Grundlagen der Theoretischen Informatik

Komplexitätstheorie (II)

11.07.2018 und 12.07.2018

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

Übersicht

1. Motivation
2. Terminologie
3. Endliche Automaten und reguläre Sprachen
4. Kellerautomaten und kontextfreie Sprachen
5. Turingmaschinen und rekursiv aufzählbare Sprachen
6. Berechenbarkeit, (Un-)Entscheidbarkeit
7. Komplexitätsklassen P und NP

Komplexitätstheorie

Inhalt

- Definition der berühmten Klassen P und NP .
- Begriff der **Reduktion**: ein Problem (eine Sprache) wird auf ein zweites reduziert. Das erste Problem ist dann höchstens so schwer wie das zweite.
- Der Begriff eines NP **-schweren** Problems.
- Einige Probleme der Graphentheorie: sie sind **NP-vollständig**.
- Die wichtigsten **Komplexitätsklassen** und ihre Struktur.

Komplexitätstheorie

Welche Arten von Komplexität gibt es?

- **Zeit**
- **Speicher**

DTIME und NTIME

Definition [NTIME(T(n)), DTIME(T(n))]

Basismodell: k-DTM $\mathcal{M}$ (ein Band für die Eingabe).

Wenn $\mathcal{M}$ mit jedem Eingabewort der Länge n höchstens $T(n)$ Schritte macht, dann wird sie $T(n)$-**zeitbeschränkt** genannt.
Die von $\mathcal{M}$ akzeptierte Sprache hat **Zeitkomplexität** $T(n)$
(tatsächlich meinen wir $\max(n+1, \lceil T(n) \rceil)$).

- **DTIME**(T(n)) ist die Klasse der Sprachen, die von $T(n)$-zeitbeschränkten DTMs akzeptiert werden.
- **NTIME**(T(n)) ist die Klasse der Sprachen, die von $T(n)$-zeitbeschränkten NTMs akzeptiert werden.

DSPACE und NSPACE

Definition [NSPACE(S(n)), DSPACE(S(n))]

Basismodell: k-DTM $\mathcal{M}$, davon ein spezielles Eingabeband (**offline DTM**).

Wenn $\mathcal{M}$ für jedes Eingabewort der Länge n maximal $S(n)$ Zellen auf den Ablagebändern benutzt, dann heißt $\mathcal{M}$ $S(n)$-**speicherbeschränkt.**
Die von $\mathcal{M}$ akzeptierte Sprache hat **Speicherkomplexität** $S(n)$
(tatsächlich meinen wir $\max(1, \lceil S(n) \rceil)$)

- **DSPACE**(S(n)) ist die Klasse der Sprachen, die von $S(n)$-speicherbeschränkten DTMs akzeptiert werden.
- **NSPACE**(S(n)) ist die Klasse der Sprachen, die von $S(n)$-speicherbeschränkten NTMs akzeptiert werden.

Wieso eine *offline*-Turing-Maschine?

Bandbeschränkung von weniger als linearem Wachstum.

Wachstumsrate von DTIME und DSPACE

Definition [P, NP, PSPACE]

$$\begin{aligned} \mathbf{P} &:= \bigcup_{i \geqslant 1} \mathbf{DTIME}(n^i) \\ \mathbf{NP} &:= \bigcup_{i \geqslant 1} \mathbf{NTIME}(n^i) \\ \mathbf{PSPACE} &:= \bigcup_{i \geqslant 1} \mathbf{DSPACE}(n^i) \end{aligned}$$

Wachstumsrate von DTIME und DSPACE

Definition [P, NP, PSPACE]

$$\begin{aligned} \mathbf{P} &:= \bigcup_{i \geqslant 1} \mathbf{DTIME}(n^i) \\ \mathbf{NP} &:= \bigcup_{i \geqslant 1} \mathbf{NTIME}(n^i) \\ \mathbf{PSPACE} &:= \bigcup_{i \geqslant 1} \mathbf{DSPACE}(n^i) \end{aligned}$$

Intuitiv

- Probleme in **P** sind effizient lösbar, jene aus **NP** können in exponentieller Zeit gelöst werden.
- **PSPACE** ist eine sehr große Klasse, weit größer als **P** oder **NP** .

