

# Theoretische Grundlagen der Informatik

## Vorlesung am 22. November 2018

INSTITUT FÜR THEORETISCHE INFORMATIK



Die **Eingabe** ist ein Wort aus  $\Sigma^*$ , zum Beispiel eine Kodierung einer Instanz  $I \in D_{\Pi}$  des Entscheidungsproblems  $\Pi$ .

- **1. Stufe:** Es wird ein Orakel aus  $\Gamma^*$  berechnet, zum Beispiel ein **Lösungsbeispiel** für  $I$ , also ein Indikator, warum  $I \in J_{\Pi}$  gelten sollte.
- **2. Stufe:** Hier wird nun dieser **Lösungsvorschlag überprüft**, d.h. es wird geprüft ob  $I \in J_{\Pi}$ .

## Beispiel TSP

- **1. Stufe:** Es wird zum Beispiel eine zykl. Permutation  $x_1 x_2 \cdots x_n$  der Knotenmenge  $V$  vorgeschlagen.  
D.h.  $(x_1, x_2, \dots, x_n), G = (V, E), c, k$  ist die Eingabe für 2. Stufe.
- **2. Stufe:** Es wird nun überprüft, ob  $x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_n$  eine Tour in  $G = (V, E)$  darstellt, deren Länge bezüglich  $c$  nicht größer als  $k$  ist.

- Das Orakel kann ein beliebiges Wort aus  $\Gamma^*$  sein.
- Darum muss in der Überprüfungsphase (2.Stufe) geprüft werden, ob das Orakel ein zulässiges Lösungsbeispiel für die gegebenen Eingabe ist.
- Ist dies der Fall, so kann die Berechnung zu diesem Zeitpunkt mit der Antwort “Ja” beendet werden.  
     $\rightsquigarrow$  gehe in Zustand  $q_J$
- Ist dies nicht der Fall, so kann die Berechnung zu diesem Zeitpunkt mit der Antwort “Nein” beendet werden.  
     $\rightsquigarrow$  gehe in Zustand  $q_N$
  
- Jede NTM  $\mathcal{M}$  hat zu einer gegebenen Eingabe  $x$  eine unendliche Anzahl möglicher Berechnungen, eine zu jedem Orakel aus  $\Gamma^*$ .
- Endet **mindestens eine** in  $q_J$ , so wird  $x$  akzeptiert.

- Die **Zeit**, die eine nichtdeterministische Turing-Maschine  $\mathcal{M}$  benötigt, um ein Wort  $x \in L_{\mathcal{M}}$  zu akzeptieren, ist definiert als die minimale Anzahl von Schritten, die  $\mathcal{M}$  in den Zustand  $q_f$  überführt.
- Die **Zeitkomplexitätsfunktion**  $T_{\mathcal{M}}: \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$  einer nichtdeterministischen Turing-Maschine  $\mathcal{M}$  ist definiert durch

$$T_{\mathcal{M}}(n) := \max \left( \left\{ 1 \right\} \cup \left\{ m \mid \begin{array}{l} \text{es gibt ein } x \in L_{\mathcal{M}} \text{ mit } |x| = n, \text{ so} \\ \text{dass die Zeit, die } \mathcal{M} \text{ benötigt,} \\ \text{um } x \text{ zu akzeptieren, } m \text{ ist} \end{array} \right\} \right)$$

- Die **Zeit**, die eine nichtdeterministische Turing-Maschine  $\mathcal{M}$  benötigt, um ein Wort  $x \in L_{\mathcal{M}}$  zu akzeptieren, ist definiert als die minimale Anzahl von Schritten, die  $\mathcal{M}$  in den Zustand  $q_f$  überführt.
- Die **Zeitkomplexitätsfunktion**  $T_{\mathcal{M}}: \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$  einer nichtdeterministischen Turing-Maschine  $\mathcal{M}$  ist definiert durch

$$T_{\mathcal{M}}(n) := \max \left( \{1\} \cup \left\{ m \mid \begin{array}{l} \text{es gibt ein } x \in L_{\mathcal{M}} \text{ mit } |x| = n, \text{ so} \\ \text{dass die Zeit, die } \mathcal{M} \text{ benötigt,} \\ \text{um } x \text{ zu akzeptieren, } m \text{ ist} \end{array} \right\} \right)$$

## Bemerkung 1

- Zur Berechnung von  $T_{\mathcal{M}}(n)$  wird für jedes  $x \in L_{\mathcal{M}}$  mit  $|x| = n$  eine kürzeste akzeptierende Berechnung betrachtet.
- Anschließend wird von diesen kürzesten die längste bestimmt.
- Somit ergibt sich eine *worst-case* Abschätzung.

- Die **Zeit**, die eine nichtdeterministische Turing-Maschine  $\mathcal{M}$  benötigt, um ein Wort  $x \in L_{\mathcal{M}}$  zu akzeptieren, ist definiert als die minimale Anzahl von Schritten, die  $\mathcal{M}$  in den Zustand  $q_f$  überführt.
- Die **Zeitkomplexitätsfunktion**  $T_{\mathcal{M}}: \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$  einer nichtdeterministischen Turing-Maschine  $\mathcal{M}$  ist definiert durch

$$T_{\mathcal{M}}(n) := \max \left( \{1\} \cup \left\{ m \mid \begin{array}{l} \text{es gibt ein } x \in L_{\mathcal{M}} \text{ mit } |x| = n, \text{ so} \\ \text{dass die Zeit, die } \mathcal{M} \text{ benötigt,} \\ \text{um } x \text{ zu akzeptieren, } m \text{ ist} \end{array} \right\} \right)$$

## Bemerkung 2

- Die Zeitkomplexität hängt nur von der Anzahl der Schritte ab, die bei einer akzeptierenden Berechnung auftreten.
- Hierbei umfasst die Anzahl der Schritte auch die Schritte der Orakelphase.
- Per Konvention ist  $T_{\mathcal{M}}(n) = 1$ , falls es keine Eingabe  $x$  der Länge  $n$  gibt, die von  $\mathcal{M}$  akzeptiert wird.

