Teil 1 | Probleme:

Problem = eine abz. unendl. Menge von Inst. + Frage

Entscheidungsproblem = Problem mit Ja/Nein Antwort

Funktionsproblem = Erg. f auf Input x

Optimierungsproblem = opt. Wert einer Funktion

Suchproblem = Finde Lösung zu Inst.

Aufzähl- = berechne alle Lösungen zu Inst.

Zählprob- = zähle Lösungen zu Inst.

EP = $\Sigma^* \to \{a,1\}$

Algorithmus = Beschr. von Rechenschr. die jede bel. Inst. von P lösen

- hält nach endl. vielen Schritten | - Schritt einfach
- liefert korrekte Antwort | - von Menschen verst.

Programm: $\neq$ Alg. basiert auf Berechnungsmodell

- nicht für alle erlaubten Inst. eine Antwort
- wenn auf jeder Inst. terminiert => Alg.

Halteproblem: (HP)

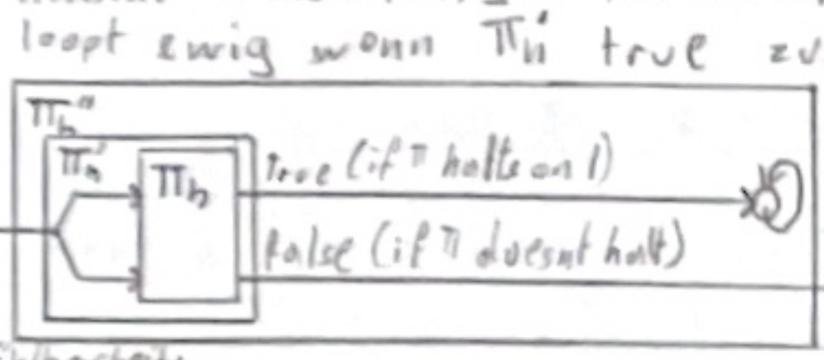
Instanz = Π Progr + I Input-Str Frage = Term. Π auf I

Bew. für Unentscheidbarkeit:

Π_h nimmt Π, I als Inputs und gibt true wenn Π term.

Π_h' nimmt Π als Input, dupl. ihn und gibt ihn an Π_h weiter

Π_h'' nimmt Π als Input, gibt ihn an Π_h' weiter und loopt ewig wenn Π_h' true zurückgibt



1) Π hält auf Π => Π_h'' hält nicht => ↯

2) Π hält nicht auf Π => Π_h'' hält ↯

Berechenbarkeit:

Funktion berechenbar <=> $\exists$ Alg: $\forall x \in G$ Input $f(x) \neq 0$ | x = Input, f(x) = Output

G = Menge als Σ^* codierbar

$0 = \{0,1\}$ für Entscheidungsprobleme

Entscheidbarkeit: Menge der Inst.

P ist entscheidbar <=> $\exists$ Alg: $\forall x \in I$: x = Input korrekte Antwort als Output

Abzählbarkeit von Σ^*: Ann $\Sigma = \{a_1, \dots, a_k\}$

- Aufz. geordnet nach Länge
- Innerhalb selber Länge lexigographisch

$\Sigma^* \to \mathbb{N}$: String auf pos. abbilden

ε	1	0
$a_1, \dots, a_k$	k	$1, \dots, k$
$a_1a_1, a_1a_2, \dots, a_ka_k$	k^2	$k+1, \dots, k^2+k$
$a_1a_1a_1, a_1a_1a_2, \dots, a_ka_ka_k$	k^3	$k^2+k+1, \dots, k^3+k^2+k$

Korrektheit:

Inst = Progr Π, Inputs I_1, I_2; Frage = liefert Π mit I_1 den Output I_2

Erreichbarer Code: (EB)

Inst = Π + $n \in \mathbb{N}$; Frage = $\exists I$ sodass Π mit I die Zeile n ausführt

Semi-Entscheidbarkeit: P ist semi-entscheid. wenn $\exists \Pi$:

- Π nimmt Inst I von P als Input
- Falls I eine pos. Inst. ist => Π liefert true
- Falls I neg. Inst => Π liefert false oder terminiert nicht

Überabzählbarkeit von $\{0,1\}^{\mathbb{N}}$:

Ann. Funkt- $\{0,1\}^{\mathbb{N}}$ abzählbar (wie Σ^*).
Jede Funkt. darstellbar als Folge von 0, 1

$f_0 = \underline{a_{00}}\ a_{01}\ a_{02} \dots$

$f_1 = a_{10}\ \underline{a_{11}}\ a_{12} \dots$

$f_2 = a_{20}\ a_{21}\ \underline{a_{22}} \dots$

$\vdots$

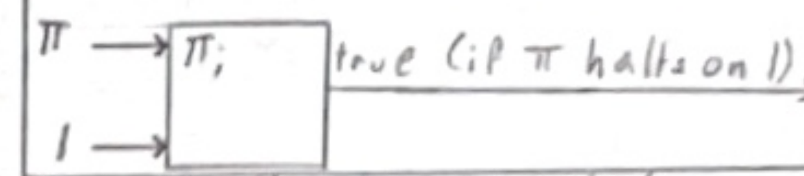
$\hat{f}: \mathbb{N} \to \{0,1\}$ mit $\hat{f}(n) = 1 - f_n(n)$

... dreht die Werte der Diag. um

=> nicht in Aufzählung enthalten

HP ist semi-entsch:

