Theoretische Informatik
WS 2024/2025

Kapitel 4

Von Typ-0 Grammatiken über Turingmaschinen zur NP-Vollständigkeit

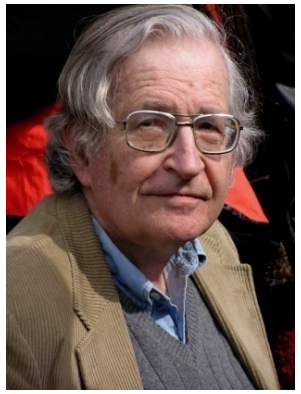
Prof. Dr. Robert Preis
Fachbereich Informatik
Fachhochschule Dortmund
Robert.Preis@fh-dortmund.de

Die Chomsky-Hierarchie zur Klassifizierung von Grammatiken

Avram Noam Chomsky

geb. 7.12.1928 in Philadelphia, USA

Professor für Linguistik am
Massachusetts Institute of Technology (MIT)
→ **http://de.wikipedia.org/wiki/Noam_Chomsky**



Typ	Typ-3 ⊂	Typ-2 ⊂	Typ-1 ⊂	Typ-0
Grammatik	$X \rightarrow aY$	$X \rightarrow aYXbY$	$aXb \rightarrow aYb$	$aXYb \rightarrow aXb$
Sprache	regulär	kontextfrei	kontextsensitiv	allgemein
Beispiel	$(ab)^n$	$a^n b^n$	$a^n b^n c^n$	$\{a^2, a^4, a^8, a^{16} \ldots\}$
Maschine	Endlicher Automat	Nichtdet. Kellerautomat	Linear beschränkter Automat	Turing-maschine

Eine Typ-0 Sprache

Alphabet = {a}

Sprache = {aa,aaaa,aaaaaaaa,…} = {a^2, a^4, a^8, a^{16}, a^{32}, …}

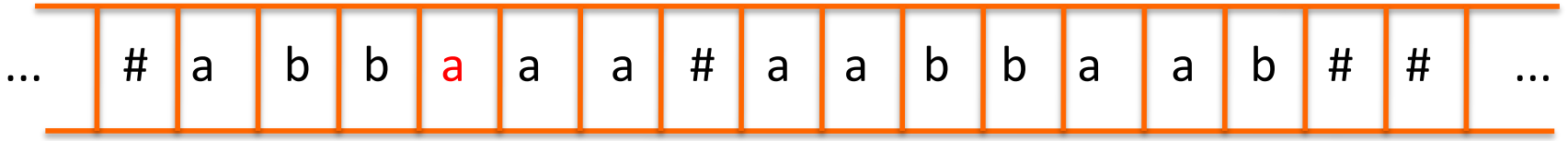
Grammatik:

1. $S \rightarrow ACaB$	generiere Grenzen A und B
2. $Ca \rightarrow aaC$	verdopple rechtes ‚a‘
3. $CB \rightarrow DB$	am rechten Ende angekommen
4. $aD \rightarrow Da$	gehe zurück nach links
5. $AD \rightarrow AC$	links angekommen, Neuanfang
6. $CB \rightarrow E$	Ende, lösche rechten Rand
7. $aE \rightarrow Ea$	gehe nach links um…
8. $AE \rightarrow \varepsilon$	…auch linken Rand zu löschen

Turingmaschine (TM) für Typ-0

(Alan Mathison Turing, England, 1912-1954)

Unendliches Speicherband:



Schreib-/Lesekopf

Zustand: q

Regeln/Übergangsfunktion/Programm:

Falls die TM im Zustand q ist und der Kopf das Zeichen a liest,

dann gehe in Zustand q‘, überschreibe a mit a‘ und bewege den Kopf nach rechts, links oder nicht.