# Die Klasse $\mathcal{NP}$

Die Klasse  $\mathcal{NP}$  ist die Menge aller Sprachen  $L$  (Entscheidungsprobleme), für die eine nichtdeterministische Turing-Maschine existiert, deren Zeitkomplexitätsfunktion polynomial beschränkt ist, d.h. es existiert ein Polynom  $p$  mit

$$T_M(n) \leq p(n).$$

( $\mathcal{NP}$  steht für **nichtdeterministisch polynomial**.)

Die Klasse  $\mathcal{NP}$  ist die Menge aller Sprachen  $L$  (Entscheidungsprobleme), für die eine nichtdeterministische Turing-Maschine existiert, deren Zeitkomplexitätsfunktion polynomial beschränkt ist, d.h. es existiert ein Polynom  $p$  mit

$$T_M(n) \leq p(n).$$

( $\mathcal{NP}$  steht für **nichtdeterministisch polynomial**.)

## Bemerkungen

- Alle Sprachen in  $\mathcal{NP}$  sind entscheidbar.
- Informell ausgedrückt gehört  $\Pi$  zu  $\mathcal{NP}$ , falls  $\Pi$  folgende Eigenschaft hat:  
Ist die Antwort bei Eingabe eines Beispiels  $I$  von  $\Pi$  Ja, dann kann die Korrektheit der Antwort in polynomialer Zeit überprüft werden.



Die Klasse  $\mathcal{NP}$  ist die Menge aller Sprachen  $L$  (Entscheidungsprobleme), für die eine nichtdeterministische Turing-Maschine existiert, deren Zeitkomplexitätsfunktion polynomial beschränkt ist, d.h. es existiert ein Polynom  $p$  mit

$$T_{\mathcal{M}}(n) \leq p(n).$$

( $\mathcal{NP}$  steht für **nichtdeterministisch polynomial**.)

**Beispiel:**  $\text{TSP} \in \mathcal{NP}$ :

Denn zu gegebenem  $G = (V, E)$ ,  $c, k$  und einer festen zykl. Permutation  $x_1 x_2 \cdots x_n$  von  $V$  kann in  $O(|V| \cdot \log C)$  (wobei  $C$  die größte vorkommende Zahl ist) Schritten überprüft werden, ob

$$\{x_i, x_{i+1}\} \in E \text{ für } i = 1, \dots, n-1 \quad \text{und} \quad \sum_{i=1}^{n-1} c(\{x_i, x_{i+1}\}) \leq k$$

gilt.

Die Klasse  $\mathcal{P}$  ist die Menge aller Sprachen  $L$  (Entscheidungsprobleme), für die eine **deterministische** Turing-Maschine existiert, deren Zeitkomplexitätsfunktion **polynomial** beschränkt ist.

- ↪ Bei Eingabe einer Instanz  $I$  von  $\Pi$  kann die Existenz einer Lösung in polynomialer Zeit überprüft werden.

Die Klasse  $\mathcal{NP}$  ist die Menge aller Sprachen  $L$  (Entscheidungsprobleme), für die eine **nichtdeterministische** Turing-Maschine existiert, deren Zeitkomplexitätsfunktion **polynomial** beschränkt ist.

- ↪ Existiert für die Eingabe einer Instanz  $I$  von  $\Pi$  eine Lösung, dann kann die Korrektheit einer Lösung in polynomialer Zeit überprüft werden.

Große Frage: Ist  $\mathcal{P} = \mathcal{NP}$ ?

- Trivialerweise gilt:  $\mathcal{P} \subseteq \mathcal{NP}$ .
- **Frage:** Gilt  $\mathcal{P} \subset \mathcal{NP}$  oder  $\mathcal{P} = \mathcal{NP}$ ?
- Die Vermutung ist, dass  $\mathcal{P} \neq \mathcal{NP}$  gilt.
- Dazu betrachten wir Probleme, die zu den schwersten Problemen in  $\mathcal{NP}$  gehören.
- Dabei ist am schwersten im folgenden Sinne gemeint:
- Wenn ein schwerstes  $\mathcal{NP}$ -Problem trotzdem in  $\mathcal{P}$  liegt, so kann man folgern, dass alle  $\mathcal{NP}$ -Probleme in  $\mathcal{P}$  liegen, d.h.  $\mathcal{P} = \mathcal{NP}$ .
- Diese schwersten  $\mathcal{NP}$ -Probleme sind also Kandidaten, um  $\mathcal{P}$  und  $\mathcal{NP}$  zu trennen.
- Es wird sich zeigen, dass alle diese schwersten  $\mathcal{NP}$ -Probleme im Wesentlichen gleich schwer sind.

Eine **polynomiale Transformation** einer Sprache  $L_1 \subseteq \Sigma_1^*$  in eine Sprache  $L_2 \subseteq \Sigma_2^*$  ist eine Funktion  $f: \Sigma_1^* \rightarrow \Sigma_2^*$  mit den Eigenschaften:

- es existiert eine polynomiale deterministische Turing-Maschine, die  $f$  berechnet;
- für alle  $x \in \Sigma_1^*$  gilt:  $x \in L_1 \Leftrightarrow f(x) \in L_2$ .

Wir schreiben dann  $L_1 \propto L_2$  ( $L_1$  ist polynomial transformierbar in  $L_2$ ).

Eine Sprache  $L$  heißt  **$\mathcal{NP}$ -vollständig**, falls gilt:

- $L \in \mathcal{NP}$  und
- für alle  $L' \in \mathcal{NP}$  gilt  $L' \propto L$ .

Ein Entscheidungsproblem  $\Pi_1$  ist **polynomial transformierbar** in das Entscheidungsproblem  $\Pi_2$ , wenn eine Funktion  $f: D_{\Pi_1} \rightarrow D_{\Pi_2}$  existiert mit folgenden Eigenschaften:

- $f$  ist durch einen polynomialen Algorithmus berechenbar;
- $\forall I \in D_{\Pi_1}: I \in J_{\Pi_1} \iff f(I) \in J_{\Pi_2}$ .