Komplexitätsklassen für Funktionen

Komplexitätsklassen für Funktionen

Eine Funktion $f : \mathbb{N} \to \mathbb{N}$ ist in **P**, falls es
eine DTM $\mathcal{M}$ und ein Polynom $p(n)$ gibt, so dass
für jedes n der Funktionswert $f(n)$ in höchstens
$p(\text{länge}(n))$ Schritten von $\mathcal{M}$ berechnet wird.

Dabei gilt $\text{länge}(n) = \lg n$, denn man braucht $\lg n$ Zeichen,
um die Zahl n binär darzustellen.

Analog funktioniert dies für alle anderen Komplexitätsklassen.

Beziehungen zwischen den Komplexitätsklassen

Frage:

Was sind die genauen Beziehungen zwischen den Komplexitätsklassen P, NP, PSPACE?

Beziehungen zwischen den Komplexitätsklassen

Frage:

Was sind die genauen Beziehungen zwischen den Komplexitätsklassen P, NP, PSPACE?

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE}$$

Beziehungen zwischen den Komplexitätsklassen

Frage:

Wie zeigen wir, dass ein gegebenes Problem in einer bestimmten Klasse ist?

Antwort

Reduktion auf ein bekanntes!

Wir brauchen eines, mit dem wir anfangen können: SAT

Komplexitätsklassen

Frage:

Können wir in **NP** Probleme finden, die **die schwierigsten in NP** sind?

Komplexitätsklassen

Frage:

Können wir in **NP** Probleme finden, die **die schwierigsten in NP** sind?

Antwort

Es gibt mehrere Wege, ein schwerstes Problem zu definieren. Sie hängen davon ab, welchen **Begriff von Reduzierbarkeit** wir benutzen.

Für einen gegebenen Begriff von Reduzierbarkeit ist die Antwort: **Ja**.

Solche Probleme werden **vollständig in der gegebenen Klasse** bezüglich des Begriffs der Reduzierbarkeit genannt.

Reduktion

Definition (Polynomial-Zeit-Reduzibilität)

Seien L_1, L_2 Sprachen.

L_1 ist Polynomial-Zeit reduzibel auf L_2, bezeichnet mit $L_1 \preceq_{\text{pol}} L_2$,
wenn es eine **Polynomial-Zeit beschränkte DTM** gibt,
die für jede Eingabe w eine Ausgabe $f(w)$ erzeugt, so dass

$$w \in L_1 \text{ gdw } f(w) \in L_2$$

Reduktion

Lemma [Polynomial-Zeit-Reduktionen]

1. Sei L_1 Polynomial-Zeit-reduzibel auf L_2 ($L_1 \preceq_{\text{pol}} L_2$). Dann gilt

   Wenn L_2 in **NP** ist dann ist auch L_1 in **NP**

   Wenn L_2 in **P** ist dann ist auch L_1 in **P**

2. Die Komposition zweier Polynomial-Zeit-Reduktionen ist wieder eine Polynomial-Zeit-Reduktion.

NP

Theorem.
Eine Sprache L is in **NP** genau dann wenn es eine Sprache L' in **P** und ein $k \geq 0$ gibt, so das für alle $w \in \Sigma$ gilt:

$$w \in L \text{ gdw. es gibt ein } c : \langle w, c \rangle \in L' \text{ und } |c| < |w|^k.$$

c wird **Zeuge** (*witness* oder Zertifikat/*certificate*) von w in L genannt.

Eine DTM, die die Sprache L' akzeptiert, wird **Prüfer** (*verifier*) von L genannt.

Wichtig:
Ein Entscheidungsproblem ist in **NP** genau dann wenn **jede Ja-Instanz ein kurzes Zertifikat** hat (d.h. seine Länge polynomial in der Länge der Eingabe ist), welche in polynomial-Zeit verifiziert werden kann.