Sei Π_i ein Interpreterprogr. nimmt ein bel. $(\Pi, I) \in$ HP. Analysiert Ausführung Π auf I.
Wenn Π auf I terminiert gibt Π_i true zurück sonst läuft Π_i endlos auf (Π, I)



Korrektheit ist semi-entsch wie HP

Π_i nimmt Inst $(\Pi, I_1, I_2) \in$ Korrektheit, überprüft ob Output O von $\Pi = I_2$ ist und gibt entsprechend true oder false zurück.
Wenn Π auf I_1 nicht terminiert läuft Π_i endlos

EB ist semi-entsch.:

Sei (I, i) Input für Π_i der entscheidet ob in $i \in \mathbb{N}$ Schritten Zeile n von Π erreicht.
Wenn (Π, I) pos => finden wir in Aufzählung immer ein (I, i) sodass Π_i true liefert
Wenn (Π, I) neg => Aufzählung endlos

Abzählbarkeit

M ist abzählbar wenn endlich oder $f: M \to \mathbb{N}$ bij.

- def. bij. $g: \mathbb{N} \to M$
- oder inj $f: M \to \mathbb{N}$ => $|M| \leq |\mathbb{N}|$

womit ein bij $g: M \to \mathbb{N}$ mit $g(a) = |\{i : i < f(a)\}|$ $\forall a \in M$

Cantorsches Abzählprinzip: $M_1 \times M_2$ mit $M_1 = \{a_1, a_2, \dots\}$, $M_2 = \{b_1, b_2, \dots\}$

$$\begin{pmatrix} (a_1,b_1) & (a_1,b_2) & (a_1,b_3) & \cdots \\ (a_2,b_1) & (a_2,b_2) & (a_2,b_3) \\ (a_3,b_1) & (a_3,b_2) & (a_3,b_3) \\ \vdots & & \vdots \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 4 & 9 \\ 2 & 3 & 8 \\ 5 & 6 & 7 \end{pmatrix}$$

Entscheidungsproblem: Inst = ϕ präd. log Formel 1. Stufe, Frage = ist ϕ gültig

EP ist unentscheidbar
- Reduktion auf HP

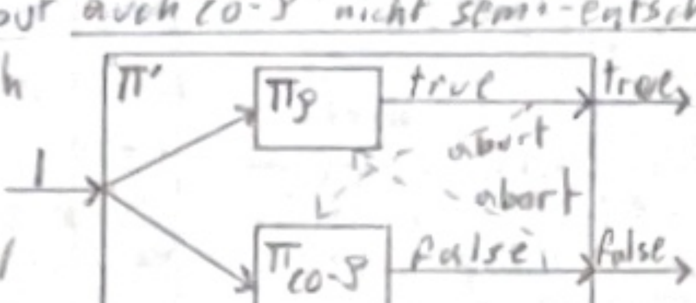
EP ist semi-entscheidbar
- Kalkül mit endl. Abl gibt es ja
- einfach alle mögl. Abl. in Kalkül aufzählen
- wenn gültig findet man eine sonst endlosschleife

Komplement: Komplement co-P zu Entscheidungsprob P hat gleiche Menge an Instanzen aber verneinte Frage

Wenn P entscheidbar => co-P entsch.
↳ Π' hat (Π, I) als Input und negiert output

Wenn P und co-P semi-entsch => P entsch
↳ $\exists \Pi_P$ => pos Inst. return true
$\exists \Pi_{co\text{-}P}$ => neg Inst return true
-> Def. Π' das beide Programme parallel ausführt. Wenn eines terminiert dann brich anderes ab und gib entspr. Output

Wenn P unentscheidbar und semi-entsch.
=> co-P nicht-semi-entsch.
Bew. Ann co-P semi
=> co-P & P semi => P entsch ↯
=> es gibt Probleme P wo auch co-P nicht semi-entsch



Selber Output: Π_1, Π_2 Frage = beide gleich out
- pos. Inst wo keines term. nicht erkennbar => co-NP semi-entsch. ↯
- neg. Inst wo nur eines term auch nicht

Programm-Äquiv: Π_1, Π_2 Frage = Outputs gleich
- wie SO nur noch komplexer

Universelles-HP: Hält Π auf bel. $I \in \Sigma^*$
- pos. Inst nicht erkennbar da wir unendlich viele I prüfen müssten
- neg. Inst. auch nicht da wir nicht-Terminierung zeigen müssten

Reduktionen:
neues Problem A leichtes B
$A \leq B$: A min so leicht wie B
neues is. schwer als H
$A \leq B$
B min so schwer A

Turing-Red $P_1 \leq P_2$ <=> co-$P_1 \leq$ co-P_2
Verf. für A verwendet Verf. für B als Unterprog.
Steuerprog

Steuerprog muss effizient sein
Cook Red = poly. time

Many-One:
Sei R eine Funkt. die Inst x von A zu Inst R(x) von B reduziert
Erg von R(x) ist Erg von A
x ist pos. Inst von A <=> R(x) pos von B

R muss effizient sein
Karp-Red = poly. time

Teil 2 | Turingmaschine: $M = (Q, T, \Gamma, \delta, q_0, \{z_0, z_1, z_2\}, B, F)$

z_1 | a_1 | ... | a_n | z_2 | Eing.
Arb. Alph. | Eing-Alph. | Start | Blank $\in \Gamma$
z_0 | B | ... | Ausg. | Zustände | Übergf.

