lessons learned TINF

Aufgabentyp: Beispiel für eine ∞-Folge $(x_n)_{n\geq 1}$ paarweise versch. Wörter, welche KK-Bedingung erfüllt oder Beweise Inexistenz

Lösungsstrategie:

$K(x_n) \leq \lceil \log_2(\alpha(|x_n|)) \rceil + c$

$K(x_n) \leq \lceil \log_2(n+1) \rceil + c$ nach Satz aus VL

$\Rightarrow \alpha(|x_n|) = n+1$ $\quad n = \log_2 |x_n|$

$\Rightarrow |x_n| = \alpha^{-1}(n+1)$ $\quad \rightarrow 2^n = |x_n|$

Benutze $\lceil \log_2(n+1) \rceil + c \leq \log_2(n) + \underbrace{2 + c}_{=:d}$ ggf

Falls $K(x_n) \leq \log_2(n^{1/k}) + c$ $\quad k \geq 2 \rightarrow$ Gegenbeweis

- Beispiel: Finde Länge von $|x_n|$ und entwerfe (arg. p.D.) Prog. welches $\forall n$ gleich bis auf n. Schätze KK ab
- Gegenbeweis: Per Widerspruch
  - Sei $n \in \mathbb{N} - \{0\}$ fix, $t :=$ obere Schranke KK für dieses fixe n.
  - Es gibt höchstens $\sum_{i=0}^{t} 2^i = 2^{t+1} - 1$ Wörter mit bin. Länge höchstens t. (kombinat. Argument)
  - Folglich können höchstens $2^{t+1} - 1$ paarweise versch. Wörter durch Programme, welche bin. Länge höchstens t haben, erzeugt werden. Prog. versch. welche Wörter generieren ist klar.
  - Für n groß genug, was nur von c abhängt, gilt:

$$2^{t+1} - 1 < n \quad \lightning \text{ zu p.D. und } K(x_i) \leq t.$$

Aufgabentyp: ¬(Menge enthält ∞-viele Zahlen zufällig)

Beweisstrategie: Per Widerspruch

- Jede Zahl α_n $\forall n \in \mathbb{N}$ lässt sich mit einem Programm C_n berechnen, das die bin. Kod. von n enthält, α_n berechnet & ausgibt.
  → Finde KK von C_n (Alter: Beschreibe Programm)
- Obere und untere Schranke so abschätzen, dass sie sich direkt miteinander vergleichen
- Ziel: Bekomme ausdruck der Form

$$\log_2(n) \leq c \quad \forall n \in \mathbb{N} \; \lightning$$

Aufgabentyp: x_n n-te kan. Wort in L und KK abschätzen

Beweisstrategie:

- Argumentiere Prog. A_L exist, das (Σ_{bool}, L) löst
- Satz aus Buch: $K(x_n) \leq \lceil \log_2(n+1) \rceil + c$ (jetzt schon auf Schlussform)
- Schätze n durch $|x_n|$ ab (#Wörter der Form $1^i 0^j 1^k$ mit Länge $i+j+k \leq |x_n|$ abschätzen)
- Form der ersten n Wörter in L: $x_1, \dots, x_n$
  mit $i+j+k \leq |x_n| \rightarrow 1 \leq i, j, k \leq |x_n| \Rightarrow$ Max. $|x_n|^3$ Wörter der Länge $i+j+k \leq |x_n|$ (# komb.). Also $n \leq |x_n|^3$ □

lessons learned - TINF

Aufgabentyp: Nichtregularität

Beweismethode:

- **Lemma 3.3:**

  B.P.W. Ang. L sei reg. $\Rightarrow \exists$ EA $M = (Q, \Sigma, \delta, q_0, F)$ s.d. $L(M) = L$. Betrachte die Wörter $(\tilde{w}^{\text{geschickt}}, w^2, \dots, w^{|Q|+1})$.
  Da die Menge der Wörter größer ist als die Anzahl der Zustände $|Q|$ existieren nach dem Schubfachprinzip $i, j \in \{1, \dots, |Q|+1\}$ mit $i < j$ s.d. $\hat{\delta}(q_0, w^i) = \hat{\delta}(q_0, w^j)$
  Nach Lemma 3.3. muss nun $\forall z \in \Sigma^*$: $w^i z \in L \Leftrightarrow w^j z \in L$
  Wähle $z =$ geschickt
  - $w^i z \in L \quad \wedge \quad w^j z \notin L$ ↯

  Somit war unsere Annahme falsch und L ist nicht regulär □

- **Pumping-Lemma:**

  B.P.W. Ang. L sei regulär. Nach dem PL $\exists n_0 \in \mathbb{N}$ s.d. $\forall w \in \Sigma^*$ mit $|w| \geq n_0$ $\exists x, y, z \in \Sigma^*$ s.d. $w = yxz$ und
  i) $|yx| \leq n_0$ ii) $|x| \geq 1$ iii) $M := \{yx^k z \mid k \in \mathbb{N}\} \subseteq L$ oder $M \cap L = \emptyset$
  Wähle das Wort ...
  Da $|w| \geq n_0$ $\exists x, y, z \in \Sigma^*$ s.d. nach PL $x = \dots^{l}$, $y = \dots^{m}$ $z = \dots^{|w| - (m+l)} \dots$ mit $m, l \in \mathbb{N}$ und wegen ii) $l \geq 1$ und wegen i) $m + l \leq n_0$. Insbesondere ist dies die einzig mögliche Zerlegung. Wir zeigen nun, dass iii) nicht hält:
  Fall 1 & 2: Wähle k schlau ... ↯
  Also ist die Annahme falsch und L ist nicht regulär. □

- **KK-Methode:** nur über Σ_{bool} möglich - nicht z.B. bei $\{0, 1, \neq\}$

  BPW. Ang. L sei regulär. Wähle x schlau (typisch einfach Wort in L)
  Betrachte L_x. Setze $m := \dots$. Dann ist das erste/zweite Wort in L_x $\dots^m$. Nach Satz 3.1 existiert eine von m (bzw. n) unabhängige Konstante c, s.d. $K(\dots^m) \leq c + \lceil \log_2(\underset{\text{Kon.}}{\downarrow} + 1) \rceil$ prol.

