Komplexitätstheorie

Kapitel 4: Platzkomplexität

Einleitung

Platzverbrauch:

- der temporäre Zwischenspeicher, der während der Berechnung verwendet wird (Datenstrukturen, Rekursionsstack, etc.)
- Im Fall von TMs: die verwendeten Bandzellen außer denen für die Eingabe

In der Praxis kann Platzverbrauch so kritisch sein wie Zeitverbrauch:

- geringer Zeitverbrauch nicht hilfreich wenn Speicher vorher erschöpft
- dies passiert nicht selten bei Algorithmen, die schwierige Probleme lösen

Wesentlicher Unterschied zur Zeitkomplexität:

Platz kann man wiederverwenden, Zeit nicht

Einleitung

Überblick:

- Polynomieller Platzverbrauch
- Quantifizierte Bool'sche Formeln (QBFs)
- Savitch's Theorem
- Logarithmischer Platzverbrauch
- DTMs vs NTMs, Erreichbarkeit in Graphen
- Komplementklassen /
  Theorem von Immerman und Scelepcsenyi

Kapitel 4

Platzkomplexitätsklassen

Platzkomplexität

Um Eingabe und "Zwischenspeicher" zu trennen, nehmen wir dediziertes Eingabeband an:

- Eingabeband wird nur gelesen, nicht beschrieben

  (TM muss per Konvention das auf diesem Band gelesene Symbol wieder zurückschreiben)

- Auf dem Eingabeband kann man sich nach links und rechts bewegen, also die Eingabe auch mehrfach lesen

- Im Zusammenhang mit Platzkomplexität ist k-Band TM also TM mit k Arbeitsbändern + 1 Eingabeband

Dies ermöglicht es uns, auch Platzbeschränkungen s mit $s(n) \leq n$ zu betrachten!

Aufgrund Ihrer Bedeutung für NP betrachten wir auch NTMs!

Platzkomplexität

Definition Platzbeschränkt

Für k-Band DTM oder NTM M und $w \in \Sigma^*$ schreiben wir $\mathsf{space}_M(w) = n$ wenn alle Berechnungen von M auf w höchstens n Bandzellen verwenden. (alle Bänder aufsummiert)

Sei $s : \mathbb{N} \to \mathbb{N}$ monoton wachsende Funktion.
M ist s-*platzbeschränkt* wenn $\mathsf{space}_M(w) \leq s(n)$ für alle w der Länge n

Nun ist:

$$\mathsf{DSpace}_k(s) := \{L \subseteq \Sigma^* \mid \exists \mathcal{O}(s)\text{-platzbeschränkte terminierende } k\text{-Band DTM } M \text{ mit } L(M) = L\}$$

$$\mathsf{NSpace}_k(s) := \{L \subseteq \Sigma^* \mid \exists \mathcal{O}(s)\text{-platzbeschränkte terminierende } k\text{-Band NTM } M \text{ mit } L(M) = L\}$$

Mehrband TMs

Auch bei Platzkomplexität werden wir uns um die Anzahl Bänder keine Gedanken machen

Bandreduktion (Platzkomplexität)

Für alle $k \geq 1$ und t: $\mathsf{DSpace}_k(t) \subseteq \mathsf{DSpace}(t)$.

Beweis: exakt derselbe wie für Zeitkomplexität
(verwende ein Band mit mehreren Spuren)

Beachte: im Gegensatz zu Zeitkomplexität kein quadratischer Blowup!

Definition DSpace, NSpace

Wir setzen $\mathsf{DSpace}(s) := \bigcup_{k \geq 1} \mathsf{DSpace}_k(s)$, analog für $\mathsf{NSpace}(s)$

Kapitel 4

Polynomieller Platzverbrauch

PSpace

Analog zu polynomieller Zeit kann man polynomiellen Platz betrachten

Mehr als polynomiell viel Platz anzunehmen ist nicht realistisch

Aber auch polynomieller Platz ist schon “recht viel”:

- Betrachte Eingabe der Größe 10.000
- kubischer Zeitverbrauch (n^3): 5 Minuten auf 3Ghz Prozessor
- kubischer Platzverbrauch: 1 Terabyte Speicher nötig!

