Grundlagen Theoretischer Informatik 2

WiSe 2009/10 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

Grundlagen Theoretischer Informatik 2 Gesamtübersicht

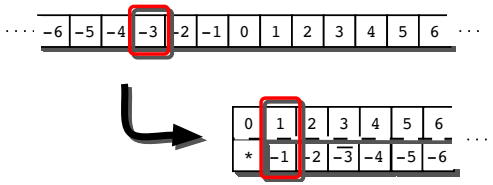
- Organisatorisches; Einführung
- Ersetzungsverfahren: Grammatiken und Automaten
- Rekursionstheorie-Einführung
- Komplexitätstheorie-Einführung

TM mit rechtsseitig unendlichem Band

Bislang: Bänder links- und rechtsseitig unendlich
Oft in der Literatur: Band hat *linkes Endezeichen* $\triangleright$.
Dieses darf nur gelesen, aber nicht verändert werden; ein weiteres Laufen nach links über $\triangleright$ hinaus ist unzulässig.
Sonst: TM-Übergänge, -Akzeptanz, -Laufzeit etc. wie zuvor.

Satz: TM mit (nur) rechtsseitig unendlichem Band ist zur "normalen" TM gleichmächtig. Genauer: Ist $\mathcal{M}$ TM mit links- und rechtsseitig unendlichem Band, die L in Polynomzeit akzeptiert, so gibt es TM $\mathcal{M}'$ mit rechtsseitig unendlichem Band, die ebenfalls L in Polynomzeit akzeptiert.

Beweis: (Idee) Falte Turingband von $\mathcal{M}$ und markiere linkes Ende mit $\triangleright$.



Noch eine Normalform

Wann akzeptiert eine Turingmaschine ein Wort?
In der Literatur finden Sie verschiedene Möglichkeiten, z.B.:

— Akzeptanz durch Endzustände
— Akzeptanz durch Haltezustand h, wobei dann das Symbol 1 (und nur dies) auf dem Band steht ($\mathrm{h}1$-Akzeptanz).

Satz: TM mit (nur) rechtsseitig unendlichem Band und $\mathrm{h}1$-Akzeptanz ist zur "normalen" TM gleichmächtig.
Genauer: Ist M TM mit links- und rechtsseitig unendlichem Band, die L in Polynomzeit akzeptiert, so gibt es TM M' mit rechtsseitig unendlichem Band, die ebenfalls L in Polynomzeit $\mathrm{h}1$-akzeptiert.

Beweis: (Idee) Erst Falten des Bandes, und dann startet eine "Aufräumphase", falls ein akzeptierender Zustand erreicht wird. Das Aufräumen dauert nicht "zu lange".

Der Satz von Cook

Satz: SAT ist NP-vollständig.

Guess & Check liefert sofort: $SAT \in NP$.

Zu zeigen bleibt: NP-Härte.

Erinnerung: Eine Sprache $L \subseteq E^*$ heißt NP-*hart*,
wenn *alle* Sprachen aus NP auf sie polynomial reduzierbar sind.

Sei also L eine beliebige Sprache aus NP.
Der Weg zu diesem Satz soll in viele Teilschritte untergliedert werden.

Das n-Schritt Halte-Problem

$\text{ACC}_{\leq} = \{\langle \text{T} \rangle \, \$ \, \langle w \rangle \, \$^n \mid \text{NTM T akzeptiert } w \text{ in höchstens } n \text{ Schritten} \}$
(Wie üblich bezeichne $\langle \text{T} \rangle$ eine Codierung der TM T (usf.).)

Satz: $\text{ACC}_{\leq}$ ist NP-vollständig.

Beweis: Härte: Da $L \in \text{NP}$, gibt es eine NTM T, die das Wortproblem (Liegt w in L?) in Zeit $p(|w|)$ entscheidet. Also gilt: $w \in L$ gdw. $\langle \text{T} \rangle \, \$ \, \langle w \rangle \, \$^{p(|w|)} \in \text{ACC}_{\leq}$.

Mitgliedschaft in NP: nächste Folie!

Beachte: Dies ist unser erstes (sozusagen generisches) NP-hartes Problem.

"Guess and Check" für $\mathrm{ACC}_{\leq}$

Erinnerung: Mitgliedschaft in NP durch G&C; dabei wird zunächst ein polynomiell langes *Zertifikat* geraten und anschließend deterministisch verifiziert.

Ein Zertifikat von $x = \langle T \rangle \, \$ \, \langle w \rangle \, \$^n \in \mathrm{ACC}_{\leq}$ ist eine Folge

$$(u_1, s_1, v_1), (u_2, s_2, v_2), \ldots, (u_m, s_m, v_m)$$

von Konfigurationen mit $m \leq n$.