Da es nur endl. viele Programme ~~der~~ mit ~~kons~~ binärer Länge $\leq c + \lceil \log_2(n+1) \rceil$ gibt, aber ∞-viele Wörter der Form $\ldots^m$, ist dies ein Widerspruch. Also ist die Annahme falsch und L ist nicht regulär. □

lessons learned TINF

Aufgabentyp: min. Hälfte der Wörter zufällig

Beweisstrategie: Kombinat. Argument $\exists$ $K(x) \geq n$ für hälfte aller Wörter

- Def. zufällig $K(x) \geq |x|$.
- $\sum_{i=0}^{n-1} 2^i = 2^{n+1} - 1$ Wörter der Länge höchstens n über Σ_{bool}.
- $\sum_{i=0}^{n-1} 2^i = 2^n - 1$ Wörter der Länge $< n$.

  Also höchstens $2^n - 1$ versch. Binärkod. von Prog. der Länge $< n$
- 2 versch. Wörter von 2 versch. Prog. ausgeben

Also: $\min 2^{n+1} - 1 - (2^n - 1) = 2^n$ Wörter mit KK von min. n, i.e. zufällige Wörter

- $\frac{1}{2}(2^{n+1} - 1) \leq 2^n$. Folgt Beh. □

Aufgabentyp: EA

- mod w $\Rightarrow$ max. w Zst.
- $|w|_a + |w|_b \cdot 2$ $\Rightarrow$ ein b hat doppeltes Gewicht wie ein a
- Klassen mit Bezug auf L definieren (müssen disjunkt sein)
- Modularer Entwurf $\rightarrow$ Produktautomat
- $|w|_a \bmod 3 = |w| \bmod 3 = (|w|_a + |w|_b) \bmod 3$

  $\Leftrightarrow$ $|w|_b \bmod 3 = 0$
- Produktautomat: erst alle Produkte aufschreiben

Aufgabentyp: EA der L akzeptiert hat min. n Zustände

Beweisstrategie: Lemma 3.3

- Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein EA mit $L(M) = L$.

  Seien $x, y \in \Sigma^*$ $x \neq y$, s.d. $\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$.

  Dann gilt $\forall z \in \Sigma^*$: $xz \in L(M) \Leftrightarrow yz \in L(M)$

  Wir konstruieren 5 paarweise versch. Wörter w_i, $1 \leq i \leq 5$, s.d für ein $z \in \Sigma^*$ folgendes gilt: $\neg(w_i z \in L(M) \Leftrightarrow w_j z \in L(M))$ (I) für $i < j$ und $i, j \in \{1, \ldots, 5\}$. Nach obigem Satz müssen dann alle w_i für $1 \leq i \leq 5$ $\hat{\delta}(q_0, w_i) \neq \hat{\delta}(q_0, w_j)$ gelten

Damit haben wir n^V Wörter mit Eigenschaft (I), woraus folgt, dass M min. n Zustände hat.

Aufgabentyp: NEA konstruieren

Konstruktion:

- EA oder NEA für Teilsprachen konstruieren
- r_{decide} (welcher sicher nichtdeterm. ist) benutzen und Übergänge prüfen

Aufgabentyp: L Reg. => L^R Reg.

Beweis:

- Transitionen umdrehen $\forall r \in F$ (Teilautomaten)

  $L(A) = \bigcup_{r \in F} L(A_f)$ => NEA
- Mit Potenzmengenkonstruktion EA erhalten

Aufgabentyp: Sprache nicht rekursir aufzählbar

Beweis: Beweis per Widerspruch

Ang. $L \in L_{RE}$ => $\exists$ TM M : $L(M) = L$

Insb. $\exists i \in \mathbb{N} - \{0\}$: $M = M_i$, also $L = L(M_i)$

Wähle das wort $w_{\alpha(i)}$ - Wort klug & in Abhängigkeit von TM in betrachteteter Sprache wählen

Fall 1: $w_{\alpha(i)} \in L(M_i)$ => . .

Fall 2: $w_{\alpha(i)} \notin L(M_i)$ => . .

Somit war Annahme falsch und $L \notin L_{RE}$ □

Beweis basiert darauf, dass in der Liste (aller möglichen TM in der Def von L) jede TM erfasst ist und daher auch die angenommene TM für die Sprache. z.B. fehlschlagen der Methode für TM wenn sie nur gerade Nummern haben könnten

Lessons learned TINF

- Keep in mind: ggf eleganter einfach transitionen umzuleiten
- Reicht es Eingabe zu transformieren? EE-Reduktion
  wenn Lösung "direkt übernommen"
  Kann Lösung durch Verwendung L_2 besser / direkt bestimmt werden / kann gdw. nicht funktionieren, bzw. wenn $1 \Leftrightarrow 12$?
  → R-Reduktion
- $(L_{empty})^c \leq_{EE} L_{diag}^c$
  $x \neq Kod(M) \Rightarrow$ ~~f(x)~~ $= Kod(B)$ $L(B) = \Sigma_i^*$ i: B ite TM
  $f(x) = x_i$
  $x = Kod(u)$: F generiert $Kod(A)$, A arbeitet unabhängig von ~~auf~~ y wie folgt:
  A generiert sukzessive $(i,j) \in (\mathbb{N}-\{0\})^2$, das i-te Wort w_i
  A simuliert j Schritte von M auf w_i.
  Falls w_i akz. wird, so akz. A Eingabe
  Dann generiert F $i \in \mathbb{N}-\{0\}$ s.d. A i-te TM, generiert i-te Wort w_i und $f(x) = w_i$
  Beweis klar
- L_{RE} via TM

Lemma 2.5: $\forall n \in \mathbb{N} - \{0\}\ \exists w_n \in \Sigma_{bool}^n$ mit $K(w_n) \geq |w_n| = n$

