Informatik IV
Theoretische Informatik

Kapitel 12
Probleme in P und NP

Sommersemester 2019

Dozent: Prof. Dr. J. Rothe

Einige Grundlagen der Komplexitätstheorie

Ziel: NP-Vollständigkeit als ressourcenbeschränktes Analagon zur RE-Vollständigkeit.

- Komplexitätstheorie untersucht den Ressourcenbedarf (insbesondere: Rechenzeit, Speicherplatz) von Algorithmen.
- Dabei sind sowohl untere als auch obere Schranken für die Ressourcen von Interesse.
- Die zwei wichtigsten Zeitkomplexitätsklassen, P und NP, wollen wir nun definieren.

Einige Grundlagen der Komplexitätstheorie

Definition ($\mathbb{P}\mathrm{ol}$)

Sei $\mathbb{P}\mathrm{ol}$ die Menge aller Polynome der Form

$$p(n) = a_m n^m + a_{m-1} n^{m-1} + \ldots + a_1 n + a_0,$$

wobei $n, m, a_i \in \mathbb{N}$ für $1 \leq i \leq m$.

Einige Grundlagen der Komplexitätstheorie

Definition (deterministische Polynomialzeit: P)

Sei $t : \mathbb{N} \to \mathbb{N}$ eine Funktion und M eine deterministische Turingmaschine mit $L(M) \subseteq \Sigma^*$ für ein Alphabet Σ. Definiere

- die Funktion $\text{time}_M : \Sigma^* \to \mathbb{N}$ durch

$$\text{time}_M(w) = \begin{cases} \text{Zahl der Takte von } M(w) & \text{falls } M \text{ bei Eingabe } w \text{ hält} \\ \text{nicht definiert} & \text{sonst;} \end{cases}$$

- die Funktion $\text{Time}_M : \mathbb{N} \to \mathbb{N}$ durch

$$\text{Time}_M(n) = \begin{cases} \max_{w:|w|=n} \text{time}_M(w) & \text{falls } \text{time}_M(w) \text{ für alle } w \text{ mit} \\ & |w| = n \text{ definiert ist} \\ \text{nicht definiert} & \text{sonst;} \end{cases}$$

Deterministische Polynomialzeit

Definition (deterministische Polynomialzeit: P)

- die Komplexitätsklasse

$$\mathrm{TIME}(t) = \left\{ A \;\middle|\; \begin{array}{l} \text{es gibt eine deterministische TM } M \text{ mit} \\ L(M) = A \text{ und } \mathrm{Time}_M(n) \leq t(n) \text{ für alle } n \end{array} \right\}.$$

- Die Komplexitätsklasse *deterministische Polynomialzeit* ist definiert durch

$$\mathrm{P} = \bigcup_{p \in \mathbb{P}\mathrm{ol}} \mathrm{TIME}(p).$$

Deterministische Polynomialzeit

Bemerkung:

- Die Klasse P ist die Klasse der durch effiziente (polynomialzeitbeschränkte) Algorithmen lösbaren Probleme.
- Um zu zeigen, dass ein Algorithmus eine polynomielle Laufzeit hat, genügt es offensichtlich zu zeigen, dass seine Komplexität in $\mathcal{O}(n^k)$, für eine Konstante $k \in \mathbb{N}$, liegt.
- Damit ist auch jeder Algorithmus mit der Laufzeit $\log n$ und $n \log n$ polynomiell, da $\log n \in \mathcal{O}(n)$ und $n \log n \in \mathcal{O}(n^2)$.
- Laufzeiten wie n^n, $n^{\log n}$, c^n, $c \geq 2, c \in \mathbb{N}$, sind dagegen nicht polynomiell, sondern exponentiell.

