6. Praktische Berechenbarkeit

Ziel:
Charakterisierung der praktisch berechenbaren Funktionen.

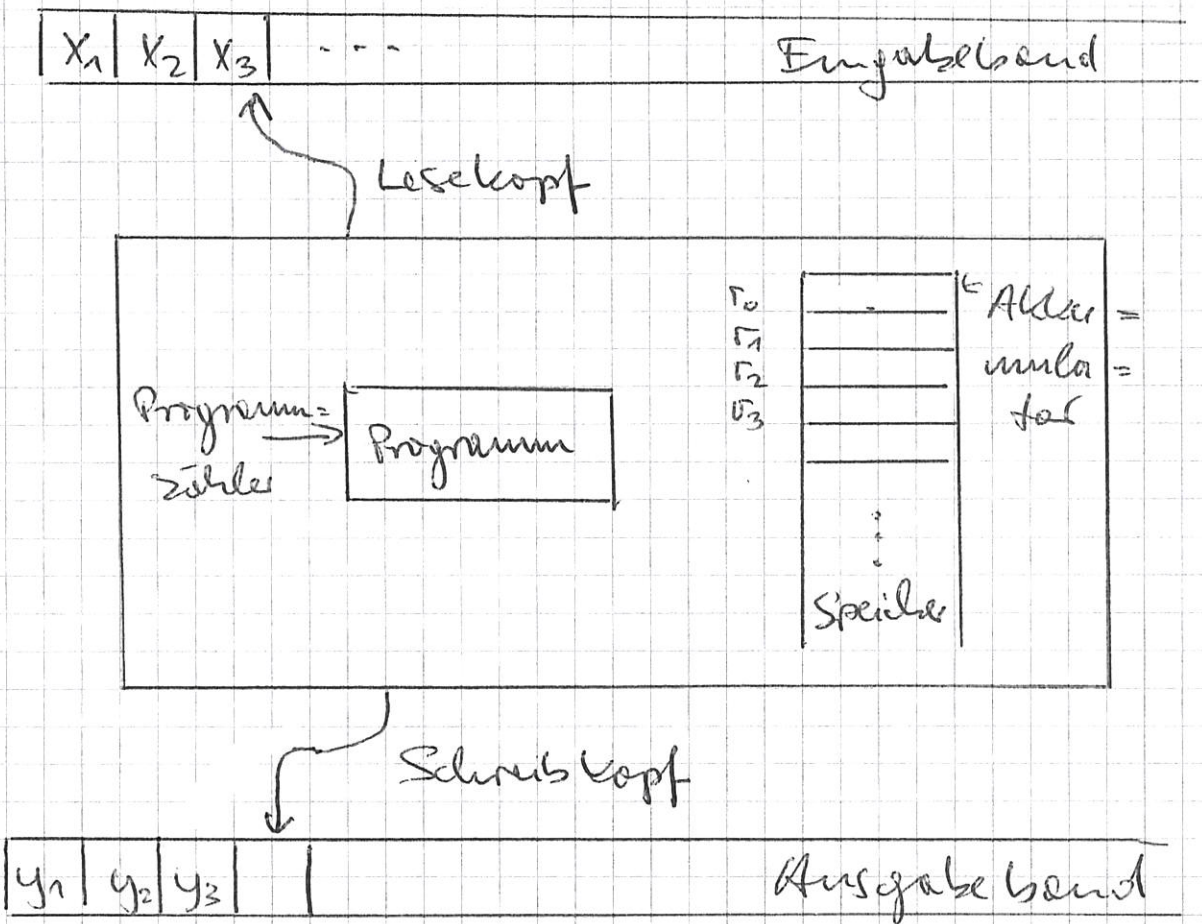
Frage:
Welche Rolle spielt unser "primitives" Rechnermodell der TM in Bezug auf die Charakterisierung der praktisch berechenbaren Funktionen?

- Sei M eine k-Band TM.
  Von M bei Eingabe x benötigte Zeit
  $\hat{=}$ Anzahl der Rechenschritte
  Von M bei Eingabe x benötigte Platz
  $\hat{=}$ Anzahl der unterschiedlichen Bandquadrate, die M während der Rechnung besucht.

- Seien $s, t : \mathbb{N}_0 \to \mathbb{N}_0$ Funktionen.
  M heißt t(·)-zeitbeschränkt, falls M für alle Eingaben x nach maximal $t(|x|)$ Schritten anhält.
  M heißt s(·)-platzbeschränkt, falls M bei jeder Eingabe x maximal $s(|x|)$ Platz benötigt.

6.1 Die Random Access Maschine

Random Access Maschine (RAM) modelliert Computer, der seine Instruktionen selbst nicht modifizieren darf.



Eigenschaften:

- Lese- und Schreibkopf bewegt sich nur von links nach rechts.
- Start: • L-Kopf und S-Kopf auf 1. $\square$
  • Ausgabeband enthält nur $\sqcup$.
- Wird eine ganze Zahl ausgedruckt $\rightsquigarrow$ S-Kopf ein Bandquadrat weiter

- **Speicher** besteht aus beliebig langer Folge $r_0, r_1, \ldots, r_i, \ldots$ von Registern

  r_i kann beliebige ganze Zahl aufnehmen.

Modell sinnvoll, falls

1) Größe des Problems klein genug, so dass es ausschließlich im Hauptspeicher eines Computers gelöst werden kann und
2) verwendeten Zahlen klein genug sind, so dass diese in ein Wort eines Computers passen.