Wir schreiben dann  $\Pi_1 \propto \Pi_2$ .

Ein Entscheidungsproblem  $\Pi$  heißt  **$\mathcal{NP}$ -vollständig**, falls gilt:

- $\Pi \in \mathcal{NP}$  und
- für alle  $\Pi' \in \mathcal{NP}$  gilt  $\Pi' \propto \Pi$ .

Eine **polynomiale Transformation** einer Sprache  $L_1 \subseteq \Sigma_1^*$  in eine Sprache  $L_2 \subseteq \Sigma_2^*$  ist eine Funktion  $f: \Sigma_1^* \rightarrow \Sigma_2^*$  mit den Eigenschaften:

- es existiert eine polynomiale deterministische Turing-Maschine, die  $f$  berechnet;
- für alle  $x \in \Sigma_1^*$  gilt:  $x \in L_1 \Leftrightarrow f(x) \in L_2$ .

Wir schreiben dann  $L_1 \propto L_2$  ( $L_1$  ist polynomial transformierbar in  $L_2$ ).

**Lemma.**  $\propto$  ist transitiv, d.h. aus  $L_1 \propto L_2$  und  $L_2 \propto L_3$  folgt  $L_1 \propto L_3$ .

**Beweis.** Die Hintereinanderausführung zweier polynomialer Transformationen ist wieder eine polynomiale Transformation.

## Korollar.

Falls  $L_1, L_2 \in \mathcal{NP}$ ,  $L_1 \propto L_2$  und  $L_1$   $\mathcal{NP}$ -vollständig, dann ist auch  $L_2$   $\mathcal{NP}$ -vollständig.

## Bedeutung.

Um also zu zeigen, dass ein Entscheidungsproblem  $\Pi$   $\mathcal{NP}$ -vollständig ist, gehen wir folgendermaßen vor. Wir beweisen:

- $\Pi \in \mathcal{NP}$  und
- $\Pi' \propto \Pi$  für ein bekanntes  $\mathcal{NP}$ -vollständiges Problem  $\Pi'$ .

## Problem.

- Wir haben noch kein “bekanntes  $\mathcal{NP}$ -vollständiges Problem”.
- Das erste  $\mathcal{NP}$ -vollständige Problem ist das Erfüllbarkeitsproblem SAT (satisfiability).

# Das Problem SAT (satisfiability)

Sei  $U = \{u_1, \dots, u_m\}$  eine Menge von booleschen Variablen.

Es heißen  $u_j, \bar{u}_j$  Literale.

Eine Wahrheitsbelegung für  $U$  ist eine Funktion  $t: U \rightarrow \{\text{wahr}, \text{falsch}\}$ .

Eine **Klausel** ist ein Boolescher Ausdruck der Form

$$y_1 \vee \dots \vee y_s \quad \text{mit} \quad y_i \in \underbrace{\{u_1, \dots, u_m\} \cup \{\bar{u}_1, \dots, \bar{u}_m\}}_{\text{Literalmenge}}$$

## Problem SAT

**Gegeben:** Menge  $U$  von Variablen, Menge  $C$  von Klauseln über  $U$ .

**Frage:** Existiert eine Wahrheitsbelegung von  $U$ , so dass jede Klausel in  $C$  erfüllt wird?

Beispiel:

$U = \{u_1, u_2\}$  mit  $C = \{u_1 \vee \bar{u}_2, \bar{u}_1 \vee u_2\}$  ist Ja-Instanz von SAT.

Wahrheitsbelegung  $t(u_1) = t(u_2) = \text{wahr}$  erfüllt alle Klauseln in  $C$ .



**Erfüllbar** (Ja-Instanz):

$$U = \{a, b, c, d, e\}, C = \{c \vee \bar{d}, \bar{a} \vee b \vee \bar{c} \vee d \vee e, \bar{c} \vee d\}$$

**Nicht erfüllbar** (Nein-Instanz):

$$U = \{a, b, c\}, C = \{a \vee b, \bar{a}, \bar{b} \vee c, \bar{c}\}$$

## Satz von Cook.

SAT ist  $\mathcal{NP}$ -vollständig.

## Erinnerung:

Problem  $\Pi$  heißt  $\mathcal{NP}$ -vollständig wenn

- $\Pi \in \mathcal{NP}$  und
- für alle  $\Pi' \in \mathcal{NP}$  gilt  $\Pi' \leq \Pi$ .

## Satz von Cook.

SAT ist  $\mathcal{NP}$ -vollständig.

## Erinnerung:

Problem  $\Pi$  heißt  $\mathcal{NP}$ -vollständig wenn

- $\Pi \in \mathcal{NP}$  und
- für alle  $\Pi' \in \mathcal{NP}$  gilt  $\Pi' \leq \Pi$ .

## Beweis:

- SAT  $\in \mathcal{NP}$  ist erfüllt:  
Für eine Instanz  $I$  von SAT (mit  $n$  Klauseln und  $m$  Variablen) und einer Wahrheitsbelegung  $t$  kann in  $O(m \cdot n)$  überprüft werden, ob  $t$  alle Klauseln erfüllt, d.h. ob  $I$  eine Ja-Instanz ist.
- Wir müssen zeigen, dass für jede Sprache  $L \in \mathcal{NP}$  gilt:  $L \leq L_{\text{SAT}}$ , wobei  $L_{\text{SAT}} = L[\text{SAT}, s]$  für ein geeignetes Kodierungsschema  $s$  ist.

# Beweis: das Setup

Wir müssen zeigen, dass für jede Sprache  $L \in \mathcal{NP}$  gilt:  $L \propto L_{\text{SAT}}$ , wobei  $L_{\text{SAT}} = L[\text{SAT}, s]$  für ein geeignetes Kodierungsschema  $s$  ist.