Eine geeignete Codierung solch einer Folge hat polynomielle Länge in $|x|$.

In der "Check-Phase" prüft der Algorithmus Folgendes:

- Ist $w = v_1$, $u_1 = \lambda$ und s_1 Anfangszustand?
- Ist s_m Endzustand?
- Für jedes $1 \leq k < m$, teste ob $(q_k, w_k) \vdash_T (q_{k+1}, w_{k+1})$ mit einer UTM.

Die Überprüfung des Zertifikats ist erfolgreich, wenn jeder Teiltest bestanden wird.

ACC Varianten I

$\text{ACC}_= = \{ \langle \text{T} \rangle \, \$ \, \langle w \rangle \, \$^n \mid \text{NTM T akzeptiert } w \text{ in genau } n \text{ Schritten} \}$

Satz: $\text{ACC}_=$ ist NP-vollständig.

Beweis: Mitgliedschaft in NP wie soeben.
Für die Härte zeigen wir: $\text{ACC}_{\leq} \leq_p \text{ACC}_=$.
Modifiziere TM T durch Hinzufügen von "Warte-Regeln" $((q, a), (q, a))$ für alle Zustände q und alle Bandsymbole a. Dies beschreibe TM T'. $\rightsquigarrow$

$$\langle \text{T} \rangle \, \$ \, \langle w \rangle \, \$^n \in \text{ACC}_{\leq} \iff \langle \text{T}' \rangle \, \$ \, \langle w \rangle \, \$^n \in \text{ACC}_=$$

ACC Varianten II

$\text{ACC}'_{=} = \{\langle \text{T} \rangle \, \$ \, \langle w \rangle \mid \text{NTM T akzeptiert } w \text{ in genau } |w| \text{ Schritten } \}$

Satz: $\text{ACC}'_{=}$ ist NP-vollständig.

Beweis: Mitgliedschaft in NP wie soeben.
Für die Härte zeigen wir: $\text{ACC}_{=} \leq_{\mathrm{p}} \text{ACC}'_{=}$.

Gilt $|w| \geq n$, überführe die $\text{ACC}_{=}$-Instanz $\langle \text{T} \rangle \, \$ \, \langle w \rangle \, \$^n$ in eine äquivalente $\text{ACC}'_{=}$-Instanz $\langle \text{T} \rangle \, \$ \, \langle w \rangle$ durch Hinzufügen von Warteschleifen wie im vorigen Beweis.

Andernfalls, definiere $w' = w \sqcup^{|w|-n}$. (Hierbei sei $\sqcup$ das Blank-Symbol.) $\rightsquigarrow$

$$\langle \text{T} \rangle \, \$ \, \langle w \rangle \, \$^n \in \text{ACC}_{=} \iff \langle \text{T} \rangle \, \$ \, \langle w' \rangle \in \text{ACC}'_{=}$$

Berechnungsteppiche

Zu der Instanz $\langle T \rangle \, \$ \, \langle w \rangle$ von $\mathrm{ACC}'_{=}$, mit Initialkonfiguration $\triangleright s w = a_{1,1} \ldots a_{1,n+2}$, kann ein Zertifikat für einen G&C Algorithmus als zweidimensionaler "Teppich" angesehen werden:

$$CC = \begin{array}{cccc} a_{11} & a_{12} & \cdots & a_{1,n+2} \\ a_{21} & a_{22} & \cdots & a_{2,n+2} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n+1,1} & a_{n+1,2} & \cdots & a_{n+1,n+2} \end{array}$$

Klar: polynomielle Teppichgröße (in $|w|$ und $\langle T \rangle$).

Wie überprüft man ein Berechnungsteppich-Zertifikat?

1. $\triangleright s w = a_{1,1} \dots a_{1,n+2}$ (Initialkonfig.),

2. $\triangleright h 1 \sqcup^{n-1} = a_{n+1,1} \dots a_{n+1,n+2}$, h ist Endzustand
   (O.b.d.A.: Normalform für TM-Akzeptanz: $h1$-Akzeptanz)
   und

3. mit Ausnahme der ersten Zeile ist jede Zeile des Teppichs Nachfolgekonfiguration der Zeile darüber.

$\rightsquigarrow$ **"Guess and Check" für $ACC^{\prime}_{=}$**

Rate Berechnungsteppich-Zertifikat und überprüfe es anschließend.