- **Programm** nicht im Speicher abgelegt
  $\Rightarrow$ kann sich selbst nicht modifizieren.

- Besteht aus Folge von eventuell markierten Instruktionen. Es gibt
  - Eingabe / Ausgabe (E/A) - Instruktionen
  - arithmetische Instruktionen
  - indirekte Adressierung
  - bedingte Sprünge
- r_0 dient als **Akkumulator**.
- Instruktion besteht aus **Operationscode** und **Adresse**, welche Operanden adressiert.

Tabelle 2.1: Auflistung der RAM-Instruktionen.

	Operationscode	Adresse
1	LOAD	Operand
2	STORE	Operand
3	ADD	Operand
4	SUB	Operand
5	MULT	Operand
6	DIV	Operand
7	READ	Operand
8	WRITE	Operand
9	JUMP	Marke
10	JGTZ	Marke
11	JZERO	Marke
12	HALT	

Operand:

$=i$	ganze Zahl i selbst
i	Inhalt des Registers r_i
$*i$	indirekte Adressierung, d.h., Inhalt des Registers r_j, wobei j diejenige ganze Zahl ist, die im Register r_i steht.

Bezeichnung:

- $c(i)$ Inhalt von r_i

Start: $c(i) = 0 \quad \forall\, i \geq 0.$

- a Operand: $v(a)$ Wert von a. D.h.,

$v(=i) = i,$

$v(i) = c(i),$

$v(*i) = c(c(i)).$

Tabelle 2.2: Effekt der RAM-Instruktionen.

	Instruktion	Effekt
1.	LOAD a	$c(0) := v(a)$
2.	STORE i	$c(i) := c(0)$
	STORE $*i$	$c(c(i)) := c(0)$
3.	ADD a	$c(0) := c(0) + v(a)$
4.	SUB a	$c(0) := c(0) - v(a)$
5.	MULT a	$c(0) := c(0) \cdot v(a)$
6.	DIV a	$c(0) := \lfloor c(0)/v(a) \rfloor$
7.	READ i	$c(i) :=$ aktuelles Eingabesymbol
	READ $*i$	$c(c(i)) :=$ aktuelles Eingabesymbol
8.	WRITE a	Drucke $v(a)$
9.	JUMP b	Der Programmzähler wird auf die Instruktion mit Markierung b gesetzt
10.	JGTZ b	**if** $c(0) > 0$ **then** JUMP b
11.	JZERO b	**if** $c(0) = 0$ **then** JUMP b
12.	HALT	Beende die Berechnung

Kosten einer RAM-Instruktion:

Ausführungszeit setzt sich zusammen aus

1. Zeit für Bereitstellung der Operanden (Speicherzugriffszeit) und
2. Zeit für die Ausführung der RAM-Instruktion (d.h. für den Leerbefehl).

Einheitskostenmaß: (unabhängig von der Größe der Zahlen)

1 pro Speicherzugriff

1 pro Ausführung eines Leerbefehls.

Logarithmisches Kostenmaß:

Kosten proportional zur Länge $L(\cdot)$ der Darstellung der Zahlen. Bei Binärdarstellung ergibt sich für $L(n)$ der Zahl n:

$$L(n) = \begin{cases} 1 & \text{falls } n = 0 \\ \lfloor \log n \rfloor + 1 & \text{sonst} \end{cases}$$

Einheitskostenmaß ist nur dann realistisch, wenn ein Programm nur Zahlen verwendet, die in eine Speicherzelle eines realen Computers passen.

Tabelle 2.3: Kosten für die Bereitstellung des Operanden.

Operand a	Kosten $t(a)$
$= i$	$L(i)$
i	$L(i) + L(c(i))$
$*i$	$L(i) + L(c(i)) + L(c(c(i)))$

Tabelle 2.4: Kosten für die Ausführung der RAM-Instruktion.

1	LOAD a	$t(a)$
2	STORE i	$L(c(0)) + L(i)$
	STORE $*i$	$L(c(0)) + L(i) + L(c(i))$
4	ADD a	$L(c(0)) + t(a)$
4	SUB a	$L(c(0)) + t(a)$
5	MULT a	$L(c(0)) + t(a)$
6	DIV a	$L(c(0)) + t(a)$
7	READ i	$L(\text{Eingabe}) + L(i)$
7	READ $*i$	$L(\text{Eingabe}) + L(i) + L(c(i))$
8	WRITE a	$t(a)$
9	JUMP b	$L(b)$
10	JGTZ b	$L(c(0)) + L(b)$
11	JZERO b	$L(c(0)) + L(b)$
12	HALT	1

Eine TM kann effizient durch eine RAM simuliert werden. Übung

Bemerkung:
Bei Verwendung des Einheitskostenmaßes kann eine TM eine RAM nicht mit lediglich polynomiellen Zeitverlust simulieren, da

- gestattet mit Zahl 2 in Akku kann RAM mit n Multiplikationen die Zahl 2^{2^n} erzeugen.
- TM benötigt $\Omega(2^n)$ Bandquadrate um 2^{2^n} zu speichern und somit auch $\Omega(2^n)$ Zeit, um diese Zahl hinzuschreiben.

Satz 1

Sei f eine Funktion, die durch eine RAM R in Zeit $t()$ unter dem logarithmischen Kostenmaß berechnet wird. Dann gibt es eine 6-Band TM M, die f in Zeit $O(t^2())$ berechnet.

