

Berechenbarkeits- und Komplexitätstheorie

Lerneinheit 5: Die Klasse NP

Prof. Dr. Christoph Karg

Studiengang Informatik
Hochschule Aalen



Wintersemester 2015/2016



Thema dieser Lerneinheit ist die Klasse **NP**. Diese Klasse enthält viele für die Praxis relevante Probleme, für die bis heute kein effizienter Algorithmus bekannt ist.

Die Lerneinheit gliedert sich in:

- Definition von NP
- Berechnung von Lösungen mittels Entscheidungsproblemen
- Effizient verifizierbare Probleme
- NP-Vollständigkeit

Definition. Die Klasse NP enthält alle Sprachen, die durch eine nichtdeterministische, polynomial zeitbeschränkte Turing Maschine akzeptierbar sind. Formal:

$$NP = \bigcup_{k=0}^{\infty} \text{NTIME}(n^k)$$

Bemerkungen.

- Die Klasse NP enthält viele für die Praxis relevante Probleme
- Es gilt: $P \subseteq NP$
- Es ist nicht bekannt, ob alle Sprachen in NP effizient lösbar sind

Berechnung von Lösungen

Ziel: Einsatz eines Entscheidungsproblems zur Berechnung einer Lösung

Beispiel: Erfüllbarkeit von aussagenlogischen Formeln (SAT)

Gegeben: Aussagenlogische Formel F über den Variablen $x_1, \dots, x_n$

Aufgabe: Berechne eine erfüllende Belegung für F , falls eine solche existiert

Ansatz: Konstruiere eine erfüllende Belegung durch schrittweise Belegung der Variablen $x_1, \dots, x_n$

Berechnung von Lösungen (Forts.)

ASSIGNMENT($F(x_1, \dots, x_n)$)

Input: Aussagenlogische Formel F mit den Variablen $x_1, \dots, x_n$

Output: Erfüllende Belegung für F , falls eine solche existiert

- 1 **if** $F \notin \text{SAT}$ **then**
- 2 **return** F ist unerfüllbar
- 3 **else**
- 4 **for** $i := 1$ **to** n **do**
- 5 *Ersetze in F jedes Vorkommen von x_i durch $(x_i \vee \neg x_i)$*
 und speichere die resultierende Formel in F'
- 6 $b_i := 1$

Berechnung von Lösungen (Forts.)

```
7   if  $F' \notin \text{SAT}$  then  
8       Ersetze in  $F$  jedes Vorkommen von  $x_i$  durch  $(x_i \wedge \neg x_i)$   
        und speichere die resultierende Formel in  $F'$   
9        $b_i := 0$   
10       $F := F'$   
11  return Belegung  $(b_1, \dots, b_n)$ 
```

Berechnung von Lösungen (Forts.)

Bemerkungen:

- Durch die Belegung der Variablen wird die Länge der Formel in jedem Schritt höchstens verdreifacht
- Die Laufzeit von $\text{ASSIGNMENT}(F(x_1, \dots, x_n))$ hängt ab von
 - ▷ Umformung der Formel F
 $\rightsquigarrow$ Aufwand $O(n \cdot |F|)$
 - ▷ Aufruf des SAT Algorithmus
 $\rightsquigarrow$ Aufwand $O((n + 1) \cdot \text{time}_{\text{SAT}}(3 \cdot |F|))$
- Ist $\text{SAT} \in \text{P}$, dann ist die Berechnung einer erfüllenden Belegung in Polynomialzeit durchführbar

Definition. Eine Sprache A ist **in Polynomialzeit verifizierbar**, falls es eine Sprache $B \in P$ und ein Polynom $p : \mathbb{N} \mapsto \mathbb{N}$ gibt so dass für alle $x \in \Sigma^*$:

$$x \in A \iff \text{es gibt ein } w \text{ mit } |w| \leq p(|x|): \langle x, w \rangle \in B$$

Das Wort w bezeichnet man als **Beleg** (**Beweis**, **witness**) für die Zugehörigkeit von x zur Sprache A .

