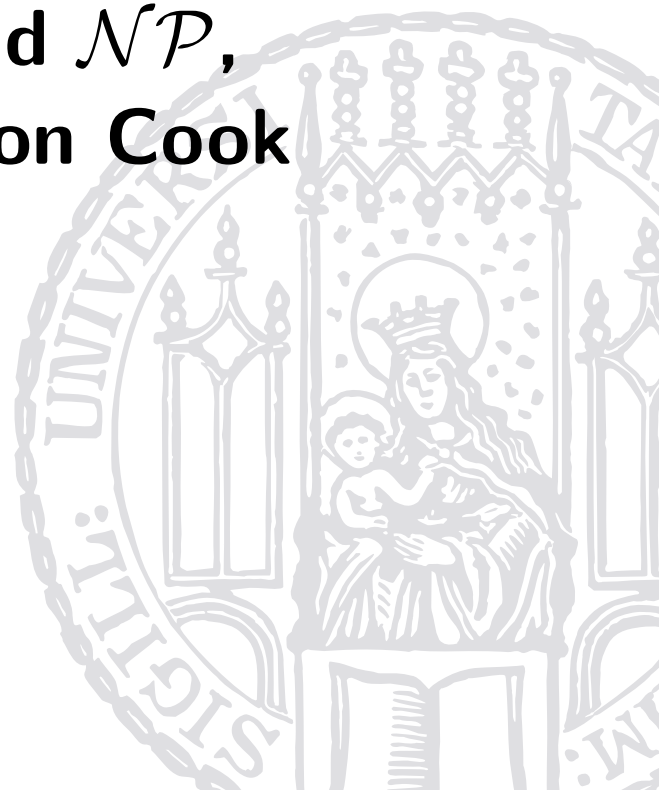
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Formale Sprachen und Komplexität
Sommersemester 2019

Komplexitätstheorie, Teil I: Laufzeitkomplexität, $\mathcal{P}$ und $\mathcal{NP}$, $\mathcal{NP}$-Vollständigkeit, Satz von Cook

Prof. Dr. David Sabel

LFE Theoretische Informatik



Letzte Änderung der Folien: 17. Juli 2019

Inhalt

- Einleitung: Was ist Komplexitätstheorie?
- Zeitkomplexität
- Komplexitätsklassen $\mathcal{P}$ und $\mathcal{NP}$
- Das $\mathcal{P}$-vs-$\mathcal{NP}$-Problem
- Polynomielle Reduktionen und $\mathcal{NP}$-Vollständigkeit
- Satz von Cook: $\mathcal{NP}$-Vollständigkeit von SAT

Komplexitätstheorie

- Teilgebiet der Theoretischen Informatik
- Grobes Thema: Komplexität von **entscheidbaren Problemen**
- Maße zum Messen des Ressourcenbedarfs von Algorithmen
  - Rechenzeit
  - Platzbedarf
  - ...
- Komplexität eines Problems = Komplexität des besten Algorithmus bezüglich des Maßes
- Ziel: Einordnung von Problemen in Komplexitätsklassen

Komplexitätstheorie: Teilbereiche

1. Nachweis von oberen Schranken:
   - Finde möglichst guten Algorithmus für konkretes Problem
   - Analysieren der Laufzeit- und Platzkomplexität
   - z.B. CYK-Algorithmus liefert obere Schranke $O(|P| \cdot n^3)$ für das Wortproblem für CFGs (mit $n =$ Wortlänge, P Produktionen der CFG)

Komplexitätstheorie: Teilbereiche

1. Nachweis von oberen Schranken:
   - Finde möglichst guten Algorithmus für konkretes Problem
   - Analysieren der Laufzeit- und Platzkomplexität
   - z.B. CYK-Algorithmus liefert obere Schranke $O(|P| \cdot n^3)$ für das Wortproblem für CFGs (mit $n =$ Wortlänge, P Produktionen der CFG)
2. Nachweis von unteren Schranken:
   - Zeige, dass es keinen besseren Algorithmus gibt.
   - Schwieriger, da man über alle Algorithmen argumentieren muss.
   - Oft ist $\Omega(n)$ ($n =$ Größe der Eingabe) eine untere Schranke für die Laufzeit, da man die Eingabe lesen muss.

Komplexitätstheorie: Teilbereiche

1. Nachweis von oberen Schranken:
   - Finde möglichst guten Algorithmus für konkretes Problem
   - Analysieren der Laufzeit- und Platzkomplexität
   - z.B. CYK-Algorithmus liefert obere Schranke $O(|P| \cdot n^3)$ für das Wortproblem für CFGs (mit $n =$ Wortlänge, P Produktionen der CFG)
2. Nachweis von unteren Schranken:
   - Zeige, dass es keinen besseren Algorithmus gibt.
   - Schwieriger, da man über alle Algorithmen argumentieren muss.
   - Oft ist $\Omega(n)$ ($n =$ Größe der Eingabe) eine untere Schranke für die Laufzeit, da man die Eingabe lesen muss.
3. Auswirkung der **Maschinenmodelle auf den Ressourcenbedarf**, insbesondere **Determinismus vs. Nichtdeterminismus**
   - Wichtige ungelöste Frage: Das $\mathcal{P}$ **vs.** $\mathcal{NP}$**-Problem**
   - **Wir beschäftigen uns in diesem Abschnitt hiermit.**

Zeitkomplexität: Annahmen und Festlegungen

- Wir betrachten nur entscheidbare Sprachen

Zeitkomplexität: Annahmen und Festlegungen

- Wir betrachten nur entscheidbare Sprachen
- Deswegen keine Einschränkung: TMs, die auf jeder Eingabe anhalten:
  - DTMs: Wir nehmen an, dass es „Verwerfe“-Zustände gibt, für die die TM keine Nachfolgekonfiguration besitzen.
  - NTMs: Haben solche Zustände von Haus aus

Zeitkomplexität: Annahmen und Festlegungen

- Wir betrachten nur entscheidbare Sprachen
- Deswegen keine Einschränkung: TMs, die auf jeder Eingabe anhalten:
  - DTMs: Wir nehmen an, dass es „Verwerfe“-Zustände gibt, für die die TM keine Nachfolgekonfiguration besitzen.
  - NTMs: Haben solche Zustände von Haus aus
- Mehrband- vs. Einband-TMs:
  - Wir nehmen Mehrband-TMs, passen zu „normalen“ Rechnern.
  - Kein „Vergeuden“ von Rechenzeit, nur um die Eingaben zu suchen.
  - Unterschied: n-Schritte auf Mehrband-TM können in $O(n^2)$ Schritten auf Einband-TM ausgeführt werden
  - Unterschied macht sich in Komplexitätsklassen $\mathcal{P}$ und $\mathcal{NP}$ nicht bemerkbar (s. später)

Deterministische Laufzeit

Definition ($time_M$)

Sei M eine stets anhaltende DTM mit Startzustand z_0.
Für Eingabe w definieren wir

$time_M(w) := i$, wenn $z_0 w \vdash_M^i uzw$ und
z ist Endzustand oder Verwerfe-Zustand.

Deterministische Laufzeit

Definition ($time_M$)

Sei M eine stets anhaltende DTM mit Startzustand z_0.
Für Eingabe w definieren wir

$time_M(w) := i$, wenn $z_0 w \vdash_M^i uzw$ und
z ist Endzustand oder Verwerfe-Zustand.