Entsprechende Komplexitätsklasse:

$$\text{PSPACE} := \bigcup_{i \geq 1} \mathsf{DSpace}(n^i)$$

PSpace

Natürliche Frage: wie verhält sich PSpace zu P, NP, ExpTime?

Theorem

Für alle (monoton wachsenden) $f : \mathbb{N} \to \mathbb{N}$:

- DTime$(f) \subseteq$ DSpace(f)
- DSpace$(f) \subseteq$ DTime$(2^{\mathcal{O}(f)})$

Es folgt P $\subseteq$ PSpace $\subseteq$ ExpTime

Für alle diese Inklusionen ist Echtheit unbekannt, es gibt also:

- P vs PSpace Problem
- PSpace vs ExpTime Problem

Man vermutet, dass alle Inklusionen echt sind. Zu NP: später mehr!

QBF

Ein "typisches" Problem in PSpace

Gültigkeit von quantifizierten Boolschen Formeln (QBFs)

So zu verstehen:

- Boolsche Formeln = AL-Formeln
- quantifiziert: zusätzliche Quantoren für Wahrheitswerte, als Präfix

Definition QBF

Eine *quantifizierte Boolsche Formel (QBF)* hat die Form

$$Q_1 p_1 \cdots Q_n p_n . \varphi$$

wobei $Q_i \in \{\forall, \exists\}$ und φ eine AL-Formel mit Variablen aus der Menge $\{p_1, \ldots, p_n\}$.

AL-Formeln dürfen hier auch die Konstanten 0,1 enthalten

QBF

Definition Gültigkeit einer QBF

QBF $Q_1 p_1 \cdots Q_n p_n \varphi$ ist *gültig* wenn

- $Q_1 = \exists$ und $Q_2 p_2 \cdots Q_n p_n \varphi[p_1/0]$ oder $Q_2 p_2 \cdots Q_n p_n \varphi[p_1/1]$ gültig
- $Q_1 = \forall$ und $Q_2 p_2 \cdots Q_n p_n \varphi[p_1/0]$ und $Q_2 p_2 \cdots Q_n p_n \varphi[p_1/1]$ gültig
- $n = 0$ und φ (welches dann variablenfrei ist) zu 1 auswertet.

Wobei: $\varphi[p/0]$ ist φ mit p ersetzt durch 0, analog für $\varphi[p/1]$

Z.B.: $(p_1 \vee p_2) \to (p_2 \vee p_3)[p_2/0] \quad = \quad (p_1 \vee 0) \to (0 \vee p_3)$

Gültigkeit überprüfen durch Auswertungsbaum!

Auswertungsbäume

Auswertungsbaum für QBF $\psi = Q_1 p_1 \cdots Q_n p_n \varphi$:

- binärer Baum der Tiefe n
- Wurzel korrespondiert zu ψ
- Übergang zu Nachfolger entspricht Elimination eines Quantors durch Festlegen des Variablenwertes
- Blätter entsprechen also AL-Formeln ohne Variablen

Erfolgreicher Auswertungsbaum:

- ”Hochpropagieren” der Auswertung:
- existentielle Quantoren = or; universelle = and (and/or-Baum)
- Wurzel muß zu 1 evaluieren

Auswertungsbaum ist "Beweis" für Gültigkeit wie in Def von NP, aber exponentiell groß!

QBF

Theorem

QBF ist in PSpace

Idee:

- verwende rekursive Prozedur, die sich aus Definition von Gültigkeit ergibt
- Rekursionstiefe ist linear, für jeden Rekursionsschritt wird linear viel Speicher gebraucht
- das ergibt quadratischen Platzbedarf, da man den Speicher wieder-verwenden kann, wenn man in anderen Ast des (exponentiell großen) Rekursionsbaumes absteigt.