Offensichtlich: Jede Sprache in P ist in Polynomialzeit verifizierbar

SAT ist effizient verifizierbar

Beispiel: die Sprache SAT ist in Polynomialzeit verifizierbar

Ansatz: Erfüllende Belegung ist ein Beleg für $F \in \text{SAT}$

Arbeitsweise des Algorithmus auf Eingabe $\langle F, w \rangle$:

1. Überprüfe, ob w eine Belegung der in F vorkommenden Variablen darstellt
2. Überprüfe, ob w eine erfüllende Belegung für F ist, d.h., ob $\langle F, w \rangle \in \text{CVP}$ ist

Für jede Formel F ist die Belegung in $O(|F|)$ Buchstaben darstellbar. Also gilt: $|w| \leq O(|F|)$.

Das entsprechende Polynom ist $p(n) = c \cdot n$, wobei c die Konstante der O -Notation ist

Satz. Eine Sprache A ist in Polynomialzeit verifizierbar genau dann, wenn $A \in \text{NP}$.

Beweis. " $\Leftarrow$ ": Sei A eine in Polynomialzeit verifizierbare Sprache. Dann existiert eine Sprache B und ein Polynom p , welche die Eigenschaften der obigen Definition erfüllen.

Ansatz: Konstruiere nichtdeterministisch ein Wort w der Länge $\leq p(|x|)$ und überprüfe, ob $\langle x, w \rangle \in P$

Effizient verifizierbare Sprachen und NP (Forts.)

GUESSANDCHECK(x)

Input: Wort x

Output: true, falls $x \in A$, false, sonst.

```
1   $i := 0; w := \varepsilon$ 
2  repeat
3      Wähle nichtdeterministisch  $a \in \Sigma \cup \{\perp\}$ 
4      if  $a \neq \perp$  then
5           $w := wa$ 
6           $i := i + 1$ 
7      until  $i \geq p(|x|)$  or  $a = \perp$ 
8      if  $\langle x, w \rangle \in B$  then
9          return true
10 else
11     return false
```

Effizient verifizierbare Sprachen und NP (Forts.)

Analyse:

- Der Algorithmus akzeptiert die Eingabe x genau dann, wenn es einen Beleg w gibt, so dass $\langle x, w \rangle \in B$. Folglich akzeptiert der Algorithmus die Sprache A
- Laufzeit:
 - ▷ Nichtdeterministische Auswahl von w
 $\rightsquigarrow O(p(|x|))$
 - ▷ Aufruf des Verifikationsalgorithmus
 $\rightsquigarrow O(|x|^k)$ für eine Konstante k

Insgesamt: polynomiale Laufzeit

Ergebnis: $A \in \text{NP}$

Effizient verifizierbare Sprachen und NP (Forts.)

“ $\Leftarrow$ ”: Betrachte eine beliebige Sprache $A \in \text{NP}$.

Es gibt eine nichtdeterministische polynomial zeitbeschränkte Turing Maschine $M = (Z, \Sigma, \Gamma, \sqcup, \delta, z_{acc}, z_{rej})$ mit $L(M) = A$.

Sei $t(n)$ ein Polynom, so dass M allen Eingaben der Länge n höchstens $t(n)$ Rechenschritte ausführt

Für alle $x \in \Sigma^*$ gilt: $x \in A$ genau dann, wenn es gibt eine Folge $C_0, C_1, \dots, C_k$ von Konfigurationen gibt mit:

- $C_0 = (\sqcup, z_s, x)$
- Für alle $i = 1, \dots, k$ gilt: C_i ist eine Folgekonfiguration von C_{i-1}
- C_k ist eine akzeptierende Endkonfiguration

Für jedes Wort x ist algorithmisch feststellbar, ob eine Konfigurationsfolge $C_0, C_1, \dots, C_k$ die obigen Eigenschaften erfüllt.

Wegen der Laufzeitschranke gilt:

- $k \leq t(|x|)$
- $|C_i| \leq t(|x|) + O(1)$ für alle $i = 0, \dots, k$

Die Gesamtlänge der Konfigurationsfolge ist $O(t(|x|)^2)$.

Effizient verifizierbare Sprachen und NP (Forts.)

Der Aufwand zur Überprüfung der obigen Eigenschaft ist also gleich $O(t(|x|)^k)$ für eine Konstante k .

Betrachte nun die Sprache

$$B_M = \left\{ \langle x, w \rangle \mid \begin{array}{l} w = \langle C_0, C_1, \dots, C_k \rangle \text{ ist eine} \\ \text{Folge von Konfigurationen einer} \\ \text{akzeptierenden Berechnung} \\ \text{von } M \text{ auf Eingabe } x \end{array} \right\}$$

Aufgrund der obigen Überlegungen folgt: $B_M \in P$.