Die Logik von Berechnungsteppichen

Betrachte die Booleschen Variablen $C_{i,j,X}$; diese seien wahr gdw. Symbol X an Position (i,j) eines Berechnungsteppichs steht. Abkürzende Schreibweise:

$$C_{i,j,a_1 \ldots a_t} :\Longleftrightarrow\ C_{i,j,a_1} \wedge C_{i,j+1,a_2} \wedge \cdots \wedge C_{i,j+t-1,a_t}$$

Syntaktische Bedingungen:

- Für jede Position des Teppichs muss ein Zeichen spezifiziert sein:

$$E_{S_1} = \bigwedge_{i,j} \bigvee_{X} C_{i,j,X}$$

- . . . aber nicht mehrere . . .

$$E_{S_2} = \bigwedge_{i,j} \bigwedge_{X} \bigwedge_{Y, X \neq Y} \overline{C_{i,j,X}} \vee \overline{C_{i,j,Y}}$$

Richtiger Anfang und glückliches Ende

- Die Initialkonfiguration, gegeben durch den Anfangszustand s und das Eingabewort $w = a_1 \dots a_n$, wird korrekt beschrieben:

$$E_{S_3}(w) = C_{1,1,\triangleright s w}.$$

- ebenso die Endkonfiguration:

$$E_{S_4} = C_{n+1,1,\triangleright h 1 \sqcup^{n-1}}.$$

Wie werden Übergänge beschrieben?

In Abhängigkeit von der TM T entwerfen wir ein Prädikat $P_T(u;v)$.
Dies beschreibt, ob das Wort $u = u_0u_1u_2$ (wobei eines der drei Zeichen ein Zustandssymbol enthält) bei $C_{i,j,u}$ sein könnte, indem $v = v_0v_1v_2$ bei $C_{i-1,j,v}$ betrachtet wird.

Betrachte die Konfiguration $C = xaqby$ von T.

- Ist $((q,b),(p,c,N))$ ein Übergang von T, so ist $xapcy$ Nachfolger von C, falls q kein Haltezustand ist. Sei also $P_T(aqb;apc) = 1$ für bel. a.
- Ist $((q,b),(p,c,R))$ ein Übergang von T, dann ist $xacpy$ Nachfolger von C, falls q kein Haltezustand ist. Sei also $P_T(aqb;acp) = 1$ für bel. a.
- Ist $((q,b),(p,c,L))$ ein Übergang von T, dann ist $xpacy$ Nachfolger von C, falls q kein Haltezustand ist. Sei also $P_T(aqb;pac) = 1$ für bel. a.

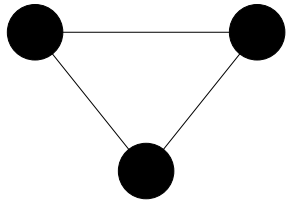
$$E_T = \bigwedge_{i=2}^{n+1} \bigvee_{j=1}^{n} \Big(\bigvee_{P_T(uvw;xyz)} C_{i-1,j,uvw} \wedge C_{i,j,xyz} \Big) \wedge \bigwedge_{k,k\notin\{j,j+1,j+2\}} \bigwedge_{x} C_{i,k,x} \Leftrightarrow C_{i-1,k,x} \Big)$$

Hierbei ist $\bigvee_{P_T(uvw;xyz)}$ Kurzschreibweise für $\bigvee_{u,v,w;x,y,z:P_T(uvw;xyz)}$.

Gesamtausdruck: $E = E_{S_1} \wedge E_{S_2} \wedge E_{S_3} \wedge E_{S_4} \wedge E_T$.

Das beweist den Satz von Cook.

Sei $G = (V, E)$ ungerichteter Graph:
$V' \subseteq V$ ist *Knotenüberdeckung*, wenn jede Kante aus E mindestens einen Endpunkt in V' hat.



