# *Komplexitätstheorie*

Montag und Donnerstag
14:15 – 15:45 Uhr
in C-221



## Veranstaltungsinhalt (vergl. KVV)

Welcher Aufwand an Rechenzeit und Speicherplatz ist erforderlich, um ein gegebenes Problem zu lösen?

- Mindestens benötigter Rechenaufwand sind untere Schranken für ein Problem!

  Beweise oft sehr schwer! Daher ging man in der Komplexitätstheorie dazu über, verschiedene Probleme miteinander zu vergleichen, um so wenigstens eine Aussage über deren relative Komplexitäten zu bekommen. Komplexitätsklassen statt Chomsky-Hierarchie!

- Obere Schranke ist jeder bekannte Algorithmus.

☑ Strukturelle Komplexitätstheorie

☐ konkrete Komplexitätstheorie (Konstruktion konkreter Algorithmen)

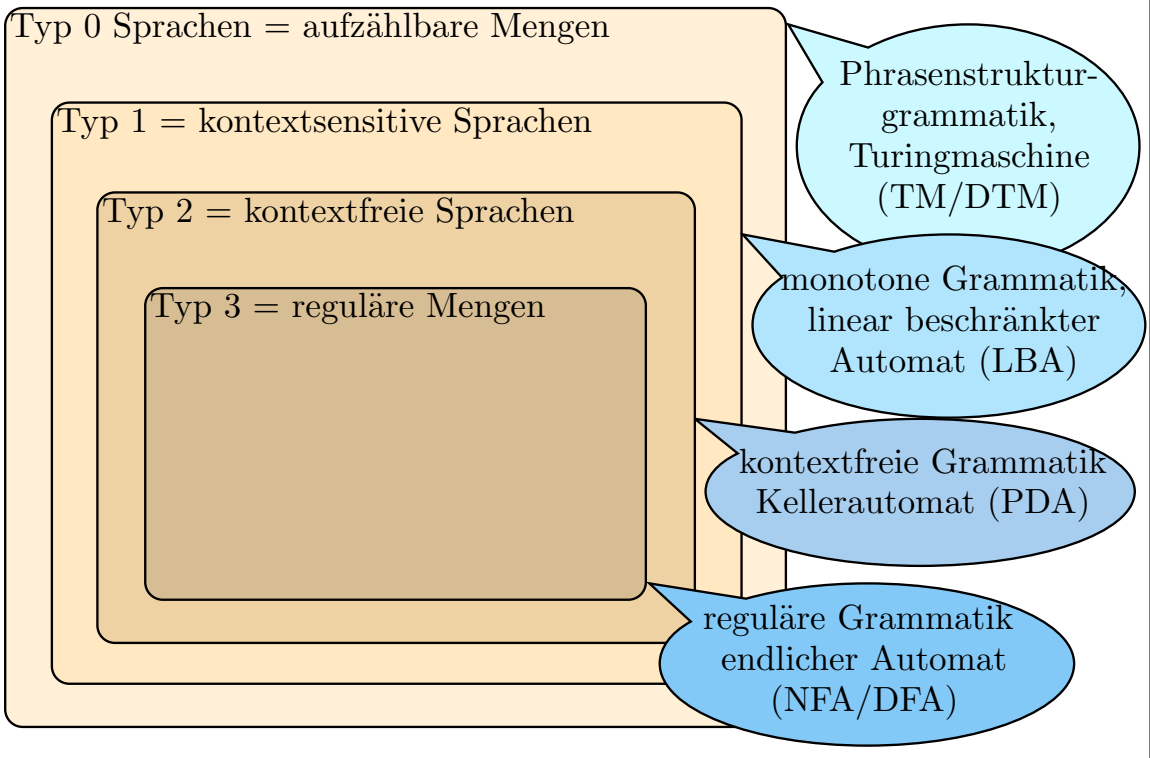
☐ Algorithmik (Analyse und Konstruktion guter Algorithmen)

# Literaturhinweise


- John E. Hopcroft & Jeffrey D. Ullman:
  Einführung in die Automatentheorie, Formale Sprachen u. Komplexitätstheorie, Addison Wesley (1988) [ein Standardwerk fürs Bachelor Studium]
- K. Rüdiger Reischuk:
  Komlexitätstheorie, Band 1 Grundlagen, B. G. Teubner (1999) [recht ausführliche Beweise]
- Christos H. Papadimitriou:
  Computational Complexity, Addison Wesley Longman (1995) [ein Standardwerk]
- Michael Sipser:
  Introduction to theTheory of Computing, 2. Aufl., PWS Publ. Comp., Boston (2006) [das Nötige]
- Juraj Hromkovič:
  Algorithmics for Hard Problems, Springer-Verlag (2003)
- Jörg Rothe:
  Complexity Theory and Cryptology, Springer-Verlag (2005) [sehr gut, aber nicht zu leichte Kost]
- Heribert Vollmer:
  Introduction to Circuit Complexity, Springer-Verlag (1999) [spezielles Themengebiet]
- Peter Bürgisser, Michael Clausen, M.Amin Shokrollahi:
  Algebraic Complexity Theory, Springer-Verlag (1997) [mathematische Orientierung]
- Dieter Jungnickel:
  Graphs, Networks and Algorithms, Springer-Verlag (2002) [prima]




# Chomsky Hierarchie



## Problemklassen

- **Optimierungsprobleme**
  *z.B.:* Traveling Salesperson:

  *Gegeben:* Landkarte mit zu besuchenden Städten
  *Gesucht:* Kürzeste Tour durch alle Städte

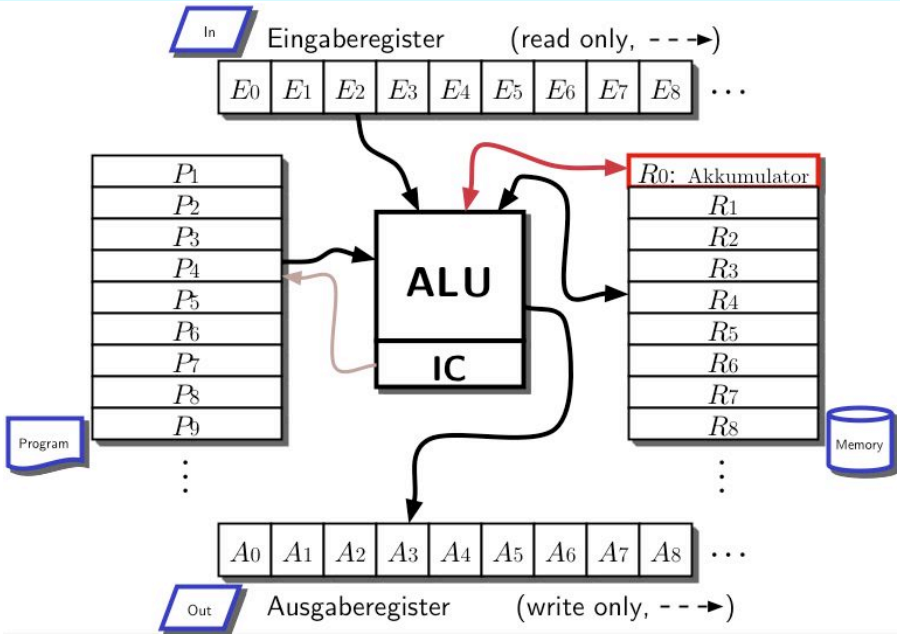
- **Entscheidungsprobleme** (Ja/Nein-Probleme)
  *z.B.:* SAT = Erfüllbarkeit Aussagenlogischer Formeln