Kapitel 4

Savitch's Theorem

Savitch's Theorem

Def. PSpace ähnelt P; was ist Platz-Klasse mit Def. ähnlich NP?

Nach alternativer Charakterisierung von NP folgendes:

$$\mathrm{NPS{\small PACE}} := \bigcup_{i \geq 1} \mathsf{NSpace}(n^i)$$

Interessanter Gegensatz:

- P vs NP ist wichtigstes offenes Problem der Informatik
- PSpace vs NPSpace wurde 1970 von W. Savitch gelöst

Grundlage: Konfigurationsgraphen

Savitch's Theorem

Definition Konfigurationsgraph

Sei M eine (1-Band) NTM und $s \in \mathbb{N}$. Dann ist $\mathsf{Conf}_{M,s}$ die Menge aller Konfigurationen von M der Länge s.

Der s-*Konfigurationsgraph* für M ist der gerichtete Graph $G_{M,s} = (V, E)$ mit

- $V = \mathsf{Conf}_{M,s}$
- $E = \{(C, C') \mid C \vdash_M C'\}$

Dies erlaubt es uns, Akzeptanz als Erreichbarkeit (GAP) zu verstehen!

Lemma

Sei M s-platzbeschränkte NTM und w Eingabe der Länge n

$w \in L(M)$ gdw. eine akzeptierende Konfiguration von $q_0 w$ in $G_{M,s(n)}$ erreichbar ist.

Savitch's Theorem

Naiver Versuch PSpace = NPSpace:

- Wenn $L \in$ NPSpace, dann gibt es p-platzbeschränkte NTM M mit $L(M) = L$, p Polynom
- Gesucht: polyplatzbeschränkte DTM, die entscheidet, ob $w \in L(M)$
- Konstruiere Konfigurationsgraph $G_{M,k}$ mit $k = p(|w|)$, entscheide ob akzeptierende Konfiguration erreichbar von $q_0 w$

Problem: $G_{M,k}$ hat Größe $|Q| \cdot |\Gamma|^k \cdot k \in 2^{\mathcal{O}(p(|w|))}$ (exponentiell!)

mögliche Zustände, Bandinhalte, Kopfpositionen

Zentrale Idee von Savitch: Entscheide Erreichbarkeit in $G_{M,k}$

ohne den ganzen Graph auf einmal zu berechnen (nur poly große Teilstücke)

Savitch's Theorem

Savitch's Resultat sagt sogar noch mehr als PSpace = NPSpace

Definition Platzkonstruierbar

Funktion $s : \mathbb{N} \to \mathbb{N}$ heißt *platzkonstruierbar* wenn es terminierende DTM M gibt mit $\mathsf{space}_M(w) = s(|w|)$ für alle $w \in \Sigma^*$.

Betrachte Platzkonstruierbarkeit als "technische Annahme", die von allen natürlichen Funktionen erfüllt wird.

Z.B.:

- n^i ist platzkonstruierbar, für alle $i \geq 1$
- 2^n ist platzkonstruierbar

Savitch's Theorem

Theorem (Savitch)

Wenn s platzkonstruierbar ist, dann $\mathsf{NSpace}(s) \subseteq \mathsf{DSpace}(s^2)$.