Turingmaschine aus Lego

http://vimeo.com/44202270#

Definition einer Turingmaschine

Eine TM ist ein 6-Tupel TM=(Q, Σ, Γ, δ, S, E) und besteht aus

Alphabeten

- Σ Eingabealphabet
- $\Gamma = \Sigma \cup \{\#, \ldots\}$ Bandalphabet

Zuständen

- Q Menge von Zuständen
- $S \in Q$ Startzustand
- $E \subseteq Q$ Endzustände

Übergangsfunktion

- δ: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,N,R\}$

Ergänzungen zu einem endlichen Automaten

Eine TM akzeptiert ein Wort, wenn dieses auf dem Band steht und von einem Start- einen Endzustand erreicht.

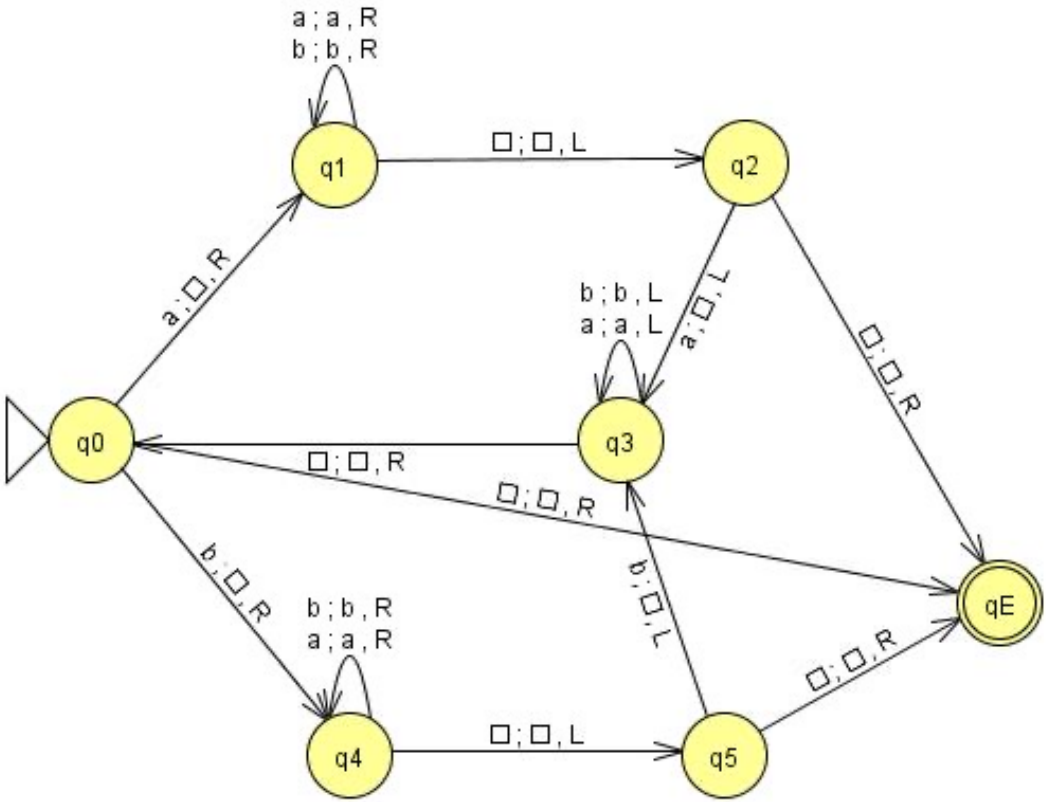
Turingmaschine für Palindrome aus {a,b}*, z.B. ‚abba' oder ‚abbbaabbba'

$\Sigma = \{a,b\}$ $\quad Q = \{q_0, q_1, \ldots\}$ $\quad S = q_0$

$\Gamma = \{a,b,\#\}$ $\quad E = \{q_E\}$

δ	a	b	#	
q_0	q_1, #, R	q_4, #, R	q_E, #, R	Erstes Zeichen merken
q_1	q_1, a, R	q_1, b, R	q_2, #, L	Rechts bis #
q_2	q_3, #, L	-	q_E, #, R	vergleichen
q_3	q_3, a, L	q_3, b, L	q_0, #, R	Links bis #
q_4	q_4, a, R	q_4, b, R	q_5, #, L	Rechts bis #
q_5	-	q_3, #, L	q_E, #, R	vergleichen
q_E	-	-	-	

Turingmaschine für Palindrome aus {a,b}*, z.B. ‚abba' oder ‚abbbaabbba'