Definition (Klasse $TIME(f(n))$)

Für eine Funktion $f : \mathbb{N} \to \mathbb{N}$ sei die Klasse $TIME(f(n))$ genau die Menge der Sprachen, für die es eine deterministische, stets anhaltende, Mehrband-TM M gibt, mit $L(M) = L$ und $time_M(w) \leq f(|w|)$ für alle $w \in \Sigma^*$

Deterministische Laufzeit

Definition ($time_M$)

Sei M eine stets anhaltende DTM mit Startzustand z_0.
Für Eingabe w definieren wir

$time_M(w) := i$, wenn $z_0 w \vdash_M^i uzw$ und
z ist Endzustand oder Verwerfe-Zustand.

Definition (Klasse $TIME(f(n))$)

Für eine Funktion $f : \mathbb{N} \to \mathbb{N}$ sei die Klasse $TIME(f(n))$ genau die Menge der Sprachen, für die es eine deterministische, stets anhaltende, Mehrband-TM M gibt, mit $L(M) = L$ und $time_M(w) \leq f(|w|)$ für alle $w \in \Sigma^*$

Sprache ist daher in $TIME(f(n))$, wenn sie von einer DTM für jede Eingabe der Länge n in $\leq f(n)$ Schritten entschieden wird.

Die Klasse $\mathcal{P}$

Definition (Polynom)

Ein Polynom ist eine Funktion $p : \mathbb{N} \to \mathbb{N}$ der Form:

$$p(n) = \sum_{i=0}^{k} a_i \cdot n^i = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$$

mit $a_i \in \mathbb{N}$ und $k \in \mathbb{N}$.

Definition (Komplexitätsklasse $\mathcal{P}$)

Die Klasse $\mathcal{P}$ ist definiert als

$$\mathcal{P} = \bigcup_{p \text{ Polynom}} TIME(p(n))$$

Nochmal: Zugehörigkeit zu $\mathcal{P}$

Formale Sprache L ist in Klasse $\mathcal{P}$ enthalten, wenn:

Es gibt stets anhaltende DTM M und ein Polynom p mit:

- $L = L(M)$
- $\forall w \in \Sigma^* : time_M(w) \leq p(|w|)$

Polynomielle Komplexität

Nachweis der polynomiellen Komplexität:

Zeige, dass die Komplexität $O(n^k)$ für Konstante k ist.

Polynomielle Komplexität

Nachweis der polynomiellen Komplexität:

Zeige, dass die Komplexität $O(n^k)$ für Konstante k ist.

Beispiele: Algorithmus mit Laufzeit

- $n \log n$ hat polynomielle Komplexität, da $n \log n \in O(n^2)$
- 2^n hat keine polynomielle Komplexität
- $n^{\log n}$ hat keine polynomielle Komplexität

Polynomielle Komplexität

Nachweis der polynomiellen Komplexität:

Zeige, dass die Komplexität $O(n^k)$ für Konstante k ist.

Beispiele: Algorithmus mit Laufzeit

- $n \log n$ hat polynomielle Komplexität, da $n \log n \in O(n^2)$
- 2^n hat keine polynomielle Komplexität
- $n^{\log n}$ hat keine polynomielle Komplexität

Sprechweisen:

- Algorithmus ist effizient $=$ Algorithmus hat polynomielle Komplexität
- Analog: Problem ist effizient lösbar $=$ Problem in $\mathcal{P}$

Polynomielle Komplexität: Andere Modelle

Sei M eine DTM mit $time_M(w) \leq f(|w|)$.
Analyse unserer Äquivalenzbeweise zeigt:

- TMs können durch GOTO-Programme simuliert werden, konstant viele Schritte, um einen TM-Schritt zu simulieren
- GOTO in WHILE: Anzahl der Durchläufe durch die einzige WHILE-Schleife nicht größer als $f(|w|)$
- WHILE evtl. sogar durch LOOP:

  $y := f(|w|);$ **LOOP** y **DO** $\ldots$ **END**,

  wenn f LOOP-berechenbar

Polynomielle Komplexität: Andere Modelle

Sei M eine DTM mit $time_M(w) \leq f(|w|)$.
Analyse unserer Äquivalenzbeweise zeigt:

- TMs können durch GOTO-Programme simuliert werden, konstant viele Schritte, um einen TM-Schritt zu simulieren
- GOTO in WHILE: Anzahl der Durchläufe durch die einzige WHILE-Schleife nicht größer als $f(|w|)$
- WHILE evtl. sogar durch LOOP:

  $y := f(|w|); \textbf{LOOP } y \textbf{ DO} \ldots \textbf{END}$,

  wenn f LOOP-berechenbar

Wenn f selbst LOOP-berechenbar ist, dann sind auch Funktionen mit Laufzeitkomplexität $f(n)$ LOOP-berechenbar.

Polynomielle Komplexität: Andere Modelle

Sei M eine DTM mit $time_M(w) \leq f(|w|)$.
Analyse unserer Äquivalenzbeweise zeigt:

- TMs können durch GOTO-Programme simuliert werden, konstant viele Schritte, um einen TM-Schritt zu simulieren
- GOTO in WHILE: Anzahl der Durchläufe durch die einzige WHILE-Schleife nicht größer als $f(|w|)$
- WHILE evtl. sogar durch LOOP:

  $y := f(|w|); \textbf{LOOP } y \textbf{ DO} \ldots \textbf{END}$,

  wenn f LOOP-berechenbar

Wenn f selbst LOOP-berechenbar ist, dann sind auch Funktionen mit Laufzeitkomplexität $f(n)$ LOOP-berechenbar.

Daher $TIME(2^n)$ oder sogar $TIME(\underbrace{2^{2^{2^{\cdots^2}}}}_{n\text{-mal}})$ noch in der Klasse der LOOP-berechenbaren Funktionen.

Kostenmaße

Konsequenz der vorherigen Aussagen:

I.a. asymptotisch kein Unterschied, ob Turingmaschine, WHILE-Programm oder GOTO-Programm

Kostenmaße

Konsequenz der vorherigen Aussagen:

I.a. asymptotisch kein Unterschied, ob Turingmaschine, WHILE-Programm oder GOTO-Programm

Vorsicht: Falle!

In $x_i := x_j$ dürfen die Zahlen nicht **zu groß** werden
(bei uniformen Kostenmaß)

Uniformes Kostenmaß:

Ausführung von $x_i := x_j$ in konstanter Zeit (1 Schritt)

Logarithmisches Kostenmaß:

Ausführung von $x_i := x_j$ in Anzahl der Bits ($|\log x_j|$-Schritte)
Turingmaschine muss so viele Bits kopieren!
Daher bei großen Zahlen logarithmisches Kostenmaß verwenden.

Beispiel

WHILE-Programm

$x_0 := 2;$
WHILE $x_1 \neq 0$ **DO**
  $x_0 := x_0 * x_0;$
  $x_1 := x_1 - 1;$
END

berechnet $2^{2^{x_1}}$ in x_0.

- Uniformes Kostenmaß: Laufzeit $O(x_1)$
- Turingmaschine: mindestens 2^{x_1} Schritte alleine um das Ergebnis auf das Band zu schreiben.
- Hier also: Logarithmisches Kostenmaß verwenden!