Vollständige und harte Probleme

Vollständige und harte Probleme

Definition [NP-vollständig, NP-hart]

- Eine Sprache L heißt **NP-hart (NP-schwer)** wenn
  jede Sprache $L' \in \mathbf{NP}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **NP-vollständig** wenn sie
  1. in **NP** ist ($L \in \mathbf{NP}$), und
  2. **NP**-hart ist

Vollständige und harte Probleme

Definition [NP-vollständig, NP-hart]

- Eine Sprache L heißt **NP-hart (NP-schwer)** wenn
  jede Sprache $L' \in \mathbf{NP}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **NP-vollständig** wenn sie
  1. in **NP** ist ($L \in \mathbf{NP}$), und
  2. **NP**-hart ist

Definition [PSPACE-vollständig, PSPACE-hart]

- Eine Sprache L heißt **PSPACE-hart (PSPACE-schwer)** wenn
  jede Sprache $L' \in \mathbf{PSPACE}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **PSPACE-vollständig** wenn sie
  1. in **PSPACE** ist ($L \in \mathbf{PSPACE}$) und
  2. **PSPACE**-hart ist

Vollständige und harte Probleme

Bemerkenswert

- Wenn gezeigt werden kann, dass auch nur ein einziges **NP**-hartes Problem in **P** liegt, dann ist $\mathbf{P} = \mathbf{NP}$.
- Wenn $\mathbf{P} \neq \mathbf{NP}$ gilt, dann ist kein einziges **NP**-vollständiges Problem in polynomieller Zeit lösbar.

Vollständige und harte Probleme

Bemerkenswert

- Wenn gezeigt werden kann, dass auch nur ein einziges **NP**-hartes Problem in **P** liegt, dann ist $\mathbf{P} = \mathbf{NP}$.
- Wenn $\mathbf{P} \neq \mathbf{NP}$ gilt, dann ist kein einziges **NP**-vollständiges Problem in polynomieller Zeit lösbar.

Eine Million Euro für den, der das „P = NP“-Problem löst!
(Millenium Probleme)

Vollständige und harte Problem

Wie zeigt man NP-Vollständigkeit?

Um zu zeigen, dass eine Sprache L **NP**-vollständig ist:

- Zeige, dass $L \in \mathbf{NP}$
- Finde bekanntermaßen **NP**-vollständige Sprache L' und
- reduziere sie auf L:

$$L' \preceq_{\text{pol}} L$$

Das genügt, da jede Sprache aus **NP** auf L' reduzierbar ist und wegen $L' \preceq_{\text{pol}} L$ dann auch auf L.

Hierfür häufig verwendet:
SAT-Problem, d.h.

$$L' = L_{\text{sat}} = SAT = \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel}\}$$

Cook's Theorem

Theorem $SAT = \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel}\}$ ist NP-vollständig.

Cook's Theorem

Theorem $SAT = \{w \mid w \text{ is a satisfiable formula of propositional logic}\}$ ist NP-vollständig.

Beweis: (Idee)

Zu zeigen: (1) $SAT \in NP$

(2) für alle $L \in NP$, $L \preceq_{\text{pol}} SAT$

Cook's Theorem

Theorem $SAT = \{w \mid w \text{ is a satisfiable formula of propositional logic}\}$ ist NP-vollständig.

Beweis: (Idee)

Zu zeigen: (1) $SAT \in NP$

(2) für alle $L \in NP$, $L \preceq_{\text{pol}} SAT$

(1) Gegeben sei F. Man kann in polynomieller Zeit bestimmen, ob F eine aussagenlogische Formel ist. Falls F aussagenlogische Formel: Wertebelegung $\mathcal{A}$ "raten", in polynomieller Zeit zeigen, dass $\mathcal{A}(F) = 1$.

Cook's Theorem

Theorem $SAT = \{w \mid w \text{ is a satisfiable formula of propositional logic}\}$ ist NP-vollständig.

Beweis: (Idee)

Zu zeigen: (1) $SAT \in NP$

(2) für alle $L \in NP$, $L \preceq_{\text{pol}} SAT$

(1) Gegeben sei F. Man kann in polynomieller Zeit bestimmen, ob F eine aussagenlogische Formel ist. Falls F aussagenlogische Formel: Wertebelegung $\mathcal{A}$ "raten", in polynomieller Zeit zeigen, dass $\mathcal{A}(F) = 1$.