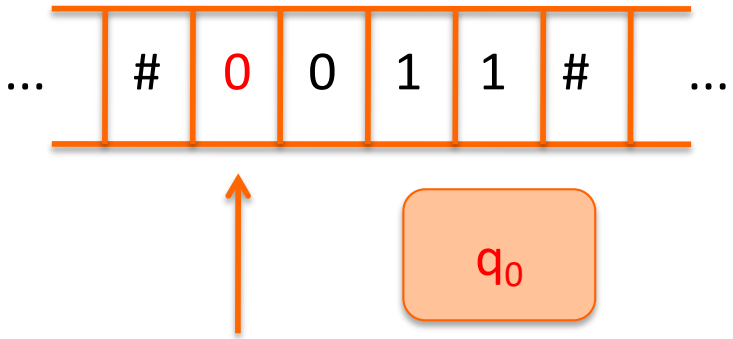
Arbeitsweise einer Turingmaschine

Arbeitsweise bei Eingabe $x \in \Sigma^*$:

Zu Beginn:

- Zustand ist q_0, der Startzustand.
- Auf dem Band steht x (mit jeweils einem Zeichen von x pro Speicherzelle), sonst nur Doppelkreuze.
- Der Kopf steht auf dem ersten Zeichen von x (bzw. auf einem beliebigen Doppelkreuz, falls $x = \varepsilon$).

Bei x = 0011:



x wird akzeptiert g. d. w. ein akzeptierender Endzustand $q \in F$ erreicht wird.

Die Sprache eine Turingmaschine

1. Eine DTM M akzeptiert Eingabe $x \in \Sigma^*$, falls es eine Rechnung gibt, die mit x einen Endzustand erreicht.
2. M akzeptiert eine Sprache $L \subseteq \Sigma^*$, falls sie jedes $x \in L$, aber kein $x \in \Sigma^* \setminus L$ akzeptiert (d.h. sie muss im negativen Fall nicht halten).

- *Für jede Typ-0-Grammatik G gibt es eine DTM, die L(G) akzeptiert.*
- *Für jede DTM M gibt es eine Typ-0-Grammatik, die L(M) erzeugt.*

Deterministische Turingmaschinen passen genau zu Typ-0 !

Eingabe und Ausgabe: Eine DTM M berechnet die Funktion $f: \Sigma^* \rightarrow (\Gamma \setminus \{\#\})^*$ mit $f(x) = \beta$. Am Ende steht die Ausgabe β auf dem Band und wird begrenzt durch:

- Links die Speicherzelle, auf die der Kopf zeigt.
- Rechts die Speicherzelle, die vor dem ersten # ist.
- ... auf dem restlichen Band kann irgendetwas stehen (nicht nur #‘s...)

Church-Turing Hypothese

„Die im intuitiven Sinne berechenbaren Funktionen sind genau die, die durch Turingmaschinen berechenbar sind.“

Gödelisierung von Turingmaschinen

Vereinfachungen von Turingmaschinen, die nicht ins Gewicht fallen:

- Das Alphabet besteht aus {0, 1}.
- Das Bandalphabet besteht aus {0, 1, #}.
- Die Zustände gehen von q_1 (Startzustand) bis q_n (einziger Endzustand).

1. Kodiere das Bandalphabet und die Kopfrichtung:
   $X_1 = 0, X_2 = 1, X_3 = \#$
   $D_1 = L, D_2 = R, D_3 = N$
2. Kodiere eine einzelne Regel:
   $\delta(q_i, X_j) = (q_k, X_l, D_m)$ wird zu $0^i10^j10^k10^l10^m$
3. Kodiere die Übergangsfunktion mit g Zeilen als Gödelnummer:
   $M = \langle M \rangle = 111\text{Code}_1 11\ \text{Code}_2 11 \text{Code}_3 \ldots 11 \text{Code}_g 111$

Die Gödelnummer ist eine Binärzahl, die eine TM darstellt.

Entschlüsselung einer Gödelnummer

Ist 11101010100100110100101010011010001001001011001010010001011001000100010001000100111 eine Gödelnummer, d.h. repräsentiert es eine Turingmaschine?

