Teil III: Komplexitätstheorie

1. Vorbemerkungen

bisher: welche Probleme sind entscheidbar (lösbar) und welche nicht?
jetzt: welche entscheidbaren Probleme sind „effizient“ zu lösen.

Beispiel: es gibt 40! verschiedene Rundreisen durch 40 Städte. Bei Überprüfung von 1 Mio. Rundreisen pro Sekunde braucht man $2{,}5 * 10^{34}$ Jahre, etwa $1{,}8 * 10^{24}$ mal das Alter des Universums.

Ziel:	Klassifikation algorithmischer Probleme gemäß ihrem Bedarf an Ressourcen (Rechenzeit, Speicherplatz)
Aspekte:	Analyse existierender Algorithmen -> obere Schranke Analyse der minimalen Komplexität -> untere Schranke

Hier: Theorie der **NP**-Vollständigkeit.

Nachweis, dass ein Problem P **NP**-vollständig ist:

- kein Beweis, dass P keine effiziente Lösung hat
- aber: Beweis, dass P *genauso schwer ist wie eine Vielzahl anderer Probleme, für die bisher niemand effiziente Lösung finden konnte*
  (was allgemein als starke Evidenz betrachtet wird, dass P nicht effizient lösbar ist).

Was heißt effizient? => in polynomieller Zeit lösbar.
Begründung: Algorithmen mit exponentieller Komplexität können nur auf sehr kleine Probleminstanzen angewendet werden; minimal größere Instanzen können drastischen Mehraufwand bedeuten.

Anforderung an zu identifizierende Komplexitätsklassen:
sollen unabhängig sein vom zugrunde liegenden Berechnungsmodell (WHILE-Programm, GOTO-Programm, TM, Mehrband-TM)

Beschränkung auf (entscheidbare) Entscheidungsprobleme (Antwort JA oder NEIN):

Beispiel Traveling Salesperson:
gegeben: Städte $c_1,..., c_n$ und Kosten für Reisen zwischen je 2 Städten

- Entscheidungsproblem:
  Gibt es Rundreise mit Gesamtkosten $\leq k$?
- Optimierungsproblem (Typ 1):
  Was sind die minimalen Gesamtkosten k einer Rundreise?
- Optimierungsproblem (Typ 2):
  Was ist eine Rundreise mit minimalen Gesamtkosten k?

Optimierungsprobleme mindestens so schwer wie Entscheidungsproblem.

Entscheidungsprobleme als Sprachen über geeignetem Alphabet Σ aufgefasst:

- Repräsentation einer Instanz des Problems als Wort über Alphabet Σ.
- Alle Wörter, für die Antwort JA lautet, bilden eine Sprache A über Σ.
- Es gibt bei jeder Eingabe haltende TM M mit A = T(M)

(O.B.d.A. können wir davon ausgehen, dass es einen einzigen akzeptierenden Zustand q_{JA} gibt. T(M) ist also die Menge der Wörter, für die M in Zustand q_{JA} terminiert)

Notation:

TM:	beliebige Turingmaschine	DTM:	deterministische TM
MBTM:	Mehrband TM	MBDTM:	deterministische MBTM

2. Die Komplexitätsklasse **P**

Def.: Sei $f: \mathbb{N} \to \mathbb{N}$ eine Funktion. Die Klasse TIME(f(n)) besteht aus allen Sprachen A, für die es eine MBDTM M gibt so dass

1. A = T(M) und
2. für alle $x \in \Sigma^*$: $time_M(x) \leq f(|x|)$.

Hierbei ist $time_M: \Sigma^* \to \mathbb{N}$ die Anzahl der Rechenschritte von M bei Eingabe x.

Also: es muss eine DTM geben, die A entscheidet und bei Eingabe eines beliebigen Wortes der Länge n nach höchstens f(n) Schritten stoppt.

Bemerkungen:

1. Der Zeitbedarf wird gemessen in Abhängigkeit von der Größe des Inputs.
2. Mehrband-TM realistischer als 1-Band-TM, bei letzteren viele Schritte, um sequentiell abgespeicherte Information zu lesen. Außerdem: Mehrband durch 1-Band simulierbar: wenn MBTM durch f(n) rechenzeitbeschränkt, so entsprechende 1BTM durch $O(f^2(n))$ beschränkt.
3. Wahl von MBTM (statt anderer Präzisierung des Algorithmenbegriffs) hat keinen Einfluss auf die hier definierten Komplexitätsklassen.
4. Gleiches gilt für die Art der Codierung von Problemen als Sprachen über einem Alphabet Σ (solange nicht völlig "unrealistische" Codierungen verwendet werden).