(2) Sei $L \in NP$. Dann existiert eine polynomiell zeitgebundene NTM $\mathcal{M}$, mit $L(\mathcal{M}) = L$. Für $\mathcal{M}$ und w kann man eine aussagenlogische Sprache definieren und in polynomieller Zeit eine Formel $T_{M,w}$ finden, so dass

$$w \in L(\mathcal{M}) \quad \text{gdw.} \quad T_{M,w} \text{ erfüllbar ist.}$$

Stephen Cook

Stephen Arthur Cook (geboren 1939)

- Einer der bedeutendsten Forschern
  in der Komplexitätstheorie.
- 1971 'The Complexity of Theorem Proving Procedures'
  - formalisiert die Polynomialzeitreduktion
  - begründet mit dem Satz von Cook das
    Problem der NP-Vollständigkeit
    und im Besonderen das P-NP-Problem.
- Professor der Informatik an der
  University of Toronto in Kanada.
- 1982: Turing award

Vollständige und harte Probleme

Nota Bene: Es gibt NP-harte Probleme, die nicht in NP sind
(und z.B. sogar nicht entscheidbar sein können).

Beispiel: $SAT \preceq_{\text{pol}} SAT_{\text{PL}}$, wobei:

$$\begin{aligned} SAT &= \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel}\} \\ SAT_{\text{PL}} &= \{w \mid w \text{ ist eine erfüllbare Formel in der Prädikatenlogik}\} \end{aligned}$$

Sei $f : \Sigma^* \to \Sigma^*$ mit $\begin{cases} f(w) = w & \text{falls } w \text{ eine aussagenlogische Formel ist} \\ f(w) = \varepsilon \text{ sonst} \end{cases}$

(kann von einer polynomial-Zeit beschränkte DTM berechnet werden).

Dann: $F \in SAT$ gdw. F ist eine erfüllbare aussagenlogische Formel
gdw. $f(F) = F$ ist eine erfüllbare Formel in der Prädikatenlogik
gdw. $f(F) \in SAT_{\text{PL}}$

- SAT ist NP-vollständig, also ist SAT_{PL} NP-hart.
- $SAT_{\text{PL}} \notin$ NP. (SAT_{PL} ist nicht entscheidbar: die Menge aller erfüllbaren prädikatenlogischen Formeln ist nicht entscheidbar).

Abgschlossenheit der Komplexitätsklassen

Abgschlossenheit der Komplexitätsklassen

P, PSPACE sind abgeschlossen unter Komplement

Alle Komplexitätsklassen, die mittels **deterministischer Turing-Maschinen** definiert sind, sind **abgeschlossen unter Komplement-Bildung**

Denn:
Wenn eine Sprache L dazu gehört, dann auch ihr Komplement
(einfach die alte Maschine ausführen und die Ausgabe invertieren).

Abgschlossenheit der Komplexitätsklassen

Abgeschlossenheit von NP unter Komplement

Frage:

Ist **NP** abgeschlossen unter Komplementbildung?

Abgschlossenheit der Komplexitätsklassen

Abgeschlossenheit von NP unter Komplement

Frage:

Ist **NP** abgeschlossen unter Komplementbildung?

Antwort:

Keiner weiß es!

Die Kompexitätsklasse co-NP

Definition [co-NP]

co-NP ist die Klasse dr Sprachen deren Komplemente in **NP** liegen:

$$\textbf{co-NP} = \{L \mid \overline{L} \in \textbf{NP}\}$$

Beziehungen zwischen Komplexitätsklassen

Die folgenden Beziehungen sind momentan noch unbekannt

1. **P** $=?$ **NP**.
2. **NP** $=?$ **co-NP**.
3. **P** $=?$ **PSPACE**.
4. **NP** $=?$ **PSPACE**.