1. Finde die trennenden Doppeleinsen mit jeweils 4 Einsen dazwischen:
   11101010100100110100101010011010001001001011001010010001011001000100010001000100111
2. Dekodiere die Regeln, benutze dabei

Zustände:	$0^x = q_x$
Alphabet:	0^1=0, 0^2=1, 0^3=#
Kopfbewegung:	0^1=L, 0^2=R, 0^3=N
$0^1 1 0^1 1 0^1 1 0^2 1 0^2$	$\delta(q_1, 0) = (q_1, 1, R)$
$0^1 1 0^2 1 0^1 1 0^1 1 0^2$	$\delta(q_1, 1) = (q_1, 0, R)$
$0^1 1 0^3 1 0^2 1 0^2 1 0^1$	$\delta(q_1, \#) = (q_2, 1, L)$
$0^2 1 0^1 1 0^2 1 0^3 1 0^1$	$\delta(q_2, 0) = (q_2, \#, L)$
$0^2 1 0^2 1 0^2 1 0^2 1 0^1$	$\delta(q_2, 1) = (q_2, 1, L)$
$0^2 1 0^3 1 0^3 1 0^3 1 0^2$	$\delta(q_2, \#) = (q_3, \#, R)$

Ergo:
Jede Turingmaschine ist eine Binärzahl (Gödelnummer) !

Nicht lösbar?

Gibt es denn auch Sprachen, die außerhalb von Typ-0 sind?

D.h. gibt es Probleme, für die es

- keine Grammatik gibt?
- keine Turingmaschine gibt?
- kein Software-Programm gibt?

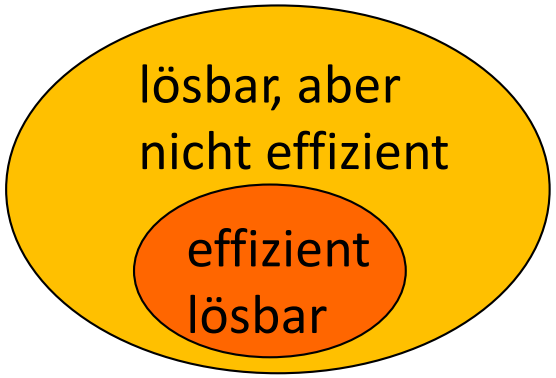
Fundamentale Fragen in der Theoretischen Informatik:

Was können Computer *überhaupt* lösen?

Berechenbarkeit

Was können Computer *effizient* lösen?

Komplexitätstheorie



Unlösbare Probleme ?

Sage ich bei dem Satz

„Ich sage jetzt nicht die Wahrheit.“

die Wahrheit oder nicht ?

Behauptung: Ich sage die Wahrheit!

Dann würde der Satz ja stimmen.

D.h. ich würde nicht die Wahrheit…das ist ein Widerspruch!

Behauptung: Ich sage nicht die Wahrheit!

Dann würde ich ja lügen und der Satz ja nicht stimmen.

Dann würde ja das Gegenteil *„Ich sage jetzt die Wahrheit“* stimmen.

Das ist aber ein Widerspruch zur Behauptung!

Keine der beiden Behauptungen ist wahr !

Unlösbare Probleme II ?

In Dortmund wohnt ein Barbier der genau diejenigen männlichen Einwohner von Dortmund rasiert, die sich nicht selbst rasieren.

PROBLEM: Rasiert sich der Barbier selbst oder nicht ?

Behauptung: Der Barbier rasiert sich selbst!

Geht nicht, weil er nur die Gruppe von Männern rasiert, die sich nicht selbst rasieren, und zu dieser Gruppe gehört er nicht!

Behauptung: Der Barbier rasiert sich nicht selbst!

Geht nicht, denn dann gehört er zu der Gruppe der Menschen, die sich nicht selbst rasieren…und genau diese Gruppe wird vom ihm rasiert und somit müsste er sich selbst rasieren!

Keine der beiden Behauptungen ist wahr !

Deshalb kann es keinen solchen Barbier geben !

