14 Weitere NP-vollständige Probleme

Es gibt eine Fülle von interessanten NP-vollständigen Problemen. In diesem Abschnitt sollen einige von ihnen vorgestellt werden. Am Schluss betrachten wir zwei NP-harte Probleme aus der Theorie der formalen Sprachen.

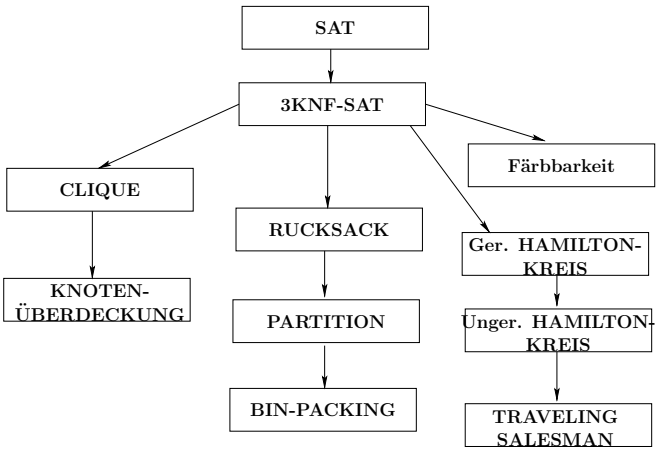
Wie kann man von einem gegebenen Problem L nachweisen, dass es tatsächlich NP-vollständig ist? Man muss dazu zwei Dinge zeigen:

1. dass es in NP liegt. Dazu genügt es, eine nichtdeterministische Turingmaschine anzugeben, die das Problem in Polynomzeit löst, d.h. die die entsprechende Sprache L erkennt und in Polynomzeit arbeitet. Für die meisten Probleme geht das mit der 'Guess and Check'-Methode , d.h. mit einer Maschine, die ähnlich arbeitet wie die Maschine für das SAT-Problem: in einer ersten, nichtdeterministischen Phase wird zu einer Eingabe w etwas (ein Lösungsvorschlag, ein Beweisversuch) geraten, und in einer zweiten, deterministischen Phase wird der geratene Lösungsvorschlag überprüft. Die Maschine kann natürlich nur dann eine 'Lösung' raten, wenn eine existiert, d.h. wenn w zu L gehört.

2. dass es NP-hart ist. Dazu genügt es, von einem als NP-hart bekannten Problem L' nachzuweisen, dass es polynomial reduzierbar auf das Problem L ist, d.h. dass $L' \leq_p L$ gilt.

   Es ist nämlich jedes Problem L'' aus NP auf L' polynomial reduzierbar, d.h. es gilt $L'' \leq L'$. Da die polynomiale Reduzierbarkeitsrelation transitiv ist (Lemma 13.3), folgt $L'' \leq L$, also dass L'' auf L polynomial reduzierbar ist. Da L'' beliebig aus NP gewählt war, ist damit auch L als NP-hart erkannt.

Wir werden nun die in die in dem folgenden Diagramm aufgeführten NP-vollständigen Probleme vorstellen und von einigen zeigen, dass sie tatsächlich NP-vollständig sind. Die Pfeile in dem Diagramm geben mögliche Reduktionen an, mit denen man zeigen kann, dass die Probleme NP-hart sind, da SAT schon als NP-hart bekannt ist (Satz 13.8). Einige der Reduktionen werden wir durchführen. Die restlichen findet man in (Schöning: Theoretische Informatik — kurzgefasst).