Beweis:

Struktur von M:

- Je ein Band zur Simulation des Eingabe- und des Ausgabebandes
- Ein Band zur Simulation des Speichers von R.
- Ein Band für den Akkumulator
- Ein Band für den Operanden
- Ein Band als „Schmierpapier".

Sei c_j Inhalt des Registers i_j, $1 \leq j \leq k$.

##	i_1	#	c_1	##	i_2	#	c_2	##	···	##	i_k	#	c_k	###	···

Abbildung 2.2: Speicherband der DTM M, die die RAM R simuliert.

Simulation der RAM-Instruktionen:

- **Speicherzugriff auf Register I.**

  TM läuft Speicherband von links nach rechts, bis ##i_j# mit $i_j = I$ gefunden.

  LOAD I — c_j wird aufs Akku-Band kopiert.

  STORE I — M kopiert alles, was rechts von ##i_j#c_j auf dem Band steht, aufs "Schmierband", überschreibt c_j mit dem Inhalt des Akku-Bandes und kopiert Inhalt des Schmierbands zurück. (entpressen)

  Falls kein i_j mit $i_j = I$ existiert, dann wird das letzte # mit I#c###, wo bei c Inhalt des Akku-Bands ist, überschrieben.

benötigte Zeit: $O(t(n))$

Annahme: Operand steht auf Operandenband

- READ und WRITE-Operationen wie Speicherzugriff

  Zeit: $O(t_c)$

- bedingte Sprünge geschehen in endlicher Kontrolle, also

  $O(1)$

- arithmetische Operationen: (Seien k_1 und k_2 die Längen der Operanden)

  - Addition, Subtraktion kann unter Verwendung der Schulmethode, wobei das Ergebnis auf dem Schmierpapier geschrieben wird, in Zeit $O(k_1 + k_2)$ von M ausgeführt werden.

    genaue Übung

  - Multiplikation, Division kann unter Verwendung der Schulmethode von M in Zeit $O(k_1 \cdot k_2)$ ausgeführt werden.

    genaue Übung

Die RAM benötigt für jede Operation mindestens die Zeit $k_1 + k_2$.

Wegen $a \cdot b \leq a^2 + b^2 \leq (a+b)^2$ gilt für die benötigte Gesamtzeit zur Durchführung aller Multiplikationen bzw. Divisionen

$A_{MD} \cdot O(t(n)) + O(t^2(n))$

A_{MD} → # Mult/Div

$O(t(n))$ ↑ Speicherzugriff

$O(t^2(n))$ ↑ Ausführung

R führt höchstens $t(n)$ Operationen aus.

$\Rightarrow$

$GT(n) \leq O(t^2(n)) + O(t(n)) + O(t^2(n))$

↑ Gesamtzeit; ↑ Speicherzugriff; ↑ # Mult/Div; ↑ Mult/Div

$= O(t^2(n))$.

□

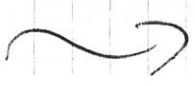
15.2 Die Sprachklassen P und NP.

- Beschränkung auf Berechnung der charakteristischen Funktion c_L von Sprache $L \subseteq \Sigma^*$

$P = \{ L \mid \exists$ Polynom p_L und eine $p_L(\cdot)$-zeitbeschränkte DTM, die c_L berechnet $\}$

DTM $M = (Q, \Sigma, \delta, q_0, F)$ heißt <u>Akzeptor</u>, falls $F = \{q_+, q_-\}$. Wir sagen, M <u>akzeptiert</u> (<u>verwirft</u>) die Eingabe x, falls M bei Eingabe x im Zustand q_+ (q_-) anhält.

$L(M) = \{ x \mid M$ akzeptiert $x \}$ ist die von M akzeptierte Sprache.



$P = \{ L \mid L = L(M)$ für einen polynomiell zeitbeschränkten Akzeptor $M \}$.

Sei $L \subseteq \Sigma^*$ und $L \in P$.

$\Rightarrow$

$\exists$ Programm Π_L, das für $x \in \Sigma^*$ in bzgl. $|x|$ polynomieller Zeit den Wert $c_L(x)$ und einen Beweis für die Korrektheit von $c_L(x)$ konstruiert.

Bemerkung:

In der Praxis treten häufig Sprachen $L \subseteq \Sigma^*$ auf, für die gilt:

1. Es ist kein Programm Π_L mit obiger Eigenschaft bekannt.
2. Bekannt ist ein Programm Π_L', das für alle $x \in \Sigma^*$ und Beweiskandidaten $B(x)$ für $c_L(x) = 1$ in bzgl. $|x|$ polynomieller Zeit überprüft, ob $B(x)$ ein korrekter Beweis für $c_L(x) = 1$ ist.

Bsp.:

$G = (V, E)$ ungerichteter Graph, $k \in \mathbb{N}$.
$V' \subseteq V$ heißt k-Clique von G, falls $|V'| = k$ und $(i,j) \in E \ \forall i,j \in V'$, $i \neq j$.

Definiere

$$\text{Clique} = \{ x \mid x \text{ ist Kodierung eines ungerichteten Graphen } G=(V,E) \text{ und } k \in \mathbb{N} \text{ mit } G \text{ enthält } k\text{-Clique} \}.$$

- <u>Mögliche Kodierung:</u>

$x = |V| \# \text{Liste der Kanten} \# k$, wobei

- Knoten von 1 bis $|V|$ durchnummeriert und Zahlen binär dargestellt werden.