Whlg. O-Notation: für $f: \mathbb{N} \to \mathbb{N}$ ist

$$O(f(n)) = \{g: \mathbb{N} \to \mathbb{N} \mid \text{es gibt } c, n_0 \text{ so dass für alle } n \geq n_0: g(n) \leq c * f(n)\}$$

Bemerkungen:

- gilt für eine TM M, dass $time_M(x) \leq f(|x|)$ für alle $x \in \Sigma^*$, so nennt man M eine f(n)-zeitbeschränkte TM.
- eine f(n)-zeitbeschränkte TM nennt man O(g(n))-zeitbeschränkt, wenn $f \in O(g(n))$.

Hier werden wir uns für Klassen von Funktionen interessieren, wo z.B. Quadrieren keine Rolle spielt (also spielt es keine Rolle. ob wir MBTMs verwenden oder nicht, oder welches Berechenbarkeitsmodell wir zugrunde legen): Komplexitätsklasse P.

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

$$p(n) = a_k n^k + a_{k-1} n^{k-1} + ... + a_1 n^1 + a_0$$

wobei a_i, k aus $\mathbb{N}$.

Die Komplexitätsklasse **P** ist wie folgt definiert:

$$\mathbf{P} = \{A \mid \text{es gibt MBDTM M und Polynom p mit } T(M) = A \text{ und } time_M(x) \leq p(|x|)\}$$
$$= \bigcup_{p \text{ Polynom}} TIME(p(n))$$

Bemerkungen:
Komplexität n log n auch als polynomiell betrachtet, denn $n \log n = O(n^2)$
dagegen $n^{\log n}$, 2^n nicht durch Polynom beschränkt.
polynomiell (also **P**) vs. exponentiell: oft als die Grenze zwischen effizient lösbar und nicht effizient lösbar betrachte (tractable vs. intractable).
P und weit größere Klassen $TIME(2^n)$ oder $TIME(2^{2^{\cdots^2}})$ sind immer noch LOOP-berechenbar. Intuitiv: Jede WHILE-Schleife kann durch LOOP-Schleife ersetzt werden. Man kann Anzahl der Durchläufe nach oben beschränken.

3. Die Komplexitätsklasse **NP**

- **NP** verwendet im Gegensatz zu **P** nichtdeterministische TM.
- Intuition: Klasse der Probleme, bei denen eine (von möglicherweise exponentiell vielen) potentielle Lösung in polynomieller Zeit überprüft werden kann.
- In nichtdeterministischer "guessing" Phase wird mögliche Lösung erzeugt (für jede mögliche Lösung eine nichtdeterministische Rechnung), in "checking" Phase überprüft.

Traveling salesperson: guessing Phase erzeugt nichtdeterministisch mögliche Rundreisen, checking Phase überprüft Kosten der jeweils konstruierten Rundreise.

akzeptierende Rechnung: Folge von Konfigurationen, beginnend mit Startkonfiguration, endend mit Endzustand q_{JA}, aufeinanderfolgende Konfigurationen stehen in |-- Relation.
T(M): Menge der Wörter, für die es eine akzeptierende Rechnung gibt.

Def.: Für eine nichtdeterministische Turingmaschine M sei

$$ntime_M(x) = \max\{k \mid k \text{ Länge einer Rechnung von M bei Eingabe x}\}$$

Sei $f: \mathbb{N} \to \mathbb{N}$ eine Funktion. NTIME(f(n)) ist die Menge aller Sprachen A, für die es eine (nichtdeterministische) MBTM M gibt mit $A = T(M)$ und $ntime_M(x) \leq f(|x|)$.

Die Komplexitätsklasse **NP** ist wie folgt definiert:

$$\mathbf{NP} = \{A \mid \text{es gibt MBTM M und Polynom p mit } T(M) = A \text{ und } ntime_M(x) \leq p(|x|)\}$$
$$= \bigcup_{p \text{ Polynom}} NTIME(p(n))$$

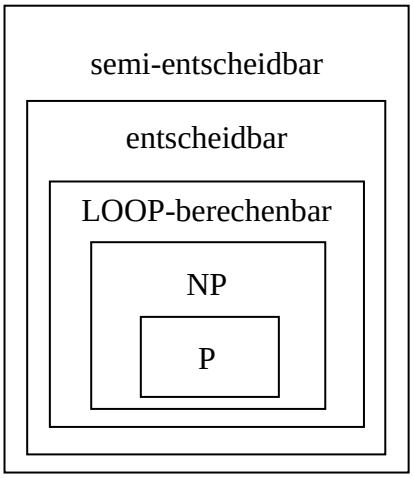
Es gilt offensichtlich: $\mathbf{P} \subseteq \mathbf{NP}$. Gilt auch $\mathbf{NP} \subseteq \mathbf{P}$?