- **Erweiterte Entscheidungsprobleme**
  *z.B.:* konstruktive Erfüllbarkeit
  *z.B.:* konstruktive Traveling Salesperson:

  *Gegeben:* Landkarte mit zu besuchenden Städten, gewisse Menge Benzin
  *Frage:* gibt es Tour durch alle Städte, die mit dem Benzin auskommt? Wenn ja, wie sieht diese aus?



## Registermaschine

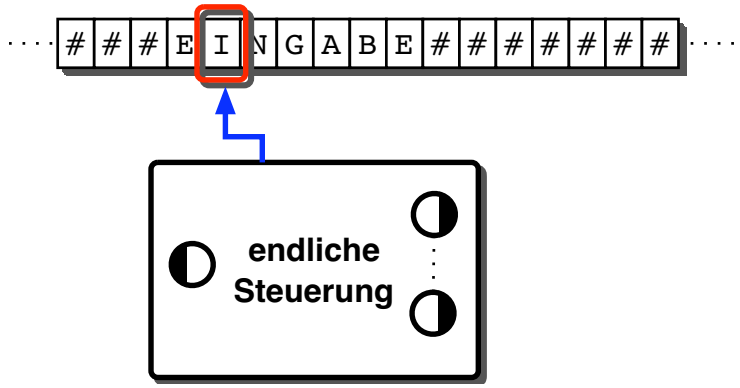



Die Registermaschine bedarf besonderer Komplexitätsmaße und wird hauptsächlich als Baustein der parallelen Algorithmen (PRAM) genutzt.

## Kriterien für Rechenaufwand

Zeitbedarf und Platzbedarf auf abstrakten Modellen, wie

Turingmaschinen und/oder Registermaschinen (RAM, PRAM)




Die 1-Band TM ist praktisch beim Akzeptieren von formalen Sprachen, weniger bei dem, was wir noch vorhaben...



## Akzeptieren bei 1-Band (D/N)TM's

***Definition:***

Eine endliche Rechnung $k_1 \vdash k_2 \vdash \ldots \vdash k_t$ heißt Erfolgsrechnung, wenn $k_1 \in \Gamma^* \cdot \{q_0\} \cdot \Gamma^*$ und $k_t \in \Gamma^* \cdot Z_{\text{end}} \cdot \Gamma^*$ gilt.

> Es ist also erlaubt, dass auf $k_t$ eine weitere Konfiguration folgt, die Rechnung also noch weitergehen könnte!

Für die DTM $A = (Z, \Sigma, \Gamma, \delta, q_0, \#, Z_{\text{end}})$ bezeichnet $L(A)$ die von $A$ akzeptierte Sprache:

$$L(A) := \{w \in \Sigma^* \mid \exists u, v \in \Gamma^* \; \exists q \in Z_{\text{end}} : \; q_0 w \vdash^* uqv\}$$

> Auch hier könnte die Rechnung also noch weitergehen!

Eine (D)TM hält, wenn es bei der vorliegenden Bandinschrift keine Folgekonfiguration gibt!

## Frage:

Würde sich die akzeptierte Sprache einer DTM ändern, wenn man forderte, dass in keiner DTM aus einem Endzustand eine weitere Konfiguration erreichbar wäre, man also in einer gegebenen DTM die Überführungsfunktion so modifizierte, dass $\delta(p, y)$ für jeden Endzustand $p$ und jedes Symbol $y$ undefiniert ist?

Antwort: Nein! (?) [so in Asteroth/Baier!] Beweis?

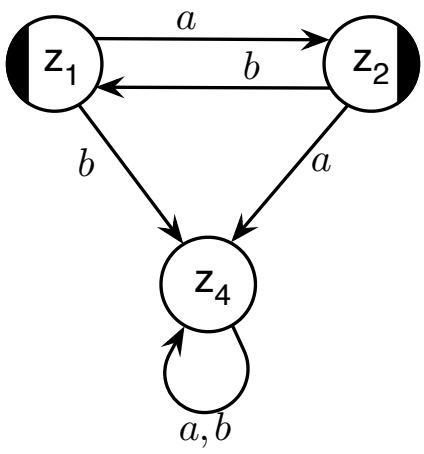
Alle (endlichen oder nie endenden) Rechnungen bei nicht akzeptierten Eingaben, also solchen Rechnungen, die nie einen Endzustand erreichen, sind in der modifizierten TM immer noch genauso möglich!

Wird ein Wort jedoch akzeptiert, dann gibt es eine Erfolgsrechnung, die einen oder mehrere Endzustände erreicht bzw. auch durchläuft. Wenn solche Rechnungen beim ersten Erreichen eines Endzustandes anhalten, weil keine Folgekonfiguration existiert, werden diese immer noch akzeptiert! An diesen Rechnungen wird vor Erreichen des Endzustandes ja nichts geändert!
Aber nun stehen nicht mehr alle Rechnungen zur Verfügung! (Beispiel?)
Also Antwort:

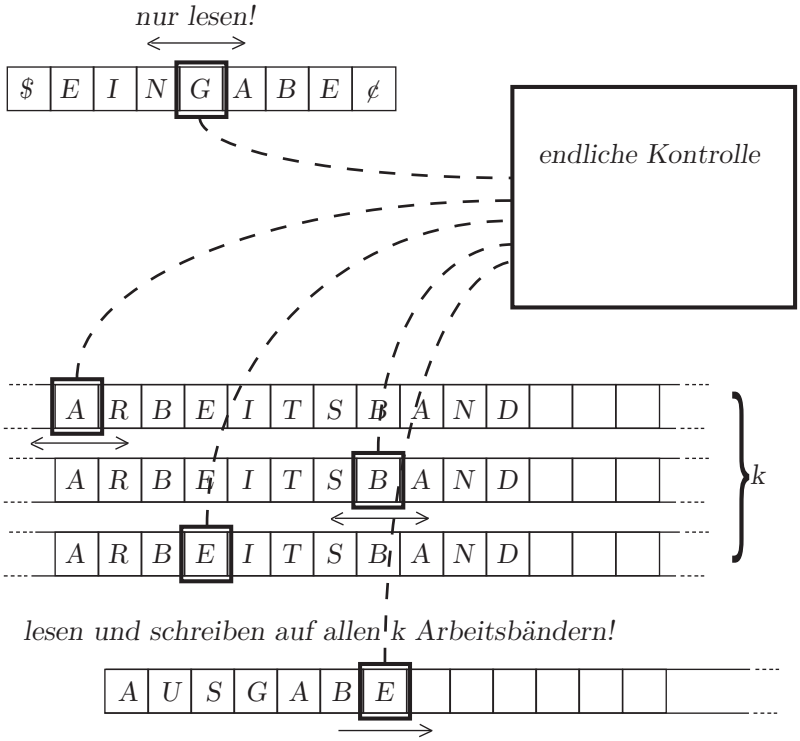
**Das ist bei DTM ein Unterschied, aber nicht bei NTM!**