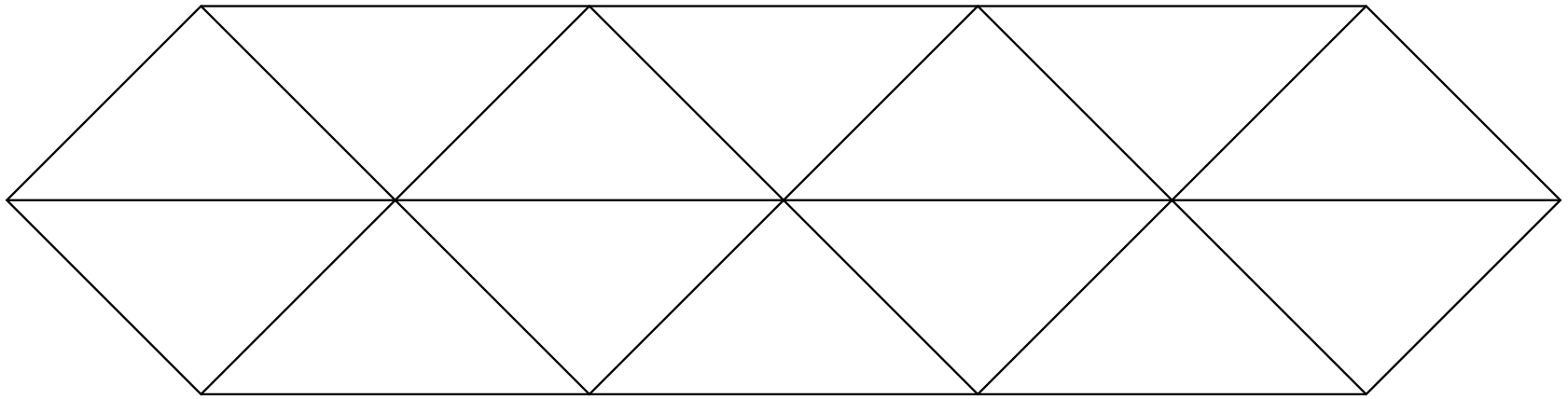
Hier sind zwei Knoten nötig!

$\Rightarrow$ VERTEX COVER als graphentheoretisches Überdeckungsproblem:
VERTEX COVER: Gegeben seien ein ungerichteter Graph $G = (V, E)$ und eine natürliche Zahl k.
Frage: Gibt es eine Menge $V' \subseteq V$ aus höchstens k Knoten, die eine Knotenüberdeckung bildet?

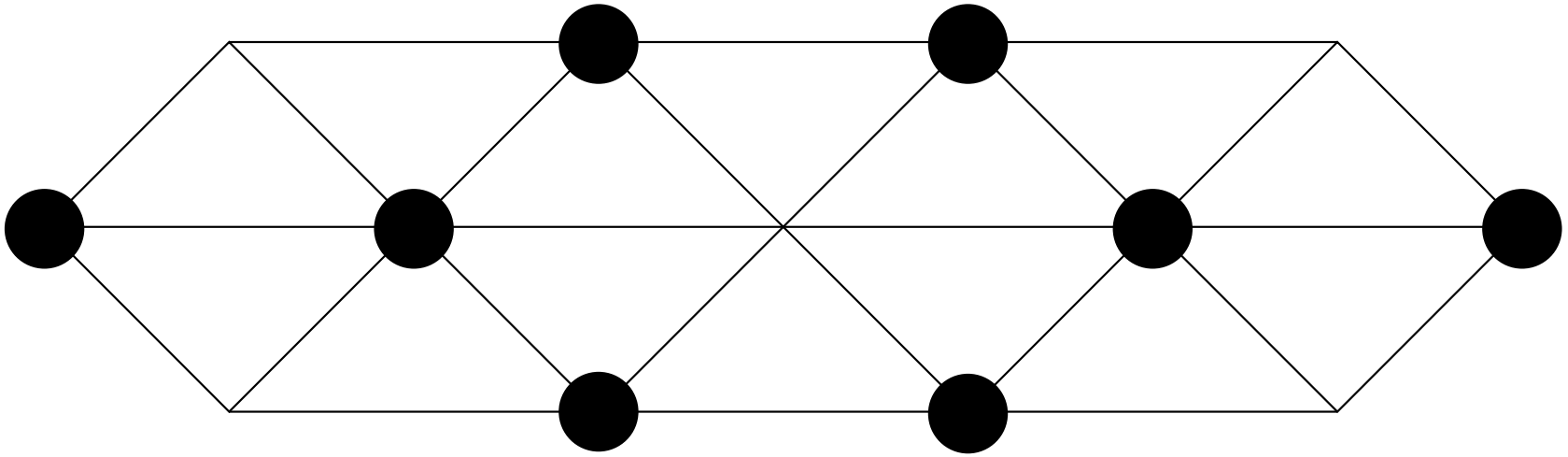
Satz: VERTEX COVER ist NP-hart.

Hinweis: Hierzu Graphen "geeignet" zu codieren!