Idee:

- Prädikat $\mathsf{Pfad}(C, C', i)$ ist wahr gdw. es Pfad in $G_{M,s(n)}$ gibt von C nach C' und mit Länge max. 2^i
- Wir interessieren uns also für $\mathsf{Pfad}(C, C', \log(|\mathsf{Conf}_{M,s(n)}|))$
- Jeder Pfad der Länge max. 2^i von C nach C' hat "Mittelpunkt" C_m: C erreicht C_m erreicht C', beides in max. 2^{i-1} Schritten
- Benutze "Teile & Herrsche":
  Um $\mathsf{Pfad}(C, C', i)$ zu entscheiden, betrachte alle möglichen Mittelpunkte C_M, entscheide rekursiv $\mathsf{Pfad}(C, C_M, i-1) \wedge \mathsf{Pfad}(C_M, C', i-1)$
- Rekursionstiefe $\log(|\mathsf{Conf}_{M,s(n)}|) \in \mathcal{O}(s(n))$, für jeden rekursiven Abstieg Speicherbedarf $\mathcal{O}(s(n))$

Konsequenzen von Savitch

Korollar

PSpace = NPSpace = co-NPSpace

(Anm.: det. Platzkomplexitätsklassen sind wie det. Zeitkomplexitätsklassen unter Komplement abgeschlossen)

Aus diesem Grund werden NPSpace und co-NPSpace nicht betrachtet.

Man kann Savitch auch auf größere Klassen anwenden:

$$\textsc{ExpSpace} := \bigcup_{i \geq 1} \mathrm{DSpace}(2^{n^i}) \qquad \textsc{NExpSpace} := \bigcup_{i \geq 1} \mathrm{NSpace}(2^{n^i})$$

Korollar

ExpSpace = NExpSpace = co-NExpSpace

Konsequenzen von Savitch

Leicht zu sehen:

Beobachtung

Für alle (monoton wachsenden) $f : \mathbb{N} \rightarrow \mathbb{N}$: NTime$(f) \subseteq$ NSpace(f)
Es folgt NP $\subseteq$ NPSpace.

Mit Savitch folgt P $\subseteq$ NP $\subseteq$ PSPACE $\subseteq$ EXPTIME

Andere Sicht auf Beweis:

Korollar

GAP $\in$ DSpace$(\log(n)^2)$

Nutzen von Savitch

Savitch ist auch nützlich, um PSpace-Algorithmen zu finden:

Wir können o.B.d.A Nicht-Determinismus benutzen!

Zum Beispiel Universalität von endlichen Automaten:
gegeben nicht-det. endlicher Automat $\mathcal{A}$, ist $\mathcal{A} = \Sigma^*$?

Lemma

Wenn $L(\mathcal{A}) \neq \Sigma^*$, dann gibt es $w \in \Sigma^* \setminus L(\mathcal{A})$ der Länge max. $2^{|Q|}$ mit Q Zustandsmenge von $\mathcal{A}$.

Theorem

Das Universalitätsproblem für endliche Automaten ist in PSpace.

Kapitel 4

PSpace-Härte und -Vollständigkeit

PSpace-Vollständigkeit

Theorem

$\text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME}$

Echtheit unbekannt!

Wie bei P vs NP: man behilft sich mit dem Begriff der Härte und Vollständigkeit

Härte gründet sich wieder auf Polynomialzeit-Reduktionen

Lemma

Wenn $L \in$ PSpace and $L' \leq_p L$, dann $L' \in$ PSpace

PSpace-Vollständigkeit

Definition PSpace-Härte, PSpace-Vollständigkeit

Problem L ist

- *PSpace-hart* wenn $L' \leq_p L$ für alle $L \in$ PSpace;
- *PSpace-vollständig* wenn L PSpace-hart und in PSpace.

Beobachtung

Wenn L PSpace-hart und

- $L \in$ P, dann P $=$ PSpace;
- $L \in$ NP, dann NP $=$ PSpace.

Nächstes Ziel: zeigen, dass QBF PSpace-Vollständig ist.