## Das Beispiel

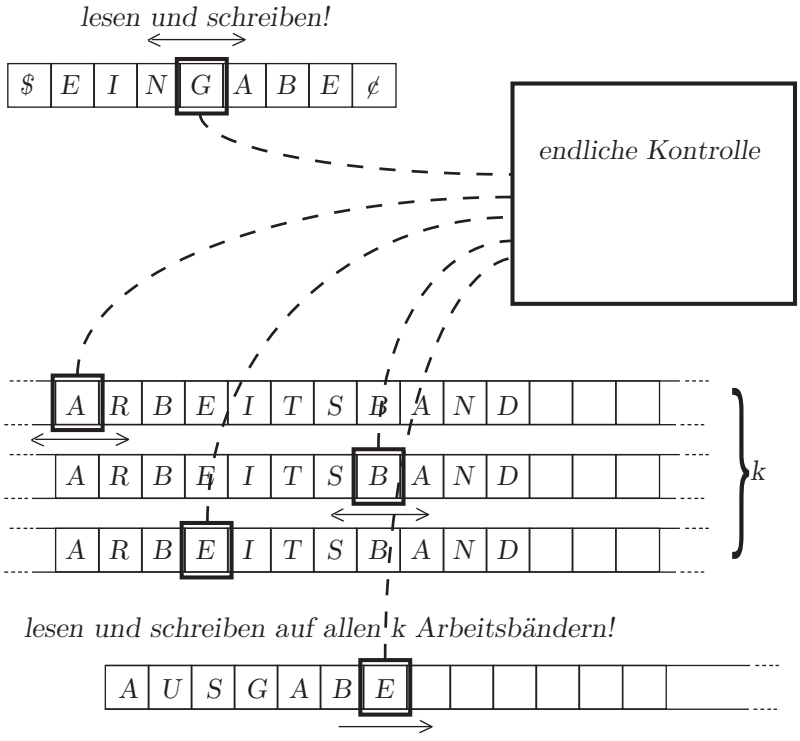


# k-Band off-line Turingmaschine






# k-Band on-line Turingmaschine



## Komplexitätsschranken bei (off-line) k-Band (D/N)TM's

Eine NTM $A$ verarbeitet ein Wort $w$ mit der Zeitbeschränkung $t$ genau dann, wenn die kürzeste Rechnung für $w$ nur $\lceil t \rceil$ Schritte hat.

Dies bezeichnet man als schwache Zeitschranke!

Genügt eine (N/D)TM einer starken Zeitschranke $f : I\!N \to I\!N$, so bedeutet dies, dass keine Rechnung existiert, die auf einer Eingabe der Länge $n, n \geq 0$ mehr als $\lfloor f(n) \rfloor + 1$ Rechenschritte macht.

Solche Funktionen nennt man *f(n)-zeitkonstruierbar* !

Bei linearer Zeit $f(n) = n$ werden also stets $n+1$ Schritte gemacht. Sublineare Zeitklassen betrachten wir erst viel später!

Ist der Unterschied zwischen starker und schwacher Zeitbeschränkung erheblich?



## Komplexitätsschranken bei (off-line) k-Band (D/N)TM's

Eine NTM $A$ verarbeitet $w$ mit der Platzbeschränkung $s \in I\!R$, wenn die Rechnung mit dem geringsten Platzverbrauch für $w$ nur $\lceil s \rceil$ Platz benötigt.

Dies bezeichnet man als schwache Platzschranke!

Genügt eine (N/D)TM einer starken Platzschranke $f : I\!N \to I\!N$, so bedeutet dies, dass keine Rechnung existiert, die auf einer Eingabe der Länge $n, n \geq 0$ mehr als $\lfloor f(n) \rfloor + 1$ Felder des Bandes (bzw. der Arbeitsbänder bei Mehrband TM) benutzt.

Solche Funktionen nennt man *f(n)-platzkonstruierbar* !

Bei sublinearem Platz wird die Länge der Eingabe logarithmisch berücksichtigt. Sublogarithmische Platzklassen betrachten wir eher nicht!

Ist der Unterschied zwischen starker und schwacher Platzbeschränkung erheblich?

Wir verwenden im Folgenden nur *starke Platz- und Zeitschranken*!

## Landau Symbolik – Wachstumsvergleich von Funktionen

praktisch ist: $\log(n) := \begin{cases} 1 & \text{, falls } n \leq 1 \\ \lfloor \log_2(n) \rfloor + 1 & \text{, sonst.} \end{cases}$

$$O(g(n)) = \{f : \mathbb{N} \to \mathbb{R} \mid (\exists c \in \mathbb{R}^+)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)[\,|f(n)| \leq c\,|g(n)|\,]\}.$$

$$o(g(n)) = \{f : \mathbb{N} \to \mathbb{R} \mid (\forall c \in \mathbb{R}^+)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)[\,|f(n)| \leq c\,|g(n)|\,]\}.$$

$f(n) \in o(g(n))$ ist äquivalent zu $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$

$$f(n) \in \Omega(g(n)) \iff g(n) \in O(f(n))$$

$$f(n) \in \omega(g(n)) \iff g(n) \in o(f(n))$$

$$f(n) \in \Theta(g(n)) \iff f(n) \in O(g(n)) \text{ und } f(n) \in \Omega(g(n)).$$



## grundlegende Komplexitätsklassen I

Für $t, s : \mathbb{N} \to \mathbb{N}$ mit $\forall n \in \mathbb{N} : t(n) \geq n$ und $s \in \Omega(\log)$, sowie $k \geq 1$ sei:

$DTIME_k(t) := \{L \subseteq \Sigma^* \mid$ es existiert eine k-Band DTM $M$ mit $L = L(M)$, die auf allen Eingaben der Länge $n$ nach höchstens $\max(t(n), n+1)$ Schritten hält.$\}$

$NTIME_k(t) := \{L \subseteq \Sigma^* \mid$ es existiert eine k-Band NTM $M$ mit $L = L(M)$, die auf allen Eingaben der Länge $n$ und bei allen Rechnungen nach höchstens $\max(t(n), n+1)$ Schritten hält.$\}$