Problem der Endlosschleife

Annahme: *Es gibt Programm „**Halten(P,x)**“, das bei Eingabe eines Programms P und einer Eingabe x entscheidet (Ausgabe ja/nein), ob das Programm P gestartet mit Eingabe x hält (d.h. keine Endlosschleife).*

Dann gibt es auch folgendes Programm **Widerspruch(P)**:

Falls Halten(P,P) „ja“ liefert, gehe in Endlosschleife, sonst halte an.

Behauptung: *Widerspruch(Widerspruch) hält!*

Dann würde Halten(W.,W.), „ja“ liefern, d.h. W.(W.) würde in eine Endlosschleife gehen … Widerspruch zur Behauptung !

Behauptung: *Widerspruch(Widerspruch) hält nicht!*

Dann würde Halten(W.,W.) „nein“ liefern, d.h. W.(W.) würde anhalten … Widerspruch zur Behauptung !

Keine der beiden Behauptungen ist wahr !

Also kann es kein Programm *Widerspruch(P)* geben !

Also kann es kein Programm *Halten(P,x)* geben !

Einige unentscheidbare Probleme (es gibt dafür keine DTM, die diese akzeptiert)

Komplement des Halteproblem

$H^c = \{<M>x \mid M \text{ ist DTM, die gestartet mit Eingabe x nicht hält}\}$

Diagonalsprache

$DIAG = \{<M> \mid M \text{ ist DTM, die } <M> \text{ nicht akzeptiert}\}$

Äquivalenzproblem

$Ä = \{<M>, <M´> \mid L(M) = L(M´)\}$

Totalitätsproblem

$T = \{<M> \mid L(M) = \Sigma^*\}$

Diagonalisierung für DIAG

Satz: DIAG ist unentscheidbar (es gibt keine DTM, die DIAG akzeptiert).

Beweisidee: Seien M_1, M_2,… alle DTMs in der Reihenfolge ihrer Gödelnummern.

Annahme: Es existiert DTM M (o.b.d.A. sei $M=M_i$), die DIAG akzeptiert.

Betrachte die unendliche Matrix mit Zeilen $<M_1>$, $<M_2>$, … und Spalten M_1, M_2,…:

„a“ falls M_i $<M_j>$ akzeptiert

„na“ falls M_i $<M_j>$ nicht akzeptiert

Wichtig ist die Diagonale, d.h. ob eine DTM ihre eigene Gödelnummer akzeptiert oder nicht.

	M_1	M_2	M_3	…	M
$<M_1>$	a	na	a		na
$<M_2>$	na	na	a		a
$<M_3>$	a	a	na		a
…				…	…
					?

Beobachtung: Die Spalte von M ist genau wie die Diagonale, nur jeweils mit vertauschten „a“ und „na“.

Ergebnis: Somit kann es M nicht geben … Widerspruch!