Unser Vorgehen: logarithmisches Kostenmaß wenn notwendig, sonst uniformes Kostenmaß

Nichtdeterminismus

Auf für NTMs nehmen wir an, dass sie auf allen Berechnungspfaden anhalten.
Beachte: Ein Wort wird nicht akzeptiert, wenn auf allen Berechnungspfaden verworfen wird

Definition ($ntime_M$)

Sei M eine Mehrband-NTM, die für jede Eingabe anhält. Dann definieren wir die Laufzeit von M als

$$ntime_M(w) = \max\{i \mid z_0 w \vdash_M^i uzw \text{ und } uzw \nvdash \ldots\}$$

Beachte: Schöning verwendet eine andere Definition (mit Minimum), es macht für die Definition der Klasse $\mathcal{NP}$ aber keinen Unterschied.

$NTIME$ und $\mathcal{NP}$

Definition

Für eine Funktion $f : \mathbb{N} \to \mathbb{N}$ bezeichne $NTIME(f(n))$ die Klasse aller Sprachen L, für die es eine nichtdeterministische Mehrband-TM M gibt mit $L(M) = L$ und für alle $w \in \Sigma^*$ gilt $ntime_M(w) \leq f(|w|)$.

Definition

Die Klasse $\mathcal{NP}$ ist definiert als

$$\mathcal{NP} = \bigcup_{p \text{ Polynom}} NTIME(p(n))$$

$\mathcal{P} \subseteq \mathcal{NP}$

Lemma

Es gilt $TIME(f(n)) \subseteq NTIME(f(n))$ und damit auch $\mathcal{P} \subseteq \mathcal{NP}$.

Beweis:

- DTM als NTM auffassen
- ergibt NTM mit einem möglichen Berechnungspfad.
- $time_M(w) = ntime_M(w)$
- $L \in TIME(f(n)) \implies L \in NTIME(f(n))$
- $\mathcal{P} \subseteq \mathcal{NP}$

$\mathcal{P}$-vs.-$\mathcal{NP}$

Die Frage $\mathcal{P} = \mathcal{NP}$ oder $\mathcal{P} \neq \mathcal{NP}$ ist ungelöst!
Das $\mathcal{P}$-vs-$\mathcal{NP}$-Problem ist eines der sieben sogenannten Millennium-Probleme, die vom Clay Mathematics Institute im Jahr 2000 als Liste ungelöster Probleme der Mathematik herausgegeben wurde und für dessen Lösung ein Preisgeld von einer Million US Dollar ausgelobt wurde.

$\mathcal{P}$-vs.-$\mathcal{NP}$ (2)

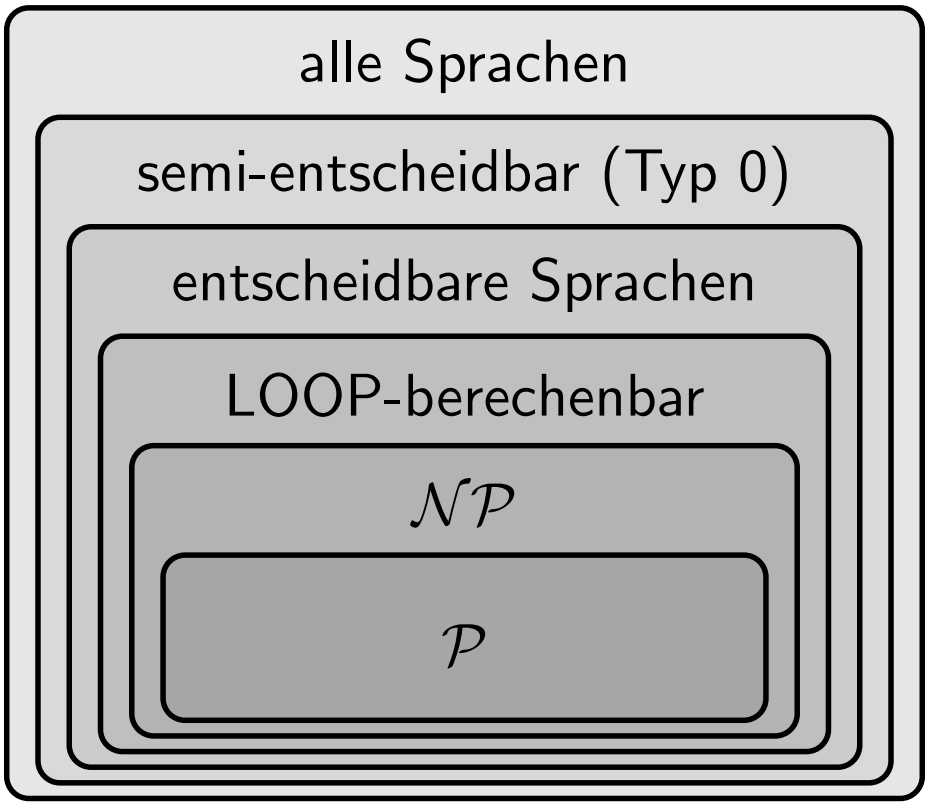
Wesentlicher Grund, der für $\mathcal{P} \neq \mathcal{NP}$ spricht:

> Man müsste für $\mathcal{P} = \mathcal{NP}$ einen Polynomialzeitalgorithmus finden, für ein Problem, für das bisher nur (deterministische) Exponentialzeitalgorithmen bekannt sind. (*)

Viele haben gesucht, keiner hat einen solchen Algorithmus gefunden.

(*) Begründung: Folgt später

Lage der Komplexitätsklasse (Schema)



Dabei unklar, ob $\mathcal{NP} = \mathcal{P}$

Polynomialzeit-Reduktion

Definition (Polynomialzeit-Reduktion (einer Sprache auf eine andere))

Sei $L_1 \subseteq \Sigma_1^*$ und $L_2 \subseteq \Sigma_2^*$ Sprachen. Dann sagen wir L_1 ist auf L_2 polynomiell reduzierbar (geschrieben $L_1 \leq_p L_2$), falls es eine totale und in deterministischer Polynomialzeit berechenbare Funktion $f : \Sigma_1^* \to \Sigma_2^*$ gibt, sodass für alle $w \in \Sigma_1^*$ gilt: $w \in L_1 \iff f(w) \in L_2$.

Polynomialzeit-Reduktion

Definition (Polynomialzeit-Reduktion (einer Sprache auf eine andere))

Sei $L_1 \subseteq \Sigma_1^*$ und $L_2 \subseteq \Sigma_2^*$ Sprachen. Dann sagen wir L_1 ist auf L_2 polynomiell reduzierbar (geschrieben $L_1 \leq_p L_2$), falls es eine totale und in deterministischer Polynomialzeit berechenbare Funktion $f : \Sigma_1^* \to \Sigma_2^*$ gibt, sodass für alle $w \in \Sigma_1^*$ gilt: $w \in L_1 \iff f(w) \in L_2$.

Analogie zu Reduktionen in der Berechenbarkeitstheorie $L_1 \leq L_2$:
Zusatz hier: Polynomialzeit!

Polynomialzeit-Reduktion

Definition (Polynomialzeit-Reduktion (einer Sprache auf eine andere))

Sei $L_1 \subseteq \Sigma_1^*$ und $L_2 \subseteq \Sigma_2^*$ Sprachen. Dann sagen wir L_1 ist auf L_2 polynomiell reduzierbar (geschrieben $L_1 \leq_p L_2$), falls es eine totale und in deterministischer Polynomialzeit berechenbare Funktion $f : \Sigma_1^* \to \Sigma_2^*$ gibt, sodass für alle $w \in \Sigma_1^*$ gilt: $w \in L_1 \iff f(w) \in L_2$.

