ &\Rightarrow x \in L_M \end{aligned}$$

Wenn nun $x \in L_M$. Dann gibt es eine Berechnung, in der x akzeptiert wird. Seien $\beta_1, \ldots, \beta_N$ die in (3) geratenen Werte für $v_1, \ldots, v_N$. Dann gilt für jede Belegung β mit $\beta(i) = \beta_i (1 \leq i \leq N)$:

$$val(\gamma_{WFF}^{-1}(x), \beta) = 1, \text{ also } x \in SAT.$$

Für 3SAT muss nur zusätzlich getestet werden, dass γ_{WFF}^{-1} eine 3-KNF-Formel ist.

NP-hart:

Hier muss nur gezeigt werden, $3SAT \leq_{pol} SAT$. Ein f:

$$f(x) = \begin{cases} x & \text{falls } \gamma_{WFF}^{-1} \text{ in 3-KNF} \\ \gamma_{WFF}(0) & \text{sonst} \end{cases}$$

Dann gilt: $x \in 3SAT \Rightarrow f(x) = x \in SAT$.

Sei $x \notin 3SAT$:

Fall 1: $x \notin Bild(\gamma_{WFF}) \Rightarrow \gamma_{WFF}^{-1} f(x) = 0 \Rightarrow f(x) \notin SAT$.

Fall 2: $\gamma_{WFF}^{-1}(x)$ nicht in 3-KNF $\Rightarrow \gamma_{WFF}^{-1} f(x) = 0 \Rightarrow f(x) \notin SAT$

Fall 3: $\gamma_{WFF}^{-1}(x)$ in 3-KNF, aber x nicht erfüllbar $\Rightarrow f(x) = x$ und damit $\gamma_{WFF}^{-1} f(x)$ nicht erfüllbar.

3D-DOMINO lässt sich wiederum auf 3SAT reduzieren, für welches bewiesen wurden, dass es NP-hart ist.

29. **Kann man beweisen, dass eine Sprache nicht NP-vollständig ist?**

    Nein, damit wäre das P-NP Problem gelöst.

30. **Welche Sprache wird von einem determinierten Kellerautomaten erkannt?**

    Eine Teilmenge der kontextfreie Sprachen (die deterministisch kontextfreien Sprachen). Sprachen wie $\{a^n b^m c^m | n, m \geq 1\} \cup \{a^n b^m d^n | n, m \geq 1\}$ können von einem determinierten Kellerautomaten nicht erkannt werden.

    Kontextfreie Sprachen werden durch kontextfreie Grammatiken erzeugt. Kontextfreie Grammatiken haben die Form $A \to w$, mit $A \in \Pi$ und $w \in (\Sigma \cup \Pi)^*$.

31. **Ist jede mit einem Kellerautomaten erkannte Sprache kontextfrei? Warum?**

    Ja. Der Kellerautomat für $L(G)$ simuliert (nichtdeterministisch) eine Linksableitung.

    Zu jedem normierten Kellerautomaten A mit nur einem Zustand gibt es eine kontextfreie Grammatik G mit $L(G) = N(A)$.

    Ein Kellerautomat $A = (\Sigma, \Gamma, \$, Q, q_0, Q_f, \delta)$ heißt normiert, wenn für alle $p, q \in Q, C \in \Gamma, W \in \Gamma^*$ gilt:

    $$\exists a.((p, a, C), (N, W, q)) \in \delta \Rightarrow \forall a \in \Sigma \cup \{\$\}.((p, a, C), (N, W, q)) \in \delta.$$

32. **Ist jede kontextfreie Sprache deterministisch? Warum nicht?**

    Von determinierten Kellerautomaten erkannte Sprachen sind eine echte Teilmenge der von nichtdeterminierten Kellerautomaten erkannten Wortmenge.

    Die deterministisch kontextfreien Sprachen sind unter Komplement abgeschlossen, dies gilt für die kontextfreien Sprachen nicht.