Ausblick Berechenbarkeit im Master: Nicht entscheidbare Sprachen

Diagonalsprache

DIAG = {<M> | M akzeptiert <M> nicht}

Halteproblem

H = {<M>x | M mit Eingabe x hält}

Äquivalenzproblem

Ä = {<M>, <M´> | L(M) = L(M´)}

Akzeptanzproblem

A = {<M>x | M akzeptiert x}

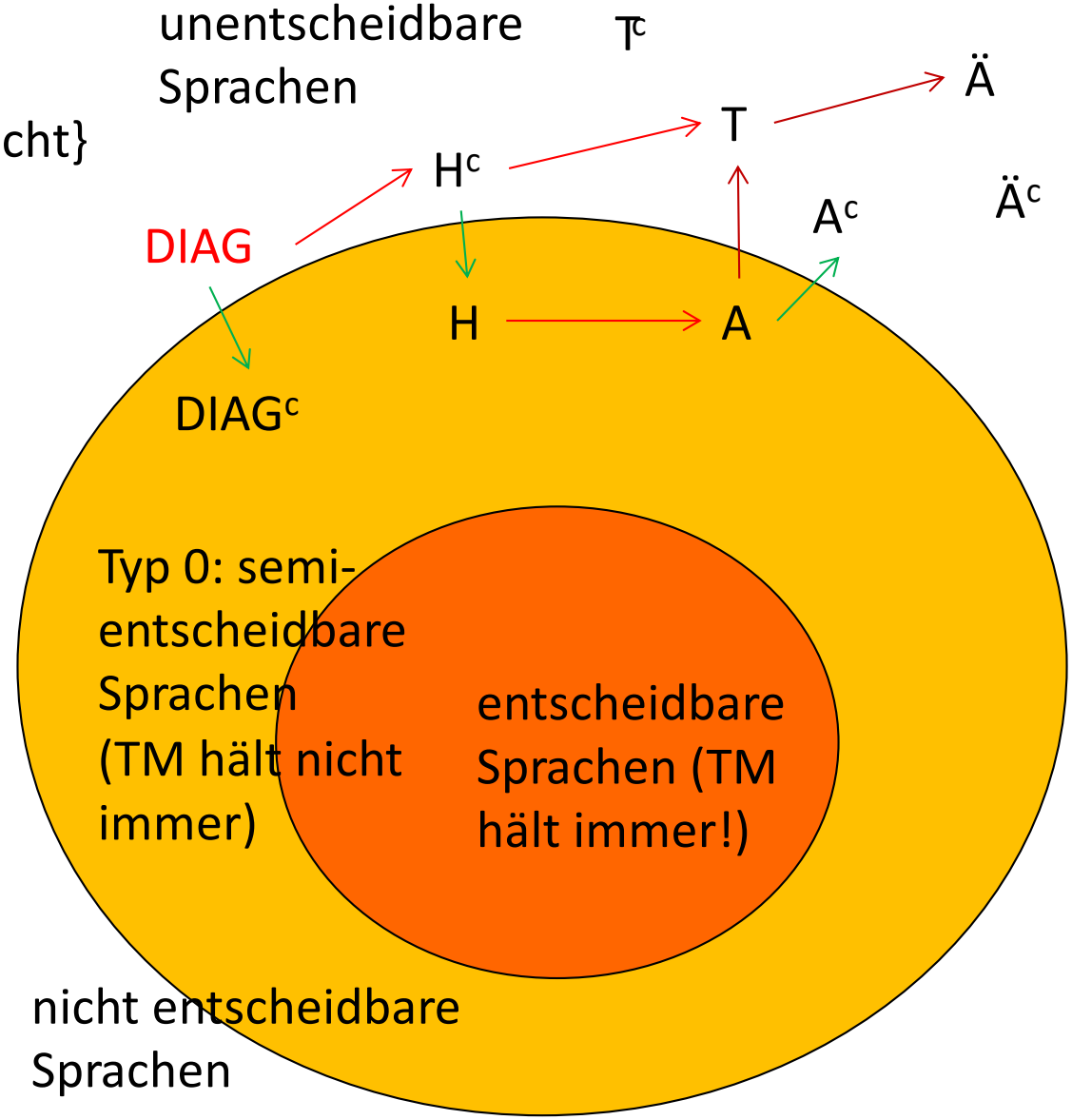
Totalitätsproblem

T = {<M> | L(M) = Σ*}

<M> ist die Gödelnummer der Turingmaschine M

→: durch Reduktionen

→: durch Entscheidbarkeitssatz



Nichtdeterministische Turingmaschinen

Eine nichtdeterministische Turingmaschine kann durch einen 6-Tupel

$$M = (Q, \Sigma, \Gamma, \delta, q_0, F)$$

beschrieben werden.

- $Q, \Sigma, \Gamma, q_0, F$ sind wie bei deterministischen TMs,
- δ ist nun wie folgt definiert:

  $$\delta: Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{R, N, L\})$$
- $\delta(q, a) = \varnothing$, falls $q \in F$ ist.

Beispiel:

NTM mit $\Gamma=\{0,1,\#\}$, $\Sigma=\{0,1\}$, $Q=\{q_0,q_1\}$, $F=\{q_1\}$

und Übergangsfunktion $\delta(q_0,\#) = \{(q_0,0,R), (q_0,1,R), (q_1,\#,N)\}$

Was macht diese NTM?