$\Rightarrow$

$$\Sigma = \{0, 1, \#, (,)\}.$$

- Bisher ist kein polynomieller Algorithmus für C_{clique} bekannt.

- Beweiskandidat $B(x)$:

  Angabe von $V' \subseteq V$ mit $|V'| = k$

- Überprüfung ob $B(x)$ korrekt kann durch Test, ob $\forall i,j \in V' \; (i,j) \in E$ erfolgen.

- Alle bereit bekannten Alg. zur Berechnung von C_{clique} überprüfen in wesentlichen nachlesende jeden Beweiskandidaten.

- Für $k \approx \frac{|V|}{2}$ existiert exponentiell viele Beweiskandidaten

  $\Rightarrow$ Laufzeit exponentiell

Annahme:

Wir hätten einen Mechanismus, der irgendwie einen Beweiskandidaten $B(x)$ in Zeit $\mathcal{O}(|B(x)|)$ hinschreibt und diese auch ein korrektes Beweis für $c_L(x) = 1$ ist, falls solcher existiert.

$\Rightarrow$

$\llcorner$ kann durch derartigen Algorithmus in Polynomzeit entschieden werden.

$\rightsquigarrow$

Erweiterung unsere Turingmaschinenmodelle mittels Einarbeitung eines derartigen Mechanismus.

Eine <u>nichtdeterministische k-Band TM</u> (NTM) ist ein 5-Tupel $M = (Q, \Sigma, \delta, q_0, F)$, wobei Q, Σ, q_0, F wie bei DTM definiert sind und δ eine Funktion

$$\delta: Q \times \Sigma^k \to 2^{Q \times \Sigma^k \times \{-1,0,+1\}^k}$$

ist.

Interpretation von

$$\delta(q, a_1, a_2, \ldots, a_k) =$$

$$\{(q'_1, c_{11}, \ldots, c_{1k}, s_{11}, \ldots, s_{1k}), \ldots,$$

$$(q'_t, c_{t1}, \ldots, c_{tk}, s_{t1}, \ldots, s_{tk})\}$$

ist die folgende:

Liest M in Zustand q für $1 \le i \le k$ auf dem i-ten Band das Zeichen $a_i \in \Sigma$, dann kann M ein beliebiges der t-Tupel in $\delta(q, a_1, \ldots, a_k)$ auswählen und geht dann wie eine DTM in die durch dieses Tupel definierte Folgekonfiguration über.

Eine NTM $M = (Q, \Sigma, \delta, q_0, F)$ mit $F = \{q_+, q_-\}$ heißt <u>nichtdeterministischer Akzeptor</u> (NA).

$L(M) = \{ x \mid \exists$ akzeptierende Berechnung von M auf Eingabe $x \}$

ist die <u>von M akzeptierte Sprache</u>.

Sei M ein NA. Die von M auf der Eingabe x benötigte Zeit $T_M(x)$ ist definiert durch

$$T_M(x) = \begin{cases} \min\{ t \mid \exists \text{ akzeptierende Berechnung der Länge } t \text{ von M auf Eingabe } x \} & \text{falls } x \in L(M) \\ 0 & \text{sonst} \end{cases}$$

Sei $t: \mathbb{N} \to \mathbb{N}$ eine Funktion. M heißt $t(\cdot)$-zeitbeschränkt, falls für alle Eingaben x gilt: $T_M(x) \le t(|x|)$.

$NP = \{ L \mid L = L(M)$ für einen polynomiell zeitbeschränkten NA $M \}$.

Erkläre, dass obige Mechanismus mit NA implementiert werden kann.

Offensichtlich $P \subseteq NP$.

Offenes Problem: gilt $P = NP$?

Üblicherweise werden Sprachen als Probleme mit ja/nein-Antwort formuliert.

Bsp.:

Cliqueproblem

gegeben: ungerichteter Graph $G = (V, E)$, $k \in \mathbb{N}$
Frage: Enthält G eine k-Clique?

Praxis:

Problem häufig nicht als solches mit ja/nein-Antwort, sondern als ein Optimierungs- oder ein funktionales Problem. Diese können jedoch in der Regel in Polynomzeit auf Problem mit ja/nein-Antwort zurückgeführt werden.

Bsp.:

Problem des Handlungsreisenden

Gegeben: n Städte $\{1, 2, \ldots, n\}$ und für alle $i, j \in \{1, 2, \ldots, n\}$ Kosten $c(i,j) \in \mathbb{N}_0$ für die direkte Reise von i nach j.

Gesucht: Eine kostengünstigste Rundreise (d.h., jede Stadt wird genau einmal betreten und genau einmal verlassen).

Eingabe: Graph $G = (V, E)$ mit $V = \{1, 2, \ldots, n\}$, $E \subseteq V \times V$ und Abbildung $c : E \to \mathbb{N}_0$

Formulierung als Problem mit ja/nein-Antwort:

Gegeben: $G = (V, E)$, $c : E \to \mathbb{N}_0$, $t \in \mathbb{N}_0$

Frage: Gibt es eine Rundreise mit Gesamt-Kosten $\leq t$?

Formulierung als Optimierungsproblem

Gegeben: $G = (V, E)$, $c : E \to \mathbb{N}_0$

Problem: Bestimme die Kosten einer kostengünstigsten Rundreise

Reduktion auf ja/nein:

Sei $t = \sum_{e \in E} c(e)$.