FSA Endz

Endl. Autom: kein EB => Überg. $(q, a; p, D_E)$ => reg. Spr.
wenn $D_E = \{R, S\}$ => $(q, a; p)$ mit $a \in T \cup \{\varepsilon\}$
=> $(q, a; p)$ mit $a \in T$ => $(q, a; p, R)$
$(q, \varepsilon; p)$ => $(q, a; p, S)$ mit bel. $a \in T$

Det. TM: Deck. Aufz. Spr.
$(q, a, X; p, Y, D_E, D_A) \in \delta$ zur Funktion wird
$\delta(q, a, X) = (p, Y, D_E, D_A)$

NF
- 1 Endz. f
- AB am Ende leer
- letzter Übergang: $\delta(s, z_2, z_0) = (f, z_2, P)$

LBA: => Kontexts. Spr.
Eing-Wort w mit $|w|$ = max. cnt d slots am AB

Keller-Automat: Kontextfrei (in NF beides = NF bei norm TM)
- EB nie nach links
- links von Lese-/Schreib-Kopf nur nicht-Blank
- rechts davon nur Blank

akzept. durch
1) TM in EZ
2) Kellersp. leer

DEA $A = (Q, \Sigma, \delta, q_0, F)$
Q... Zust. | q_0 ... Start
Σ ... Eing-Alp | F ... Endz.
$\delta: Q \times \Sigma \to Q$ (total)

Überg-Funkt

Akz. Spr. $\mathcal{L}(A) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$

Falle = $q \in Q - F$ mit $\delta(q, a) = q$

Reg(Σ) kleinste Menge über Σ:
- $\{\}, \{a\}, \{\varepsilon\} \in \text{reg}(\Sigma)$, $\forall a \in \Sigma$
- $A, B \in \text{reg}(\Sigma)$ => $A \cup B$, $A \cdot B$, $A^* \in \text{reg}(\Sigma)$

Reg-Menge

Erw. Überg. $\delta^*: Q \times \Sigma^* \to Q$
$\delta^*(q, \varepsilon) = q$; $\delta^*(q, wa) = \delta^*(\delta(q, a), w)$ mit $a \in \Sigma$, $w \in \Sigma^*$

NEA:
- NEA: $\delta: Q \times \Sigma \to \mathcal{P}(Q)$
- ε-NEA: $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$
- Erw: $\delta^*: Q \times \Sigma^* \to \mathcal{P}(Q)$

Akz. Spr.: $\mathcal{L}(A) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \cap F \neq \emptyset\}$

- Typ-0 => rek. aufz. Spr.
- monoton => monotone Spr.
- Typ-1 => kontextsensitive Spr.
- Typ-2 => kontextfreie Spr.
- Typ-3 => reguläre Spr.

Minimalautomat: Jede reg. Spr. kann zu min. DEA konstruiert werden, der bis auf Umbenennung eindeutig ist

Minimalisierungs-Alg.:
$A = (Q, \Sigma, \delta, q_0, F)$ DEA mit $\mathcal{L}(A) = L$
$A^r = (Q, \Sigma, \{(q, a, p) \mid (p, a, q) \in \delta\}, F, \{q_0\})$ mit $\mathcal{L}(A^r) = L^r$
1) Konstr. NEA A^r durch Spiegelung von A
2) Konstr. DEA B durch Determinisierung von A^r
3) Konstr. NEA B^r durch Spiegelung von B
4) Konstr DEA C durch Determinisierung von B^r
C ist zu A äquiv. Minimalautomat

Pumping-Lemma für reg Spr:
Sei L unendl. reg. Spr. => $\exists m > 0$:
$\forall w \in L$ mit $|w| \geq m$: $\exists x, y, z$: $w = xyz$ mit $|xy| \leq m$ und $|y| > 0$ sodass $w_i = xy^iz \in L$ $\forall i \geq 0$

Beweis-Alg.: Ann L ist reg.
Dann ist für bel. m ein $w \in L$ mit $|w| \geq m$
Betrachte bel. Zerlegung $w = xyz$ mit $|xy| \leq m$ und $|y| > 0$ => stelle Aussage auf
Wähle ein i sodass Aussage nicht mehr gelten kann

Grammatik: $G = (N, T, P, S)$
N ... endl. Menge an Non-Term
T ... Terminalsymbole
$P \subseteq (N \cup T)^+ \times (N \cup T)^*$ ... Produktionen
$S \in N$ ... Startsymbol

- G ist unb. Grammatik (Typ-0)
- $|\alpha| \leq |\beta|$ monoton
- $\alpha = uAv$, $\beta = uwv$ für $A \in N$; $w \in (N \cup T)^+$; $u, v \in (N \cup T)^*$ (Typ-1) Kontexts.
- $A \to \beta$ für $A \in N$ kontextfrei (Typ-2)
- $A \to aB$ oder $A \to \varepsilon$ regulär (Typ-3)

(Aufb. Ca/Clo) Kontexts

Eindeutigkeit/Mehrdeutigkeit:
$G = (N, T, P, S)$ ist kf.
eindeutig <=> jedes ableitbare Terminalwort hat genau eine Linksableitung
mehrdeutig <=> nicht eindeutig
Sprache L ist inherent mehrd. wenn jede Grammatik die L erzeugt mehrdeutig ist

Chomsky-NF:
Alle Prod. sind der Form $A \to BC$ oder $A \to a$
$S \to \varepsilon$ nur erlaubt wenn S nicht auf rechter Seite vorkommt

Greibach-NF: $A \to aw$ mit $(a \in T, A \in N, w \in N^*)$

Erw. Greibach-NF: $(a \in T, A \in N, w \in (N \cup T)^*)$

Dyck-Sprachen:
Sei D_n über $\Gamma_n = \{(_1,)_1, \dots, (_n,)_n\}$ die kleinste Menge:
- $\varepsilon \in D_n$
- $v \in D_n$ => $(_i\, v\,)_i \in D_n$, $1 \leq i \leq n$
- $v_1, v_2 \in D_n$ => $v_1v_2 \in D_n$

Satz von Chomsky-Schützenberger:
Spr. L über Σ ist kf genau dann wenn $n > 0$, $h: \Gamma_n^* \to \Sigma^*$ existiert, sodass $L = h(D_n \cap R)$ wobei R eine reg. Spr. über Γ_n ist

Abschlusseigensch.