QBF

Wir verwenden QBFs, bei denen Quantoren nicht unbedingt Präfix sind

Definition generelle QBF

Sei AV unendlich abzählbare Menge von *Aussagenvariablen*.
Menge der *generellen QBFs* ist kleinste Menge so dass:

- 0,1 sind generelle QBFs
- jedes $v \in$ AV ist generelle QBF
- wenn φ, ψ generelle QBFs und $v \in$ AV, so sind auch $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\exists v.\varphi$ und $\forall v.\varphi$ generelle QBFs

Freie Variable ist Variable, die nicht durch Quantor gebunden ist.
Generelle QBF ohne freie Variable heißt *genereller QBF Satz*.

QBF

Definition generelle QBF Semantik

WZ π *erfüllt* generelle QBF

- 0 niemals und 1 immer;
- v wenn $\pi(v) = 1$;
- $\neg\varphi$, wenn π nicht φ erfüllt;
- $\varphi \wedge \psi$ wenn π sowohl φ als auch ψ erfüllt;
- $\varphi \vee \psi$ wenn π φ oder ψ erfüllt (oder beides);
- $\exists v.\varphi$ wenn $\pi[v/0]$ φ erfüllt oder $\pi[v/1]$ φ erfüllt;
- $\exists v.\varphi$ wenn $\pi[v/0]$ φ erfüllt und $\pi[v/1]$ φ erfüllt.

Genereller QBF Satz φ ist gültig, wenn er von jeder (äquivalent: einer) WZ erfüllt wird.

QBF

Zur Unterscheidung nennen wir die ursprünglichen definierten *Präfix-QBF*

Lemma

Jeder generelle QBF Satz φ kann in polynomieller Zeit in eine Präfix-QBF φ' umgewandelt werden, so dass φ gültig gdw. φ' gültig.

Theorem

QBF ist PSpace-hart, also PSpace-vollständig.

Strategie: Zeigen, dass $L \leq_p$ QBF für alle $L \in$ PSPACE.

QBF

Sei $L \in \textsc{PSpace}$ und M eine $p(n)$-platzbeschränkte DTM mit $L(M) = L$.

Ziel: gegeben w, finden von generellem QBF Satz φ_w so dass

1. φ_w gültig gdw. M akzeptiert w und
2. Länge von φ_w polynomiell beschränkt in der von w

Wir können nicht den Matrix-Ansatz von Cook verwenden, denn M hat u.U. exponentielle Laufzeit

Stattdessen: Reformulierung des Beweises von Savitch's Theorem "in der Sprache von QBF"

QBF

Zur Erinnerung: $\mathsf{Pfad}(C, C', i)$ ist wahr, wenn es Pfad mit Länge $\leq 2^i$ von C nach C' in $G_{M,p(n)}$ gibt.

Wir repräsentieren Konfiguration durch folgende Variablen:

- Z_q für jedes $q \in Q$ beschreibt Zustand;
- $B_{a,i}$ für jedes $a \in \Gamma$ und $i \leq p(n)$ beschreibt Symbol auf i-ter Bandzelle (Nummerierung beginnt mit 0);
- K_i für jedes $i \leq n^d$ beschreibt Kopfposition.

Diese Formel garantiert, dass die Variablen "legale" Konfiguration beschreiben:

$$\psi_{\mathsf{conf}}(\overline{C}) := \bigvee_{q \in Q} Z_q \wedge \bigwedge_{q,q' \in Q, q \neq q'} \neg(Z_q \wedge Z_{q'}) \wedge \bigvee_{i \leq p(n)} K_i \wedge \bigwedge_{i,i' \leq p(n), i \neq i'} \neg(K_i \wedge K_{i'}) \wedge$$

$$\bigwedge_{i \leq p(n)} \bigvee_{a \in \Gamma} B_{a,i} \wedge \bigwedge_{i \leq p(n), a,a' \in \Gamma, a \neq a'} \neg(B_{a,i} \wedge B_{a',i})$$

QBF

Sei $R(i) = i + 1$ und $L(i) = i - 1$ wenn $i > 1$ und $L(0) = 0$.