Analogie zu Reduktionen in der Berechenbarkeitstheorie $L_1 \leq L_2$:
Zusatz hier: Polynomialzeit!

Nächste Analogie:

Berechenbarkeitstheorie:	Komplexitätstheorie:
$L_1 \leq L_2$ und L_2 (semi-) entscheidbar	$L_1 \leq_p L_2$ und $L_2 \in (\mathcal{N})\mathcal{P}$
$\implies$ L_1 (semi-) entscheidbar	$\implies$ $L_1 \in (\mathcal{N})\mathcal{P}$

Polynomialzeit-Reduktion (Eigenschaften)

Lemma

Falls $L_1 \leq_p L_2$ und $L_2 \in \mathcal{P}$, dann gilt $L_1 \in \mathcal{P}$. Ebenso gilt: Falls $L_1 \leq_p L_2$ und $L_2 \in \mathcal{NP}$, dann gilt $L_1 \in \mathcal{NP}$.

Beweis:

- Sei $L_1 \leq_p L_2$ und f in Polynomialzeit berechenbar
- Sei M_f die DTM, die f in Polynomialzeit durch p beschr. berechnet.
- Sei $L_2 \in \mathcal{P}$, $L(M_2) = L_2$, wobei Schritte von $M_2 \leq q(|w|)$
- $L(M_f; M_2) = L_1$, $M_f; M_2$ hält stets in Polynomialzeit:
  $|f(w)| \leq |w| + p(|w|)$
  (da M_f nicht mehr in $p(|w|)$-Schritten schreiben kann)
  $M_f; M_2$ höchstens $r(|w|) := p(|w|) + q(|w| + p(|w|))$ Schritte
- Es gilt $L_1 \in \mathcal{P}$.

Für $\mathcal{NP}$ analog.

Polynomialzeit-Reduktion (Eigenschaften) (2)

Lemma

Die Relation $\leq_p$ ist transitiv, d.h. wenn $L_1 \leq_p L_2$ und $L_2 \leq_p L_3$, dann gilt auch $L_1 \leq_p L_3$

Analog zum vorherigen Beweis:
Komposition von zwei Polynomen bleibt Polynom.

$\mathcal{NP}$-Vollständigkeit

Definition ($\mathcal{NP}$-Vollständigkeit)

Eine Sprache L_0 heißt $\mathcal{NP}$-vollständig, wenn gilt

1. $L_0 \in \mathcal{NP}$ und
2. L_0 ist $\mathcal{NP}$-hart (manchmal auch $\mathcal{NP}$-schwer genannt): Für alle $L \in \mathcal{NP}$ gilt $L \leq_p L_0$

$\mathcal{NP}$-vollständige Probleme sind die schwierigsten Probleme in $\mathcal{NP}$:

$\mathcal{NP}$-Härte besagt, dass man mit dem $\mathcal{NP}$-vollständigen Problem alle anderen Probleme aus $\mathcal{NP}$ (in zusätzlicher deterministischer Polynomialzeit) lösen kann

$\mathcal{NP}$-Vollständigkeit beweisen

Nachweis der $\mathcal{NP}$-Vollständigkeit von L

- Zugehörigkeit zu $\mathcal{NP}$: Gebe polynomiell Laufzeit-beschränkte NTM an, die L entscheidet
- $\mathcal{NP}$-Härte: Statt jedes mal neu zu beweisen, dass alle Probleme aus $\mathcal{NP}$ auf L polynomiell reduzierbar, wähle ein $\mathcal{NP}$-hartes Problem L_0 und zeige $L_0 \leq_p L$.

  Dann folgt L ist $\mathcal{NP}$-hart:

  Da L_0 $\mathcal{NP}$-hart gilt $L' \leq_p L_0$ für alle $L \in \mathcal{NP}$ und damit $L' \leq_p L_0 \leq_p L$ und mit Transitivität von $\leq_p$:
  $L' \leq_p L$ für alle $L' \in \mathcal{NP}$.

Komplett analog zum Vorgehen wie bei der Unentscheidbarkeit, wesentlicher Unterschied: Polynomialzeit-Reduktion

Satz

Sei L ein $\mathcal{NP}$-vollständiges Problem. Dann gilt
$L \in \mathcal{P} \iff \mathcal{P} = \mathcal{NP}$.

Satz

Sei L ein $\mathcal{NP}$-vollständiges Problem. Dann gilt
$L \in \mathcal{P} \iff \mathcal{P} = \mathcal{NP}$.

Beweis:

- Sei L $\mathcal{NP}$-vollständig und $L \in \mathcal{P}$
- Aus $\mathcal{NP}$-Härte von L folgt:
  Für alle $L' \in \mathcal{NP}$: $L' \leq_p L$ und damit $L' \in \mathcal{P}$.
- Da dies für alle $L' \in \mathcal{NP}$ gilt, folgt $\mathcal{P} = \mathcal{NP}$.

Satz

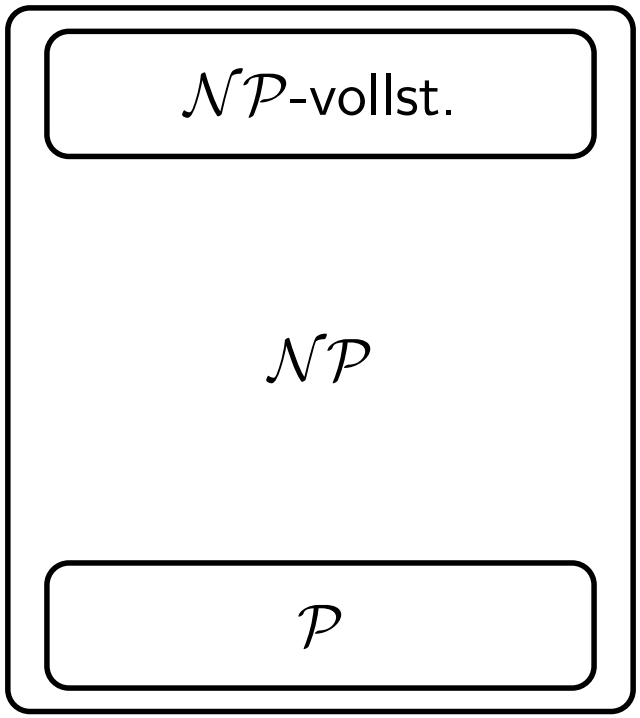
Sei L ein $\mathcal{NP}$-vollständiges Problem. Dann gilt
$L \in \mathcal{P} \iff \mathcal{P} = \mathcal{NP}$.

Beweis:

- Sei L $\mathcal{NP}$-vollständig und $L \in \mathcal{P}$
- Aus $\mathcal{NP}$-Härte von L folgt:
  Für alle $L' \in \mathcal{NP}$: $L' \leq_p L$ und damit $L' \in \mathcal{P}$.
- Da dies für alle $L' \in \mathcal{NP}$ gilt, folgt $\mathcal{P} = \mathcal{NP}$.

Also: Es reicht aus für ein $\mathcal{NP}$-vollständiges Problem nachzuweisen, dass es in $\mathcal{P}$ bzw. nicht in $\mathcal{P}$ liegt, um die $\mathcal{P}$-vs-$\mathcal{NP}$-Frage ein für allemal beantworten.