	$\mathcal{L}_3$	$\mathcal{L}_2$	$\mathcal{L}_1$	$\mathcal{L}_0$
Vereinigung	✓	✓	✓	✓
Konkat	✓	✓	✓	✓
Kleene-Star	✓	✓	✓	✓
Komplement	✓	X	✓	X
Durchsch.	✓	X	✓	✓
Durchsch. reg Mengen	✓	✓	✓	✓
Homomorph.	✓	✓	X	✓
ε-frei Hom.	✓	✓	✓	✓
gsm-Abb.	✓	✓	X	✓
ε-frei gsm.	✓	✓	✓	✓

Entsch. Probleme

Reg:
- $w \in L$
- L leer?
- L endl/unendl
- $L = L'$

} entscheidbar

Kf.:
- kf. leer/unendl/endl
- $w \in L$
- $L = \Sigma^*$
- $L_1 \subseteq L_2$
- L reg?
- $L_1 = L_2$
- $L_1 \cap L_2 = \{\}$
- $L_1 \cap L_2$ bzw $\Sigma^* - L$ kf?

} unentscheid.

Chomsky-Hierarchie

Spr. Fam.	Gramm.	
$\mathcal{L}_0$ rek. aufz.	unbeschr.	TM = (EA + RAM)
$\mathcal{L}_1$ (Kontexts)	Kontexts. monoton	LBA = (EA beschr. RAM)
$\mathcal{L}_2$ (Kf.)	Kontext-frei	Kellerautom = (EA + Stack)
$\mathcal{L}_3$ (reg.)	regulär	endl. Autom (EA)

$\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_{rec} \subset \mathcal{L}_0$

gsm-Abbildung:
$f_M(w)$ def durch alle Ausgabewörter v die bei Analyse von w auf einem Pfad zw. q_0 und $p \in F$ ergeben
$f_M(L) = \{v \in \Gamma^* \mid v \in f_M(w)$ für $w \in \Sigma^*\}$

$M_h = (\{q\}, \Sigma, \Gamma, \delta, q, \{q\})$ mit $\delta(q, a) = (q, h(a))$ ist gsm eines hom.

Reg.:
- Plus-operator $A^+ = A^* \cdot A$
- Spiegelung: Start & End vertauschen, Überg. umkehren
- Differenz $A - B = A \cap \overline{B}$

K.f.
- Schnitt mit reg-Sprache

Pumping-Lemma für kf-Spr:
Sei L eine unendl. kf Spr.
Dann $\exists m > 0$: $\forall w \in L$ mit $|w| \geq m$ Wörter u, v, x, y, z sodass $w = uvxyz$ mit $|vxy| \leq m$ und $|vy| \geq 1$ sowie
$w_i = uv^ixy^iz \in L$ $\forall i \geq 0$

Homomorphism:
Abb $h: \Sigma \to \Gamma^*$ ist hom.
1) $h(\varepsilon) = \varepsilon$
2) $h(wa) = h(w)h(a)$ mit $w \in \Sigma^*$, $a \in \Sigma$
$h(L) = \{h(w) \mid w \in L\}$
ε-frei <=> $\forall a \in \Sigma$ $h(a) \neq \varepsilon$

Korollar zum PL:
Sei $L \subseteq \{a\}^*$, sodass $L = \{a^{f(n)} \mid n \geq 0\}$ für streng monoton wachsende Funktionen in $\mathbb{N}$. Gibt es für jede k ein n(k) sodass $f(n(k)+1) - f(n(k)) \geq k$ => nicht kf

Rek. (entsch) Sprachen:
$L \in \mathcal{L}_0(\Sigma)$ ist rek. (entsch) wenn $\overline{L} = \Sigma^* - L$ rek. aufz.
d.h $\overline{L} \in \mathcal{L}_0(\Sigma)$

Wortproblem:
Das Problem $w \in L$ ist für rek. Sprachen entsch.

Church-Turing-These 1:
Wenn eine Funktion in irgendeinem Sinn berechenbar ist, so ist sie schon auf einer TM berechenbar

Church-Turing These 2:
Gibt es ein beschr. Verfahren zur exakten Spez einer form. Sprache L so existiert eine TM die L akzeptiert

Form. Sprachen und Funktionen lassen sich ineinander übersetzen
- $L \subseteq \Sigma^* \Rightarrow \Sigma^* \mapsto \{0,1\}$
- $(\Sigma^*)^k \longrightarrow (\Sigma)^m \Rightarrow \mathbb{N}^k \mapsto \mathbb{N}^m$

Rek. aufz. (semi-entsch.) Spr.:
L ist rek. aufz. <=> TM akzeptiert L
Rek. (entsch.) Spr.:
L ist rekursiv <=> eine TM die immer hält akzeptiert L

... in bezug zu Church-Turing 2 (nur det. TM)