- 2^n Wörter in Σ_{bool}^n
- sei für $1 \leq i \leq 2^n$ $C\text{-}Prog(x_i) \in \{0,1\}^*$ der Maschinencode eines Programms $Prog(x_i)$, das x_i generiert und $K(x_i) = |C\text{-}Prog(x_i)|$ ($Prog(x_i)$ EIN kürzestes Programm das x_i generiert)
- 2^n versch. Maschinencodes $C\text{-}Prog(x_1), \dots, C\text{-}Prog(x_{2^n})$ von kürzesten Programmen für $x_1, \dots, x_{2^n}$, da die Kodierungen $C\text{-}Prog(x_i)$ & $C\text{-}Prog(x_j)$ für $x_i \neq x_j$ versch. sind
- Anzahl aller unterschiedlicher Wörter über Σ_{bool} welche nicht-leer und max. länge $n-1$ haben: $\sum_{i=1}^{n-1} 2^i = 2^n - 2 < 2^n$
- Also können wir nicht 2^n untersch. Maschinencodes der Länge $\leq n-1$ haben $\Rightarrow \exists\, C\text{-}Prog(x_i)$ mit $|C\text{-}Prog(x_i)| \geq n$

LHS = $K(x_i)$ □

Satz 2.2: z_n n-te Wort bzgl. kanon. Ordnung in $(\Sigma_{bool})^*$.

Falls A Programm das (Σ_{bool}, L) löst, dann: $K(z_n) \leq \lceil \log_2(n+1) \rceil + c$

- C_n Programm entwerfen das A_L enthält & z_n generiert

C_n:

```
begin                                   binäre Länge von n: ⌈log2(n+1)⌉
  i=0
  x:=λ
  while i<n do                                              binäre
    begin                                                   Länge von
      Berechne A(x)                                         Cn =: c
      if AL(x) = 1 then                                     außer n
        begin
          i=i+1
          z=x
        end
      x:= Nachfolger von x in kanon. Ordnung
    end
  write(z)
end
```

- C_n generiert nacheinander Wörter aus $(\Sigma_{bool})^*$ in kanon. Ordnung und zählt Anzahl der von A_L akzeptierten Wörter
- Ausgabe ist z_n n-te Wort in L
- alle Programme C_n gleich bis auf Parameter n: $K(z_n) \leq c + \lceil \log_2(n+1) \rceil$ □

Lemma 2.6: $n_1, \dots$ steigende Folge unendlich vieler $n_i \in \mathbb{N}$ mit $K(n_i) \geq \lceil \log_2(n_i) \rceil / 2$. Sei q_i größte Primzahl welche n_i teilt

Dann: $Q = \{q_i \mid i \in \mathbb{N} - \{0\}\}$ unendlich

Beweis: Per Widerspruch

- Angenommen Q ist endlich, sei p_m größte Primzahl in Q
- Eindeutige Primfaktorzerlegung jedes $n_i = p_1^{r_{i,1}} \cdot \dots \cdot p_m^{r_{i,m}}$ mit $r_{i,1}, \dots, r_{i,m} \in \mathbb{N}$
- Also kann man Programm A für bin. Darst. von n_i bei gegeben $r_{i,1}, \dots, r_{i,m}$ entwerfen
- Sei c bin. Länge von A außer der $r_{i,1}, \dots, r_{i,m}$.
- $K(n_i) \leq c + 8 \cdot (\lceil \log(r_{i,1}+1) \rceil + \dots + \lceil \log(r_{i,m}+1) \rceil) \quad \forall i \in \mathbb{N} - \{0\}$
  ↑ Ascii-Codierung damit Darstellung von $r_{i,j}$ eindeutig voneinander trennbar
- $r_{i,j} \leq \log_2(n_i) \quad \forall 1 \leq j \leq m : K(n_i) \leq c + 8 \cdot m \lceil \log_2(\log_2(n_i)+1) \rceil$
- m, c sind Konstanten! Also kann RHS $\geq \lceil \log_2(n_i) \rceil / 2$ nur für endlich viele Zahlen i gelten ↯ zu $K(n_i) \geq$ RHS

Lemma 3.3: $A = (Q, \Sigma, \delta_A, q_0, F)$ EA, $x, y \in \Sigma^* \; x \neq y$ s.d. $(q_0, x) \vdash_A^* (p, \lambda) \wedge (q_0, y) \vdash_A^* (p, \lambda)$ für ein $p \in Q$. (⊛)

Dann: $\forall z \in \Sigma^* \; \exists r \in Q : xz, yz \in K[r]$. Insbesondere

$$xz \in L(A) \iff yz \in L(A)$$

Beweis:

- ⊛ $\Rightarrow (q_0, xz) \vdash_A^* (p, z) \wedge (q_0, yz) \vdash_A^* (p, z) \quad \forall z \in \Sigma^*$
- Wenn Falls $r = \hat{\delta}_A(p, z)$ (also $(p, z) \vdash_A^* (r, \lambda)$), dann:
  $(q_0, xz) \vdash_A^* (p, z) \vdash_A^* (r, \lambda)$
  $(q_0, yz) \vdash_A^* (p, z) \vdash_A^* (r, \lambda)$.
- Wenn Falls $r \in F$, dann $xz, yz \in L(A)$
- Falls $r \notin F$, dann $xz, yz \notin L(A)$
- Wenn

TINF

Satz 3.1: $L \subseteq (\Sigma_{bool})^*$ regulär. $L_x = \{y \in (\Sigma_{bool})^* \mid xy \in L\}$ $\forall x \in (\Sigma_{bool})^*$. Dann $\exists c$ (wobei c von $x \delta y$) s.d. $\forall x, y \in (\Sigma_{bool})^*$