Beispiele

NP-vollständige Probleme

- Ist eine logische Formel erfüllbar? (**Satisfiability**)
- Ist ein (un-) gerichteter Graph hamiltonsch? (**Hamiltonian circle**)
- Gibt es in einem Graphen eine Clique der Größe k? (**Clique of size** k)
- Ist ein Graph mit drei Farben zu färben? (3-**colorability**)
- Gibt es in einer Menge von ganzen Zahlen eine Teilmenge mit der Gesamtsumme x? (**Subset Sum**)

NP-vollständige Probleme

- Ist eine logische Formel erfüllbar? (**Satisfiability**)
- Ist ein (un-) gerichteter Graph hamiltonsch? (**Hamiltonian circle**)
- Gibt es in einem Graphen eine Clique der Größe k? (**Clique of size** k)
- Ist ein Graph mit drei Farben zu färben? (3-**colorability**)
- Gibt es in einer Menge von ganzen Zahlen eine Teilmenge mit der Gesamtsumme x? (**Subset Sum**)

NP-vollständige Probleme

Definition[CNF, DNF]

DNF: Eine Formel ist in **disjunktiver Normalform**, wenn sie von folgender Form ist:

$$(l_{11} \wedge \ldots \wedge l_{1n_1}) \vee \ldots \vee (l_{m1} \wedge \ldots \wedge l_{mn_m})$$

CNF: Eine Formel ist in **konjunktiver Normalform**, wenn sie von folgender Form ist:

$$(l_{11} \vee \ldots \vee l_{1n_1}) \wedge \ldots \wedge (l_{m1} \vee \ldots \vee l_{mn_m})$$

.....

NP-vollständige Probleme

Definition[*k*-CNF, *k*-DNF]

k*-DNF:** Eine Formel ist in ***k-*DNF* wenn sie in DNF ist und jede ihrer Konjuntionen genau ***k*** Literale hat.

k*-CNF:** Eine Formel ist in ***k-*CNF* wenn sie in CNF ist und jede ihrer Disjunktion genau ***k*** Literale hat.

NP-vollständige Probleme

Definition[SAT, CNF-SAT, k-CNF-SAT]

- SAT $= L_{\text{sat}} = \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel}\}$
- CNF-SAT $= \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel in CNF}\}$
- $\boldsymbol{k}$-CNF-SAT$= \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel in } k\text{-CNF}\}$

Definition[DNF-SAT, k-DNF-SAT]

- DNF-SAT $= \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel in DNF}\}$
- $\boldsymbol{k}$-DNF-SAT$= \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel in } k\text{-DNF}\}$

NP-vollständige Probleme

Theorem [NP-vollständige Probleme]

Die folgenden Probleme liegen in **NP** und sind **NP**-vollständig:

- L_{sat} (SAT)
- CNF-SAT
- k-CNF-SAT für $k \geq 3$

NP-vollständige Probleme

Theorem [NP-vollständige Probleme]

Die folgenden Probleme liegen in **NP** und sind **NP**-vollständig:

- L_{sat} (SAT)
- CNF-SAT
- k-CNF-SAT für $k \geq 3$

Theorem [Probleme in P]

Die folgenden Probleme liegen in P:

- DNF-SAT
- k-DNF-SAT für alle k
- 2-CNF-SAT

NP-vollständige Probleme

- Ist eine logische Formel erfüllbar? (**Satisfiability**)
- Ist ein (un-) gerichteter Graph hamiltonsch? (**Hamiltonian circle**)
- Gibt es in einem Graphen eine Clique der Größe k? (**Clique of size** k)
- Ist ein Graph mit drei Farben zu färben? (3-**colorability**)
- Gibt es in einer Menge von ganzen Zahlen eine Teilmenge mit der Gesamtsumme x? (**Subset Sum**)

NP-vollständige Probleme

Definition [Hamilton Circle]

Hamilton-Kreis: Weg entlang der Kanten in einem Graphen, der jeden Knoten genau einmal besucht und wieder zum Ausgangspunkt zurückkehrt.

$L_{\mathbf{Ham_{undir}}}$**:** Die Sprache, die aus allen ungerichteten Graphen besteht, in denen es einen Hamilton-Kreis gibt.