$DSPACE_k(s) := \{L \subseteq \Sigma^* \mid$ es existiert eine k-Band DTM $M$ mit $L = L(M)$, die auf allen Eingaben der Länge $n$ höchstens $\lfloor s(n) \rfloor + 1$ Arbeitsbandfelder benötigt.$\}$

$NSPACE_k(s) := \{L \subseteq \Sigma^* \mid$ es existiert eine k-Band NTM $M$ mit $L = L(M)$, die auf allen Eingaben der Länge $n$ und bei allen Rechnungen höchstens $\lfloor s(n) \rfloor + 1$ Arbeitsbandfelder benötigt.$\}$

## grundlegende Komplexitätsklassen II

$$DTIME(t) := \bigcup_{k \geq 1} DTIME_k(t)$$

$$NTIME(t) := \bigcup_{k \geq 1} NTIME_k(t)$$

$$DSPACE(s) := \bigcup_{k \geq 1} DSPACE_k(s)$$

$$NSPACE(s) := \bigcup_{k \geq 1} NTIME_k(s)$$

In der Literatur wird bisweilen

*TIME(t)* anstelle von *DTIME(t)*

und

*SPACE(s)* anstelle von *DSPACE(s)*

notiert!



## Beispiele

$$\{a^n b^n c^n \mid n \in I\!N\} \in DSPACE(\log n)$$

$$\{w\$w \mid w \in \Sigma^*\} \in DSPACE(\log n)$$

$$\{p \in \{1\}\{0,1\}^* \mid [p]_2 \text{ ist Primzahl}\} \in DSPACE(n)$$

$$\{p \in \{1\}\{0,1\}^* \mid [p]_2 \text{ ist Primzahl}\} \in DTIME(n^{12})$$

Letzteres Ergebnis vom 10. Juli 2002 stammt von Manindra Agrawal und den Bachelor-Studenten Neeraj Kayal und Nitin Saxena!

Wie zeigen ***Sie*** die ersten drei Aussagen?

## Einige wichtige Ergebnisse aus der F3 Vorlesung:

**Speed up für 1-Band TM:**

Zu jeder $t(n)$-zeitbeschränkten TM mit $\inf_{n\to\infty}(\frac{t(n)}{n}) = \infty$ und jedem $c \in \mathbb{R}$ mit $c > 0$ gibt es eine äquivalente $c \cdot t(n)$-zeitbeschränkte TM.

**Bandkompression für 1-Band TM:**

Zu jeder $s(n)$-platzbeschränkten TM und jeder reellen Zahl $c \in \mathbb{R}$ mit $c > 0$ gibt es eine äquivalente TM die $c \cdot s(n)$-platzbeschränkt ist.

Gilt das auch bei Mehrband Turingmaschinen?



## Bandreduktion, speed up und Bandkompression bei $k$-DTM

Durch Spurenbildung wird aus $k$ Bändern *1* Band und das Ergebnis der Bandkompression für *1*-DTM wird übertragbar:

$$\begin{aligned} DSPACE_1(s) &= DSPACE(s) \\ NSPACE_1(s) &= NSPACE(s) \\ \forall c > 0 : DSPACE(\lceil c \cdot s(n) \rceil) &= DSPACE(s(n)) \end{aligned}$$

Für (deterministische!) Zeitklassen gibt es diese **Bandreduktion nicht**! Dort gilt [Beweis wie für *1*-DTM] zunächst nur das speed up:

Für $t : \mathbb{N} \to \mathbb{N}$ mit $\frac{1}{t(n)} \in o(\frac{1}{n})$ ($t(n)$ wächst schneller as $n$.) gilt:

$$\begin{aligned} \forall k \geq 1 \forall c > 0 : DTIME_k(\lceil c \cdot t(n) \rceil) &= DTIME_k(t(n)) \\ \text{und also auch} & \\ DTIME(\lceil c \cdot t(n) \rceil) &= DTIME(t(n)) \end{aligned}$$

## Für lineare Zeit bei $k$-DTM, $k \geq 2$!

$$\forall k \geq 2 \forall c_1, c_2 > 1 : DTIME_k(\lceil c_1 \cdot n \rceil) = DTIME_k(\lceil c_2 \cdot n \rceil)$$

Sowohl $k > 1$ als auch $c_1 > 1$ und $c_2 > 1$ sind hier wichtig!

$$PAL = \{w \in \{a,b\}^* \mid w = w^{rev}\} \notin DTIME_1(n)$$

dies beweist man mit Hilfe von Kreuzungsfolgen und direkt [HU, Üb. 12.2(a)] oder mit Kolmogorov Komplexität! [Papadimitriou (1994) S. 52]

aber

$$PAL = \{w \in \{a,b\}^* \mid w = w^{rev}\} \in DTIME_2(n)$$



## wichtige Komplexitätsklassen

$P$

die Klasse der Sprachen (algorithmischen Probleme), die man in Polynomialzeit erkennen (lösen) kann und

$PSPACE$

die Klasse der Sprachen (algorithmischen Probleme), die man mit polynomiell viel Platz erkennen (lösen) kann.

Das ist ziemlich umgangssprachlich, und nicht präzise:

Ist $PSPACE$ nur über deterministische Turingmaschinen erklärt?

## wichtige Komplexitätsklassen

Wegen der Kompressions- und speed up- Sätze folgt:

$$P = \bigcup_{i \geq 1} DTIME(n^i) = \bigcup_{p \in POLY} DTIME(p(n))$$

$$NP = \bigcup_{i \geq 1} NTIME(n^i) = \bigcup_{p \in POLY} NTIME(p(n))$$

$$PSPACE = \bigcup_{i \geq 1} DSPACE(n^i) = \bigcup_{p \in POLY} DSPACE(p(n))$$

$$NPSPACE = \bigcup_{i \geq 1} NSPACE(n^i) = \bigcup_{p \in POLY} NSPACE(p(n))$$

Hierbei bezeichnet $POLY$ die Klasse aller Polynome aus $\mathbb{R}_{[x]}$!



## weitere Komplexitätsklassen

$$L := DSPACE(\log n)$$

$$NL := NSPACE(\log n)$$

$$DLBA := DSPACE(n)$$

$$\mathcal{C}s = NLBA := NSPACE(n)$$

$$EXPSPACE := \bigcup_{p \in POLY} DSPACE(2^{p(n)})$$

$$EXPTIME := \bigcup_{p \in POLY} DTIME(2^{p(n)})$$

## Determinismus versus Nichtdeterminismus

$(TM =)\ NTM \quad \approx \quad DTM$

$(LBA =)\ NLBA \quad ? \quad DLBA$

$(PDA =)\ NPDA \quad \subsetneq \quad DPDA$

$(FA =)\ NFA \quad = \quad DFA$

Für $f(n){:=}n$ ist dies das $NLBA$ ? $DLBA$ Problem und ungelöst! Es kann z.B. $NSPACE(c^n) \subsetneq NSPACE((c+\epsilon)^n)$ gezeigt werden! Und sonst?