P-**NP**-Problem: eines der wichtigsten ungelösten Probleme der theoretischen Informatik.

Viele bekannte Probleme der Informatik liegen in **NP** und man konnte nicht zeigen, ob sie in **P** liegen (Traveling Salesperson, Erfüllbarkeit Aussagenlogik, ...).
Gibt es keinen "effizienten" Algorithmus (d.h., liegen die Probleme nicht in **P**?), oder hat man bisher nur keinen gefunden? Allgemein wird vermutet, dass die erste Alternative zutrifft.

Man kann zeigen, dass es "schwierigste" Probleme in **NP** gibt, und zwar die **NP**-vollständigen Probleme.

Graphischer Überblick (**P** $\subset$ **NP** vermutet):



Hinweis: man kann $ntime_M$ alternativ auch so definieren (siehe etwa Schöning):

Def. (alternativ): Für eine nichtdeterministische Turingmaschine M sei

$ntime_M(x) =$

$\min\{k \mid k$ Länge einer akzept. Rechnung von M bei Eingabe $x\}$	falls $x \in T(M)$
0	falls $x \notin T(M)$

Diese Definition geht davon aus, dass man immer „richtig rät“, und dass Eingaben, die nicht zur Sprache gehören, keine Rolle spielen. Der Vorteil unserer obigen Definition: man sieht sofort, dass **NP** nur entscheidbare Probleme umfasst, und nichtakzeptierende Rechnungen von M fallen nicht „unter den Tisch“.

Aber: beide Definitionen von $ntime_M$ führen zur selben Klasse **NP.** Das liegt intuitiv daran, dass man aus einer nichtdeterministischen Turingmaschine, deren kürzeste akzeptierende Rechnung bei Eingabe x durch Polynom p beschränkt ist, immer eine äquivalente nichtdeterministische TM konstruieren kann, für die alle Rechnungen (inklusive der verwerfenden) polynomiell beschränkt sind: man muss nur einen Zähler mitführen, der dafür sorgt, dass nach $p(|x|) + 1$ Schritten der ursprünglichen TM verwerfend angehalten wird (für einen vollständigen Beweis siehe Asteroth/Baier, 143f).

4. **NP**-Vollständigkeit

Die **NP**-vollständigen Probleme sind, intuitiv, die schwierigsten in **NP**: wenn man eines dieser Probleme effizient lösen könnte, dann könnte man alle Probleme in **NP** effizient lösen.

Def.: Seien $A \subseteq \Sigma^*$ und $B \subseteq \Gamma^*$ Sprachen. A heißt auf B polynomiell reduzierbar ($A \leq_p B$) falls es eine totale, polynomiell berechenbare Funktion $f: \Sigma^* \rightarrow \Gamma^*$ gibt, so dass für alle $x \in \Sigma^*$:

$$x \in A \Leftrightarrow f(x) \in B.$$

Lemma: Falls $A \leq_p B$ und $B \in \mathbf{P}$ ($B \in \mathbf{NP}$), so ist auch $A \in \mathbf{P}$ ($A \in \mathbf{NP}$).

Beweis (für $A \in \mathbf{P}$): Übersetzung von A in B in polynomieller Zeit, Lösung von B in polynomieller Zeit => ergibt insgesamt polynomielles Entscheidungsverfahren für A.
(für $A \in \mathbf{NP}$ entsprechend).

Def.: Eine Sprache A heißt **NP**-hart, falls für alle Sprachen $L \in \mathbf{NP}$ gilt: $L \leq_p A$.

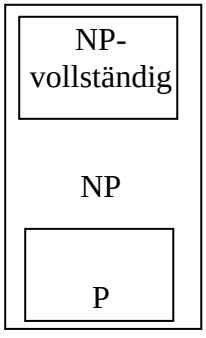
Def.: Eine Sprache A heißt **NP**-vollständig, falls gilt: A ist **NP**-hart und $A \in \mathbf{NP}$.

Anmerkung: die Relation $\leq_p$ ist transitiv. Falls A **NP**-hart und $A \leq_p B$, dann muss auch B **NP**-hart sein (da für alle Sprachen L gilt $L \leq_p A$ und außerdem $A \leq_p B$, muss wegen Transitivität auch $L \leq_p B$ gelten).