TM M berechnet $f: \mathbb{N}^k \to \mathbb{N}^m$ wenn M auf Eingabe $(n_1, \dots, n_k)$. $f(n_1, \dots, n_k)$ ausgibt: $F(M) = f$

f ist Turing-berechenbar <=> $F(M) = f$ für M

Univ. TM: $\langle M \rangle$ ... Code von M
Eine TM U ist universell <=> ∀ TM M und $\forall (n_1, \dots, n_k) \in \mathbb{N}^k$:
$F(U)(\langle M \rangle, n_1, \dots, n_k) = F(M)(n_1, \dots, n_k)$

Registermaschine:
m-RM: $R = (S, O, T)$
$S = \mathbb{N}^m$ Speicherzust.
$O = \{A_i \mid 1 \le i \le m\} \cup \{S_i \mid 1 \le i \le m\}$ Speicherbefehle
$T = \{t_i \mid 1 \le i \le m\}$ Testbefehle

RM-Programm:
- startmarke n
- endl. Menge an Anweisungen
  - (r, f, p) r: do f then goto p
  - (r, t, p, q) r: if t then goto p else goto q

r ... Kennmarke, p,q ... Sprungmarken

Semantik:
A_i: $R(i) := R(i) + 1$
S_i: $R(i) := R(i) - 1$
t_i: $R(i) == 0$

Konfiguration: $r: (n_1, \dots, n_m)$

Univ. Reg.-Masch.:
∃ univ. progr. P_U: ⟨F⟩ Code eines RM-Progr. das $f: \mathbb{N}^k \to \mathbb{N}$ dann berechnet P_U die Funktion $u: \mathbb{N}^{k+1} \to \mathbb{N}$ sodass $u(\langle F \rangle, n_1, \dots, n_k) = f(n_1, \dots, n_k)$

RM = TM berechnungsäquiv.
=> **Church-Turing These 3:** Wenn arithm. Funk. inform. berechenbar ist, so ist sie auf RM berechenbar
2-RM berechnungsuniv. (mit entspr. Cod.)

λ-Terme:
- Jede Var x, y ... ist ein λ-Term
- M, N λ-Terme => (MN) λ-Term
- x Var, M λ-Term => (λx. M) λ-Term

FV(t) & GV(t)
- $FV(x) = \{x\}$ $GV(x) = \{x\}$
- $FV(MN) = FV(M) \cup FV(N)$ $GV(MN) = GV(M) \cup GV(N)$
- $FV(\lambda x. M) = FV(M) - \{x\}$ $GV(\lambda x. M) = GV(M) \cup \{x\}$

Currying:
mehrstellige Funk. => iter. Anwendung einst.
$f: A \times B \times \dots \times Z \Rightarrow f: A \to (B \to (\dots Y \to Z) \dots)$
verlangt prefix Notation: $m+n \to +(m,n) \to (+\,m)\,n$

α-Konversion: $M\{y/x\}$ geb. x in M durch y ersetzen

Substitution freier x durch N:
$x[N/x] = N$ $(\lambda x. M)[N/x] = \lambda x. M$
$y[N/x] = y$ wenn $x \neq y$ $(\lambda y. M)[N/x] = \lambda y. (M[N/x])$ wenn $x \neq y$, $y \notin FV(N)$
$(MP)[N/x] = (M[N/x])(P[N/x])$ $(\lambda y. M)[N/x] = \lambda y'. (M\{y'/y\})[N/x]$ wenn $x \neq y$, $y \in FV(N)$, y' neu

β-Reduktionen:
Ein Term (λx.M)N in t wird durch M[N/x] ersetzt. $t \to_\beta t'$. (λx.M)N ... Redex in t
Ein Term ohne Redex ist in NF

β-Äquiv.: $M =_\beta M'$ <=> nach β-Red. M zu M' transf.

Fixpunkt: $FN =_\beta N$ ... N ist FP von F

Fixpunkt-Satz: Jeder λ-Term hat einen FP

Y-Gen: $Y = \lambda f. (\lambda x. f(xx))(\lambda x. f(xx))$ ist FP-Gen.
=> $YF =_\beta FYF$ für ∀F ∈ λ-Terme

Rek. Funktionen:
Berechenbare Grundfunktionen def. durch Op. komplexere Funk.
Obda: $\mathbb{N}^k \mapsto \mathbb{N}$, $k \geq 0$
- konst. C_n^k: $C_n^k(x_1, \dots, x_k) = n$
- Nachf. S: $S(x) = x + 1$
- Proj. P_i^k: $P_i^k(x_1, \dots, x_k) = x_i$ mit $(1 \le i \le k)$

Komposition:
$h: \mathbb{N}^m \mapsto \mathbb{N}$ und für $1 \le i \le m$, $g_i: \mathbb{N}^k \mapsto \mathbb{N}$ ist $f = (h \circ (g_1, \dots, g_m)): \mathbb{N}^k \mapsto \mathbb{N}$ def. als $f(x_1, \dots, x_k) = h(g_1(x_1, \dots, x_k), \dots, g_m(x_1, \dots, x_k))$

Prim. Rek.: Ind. (rek.) Funk-def haben startfunktion und Übergangsfunk
Sei $g: \mathbb{N}^k \mapsto \mathbb{N}$ und $h: \mathbb{N}^{k+2} \mapsto \mathbb{N}$, ist $f = Pr(g, h): \mathbb{N}^{k+1} \mapsto \mathbb{N}$ def. durch $f(0, x_1, \dots, x_k) = g(x_1, \dots, x_k)$ und $f(m+1, x_1, \dots, x_k) = h(m, f(m, x_1, \dots, x_k), x_1, \dots, x_k)$

μ/μ̄-Rek. Funk.: Pr-Funk sind nur total
Geg. totale Funk. $g: \mathbb{N}^{k+1} \mapsto \mathbb{N}$ sind $f = \mu g: \mathbb{N}^k \mapsto \mathbb{N}$, $f' = \bar{\mu} g: \mathbb{N}^{k+1} \mapsto \mathbb{N}$ definiert durch $f(x_1, \dots, x_k) = \min_{y \geq 0} [g(y, x_1, \dots, x_k) = 0]$ bzw. $f'(x_1, \dots, x_k) = \min_{y \geq 0} [g(y, x_2, \dots, x_k) = x_1]$
=> μ-rek. / partiell rek.