Das Erfüllbarkeitsproblem der Aussagenlogik: SAT

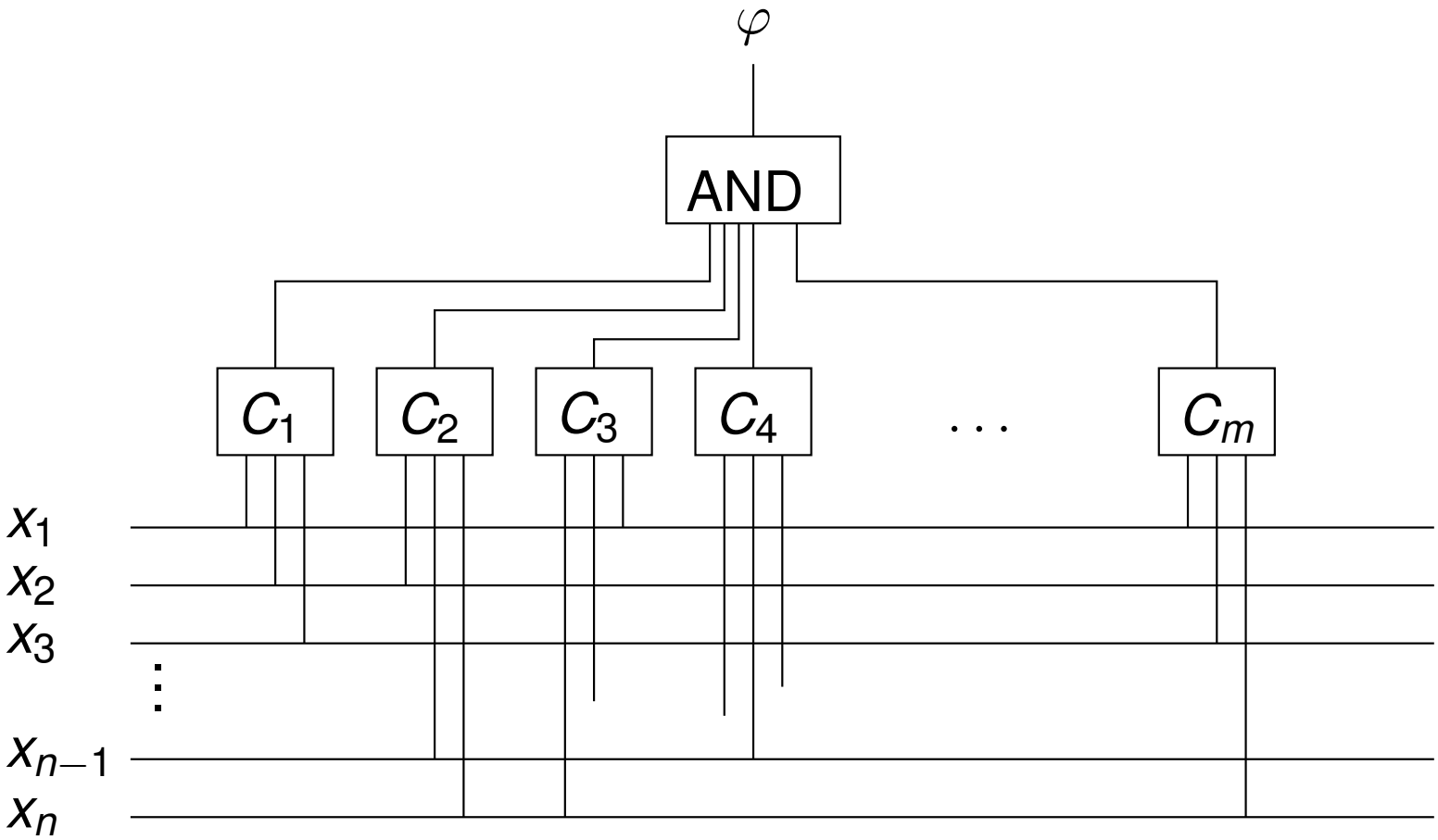

Abbildung: Eine 3-SAT-Instanz, repräsentiert durch einen Booleschen Schaltkreis

Das Erfüllbarkeitsproblem der Aussagenlogik: SAT

- SAT ist in vielen Anwendungen wichtig, es spielt eine zentrale Rolle im Zusammenhang mit *„constraint satisfaction“*, in der Theorie der Schaltkreise usw.

- So genannte SAT-Solver werden bei vielen praktischen Aufgaben als Subroutinen verwendet.

- Boolesche Formeln (oder Boolesche Ausdrücke) werden syntaktisch und semantisch wie folgt definiert.

Boolesche Ausdrücke, KNF, DNF

Definition

- Die Boolesche Variablen $x_1, x_2, \ldots$ können Werte in $\{0, 1\}$ annehmen. Die Menge aller Booleschen Variablen bezeichen wir mit X, d.h., $X = \{x_1, x_2, \ldots\}$.
- **Syntax:** Die *Menge der Booleschen Ausdrücke* wird induktiv definiert durch:
  - Jedes $x \in X$ ist ein (atomarer) Boolescher Ausdruck.
  - Sind F und G Boolesche Ausdrücke, so sind auch
    - die Negation $\neg F$,
    - die Konjunktion $F \wedge G$ und
    - die Disjunktion $F \vee G$

    Boolesche Ausdrücke.
  - Nichts sonst ist ein Boolescher Ausdruck.