Beispiel für die Begriffe: Betrachte folgenden Graphen:

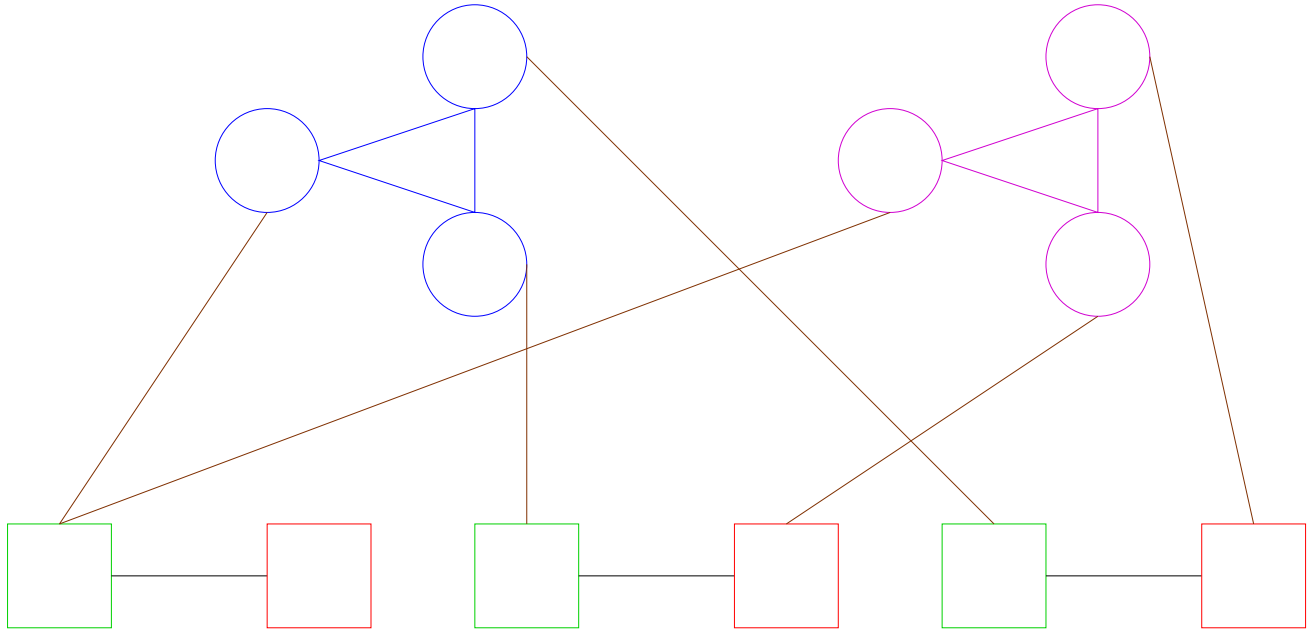


Beispiel: Lösung zu VERTEX COVER mit $k \geq 8$:

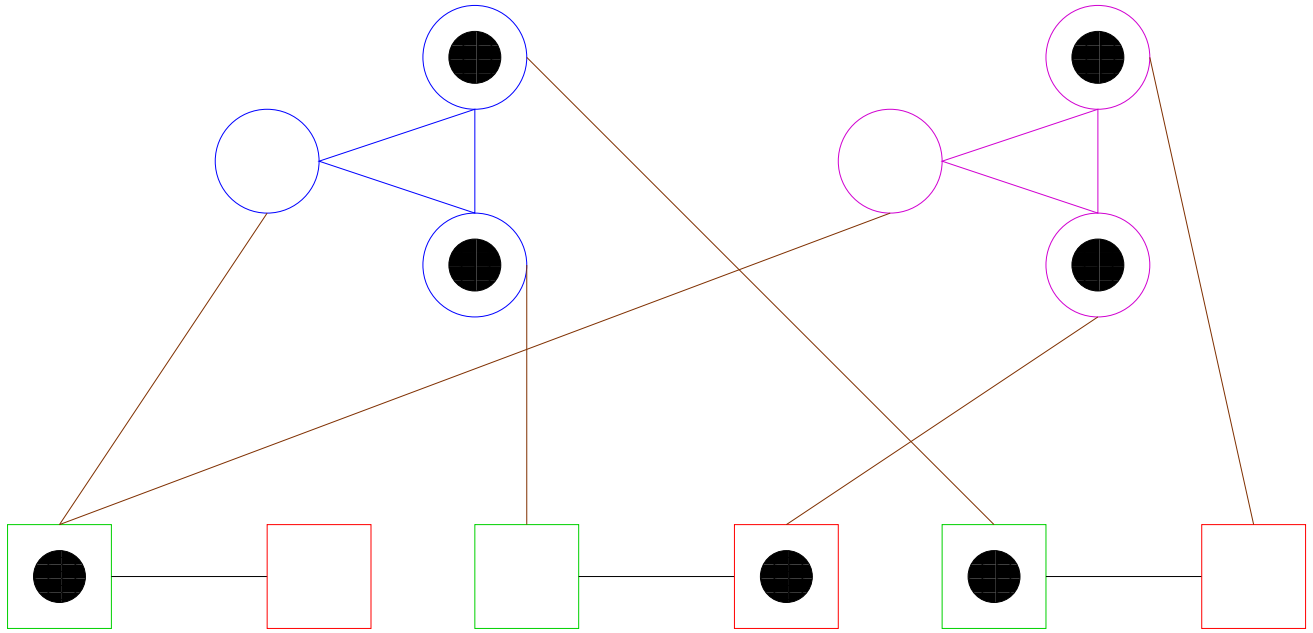


Warum ist Lösung kleinstmöglich? Betrachte Dreiecke!

Grundidee: Codiere $3-SAT'$ (o.ä.) durch "Gadgets" für Variablen und Klauseln.
"Oben": $3m$ Knoten für die m Klauseln, "unten" $2n$ Knoten für die n Variablen.
Kanten zwischen Klauselknoten und Literalknoten kennzeichnen Vorkommen.
Im Beispiel: $(x \vee y \vee z) \wedge (x \vee \bar{y} \vee \bar{z})$. Allg.: Formel $w \mapsto G(w)$.



Konstruktion: $w \in 3-SAT'$ gdw.
$(G(w), 2m + n) \in$ VERTEX COVER.
Im Beispiel: $(x \vee y \vee z) \wedge (x \vee \bar{y} \vee \bar{z})$.



Punkte markieren die Knotenüberdeckung.
Wegen der Dreiecke oben und Kanten unten folgt Minimalität.

Abschließend

Ist alles unbekannt, was Komplexitätsklassen angeht?
Muss man sich stets mit Aussagen abspeisen, die bestenfalls “Vollständigkeit” zeigen?

NEIN!

Durch Diagonalisierung kann man z.B. P von EXP trennen.

Das Polynomialzeit Halte-Problem

$\text{HALT}_P = \{\langle M\rangle \,\$\, \langle w\rangle \mid \text{TM } M \text{ hält auf } w \text{ und liefert 1 in höchstens } 2^{|w|} \text{ Schritten } \}$

Satz: HALT_P ist in Exponentialzeit entscheidbar, liegt aber nicht in P.

Die Entscheidbarkeit in Exponentialzeit ist trivial.
Z.z.: $\text{HALT}_P \notin P$. Andernfalls gölte:

$$H_0 = \{\langle M\rangle \mid \langle M\rangle \,\$\, \langle M\rangle \in \text{HALT}_P\} \in P$$

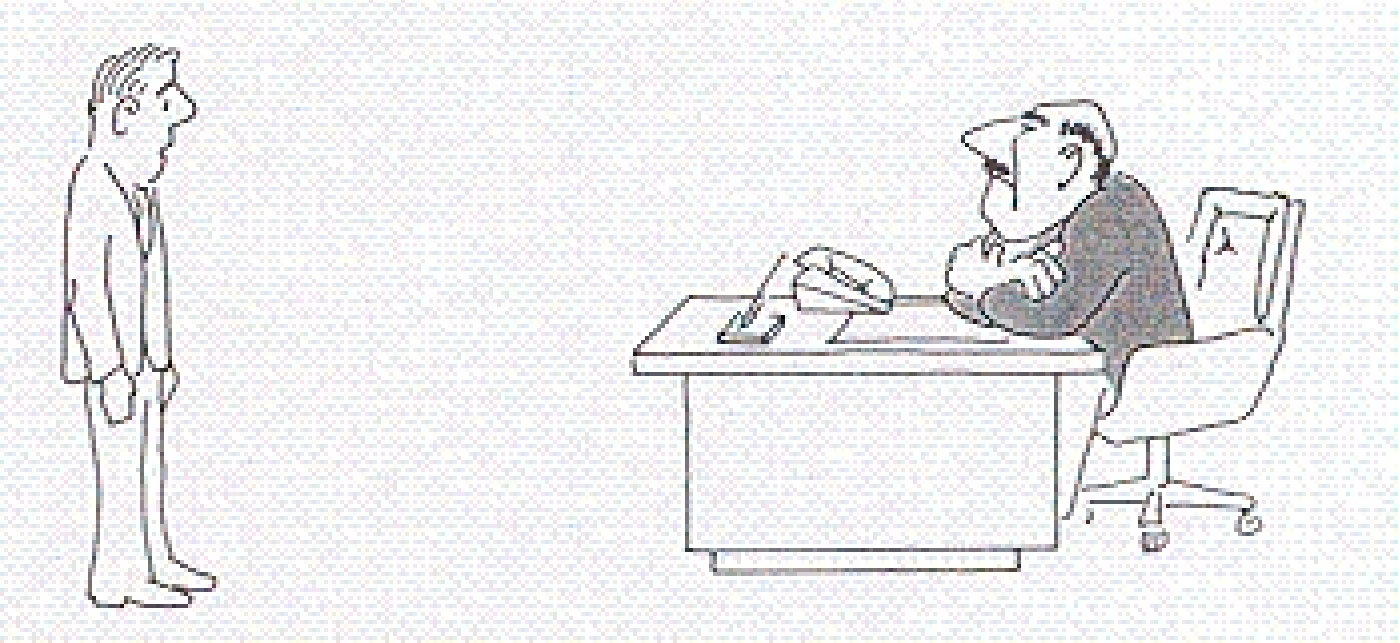
Also läge das Komplement $\overline{H_0}$ ebenfalls in P.
Sei M_0 eine TM, die $\overline{H_0}$ in pol. Zeit p entscheidet, mit $p(x) \leq 2^x$ für alle $x \geq |\langle M_0\rangle|$.