- Dazu muss für jede Sprache  $L \in \mathcal{NP}$  eine polynomiale Transformation  $f_L$  angegeben werden, für die gilt, dass für alle  $x \in \Sigma^*$  ( $\Sigma$  Alphabet zu  $L$ ) gilt

$$x \in L \iff f_L(x) \in L_{\text{SAT}}.$$

- Wir benutzen, dass es eine NTM  $\mathcal{M}$  zu  $L$  gibt, die  $L$  in polynomialer Laufzeit entscheidet.
- $\mathcal{M}$  sei gegeben durch  $(Q, \Sigma, \sqcup, \Gamma, q_0, \delta, q_J, q_N)$  und akzeptiere die Sprache  $L = L_{\mathcal{M}}$  in der Laufzeit  $T_{\mathcal{M}}(n) \leq p(n)$ , wobei  $p$  ein Polynom ist. O.B.d.A. gilt  $p(n) \geq n$ .

- Sei  $x \in \Sigma^*$  eine Eingabe mit  $n := |x|$ .
- Bei einer akzeptierenden Berechnung von  $\mathcal{M}$  für  $x$  ist die Anzahl der Berechnungsschritte beschränkt durch  $p(n)$ .
- An einer so beschränkten Berechnung können höchstens die Zellen  $-p(n)$  bis  $p(n) + 1$  des Bandes beteiligt sein.

Die Berechnung der deterministischen Stufe ist zu jedem Zeitpunkt eindeutig festgelegt durch:

- den jeweiligen Bandinhalt dieser  $-p(n)$  bis  $p(n) + 1$  Plätze,
- den Zustand der endlichen Kontrolle
- und der Position des Lese-/Schreibkopfs.

Im Folgenden beschreiben wir Bandinhalt, Zustand der endlichen Kontrolle und Position des Lese-/Schreibkopfs vollständig durch Variablen einer Instanz von SAT.

# Beweis: Konstruktion der Variablen

Bezeichne

- die Zustände aus  $Q$  durch  $q_0, q_1 = q_J, q_2 = q_N, q_3, \dots, q_r$
- die Symbole aus  $\Gamma$  durch  $s_0 = \sqcup, s_1, \dots, s_\ell$  mit  $|\Gamma| = \ell + 1$ .

Es gibt drei Typen von Variablen in der zugehörigen SAT-Instanz

Variable	Gültigkeitsbereich	Bedeutung
$Q[i, k]$	$0 \leq i \leq p(n)$ $0 \leq k \leq r$	zum Zeitpunkt $i$ der Überprüfungsphase ist $\mathcal{M}$ in Zustand $q_k$
$H[i, j]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n) + 1$	zum Zeitpunkt $i$ der Überprüfungsphase ist der Lese-/Schreibkopf an Position $j$ des Bandes
$S[i, j, k]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n) + 1$ $0 \leq k \leq \ell$	zum Zeitpunkt $i$ der Überprüfungsphase ist der Bandinhalt an Position $j$ das Symbol $s_k$

Variable	Gültigkeitsbereich	Bedeutung
$Q[i, k]$	$0 \leq i \leq p(n)$ $0 \leq k \leq r$	zum Zeitpunkt $i$ der Überprüfungsphase ist $\mathcal{M}$ in Zustand $q_k$
$H[i, j]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n) + 1$	zum Zeitpunkt $i$ der Überprüfungsphase ist der Lese-/Schreibkopf an Position $j$ des Bandes
$S[i, j, k]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n) + 1$ $0 \leq k \leq \ell$	zum Zeitpunkt $i$ der Überprüfungsphase ist der Bandinhalt an Position $j$ das Symbol $s_k$

- Eine Berechnung von  $\mathcal{M}$  induziert in kanonischer Weise eine Wahrheitsbelegung dieser Variablen.
- Wir benutzen folgende Konvention:
- Falls  $\mathcal{M}$  vor dem Zeitpunkt  $p(n)$  stoppt, bleibt  $\mathcal{M}$  in allen folgenden Zuständen in demselben Zustand und der Bandinhalt unverändert.

# Beweis: Konstruktion der Variablen

Variable	Gültigkeitsbereich	Bedeutung
$Q[i, k]$	$0 \leq i \leq p(n)$ $0 \leq k \leq r$	zum Zeitpunkt $i$ der Überprüfungsphase ist $\mathcal{M}$ in Zustand $q_k$
$H[i, j]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n) + 1$	zum Zeitpunkt $i$ der Überprüfungsphase ist der Lese-/Schreibkopf an Position $j$ des Bandes
$S[i, j, k]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n) + 1$ $0 \leq k \leq \ell$	zum Zeitpunkt $i$ der Überprüfungsphase ist der Bandinhalt an Position $j$ das Symbol $s_k$

Der Bandinhalt zum Zeitpunkt 0 der Überprüfungsphase sei

- Eingabe  $x$  auf Platz 1 bis  $n$
- Orakel  $w$  auf Platz  $-1$  bis  $-|w|$
- ansonsten Blanks.



- Eine beliebige Wahrheitsbelegung muss nicht notwendigerweise eine Berechnung induzieren (zum Beispiel  $Q[i, k] = Q[i, \ell]$  für  $k \neq \ell$ ).

Also konstruiere Transformation  $f_L$  die Klauseln einführt, so dass äquivalent ist:

- Für Eingabe  $x$  gibt es eine akzeptierende Berechnung, deren Überprüfungsphase höchstens  $p(n)$  Zeit benötigt, und deren Orakel höchstens Länge  $p(n)$  hat.
- Es gibt eine erfüllende Belegung für die SAT-Instanz  $f_L(x)$ .

Also konstruiere Transformation  $f_L$  die Klauseln einführt, so dass äquivalent ist:

- Für Eingabe  $x$  gibt es eine akzeptierende Berechnung, deren Überprüfungsphase höchstens  $p(n)$  Zeit benötigt, und deren Orakel höchstens Länge  $p(n)$  hat.
- Es gibt eine erfüllende Belegung für die SAT-Instanz  $f_L(x)$ .