Vermutete Lage der Probleme



Es ist bekannt (Ladner 1975):

Unter der Annahme $\mathcal{P} \neq \mathcal{NP}$ gibt es Probleme in $\mathcal{NP}$ gibt, die nicht in $\mathcal{P}$ liegen und nicht $\mathcal{NP}$-vollständig sind.

Möglicher Kandidat:

Graph-Isomorphismus-Problem. Weder ein polynomieller Algorithmus noch dessen $\mathcal{NP}$-Vollständigkeit bekannt.

SAT-Problem

- Wir brauchen ein erstes Problem, dessen $\mathcal{NP}$-Vollständigkeit wir per Hand nachweisen müssen.
- Dafür nehmen wir das SAT-Problem.

Definition (SAT-Problem)

Das Erfüllbarkeitsproblem der Aussagenlogik (kurz SAT) ist:

gegeben: Eine Aussagenlogische Formel F
gefragt: Ist F erfüllbar, d.h. gibt es eine erfüllende Belegung der Variablen mit den Wahrheitswerten 0 und 1, sodass F den Wert 1 erhält.

Als formale Sprache:

$$\text{SAT} = \{code(F) \in \Sigma^* \mid F \text{ ist erfüllbare Formel der Aussagenlogik}\}$$

SAT $\in \mathcal{NP}$

Lemma

SAT $\in \mathcal{NP}$

Beweis:

- $code(F)$ Eingabe einer NTM M
- M berechnet, welche Variablen in F vorkommen
- Sei dies $\{x_1, \ldots, x_n\}$.

SAT $\in \mathcal{NP}$

Lemma

SAT $\in \mathcal{NP}$

Beweis:

- $code(F)$ Eingabe einer NTM M
- M berechnet, welche Variablen in F vorkommen
- Sei dies $\{x_1, \ldots, x_n\}$.
- M verwendet Nichtdeterminismus, um Belegung $I : \{x_1, \ldots, x_n\} \to \{0, 1\}$ zu „raten“

SAT $\in \mathcal{NP}$

Lemma

SAT $\in \mathcal{NP}$

Beweis:

- $code(F)$ Eingabe einer NTM M
- M berechnet, welche Variablen in F vorkommen
- Sei dies $\{x_1, \ldots, x_n\}$.
- M verwendet Nichtdeterminismus, um Belegung $I : \{x_1, \ldots, x_n\} \to \{0, 1\}$ zu „raten“
- Jede der 2^n n.d. Berechnungen berechnet Wert von $I(F)$

SAT $\in \mathcal{NP}$

Lemma

SAT $\in \mathcal{NP}$

Beweis:

- $code(F)$ Eingabe einer NTM M
- M berechnet, welche Variablen in F vorkommen
- Sei dies $\{x_1, \ldots, x_n\}$.
- M verwendet Nichtdeterminismus, um Belegung $I : \{x_1, \ldots, x_n\} \to \{0, 1\}$ zu „raten“
- Jede der 2^n n.d. Berechnungen berechnet Wert von $I(F)$
- Akzeptanz bei 1, sonst verwerfen.

SAT $\in \mathcal{NP}$

Lemma

SAT $\in \mathcal{NP}$

Beweis:

- $code(F)$ Eingabe einer NTM M
- M berechnet, welche Variablen in F vorkommen
- Sei dies $\{x_1, \ldots, x_n\}$.
- M verwendet Nichtdeterminismus, um Belegung $I : \{x_1, \ldots, x_n\} \to \{0, 1\}$ zu „raten“
- Jede der 2^n n.d. Berechnungen berechnet Wert von $I(F)$
- Akzeptanz bei 1, sonst verwerfen.
- Da jede Belegung überprüft wird, gilt M akzeptiert eine Formel F g.d.w. $F \in$ SAT.

SAT $\in \mathcal{NP}$

Lemma

SAT $\in \mathcal{NP}$

Beweis:

- $code(F)$ Eingabe einer NTM M
- M berechnet, welche Variablen in F vorkommen
- Sei dies $\{x_1, \ldots, x_n\}$.
- M verwendet Nichtdeterminismus, um Belegung $I : \{x_1, \ldots, x_n\} \to \{0, 1\}$ zu „raten“
- Jede der 2^n n.d. Berechnungen berechnet Wert von $I(F)$
- Akzeptanz bei 1, sonst verwerfen.
- Da jede Belegung überprüft wird, gilt M akzeptiert eine Formel F g.d.w. $F \in$ SAT.
- Jeder Berechnungspfad von M läuft in Polynomialzeit in $|code(F)|$, da die Anzahl der Variablen durch die Eingabegröße beschränkt ist.

$\mathcal{NP}$ = Polynomiell verifizierbar

- Nachweis, dass Sprache in $\mathcal{NP}$-liegt geht oft so wie bei SAT
- Verwende Nichtdeterminismus um potentielle Lösung zu raten
- Zeige, dass eine Lösung in Polynomialzeit verifiziert werden kann.

Hilfssatz für den $\mathcal{NP}$-Härte Beweis von SAT

Lemma

Für aussagenlogische Variablen $\{x_1, \ldots, x_n\}$ gibt es eine aussagenlogische Formel $exactlyOne(x_1, \ldots, x_n)$ sodass $I(exactlyOne(x_1, \ldots, x_n)) = 1$ g.d.w. I genau eine der Variablen x_i auf 1 setzt und alle anderen auf 0.
Dabei ist die Größe der Formel $exactlyOne(x_1, \ldots, x_n)$ in $O(n^2)$.

Hilfssatz für den $\mathcal{NP}$-Härte Beweis von SAT

Lemma

Für aussagenlogische Variablen $\{x_1, \ldots, x_n\}$ gibt es eine aussagenlogische Formel $exactlyOne(x_1, \ldots, x_n)$ sodass $I(exactlyOne(x_1, \ldots, x_n)) = 1$ g.d.w. I genau eine der Variablen x_i auf 1 setzt und alle anderen auf 0.
Dabei ist die Größe der Formel $exactlyOne(x_1, \ldots, x_n)$ in $O(n^2)$.

Beweis:

$$exactlyOne(x_1, \ldots, x_n) := (x_1 \vee \ldots \vee x_n) \wedge \bigwedge_{1 \leq i < j \leq n} \neg(x_i \wedge x_j)$$

- $(x_1 \vee \ldots \vee x_n)$ sichert $atLeastOne(x_1, \ldots, x_n)$ zu
- $\bigwedge_{1 \leq i < j \leq n} \neg(x_i \wedge x_j)$ sichert $atMostOne(x_1, \ldots, x_n)$ zu
- Größe $O(n^2)$

$\mathcal{NP}$-Härte von SAT: Ideen

- Wir müssen zeigen:

$$L \leq_p \text{SAT für alle } L \in \mathcal{NP}$$

$\mathcal{NP}$-Härte von SAT: Ideen

- Wir müssen zeigen:

$$L \leq_p \text{SAT für alle } L \in \mathcal{NP}$$

- Da $L \in \mathcal{NP}$, gibt es polynomiell zeitbeschränkte NTM M, die L akzeptiert.