Ergebnis: A ist in Polynomialzeit verifizierbar

Definition. Sei A eine beliebige Sprache.

- A ist **NP-hart**, falls $B \leq_m^P A$ für jede Sprache $B \in \text{NP}$.
- A ist **NP-vollständig**, falls A NP-hart und in NP ist.

Bemerkungen

- Die NP-vollständigen Probleme sind die schwierigsten Probleme in NP
- Ist ein NP-vollständiges Problem in P, dann folgt $P = \text{NP}$
- Ist $P \neq \text{NP}$, dann gibt es für keines der NP-vollständigen Probleme einen effizienten Algorithmus

SAT ist NP-vollständig

Satz. (Cook) SAT ist NP-vollständig

Beweis. Da SAT effizient verifizierbar ist ($\rightsquigarrow$ CVP), gilt $\text{SAT} \in \text{NP}$.

Sei A eine beliebige Sprache in NP. Laut Annahme gibt es eine nichtdeterministische Turing Maschine

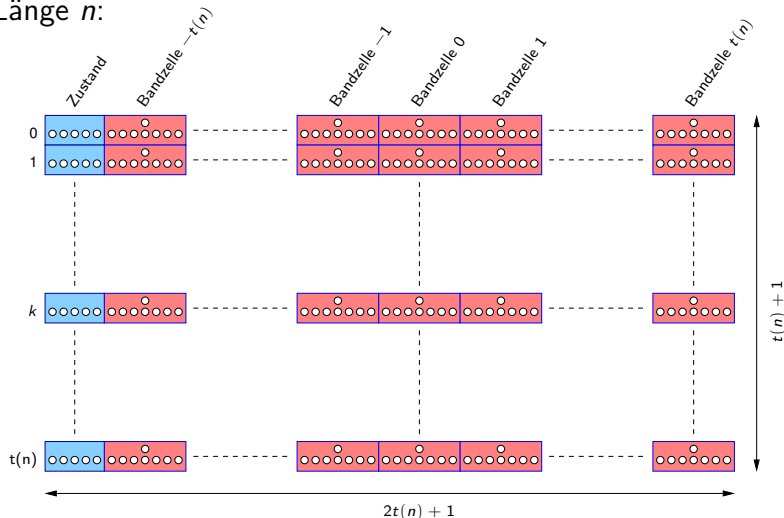
$$M = (Z, \Sigma, \Gamma, \sqcup, \delta, z_s, z_{acc}, z_{rej}),$$

die A akzeptiert und deren Laufzeit durch das Polynom $t(n)$ beschränkt ist

Zu zeigen: $A \leq_m^p \text{SAT}$

SAT ist NP-vollständig (Forts.)

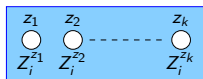
Idee: Konstruktion eines “Schaltkastens” für Eingaben der Länge n :



SAT ist NP-vollständig (Forts.)

Bemerkungen:

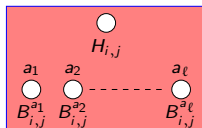
- Der Schaltkasten besteht aus einer Vielzahl Boolescher Variablen
- Jede Zeile des Schaltkastens steht für eine Konfiguration von M
- Zustandsbox: Zustandsmenge $Z = \{z_1, \dots, z_k\}$



$Z_i^z = 1 \rightsquigarrow$ Zustand der i -ten Konfiguration ist z

SAT ist NP-vollständig (Forts.)

- Bandzellenbox: Arbeitsalphabet $\Gamma = \{a_1, \dots, a_\ell\}$



- ▷ $H_{i,j} = 1 \rightsquigarrow$ in der i -ten Konfiguration befindet sich der S/L-Kopf an Position j
- ▷ $B_{i,j}^{a_m} = 1 \rightsquigarrow$ in der j -ten Bandzelle der i -ten Konfiguration ist der Buchstabe a_m gespeichert

Achtung: nicht jede Belegung der Variablen ist eine korrekte Konfigurationsfolge

Vorgehen: Überprüfung der Korrektheit mittels einer aussagenlogischen Formel

SAT ist NP-vollständig (Forts.)

Hilfsformel:

$$G(x_1, \dots, x_m) = \left(\bigvee_{i=1}^m x_i \right) \wedge \left(\bigwedge_{i=1}^{m-1} \bigwedge_{j=i+1}^m (\neg x_i \vee \neg x_j) \right)$$

Lemma. $G(x_1, \dots, x_m) = 1$ genau dann, wenn genau eine der Variablen $x_1, \dots, x_m$ mit 1 belegt ist.