Im Einzelnen handelt es sich um die folgenden Probleme (nach Schöning, S. 164-165):

Satz 14.1 (NP-vollständige Probleme)**.** NP*-vollständig sind:*

3KNF-Sat

- *gegeben: Eine Boolesche Formel* F *in konjunktiver Normalform (Klauselform) mit höchstens* 3 *Literalen pro Klausel.*
- gefragt: *Ist* F *erfüllbar?*

CLIQUE

- gegeben: *Ein ungerichteter Graph* $G = (V, E)$ *und eine Zahl* $k \in \mathbb{N}$
- gefragt: *Besitzt* G *eine 'Clique' der Größe mindestens* k*?*

(Eine 'Clique' der Größe k *ist eine Menge* $V' \subseteq V$ *mit* $|V'| \geq k$ *und* $\{u, v\} \in E$ *für alle* $u, v \in V'$ *mit* $u \neq v$*.)*

KNOTENÜBERDECKUNG

- gegeben: *Ein ungerichteter Graph* $G = (V, E)$ *und eine Zahl* $k \in \mathbb{N}$
- gefragt: *Besitzt* G *eine 'überdeckende Knotenmenge' der Größe höchstens* k*?*

(Eine 'Knotenüberdeckung' der Größe k *ist eine Menge* $V' \subseteq V$ *mit* $|V'| \leq k$*, so dass für alle* $\{u, v\} \in E$ *gilt:* $u \in V'$ *oder* $v \in V'$*.)*

RUCKSACK *(oder* ***SUBSET SUM****)*

- gegeben: *Natürliche Zahlen* $a_1, a_2, ..., a_k, b \in \mathbb{N}$
- gefragt: *Gibt es eine Teilmenge* $J \subseteq \{1, 2, ..., k\}$ *mit*

$$\sum_{i \in J} a_i = b$$

PARTITION

- gegeben: *Natürliche Zahlen* $a_1, a_2, ..., a_k \in \mathbb{N}$
- gefragt: *Gibt es eine Teilmenge* $J \subseteq \{1, 2, ..., k\}$ *mit*

$$\sum_{i \in J} a_i = \sum_{i \notin J} a_i$$

BIN PACKING

- gegeben: *Eine 'Behältergröße'* $b \in \mathbb{N}$*, die Anzahl* $k \in \mathbb{N}$ *der Behälter, Objekte* $a_1, a_2, ..., a_n \in \mathbb{N}$*.*
- gefragt: *Können die Objekte so auf die* k *Behälter verteilt werden, dass kein Behälter überläuft?*

(Gibt es eine Abbildung $f : \{1, ..., n\} \to \{1, ..., k\}$*, so dass für alle* $j \in \{1, ..., k\}$ *gilt* $\sum_{f(i)=j} a_i \leq b$*?)*

GERICHTETER HAMILTON-KREIS

- gegeben: *Ein gerichteter Graph* $G = (V, E)$*.*
- gefragt: *Besitzt* G *einen Hamilton-Kreis?*

(D.h. gibt es Permutation π *der Knotenindizes* $(v_{\pi(1)}, v_{\pi(2)}, ..., v_{\pi(n)})$*, so dass stets* $(v_{\pi(i)}, v_{\pi(i+1)}) \in E$ *und* $(v_{\pi(n)}, v_{\pi(1)}) \in E$*)*

UNGERICHTETER HAMILTON-KREIS

- gegeben: *Ein ungerichteter Graph* $G = (V, E)$*.*

- gefragt: *Besitzt G einen Hamilton-Kreis?*

(D.h. gibt es Permutation π der Knotenindizes $(v_{\pi(1)}, v_{\pi(2)}, ..., v_{\pi(n)})$, so dass stets $\{v_{\pi(i)}, v_{\pi(i+1)}\} \in E$ und $\{v_{\pi(n)}, v_{\pi(1)}\} \in E$)

TRAVELING SALESMAN

- gegeben: *Eine $n \times n$-Matrix $(M_{i,j})$ von 'Entfernungen' zwischen n 'Städten' und eine Zahl k.*
- gefragt: *Gibt es Permutation π (eine 'Rundreise'), so dass*

$$\sum_{i=1}^{n-1} M_{v_{\pi(i)}, v_{\pi(i+1)}} + M_{v_{\pi(n)}, v_{\pi(1)}} \leq k$$

FÄRBBARKEIT

- gegeben: *Ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}$*
- gefragt: *Gibt es eine Färbung der Knoten in V mit k verschiedenen Farben, so dass keine zwei benachbarten Knoten in G dieselbe Farbe haben.*