$K(y) \leq \lceil \log_2(n+1) \rceil + c$ für y n-te Wort in L_x

L regulär $\Rightarrow \exists$ EA M s.d. $L(M) = L$

Sei $y \in L_x$ das n-te Wort für ein $x \in (\Sigma_{bool})^*$. $A_{x,y}$ generiert y:

```
A_{x,y}: begin
           z := λ
           i := 0
           while i<n do
             begin
               Simuliere die Arbeit von M aus dem Zustand δ̂(q0, x) auf z;
               if δ(δ̂(q0, x), z) ∈ F then
                 i := i+1
                 y = z  else: z := kanonischer Nachfolger von z
             end
           write(y)
         end
```

- $\forall x, y$ ist $A_{x,y}$ gleich, bis auf n und den Zustand $\hat{\delta}(q_0, x)$.
- $\hat{\delta}(q_0, x)$ können wir mit einem spez. Zeiger auf $\hat{\delta}(q_0, x)$ in die Beschreibung von M einbetten
- Nur $|Q|$ viele Möglichkeiten für Zeiger $\Rightarrow \exists const_M$ welche Länge von der vollst. Beschr. von M mit Zeiger auf einen Zustand beinhaltet.
- Länge der Beschreibung von $A_{x,y}$ außer n, M, $\hat{\delta}(q_0, x)$ ist eine Konstante d bzgl. $x \delta y$. n kann mit $\lceil \log_2(n+1) \rceil$ Bits darstellen. Also

$K(y) \leq \lceil \log_2(n+1) \rceil + const_M + d$.

Lemma 3.4: Pumping-Lemma

Sei $L \subseteq (\Sigma_{\text{klein}})^*$ regulär. Dann $\exists n_0 \in \mathbb{N}$ s.d. $\forall w \in (\Sigma_{\text{klein}})^*$ mit $|w| \geq n_0$ eine Zerlegung $w = yxz$ existiert, s.d.

i) $|yx| \leq n_0$, ii) $|x| \geq 1$, iii) $\underbrace{\{yx^k z \mid k \in \mathbb{N}\}}_{=: M} \subseteq L$ oder $M \cap L = \emptyset$

Beweis: Direkter Beweis

- Sei $L \subseteq \Sigma^*$ regulär. $\exists$ EA $A = (Q, \Sigma, \delta_A, q_0, F)$ s.d. $L(A) = L$
- $n_0 := |Q|$ und $w \in \Sigma^*$ mit $w \geq n_0$, also $w = w_1 \dots w_{n_0} u$, $w_i \in \Sigma$, $u \in \Sigma^*$.
- Betrachte Berechnung $w_1 \dots w_{n_0}$ auf A:
  $(q_0, w_1 \dots w_{n_0}) \vdash_A (q_1, w_2 \dots w_{n_0}) \vdash_A \dots \vdash_A (q_{n_0}, \lambda)$ (I)
- In n_0+1 Konfigurationen kommen n_0+1 Zustände $q_0, \dots, q_{n_0}$ vor.
  Da $|Q| = n_0$ $\exists i, j \in \{0, \dots, n_0\}$ $i < j$ mit $q_i = q_j$
- Also (I):
  $(q_0, w_1 \dots w_{n_0}) \vdash_A^* (q_i, w_{i+1} \dots w_{n_0}) \vdash_A^* (q_i, w_{j+1} \dots w_{n_0}) \vdash_A^* (q_{n_0}, \lambda)$
- $y := w_1 \dots w_i$, $x := w_{i+1} \dots w_j$ ($|x| = j - i \rightarrow x \neq \lambda$), $z := w_{j+1} \dots w_{n_0} u$
- Prüfe i) & iii) direkt
- Prüfe iii): $(q_0, yx) \vdash_A^* (q_i, x) \vdash_A^* (q_i, \lambda)$
  $\Rightarrow (q_i, x^k) \vdash_A^* (q_i, \lambda)$ $\forall k \in \mathbb{N}$
- $(q_0, yx^k z) \vdash_A^* (q_i, x^k z) \vdash_A^* (q_i, z) \vdash_A^* (\hat{\delta}(q_i, z), \lambda)$
  Berechnung von $yx^k z$ $\forall k \in \mathbb{N}$ auf A, landen im selben Zustand
  - Falls $\hat{\delta}(q_i, z) \in F \rightarrow$ A akzeptiert alle $yx^k z$. Falls nicht dann akzeptiert A keins. Also folgt iii) $\square$

Lemma 5.2: $(\mathbb{N}-\{0\}) \times (\mathbb{N}-\{0\})$ ist abzählbar

Beweis:

- Betrachte Matrix $M_{\mathbb{N}\times\mathbb{N}}$, wobei unendl. viele Zeilen & Spalten beginnend mit 1 nummeriert sind
- i-te Zeile & j-te Spalte: $(i,j) \in (\mathbb{N}-\{0\}) \times (\mathbb{N}-\{0\})$
- $M_{\mathbb{N}\times\mathbb{N}}$ enthält klarerweise alle Ele. aus $(\mathbb{N}-\{0\}) \times (\mathbb{N}-\{0\})$
- Lösung: Nebendiagonalen durchnummerieren

  Formal: $(a,b) < (c,d) \iff a+b < c+d$ or $(a+b = c+d \wedge b < d)$

  Nummerierung: $f((a,b)) = \binom{a+b-1}{2} + b$, da

  (a,b) das b-te Element auf der $(a+b-1)$-ten Nebendiag. & auf den ersten $(a+b-2)$ Nebendiag. sind

  $\sum_{i=1}^{a+b-2} i = \binom{a+b-1}{2}$ Elemente
- f Bij. von $(\mathbb{N}-\{0\}) \times (\mathbb{N}-\{0\}) \longrightarrow \mathbb{N}-\{0\}$ □