Damit können wir dann schließen:

- $x \in L \Leftrightarrow$  es existiert eine akzeptierende Berechnung von  $\mathcal{M}$  bei Eingabe  $x$
- $\Leftrightarrow$  es existiert eine akzeptierende Berechnung von  $\mathcal{M}$  bei Eingabe  $x$  mit höchstens  $p(n)$  Schritten in der Überprüfungsphase und einem Orakel  $w$  der Länge  $|w| \leq p(n)$
- $\Leftrightarrow$  es existiert eine erfüllende Wahrheitsbelegung für die Klauselmenge  $f_L(x)$

Klausel-  
gruppe

Einschränkung / Bedeutung

---

- $G_1$  Zum Zeitpunkt  $i$  ist  $\mathcal{M}$  in genau einem Zustand.
- $G_2$  Zum Zeitpunkt  $i$  hat der Lese-/Schreibkopf genau eine Position.
- $G_3$  Zum Zeitpunkt  $i$  enthält jede Bandstelle genau ein Symbol aus  $\Gamma$ .
- $G_4$  Festlegung der Anfangskonfiguration zum Zeitpunkt 0:  $\mathcal{M}$  ist im Zustand  $q_0$ , der Lese-/Schreibkopf steht an Position 1 des Bandes; in den Zellen 1 bis  $n$  steht das Wort  $x = s_{k_1} \cdot \dots \cdot s_{k_n}$
- $G_5$  Zum Zeitpunkt  $p(n)$  hat  $\mathcal{M}$  den Zustand  $q_J$  erreicht.
- $G_6$  Zu jedem Zeitpunkt  $i$  folgt die Konfiguration von  $\mathcal{M}$  zum Zeitpunkt  $i + 1$  aus einer einzigen Anwendung von  $\delta$  aus der Konfiguration von  $\mathcal{M}$  zum Zeitpunkt  $i$ .

Zum Zeitpunkt  $i$  ist  $\mathcal{M}$  in genau einem Zustand.

Konstruktion:

- Zu jedem Zeitpunkt  $i$  ist  $\mathcal{M}$  in mindestens einem Zustand

$$Q[i, 0] \vee \dots \vee Q[i, r] \quad \text{für } 0 \leq i \leq p(n)$$

- Zu jedem Zeitpunkt  $i$  ist  $\mathcal{M}$  in nicht mehr als einem Zustand

$$\overline{Q[i, j]} \vee \overline{Q[i, j']} \quad \text{für } 0 \leq i \leq p(n), 0 \leq j < j' \leq r$$

### Zum Zeitpunkt $i$ hat der Lese-/Schreibkopf genau eine Position

Konstruktion:

- Zum Zeitpunkt  $i$  hat der Lese-/Schreibkopf mindestens eine Position

$$H[i, -p(n)] \vee \dots \vee H[i, p(n) + 1] \quad \text{für } 0 \leq i \leq p(n)$$

- Zum Zeitpunkt  $i$  hat der Lese-/Schreibkopf höchstens eine Position

$$\overline{H[i, j]} \vee \overline{H[i, j']} \quad \text{für } \begin{cases} 0 \leq i \leq p(n) \\ -p(n) \leq j < j' \leq p(n) + 1 \end{cases}$$

**Zum Zeitpunkt  $i$  enthält jede Bandstelle genau ein Symbol**

Konstruktion:

- Zum Zeitpunkt  $i$  enthält Bandstelle  $j$  mindestens ein Symbol

$$S[i, j, 0] \vee S[i, j, 1] \vee \dots \vee S[i, j, \ell] \quad \text{für} \quad \begin{cases} 0 \leq i \leq p(n) \\ -p(n) \leq j \leq p(n) + 1 \end{cases}$$

- Zum Zeitpunkt  $i$  enthält Bandstelle  $j$  höchstens ein Symbol

$$\overline{S[i, j, k]} \vee \overline{S[i, j, k']} \quad \text{für} \quad \begin{cases} 0 \leq i \leq p(n) \\ -p(n) \leq j \leq p(n) + 1 \\ 0 \leq k < k' \leq \ell \end{cases}$$

### Festlegung der Anfangskonfiguration zum Zeitpunkt 0

Konstruktion:

- $\mathcal{M}$  ist im Zustand  $q_0$

$$Q[0, 0]$$

- der Lese-/Schreibkopf steht an Position 1 des Bandes

$$H[0, 1]$$

- in den Zellen 1 bis  $n$  steht das Wort  $x = s_{k_1} \dots s_{k_n}$

$$\begin{cases} S[0, 0, 0], S[0, 1, k_1], \dots, S[0, n, k_n] & \text{für Eingabe } x = s_{k_1} \dots s_{k_n} \\ S[0, n+1, 0], \dots, S[0, p(n)+1, 0] & \text{für rechts der Eingabe} \end{cases}$$

## Klauselgruppe 5:

Zum Zeitpunkt  $p(n)$  hat  $\mathcal{M}$  den Zustand  $q_j$  erreicht.

Konstruktion:

$$Q[p(n), 1]$$



**Zu jedem Zeitpunkt  $i$  folgt die Konfiguration von  $\mathcal{M}$  zum Zeitpunkt  $i + 1$  aus einer einzigen Anwendung von  $\delta$  aus der Konfiguration von  $\mathcal{M}$  zum Zeitpunkt  $i$ .**

Wir unterteilen Klauselgruppe  $G_6$  in zwei Teilgruppen  $G_{6,1}$ ,  $G_{6,2}$ .

- $G_{6,1}$ :  
Falls  $\mathcal{M}$  zum Zeitpunkt  $i$  an der Position  $j$  das Symbol  $s_k$  hat und der Lese-/Schreibkopf nicht an der Position  $j$  steht, dann hat  $\mathcal{M}$  auch zum Zeitpunkt  $i + 1$  an Position  $j$  das Symbol  $s_k$ .
- $G_{6,2}$ :  
Der Wechsel von einer Konfiguration zur nächsten entspricht tatsächlich  $\delta$ .