$\mathcal{NP}$-Härte von SAT: Ideen

- Wir müssen zeigen:

$$L \leq_p \text{SAT für alle } L \in \mathcal{NP}$$

- Da $L \in \mathcal{NP}$, gibt es polynomiell zeitbeschränkte NTM M, die L akzeptiert.
- Erstelle Formel F, sodass F erfüllbar g.d.w. M akzeptiert

$\mathcal{NP}$-Härte von SAT: Ideen

- Wir müssen zeigen:

$$L \leq_p \text{SAT für alle } L \in \mathcal{NP}$$

- Da $L \in \mathcal{NP}$, gibt es polynomiell zeitbeschränkte NTM M, die L akzeptiert.
- Erstelle Formel F, sodass F erfüllbar g.d.w. M akzeptiert
- Da Laufzeit polynomiell beschränkt, beschränkt dies auch die Größe der Formel und wir haben $\leq_p$

SAT ist $\mathcal{NP}$-hart (1)

Lemma

SAT ist $\mathcal{NP}$-hart.

Beweis: Wir müssen zeigen: $L \leq_p$ SAT

- Sei $L \in \mathcal{NP}$ beliebig.
- Sei M die NTM mit $L(M) = L$ und $ntime_M(w) \leq p(|w|)$.
- Sei w eine Eingabe für M.

Ziel: Konstruiere aussagenlogische Formel F, sodass gilt

$$w \in L \iff F \text{ ist erfüllbar}$$

Dabei muss F in Polynomialzeit konstruierbar sein.
D.h. wir geben eine in Polynomialzeit berechenbare Funktion $f(w)$ an, sodass $w \in L \iff f(w) \in$ SAT.

SAT ist $\mathcal{NP}$-hart (2)

Notation (o.B.d.A):	
Eingabe:	$w = a_1 \cdots a_n \in \Sigma^*$
Bandalphabet:	$\Gamma = \{b_1, \ldots, b_l\}$
Zustände:	$Z = \{z_0, \ldots, z_k\}$
Startzustand:	z_0

SAT ist $\mathcal{NP}$-hart (2)

Eingabe:	$w = a_1 \cdots a_n \in \Sigma^*$
Bandalphabet:	$\Gamma = \{b_1, \ldots, b_l\}$
Zustände:	$Z = \{z_0, \ldots, z_k\}$
Startzustand:	z_0

SAT ist $\mathcal{NP}$-hart (2)

Eingabe:	$w = a_1 \cdots a_n \in \Sigma^*$
Bandalphabet:	$\Gamma = \{b_1, \ldots, b_l\}$
Zustände:	$Z = \{z_0, \ldots, z_k\}$
Startzustand:	z_0

Aussagenlogische Variablen in der Formel F:

Variable	Index-Bereich	Bedeutung
$State_{t,z}$	$t = 0, 1, \ldots, p(n)$ $z = z_0, \ldots, z_k$	$State_{t,z} = 1$ g.d.w. nach t Schritten ist M im Zustand z.
$Pos_{t,i}$	$t = 0, 1, \ldots, p(n)$ $i = -p(n), \ldots, p(n)$	$Pos_{t,z} = 1$ g.d.w. nach t Schritten ist der Schreib-Lesekopf auf Position i
$Tape_{t,i,b}$	$t = 0, 1, \ldots, p(n)$ $i = -p(n), \ldots, p(n)$ $b = b_1, \ldots, b_l$	$Tape_{t,i,b} = 1$ g.d.w. nach t Schritten steht in Position i das Zeichen b

SAT ist $\mathcal{NP}$-hart (2)

Eingabe:	$w = a_1 \cdots a_n \in \Sigma^*$
Bandalphabet:	$\Gamma = \{b_1, \ldots, b_l\}$
Zustände:	$Z = \{z_0, \ldots, z_k\}$
Startzustand:	z_0

Aussagenlogische Variablen in der Formel F:

Variable	Index-Bereich	Bedeutung
$State_{t,z}$	$t = 0, 1, \ldots, p(n)$ $z = z_0, \ldots, z_k$	$State_{t,z} = 1$ g.d.w. nach t Schritten ist M im Zustand z.
$Pos_{t,i}$	$t = 0, 1, \ldots, p(n)$ $i = -p(n), \ldots, p(n)$	$Pos_{t,z} = 1$ g.d.w. nach t Schritten ist der Schreib-Lesekopf auf Position i
$Tape_{t,i,b}$	$t = 0, 1, \ldots, p(n)$ $i = -p(n), \ldots, p(n)$ $b = b_1, \ldots, b_l$	$Tape_{t,i,b} = 1$ g.d.w. nach t Schritten steht in Position i das Zeichen b

- Bandpositionen: Position 0 am Anfang
- Bereiche reichen aus, da die TM nicht mehr als $p(n)$ Schritte macht

SAT ist $\mathcal{NP}$-hart (3)

Aufbau der Formel F:

$$F = Rand \wedge Anfang \wedge Transition \wedge Ende$$

- $Rand$: Randbedingungen
- $Anfang$: Anfangsbedingungen
- $Transition$: Bedingungen für den Zustandsübergang
- $Ende$: Endbedingung

SAT ist $\mathcal{NP}$-hart (4)

Randbedingungen:

Zu jedem Zeitpunkt t:

- ... ist M in genau einem Zustand z:

SAT ist $\mathcal{NP}$-hart (4)

Randbedingungen:

Zu jedem Zeitpunkt t:

- ... ist M in genau einem Zustand z:

$$\bigwedge_{t \in \{0,\ldots,p(n)\}} exactlyOne(State_{t,z_0}, \ldots, State_{t,z_k})$$

SAT ist $\mathcal{NP}$-hart (4)

Randbedingungen:

Zu jedem Zeitpunkt t:

- ... ist M in genau einem Zustand z:

$$\bigwedge_{t \in \{0,\ldots,p(n)\}} exactlyOne(State_{t,z_0}, \ldots, State_{t,z_k})$$

- ... ist der Kopf von M in genau einer Position auf dem Band:

SAT ist $\mathcal{NP}$-hart (4)

Randbedingungen:

Zu jedem Zeitpunkt t:

- ... ist M in genau einem Zustand z:

$$\bigwedge_{t\in\{0,\ldots,p(n)\}} exactlyOne(State_{t,z_0},\ldots,State_{t,z_k})$$

- ... ist der Kopf von M in genau einer Position auf dem Band:

$$\bigwedge_{t\in\{0,\ldots,p(n)\}} exactlyOne(Pos_{t,-p(n)},\ldots,Pos_{t,p(n)})$$

SAT ist $\mathcal{NP}$-hart (4)

Randbedingungen:

Zu jedem Zeitpunkt t:

- ... ist M in genau einem Zustand z:

$$\bigwedge_{t\in\{0,\ldots,p(n)\}} exactlyOne(State_{t,z_0},\ldots,State_{t,z_k})$$

- ... ist der Kopf von M in genau einer Position auf dem Band:

$$\bigwedge_{t\in\{0,\ldots,p(n)\}} exactlyOne(Pos_{t,-p(n)},\ldots,Pos_{t,p(n)})$$

- ... befindet sich in jeder Bandzelle genau ein Symbol aus Γ:

SAT ist $\mathcal{NP}$-hart (4)

Randbedingungen:

Zu jedem Zeitpunkt t:

- ... ist M in genau einem Zustand z:

$$\bigwedge_{t\in\{0,\ldots,p(n)\}} exactlyOne(State_{t,z_0},\ldots,State_{t,z_k})$$