Idee: Binärsuche auf $[0 .. t]$

$\rightsquigarrow$

(1) MIN := 0 ; MAX := t ;

(2) while MAX − MIN ≥ 1

do

$t := \lfloor \frac{MIN + MAX}{2} \rfloor$;

Löse " Gibt es eine Rundreise der Länge $\leq t$ "

if ja

then

MAX := t

else
MIN := t+1
fi
od.

Analyse:

ja/nein : $\lceil \log k \rceil$

Wegen $\log(a+b) \leq \log a + \log b$ für $a,b \geq 2$ gilt:

$$\log k \leq \sum_{e \in E} \log c(e)$$

$\Rightarrow$

$\log k$ ist durch die Länge der Eingabe beschränkt.

$\Rightarrow$

(Existenz polynomieller Alg. für ja/nein
$\Rightarrow$ " " " " Opt. problem

Formulierung als funktionales Problem

Gegeben: $G = (V, E)$, $c: E \to \mathbb{N}_0$
Problem: Bestimme Rundreise minimaler Kosten.

Reduktion auf ja/nein:

(1) Bestimme unter Verwendung des Optimierungsproblems die Kosten t_0 einer kostengünstigsten Rundreise;

(2) $E' := E$;
for alle $e \in E$
do
$E' := E' \setminus \{e\}$;
Löse "Gibt es eine Rundreise mit Gesamtkosten t_0 in $G' := (V, E')$?";
if nein
then
$E' := E' \cup \{e\}$
fi
od.

Nach der Terminierung definiert E' gerade eine Rundreise der Länge t_0.

.3 NP-vollständige Probleme

Ziel:
Definition von "schwierigste" Probleme in NP unter Verwendung von Reduktion.

Seien Σ und Γ Alphabete. $f: \Sigma^* \to \Gamma^*$ heißt polynomielle Transformation, falls f von einer polynomiell zeitbeschränkten DTM berechnet werden kann.

Seien $L_1 \subseteq \Sigma^*$ und $L_2 \subseteq \Gamma^*$.

L_1 heißt <u>polynomiell reduzierbar auf</u> L_2

$L_1 \leq_{pol} L_2 :\Leftrightarrow$

$\exists$ polynomielle Transformation $f: \Sigma^* \to \Gamma^*$ mit:

$$x \in L_1 \Leftrightarrow f(x) \in L_2 \quad \forall x \in \Sigma^*$$

L heißt <u>NP-hart</u>, falls $\forall L' \in NP : L' \leq_{pol} L$

L heißt <u>NP-vollständig</u>, falls

1. $L \in NP$ und
2. L NP-hart ist.

<u>Satz 8.2</u>

Sei L_0 NP-vollständig. Dann gilt:

a) $L_0 \in P \Leftrightarrow P = NP$ und

b) $L_0 \leq_{pol} L_1 \wedge L_1 \in NP \Rightarrow L_1$ ist NP-vollständig

<u>Beweis:</u>

a) $\Leftarrow$ a

$P = NP \Rightarrow L_0 \in P$ ✓

$\Rightarrow$ a

Sei $L_0 \in P$. $\Rightarrow$

$\exists$ Polynom $p(\cdot)$ und $p(\cdot)$-zeitbeschränkte DTM M mit $L(M) = L_0$.

Betrachte $L \in NP$ beliebig.

L_0 NP-vollständig $\Rightarrow$

$\exists$ polynomielle Transformation f, so dass

$$x \in L \Leftrightarrow f(x) \in L_0.$$

Sei M' polynomiell zeitbeschränkte DTM, die f berechnet. Sei das Polynom $q(\cdot)$ eine obere Schranke für die Laufzeit von M'.

Ziel:
Konstruktion einer polynomiell zeitbeschränkten DTM M'' mit $L(M'') = L$.

Arbeitsweise von M'':

(1) Auf Eingabe x verhält sich M'' wie M' und berechnet $f(x)$ in $\leq q(|x|)$ Schritten.

(2) In maximal $q(|x|)$ Schritten geht M'' in die Startkonfiguration von M bei Eingabe $f(x)$ über.

(3) M'' entscheidet mit Hilfe von M in $p(|f(x)|)$ Schritten, ob $f(x) \in L_0$.

(4) M'' akzeptiert x genau dann, wenn $f(x) \in L_0$.

$\rightsquigarrow$

$L(M'') = L$. ✓

Ferner gilt:

$$\begin{aligned} T_{M'}(x) &\leq 2 \cdot q(|x|) + p(|f(x)|) \\ &\leq 2 \cdot q(|x|) + p(q(|x|)) \\ &\leq r(x), \end{aligned}$$

für ein Polynom $r(\cdot)$. Also gilt $L \in P$.

b) Sei $L \in NP$ beliebig.

z.z. $L \leq_{pol} L_1$

NP-Vollständigkeit von L_0 $\Rightarrow$

$\exists$ poly. Transformation f mit

$x \in L \Leftrightarrow f(x) \in L_0$.

$L_0 \leq_{pol} L_1$ $\Rightarrow$

$\exists$ poly Transformation g mit

$y \in L_0 \Leftrightarrow g(y) \in L_1$.

Also gilt:

$x \in L \Leftrightarrow f(x) \in L_0 \Leftrightarrow g(f(x)) \in L_1$.

$\Rightarrow$

$g \circ f$ transformiert L auf L_1.

g und f in Polynomzeit berechenbar

$\Rightarrow$

$g \circ f$ kann auch durch eine polynomiell zeitbeschränkte DTM berechnet werden.