$L_{\mathbf{Ham_{dir}}}$**:** Die Sprache, die aus allen gerichteten Graphen besteht, in denen es einen Hamilton-Kreis gibt.

NP-vollständige Probleme

- Ist eine logische Formel erfüllbar? (**Satisfiability**)
- Ist ein (un-) gerichteter Graph hamiltonsch? (**Hamiltonian circle**)
- Gibt es in einem Graphen eine Clique der Größe k? (**Clique of size** k)
- Ist ein Graph mit drei Farben zu färben? (3-**colorability**)
- Gibt es in einer Menge von ganzen Zahlen eine Teilmenge mit der Gesamtsumme x? (**Subset Sum**)

NP-vollständige Probleme

Definition[Maximale Clique: L_{Clique_k}]

Eine **Clique** in einem Graphen ist ein **vollständiger Teilgraph von** G.

Für $k \in \mathbb{N}$:

L_{Clique_k} Die Sprache, die aus allen ungerichteten Graphen besteht, die eine Clique der Größe k enthalten.

$L_{\text{Clique}_{\leq k}}$ Die Sprache, die aus allen ungerichteten Graphen besteht, die eine Clique der Größe $\leq k$ enthalten.

$L_{\text{Clique}} = \{(G, k) \mid G$ ungerichteter Graph, der eine Clique der Größe k enthält.$\}$

NP-vollständige Probleme

- Ist eine logische Formel erfüllbar? (**Satisfiability**)
- Ist ein (un-) gerichteter Graph hamiltonsch? (**Hamiltonian circle**)
- Gibt es in einem Graphen eine Clique der Größe k? (**Clique of size** k)
- Ist ein Graph mit drei Farben zu färben? (3-**colorability**)
- Gibt es in einer Menge von ganzen Zahlen eine Teilmenge mit der Gesamtsumme x? (**Subset Sum**)

NP-vollständige Probleme

Definition[k-colorability: $L_{\textbf{Color}_{\leq k}}$]

Ein (ungerichteter) Graph heißt **k-färbbar**, falls jeder Knoten mit einer von k Farben so gefärbt werden kann, dass benachbarte Knoten verschiedene Farben haben.

Für $k \in \mathbb{N}$:

$L_{\textbf{Color}_{\leq k}}$ Die Sprache, die aus allen ungerichteten, mit höchstens k Farben färbbaren Graphen besteht.

NP-vollständige Probleme

Einige Beispiel-Reduktionen

- $L_{\text{CNF-SAT}} \preceq_{\text{pol}} L_{\text{Clique}_{\leq k}}$,
- $L_{\text{Ham}_{\text{dir}}} \preceq_{\text{pol}} L_{\text{Ham}_{\text{undir}}}$,
- $L_{\text{Ham}_{\text{undir}}} \preceq_{\text{pol}} L_{\text{Ham}_{cost \leq k}}$, L_{Clique_k},
- $L_{\text{SAT}} \preceq_{\text{pol}} L_{\text{3-CNF-SAT}}$,
- $L_{\text{SAT}} \preceq_{\text{pol}} L_{\text{CNF-SAT}}$,
- $L_{\text{3-CNF-SAT}} \preceq_{\text{pol}} L_{\text{Color}_{\leq k}}$.

NP-vollständige Probleme

Einige Beispiel-Reduktionen

- $L_{\text{CNF-SAT}} \preceq_{\text{pol}} L_{\text{Clique}_{\leq k}}$,
- $L_{\text{Ham}_{\text{dir}}} \preceq_{\text{pol}} L_{\text{Ham}_{\text{undir}}}$,
- $L_{\text{Ham}_{\text{undir}}} \preceq_{\text{pol}} L_{\text{Ham}_{cost \leq k}}$, L_{Clique_k},
- $L_{\text{SAT}} \preceq_{\text{pol}} L_{\text{3-CNF-SAT}}$,
- $L_{\text{SAT}} \preceq_{\text{pol}} L_{\text{CNF-SAT}}$,
- $L_{\text{3-CNF-SAT}} \preceq_{\text{pol}} L_{\text{Color}_{\leq k}}$.