- ... ist der Kopf von M in genau einer Position auf dem Band:

$$\bigwedge_{t\in\{0,\ldots,p(n)\}} exactlyOne(Pos_{t,-p(n)},\ldots,Pos_{t,p(n)})$$

- ... befindet sich in jeder Bandzelle genau ein Symbol aus Γ:

$$\bigwedge_{t\in\{0,\ldots,p(n)\}} \bigwedge_{i\in\{-p(n),\ldots,p(n)\}} exactlyOne(Tape_{t,i,b_1},\ldots,Tape_{t,i,b_l})$$

SAT ist $\mathcal{NP}$-hart (4)

Randbedingungen:

Zu jedem Zeitpunkt t:

- ... ist M in genau einem Zustand z:

$$\bigwedge_{t\in\{0,\ldots,p(n)\}} exactlyOne(State_{t,z_0},\ldots,State_{t,z_k})$$

- ... ist der Kopf von M in genau einer Position auf dem Band:

$$\bigwedge_{t\in\{0,\ldots,p(n)\}} exactlyOne(Pos_{t,-p(n)},\ldots,Pos_{t,p(n)})$$

- ... befindet sich in jeder Bandzelle genau ein Symbol aus Γ:

$$\bigwedge_{t\in\{0,\ldots,p(n)\}} \bigwedge_{i\in\{-p(n),\ldots,p(n)\}} exactlyOne(Tape_{t,i,b_1},\ldots,Tape_{t,i,b_l})$$

Daher:

$$Rand := \bigwedge_{t\in\{0,\ldots,p(n)\}} \begin{pmatrix} exactlyOne(State_{t,z_0},\ldots,State_{t,z_k}) \\ \wedge exactlyOne(Pos_{t,-p(n)},\ldots,Pos_{t,p(n)}) \\ \wedge \bigwedge_{i\in\{-p(n),\ldots,p(n)\}} exactlyOne(Tape_{t,i,b_1},\ldots,Tape_{t,i,b_l}) \end{pmatrix}$$

SAT ist $\mathcal{NP}$-hart (5)

Anfangsbedingungen:

Fixieren die Bedingungen zum Zeitpunkt $t = 0$:

SAT ist $\mathcal{NP}$-hart (5)

Anfangsbedingungen:

Fixieren die Bedingungen zum Zeitpunkt $t = 0$:

- M ist im Startzustand:

SAT ist $\mathcal{NP}$-hart (5)

Anfangsbedingungen:

Fixieren die Bedingungen zum Zeitpunkt $t = 0$:

- M ist im Startzustand: $State_{0,z_0}$

SAT ist $\mathcal{NP}$-hart (5)

Anfangsbedingungen:

Fixieren die Bedingungen zum Zeitpunkt $t = 0$:

- M ist im Startzustand: $State_{0,z_0}$
- Der Schreib-Lesekopf ist auf Position 0:

SAT ist $\mathcal{NP}$-hart (5)

Anfangsbedingungen:

Fixieren die Bedingungen zum Zeitpunkt $t = 0$:

- M ist im Startzustand: $State_{0,z_0}$
- Der Schreib-Lesekopf ist auf Position 0: $Pos_{0,0}$

SAT ist $\mathcal{NP}$-hart (5)

Anfangsbedingungen:

Fixieren die Bedingungen zum Zeitpunkt $t = 0$:

- M ist im Startzustand: $State_{0,z_0}$
- Der Schreib-Lesekopf ist auf Position 0: $Pos_{0,0}$
- Die Eingabe $w = a_1 \cdots a_n$ steht auf dem Band, und alle anderen Zellen enthalten das Blank-Symbol.

SAT ist $\mathcal{NP}$-hart (5)

Anfangsbedingungen:

Fixieren die Bedingungen zum Zeitpunkt $t = 0$:

- M ist im Startzustand: $State_{0,z_0}$
- Der Schreib-Lesekopf ist auf Position 0: $Pos_{0,0}$
- Die Eingabe $w = a_1 \cdots a_n$ steht auf dem Band, und alle anderen Zellen enthalten das Blank-Symbol.

$$\Big(\bigwedge_{i\in\{0,\ldots,n-1\}} Tape_{0,i,a_{i+1}}\Big) \wedge \Big(\bigwedge_{i\in\{-p(n),\ldots,-1\}} Tape_{0,i,\square}\Big) \wedge \Big(\bigwedge_{i\in\{n,\ldots,p(n)\}} Tape_{0,i,\square}\Big).$$

SAT ist $\mathcal{NP}$-hart (5)

Anfangsbedingungen:

Fixieren die Bedingungen zum Zeitpunkt $t = 0$:

- M ist im Startzustand: $State_{0,z_0}$
- Der Schreib-Lesekopf ist auf Position 0: $Pos_{0,0}$
- Die Eingabe $w = a_1 \cdots a_n$ steht auf dem Band, und alle anderen Zellen enthalten das Blank-Symbol.

$$\Big(\bigwedge_{i\in\{0,\ldots,n-1\}} Tape_{0,i,a_{i+1}}\Big) \wedge \Big(\bigwedge_{i\in\{-p(n),\ldots,-1\}} Tape_{0,i,\square}\Big) \wedge \Big(\bigwedge_{i\in\{n,\ldots,p(n)\}} Tape_{0,i,\square}\Big).$$

Daher:

$$Anfang := State_{0,z_0} \wedge Pos_{0,0} \wedge \Big(\bigwedge_{i\in\{0,\ldots,n-1\}} Tape_{0,i,a_{i+1}}\Big) \wedge \Big(\bigwedge_{i\in\{-p(n),\ldots,-1\}} Tape_{0,i,\square}\Big) \wedge \Big(\bigwedge_{i\in\{n,\ldots,p(n)\}} Tape_{0,i,\square}\Big)$$

SAT ist $\mathcal{NP}$-hart (6)

Transitionsbedingungen:

SAT ist $\mathcal{NP}$-hart (6)

Transitionsbedingungen:

- Für Übergang von t zu $t+1$: Zustand, Bandinhalt, Position ändern.
  Sei $dir(N) = 0, dir(L) = -1, dir(R) = 1$.

SAT ist $\mathcal{NP}$-hart (6)

Transitionsbedingungen:

- Für Übergang von t zu $t+1$: Zustand, Bandinhalt, Position ändern.
  Sei $dir(N) = 0, dir(L) = -1, dir(R) = 1$.

$$\bigwedge_{\substack{t\in\{0,\ldots,p(n)-1\},\\ z\in Z,\\ i\in\{-p(n)+1,\ldots,p(n)-1\},\\ b\in\Gamma}} \left(\begin{array}{l} (State_{t,z} \wedge Pos_{t,i} \wedge Tape_{t,i,b}) \\ \quad \Longrightarrow \bigvee_{(z',b',y)\in\delta(z,b)} (State_{t+1,z'} \wedge Pos_{t+1,i+dir(y)} \wedge Tape_{t+1,i,b'}) \end{array} \right)$$

SAT ist $\mathcal{NP}$-hart (6)

Transitionsbedingungen:

- Für Übergang von t zu $t+1$: Zustand, Bandinhalt, Position ändern.
  Sei $dir(N) = 0, dir(L) = -1, dir(R) = 1$.

$$\bigwedge_{\substack{t\in\{0,\ldots,p(n)-1\},\\ z\in Z,\\ i\in\{-p(n)+1,\ldots,p(n)-1\},\\ b\in\Gamma}} \left(\begin{array}{l} \left(State_{t,z} \wedge Pos_{t,i} \wedge Tape_{t,i,b}\right) \\ \quad \Longrightarrow \bigvee_{(z',b',y)\in\delta(z,b)} \left(State_{t+1,z'} \wedge Pos_{t+1,i+dir(y)} \wedge Tape_{t+1,i,b'}\right) \end{array} \right)$$

- Zellen auf denen der Kopf nicht steht, bleiben unverändert.