Satz 5.3: $[0,1]$ ist überabzählbar

Beweis: per Widerspruch

- Angenommen $[0,1]$ abzählbar, also $f: [0,1] \hookrightarrow \mathbb{N}-\{0\}$
- i-te Zahl in $[0,1]$: $a_i = 0,a_{i_1}a_{i_2}\ldots$ mit a_{i_j} aus $\{0,1,\ldots,9\}$ $j \geq 1$ in $\mathbb{N}$, $f(a_i) = i$
- f bestimmt Nummerierung der reellen Zahlen aus $[0,1]$
- Benutze Diagonalisierungsmethode
- Auflistung der reellen Zahlen in $[0,1]$ als Matrix $M = [a_{ij}]_{1 \leq i,j \leq \infty}$
- Konstruiere $c = 0,c_1c_2\ldots$ s.d. $c_i \neq a_{ii} \wedge c_i \notin \{0,9\}$ $\forall i \in \mathbb{N}-\{0\}$
- Wähle $c_i \in \{1,\ldots,8\} - a_{ii}$
- c unterscheidet sich von jeder reellen Zahl ~~aus M~~ von a_i in min. ~~einer~~ einer Ziffer j, i.e. min. in der i-ten Dezimalstelle nach 0, $\forall i \in \mathbb{N}-\{0\}$
- Darstellung von c eindeutig, da $c_i \notin \{0,9\} \Rightarrow c \neq a_i$ $\forall i \in \mathbb{N}-\{0\}$ ↯ □

Satz 5.4: $\mathcal{P}((\Sigma_{bool})^*)$ überabzählbar

Beweis: Direkt

- Wir zeigen $|\mathcal{P}((\Sigma_{bool})^*)| \geq |[0,1]|$
- reelle Zahl in $[0,1]$ binär Darstellen:
  $a = 0, a_1 a_2 \ldots$ $a_i \in \Sigma_{bool}$ stellt $Nummer(a) = \sum_{i \geq 1} a_i \frac{1}{2^i}$ dar
- Falls Zahl mehrere Darstellung hat (gibt höchstens 2 Dast.) so wähle Lexikographisch letzte.
- Definiere $f(a) = \{a_1, a_2 a_3, a_4 a_5 a_6, \ldots, a_{\binom{n}{2}+1} a_{\binom{n}{2}+2} \ldots a_{\binom{n+1}{2}}, \ldots\}$
  $f: [0,1] \to \mathcal{P}((\Sigma_{bool})^*)$.
- $f(a)$ Sprache über (Σ_{bool}) die genau ein Wort der Länge n $\forall n \in \mathbb{N} - \{0\}$ enthält
  $\Rightarrow f(b) \neq f(c)$ wenn sich die zwei Binärdarstellungen $b \& c$ ~~un gen~~ unterscheiden
  $\Rightarrow f$ injektiv $\square$

THINF Sätze

Lemma 5.5: $(L_{diag})^c \notin L_{RE}$.

- $(L_{diag})^c = \{x \in \{0,1\}^* \mid x = w_i$ für ein $i \in \mathbb{N} - \{0\} \wedge$ M_i akzeptiert $w_i\}$

TM D, die $(L_{Diag})^c$ akzeptiert arbeitet wie folgt:

Eingabe: $x \in \{0,1\}^*$

1) Berechne i, s.d. $x = w_i$ in kan. Ord.
2) generiere Kod(M_i), i-te TM M_i
3) Simuliere M_i auf x_i
   - M_i akzeptiert x_i $\Rightarrow$ D akzeptiert eingabe x
   - M_i akz. nicht x_i $\Rightarrow$ D akz. nicht Eingabe x (auch hält) (also M_i hält)
   - M_i hält nicht auf w_i $\Rightarrow$ D simuliert unendlich Lange Arbeit von M_i auf w_i, also hält auch nicht, i.e $x \notin L(D)$

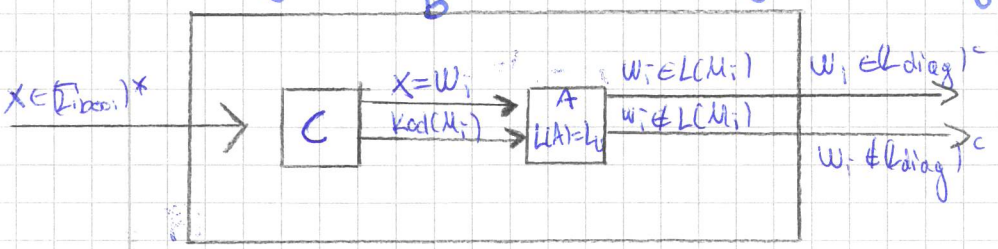
Es ist klar, dass $L(D) = (L_{diag})^c$ □

Satz 5.7: $L_u \notin L_R$

Beweis: 1) Zeige: $(L_{diag})^c \leq_R L_u$

- A Algo. L_u entscheidet (in R, also hält immer!)

B Algo. der mit A als Teilprogramm $(L_{diag})^c$ entscheidet.



- Für Eingabe $x \in (\Sigma_{bool})^*$ berechnet Teilprog. C ein $i \in \mathbb{N} - \{0\}$ s.d. $w_i = x$ und Kod(M_i)
- A bekommt $w_i = x$ & Kod(M_i) als Eingabe
- Die Entscheidung von Kod(a_i) $\neq x$ wird als Resultat von B für Eingabe x übernommen.
- $L(B) = (L_{diag})^c \wedge$ B hält immer, da A a priori immer hält und seine Entscheidung liefert □

Beweis 2: via TM

Z.: $(L_{diag})^c \leq_{EE} L_u$ · M berechnet:

Beschreibe TM M, s.d. $f_M: \{0,1\}^* \to \{0,1,\#\}^*$ und

$x \in (L_{diag})^c \iff f_M(x) \in L_u$ (I)

M arbeitet wie folgt:

Eingabe $x \in \{0,1\}^*$. M berechnet $i \in \mathbb{N}-\{0\}$ s.d. $x = w_i$.

Danach berechnet M $Kod(M_i)$, M_i i-te TM.

M hält mit Inhalt $Kod(M_i) \# x$ auf dem Band. M sim x_i ...