Umwandlung NTM in DTM

Satz: *Jede NTM M kann durch eine DTM M‘ simuliert werden, braucht aber eventuell exponentiell viel Zeit.*

Beispiel:	NTM mit n Schritten	DTM mit 2^n Schritten
n=1:	1	$2^1 = 2$
n=10:	10	$2^{10} = 1.024$
n=100:	100	$2^{100} = 1{,}27*10^{30}$
n=1.000:	1.000	$2^{1.000} = 1{,}07*10^{301}$

Und wenn man den schnellsten Rechner der Welt verwendet ($33*10^{15}$ Schritte pro Sekunde) ?

Bei n=100 und $2^{100} = 1{,}27*10^{30}$ Schritten: $3{,}84*10^{13}$ Sekunden = 1.218.089 Jahre !

Problem: Alle existierenden physikalischen Rechner sind deterministisch.

…geht das nicht besser als exponentiell?

Wer von Ihnen das herausbekommt, bekommt eine 1,0 in der Klausur!

Der exponentielle Zeitaufwand ist die Grundlage für das P-NP Problem !

Komplexitätstheorie: Wie schnell kann ich ein Problem lösen?

Wie viele Schritte brauche ich, wenn die Eingabe aus n Elementen besteht und der Aufwand abhängig von n ist ?

Aufwand:	**n=1**	**n=10**	**n=100**	**n=1.000**	**n=1.000.000**
1	1	1	1	1	1
log(n)	0	1	2	3	6
$\sqrt{n}$	1	3	10	31	1.000
n	1	10	100	1.000	1.000.000
n*log(n)	0	10	200	3.000	6.000.000
n^2	1	100	10.000	1.000.000	10^{12}
n^3	1	1.000	1.000.000	1.000.000.000	10^{18}
2^n	2	1.024	$\approx 10^{30}$	$\approx 10^{300}$	$\approx 10^{300.000}$
n!	1	$\approx 3*10^6$	$\approx 10^{158}$	$\approx 10^{2.568}$	$\approx 10^{5.500.000}$
n^n	1	10^{10}	$\approx 10^{200}$	$\approx 10^{3.000}$	$\approx 10^{6.000.000}$

Effizient (polynomiell)

Uneffizient (exponentiell)

Das P-NP Problem: Was ist in polynomieller Zeit (effizient) möglich?

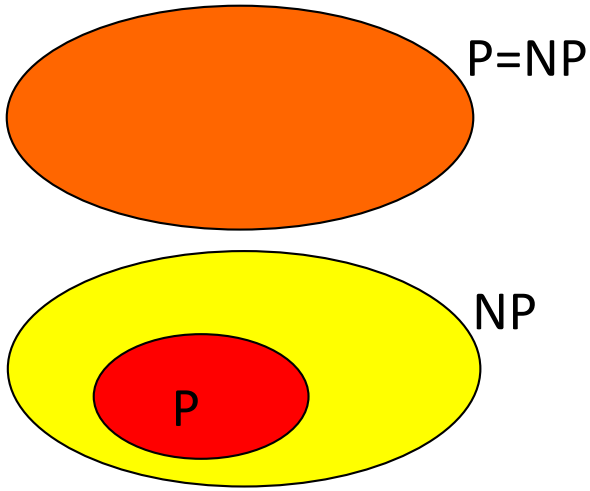
Polynomielle Zeiten auf unterschiedlichen Turingmaschinen:

- **P** ist die Klasse der Sprachen, die von einer *deterministischen* TM in polynomieller Zeit entschieden werden können.
- **NP** ist die Klasse der Sprachen, die von einer *nicht-deterministischen* TM in polynomieller Zeit entschieden werden können.

Es ist klar, dass P $\subseteq$ NP gilt. Frage:

Gilt P = NP?

Oder P ≠ NP ?



Es ist eines von sieben *Millennium Prize Problems* des Clay Mathematics Institute (CMI) of Cambridge, Massachusetts, USA.

Preisgeld: US $ 1.000.000

Wissenschaftliche Ehrung: Unbezahlbar !