Implikationen des Satzes 1.2:

- NP-vollständige Probleme sind die schwierigsten Probleme in NP.
- b) impliziert folgende einfache Technik, um die NP-Vollständigkeit einer Sprache zu beweisen:

1. Zeige für eine Sprache $L_0 \in NP$, dass L_0 NP-vollständig ist. D.h., zeige $L \leq_{pol} L_0$ $\forall L \in NP$.

2. Um für eine Sprache L_1 die NP-Vollständigkeit zu beweisen, genügt es dann, $L_1 \in NP$ und $L_0 \leq_{pol} L_1$ zu zeigen. L_0 kann durch eine beliebige Sprache L_0', für die die NP-Vollständigkeit bereits bewiesen ist, ersetzt werden.

Definition des Erfüllbarkeitsproblem der Aussagenlogik:

Bezeichnungen

- $V = \{x_1, x_2, \dots\}$ unendliche Menge von (aussagenlogischen) Variablen.
- Sei $x_i \in V$. x_i und $\bar{x}_i$ Literale
- Seien $y_1, y_2, \dots, y_k$ Literale. $(y_1 \vee y_2 \vee \dots \vee y_k)$ Klausel vom Grad k.
- $c_1, c_2, \dots, c_m$ Klauseln vom Grad $\leq k$,

$c_1 \wedge c_2 \wedge \dots \wedge c_m$ Ausdruck in konjunktiver

Normalform mit höchstens k Literalen pro Klausel.

- $\psi : V \to \{0,1\}$ Belegung oder Existenz der Variablen

$0 \hat{=}$ false $\quad 1 \hat{=}$ true

Erweiterung von ψ:

$$\psi(y) = \begin{cases} \psi(x_i) & \text{falls } y = x_i \\ 1 - \psi(x_i) & \text{falls } y = \overline{x_i} \end{cases}$$

$$\psi(y_1 \vee y_2 \vee \dots \vee y_\ell) = \max\{\psi(y_i) \mid 1 \le i \le \ell\}$$

$$\psi(c_1 \wedge c_2 \wedge \dots \wedge c_m) = \min\{\psi(c_j) \mid 1 \le j \le m\}.$$

Ausdruck α heißt erfüllbar, falls Belegung ψ mit $\psi(\alpha) = 1$ existiert. SAT (SAT(k)) bezeichnet die Menge aller erfüllbaren Ausdrücke in konjunktiver Normalform (mit höchstens k Literalen pro Klausel). Das Problem, ob ein gegebener Ausdruck in konjunktiver Normalform erfüllbar ist, d.h., in SAT liegt, heißt Erfüllbarkeitsproblem der Aussagenlogik.

Satz 6.3

SAT ist NP-vollständig.

Beweis:

a) $SAT \in NP$ ✓

b) z.z. $L \leq_{pol} SAT \quad \forall L \in NP$.

Sei $L \in NP$ beliebig.

$\Rightarrow$ Konstruktion Beweis Le 5.7.

$\exists$ $p(\cdot)$ - zeitbeschränkter Einband-NA M mit $L(M) = L$ für ein Polynom $p(\cdot)$.

Sei $M = (Q, \Sigma, \delta, q_0, \{q_+, q_-\})$ mit

- $Q = \{q_0, q_1, \ldots, q_s\}$, wobei $q_{s-1} = q_-$, $q_s = q_+$
- $\Sigma = \{a_1, a_2, \ldots, a_r\}$, wobei $a_1 = \sqcup$ und $a_2 = \$$
- der Eigenschaft, dass sich Haltekonfigurationen sich unendlich oft wiederholen. Wir erreichen dies, indem wir δ wie folgt erweitern

  Falls $\delta(q_i, a_j) = \emptyset$, dann füge $\delta(q_i, a_j) = \{(q_i, a_j, 0)\}$ hinzu, $0 \leq i \leq s$, $1 \leq j \leq r$.

Sei $x \in \Sigma^*$ eine beliebige Eingabe von M mit $|x| = n$. Dann gilt

$x \in L(M) \Leftrightarrow \exists$ Folge $K_0, K_1, \ldots, K_{p(n)}$ von Konfigurationen, so dass

1. K_0 ist die Startkonfiguration von M bei Eingabe x.

2. K_{i+1} ist eine Folgekonfiguration von K_i für $0 \leq i \leq p(n)-1$ und

3. der Zustand in $K_{p(n)}$ ist q_s.

Die Übergangsfkt. $\delta: Q \times \Sigma \to 2^{Q \times \Sigma \times \{-1,0,1\}}$ definiert eine Relation aus $(Q \times \Sigma) \times (Q \times \Sigma \times \{-1,0,1$

Nummeriere Tupel dieser Relation in beliebiger Reihenfolge durch. Sei m die Anzahl dieser Tupel.

Ziel:

Konstruktion eines Ausdruckes $\alpha(x)$ mit:

$\alpha(x)$ erfüllbar $\Leftrightarrow$ M akzeptiert Eingabe x.

Tabelle 2.5: Zur Simulation benötigte Variablen.