Church-Turing-These 4:
Wenn eine arithm. Funktion inform. berechenbar ist, dann lässt sie sich als λ-Term darstellen und im λ-Kalk ber.

Eine arithm Funktion ist genau dann λ-definierbar wenn sie Turing-berechenbar ist

Theorem
Jede auf TM berechenb. Funktion ist partiell rek

Church-Turing These 5:
Die Klasse der inform. berechenb. Funktionen stimmt mit der Klasse part. rek. Funk. überein

Ent. Eigenschaften:
Spr. Eigensch. $\mathcal{E} \subset \mathcal{P}(\Sigma^*)$ Teilmenge rek. aufz. Spr. über Σ.
$\mathcal{E}$ ist trivial wenn entweder leer oder aus allen rek. aufz. Spr. besteht

Sprach-Code Eig.
Für jede $\mathcal{E}$ sei
$L_\mathcal{E} = \{\langle M \rangle \mid L(M) \in \mathcal{E}\}$ Menge aller Codes von TM die eine Spr $L \in \mathcal{E}$ akzeptieren

Satz von Rice:
Für jede nicht-triv Eigenschaft $\mathcal{E}$ rek. aufz. Spr. gilt $L_\mathcal{E}$ ist unentscheidbar

Sprach-Code Funk-Eig.:
Für jede Funk-Eig. $\mathcal{F}$ ist
$L_\mathcal{F} = \{\langle M \rangle \mid F(M) \in \mathcal{F}\}$ Menge aller TM Codes die Funk $f \in \mathcal{F}$ berechnen

Satz von Rice für Funk.
Für jede nicht-triv Eigenschaft $\mathcal{F}$ von Turing-berechenb. Funktionen ist Sprache $C(\mathcal{F})$ unentscheidbar

NP-Hard/NP-Vollst.: C ist eine Kompl-Klasse
$\mathcal{P}$ ist C-Hard <=> ∀P' ∈ C auf $\mathcal{P}$ reduziert
$\mathcal{P}$ ist C-Compl. <=> $\mathcal{P} \in C \wedge \mathcal{P} \in$ C-Hard
$\mathcal{P} \in$ C-Hard, $\mathcal{P} \le_R \mathcal{P}^*$ => $\mathcal{P}^*$ C-Hard

Teil 3

Komplexität eines Alg:
Anhand einer Funktion f(), sodass für alle Inst. der Größe n gilt, dass die Antwort in O(f(n)) Operationen bzw Bits im Speicher berechnet werden kann.
$f, g: \mathbb{N} \mapsto \mathbb{N}$ $g(n) = O(f(n))$ <=>
$\exists c, n_0 \in \mathbb{N}: \forall n \geq n_0: g(n) \le c \cdot f(n)$

Klasse-P:
Enthält jene Probleme $\mathcal{P}$ sodass
- es gibt ein Progr. Π für $\mathcal{P}$
- ∀I ∈ Inst.: Laufzeit in $O(|I|^k)$ k ... konst.

Problem ohne poly-time Alg. = Intractable
Intractability – kein Alg bekannt ≤ NP
bew. Intr. = kein Alg. möglich => EXPTIME

Spezialfall
$\mathcal{P} \in NP$ wenn $\mathcal{P}_2$ spez. von $\mathcal{P}$ ist gilt $\mathcal{P}_2 \in NP$

Klasse-NP:
Non-determ. poly-time
Zert.-Rel.
- pos. Inst. haben kompakte Zertifikate
- Geg. Inst I von $\mathcal{P}$ und Zert-Kandidat c, ist es einfach zu prüfen ob c ein Zert. ist

$\mathcal{P}$ ist Problem Inst($\mathcal{P}$) ... Menge an Instanzen. $R \subseteq Inst(\mathcal{P}) \times STRINGS$ sodass $I \in Inst(\mathcal{P})$ ist pos. Inst <=> ∃ c ∈ STRINGS: $(I, c) \in R$

R ist pol. balanciert <=> $\forall (v_1, v_2) \in R: |v_2| \le |v_1|^k$, $k \geq 1$
R ist pol. entsch. <=> $\exists Alg(v_1, v_2), v_1, v_2 \in R: O(|v_1, v_2|^k)$
$\mathcal{P}$ in NP <=> pol. bal. & pol. entsch.