Wenn M_0 das Wort $\langle M_0\rangle$ akzeptiert, dann $\langle M_0\rangle \in L(M_0) = \overline{H_0}$, was der Annahme widerspricht,
H_O enthalte alle TMs, die höchstens $2^{|M_0|}$ Schritte machen, wenn sie ihre eigene Beschreibung als Eingabe bekommen.
$\rightsquigarrow M_0$ akzeptiert $\langle M_0\rangle$ nicht. $\rightsquigarrow \langle M_0\rangle \notin L(M_0) = \overline{H_0}$, d.h., $\langle M_0\rangle \in H_0$.
Nach Definition von H_0 bedeutet dies: M_0 macht $2^{|M_0|}$ Schritte auf $\langle M_0\rangle$, Widerspruch!

Daher liegen H_0 (und folglich ebensowenig HALT_P) nicht in P.

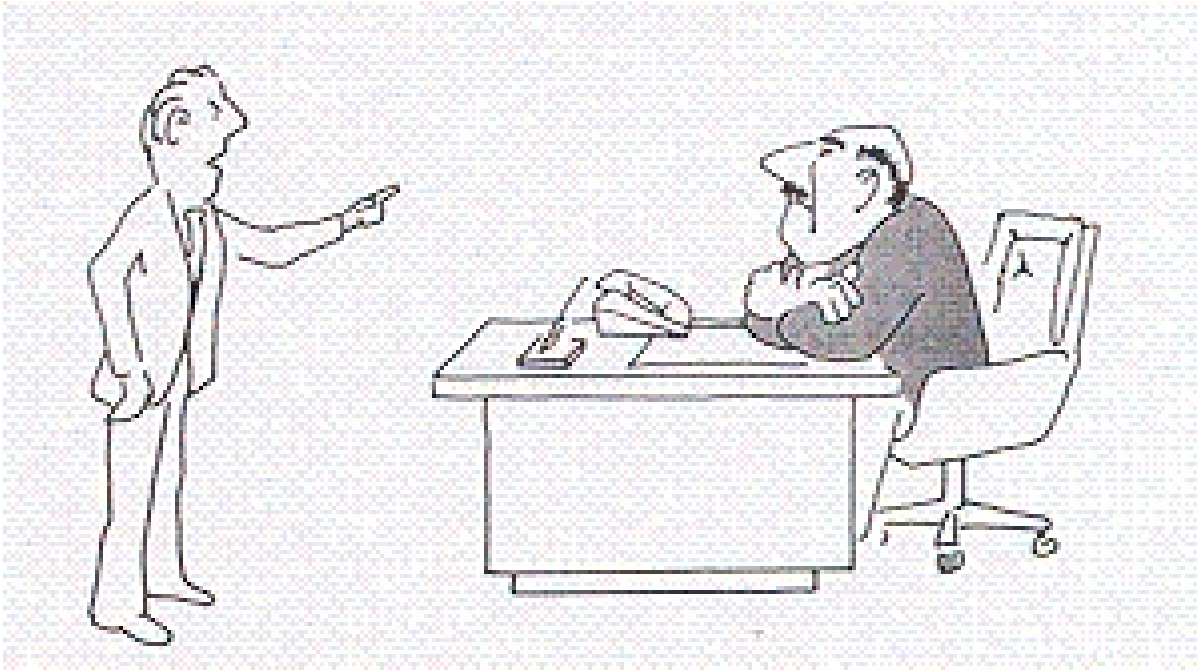
Motivation

siehe `http://max.cs.kzoo.edu/~kschultz/CS510/ClassPresentations/NPCartoons.html` wiederum aus Garey / Johnson