Variablen	beabsichtigte Bedeutung
$q_{t,k} \quad 0 \leq t \leq p(n), 0 \leq k \leq s$	$q_{t,k} = 1 \iff q_k$ ist der Zustand von K_t.
$a_{t,i,j} \quad 0 \leq t, i \leq p(n), 1 \leq j \leq r$	$a_{t,i,j} = 1 \iff a_j$ ist der Inhalt des i-ten Bandquadrates in K_t.
$s_{t,i} \quad 0 \leq t, i \leq p(n)$	$s_{t,i} = 1 \iff$ In K_t steht der L/S-Kopf auf Bandquadrat i.
$b_{t,l} \quad 0 \leq t < p(n), 1 \leq l \leq m$	$b_{t,l} = 1 \iff$ Das Tupel l findet Anwendung beim Übergang von K_t nach K_{t+1}.

Ziel:

Definition von Ausdrücken, die den Variablen die beabsichtigte Bedeutung geben.

Die Ausdrücke müssen folgende Bedingungen sicherstellen:

1. Startbedingungen

K_0 ist Startkonfiguration. D.h.,

q_0 Zustand, L/S-Kopf auf 1. Bandquadrat und $\$x_{\sqcup}^{p(n)-n}$ Bandinhalt.

2. Randbedingungen

$\forall K_t$, $0 \leq t \leq p(n)$ gilt:

- M ist in genau einem Zustand,
- der L/S-Kopf steht auf genau einem Bandquadrat,
- jedes Bandquadrat enthält genau ein Zeichen und
- falls $t < p(n)$, dann findet beim Übergang nach K_{t+1} genau ein Tupel der Übergangsrelation seine Anwendung.

3. Übergangsverhalten

Für $0 \leq t < p(n)$ ergibt sich der Zustand, die Position des L/S-Kopfes und die Beschriftung von K_{t+1} aus K_t und dem Tupel, das beim Übergang von K_t nach K_{t+1} seine Anwendung findet.

Seien S, R und U $\hat{=}$ Ausdrücke, die die Start-, Randbedingungen und Übergangsverhalten sicherstellen. Dann definieren wir:

$$\alpha(x) = S \wedge R \wedge U \wedge q_{p(n),s}.$$

Ziel:
Konstruktion von S, R und U.

Sei $x = a_{j_1} a_{j_2} \dots a_{j_n}$.

$$\rightsquigarrow \quad S = q_{0,0} \wedge s_{0,1} \wedge a_{0,1,j_1} \wedge a_{0,2,j_2} \wedge \dots \wedge a_{0,n,j_n} \wedge a_{0,n+1,1} \wedge \dots \wedge a_{0,p(n),1}$$

Anzahl Literale: $p(n)+3$.

Wir möchten für eine gegebene Variablenmenge $\{x_1, x_2, \dots, x_l\}$ ausdrücken, dass genau eine dieser Variablen wahr ist. Es gilt:

$$\text{Genau-eine}(x_1, \dots, x_l) = \text{Mindestens-eine}(x_1, \dots, x_l) \wedge \text{Höchstens-eine}(x_1, \dots, x_l),$$

wobei

$$\begin{aligned}
\text{Mindestens-eine}(x_1, \dots, x_l) &= x_1 \vee \dots \vee x_l \text{ und} \\
\text{Höchstens-eine}(x_1, \dots, x_l) &= \neg\text{Mindestens-zwei}(x_1, \dots, x_l) \\
&= \neg\Big(\bigvee_{1 \le i < j \le l} (x_i \wedge x_j)\Big) \\
&\underset{\text{de Morgan}}{=} \bigwedge_{1 \le i < j \le l} (\overline{x}_i \vee \overline{x}_j).
\end{aligned}$$

Der Ausdruck Genau-eine$(x_1, \dots, x_l)$ ist in konjunktiver Normalform und enthält $l + 2\sum_{j=1}^{l-1} j = l^2$ Literale.

Seien

$$\begin{aligned}
R_{\text{Zustände}}(t) &= \text{Genau-eine}(q_{t,0}, q_{t,1}, \dots, q_{t,s}) \\
R_{\text{Position}}(t) &= \text{Genau-eine}(s_{t,1}, s_{t,2}, \dots, s_{t,p(n)}) \\
R_{\text{Band}}(t) &= \bigwedge_{0 \le i \le p(n)} \text{Genau-eine}(a_{t,i,1}, \dots, a_{t,i,r}) \\
R_{\text{Übergang}}(t) &= \text{Genau-eine}(b_{t,1}, b_{t,2}, \dots, b_{t,m}).
\end{aligned}$$

Alle Ausdrücke sind in konjunktiver Normalform und enthalten insgesamt

$$(s+1)^2 + p^2(n) + p(n)r^2 + m^2$$

Literale. Ferner stellt

$$R = \bigwedge_{0 \le t \le p(n)} (R_{\text{Zustände}}(t) \wedge R_{\text{Position}}(t) \wedge R_{\text{Band}}(t) \wedge R_{\text{Übergang}}(t))$$

die Randbedingungen sicher.

Anzahl Literale:

$$(p(n)+1)\left((s+1)^2+\left(p^{2(n)}+\left(p(n)+1\right)^2 r^2+m^2\right)\right)=O(p^3(n))$$

Sei $U(t)$, $0 \leq t < p(n)$, der Ausdruck, der die Korrektheit des Übergangs von K_t nach K_{t+1} sicherstellt. Dann muß $U(t)$ konkret folgende Bedingungen gewährleisten:

1. Falls M zum Zeitpunkt t das i-te Bandquadrat nicht liest, dann ändert sich der Inhalt dieses Bandquadrates nicht.
2. Falls M zum Zeitpunkt t das i-te Bandquadrat liest und beim Übergang nach K_{t+1} das l-te Tupel $((q_{k_l}, a_{j_l}), (q_{\tilde{k}_l}, a_{\tilde{j}_l}, r_l))$ der Übergangsrelation seine Anwendung findet, dann gilt zum Zeitpunkt t:
   - q_{k_l} ist der Zustand von M und
   - a_{j_l} ist der Inhalt des i-ten Bandquadrates

   und zum Zeitpunkt $t+1$
   - $q_{\tilde{k}_l}$ ist der Zustand von M,
   - $a_{\tilde{j}_l}$ ist der Inhalt des i-ten Bandquadrates und
   - $i + r_l$ ist die Position des L/S-Kopfes.