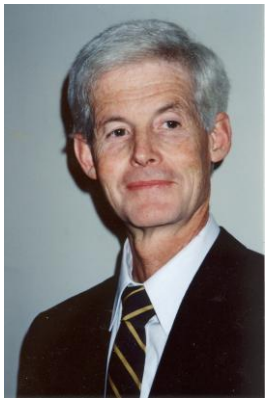
Satz: Sei A **NP**-vollständig. Dann gilt: $A \in \mathbf{P} \Leftrightarrow \mathbf{P} = \mathbf{NP}$.

Beweis:
"->"Sei $A \in \mathbf{P}$ und L eine beliebige Sprache aus **NP**. Da A **NP**-hart ist, gilt $L \leq_p A$, damit auf Grund obigen Lemmas $L \in \mathbf{P}$, womit gezeigt ist, dass $\mathbf{NP} \subseteq \mathbf{P}$. Da offensichtlich auch $\mathbf{P} \subseteq \mathbf{NP}$ ergibt sich $\mathbf{P} = \mathbf{NP}$.
"<-" offensichtlich, denn wenn $\mathbf{P} = \mathbf{NP}$ und $A \in \mathbf{NP}$, dann gilt natürlich auch $A \in \mathbf{P}$.

Vermutete Situation:



NP-Härte ist eine Eigenschaft, deren Nachweis auf den ersten Blick äußerst schwierig scheint, denn man muss ja zeigen, dass wirklich alle Probleme in **NP** sich auf ein bestimmtes Problem reduzieren lassen.



Stephen A. Cook (Univ. Toronto) ist es 1971 gelungen, einen Beweis für die **NP**-Härte und **NP**-Vollständigkeit eines Problems zu führen: der Erfüllbarkeit der Aussagenlogik. Die Bedeutung dieses Resultats (für die Cook 1982 der Turing Award verliehen wurde) besteht darin, dass man weitere **NP**-Vollständigkeitsresultate relativ einfach durch Reduktion führen kann, wenn man bereits ein **NP**-vollständiges Problem kennt.

Def.: Das Erfüllbarkeitsproblem der Aussagenlogik, SAT, ist das folgende:

gegeben:	eine Formel F der Aussagenlogik
gefragt:	ist F erfüllbar, d.h. gibt es eine Belegung der Variablen von F mit 0,1, so dass F unter dieser Belegung zu 1 ausgewertet wird?

Formulierung als Sprache:
SAT = {F $\in \Sigma^*$ | F erfüllbare aussagenlogische Formel} für geeignetes Alphabet Σ.

Satz (Cook): Das Erfüllbarkeitsproblem der Aussagenlogik SAT ist **NP**-vollständig.

Beweis:
1) SAT in **NP**
das ist der „einfache“ Teil des Beweises. Es gibt eine nichtdeterministische TM, die Variablenbelegungen "rät", d.h. es gibt 2^k mögliche nichtdeterministische Berechnungen (k ist die Anzahl der Variablen in F). Der Wert von F unter gegebener Variablenbelegung kann in polynomieller Zeit berechnet werden.

2) SAT **NP**-hart
Sei L beliebiges **NP**-Problem. Dann ist L=T(M) für eine nichtdeterministische, polynomiell beschränkte TM M. Sei p das Polynom, das die Rechenzeit von M beschränkt.
Wir definieren eine Formel F, so dass $x = x_1...x_n \in L \Leftrightarrow$ F erfüllbar.
Grundidee hierbei ist, dass jedes Modell von F genau eine akzeptierende Berechnung von M modelliert. Dazu müssen Aussagenvariablen verwendet werden, die komplett den Zustand und Bandinhalt von M nach jedem Schritt repräsentieren. Das sind zwar immens viele, aber wegen der Beschränkung durch p eben nur polynomiell viele.

Sei $\Gamma = \{a_1,...,a_r\}$ Arbeitsalphabet von M, $Z = \{z_0,...,z_k\}$ Zustandsmenge, z_0 Anfangszustand.

F ist aufgebaut aus folgenden aussagenlogischen Variablen:

Variable:	Indizes:	Bedeutung:
$zust_{t,zj}$	t = 0,1,..., p(n) j = 0, ..., k	M ist nach t Schritten in Zustand z_j
$pos_{t,i}$	t = 0,1,..., p(n) i = -p(n), ..., p(n)	M ist nach t Schritten in Position i

$band_{t,i,aj}$	$t = 0,1,..., p(n)$ $i = -p(n), ..., p(n)$ $a_j \in \Gamma$	nach t Schritten steht a_j auf Band in Position i