Choice-Progr.
choice(Ass1, Ass2) teilt Berechnung in zwei Zweige.
LZ ist LZ des langsten Zweigs
Progr. liefert true wenn min 1 Zweig true liefert
for all $1 \le i \le n$ do
  choice(t_i = true, t_i = false)
return modelcheck(s, [$t_1, \dots, t_n$]);
$\mathcal{P}$ in NP <=>
- choice Progr Π für $\mathcal{P}$
- LZ in $O(|I|^k)$

Klasse-L:
$\mathcal{P} \in L$ <=> ∀I ∈ $\mathcal{P}$:
Π benötigt $O(\log_2 |I|)$ Bits im Speicher
Input in sep. Read-only-Memory
$c \log_2 n + d < n$
L ⊆ P

P-Space:
$\mathcal{P} \in$ PSPACE <=> ∀I ∈ $\mathcal{P}$:
Π benötigt $O(|I|^k)$ bits im Speicher
PSPACE-Compl => Intractable
NP ⊆ PSPACE

EXPTIME:
$\mathcal{P} \in$ EXPTIME <=> ∀I ∈ $\mathcal{P}$: ∃Π: Π benötigt $O(2^{|I|^k})$ LZ

PSPACE ⊆ EXPTIME
P ⊂ EXPTIME

L ⊆ NL
PSPACE ⊆ NPSPACE
EXPSPACE ⊆ NEXPSPACE

es ist noch offen ob L = NL aber:
PSPACE = NPSPACE
EXPSPACE = NEXPSPACE

$Add = Pr(P_1^1, S \circ P_2^3)$
$Add(0,5) = P_1^1(5) = 5$
$Add(1,5) = S \circ P_2^3(0,5,5) = S(5) = 6$
$Add(2,5) = S \circ P_2^3(1,6,5) = S(6) = 7$
$Add(3,5) = S \circ P_2^3(1,7,5) = S(7) = 8$

$Mult = Pr(C_0^1, Add \circ (P_2^3, P_3^3))$
$Fact = Pr(C_1^0, Mult \circ (P_1^2, S \circ P_2^2))$

$APred = Pr(C_0^0, P_1^2)$
$Signum = Pr(C_0^0, C_1^2)$
$ASub = Pr(P_1^1, APred \circ P_2^3) \circ (P_2^2, P_1^2)$
$Not = ASub \circ (C_1^1, P_1^1)$
$And = Mult$
$Or = Signum \circ Add$

$T = \lambda xy. x$ $F = \lambda xy. y$
if_then_else $= \lambda pxy. pxy$
and $= \lambda pq. pqp$
or $= \lambda pq. ppq$
not $= \lambda pab. pba$
succ $= \lambda nfx. f(nfx)$ mult $= \lambda mnf. m(nf)$
add $= \lambda mnfx. mf(nfx)$

Teil 4

Syntax
KF-Grammatik abarbeiten.
Terminale in " "

$[\Pi]: S \mapsto S$

Operationale Semantik
Programm = Abbildung über Zustände S
$S \stackrel{def}{=} \{\sigma \mid \sigma: \mathcal{L}(Var) \mapsto \mathbb{Z}\}$

Konfig.: $C \stackrel{def}{=} \underbrace{(\mathcal{L}(Progr) \times S)}_{(\Pi, \sigma)} \cup \underbrace{S}_{\text{Endzust.}}$
$(\varepsilon, \sigma) \Rightarrow \sigma'$
(abort, σ) ⇏ ...
$(\Pi, \sigma) \Rightarrow (\Pi', \sigma')$ $\Rightarrow \sigma_e$

Hintereinander ausf.
$$\frac{(\Pi, \sigma) \Rightarrow (\Pi', \sigma')}{(\{\Pi; \Omega\}, \sigma) \Rightarrow (\{\Pi'; \Omega\}, \sigma')} \qquad \frac{(\Pi, \sigma) \Rightarrow \sigma'}{(\{\Pi; \Omega\}, \sigma) \Rightarrow (\Omega, \sigma')}$$

Konditionale
$$\frac{[e]\sigma \neq 0}{(\text{if } e \text{ then } \Pi \text{ else } \Omega, \sigma) \Rightarrow (\Pi, \sigma)} \qquad \frac{[e]\sigma = 0}{(\text{if } e \text{ then } \Pi \text{ else } \Omega, \sigma) \Rightarrow (\Omega, \sigma)}$$

While
$$\frac{[e]\sigma = 0}{(\text{while } e \text{ do } \Pi, \sigma) \Rightarrow \sigma} \qquad \frac{[e]\sigma \neq 0}{(\text{while } e \text{ do } \Pi, \sigma) \Rightarrow (\{\Pi; \text{while } e \text{ do } \Pi\}, \sigma)}$$

Auswertung
$[v]\sigma = \sigma(v)$ wenn $v \in \mathcal{L}(Var)$
$[n]\sigma = [n]$ wenn $[n] \in \mathbb{Z}$
$[u\,e]\sigma = [u]([e]\sigma)$ wenn $[u]: \mathbb{Z} \mapsto \mathbb{Z}$
$[e\ b\ e']\sigma = [b]([e]\sigma, [e']\sigma)$ w. $[b]: \mathbb{Z}^2 \mapsto \mathbb{Z}$

Zuweisung
$(v := e, \sigma) \Rightarrow \sigma'$
$\sigma'(w) = \begin{cases} [e]\sigma & \text{für } v = w \\ \sigma(w) & \text{sonst} \end{cases}$

Programmablauf
$\gamma_1 \Rightarrow \dots \gamma_k$ Endlich
$\nexists \gamma_{k+1}: \gamma_k \Rightarrow \gamma_{k+1}$ vollst.
$\gamma_k \in S$ ... final
$\gamma_k \in (\mathcal{L}(Progr) \times S)$ ... steckt

Strukt. Op. Sem.
$[\Pi]: S \mapsto S$ :: $[\Pi]\sigma = \sigma'$
<=> $(\Pi, \sigma) \Rightarrow^* \sigma'$ ∀σ, σ' ∈ S