Für die Klasse der Polynome ist dies das $P$ ? $NP$ Problem und ungelöst! Es kann hier nur $NTIME(f) \subseteq \bigcup_{c>1} DTIME(c^f)$ gezeigt werden! Und sonst? ...

*Was sagen sie zu:*

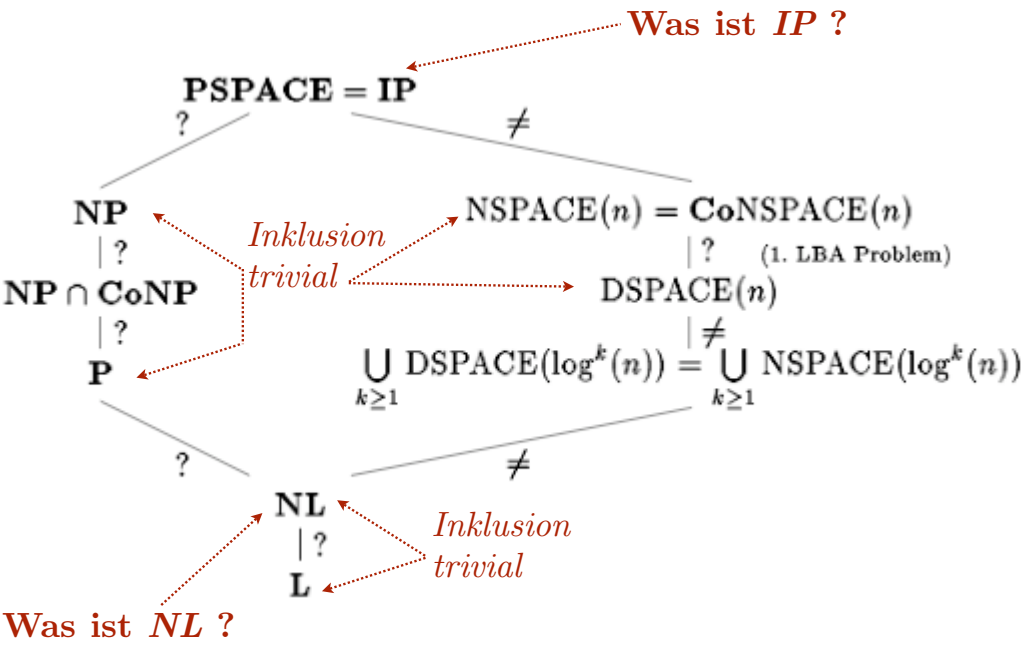
$DSPACE(f) = NSPACE(f)$?

*oder*

$DTIME(f) = NTIME(f)$ ?



## Hierarchie von Komplexitätsklassen




Solche Diagramme wollen wir lernen zu verstehen und vielleicht auch zu beweisen!

## Bandreduktion bei *DTIME* auf 1 Band nicht mehr linear!

$$(\mathbf{a}) \quad DTIME(t) \subseteq DTIME_1(t^2)$$

$$(\mathbf{b}) \quad DTIME(t) \subseteq DTIME_2(t \cdot \log(t))$$

Beweis (**a**):
alle k Bänder hintereinander Schreiben, Markierungen für alle Bandenden und die Köpfe. In *t(n)* Schritten werden niemals mehr als *t(n)* Felder beschrieben! Für einen Schritt der *k-DTM* wird gesamtes Band v.l.n.r. durchlaufen und Symbole unter den Köpfen gemerkt. Neue zu Schreibendes wird beim zweiten Lauf geändert. Soll ein neues Feld auf einem der $k$ Bänder am Rand ergänzt werden, wird neues Symbol geschrieben und Platz beim zweiten Lauf über alles erzeugt (verschieben des Restes).
Maximal wird die Bandinschrift *k(t(n)+c)* lang! ***Ein*** simulierter Schritt kostet schlimmstenfalls *4· (k(t(n)+c))* Schritte, mithin insgesamt $O(t^2(n))$.
Für *NTIME* folgt ebenso:

$$(\mathbf{a}') \quad NTIME(t) \subseteq NTIME_1(t^2)$$



## $DTIME(t) \subseteq DTIME_2(t \cdot \ell og(t))$

Beweis (**b**) sehr aufwändig:

- Bildung von Doppelspuren
- Einteilung in wachsende Blöcke
- geschicktes Verschieben der Blöcke
- sorgfältige Abschätzung

Vergleichen Sie dazu Satz 12.6. in
Hopcroft/Ullman: Einführung in die Automatentheorie,
Formale Sprachen und Komplexitätstheorie,
Addison Wesley (1988), deutsche Ausgabe.
(in Hopcroft/Motwani/Ullman finden Sie das nicht mehr!)

## Zeit versus Platz I

Offensichtlich gilt ja

$$DTIME(f(n)) \subseteq DSPACE(f(n)),$$

weil keine TM mehr Felder benutzen kann, als ihr Schritte zur Verfügung stehen!

Es kann jedoch mehr gezeigt werden:

$$DTIME\left(f(n)\right) \subseteq DSPACE\left(\frac{f(n)}{\log(f(n)}\right).$$

J.E. Hopcroft / W.J. Paul / L.G. Valiant:
On time vs. space and related problems, Proc. 16th Annual IEEE Symp. on Foundations of Computer Science, (1975) pp. 57–64.

**Grundidee:** (rekursives) Wiederholen von Rechnungen (möglich wegen Determinismus). Platzbeschränkte (N)TM braucht nie den gesamten Bandinhalt der zeitbeschränkten DTM zu kennen. Jeweils aktueller Bereich kann durch Rechnungswiederholung innerhalb der Zeitschranke stets neu konstruiert werden. (Darstellung ist hier zu kompliziert...) darzustellen



## Zeit versus Platz II

$$NTIME\left(f(n)\right) \subseteq DSPACE\left(f(n)\right)$$

**Beweisidee:**
Sei $M_1$ eine $k$-$NTM$ die $L$ in $f(n)$ Zeit akzeptiert. $DTM$ $M_2$ erzeugt eine der möglichen Konfigurationsfolgen der $NTM$ $M_1$ als $f(n)$ lange Folge von natürlichen Zahlen aus $\{0,1,...d\}$ ($d$ ist Maximalzahl von Verzweigungen in einem Zustand). Das Überprüfen einer erzeugten Rechnung auf Akzeptanz benötigt nur $f(n)$ Zeit und Platz. Es müssen exponentiell viele Rechnungen überprüft werden, was durch erneute Verwendung des benutzten Platzes nicht mehr Platz braucht. Da $f$ eine platzkonstruierbare Funktion ist, kann die jede Zahlenfolge in wiedergenutztem $O(f(n))$-Platz notiert werden!