Beweis.

- Der erste Teil der Formel wird wahr, wenn mindestens eine der Variablen mit 1 belegt ist
- Der zweite Teil der Formel wird wahr, wenn höchstens eine der Variablen mit 1 belegt ist

Da beide Teile $\wedge$ -verknüpft sind, wird G genau dann wahr, wenn genau eine der Variablen mit 1 belegt ist

SAT ist NP-vollständig (Forts.)

Zu überprüfen:

- Pro Konfiguration ist exakt ein Zustand ausgewählt
- Pro Konfiguration ist genau eine Kopfposition ausgewählt
- In jeder Bandzelle ist genau ein Buchstabe ausgewählt
- In der ersten Zeile steht die Startkonfiguration von M auf Eingabe x
- Die Konfiguration in Zeile j ist eine Folgekonfiguration der Konfiguration in Zeile $j - 1$
- Es gibt eine akzeptierende Endkonfiguration

SAT ist NP-vollständig (Forts.)

Konsistenz:

$$\begin{aligned}\text{STATE}(i) &= G(Z_i^{z_1}, \dots, Z_i^{z_k}) \\ \text{CELL}(i, j) &= G(B_{i,j}^{a_1}, \dots, B_{i,j}^{a_\ell}) \\ \text{HEAD}(i) &= G(H_{i,-t(n)}, \dots, H_{i,t(n)})\end{aligned}$$

Insgesamt:

CONSISTENCY =

$$\bigwedge_{i=0}^{t(n)} \left(\text{STATE}(i) \wedge \text{HEAD}(i) \wedge \bigwedge_{j=-t(n)}^{t(n)} \text{CELL}(i, j) \right)$$

SAT ist NP-vollständig (Forts.)

Startkonfiguration für die Eingabe $x = x_1 \dots x_n$:

$$\text{STARTCONF}(x_1 \dots x_n) = Z_0^{z_s} \wedge H_{0,0} \\ \wedge \left(\bigwedge_{j=-t(n)}^{-1} B_{0,j} \right) \wedge B_{0,0}^{x_1} \wedge \dots \wedge B_{0,n-1}^{x_n} \wedge \left(\bigwedge_{j=n}^{t(n)} B_{0,j} \right)$$

Akzeptierende Konfiguration:

$$\text{ACCEPT} = \left(\bigvee_{i=0}^{t(n)} Z_i^{z_{acc}} \right)$$

SAT ist NP-vollständig (Forts.)

Folgekonfigurationen:

Teil 1: Befindet sich der S/L-Kopf nicht auf der Bandzelle, dann muss der Inhalt mit dem der nächsten Konfiguration identisch sein

$$\text{PARTI}(i) =$$

$$\bigwedge_{j=-t(n)}^{t(n)} \left(\neg H_{ij} \rightarrow \left((B_{ij}^{a_1} \leftrightarrow B_{i+1,j}^{a_1}) \wedge \dots \wedge (B_{ij}^{a_\ell} \leftrightarrow B_{i+1,j}^{a_\ell}) \right) \right)$$

wobei

$$A \rightarrow B \equiv \neg A \vee B$$

$$A \leftrightarrow B \equiv (\neg A \vee B) \wedge (A \vee \neg B)$$

SAT ist NP-vollständig (Forts.)

Teil 2. Befindet sich der S/L-Kopf auf der Bandzelle, dann muss ein Rechenschritt ausgeführt werden

$$\text{PARTII}(i) = \bigwedge_{j=-t(n)}^{t(n)} \bigwedge_{z \in Z} \bigwedge_{a \in \Gamma} \text{TRANSITION}(i, j, z, a)$$

$\text{TRANSITION}(i, j, z, a)$ wird anhand $\delta(z, a)$ definiert

Fallunterscheidung:

- $\|\delta(z, a)\| > 0$
- $\|\delta(z, a)\| = 0$
- $z = z_{acc}$

SAT ist NP-vollständig (Forts.)