**Falls  $\mathcal{M}$  zum Zeitpunkt  $i$  an der Position  $j$  das Symbol  $s_k$  hat und der Lese-/Schreibkopf nicht an der Position  $j$  steht, dann hat  $\mathcal{M}$  auch zum Zeitpunkt  $i + 1$  an Position  $j$  das Symbol  $s_k$ .**

Konstruktion:

$$\left( S[i, j, k] \wedge \overline{H[i, j]} \right) \implies S[i + 1, j, k] \quad \text{für} \quad \begin{cases} 0 \leq i < p(n) \\ -p(n) \leq j \leq p(n) + 1 \\ 0 \leq k \leq \ell \end{cases}$$

Dies ergibt die Klausel

$$\overline{S[i, j, k]} \vee H[i, j] \vee S[i + 1, j, k] \quad \text{für} \quad \begin{cases} 0 \leq i < p(n) \\ -p(n) \leq j \leq p(n) + 1 \\ 0 \leq k \leq \ell \end{cases}$$

**Der Wechsel von einer Konfiguration zur nächsten entspricht tatsächlich  $\delta$ .**

■ Sei  $\delta(q_a, s_u) = (q_b, s_v, d)$ ,  $d \in \{-1, 0, 1\}$  (steht für L,N,R)

Konstruktion:

$$\begin{aligned} & (H[i, j] \wedge Q[i, a] \wedge S[i, j, u]) \Rightarrow H[i + 1, j + d] \\ \text{und } & (H[i, j] \wedge Q[i, a] \wedge S[i, j, u]) \Rightarrow Q[i + 1, b] \\ \text{und } & (H[i, j] \wedge Q[i, a] \wedge S[i, j, u]) \Rightarrow S[i + 1, j, v] \end{aligned}$$

Dies ergibt folgende Klauseln

$$\begin{aligned} & \overline{H[i, j]} \vee \overline{Q[i, a]} \vee \overline{S[i, j, u]} \vee H[i + 1, j + d] \\ & \overline{H[i, j]} \vee \overline{Q[i, a]} \vee \overline{S[i, j, u]} \vee Q[i + 1, b] \\ & \overline{H[i, j]} \vee \overline{Q[i, a]} \vee \overline{S[i, j, u]} \vee S[i + 1, j, v] \end{aligned}$$

für  $0 \leq i < p(n)$ ,  $-p(n) \leq j \leq p(n) + 1$ ,  $0 \leq a, b \leq r$ ,  $0 \leq u, v \leq \ell$

- Wir beweisen, dass SAT  $\mathcal{NP}$ -vollständig ist.
- Für jede beliebige aber feste Sprache  $L \in \mathcal{NP}$  konstruieren wir eine polynomiale Reduktion  $L \leq L_{\text{SAT}}$ , d.h.
  - wir konstruieren eine Abbildung  $f_L: \Sigma^* \rightarrow D_{\text{SAT}}$
  - so dass für alle  $x \in \Sigma^*$  gilt:  $x \in L \Leftrightarrow f_L(x) \in J_{\text{SAT}}$
  - und  $f_L$  polynomial berechenbar ist.

- Wir beweisen, dass SAT  $\mathcal{NP}$ -vollständig ist.
- Für jede beliebige aber feste Sprache  $L \in \mathcal{NP}$  konstruieren wir eine polynomiale Reduktion  $L \leq L_{\text{SAT}}$ , d.h.
  - wir konstruieren eine Abbildung  $f_L: \Sigma^* \rightarrow D_{\text{SAT}}$
  - so dass für alle  $x \in \Sigma^*$  gilt:  $x \in L \Leftrightarrow f_L(x) \in J_{\text{SAT}}$
  - und  $f_L$  polynomial berechenbar ist.
- Wir betrachten eine NTM  $\mathcal{M}$  die  $L$  entscheidet.
- Wir betrachten Polynom  $p(n)$  das die Laufzeit von  $\mathcal{M}$  beschränkt.
- Für beliebiges  $x \in \Sigma^*$  konstruieren wir  $f_L(x)$  mit Variablen  $Q[i, j], H[i, j], S[i, j, k]$  und Klauselmenge  $C := G_1 \cup G_2 \cup \dots \cup G_6$ .

- Wir beweisen, dass SAT  $\mathcal{NP}$ -vollständig ist.
- Für jede beliebige aber feste Sprache  $L \in \mathcal{NP}$  konstruieren wir eine polynomiale Reduktion  $L \leq L_{\text{SAT}}$ , d.h.
  - wir konstruieren eine Abbildung  $f_L: \Sigma^* \rightarrow D_{\text{SAT}}$
  - so dass für alle  $x \in \Sigma^*$  gilt:  $x \in L \Leftrightarrow f_L(x) \in J_{\text{SAT}}$
  - und  $f_L$  polynomial berechenbar ist.
- Wir betrachten eine NTM  $\mathcal{M}$  die  $L$  entscheidet.
- Wir betrachten Polynom  $p(n)$  das die Laufzeit von  $\mathcal{M}$  beschränkt.
- Für beliebiges  $x \in \Sigma^*$  konstruieren wir  $f_L(x)$  mit Variablen  $Q[i, j], H[i, j], S[i, j, k]$  und Klauselmenge  $C := G_1 \cup G_2 \cup \dots \cup G_6$ .