## Platz versus Zeit I

$$NSPACE\,(f(n)) \subseteq DTIME\left(k^{\log(n)+f(n)}\right)$$

**Beweis mit der Erreichbarkeitsmethode:**

Sei $M_1$ eine $k$-$NTM$ mit Ein- und Ausgabeband die $L$ in $f(n)$ Platz akzeptiert. Konfigurationen von $M_1$ sind $2k-2$ Tupel der Form:

$$(q, i, u_2, v_2, u_3, v_3, \ldots, u_{k-1}, v_{k-1})$$

Das $k$-te Ausgabeband interessiert nicht, $i$ gibt die Position des Lesekopfes an, und $u_j v_j$ ist Inschrift von Band $j$ mit $\mid u_j v_j \mid \leq f(n)$ und $LSK$ auf erstem Symbol von $v_j$.

Wieviele Konfigurationen sind für $M_1$ möglich?



## Platz versus Zeit II

Wieviele Konfigurationen sind für $M_1$ möglich? $(q, i, u_2, v_2, u_3, v_3, \ldots, u_{k-1}, v_{k-1})$

$|Z|$ viele Zustände und $n+1$ verschiedene Werte für $i$, auf allen Bändern zusammen maximal $|\Gamma|^{(2k-2)f(n)}$ viele verschiedene Inschriften.

Insgesamt kann $M_1$ also maximal

$$(n+1)\cdot {c_1}^{f(n)} = 2^{\log(n+1)} {c_1}^{f(n)} = c_1^{\log_{c_1}(2)\log(n+1)+f(n)} = c_1^{c_2\log(n+1)+f(n)}$$

viele Konfigurationen beim Akzeptieren durchlaufen.

Der Konfigurationsgraph von $M_1$ enthält alle Konfigurationen und für jeden Schritt von $M_1$ eine gerichtete Kante zwischen diesen.

Es muss nun nur geprüft werden, ob in dem Konfigurationsgraph ein Pfad von einer Start- zu einer Endkonfiguration existiert, was in Zeit $O$(Knotenzahl+Kantenzahl) möglich ist. Wegen Kantenzahl $\leq$ Knotenzahl$^2$ Also in $c_3((c_1{}^{c_2\log(n+1)+f(n)})^2$. Mit $c := c_3 {c_1}^2$ folgt also Erreichbarkeit entscheidbar in $DTIME\left(c^{\log(n+1)+f(n)}\right) = DTIME\left(k^{\log(n)+f(n)}\right)$.

## Platz versus Zeit III

Ein Algorithmus für Knotenerreichbarkeit im Graphen arbeitet in Tiefen- oder Breitensuche über die zuerst konstruierte Adjazenzmatrix oder durch bedarfsweises Erzeugen der jeweiligen Folgekonfiguration. Beides kann in der selben Zeitschranke der zu konstruierenden *DTM* $M_2$ geschehen!

Als Ergebnis der letzten drei Resultate erhält man:

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE$$

Wodurch der linke Pfad im der Graphik von Folie 26 bewiesen wurde!



## Padding für deterministische Komplexitätsklassen

Das *Padding* ist eine Technik zur Rückführung ähnlicher, verschieden "hoher" Fragestellungen und wird daher auch Translationstechnik bezeichnet.

**Definition:**
Für $L \subseteq \Sigma^*, a \notin \Sigma$ und $f : \mathbb{N} \to \mathbb{N}$ mit $f(n) \geq n$ sei:

$$L(f,a) := \{wa^m \mid w \in L, m = f(|w|) - |w|\}.$$

**Satz:**
Sei $f$ zeit- (bzw. platz-)konstruierbar

a) Für $t : \mathbb{N} \to \mathbb{N}$ mit $t(n) \geq n$ gilt:
$L \in DTIME\,(t(f(n))) \iff L(f,a) \in DTIME\,(t(n))$.

b) Für $s : \mathbb{N} \to \mathbb{N}$ mit $s(n) \geq \log(n)$ gilt:
$L \in DSPACE\,(s(f(n)))) \iff L(f,a) \in DSPACE\,(s(n))$.

## Beweisskizze für Padding

$\Leftarrow$

Zu gegebenem $w$ konstruiere $wa^m$ mit $m = f(|w|) - |w|$ in $O(f(|w|))$ Schritten [bzw. $O(f(|w|))$ Platz]. Erkennung, ob $w \in L$ ist, kostet $t(|wa^m|) = t(f(|w|))$ Schritte [bzw. $s(f(|w|))$] Platz.

Insgesamt also ein Zeit- [bzw. Platz-]aufwand von $O((t \circ f)(n))$ [*bzw.* $O((s \circ f)(n))$].

$\Rightarrow$

Zu gegebenem $wa^m$, prüfe zunächst, ob $m = f(|w|) - |w|$ in $O(f(|w|)) = O(|wa^m|)$ Schritten.

Danach teste $w \in L$ in $t(f(|w|)) = t(|wa^m|)$ Schritten.

Insgesamt also in $DTIME(t(n))$.

Für Platz gilt analoges Argument.



## Anwendung der Padding- oder Translations-Technik

**Korollar:**

a) Wenn $P = PSPACE$, dann $EXPTIME = EXPSPACE$.

b) Wenn $P = L$, dann $EXPTIME = PSPACE$.

**Beweis:**
a) Sei $L \in EXPSPACE$, d.h. $\exists p \in POLY$ mit $L \in DSPACE\left(2^{p(n)}\right)$. Setze $f(n) := 2^{p(n)}$, dann folgt aus dem Paddingsatz:

$$L(f, a) \in DSPACE\,(n) = DLBA.$$

$P = PSPACE$ impliziert $L(f, a) \in P$ und es gibt Polynom $q \in POLY$, so dass $L(f, a) \in DTIME\,(q(n))$. Mit Paddingsatz folgt $L \in DTIME\,(q(f(n))) \in EXPTIME$, da

$$\left(2^{p(n)}\right)^r = 2^{r \cdot (p(n))} = 2^{p'(n)}.$$

Für b) wird analog argumentiert!

## Padding für nichtdeterministische Komplexitätsklassen

Das *Padding* Ergebnis gilt auch im nichtdeterministischen Fall, wie man analog dazu nachweist!

**Satz:**
Sei $f$ zeit- (bzw. platz-)konstruierbar

a) Für $t : \mathbb{N} \to \mathbb{N}$ mit $t(n) \geq n$ gilt:

$$L \in NTIME\,(t(f(n))) \iff L(f,a) \in NTIME\,(t(n)).$$

b) Für $s : \mathbb{N} \to \mathbb{N}$ mit $s(n) \geq \log(n)$ gilt:

$$L \in NSPACE\,(s(f(n))) \iff L(f,a) \in NSPACE\,(s(n)).$$

Für $L \subseteq \Sigma^*, a \notin \Sigma$ und $f : \mathbb{N} \to \mathbb{N}$ mit $f(n) \geq n$ war ja:
$L(f,a) := \{wa^m \mid w \in L, m = f(|w|) - |w|\}$.