NP-vollständige Probleme

Beispiel: SAT $\preceq_{\text{pol}}$ 3-CNF-SAT

1. Schritt: Wir eliminieren $\rightarrow, \leftrightarrow$ und ziehen $\neg$ ganz nach innen (NNF)

2. Schritt: Zusätzliche Klammern einführen ($\wedge, \vee$ binär)

3. Schritt: Wir ordnen jedem Klammerausdruck A op B (op $\in \{\vee, \wedge\}$) ein neues Atom $P_{A \text{ op} B}$ zu. Das $P_{A \text{ op } B}$ denselben Wahrheitswert erhält wie A op B erzwingen wir durch die Formel $P_{A \text{ op } B} \leftrightarrow (A \text{ op } B)$.

4. Schritt: Wir ersetzen:

$P_{A \vee B} \leftrightarrow (A \vee B)$ durch $(\neg P_{A \vee B} \vee A \vee B) \wedge (\neg A \vee P_{A \vee B}) \wedge (\neg A \vee P_{A \vee B})$

$P_{A \wedge B} \leftrightarrow (A \wedge B)$ durch $(\neg P_{A \vee B} \vee A) \wedge (\neg P_{A \vee B} \vee B) \wedge (\neg A \vee \neg B \vee P_{A \vee B})$

$\mapsto P_F \wedge \text{Rename}(F) =: f(F)$

$F \in \text{SAT}$ gdw. F erfüllbare aussagenlogische Formel

gdw. $P_F \wedge \mathit{Rename}(F)$ erfüllbar

gdw. $f(F) \in$ 3-CNF-SAT

Beispiel

Sei F die Formel:

$$[(Q \wedge \neg P \wedge \neg(\neg(\neg Q \vee \neg R))) \vee (Q \wedge \neg P \wedge \neg(Q \wedge \neg P))] \wedge (P \vee R).$$

1. Schritt: Wir eliminieren $\rightarrow, \leftrightarrow$ und ziehen $\neg$ ganz nach innen (NNF)

$$F_1 = [(Q \wedge \neg P \wedge (\neg Q \vee \neg R)) \vee (Q \wedge \neg P \wedge (\neg Q \vee P))] \wedge (P \vee R)$$

2. Schritt: Zusätzliche Klammern einführen ($\wedge, \vee$ binär)

$$F_2 = [((Q \wedge \neg P) \wedge (\neg Q \vee \neg R)) \vee (Q \wedge (\neg Q \vee P) \wedge \neg P)] \wedge (P \vee R)$$

3. Schritt: Wir ordnen jedem Klammerausdruck ein neues Atom:

$$[\underbrace{\underbrace{\underbrace{(Q \wedge \neg P)}_{P_1} \wedge \underbrace{(\neg Q \vee \neg R)}_{P_2}}_{P_6} \vee \underbrace{\underbrace{(Q \wedge \neg P)}_{P_1} \wedge \underbrace{(\neg Q \vee P)}_{P_4}}_{P_7}}_{P_8}] \wedge \underbrace{(P \vee R)}_{P_5}.$$

P_F

Example

3. Schritt: Wir ordnen jedem Klammerausdruck ein neues Atom: inside).

$$\overbrace{[\overbrace{\underbrace{(\underbrace{(Q \wedge \neg P)}_{P_1} \wedge \underbrace{(\neg Q \vee \neg R)}_{P_2})}_{P_6} \vee \underbrace{(\underbrace{(Q \wedge \neg P)}_{P_1} \wedge \underbrace{(\neg Q \vee P)}_{P_4})}_{P_7}]}^{P_8} \wedge \underbrace{(P \vee R)}_{P_5}}^{P_F}.$$