$U(t)$ muß also Implikationen ausdrücken. Wir schreiben $x \to y$ für x impliziert y. Es gilt: $x \to y \equiv \overline{x} \vee y$. Also gewährleistet folgender Ausdruck die Bedingung 1:

$$\bigwedge_{1 \leq j \leq r} ((\overline{s}_{t,i} \wedge a_{t,i,j}) \to a_{t+1,i,j}) \equiv \bigwedge_{1 \leq j \leq r} (s_{t,i} \vee \overline{a}_{t,i,j} \vee a_{t+1,i,j})$$

Den Ausdruck für Bedingung 2 überlegt man sich analog. Dies führt zu folgendem Ausdruck in konjunktiver Normalform für $U(t)$:

$$\begin{aligned} U(t) = \bigwedge_{0 \leq i \leq p(n)} \Bigg[& \bigwedge_{1 \leq j \leq r} (s_{t,i} \vee \overline{a}_{t,i,j} \vee a_{t+1,i,j}) \\ & \wedge \bigwedge_{1 \leq l \leq m} \big[(\overline{s}_{t,i} \vee \overline{b}_{t,l} \vee q_{t,k_l}) \\ & \qquad \wedge (\overline{s}_{t,i} \vee \overline{b}_{t,l} \vee a_{t,i,j_l}) \\ & \qquad \wedge (\overline{s}_{t,i} \vee \overline{b}_{t,l} \vee q_{t+1,\tilde{k}_l}) \\ & \qquad \wedge (\overline{s}_{t,i} \vee \overline{b}_{t,l} \vee a_{t+1,i,\tilde{j}_l}) \\ & \qquad \wedge (\overline{s}_{t,i} \vee \overline{b}_{t,l} \vee s_{t+1,i+r_l}) \big] \Bigg]. \end{aligned}$$

$$\rightsquigarrow$$

$$U = \bigwedge_{0 \leq t < p(n)} U(t).$$

$U(t)$ enthält $(p(n)+1)(3r+15m)$ Literale.

$$\Rightarrow$$

U enthält $(p(n)+1)^2(3r+15m)$ Literale.

$\Rightarrow$

$\alpha(x) = S \wedge \mathcal{L} \wedge K \wedge \varphi_{p(n),S}$

enthält

$$p(n)+3 + (p(n)+1)\left((s+1)^2 + \cancel{p^2(n)} + \left(p(n)+1\right)^2 r^2 + m^2\right) + (p(n)+1)^2(3r+5m)+1$$

$$= O(p^3(n))$$

Literale.

$\Rightarrow$

Transformation kann in Polynomzeit durchgeführt werden. (Übung)

Noch z.z.:

$\alpha(x)$ erfüllbar $\Leftrightarrow x \in L$.

Bew.:

„$\Leftarrow$“

$x \in L \Rightarrow$ $\exists$ Berechnung von M, die x in $p(|x|)$ Schritten akzeptiert.

$\Rightarrow$

$\exists$ Konfigurationen $K_0, K_1, \ldots, K_{p(n)}$, $n = |x|$ so dass

i) K_0 Startkonfiguration bei Eingabe x,

ii) K_{t+1} Folgekonfiguration von K_t, $0 \leq t < p(n)$ und

iii) $K_{p(n)}$ akzeptierende Konfiguration

sind.

Definition der Einsetzung ψ:

Für $0 \le t \le p(n)$, $0 \le k \le s$, $1 \le i \le p(n)$, $1 \le j \le r$ und $1 \le l \le m$ setzen wir

$0 \le t \le p(n)$, $0 \le k \le s$

$$\psi(q_{t,k}) = \begin{cases} 1 & \text{falls } q_k \text{ Zustand in } K_t \\ 0 & \text{sonst.} \end{cases}$$

$0 \le t \le p(n)$, $0 \le i \le p(n)$, $1 \le j \le r$

$$\psi(a_{t,i,j}) = \begin{cases} 1 & \text{falls } a_j \text{ das } i\text{-te Zeichen der Bandinschrift von } K_t \\ 0 & \text{sonst.} \end{cases}$$

$0 \le t \le p(n)$, $0 \le i \le p(n)$

$$\psi(s_{t,i}) = \begin{cases} 1 & \text{falls in } K_t \text{ das } i\text{-te Band quadrat gelesen wird} \\ 0 & \text{sonst} \end{cases}$$

$0 \le t < p(n)$, $1 \le l \le m$

$$\psi(b_{t,l}) = \begin{cases} 1 & \text{falls beim Übergang von } K_t \text{ nach } K_{t+1} \text{ das } l\text{-te Tupel der Übergangsrelation verwendet wird} \\ 0 & \text{sonst} \end{cases}$$

Es ist leicht zu sehen, dass ψ den Ausdruck $\alpha(x)$ erfüllt. (Übung 2)