    Für deterministische Sprachen gilt also $L \subseteq \Sigma^*$ deterministisch kontextfrei, so auch $\Sigma^* \backslash L$.

    Beispiel: $\Sigma = \{a, b, c\}, L_1 = \{a^n b^n c^m | n, m \in \mathbb{N}\}, L_2 = \{a^n b^m c^m | n, m \in \mathbb{N}\}$.

    $\overline{L} = \Sigma^* \backslash L$.

    L_1 und L_2 sind deterministisch kontextfrei, somit auch $\overline{L_1}$ und $\overline{L_2}$.

    Sei $L = \overline{L_1} \cup \overline{L_2}$. L ist kontextfrei (Vereinigung).

Wäre L deterministisch, so auch $\overline{L} = \overline{\overline{L_1} \cup \overline{L_2}} = L_1 \cap L_2 = \{a^n b^n c^n | n \in \mathbb{N}\}$. $\overline{L}$ ist aber nicht kontextfrei! (Pumping Lemma f. kfS)

33. **Gibt es einen Unterschied zwischen den deterministisch erkannten und den nichtdeterministisch erkannten kontextfreien Sprachen? Sind sie gleich?**

    Ein (nichtdeterministischer) Kellerautomat erkennt jede kfS. Diese muss aber nicht deterministisch sein. Man kann einen deterministischen Kellerautomaten aus dem nichtdeterministischen konstruieren. Dieser erkennt aber nur eine echte Teilmenge der von dem nichtdeterministischen KA erkannten Wortmenge.

34. **Wie sind die Abschlusseigenschaften der deterministischen und der nichtdeterministischen erkannten kontextfreien Sprachen bzgl. Vereinigung, Schnitt und Komplement?**

    "+": abgeschlossen, "-": nicht abgeschlossen

Operation	**-**	**kontextfrei**	**deterministisch**
Vereinigung	$L_1 \cup L_2$	+	-
Durchschnitt	$L_1 \cap L_2$	-	-
Komplement	$\Sigma^* \backslash L$	-	+
Konkatenation	$L_1 \cdot L_2$	+	-
Stern	L^*	+	-
Schnitt mit regulären Mengen	$L_1 \cap R$	+	+

    Dagegen sind reguläre Mengen unter Vereinigung, Komplexprodukt, Stern, Durchschnitt, Mengendifferenz, Komplement und Spiegelung abgeschlossen!

35. **Wie beweist man, dass eine Sprache nicht kontextfrei/regulär ist?**

    Jeweils mittels Pumping Lemma für kfS und reguläre Sprachen:

    kfS:

    Sei L kontextfrei. Dann gibt es ein $p \in \mathbb{N}$, so dass man jedes Wort $z \in L$ mit $lg(z) \geq p$ in Teilworte u, v, w, x, y zerlegen kann, dass gilt:

$$z = uvwxy,$$
$$vx \neq \epsilon, \quad lg(vwx) \leq p,$$
$$\forall (i) \geq 0. uv^i wx^i y \in L.$$

    Ist $L = L(G)$ mit einer Chomsky Grammatik $G = (\Pi, \Sigma, R, S)$, so gilt die Aussage für jedes $p \geq 2^{\#\Pi}$.

    reguläre Sprachen:

    Sei $L \subseteq \Sigma^*$ regulär. Dann gilt:

    Es gibt ein $n \in \mathbb{N}$, so dass für alle $t, \bar{t}, z \in \Sigma^*$ mit $tz\bar{t} \in L$ und $lg(z) = n$ gilt:

$$\exists u, v, w \in \Sigma^*. (z = uvw \wedge v \neq \epsilon \wedge \forall i \geq 0. tuv^i w\bar{t} \in L).$$

36. **Sind die in Polynomialzeit deterministisch und nichtdeterministisch erkannten Sprachen gleich mächtig?**

    Das ist das P-NP Problem.

$$\bigcup_{\infty} ZEIT(n^k) \subseteq \bigcup_{\infty} NZEIT(n^k).$$