Theorem
$[\varepsilon]\sigma = \sigma$
$[v := e]\sigma = \sigma'$, $\sigma'(w) = \begin{cases} [e]\sigma & \text{für } w = v \\ \sigma(w) & \text{sonst} \end{cases}$
$[\{\Pi; \Omega\}]\sigma = [\Omega]([\Pi]\sigma)$
$[\text{if } e \text{ then } \Pi \text{ else } \Omega]\sigma = \begin{cases} [\Pi]\sigma & \text{für } [e]\sigma \neq 0 \\ [\Omega]\sigma & \text{sonst} \end{cases}$
$[\text{while } e \text{ do } \Pi]\sigma = \begin{cases} [\text{while } e \text{ do } \Pi]([\Pi]\sigma) & \text{f. } [e]\sigma \neq 0 \\ \sigma & \text{sonst} \end{cases}$

Nat. Sem.
Theorem = def. von $[\Pi]: S \mapsto S$
$[\Pi]\sigma = [\{z := 0; \text{while} \dots\}]\sigma$ $(\Pi, \sigma) = (\{z := 0; \text{while}\}, \sigma)$
$= [\text{while} \dots]([z := 0]\sigma)$ $(z := 0, \sigma) \Rightarrow \sigma_1$
$= [\text{while} \dots]\sigma_1$ $\Rightarrow (\text{while}, \sigma_1)$
σ: $x \mapsto 3, \dots, z \mapsto 1$
σ_1: $z \mapsto [0]\sigma = 0$

Korrektheitsaussagen
Progr. + Spez   {F} Π {G}
Wenn Vorb. wahr => Nachb. wahr
∀σ ∈ S: $[F]\sigma \neq 0 \Rightarrow [G]\sigma' \neq 0$ mit $\sigma' = [\Pi]\sigma$

Partielle Korrektheit:
Vorb wahr & Term => Nachb wahr

Totale Korrektheit:
Vorb wahr => Term & Nachb. wahr
=> TK = PK + Term

Korrekt: Jede bew. Auss ist wahr
Vollst.: Jede wahre Auss. lässt sich bew.

Hoare Kalk.
Wenn Präm. gültig => Konkl. wahr

Implikationsregel
$$\frac{F \supset F' \quad \{F'\} \Pi \{G'\} \quad G' \supset G}{\{F\} \Pi \{G\}} \text{(imp)}$$

Zuweisung
(1) $\{F\}\ v := e\ \{F \wedge v = e\}$ (zw) <=> $v \notin F$, $v \notin e$
(2) $\{F\}\ v := e\ \{F[e^{-1}(v)/v]\}$ nur wenn ∃ e^{-1}
(3) $\{F\}\ v := e\ \{\exists v' (F[v'/v] \wedge v = e[v'/v])\}$ (zw') <=> $v' \notin F \cup e$
(4) $\{G[e/v]\}\ v := e\ \{G\}$ (zw)

Hintereinander ausf.
$$\frac{\{F\} \Pi \{G\} \quad \{G\} \Omega \{H\}}{\{F\} \{\Pi; \Omega\} \{G\}} \text{(ha)}$$

Konditionale
$$\frac{\{F \wedge e\} \Pi \{G\} \quad \{F \wedge \neg e\} \Omega \{G\}}{\{F\} \text{ if } e \text{ then } \Pi \text{ else } \Omega \{G\}} \text{(if)}$$

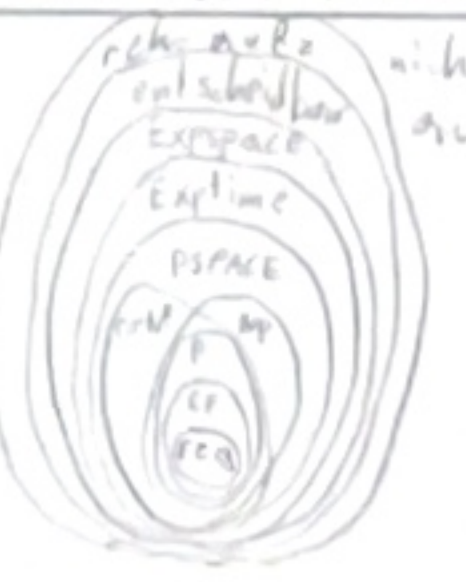
Abort
{F} abort {G} (ab) <=> part
{0} abort {G} (abt) <=> tot.

leer progr.
{F} ε {F}

(if')
$\{F\} \Pi \{H\}$ $\{G\} \Omega \{H\}$
$\{(e \supset F) \wedge (\neg e \supset G)\}$ if e ... {H}

Part. Korr. Schleifen
$$\frac{\{INV \wedge e\}\ \Pi\ \{INV\}}{\{INV\} \text{ while } e \text{ do } \Pi\ \{INV \wedge \neg e\}} \text{(wh)}$$

Totale Korr. Schleifen
t ... Term.-Funk. t ≥ 0, wird kleiner
$$\frac{\{INV \wedge e\}\ \Pi\ \{INV\} \quad \{INV \wedge e \wedge t = t_0\}\ \Pi\ \{t < t_0\}}{\{INV\} \text{ while } e \text{ do } \Pi\ \{INV \wedge \neg e\}}$$
$$\frac{\{INV \wedge e \wedge t = t_0\}\ \Pi\ \{INV \wedge 0 \le t < t_0\}}{\{INV\} \text{ while } e \text{ do } \Pi\ \{INV \wedge \neg e\}} \text{(wht)}$$



nicht-rek aufz.