Boolesche Ausdrücke, KNF, DNF

Bemerkung: Sämtliche Booleschen Operationen lassen sich durch die Negation, die Konjunktion und die Disjunktion ausdrücken.
Beispielsweise ergibt sich die Implikation $\alpha \Rightarrow \beta$ durch $\neg\alpha \vee \beta$.

Definition

- **Semantik:** Eine *Belegung* (mit Wahrheitswerten) ist eine Abbildung $\beta : X \to \{0, 1\}$.

  Für nicht atomare Boolesche Ausdrücke H wird der Wahrheitswert $\beta(H)$ induktiv wie folgt definiert:

  - Ist $H = \neg F$, so ist $\beta(H) = 1$, falls $\beta(F) = 0$, und sonst ist $\beta(H) = 0$.
  - Ist $H = F \wedge G$, so ist $\beta(H) = 1$, falls $\beta(F) = 1$ und $\beta(G) = 1$, und sonst ist $\beta(H) = 0$.
  - Ist $H = F \vee G$, so ist $\beta(H) = 1$, falls $\beta(F) = 1$ oder $\beta(G) = 1$, und sonst ist $\beta(H) = 0$.

Boolesche Ausdrücke, KNF, DNF

Definition

- Ein Boolescher Ausdruck H heißt *erfüllbar*, falls es eine wahrmachende Belegung für ihn gibt, d.h., falls $\beta(H) = 1$ für eine geeignete Belegung $\beta : X \to \{0, 1\}$ gilt.
- Ein *Literal* ist eine Variable oder ihre Negation.
- Ein Boolescher Ausdruck H ist in *konjunktiver Normalform* (kurz: KNF), falls er Konjunktion von Disjunktionen von Literalen ist:

$$H = \bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{n} L_{ij} \right) .$$

Die Disjunktionen $\left(\bigvee_{j=1}^{n} L_{ij} \right)$ heißen *Klauseln*.

Boolesche Ausdrücke, KNF, DNF

Definition

- Ein Boolescher Ausdruck H ist in *disjunktiver Normalform* (kurz: DNF), falls er Disjunktion von Konjunktionen von Literalen ist:

$$H = \bigvee_{i=1}^{m} \left(\bigwedge_{j=1}^{n} L_{ij} \right) .$$

  Die Konjunktionen $\left(\bigwedge_{j=1}^{n} L_{ij} \right)$ heißen *Monome*.

- Sei $k \geq 0$. Ein Boolescher Ausdruck H ist in *k-KNF* (bzw. in *k-DNF*), falls H in KNF (bzw. in DNF) ist und jede Klausel von H höchstens k Literale enthält.

- Ein Boolescher Ausdruck H heißt *Hornformel*, falls H in KNF ist und jede Klausel von H höchstens ein positives Literal enthält.

Boolesche Ausdrücke, KNF, DNF

Beispiel:

- Atomare Boolesche Ausdrücke: x_1, x_2, x_3, x_4
- Boolesche Ausdrücke: $\neg x_3, \quad x_2 \wedge x_3, \quad (x_2 \vee x_4) \wedge \neg x_3, \quad x_2 \vee \neg x_2, \quad x_3 \wedge \neg x_3$
- erfüllbare Boolesche Ausdrücke:
  - $\neg x_3$ mit $\beta(x_3) = 0$,
  - $x_2 \wedge x_3$ mit $\beta(x_2) = \beta(x_3) = 1$
  - nicht erfüllbar ist der Boolesche Ausdruck $x_3 \wedge \neg x_3$
- Literale: x_3, x_1, $\neg x_3$

Boolesche Ausdrücke, KNF, DNF

Beispiel:

- ein Boolescher Ausdruck in KNF:
  $(\neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$
- ein Boolescher Ausdruck in DNF:
  $(x_3 \wedge \neg x_3) \vee (x_1 \wedge \neg x_2 \wedge x_3) \vee (x_2 \wedge x_3 \wedge \neg x_4)$
- ein Boolescher Ausdruck in 4-KNF:
  $(\neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3 \vee \neg x_4)$
- ein Boolescher Ausdruck in 3-DNF:
  $(x_3 \wedge \neg x_3) \vee (x_1 \wedge \neg x_2 \wedge x_3) \vee (x_2 \wedge x_3 \wedge \neg x_4)$
- Hornformel: $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee \neg x_3 \vee \neg x_4)$

Varianten des Erfüllbarkeitsproblems

Definition

- $\text{SAT} = \{H \mid H \text{ ist erfüllbarer Boolescher Ausdruck}\}$.
- $\text{KNF-SAT} = \{H \mid H \text{ ist erfüllbarer Boolescher Ausdruck in KNF}\}$.
- $\text{DNF-SAT} = \{H \mid H \text{ ist erfüllbarer Boolescher Ausdruck in DNF}\}$.
- $k\text{-SAT} = \{H \mid H \text{ ist erfüllbarer Boolescher Ausdruck in } k\text{-KNF}\}$, wobei $k \geq 0$.
- $\text{Horn-SAT} = \{H \mid H \text{ ist erfüllbare Hornformel}\}$.