Beh.: (I)

Beweis: $x = w_i \in (L_{diag})^c \iff M_i$ akzeptiert w_i

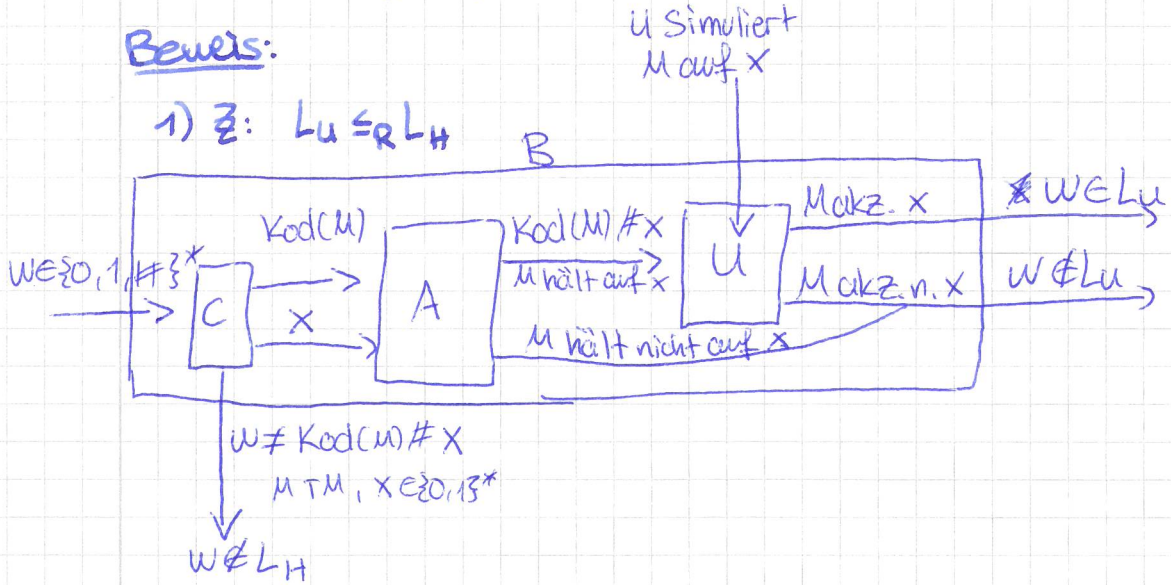
$\iff w_i \in L(M_i)$

$\iff Kod(M_i) \# w_i \in L_u$. □

Satz 5.8: $L_H \notin L_R$

Beweis:

1) Z.: $L_u \leq_R L_H$



Offensichtlich $L(B) = L_u$ und B hält immer, nach Annahme an A

2) Z $L_u \leq_{EE} L_H$

Beschreibe TM M. Für eine Eingabe $w \in \{0,1,\#\}^*$ arbeitet M wie folgt:

Prüfe ob $w = Kod(\bar{M}) \# x$ für $\bar{M}$ TM und $x \in \{0,1\}^*$

1) $w \neq Kod(M) \# x$. M gen. $Kod(M_1)$, die für jede Eingabe in einer Endlosschleife im Zustand q_0 läuft. M hält mit Bandinhalt $M(w) = Kod(M_1) \# x$

2) $w = Kod(M) \# x$. M mod. Kod. von $\bar{M}$ zu folgender TM M_2 mit $L(M_2) = L(\bar{M})$. M_2 arbeitet genau wie $\bar{M}$, nur dass alle Transitionen

zu
~~welche in~~ q_{reject} von $\bar{M}$ zu einem neuen Zustand p umgeleitet werden, s.d. M_2 ~~nie~~ für jede Eingabe in Endlosschleife läuft. Daher führt M_2 $\forall y \notin L(\bar{M}) = L(M_2)$ eine unendl. Berechnung aus. M beendet seine Arbeit mit Bandinhalt $M(w) = Kod(M_2)\#x$.

Beh.: $x \in L_d \iff f(x) \in L_H$

Beweis: "$\Rightarrow$" Direkt "$\Leftarrow$" Kontraposition $\square$

Lemma 5.6: $(L_{empty})^c \in L_{RE}$

Beweis:

1) Für jede NTM $\exists$ TM mit gleicher Lösungsmenge.
Also reicht es z.z., dass NTM M_1 mit $L(M_1) = (L_{empty})^c$
M_1 arbeitet auf Eingabe $x \in \{0,1\}^*$ wie folgt:
i) M_1 überprüft determ. ob $x = Kod(M)$ für M TM
→ Falls nein: M_1 akz. x
ii) Falls $x = Kod(M)$: M_1 wählt nichtdeterministisch (nicht im Sinne $\exists$)
ein $y \in \{0,1\}^*$ und simuliert determ. Berechnung von M auf y.
iii) • M verwirft y → M_1 verwirft x
• M akz. y → M_1 akz. x
• M hält nicht auf y → M_1 hält nicht auf x ($x \notin L(M_1)$)

Nach i) alle Wörter ✓ die keine TM. Falls $L(M) \neq \emptyset \Rightarrow \exists y \in \{0,1\}^*$ mit $y \in L(M)$ ~~$\Rightarrow L(M_1) \neq \emptyset$~~

Beh.: $(L_{empty})^c = L(M_1)$

Beweis: "$\Rightarrow$" Direkt
"$\Leftarrow$" Direkt

2) Konstruiere determ. TM A s.d. $L(A) = (L_{empty})^c$
A arbeitet auf w wie folgt
i) w keine $Kod(M)$ → A akzeptiert w
ii) $w = Kod(M)$: A konstruiert systematisch (wie Diagonale) alle Paare $(i,j) \in (\mathbb{N} - \{0\})^2$.
Für jedes Paar (i,j) gen. A w_i über $\{0,1\}^*$ und simuliert
j Berechnungsschritte von M auf w_i. Falls für ein

(k,L) die TM M w_k in L Schritte akzeptiert, hält M und akz. w. Sonst akzeptiert A Eingabe w nicht, da A ∞-lange arbeitet.