F verwendet Teilformel G: $G(x_1, ..., x_m) \Leftrightarrow x_i$ wahr für genau ein i
(Definition später, Größe $O(m^2)$). Die gesuchte Formel F ist eine Konjunktion von 5 Teilformeln:

$$F = R \wedge A \wedge Ü_1 \wedge Ü_2 \wedge E$$

R (Randbedingungen) besagt, dass sich M immer in genau 1 Zustand befindet, LS-Kopf an genau 1 Position, in jeder Bandposition genau 1 Zeichen aus Γ:

$$R = \bigwedge_t [G(zust_{t,z1},..., zust_{t,zk}) \wedge G(pos_{t,-p(n)},..., pos_{t,p(n)}) \wedge \bigwedge_i G(band_{t,i,a1}, ..., band_{t,i,ar})]$$

A beschreibt die Anfangskonfiguration von M:

$$A = zust_{0,z0} \wedge pos_{0,1} \wedge \bigwedge_{j=1...n} band_{0,j,xj} \wedge \bigwedge_{j=-p(n)...0} band_{0,j,□} \wedge \bigwedge_{j=-n+1...p(n)} band_{0,j,□}$$

$Ü_1$ beschreibt Übergänge von t zu t+1 an Kopfposition, L,N,R durch -1,0,+1 dargestellt:

$$Ü_1 = \bigwedge_{t,z,i,a} [(zust_{t,z} \wedge pos_{t,i} \wedge band_{t,i,a} \rightarrow \bigvee_{z',a',y,\ (z',a',y)\in\delta(z,a)} (zust_{t+1,z'} \wedge pos_{t+1,i+y} \wedge band_{t+1,i,a'})]$$

$Ü_2$ besagt, dass der Bandinhalt auf anderen Positionen unverändert bleibt:

$$Ü_2 = \bigwedge_{t,i,a} (\neg pos_{t,i} \wedge band_{t,i,a} \text{ -> } band_{t+1,i,a})$$

E besagt, dass am Ende der Berechnung der akzeptierende Endzustand z_{JA} erreicht sein muss:

$$E = zust_{p(n),zJA}$$

hier wird angenommen, dass für den Endzustand z_{JA} und alle a gilt: $(z_{JA}, a, 0) \in \delta(z_{JA}, a)$.

F beschreibt nun vollständig alle möglichen Berechnungen von M: jedes Modell von F entspricht einer nichtdeterministischen Berechnung von M, die im akzeptierenden Zustand z_{Ja} endet. Es gilt also wie gewünscht: $x \in L \Leftrightarrow$ F erfüllbar.

F kann in polynomieller Zeit aus x berechnet werden. Dazu müssen wir nur noch zeigen, dass G polynomiell ist:

$$G(x_1,...,x_k) = \bigvee_{i=1,...,k} x_i \ \wedge \bigwedge_{j=1,\ldots,k-1} \bigwedge_{r=j+1,\ldots k} \neg(x_j \wedge x_r)$$

Diese Formel ist wie gewünscht wahr genau dann wenn genau eines der x_i wahr ist. $O(k^2)$

Beispiel: $G(x_1,x_2,x_3) = (x_1 \vee x_2 \vee x_3) \wedge \neg(x_1 \wedge x_2) \wedge \neg(x_1 \wedge x_3) \wedge \neg(x_2 \wedge x_3)$

Damit sind sowohl G als auch die gesamte Formel F polynomiell in der Länge des Eingabewortes x. F kann somit in polynomieller Zeit aus x, M und p berechnet werden. Damit ist L auf SAT reduzierbar und der Beweis abgeschlossen.

Beispiele weiterer **NP**-vollständiger Probleme:

3SAT	gegeben:	Formel F in KNF mit höchstens 3 Literalen pro Klausel (3KNF)
	gefragt:	Ist F erfüllbar?
Clique	gegeben:	ungerichteter Graph G = (V,E), $k \in \mathbb{N}$
	gefragt:	besitzt G Clique der Größe k, d.h. Teilmenge V' von V so dass $\|V'\| = k$ und für alle $u,v \in V'$ mit $u \neq v$ gilt: $\{u,v\} \in E$. (Beispiel Party: Gruppe der Größe k so dass jeder jeden kennt?)
Rucksack	gegeben:	$a_1,...,a_k \in \mathbb{N}$, $b \in \mathbb{N}$
	gefragt:	gibt es $I \subseteq \{1,...,k\}$ mit $\Sigma_{i\in I} a_i = b$
Partition:	gegeben:	$a_1,...,a_k \in \mathbb{N}$
	gefragt:	gibt es $I \subseteq \{1,...,k\}$ mit $\Sigma_{i\in I} a_i = \Sigma_{j\notin I} a_j$
BinPacking:	gegeben:	"Behältergröße" $b \in \mathbb{N}$, Anzahl Behälter $k \in \mathbb{N}$, $a_1,...,a_n \leq b$
	gefragt:	gibt es f: {1,...,n} -> {1,...,k} so dass für alle $j \in \{1,...,k\}$: $\Sigma_{f(i)=j} a_i \leq b$