F ist erfüllbar genau dann, wenn $P_F \wedge \text{Rename}(F)$ erfüllbar:

$$\begin{array}{llllll} P_F & \wedge & (P_F \leftrightarrow (P_8 \wedge P_5)) & \wedge & (P_1 \leftrightarrow (Q \wedge \neg P)) \\ & \wedge & (P_8 \leftrightarrow (P_6 \vee P_7)) & \wedge & (P_2 \leftrightarrow (\neg Q \vee \neg R)) \\ & \wedge & (P_6 \leftrightarrow (P_1 \wedge P_2)) & \wedge & (P_4 \leftrightarrow (\neg Q \vee P)) \\ & \wedge & (P_7 \leftrightarrow (P_1 \wedge P_4)) & \wedge & (P_5 \leftrightarrow (P \vee R)) \end{array}$$

Example

F ist erfüllbar genau dann, wenn $P_F \wedge \text{Rename}(F)$ erfüllbar:

$$
\begin{array}{llllll}
P_F & \wedge & (P_F \leftrightarrow (P_8 \wedge P_5)) & \wedge & (P_1 \leftrightarrow (Q \wedge \neg P)) \\
& \wedge & (P_8 \leftrightarrow (P_6 \vee P_7)) & \wedge & (P_2 \leftrightarrow (\neg Q \vee \neg R)) \\
& \wedge & (P_6 \leftrightarrow (P_1 \wedge P_2)) & \wedge & (P_4 \leftrightarrow (\neg Q \vee P)) \\
& \wedge & (P_7 \leftrightarrow (P_1 \wedge P_4)) & \wedge & (P_5 \leftrightarrow (P \vee R))
\end{array}
$$

Step 4: CNF berechnen:

$$
\begin{array}{llll}
P_F & \wedge & (\neg P_F \vee P_8) \wedge (\neg P_F \vee P_5) & \wedge & (\neg P_1 \vee Q) \wedge (\neg P_1 \vee \neg P) \\
& \wedge & (\neg P_8 \vee \neg P_5 \vee P_F) & \wedge & (\neg Q \vee P \vee P_1) \\
& \wedge & (\neg P_8 \vee P_6 \vee P_7) & \wedge & (\neg P_2 \vee \neg Q \vee \neg R) \\
& \wedge & (\neg P_6 \vee P_8) \wedge (\neg P_7 \vee P_8) & \wedge & (Q \vee P_2) \wedge (R \vee P_2) \\
& \wedge & (\neg P_6 \vee P_1) \wedge (\neg P_6 \vee P_2) & \wedge & (\neg P_4 \vee \neg Q \vee P) \\
& \wedge & (\neg P_1 \vee \neg P_2 \vee P_6) & \wedge & (Q \vee P_4) \wedge (\neg P \vee P_4) \\
& \wedge & (\neg P_7 \vee P_1) \wedge (\neg P_7 \vee P_4) & \wedge & (\neg P_5 \vee P \vee R) \\
& \wedge & (\neg P_1 \vee \neg P_4 \vee P_7) & \wedge & (\neg P \vee P_5) \wedge (\neg R \vee P_5)
\end{array}
$$

NP-vollständige Probleme

Beispiel: CNF-SAT $\preceq_{\text{pol}}$ Clique$_{\leq k}$

Gegeben eine Instanz von CNF
(eine Konjunktion von Klauseln $C_1 \wedge C_2 \wedge \ldots \wedge C_k$)

Wir konstruieren daraus einen Graphen:

Knoten: die Paare (x, i), so dass x ein Literal ist, dass in der Klausel C_i vorkommt.

Kanten: Es gibt eine Kante zwischen (x, i) und (y, j) falls:
(1) $i \neq j$, und
(2) x, y sind nicht komplementär.

Es gilt dann:
Die CNF-Formel ist erfüllbar genau dann,
wenn der zugeordnete Graph eine Clique der Größe k hat.

NP-vollständige Probleme

- Ist eine logische Formel erfüllbar? (**Satisfiability**)
- Ist ein (un-) gerichteter Graph hamiltonsch? (**Hamiltonian circle**)
- Gibt es in einem Graphen eine Clique der Größe k? (**Clique of size** k)
- Ist ein Graph mit drei Farben zu färben? (3-**colorability**)
- Gibt es in einer Menge von ganzen Zahlen eine Teilmenge mit der Gesamtsumme x? (**Subset Sum**)