Beh.: $L(A) = (L_{empty})^C$

Beweis: wie üblich □

Lemma 5.8: $L_{H,\lambda} \notin L_R$

Beweis: $\exists$: $L_H \leq_{EE} L_{H,\lambda} = \{Kod(M) \mid M \text{ hält auf } \lambda\}$

Konst. TM M welche eine Abb. $f_M: \{0,1,\#\}^* \to \{0,1\}^*$ berechnet s.d. $x \in L_H \iff f_M(x) \in L_{H,\lambda}$

M arbeitet auf einer Eingabe $w \in \{0,1,\#\}^*$ wie folgt:

1) M prüft ob $w = Kod(\overline{M}) \# x$.

Nein: M hält mit Bandinhalt $M(w) = Kod(B)$ wobei B für jede Eingabe in eine Endlosschleife geht

JA: M hält mit $M(w) = Kod(A)$, wobei A unabhängig von seiner Eingabe $\overline{M}$ auf x simuliert. Falls $\overline{M}$ auf x hält, so akzeptiert A seine Eingabe. Ansonsten geht A auch in Endlosschleife, da $\overline{M}$ ∞-lange arbeitet

Beh.: $x \in L_H \iff f_M(x) \in L_{H,\lambda}$

Beweis: wie üblich

Satz 6.2.: $SPACE(S(n)) = \bigcup_{c \in \mathbb{N}} TIME(c^{S(n)})$ $S(n) \geq \log_2(n)$

Beweis: Sei $L \in SPACE(S(n))$. Also $\exists A$ MTM mit $Space_A(n) \in \mathcal{O}(S(n)) \rightarrow Space_A(n) \leq d \cdot S(n)$ für d const.

- Lemma 6.1 $\Rightarrow \exists M$ 1-B TM (↳ $\{Q, \Sigma, \Gamma, q_{acc}, q_{rej}, q_0\}$) s.d. $L = L(M)$ ∧ $Space_M(n) \leq Space_A(n) \leq d \cdot S(n)$ und M hält immer
- Aus Konfiguration $C := (q, w, i, x, j)$ von M definiere innere Konfig. von C als $In(C) = (q, i, x, j)$ (Teile die sich während Berechnung ändern können) (w: Inhalt Eingabeband)
- Betrachte Mengen $InKonf(n)$ aller möglichen inneren Konfigurationen (q, i, x, j) von M auf allen Eingabewörtern der Länge n

Dann $0 \leq i \leq n+1$, $|x| \leq Space_M(n) \leq d \cdot S(n)$, $0 \leq j \leq Space_M(n) \leq d \cdot S(n)$

- Mit $n+2 \leq 4^{\log_2(n)} \leq 4^{S(n)}$ für $n \geq 2$ und $Space_M(n) \leq |\Gamma|^{Space_M(n)}$ folgt:

$$|InKonf_M(n)| \leq |Q| \cdot (n+2) \cdot |\Gamma|^{Space_M(n)} \cdot Space_M(n)$$
$$\leq \underbrace{(\max\{Q, 4, |\Gamma|\})}_{=:C}^{4d \cdot S(n)}$$

- Folgt: Jede Berechnung von M auf Wort w mit $n = |w|$, länger als $|InKonf_M(n)|$ enthält 2 identische innere Konfigurationen C_i, C_j, $i < j$ und $C_i = C_j$.
- M deterministisch $\Rightarrow$ ∞ Berechnung mit Endlosschleife $C_i, \ldots, C_j$
- Folglich endliche Berechnung von M auf w höchstens Länge $|InKonf_M(n)|$. Also alle Berechnungen von M endlich (nach def. Speicherplatzklassen)
- Folglich M arbeitet in Zeitkomplexität $C^{S(n)}$

Satz 65.: $NTime(s(n)) \subseteq Space(s(n))$ $s(n) \geq \log_2(n)$ + Platzkonst.

Beweis:

Sei $L \in NTime(s(n)) \Rightarrow \exists$ NTM M mit k-Bändern M s.d. $L(M) = L$ ∧ $Time_M(n) \in O(s(n))$.

Setze $r := \max\{|\delta(u)| \mid u \in Q \times (\Sigma \cup \{\vdash, \$\}) \times \Gamma^k\}$

Obere Schranke für # möglicher Aktionen von M aus einer Konfiguration.

Sei $T_{M,x}$ Berechnungsbaum von M auf $x \in \Sigma^*$, wobei die n.det. Entsch. von M auf jedem Argument mit $1, \dots, r$ nummeriert. Dann jede Kante von $T_{M,x}$ entsprechende Nummer zuordnen.

Betrachte nun Berechnung als Folge von Kanten statt Knoten.

Dann kann man jeder Berechnung $|C| = L$ ein Wort $z = z_1 \dots z_L$ mit $z_i \in \{1, \dots, r\}$ zuordnen.

o.B.d.A ~~setze~~ existiert d, s.d. alle Berechnungen von M auf w höchstens $d \cdot s(|w|)$ Lang. (i.e. nicht mehr als $d \cdot s(|w|)$ Speicherplatz)

A $(k+2)$-Band TM s.d. alle Berechnungen von M höchstens Länge $|\text{Inkonf}_M(n)|$ haben. Für $w \in \Sigma^*$ arb. A wie folgt:

i) A schreibt $0^{s(|w|)}$ auf $k+2$ - te Band

ii) " $0^{d \cdot s(|w|)}$ $k+1$ - te Band und löscht Inhalt des $k+2$ ten Bandes

iii) A gen. der Reihe nach Wörter $z \in \{1, \dots, r\}^*$ (Länge $\leq d \cdot s(n)$) auf $k-2$ ten Band ∧ sim. für z auf $1-k$ Bändern ~~A auf z~~. M auf w, ~~falls~~ die Berechnung z. Falls M akz. $w \to A$ ✓

Sonst ↯

Klar, dass $L(A) = L(M)$, da M keine längeren Berechnungen als $d \cdot s(n)$ hat.

$Space_M(n) \leq s(n) \cdot d$, da $Space_M(n) \leq Time_M(n)$.