Fall 1: Angenommen, $\delta(z, a) = \{(z_1, b_1, D_1), \dots, (z_q, b_q, D_q)\}$,
wobei $b_r \in \Gamma$ und $D_r \in \{L, R, N\}$

$$\text{TRANSITION}(i, j, z, a) = (H_{i,j} \wedge Z_i^z \wedge B_{i,j}^a) \rightarrow$$
$$\left(\bigvee_{r=1}^q \left(H_{i+1, j+m(D_r)} \wedge Z_{i+1}^{z_r} \wedge B_{i+1, j+m(D_r)}^{b_r} \right) \right)$$

wobei

$$m(D) = \begin{cases} -1 & D = L \\ 0 & D = N \\ 1 & D = R \end{cases}$$

SAT ist NP-vollständig (Forts.)

Fall 2: Angenommen, $\delta(z, a) = \emptyset$

$$\text{TRANSITION}(i, j, z, a) = (H_{i,j} \wedge Z_i^z \wedge B_{i,j}^a) \rightarrow (H_{i,j} \wedge \neg H_{i,j})$$

In diesem Fall ist die Formel unerfüllbar

Fall 3: $z = z_{acc}$

$$\begin{aligned} \text{TRANSITION}(i, j, z_{acc}, a) = \\ (H_{i,j} \wedge Z_i^{z_{acc}} \wedge B_{i,j}^a) \rightarrow (H_{i+1,j} \wedge Z_{i+1}^{z_{acc}} \wedge B_{i+1,j}^a) \end{aligned}$$

Dies entspricht der Überföhrungsfunktion

$$\delta(z_{acc}, a) = \{(z_{acc}, a, N)\}$$

SAT ist NP-vollständig (Forts.)

Insgesamt:

$$\begin{aligned} \text{EVAL}(x) = & \\ & \text{CONSISTENCY} \wedge \text{STARTCONF}(x) \wedge \text{ACCEPT} \\ & \wedge \bigwedge_{i=0}^{t(n)} (\text{PARTI}(i) \wedge \text{PARTII}(i)) \end{aligned}$$

Zu zeigen:

- $\text{EVAL}(x)$ ist erfüllbar genau dann, wenn M die Eingabe x akzeptiert
- $\text{EVAL}(x)$ ist in Polynomialzeit aus x berechenbar

SAT ist NP-vollständig (Forts.)

Behauptung: $\text{EVAL}(x) \in \text{SAT}$ genau dann, wenn $x \in A$

“ $\Rightarrow$ ”: Angenommen, $\text{EVAL}(x)$ ist erfüllbar.

Dann liefert die Belegung der Variablen eine Konfigurationsfolge von M auf Eingabe x , die bei der Startkonfiguration beginnt und in einer akzeptierenden Konfiguration endet.

Folglich gibt es im Berechnungsbaum von M auf Eingabe x mindestens einen akzeptierenden Berechnungspfad, d.h., $x \in A$

“ $\Leftarrow$ ”: Angenommen, $x \in A$.

Dann gibt es eine akzeptierende Konfigurationsfolge von M auf Eingabe x . Anhand dieser Folge kann eine erfüllende Belegung für $\text{EVAL}(x)$ abgeleitet werden

Somit: $\text{EVAL}(x) \in \text{SAT}$

SAT ist NP-vollständig (Forts.)

Aufwand zur Berechnung von $\text{EVAL}(x)$ anhand von x , $|x| = n$:

- CONSISTENCY: $O(t(n)^2)$
- STARTCONF: $O(t(n))$
- ACCEPT: $O(t(n))$
- TRANSITION: $O(1)$
- PARTI: $O(t(n))$
- PARTII: $O(t(n))$

$\rightsquigarrow$ Gesamt: $O(t(n)^2)$

Somit: $A \leq_m^P \text{SAT}$

Korollar 1. KNF-SAT ist NP-vollständig

Beweis. Alle Teilformeln in $\text{EVAL}(x)$ kann man in äquivalente KNF-Formeln umformen. Somit gilt $A \leq_m^p \text{KNF-SAT}$ für alle $A \in \text{NP}$.

Korollar. 3-SAT ist NP-vollständig

Beweis. $\text{KNF-SAT} \leq_m^p \text{3-SAT}$

- NP enthält eine Vielzahl von Problemen, die in der Praxis häufig vorkommen
- Jedes Problem in NP ist effizient verifizierbar
- Es ist nicht bekannt, ob alle Sprachen in NP effizient entscheidbar sind, d.h., ob $P = NP$
- Die NP-vollständigen Probleme sind die schwierigsten Probleme in NP
- Falls ein NP-vollständiges Problem in P ist, dann ist $P = NP$