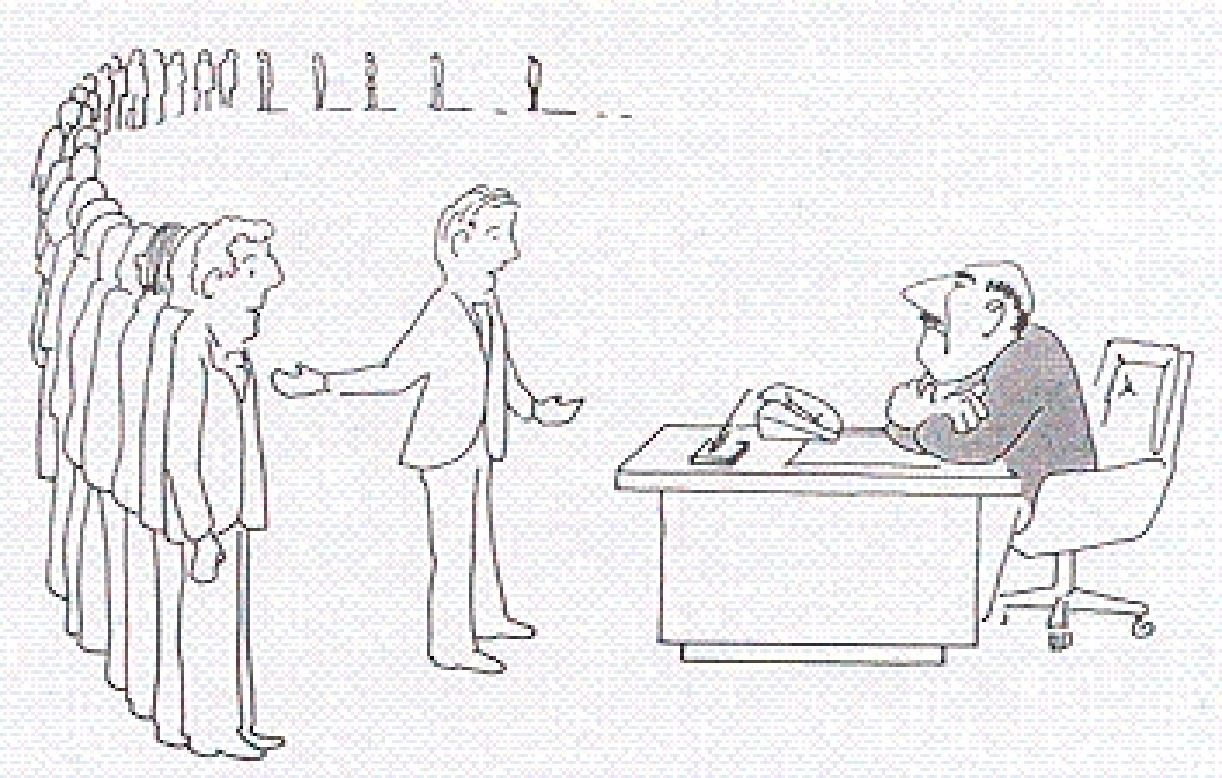
Sorry Chef, aber ich kann für das Problem keinen guten Algorithmus finden. . .

Die beste Antwort wäre hier aber. . .



. . . Ich kann aber beweisen, dass es für das Problem keinen guten Algorithmus geben kann !

Was die Komplexitätstheorie statt dessen liefert. . .



. . . Ich kann aber beweisen, dass das alle anderen auch nicht können !

Das Credo: $\mathrm{P} \subsetneq \mathrm{NP}$

Folgerung: Für NP-harte Probleme “glaubt man” nicht an Polynomzeitalgorithmen zu ihrer Lösung.

“Ergo” (?) Exponentialzeitalgorithmen sind unvermeidlich für exakte Lösungen NP-harter Probleme.

Ähnliche Zusammenhänge und Folgerungen sind in anderen Bereichen der KT möglich und üblich.

Nochmals Hinweise zum Master-Programm:
Vorlesungen zu Approximationsalgorithmen / Parameterisierten Algorithmen.
Klassischer Theorie-Kanon ebenfalls möglich:
Vorlesungen zu Formalen Sprachen und Komplexitätstheorie.
Sinnvoll ergänzend: Vorlesungen zu Lernalgorithmen und Rekursions/Lerntheorie