Folgende Formel sagt, dass $\overline{C}'$ sich in einem Schritt aus $\overline{C}$ ergibt (wobei die Variable in $\overline{C}$ und $\overline{C}'$ durch $\cdot'$ unterschieden werden)

$$\psi_{\text{next}}(\overline{C}, \overline{C}') := \psi_{\text{conf}}(\overline{C}) \wedge \psi_{\text{conf}}(\overline{C}') \wedge$$

$$\bigwedge_{i \leq p(n)} \Big(K_i \rightarrow \Big(\bigwedge_{j \leq p(n), j \neq i, a \in \Gamma} (B_{a,j} \leftrightarrow B'_{a,j}) \wedge \bigwedge_{\delta(q,a)=(q',a',M), M(i) \leq p(n)} (Z_q \wedge B_{a,i}) \rightarrow (Z'_q \wedge B'_{b,i} \wedge K'_{M(i)}) \Big) \Big)$$

Wir definieren nun $\text{Pfad}(C, C', i)$ für den Fall $i = 0$:

$$\psi^0_{\text{reach}}(\overline{C}, \overline{C}') := \psi_{\text{eq}}(\overline{C}, \overline{C}') \vee \psi_{\text{next}}(\overline{C}, \overline{C}')$$

wobei $\psi_{\text{eq}}(\overline{C}, \overline{C}')$ sagt, dass $\overline{C}$ und $\overline{C}'$ identisch

QBF

Für $i > 0$ ist es verlockend, zu schreiben

$$\psi^i_{\mathsf{reach}}(\overline{C},\overline{C}') := \exists \overline{C}''.\psi^{i-1}_{\mathsf{reach}}(\overline{C},\overline{C}'') \wedge \psi^{i-1}_{\mathsf{reach}}(\overline{C}'',\overline{C}')$$

aber das führt zu QBF der Größe min. 2^i (wiederholte Verdopplung!)

Universelle Quantoren helfen:

$$\begin{aligned}\psi^i_{\mathsf{reach}}(\overline{C},\overline{C}') \quad := \quad & \exists \overline{C}'' \forall \overline{K} \forall \overline{K}'.(\ (\psi_{\mathsf{eq}}(\overline{C},\overline{K}) \wedge \psi_{\mathsf{eq}}(\overline{C}'',\overline{K}')) \rightarrow \psi^{i-1}_{\mathsf{reach}}(\overline{K},\overline{K}') \wedge \\ & (\psi_{\mathsf{eq}}(\overline{C}'',\overline{K}) \wedge \psi_{\mathsf{eq}}(\overline{C}',\overline{K}')) \rightarrow \psi^{i-1}_{\mathsf{reach}}(\overline{K},\overline{K}')\ )\end{aligned}$$

Leicht zu finden:

- Formel $\psi^w_{\mathsf{input}}(\overline{C})$ die sagt, dass $\overline{C}$ initiale Konfiguration für Eingabe w ist
- Formel $\psi_{\mathsf{acc}}(\overline{C})$ die sagt, dass $\overline{C}$ akzeptierend ist.

QBF

M ist $p(n)$-platzbeschränkt

$$\Rightarrow \quad M \text{ ist } T(n) = 2^{q(n)}\text{-zeitbeschränkt für Polynom } q.$$

Die Reduktions-QBF ist nun:

$$\psi_w := \exists\overline{C}\exists\overline{C'}.(\ \psi^w_{\mathsf{input}}(\overline{C}) \wedge \psi_{\mathsf{acc}}(\overline{C}) \wedge \psi^{T(|w|)}_{\mathsf{reach}}(\overline{C}, \overline{C}')\ )$$

Lemma

M akzeptiert w gdw. φ_w gültig.