Hamilton-Kreis (gerichtet, ungerichtet analog)

	gegeben:	gerichteter Graph G = (V,E), $V = \{v_1, ..., v_n\}$
	gefragt:	gibt es Hamilton Kreis, d.h. Permutation π der Knotenindizes, so dass $(v_{\pi(i)}, v_{\pi(i+1)}) \in E$ und $(v_{\pi(n)}, v_{\pi(1)}) \in E$. ($\pi$(i) alter Index)

Traveling Salesperson:

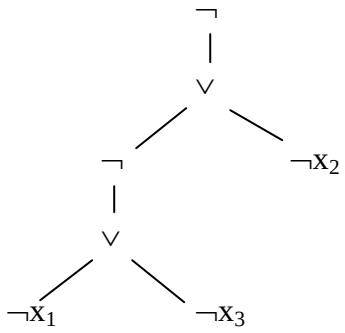
	gegeben:	n x n Matrix $(M_{i,j})$ von "Entfernungen" zwischen n "Städten", k
	gefragt:	gibt es Permutation π der Städteindizes, so dass $\Sigma_{i=1,...,n-1} M_{\pi(i),\pi(i+1)} + M_{\pi(n),\pi(1)} \leq k$

Beispielbeweise:

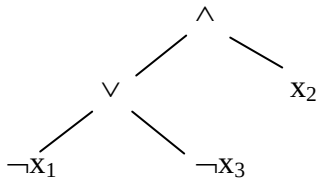
Satz: 3SAT ist **NP**-vollständig
Beweis: Inklusion: Spezialfall von SAT, SAT in **NP**, also auch 3SAT in **NP**.
NP-Härte: $SAT \leq_p 3SAT$
gesucht: Formel F' in 3KNF so dass F' erfüllbar gdw F erfüllbar; F' aus F in polynomieller Zeit berechenbar.
Statt eines kompletten Beweises Veranschaulichung an einem Beispiel:

$F = \neg(\neg(\neg x_1 \vee \neg x_3) \vee \neg x_2)$

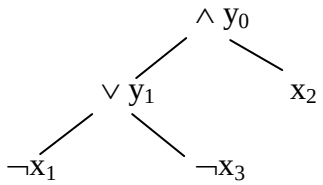
Repräsentation als Baum:



Durch Anwendung der deMorganschen Regeln und Elimination doppelter Negation lässt sich Negation vor Atome bringen:



Wir führen neue Variablen ein und ordnen der Wurzel y_0 und jedem inneren Knoten ein y_i zu (verschiedene Knoten erhalten verschiedene Variablen):



Für jeden inneren Knoten bilden wir Formeln der Gestalt $y_i <-> l_i \bullet r_i$, wobei $\bullet$ der Junktor des Knotens ist, l_i und r_i sein linker und rechter Nachfolger. Im Beispiel:

$$y_0 <-> (y_1 \wedge x_2) \qquad\qquad y_1 <-> (\neg x_1 \vee \neg x_3)$$

Die gesuchte Formel ist die Konjunktion von y_0 und all diesen Formeln, also hier:

$$F' = y_0 \wedge (y_0 <-> (y_1 \wedge x_2)) \wedge (y_1 <-> (\neg x_1 \vee \neg x_3))$$

F und F' sind erfüllbarkeitsgleich: jedes Modell von F kann zu einem Modell von F' erweitert werden, indem man die Belegungen der Blätter entsprechend dem Junktor im Baum zur Wurzel propagiert. Umgekehrt ist jedes Modell von F' eingeschränkt auf die Variablen in F Modell von F.

Im Beispiel: $x_1, x_2, \neg x_3$ ist ein Modell von F, $x_1, x_2, \neg x_3, y_1, y_0$ ist ein Modell von F'.

Die Teilformeln von F' lassen sich in 3KNF umformen (-L Komplement von L):

$a <-> (b \vee c)$ ➔ $(a \vee -b) \wedge (\neg a \vee b \vee c) \wedge (a \vee -c)$
$a <-> (b \wedge c)$ ➔ $(\neg a \vee b) \wedge (\neg a \vee c) \wedge (a \vee -b \vee -c)$

Insgesamt ergibt sich eine Formel in 3KNF. Alle Umformungen lassen sich in polynomieller Zeit durchführen. Damit ist SAT auf 3SAT polynomiell reduziert und aus der **NP**-Härte von SAT ergibt sich die von 3SAT.