$k+1$ Band benutzt $s(n)d$ Felder, $k+2$ benutzt höchst. $ds(n)$

$\Rightarrow Space_A(n) \leq d \cdot s(n)$ ∎

Lemma 6.9.: $SAT \leq_p$ Clique

Beweis: $F = F_1 \wedge \dots \wedge F_m$ $F_i = C_{i_1} \vee \dots \vee C_{i_k}$

$i_L \in \{x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n\}$ $1 \leq i \leq m$, $1 \leq k$ eine KNF

Konstruiere Eingabe für Clique (G, k) wie folgt:

- $k = m$
- Knoten $V \equiv$ Jedes literal 1 Knoten
- Kante gdw. Knoten nicht in selber Klausel und nicht Negation voneinander. Poly. Zeit klar

Lemma 6.10: Clique $\leq_p$ VC

Beweis:

~~$G = (V$~~ $G = (V, E)$ 1 k Eingabe für Clique.

Konstruiere Eingabe $(\bar{G}, m)$ für VC wie folgt:

- $m = |V| - k$
- $\bar{G} = G^c$

Konstruktion klar in poly. Zeit

Lemma 6.11: $SAT \leq_p 3\text{-}SAT$

Beweis:

Sei $F = F_1 \wedge \dots \wedge F_m$ KNF. Konstruktion:

Falls $|F_i| < 4$: $C_i = F_i$

Falls $|F_i| \geq 4$ (~~mehr als~~ min. 4 Literale):

Sei $F_i = z_{i_1} \vee \dots \vee z_{i_k}$. Sei C_i über $\{x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n, y_{i,1}, \dots, y_{i,k-3}, \bar{y}_{i,1}, \dots, \bar{y}_{i,k-3}\}$ $\wedge$ bei C_j für $j \neq i$ nicht verwendet

$C_i = (z_1 \vee z_2 \vee y_{i,1}) \wedge (\bar{y}_{i,1} \vee z_3 \vee y_{i,2}) \wedge (\bar{y}_{i,2} \vee z_4 \vee y_{i,3})$

$\wedge \dots \wedge (\bar{y}_{i,k-4} \vee z_{k-2} \vee y_{i,k-3}) \wedge (\bar{y}_{i,k-3} \vee z_{k-1} \vee z_k)$

Poly. Zeit klar

Lessons learned TWF

- Lösungsmenge wird von TM immer in endlicher Zeit verifiziert
- Algo. ≡ TM welcher immer hält
- Wir geben einen Algo. F an, der eine Eingabe ~~x∈~~ x für L_1 in eine Eingabe f(x) für L_2 transformiert ← für EE-Reduktionen
- Falls 2 TM basically gleich sind → Transitionen umleiten
- Falls Lösungsmenge Wörter enthält, für die nicht halten:

  Variante 1) F generiert die Kodierung einer TM ← EE Reduktion
  M', welche unabhängig von seiner Eingabe M auf x simuliert; dann berechnet F weiter den Index von M' und f(w) = w_i (falls die Ausgabe ist ~~tt~~; die i-te TM f(w_i) ~~verwer~~ nicht akzeptieren soll)

  sobald Eingabe vorkommt da Output gleich

  R-Red, falls Output invers

  Variante 2) F generiert die Kodierung einer TM M', wobei M' für seine Eingabe y∈{0,1}* |y|-Schritte von M auf x simuliert. Falls hält (innerhalb von |y| Schritten): verweigern, sonst akzeptieren → Lösungsmenge wird Σ^*

  EE-Red. ↳ L_{HALT}
- $\leq_R$ kann man auch via $\leq_{EE}$ zeigen
- $L_M \leq_R L_{ungleich} = \{kod(M)\#kod(M') \mid L(M) \neq L(M')\}$ via $\leq_{EE}$

  Setze B TM $L(B) = \emptyset$. $w \neq kod(M)\#x$: $f(w) = \lambda$

  $w = kod(M)\#x$: generiere TM A welche Eingabe y verwirft wenn $y \neq x$ und sonst simuliert A die Arbeit von M auf x λ übernimmt die Ausgabe

  $\Rightarrow L(A) = \begin{cases} \emptyset, & \text{sonst} \\ \{x\}, & w \in L(M) \end{cases}$

- Reguläre Grammatiken via EA zeigen
- man kann anstelle nur 1 Wort n.d. zu wählen auch 11 Wörter n.d. wählen
  → Wenn sie in Schnitten von Lgsmengen liegen, dann kann man sie auch hintereinander schalten
- $L \notin L_{RE}$ via
  1) $\leq_{EE}$ Reduktion von Sprache die nicht in L_{RE} liegt
     $(L)^c \notin L_{RE}$ $\forall L^c$ s.d. $L \in L_{RE}$
  2) via $L^c \in L_{RE}$ zeigen, falls bekannt, dass $L \notin L_R$
- Vereinigung von Grammatiken $S \to A|B$
- $\in NP$
  Variante 1): NMTM angeben welche in poly. Zeit arbeitet ∧ $L = L_{gewünscht}$ via n.d. Wahl
  (poly. Zeit klar)
  Variante 2): Angenommen (z.B. KUF Formel F) $\in L$.
  Dann $\exists$ (& w (z.B. Belegung ∧ weitere Eigenschaft) s.d. ... gilt. Wir Benutzen w als Zeugen.
  Gegeben w, können wir in poly. Zeit jede Klausel durchgehen und Eigenschaft überprüfen. $\Rightarrow \in NP$
- $\neg(\forall: A(x)) = \exists: \neg A(x)$
- $L_{diag} \leq_{EE} L_{all}^c$, ~~$L_{\emptyset} \notin L_{RE}$~~ $L_{\emptyset} \notin L_{all}^c \Rightarrow L_{all}^c \notin L_{RE}$
- Argumentation nichtdeterministische Wahl

"⇒" (∃... s.d. Eingabe ✓). Also gibt es dann eine akzeptierende Berechnung von M auf Eingabe w, in der M genau diese Wort x nichtdeterministisch wählt.

"⇐" (∄ s.d. Eingabe ✓). Also gibt es keine akz. Berechnung von M auf w.