QBF ist ein bischen wir "SAT für PSpace"

Es gibt mittlerweile recht effiziente QBF Solver (aber nicht so robust wie SAT-Solver)

PSpace-Vollständigkeit

Natürlichere Probleme, die PSpace-Vollständig sind:

- Universalität von endlichen Automaten
- Schnittproblem für endliche Automaten:
  gegeben nicht-det. endliche Automaten $\mathcal{A}$, $\mathcal{A}'$, ist $\mathcal{A} \cap \mathcal{A}' = \emptyset$?
- SQL Anfrageproblem
  gegeben Instanz von relationaler Datenbank D, SQL Anfrage q,
  Tupel t: ist t in der Antwort von q auf D enthalten?
- Manche Spielprobleme, z.B.
  Brettspiele wie HEX, bei denen man einen gelegten Stein nicht wieder entfernen darf (generalisiert auf Felder beliebiger Größe)

Kapitel 4

Logarithmischer Platz

LogSpace

Die Definition und Analyse von PSpace hat Struktur im Raum der nicht effizient lösbaren Probleme offengelegt

Hier: weitere Analyse des Raumes der effizient lösbaren Probleme

Neue Komplexitätsklasse: $\textsc{LogSpace} := \text{DSpace}(\log(n))$

Es folgt aus $\text{DSpace}(s) \subseteq \text{DTime}(2^{\mathcal{O}(s)})$:

Theorem

$\textsc{LogSpace} \subseteq \text{P}$

Aus $\log(n^i) = i \cdot \log(n)$ folgt $\textsc{LogSpace} := \bigcup_{i \geq 0} \text{DSpace}(\log(n^i))$

LogSpace

Schon gesehen: $L = \{u\overleftarrow{u} \mid u \in \{a,b\}^*\} \in \textsc{LogSpace}$

Ein natürlicheres Problem in LogSpace:

UTREE ist die Menge der ungerichteten Graphen G, die Bäume sind (= azyklisch und zusammenhängend).

Theorem

UTREE $\in$ $\textsc{LogSpace}$

Intuitiv:
LogSpace beschreibt, was man mit einer fixen Zahl binärer Zähler erreichen kann.

LogSpace

Kompositionalität:

Komposition erfordert 1. TM mit Ausgabe und 2. Zwischenspeichern von Ergebnissen

Aber:

- LOGSPACE Algorithmus hat Ausgabe mit max. Größe n^k,
  für Ausgabe benötigter Platz darf also nicht mitgezählt werden
- Speicherung der Ausgabe als “Zwischenergebnis” bei Komposition in LogSpace also auch unmöglich!

Wir definieren uns erstmal ein entsprechendes Maschinenmodell

LogSpace Transduktor

Definition LogSpace Tranduktor

Ein *LogSpace Transduktor* ist DTM M mit

- einem Eingabeband, von dem nur gelesen wird;
- einer festen Zahl von logarithmisch in der Eingabelänge beschränkten Arbeitsbändern
- einem Ausgabeband, auf das nur gerschrieben wird und so dass in jedem Schritt:
  - entweder ein Symbol auf Ausgabeband geschrieben und Kopf einen Schritt nach rechts bewegt wird
  - oder nichts auf Ausgabeband geschrieben wird und der Kopf seine Position behält.

Abbildung $f : \Sigma^* \to \Gamma^*$ ist *LogSpace-berechenbar*, wenn es Logspace-Tranduktor gibt, der bei jeder Eingabe $w \in \Sigma^*$ anhält und $f(w)$ auf Ausgabeband schreibt.

Kompositionalität

LogSpace ist abgeschlossen unter:

- Ausführen zweier Algorithmen, Kombination der Ergebnisse
- Ausführen eines Algorithmus in jedem Schritt eines anderen
- Anwenden eines Algorithmus auf Ergebnis eines anderen

Der Beweis ist in allen Fällen ähnlich, wir zeigen nur letzteres

Theorem

Wenn $f : \Sigma^* \to \Gamma^*$ und $g : \Omega^* \to \Sigma^*$ LogSpace-berechenbar, so auch $f(g) : \Omega^* \to \Gamma^*$.