Anmerkung: 2SAT (also Erfüllbarkeit von Formeln in 2KNF) liegt in **P**. Es gibt nur $4n^2$ syntaktisch verschiedene Klauseln mit genau 2 (nicht notwendigerweise verschiedenen) Literalen über n Variablen. Mit dem vollständigen Resolutionsverfahren entstehen aus 2-Klauseln nur neue Klauseln der Länge ≤ 2. Die Menge der möglichen Resolventen ist also polynomial beschränkt, und damit auch die Rechenzeit.

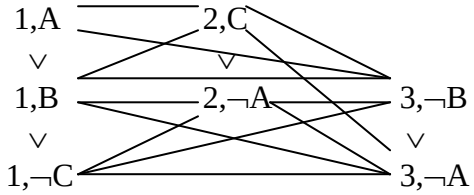
Satz: Clique ist **NP**-vollständig.

Beweis: Inklusion: rate Teilmenge V' der Größe k, teste in polynomieller Zeit, dass Knoten in V' paarweise mit Kanten verbunden sind.
NP-Härte: wir zeigen $3SAT \leq_p$ Clique:
Sei F Formel in 3KNF, d.h. $F = (z_{11} \vee z_{12} \vee z_{13}) \wedge ... \wedge (z_{m1} \vee z_{m2} \vee z_{m3})$.
In polynomieller Zeit erzeugen wir den zugeordneten Graph G = (V,E) mit
$V = \{(1,1), (1,2), (1,3), ..., (m,1), (m,2), (m,3)\}$
$E = \{\{(i,j), (p,q)\} \mid i \neq p, z_{ij} \neq -z_{pq}\}$

-z ist das Komplement des Literals z. Intuitiv: (i,j) und (p,q) "kennen sich", wenn sie aus verschiedenen Disjunktionen stammen und sich nicht widersprechen. Jetzt gilt:

F erfüllbar durch Belegung B
- ⇔ es gibt in jeder Klausel ein Literal, das unter B Wert 1 hat
- ⇔ es gibt Literale $z_{1,j1}, ..., z_{m,jm}$ die paarweise nicht komplementär sind
- ⇔ es gibt Knoten $(1,j_1), (2,j_2) ... (m,j_m)$ die paarweise verbunden sind
- ⇔ G hat Clique der Größe m.

Beispiel: $(A \vee B \vee \neg C) \wedge (C \vee \neg A) \wedge (\neg B \vee \neg A)$
statt Index wird zur besseren Lesbarkeit das Literal selbst als 2. Komponente verwendet:



erfüllende Belegungen der Formel:	Entsprechende Cliquen der Größe 3:
A ¬B C	{(1, A), (2, C), (3, ¬B)}
¬A ¬B C	{(1, B), (2, C), (3, ¬A)}
¬A B ¬C	{(1, ¬C), (2, ¬A), (3, ¬A)}
¬A ¬B ¬C	{(1, ¬C), (2, ¬A), (3, ¬B)} oder {(1, ¬C), (2, ¬A), (3, ¬A)}

Sei L ein Problem (eine Sprache) über Σ^*. Das komplementäre Problem L^c ist die Sprache $\Sigma^* \backslash L$. Für deterministische Entscheidungsverfahren ist die Komplexität von L und L^c gleich.

Satz: $L \in \mathbf{P}$ impliziert $L^c \in \mathbf{P}$.
Beweis: JA und NEIN Ausgaben sind bei deterministischen Verfahren vertauschbar.

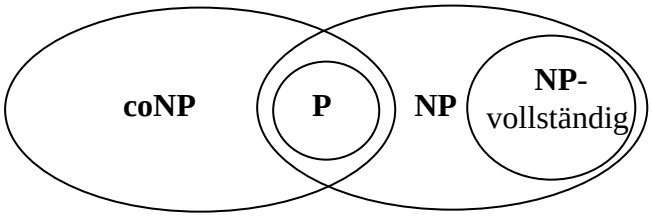
Für nichtdeterministische Verfahren sind die Antworten JA und NEIN nicht symmetrisch!!

Beispiel SAT: es gibt nichtdeterministische TM, die Belegung B "rät" und in polynomieller Zeit testet, ob B gegebene Formel F wahr macht. Betrachte das komplementäre Problem UNSAT: wir können nicht einfach Belegung B raten und verifizieren, dass B die Formel F falsch macht, denn es kann ja eine andere Belegung geben, die B wahr macht. (Ja und Nein bei TM für SAT vertauschen liefert alle Formeln, die eine nichterfüllende Belegung haben, also alle Formeln, die nicht Tautologien sind, nicht aber alle Kontradiktionen).