## Der Satz von Savitch I

Für jede platzkonstruierbare Funktion $s$ mit $s(n) \geq \log(n)$ gilt:

$$NSPACE\,(s(n)) \subseteq DSPACE\left(s^2(n)\right)$$

**Beweis:**
Der Beweis gelingt, indem wir zuerst zeigen, dass

$$NL = NSPACE\,(\log(n)) \subseteq DSPACE(\log^2(n))$$

gilt und dann Padding benutzen.

Sei $m$ eine $\log(n)$-platzbeschränkte $NTM$, dann ist die Zahl der möglichen Konfigurationen bei Eingaben der Länge $n$ beschränkt durch ein Polynom $p(n)$ der Größenordnung

$$n \cdot \log(n) \cdot |Z| \cdot |\Gamma|^{\log(n)} \in O\left(n^2 \log(n)\right) \subseteq O\left(n^3\right).$$

Jede Konfiguration kann wegen der Platzbeschränkung von $M$ in $c_1 \cdot \log(n)$ Platz gespeichert werden.

## Der Satz von Savitch II

Rekursiv wird nun für $i > 1$

$$PFAD(x, y, i) \iff \exists z : PFAD(x, z, \lceil \frac{i}{2} \rceil) \wedge PFAD(z, y, \lfloor \frac{i}{2} \rfloor)$$

für alle Konfigurationen $z$ geprüft, wobei das Prädikat $PFAD(x, y, i)$ erfüllt ist, wenn in der $NTM$ $M$ von Konfiguration $x$ zur Konfiguration $y$ ein Pfad mit Länge $i$ existiert. (Die Fälle $i = 0$ mit $x = y$ und $i = 1$ sind in einem Schritt zu testen.)

Da eine schleifenfreie Rechnung in $M$ maximal die Länge $p(n)$ benötigt, kann auch stets $i \leq p(n)$ sichergestellt werden.

Im Einzelnen könnte man den Algorithmus wie folgt formulieren:



## Das Programm nach Savitch

```
begin
  unveränderlich: NTM M und Polynom p
  Eingabe: w mit n := |w|

  function PFAD(x, y : KONF; i : integer) : boolean
  begin
  zwi: boolean; z: KONF
    if i ≤ 1
      then PFAD := (x = y ∨ x ⊢_M y)
      else
          begin
            zwi := false;
            for all z, |z| ≤ log(n) do
              if PFAD(x, z, ⌈i/2⌉) ∧ PFAD(z, y, ⌊i/2⌋)
              then zwi := true;
            od;
          PFAD := zwi;
          end;
  end (* function *)

  x_0 := Anfangskonfiguration für w
  for all Endkonfigurationen y_f, |y_f| ≤ log(n) do
    if PFAD(x_0, y_f, p(n))
    then PFAD := true; STOPP
  od
end.
```

## Analyse des Algorithmus und Endergebnis

Pro Rekursionsschritt benötigt man für jede Konfiguration der $\log(n)$-platzbeschränkten NTM $O(\log(n))$ viel Speicherplatz.
Da die Rekursionstiefe durch $\log(i) \leq \log(p(n)) \in O(\log(n))$ beschränkt ist, folgt für den Gesamtbedarf eine Platzschranke in $O(\log^2(n))$.
Gezeigt wurde somit: $NSPACE\,(\log(n)) \subseteq DSPACE(\log^2(n))$

Mit $f(n) := 2^{s(n)}$ ist $f$ platzkonstruierbar und es folgt

$$
\begin{aligned}
NSPACE\,(s(n)) &= NSPACE\,(\lceil \log(f(n)) \rceil) && \text{Einsetzen} \\
L(f,a) &\in NSPACE\,(\lceil \log(n) \rceil) && \text{Padding} \\
L(f,a) &\in DSPACE\left(\lceil \log(n) \rceil^2\right) && \text{eben gezeigt} \\
L &\in DSPACE\left(\lceil \log(f(n)) \rceil^2\right) && \text{Padding (zurück)} \\
L &\in DSPACE\left(s^2(n)\right) && \text{Einsetzen}
\end{aligned}
$$



## einfache Folgerungen

**Korollar:**

$$
\begin{aligned}
PSPACE &= NPSPACE \\
\bigcup_{c>1} DSPACE\,(c^n) &= \bigcup_{c>1} NSPACE\,(c^n) \\
EXPSPACE &= NEXPSPACE
\end{aligned}
$$

**Beweis** (reine Arithmetik)**:**

$$
\begin{aligned}
&\text{a)} \quad p \in POLY \rightarrow p^2 \in POLY \\
&\text{b)} \quad (c^n)^2 = (c^2)^n \\
&\text{c)} \quad (c^{p(n)})^2 = c^{2 \cdot p(n)}
\end{aligned}
$$

## weitere Bemerkungen

**Bemerkung:**
aus $L = NL$, dh. aus $DSPACE(\log\ n) = NSPACE(\log\ n)$, folgt

$$DLBA = CS$$

*(Zum Beweis benutzt man Padding mit $f(n) := 2^n$)*

Für Zeitkomplexitätsklassen ist eine Beziehung wie die von Savitch für Platzklassen unbekannt!

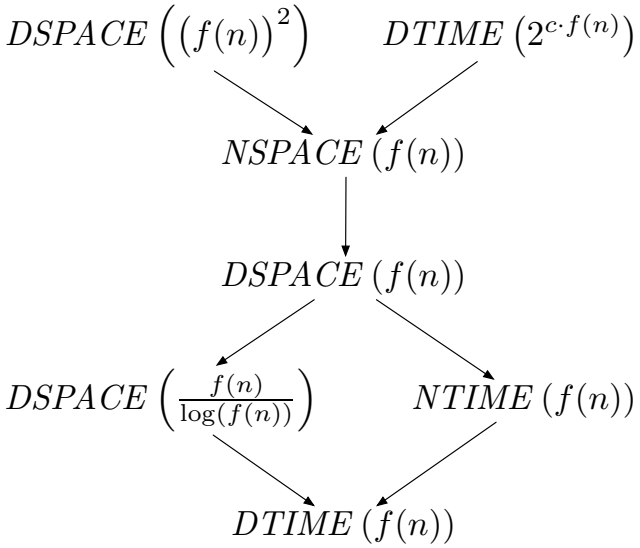
Es gilt sogar:

$$NTIME(n) \subseteq P \text{ genau dann, wenn } P = NP$$

*P, NP* und *Reduktionen* kommen noch später noch ausführlich vor!



## bekannte Relationen zwischen *TIME* und *SPACE*




Es gelte $\forall n: f(n) \geq \log(n)$ und $\rightarrow$ steht für $\supseteq$,
aber nicht alle Inklusionen wurden bisher als echt nachgewiesen! Finden Sie im Verlauf der Veranstaltung heraus, welche Inklusionen tatsächlich als echte Inklusionen nachgewiesen wurden!