SAT ist $\mathcal{NP}$-hart (6)

Transitionsbedingungen:

- Für Übergang von t zu $t+1$: Zustand, Bandinhalt, Position ändern.
  Sei $dir(N) = 0, dir(L) = -1, dir(R) = 1$.

$$\bigwedge_{\substack{t\in\{0,\ldots,p(n)-1\},\\ z\in Z,\\ i\in\{-p(n)+1,\ldots,p(n)-1\},\\ b\in\Gamma}} \left(\begin{array}{l} (State_{t,z} \wedge Pos_{t,i} \wedge Tape_{t,i,b}) \\ \quad \Longrightarrow \bigvee_{(z',b',y)\in\delta(z,b)} (State_{t+1,z'} \wedge Pos_{t+1,i+dir(y)} \wedge Tape_{t+1,i,b'}) \end{array} \right)$$

- Zellen auf denen der Kopf nicht steht, bleiben unverändert.

$$\bigwedge_{\substack{t\in\{0,\ldots,p(n)-1\},\\ i\in\{-p(n),\ldots,p(n)\},\\ b\in\Gamma}} ((\neg Pos_{t,i} \wedge Tape_{t,i,b}) \implies Tape_{t+1,i,b})$$

SAT ist $\mathcal{NP}$-hart (7)

Transitionsbedingungen:

Ergibt zusammen:

$$
\begin{aligned}
&Transition := \\
&\Bigg(\bigwedge_{\substack{t\in\{0,\ldots,p(n)-1\},\\ z\in Z,\\ i\in\{-p(n)+1,\ldots,p(n)-1\},\\ b\in\Gamma}} \Bigg(\begin{array}{l} (State_{t,z} \wedge Pos_{t,i} \wedge Tape_{t,i,b}) \\ \quad \Longrightarrow \bigvee_{(z',b',y)\in\delta(z,b)} (State_{t+1,z'} \wedge Pos_{t+1,i+dir(y)} \wedge Tape_{t+1,i,b'}) \end{array} \Bigg) \Bigg) \\
&\wedge \Bigg(\bigwedge_{\substack{t\in\{0,\ldots,p(n)-1\},\\ i\in\{-p(n),\ldots,p(n)\},\\ b\in\Gamma}} \Big((\neg Pos_{t,i} \wedge Tape_{t,i,b}) \implies Tape_{t+1,i,b} \Big) \Bigg)
\end{aligned}
$$

SAT ist $\mathcal{NP}$-hart (8)

Endbedingung: Ein akzeptierender Zustand wird erreicht:

SAT ist $\mathcal{NP}$-hart (8)

Endbedingung: Ein akzeptierender Zustand wird erreicht:

$$Ende := \bigvee_{z \in E, t \in \{0,\ldots,p(n)\}} State_{t,z}$$

SAT ist $\mathcal{NP}$-hart (9)

$$F = \bigwedge_{t\in\{0,\ldots,p(n)\}} \left(\begin{array}{l} exactlyOne(State_{t,z_0},\ldots,State_{t,z_k}) \\ \wedge exactlyOne(Pos_{t,-p(n)},\ldots,Pos_{t,p(n)}) \\ \wedge \bigwedge_{i\in\{-p(n),\ldots,p(n)\}} exactlyOne(Tape_{t,i,b_1},\ldots,Tape_{t,i,b_l}) \end{array} \right)$$

$$\wedge State_{0,z_0} \wedge Pos_{0,0} \wedge \left(\bigwedge_{i\in\{0,\ldots,n-1\}} Tape_{0,i,a_{i+1}}\right) \wedge \left(\bigwedge_{i\in\{-p(n),\ldots,-1\}} Tape_{0,i,\square}\right) \wedge \left(\bigwedge_{i\in\{n,\ldots,p(n)\}} Tape_{0,i,\square}\right)$$

$$\wedge \Bigg(\bigwedge_{\substack{t\in\{0,\ldots,p(n)-1\},\\ z\in Z,\\ i\in\{-p(n)+1,\ldots,p(n)-1\},\\ b\in\Gamma}} \Big(\begin{array}{l} (State_{t,z} \wedge Pos_{t,i} \wedge Tape_{t,i,b}) \\ \implies \bigvee_{(z',b',y)\in\delta(z,b)} (State_{t+1,z'} \wedge Pos_{t+1,i+dir(y)} \wedge Tape_{t+1,i,b'}) \end{array} \Big) \Bigg)$$

$$\wedge \Bigg(\bigwedge_{\substack{t\in\{0,\ldots,p(n)-1\},\\ i\in\{-p(n),\ldots,p(n)\},\\ b\in\Gamma}} \big((\neg Pos_{t,i} \wedge Tape_{t,i,b}) \implies Tape_{t+1,i,b} \big) \Bigg)$$

$$\wedge \bigvee_{z\in E, t\in\{0,\ldots,p(n)\}} State_{t,z}$$

SAT ist $\mathcal{NP}$-hart (10)

- Wenn $w \in L$, dann $z_0 w \vdash_M^r u z_e v$ mit $z_e \in E$ und $r \leq p(n)$.
- Lauf liefert Belegung I der Variablen von F, sodass $I(F) = 1$
  - In $Rand$ belege die besuchten Zustände, Positionen und Bandinhalte mit 1. Falls die Folge nach $t_e < p(n)$ Schritten endet, so setze die Variablen für $t > t_e$ auf die Werte für den Zeitpunkt t_e.
  - In $Anfang$ liefert die Belegung für die dort vorkommenden Variablen. Diese passen zur Anfangskonfiguration.
  - Für $Transition$ setze $I(State_{t,z}) = 1, I(Pos_{t,i}) = 1, Tape_{t,i,b}$ entsprechend der besuchten Zustände und alle anderen auf 0. Das macht $I(Transition) = 1$ und Variable $State_{t,z_e}$ für $z_e \in E$ wird dadurch auf 1 gesetzt.
- Daher $I(F) = 1$

SAT ist $\mathcal{NP}$-hart (11)

- Umgekehrt: Wenn es eine erfüllende Belegung I gibt mit $I(F) = 1$, dann kann daraus ein akzeptierender Lauf für die TM auf Eingabe w konstruiert werden.
- Damit gilt $w \in L \iff F$ ist erfüllbar.
- F kann in (deterministischer) Polynomialzeit berechnet werden:

  Größe von F: (Anzahl an Variablenvorkommen):

Subformel	Größe
$Rand$:	$O(p(n)^3)$
$Anfang$:	$O(p(n))$
$Transition$:	$O(p(n)^2)$
$Ende$:	$O(p(n))$
F	$O(p(n)^3)$

Satz von Cook

Insgesamt haben wir damit gezeigt:

Satz von Cook

Das Erfüllbarkeitsproblem der Aussagenlogik ist $\mathcal{NP}$-vollständig.