Varianten des Erfüllbarkeitsproblems

Theorem

DNF-SAT, 2-SAT *und* Horn-SAT *sind in* P.

Beweisidee: Insbesondere wird beim Beweis von $2\text{-SAT} \in \text{P}$ zu einem gegebenen Booleschen Ausdruck H in 2-KNF ein gerichteter Graph G_H wie folgt konstruiert:

(a) Knoten von G_H sind alle Variablen von H und ihre Negationen;

(b) Kanten von G_H: (α, γ) ist genau dann Kante von α zu γ, wenn es in H eine Klausel der Form $(\neg\alpha \vee \gamma)$ (bzw. $(\gamma \vee \neg\alpha)$) gibt.

Varianten des Erfüllbarkeitsproblems

Lemma

Eine Boolesche Formel H in 2-KNF ist genau dann unerfüllbar, wenn es in G_H einen Knoten x gibt, so dass es in G_H einen Pfad von x zu $\neg x$ und einen Pfad von $\neg x$ zu x gibt. **ohne Beweis**

Der Polynomialzeit-Algorithmus für 2-SAT ergibt sich dann aus diesem Lemma. ❑

Nichtdeterministische Polynomialzeit

Nun betrachten wir *nichtdeterministische* polynomialzeitbeschränkte Turingmaschinen.

Definition (nichtdeterministische Polynomialzeit: NP)

Sei $t : \mathbb{N} \to \mathbb{N}$ eine Funktion und M eine nichtdeterministische TM mit $L(M) \subseteq \Sigma^*$ für ein Alphabet Σ. Definiere

- die Funktion $\text{ntime}_M : \Sigma^* \to \mathbb{N}$ durch

$$\text{ntime}_M(w) = \begin{cases} \min[\text{Zahl der Konfigurationen in einer} & \text{falls } w \in L(M) \\ \quad \textit{akzeptierenden} \text{ Rechnung von } M(w)] & \\ \text{nicht definiert} & \text{sonst;} \end{cases}$$

Nichtdeterministische Polynomialzeit

Definition (nichtdeterministische Polynomialzeit: NP)

- die Funktion $\text{Ntime}_M : \mathbb{N} \to \mathbb{N}$ durch

$$\text{Ntime}_M(n) = \begin{cases} \max_{w:|w|=n} \text{ntime}_M(w) & \text{falls } \text{ntime}_M(w) \text{ für alle } w \text{ mit} \\ & |w| = n \text{ definiert ist} \\ \text{nicht definiert} & \text{sonst;} \end{cases}$$

- die Komplexitätsklasse

$$\text{NTIME}(t) = \left\{ A \;\middle|\; \begin{array}{l} \text{es gibt eine nichtdeterministische TM } M \text{ mit} \\ L(M) = A \text{ und } \text{Ntime}_M(n) \leq t(n) \text{ für alle } n \end{array} \right\};$$

Nichtdeterministische Polynomialzeit

Definition (nichtdeterministische Polynomialzeit: NP)

- die Komplexitätsklasse *nichtdeterministische Polynomialzeit*:

$$\mathrm{NP} = \bigcup_{p \in \mathbb{P}\mathrm{ol}} \mathrm{NTIME}(p).$$

Bemerkung:

- Offensichtlich gilt:

$$\mathrm{P} \subseteq \mathrm{NP}.$$

- Ob die Umkehrung auch gilt, ist ein berühmtes offenes Problem, das so genannte „P-NP-Problem“.
- Die allgemeine Vermutung ist, dass $\mathrm{P} \neq \mathrm{NP}$ gilt, aber ein Beweis dieser Vermutung steht noch aus (scheint sehr schwierig zu sein).

Nichtdeterministische Polynomialzeit

- Das Verhältnis von P zu NP ist ähnlich dem Verhältnis von REC zu RE; dies spiegelt sich in einer Vielzahl von analogen Ergebnissen wider.
- Beispielsweise gilt für NP eine „polynomialzeitbeschränkte“ Version des Projektionssatzes.