Def.: Sei **C** eine Komplexitätsklasse. Die Klasse **coC** besteht aus allen Sprachen, deren Komplement in **C** liegt.

Es gilt also UNSAT $\in$ **coNP**.

Während **P** = **coP**, ist der Zusammenhang zwischen **NP** und **coNP** ebenso ungelöst wie das **P**-**NP**-Problem. Vermutete Situation:

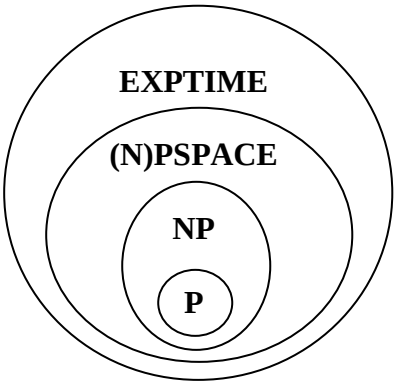


Eine weitere wichtige Komplexitätsklasse ist **EXPTIME**, die Klasse aller Probleme, die sich in exponentieller Zeit durch eine deterministische TM lösen lassen (n Eingabegröße):

$$\mathbf{EXPTIME} = \bigcup_{c>1} \bigcup_{k\geq 1} \mathbf{TIME}(c^{n^k})$$

Analog zu den Klassen **P** und **NP**, die auf der Basis von Zeitkomplexität definiert sind, lassen sich auch die Klassen **PSPACE** und **NPSPACE** auf der Basis des benötigten Speicherplatzes definieren: **PSPACE**/**NPSPACE** ist die Klasse von Problemen, die durch eine deterministische/nichtdeterministische TM mit polynomiellem Speicherplatz gelöst werden können. Folgende Beziehungen lassen sich zeigen:

NP ⊆ PSPACE = NPSPACE ⊆ EXPTIME



Die polynomielle Hierarchie (**PH**) ist die vermutete Struktur von Komplexitätsklassen zwischen **NP** und **PSPACE**. Ausgangspunkt ist die Frage, ob durch die Hinzunahme von *Orakeln*, die Leistungsfähigkeit einer Turingmaschine gesteigert werden kann.

Orakel sind Erweiterungen einer Turingmaschine. Eine Turingmaschine mit Orakel *A* (wobei *A* eine Sprache ist), kann in einem Schritt entscheiden, ob ein Wort *w* zu *A* gehört oder nicht.

P^{NP} (sprich: "P hoch NP") ist die Menge aller Probleme, die sich von einer deterministischen Turingmaschine entscheiden lassen, die in Abhängigkeit von der Eingabelänge nur polynomiellen Zeitverbrauch aufweist, zur Lösung jedoch ein Orakel benutzen kann, das in der Lage ist, ein Problem aus NP zu entscheiden

PH wird folgendermaßen definiert:

$$\Delta_0^{\mathrm{P}} := \Sigma_0^{\mathrm{P}} := \Pi_0^{\mathrm{P}} := \mathrm{P},$$

$$\Delta_{i+1}^{\mathrm{P}} := \mathrm{P}^{\Sigma_i^{\mathrm{P}}} \qquad \Sigma_{i+1}^{\mathrm{P}} := \mathrm{NP}^{\Sigma_i^{\mathrm{P}}} \qquad \Pi_{i+1}^{\mathrm{P}} := \mathrm{coNP}^{\Sigma_i^{\mathrm{P}}}$$

insbesondere:

$$\Sigma_1^{\mathrm{P}} = \mathrm{NP} \qquad \Pi_1^{\mathrm{P}} = \mathrm{coNP} \qquad \Delta_2^{\mathrm{P}} = \mathrm{P}^{\mathrm{NP}}$$

Anders als zunächst vermutet, können sich durch **PH** nicht alle Komplexitätsklassen abbilden lassen. Zwar ist die genaue Struktur der Hierarchie weiterhin unbekannt, jedoch konnte sich folgender Sachverhalt beweisen lassen

$$\mathrm{PH} := \bigcup_{i=0}^{\infty} \Delta_i^{\mathrm{P}} \subseteq \mathrm{PSPACE}$$

Ob **PH** = **PSPACE** gilt, ist bis heute unbekannt. Zudem weiß man, dass **PH** im Falle der Gleichheit von **P** und **NP** kollabiert, d.h. es würde in diesem Fall gelten:

$$\forall i \geq 0 : \Delta_i^{\mathrm{P}} = \mathrm{P}$$

(Analog für Σ_i und Π_i).