$x \in L \implies \begin{array}{l} \exists \text{ akzeptierende} \\ \text{Berechnung von } \mathcal{M} \\ \text{für Eingabe } x \end{array} \implies \begin{array}{l} \exists \text{ Wahrheitsbelegung} \\ \text{die alle Klauseln in } C \\ \text{erfüllt.} \end{array}$

- Wir beweisen, dass SAT  $\mathcal{NP}$ -vollständig ist.
- Für jede beliebige aber feste Sprache  $L \in \mathcal{NP}$  konstruieren wir eine polynomiale Reduktion  $L \leq L_{\text{SAT}}$ , d.h.
  - wir konstruieren eine Abbildung  $f_L: \Sigma^* \rightarrow D_{\text{SAT}}$
  - so dass für alle  $x \in \Sigma^*$  gilt:  $x \in L \Leftrightarrow f_L(x) \in J_{\text{SAT}}$
  - und  $f_L$  polynomial berechenbar ist.
- Wir betrachten eine NTM  $\mathcal{M}$  die  $L$  entscheidet.
- Wir betrachten Polynom  $p(n)$  das die Laufzeit von  $\mathcal{M}$  beschränkt.
- Für beliebiges  $x \in \Sigma^*$  konstruieren wir  $f_L(x)$  mit Variablen  $Q[i, j], H[i, j], S[i, j, k]$  und Klauselmenge  $C := G_1 \cup G_2 \cup \dots \cup G_6$ .

$$x \in L \iff \begin{array}{l} \exists \text{ akzeptierende} \\ \text{Berechnung von } \mathcal{M} \\ \text{für Eingabe } x \end{array} \iff \begin{array}{l} \exists \text{ Wahrheitsbelegung} \\ \text{die alle Klauseln in } C \\ \text{erfüllt.} \end{array}$$

- Wir beweisen, dass SAT  $\mathcal{NP}$ -vollständig ist.
- Für jede beliebige aber feste Sprache  $L \in \mathcal{NP}$  konstruieren wir eine polynomiale Reduktion  $L \leq L_{\text{SAT}}$ , d.h.
  - wir konstruieren eine Abbildung  $f_L: \Sigma^* \rightarrow D_{\text{SAT}}$
  - so dass für alle  $x \in \Sigma^*$  gilt:  $x \in L \Leftrightarrow f_L(x) \in J_{\text{SAT}}$
  - und  $f_L$  polynomial berechenbar ist.
- Wir betrachten eine NTM  $\mathcal{M}$  die  $L$  entscheidet.
- Wir betrachten Polynom  $p(n)$  das die Laufzeit von  $\mathcal{M}$  beschränkt.
- Für beliebiges  $x \in \Sigma^*$  konstruieren wir  $f_L(x)$  mit Variablen  $Q[i, j], H[i, j], S[i, j, k]$  und Klauselmenge  $C := G_1 \cup G_2 \cup \dots \cup G_6$ .

$$x \in L \iff \begin{array}{l} \exists \text{ akzeptierende} \\ \text{Berechnung von } \mathcal{M} \\ \text{für Eingabe } x \end{array} \iff \begin{array}{l} \exists \text{ Wahrheitsbelegung} \\ \text{die alle Klauseln in } C \\ \text{erfüllt.} \end{array}$$



# Polynomialität der Transformation

Wir schätzen die Anzahl der Literale in den Klauselgruppen ab.

Zum Zeitpunkt  $i$  ist  $\mathcal{M}$  in genau einem Zustand.

Konstruktion:

- Zu jedem Zeitpunkt  $i$  ist  $\mathcal{M}$  in mindestens einem Zustand

$$Q[i, 0] \vee \dots \vee Q[i, r] \quad \text{für } 0 \leq i \leq p(n)$$

- Zu jedem Zeitpunkt  $i$  ist  $\mathcal{M}$  in nicht mehr als einem Zustand

$$\overline{Q[i, j]} \vee \overline{Q[i, j']} \quad \text{für } 0 \leq i \leq p(n), 0 \leq j < j' \leq r$$

**Abschätzung:**

$$(r + 1) \cdot (p(n) + 1) + 2 \cdot (p(n) + 1) \frac{1}{2} r(r + 1)$$

**Zum Zeitpunkt  $i$  hat der Lese-/Schreibkopf genau eine Position**

Konstruktion:

- Zum Zeitpunkt  $i$  hat der Lese-/Schreibkopf mindestens eine Position

$$H[i, -p(n)] \vee \dots \vee H[i, p(n) + 1] \quad \text{für } 0 \leq i \leq p(n)$$

- Zum Zeitpunkt  $i$  hat der Lese-/Schreibkopf höchstens eine Position

$$\overline{H[i, j]} \vee \overline{H[i, j']} \quad \text{für } \begin{cases} 0 \leq i \leq p(n) \\ -p(n) \leq j < j' \leq p(n) + 1 \end{cases}$$

**Abschätzung:**

$$(2p(n) + 2) \cdot (p(n) + 1) + 2 \cdot (p(n) + 1) \frac{1}{2} (2p(n) + 1)(2p(n) + 2)$$

**Zum Zeitpunkt  $i$  enthält jede Bandstelle genau ein Symbol**

Konstruktion:

- Zum Zeitpunkt  $i$  enthält jede Bandstelle mindestens ein Symbol

$$S[i, j, 0] \vee S[i, j, 1] \vee \dots \vee S[i, j, \ell] \quad \text{für} \quad \begin{cases} 0 \leq i \leq p(n) \\ -p(n) \leq j \leq p(n) + 1 \end{cases}$$

- Zum Zeitpunkt  $i$  enthält jede Bandstelle höchstens ein Symbol

$$\overline{S[i, j, k]} \vee \overline{S[i, j, k']} \quad \text{für} \quad \begin{cases} 0 \leq i \leq p(n) \\ -p(n) \leq j \leq p(n) + 1 \\ 0 \leq k < k' \leq \ell \end{cases}$$

**Abschätzung:**

$$(\ell + 1) \cdot (p(n) + 1)(2p(n) + 2) + 2 \cdot (p(n) + 1)(2p(n) + 2) \frac{1}{2} \ell(\ell + 1)$$