## Immermann, Szelepcsény (1987): *NSPACE(f) = co-NSPACE(f)*

**Definition:**
Für $L \subseteq \Sigma^*$ ist $\Sigma^* \backslash L := \{w \in \Sigma^* \mid w \notin L\}$ das Komplement von $L$ (bezüglich $\Sigma^*$).
Für eine Sprachklasse $\mathcal{L}$ sei $co{-}\mathcal{L} := \{\Sigma^* \backslash L \mid L \in \mathcal{L} \wedge L \subseteq \Sigma^*\}$.

**Satz:**
Für $s(n) \geq \log(n)$ gilt:

$$co{-}NSPACE\,(s(n)) \subseteq NSPACE\,(s(n))$$

Wir skizzieren den Beweis nur für platzkonstruierbare Funktionen $s$, er gilt aber ebenso für beliebige Funktionen (schöne Darstellung in: [K. Rüdiger Reischuk: Komplexitätstheorie, Band I: Grundlagen, Teubner (1999)])



## $co{-}NSPACE(\log\, n) \subseteq NSPACE(\log\, n)$

Aus obigem Ergebnis folgt $co{-}NSPACE(s(n)) \subseteq NSPACE(s(n))$ für beliebige (platzkonstruierbare) Funktionen $s$ über die bekannten Padding- oder Translations-Techniken.

In folgender Beweisskizze sei nun:
$M_1$ eine *NTM* mit $L := L(M_1) \in \underline{NSPACE}(\log\, n)$.
Gesucht wird log-platzbeschränkte $M_2$ mit $L(M_2) = \Sigma^* \backslash L$.

**Definition:**
Für $x \in \Sigma^*$, $n := |x|$ sei
$\mathcal{E}_t(x) := \{K \in KONF_{M_1} \mid K_0^x \overset{\leq t}{\underset{M_1}{\vdash}} K\}$ die Menge der Konfigurationen, die in $M_1$ bei Eingabe von $x$ in höchstens $t$ Schritten aus der Startkonfiguration $K_0^x$ erreichbar sind.

$\mathcal{E}(x) := \bigcup_{t \geq 0} \mathcal{E}_t(x), \quad N_t := |\mathcal{E}_t(x)|, \quad N := |\mathcal{E}(x)|.$

Aus früheren Beweisen wissen wir, dass $N \leq p(n)$ für ein Polynom $p \in O(n^3)$ und damit $\log(N) \leq \log(p(n)) \in O(\log n)$ ist.

## Der Hauptteil

Ist $N$ bekannt, so akzeptiert folgender Algorithmus die Wörter der Sprache $\Sigma^* \backslash L$, wobei $M_1$ und $N$ fest sind:

**Eingabe:** $x \in \Sigma^*$ mit $n := |x|$

**begin**
$i$: **integer**; $K_0^x, K$: **KONF**; $i := 0$ $\quad (* \; i$ zählt bis $N \; *)$
**for each** $K$ mit $|K| = \log n$ **do**
**if** $K_0^x \vdash_{M_1}^* K$ **then**
**begin** $i := i + 1$;
**if** $K$ ist akzeptierend **then** *verwerfe* und **stopp**
**end**;
**if** $i = N$
**then** *akzeptiere* und **stopp**
**else** *verwerfe* und **stopp**
**od** $\quad (*$ end for each $*)$
**end.**



## Analyse und wichtige Nebenrechnung

Dieser Algorithmus kann von $\log(n)$-platzbeschränkter NTM ausgeführt werden, weil:

- $N \leq p(n) \in O(n^3)$ und damit benötigt die Binärdarstellung von $N$ nur $\log(N) \in O(\log(n))$ Platz.
- für die Konfigurationen $K$ gilt ebenfalls $|K| \in O(\log(n))$ und diese können systematisch erzeugt werden!

Was fehlt ist die Bestimmung von $N := |\mathcal{E}(x)|$ in $NSPACE\,(\log n)$, um so mit diesem Algorithmus die Frage $x \in \bar{L} := \Sigma^* \backslash L$ mit einer insgesamt $\log(n)$-beschränkten NTM zu entscheiden!

Wir berechnen induktiv $N_0, N_1, \ldots$ beginnend bei $N_0 := 1$. Wenn $N_t$ berechnet sei, so berechnet folgender Algorithmus den nächsten Wert $N_{t+1}$:

## wichtige Nebenrechnung

**Eingabe:** $x \in \Sigma^*$ mit $n := |x|$ und $N_t$

**begin**
$m := 0$ $(*\ m$ zählt bis $N_{t+1}\ *)$
**for each** $K$ mit $|K| = \log n$ **do**
**begin** $i := 0$ $(*$ zählt bis $N_t\ *)$
$found :=$ **false** ;
**for each** $K'$ mit $|K'| = \log n$ **do**
**if** $K_0^x \vdash_{M_1}^{<t} K'$ **then**
**begin** $i := i + 1$;
**if** $K = K'$ **or** $K' \vdash_{M_1}^{1} K$ **then** $found :=$ **true**;
**end**;
**od**; $(*$ end for K' $*)$
**if** $i < N_t$ **then** *verwerfe* und **stopp**;
**if** *found* **then** $m := m + 1$;
**end**



## Komplementabschluss (Ende)

Wegen $K \in \mathcal{E}_{t+1}(x) \leftrightarrow \exists K' \in \mathcal{E}_t(x) : K' = K$ oder $K' \vdash_{M_1}^{1} K$, zählt $i$ bis $N_t$ und $m$ bis $N_{t+1}$.

Da alle Konfigurationen $\log(n)$-platzbeschränkt sind, kann der Algorithmus von einer NTM in $NSPACE\,(\log n)$ ausgeführt werden.

Iteration bis zum ersten $t$ mit $N_t = N_{t+1}$ liefert $N = N_t$.

Insgesamt folgt also das gewünschte Ergebnis:

$$\bar{L} = \Sigma^* \backslash L \in NSPACE\,(\log n)$$

**Korollar:**

$$NL = co\text{-}NL \quad \text{und} \quad co\text{-}\mathcal{C}s = \mathcal{C}s \quad \text{und} \quad co\text{-}PSPACE = PSPACE$$

## Fragen zum Komplementabschluss

Gilt Komplementabgeschlossenheit für alle deterministischen Komplexitätsklassen?

Zum Beispiel gilt doch $co-P = P$, weil für $L \in DTIME\,(p(n))$ ein Wort $w$ nach $p(n)$ Schritten entweder akzeptiert oder abgelehnt wurde.

Gilt also stets $DTIME\,(f(n)) = co\text{-}DTIME\,(f(n))$?

**nein!**

Wenn $f$ keine zeitkonstruierbare Funktion ist, kann man diese Eigenschaft nicht auf diese Art beweisen!