Theorem

$A \in \mathrm{NP} \iff$ *es gibt eine Menge* $B \in \mathrm{P}$ *und ein Polynom* $p \in \mathbb{P}\mathrm{ol}$, *so dass für alle* $x \in \Sigma^*$*:*

$$x \in A \iff (\exists y \in \Sigma^*)\,[|y| \leq p(|x|) \ \wedge\ (x, y) \in B].$$

ohne Beweis

SAT ist in NP

- Die Klasse NP enthält eine Vielzahl wichtiger Probleme.
- Eines davon ist das uneingeschränkte Erfüllbarkeitsproblem.

Lemma

SAT *ist in* NP.

Beweis: Betrachte die folgende nichtdeterministische Turingmaschine M, die SAT wie folgt in Polynomialzeit akzeptiert:

SAT ist in NP

- Eingabe: Eine Boolesche Formel $\varphi(x_1, x_2, \ldots, x_n)$.
- Rate nichtdeterministisch eine Belegung t der Variablen $x_1, x_2, \ldots, x_n$ mit Wahrheitswerten. („Ratephase“ bzw. „Auswahlphase“)

  Es existieren 2^n mögliche nichtdeterministische unabhängige Rechnungen – für jede Belegung eine.
- Verifiziere auf jedem solchen Berechnungspfad, ob die dort geratene Belegung t die Formel φ erfüllt. („Verifikationsphase“)
- Akzeptiere genau dann, wenn dies der Fall ist (d.h., genau dann, wenn $t(\varphi) = 1$).

Offenbar gilt $L(M) = \text{SAT}$. Da M in Polynomialzeit arbeitet, folgt $\text{SAT} \in \text{NP}$. ❑

SAT ist in NP

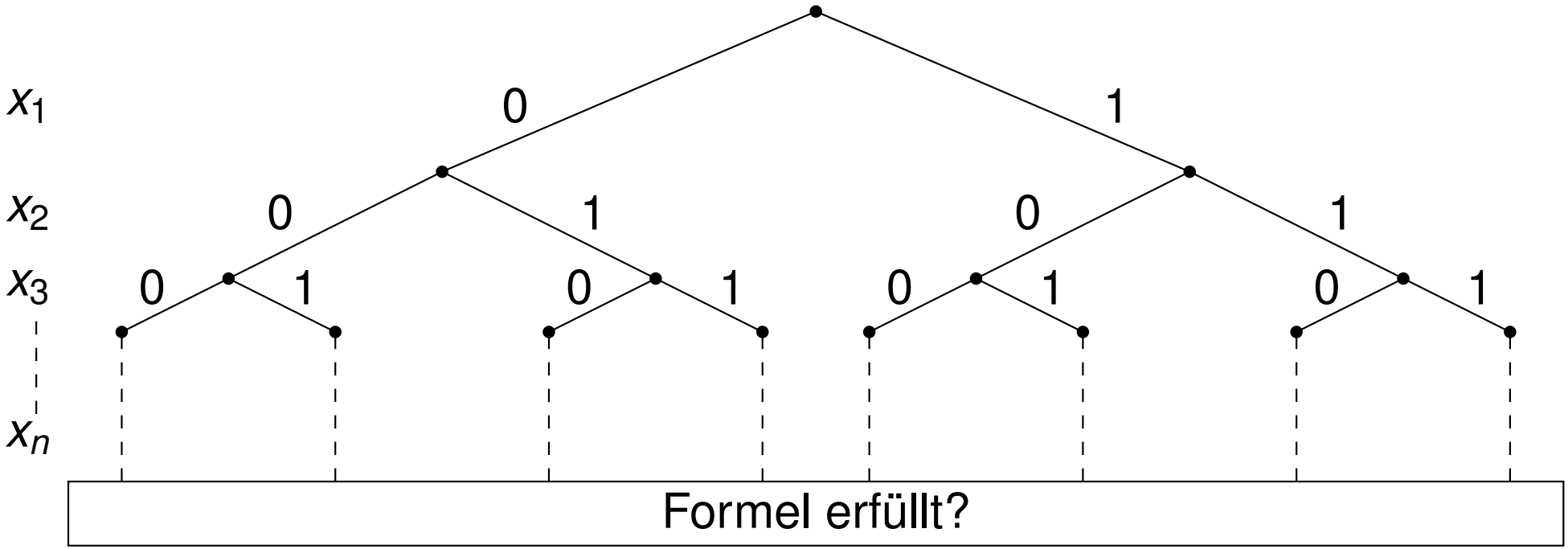

Abbildung: Berechnungsbaum eines nichtdeterministischen Algorithmus für SAT