## Festlegung der Anfangskonfiguration zum Zeitpunkt 0

- $\mathcal{M}$  ist im Zustand  $q_0$

$$Q[0, 0]$$

- der Lese-/Schreibkopf steht an Position 1 des Bandes

$$H[0, 1]$$

- in den Zellen 1 bis  $n$  steht das Wort  $x = s_{k_1} \dots s_{k_n}$

$$\begin{cases} S[0, 0, 0], S[0, 1, k_1], \dots, S[0, n, k_n] & \text{für Eingabe } x = s_{k_1} \dots s_{k_n} \\ S[0, n+1, 0], \dots, S[0, p(n)+1, 0] & \text{für rechts der Eingabe} \end{cases}$$

## Abschätzung:

$$2 + (n + 1) + (p(n) + 1 - n)$$

# Polynomialität — Klauselgruppe 5:

**Bis zum Zeitpunkt  $p(n)$  hat  $\mathcal{M}$  den Zustand  $q_J$  erreicht.**

Konstruktion:

$$Q[p(n), 1]$$

**Abschätzung:**

1

Falls  $\mathcal{M}$  zum Zeitpunkt  $i$  an der Position  $j$  das Symbol  $s_k$  hat und der Lese-/Schreibkopf nicht an der Position  $j$  steht, dann hat  $\mathcal{M}$  auch zum Zeitpunkt  $i + 1$  an Position  $j$  das Symbol  $s_k$ .

Konstruktion:

$$\left( S[i, j, k] \wedge \overline{H[i, j]} \right) \implies S[i + 1, j, k] \quad \text{für} \quad \begin{cases} 0 \leq i < p(n) \\ -p(n) \leq j \leq p(n) + 1 \\ 0 \leq k \leq \ell \end{cases}$$

Dies ergibt die Klausel

$$\overline{S[i, j, k]} \vee H[i, j] \vee S[i + 1, j, k] \quad \text{für} \quad \begin{cases} 0 \leq i < p(n) \\ -p(n) \leq j \leq p(n) + 1 \\ 0 \leq k \leq \ell \end{cases}$$

**Abschätzung:**

$$3 \cdot p(n)(2p(n) + 2)(\ell + 1)$$

Der Wechsel von einer Konfiguration zur nächsten entspricht tatsächlich  $\delta$ .

- Sei  $\delta(q_a, s_u) = (q_b, s_v, d)$ ,  $d \in \{-1, 0, 1\}$  (steht für L,N,R)

$$\overline{H[i, j]} \vee \overline{Q[i, a]} \vee \overline{S[i, j, u]} \vee H[i + 1, j + d]$$

$$\overline{H[i, j]} \vee \overline{Q[i, a]} \vee \overline{S[i, j, u]} \vee Q[i + 1, b]$$

$$\overline{H[i, j]} \vee \overline{Q[i, a]} \vee \overline{S[i, j, u]} \vee S[i + 1, j, v]$$

für  $0 \leq i < p(n)$ ,  $-p(n) \leq j \leq p(n) + 1$ ,  $0 \leq a, b \leq r$ ,  $0 \leq u, v \leq \ell$

**Abschätzung:**

$$4 \cdot 3 \cdot p(n)(2p(n) + 2)(r + 1)(\ell + 1)$$



Wir schätzen die Anzahl der Literale in den Klauselgruppen ab.

- $G_1: (p(n) + 1)(r + 1)^2$
- $G_2: 4(p(n) + 1)^3$
- $G_3: 2(p(n) + 1)^2(\ell + 1)^2$
- $G_4: p(n) + 4$
- $G_5: 1$
- $G_6: 6p(n)(p(n) + 1)(\ell + 1)(4r + 5)$

Wir schätzen die Anzahl der Literale in den Klauselgruppen ab.

- $G_1: (p(n) + 1)(r + 1)^2$
  - $G_2: 4(p(n) + 1)^3$
  - $G_3: 2(p(n) + 1)^2(\ell + 1)^2$
  - $G_4: p(n) + 4$
  - $G_5: 1$
  - $G_6: 6p(n)(p(n) + 1)(\ell + 1)(4r + 5)$
- 
- $r$  und  $\ell$  sind Konstanten, die durch  $\mathcal{M}$  (und damit durch  $L$ ) induziert werden
  - $p(n)$  ist ein Polynom in  $n$

Wir schätzen die Anzahl der Literale in den Klauselgruppen ab.

- $G_1: (p(n) + 1)(r + 1)^2$
  - $G_2: 4(p(n) + 1)^3$
  - $G_3: 2(p(n) + 1)^2(\ell + 1)^2$
  - $G_4: p(n) + 4$
  - $G_5: 1$
  - $G_6: 6p(n)(p(n) + 1)(\ell + 1)(4r + 5)$
- Also sind alle Größen polynomial in  $n$ .
- Die angegebene Funktion  $f_L$  ist damit eine polynomiale Transformation von  $L$  nach  $L_{\text{SAT}}$ .

## Satz von Cook.

SAT ist  $\mathcal{NP}$ -vollständig.

### Beweisidee:

- Zu gegebener Sprache  $L \in \mathcal{NP}$  und Eingabe  $x \in \Sigma^*$  konstruiere eine SAT-Instanz  $f_L(x) \in D_{\text{SAT}}$ .
- Variablen der SAT-Instanz kodieren mögliche Zustände der NTM zu verschiedenen Zeitpunkten.
- Klauseln der SAT-Instanz garantieren
  - sinnvolle Zustandsübergänge, so wie von der NTM definiert
  - Erfüllbarkeit genau dann, wenn die NTM akzeptiert
- Dazu brauchen wir nur polynomial viele Variablen und Klauseln.

## Satz von Cook.

SAT ist  $\mathcal{NP}$ -vollständig.

Damit haben wir gezeigt:

- SAT gehört zu den schwersten Problemen in  $\mathcal{NP}$ .
- Könnte man SAT in polynomialer Zeit lösen, so könnte man alle Probleme in  $\mathcal{NP}$  in polynomialer Zeit lösen.
- Lässt sich SAT in polynomialer Zeit auf ein Problem  $\Pi$  transformieren, so muss  $\Pi$   $\mathcal{NP}$ -vollständig sein.
  - Dazu mehr in der nächsten Vorlesung.