Für jedes dieser Probleme ist der Nachweis, dass es in NP liegt, leicht mit der Guess and Check-Methode zu führen. Dazu zeigt man, dass man bei jedem dieser Probleme auf eine Eingabe hin einen Vorschlag für eine Lösung in Polynomzeit raten und diesen Vorschlag in Polynomzeit verifizieren kann. In allen Fällen handelt es sich um arithmetische oder Graphenprobleme mit Summenbildungen. Daher ist dies für jedes Problem unmittelbar einsichtig. Wir beschränken uns daher jeweils auf die entsprechende Reduktion in dem Diagramm oben, um nachzuweisen, dass das jeweilige Problem NP-hart ist.

Satz 14.2. ***3KNF-SAT*** *ist NP-vollständig.*

Zur Erinnerung: ein *Literal* ist eine Variable oder eine negierte Variable. Eine *Klausel* ist eine Disjunktion von Literalen. Eine aussagenlogische Formel ist in *konjunktiver Normalform* (**KNF**), wenn sie eine Konjunktion von Klauseln ist. Eine Formel ist in **3KNF**, wenn sie eine Konjunktion von Klauseln ist, von denen jede höchstens drei Literale enthält.

Beweis: Wir müssen **SAT** auf **3KNF-SAT** polynomial reduzieren. Ausgehend von einer beliebigen Booleschen Formel F müssen wir dazu eine konjunktive Form F' mit höchstens 3 Literalen pro Klausel angeben, so dass gilt

$$F \text{ erfüllbar } \Leftrightarrow F' \text{ erfüllbar}$$

Das bedeutet nicht Äquivalenz im strengen Sinne, sondern Erfüllbarkeitsäquivalenz. Wir haben es hier also nicht mit einer allgemeinen Umrechnung in eine konjunktive Form zu tun, sondern nur mit einer Umformung, die lediglich die Erfüllbarkeit erhalten muss.

Beispiel:

$$\begin{aligned}
F = F_0 &= \neg(\neg(x_1 \vee \neg x_3) \vee x_2) \\
\text{Schritt 1: } F_1 &= ((x_1 \vee \neg x_3) \wedge \neg x_2 \\
\text{Schritt 2: } F_2 &= (y_1 \wedge \neg x_2) \\
&\quad \wedge (y_1 \Leftrightarrow (x_1 \vee \neg x_3)) \\
F_3 &= y_2 \\
&\quad \wedge [y_2 \Leftrightarrow (y_1 \wedge \neg x_2)] \\
&\quad \wedge [y_1 \Leftrightarrow (x_1 \vee \neg x_3)] \\
\text{Schritt 3: } F_4 &= y_2 \\
&\quad \wedge (\neg y_2 \vee y_1) \wedge (\neg y_2 \vee \neg x_2) \wedge (y_2 \vee \neg y_1 \vee x_2) \\
&\quad \wedge (y_1 \vee \neg x_1) \wedge (y_1 \vee x_3) \wedge (\neg y_1 \vee x_1 \vee \neg x_3)
\end{aligned}$$

Stets: F_i erfüllbar $\Leftrightarrow$ F_{i+1} erfüllbar, damit: F erfüllbar $\Leftrightarrow$ F_4 erfüllbar

Aufwand der Umformungen: Polynomial in der Länge von F

Anmerkung: Das entsprechende Problem **2KNF-SAT** mit je maximal zwei Literalen liegt bereits in P...

Satz 14.3. *Das **CLIQUE**-Problem ist NP-vollständig.*

Beweis: Zeige **3KNF-SAT** $\leq_p$ **CLIQUE**

Sei F eine Formel in **3KNF**. Da man eine Klausel mit nur einem Literal z durch die Klausel $(z \vee z \vee z)$ und eine Klausel $(y \vee z)$ mit nur zwei Literalen durch die Klausel $(y \vee z \vee z)$ ersetzen kann, können wir annehmen, dass F genau drei Literale pro Klausel hat:

$$F = (z_{11} \vee z_{12} \vee z_{13}) \wedge ... \wedge (z_{m1} \vee z_{m2} \vee z_{m3})$$

Dabei sind die z_{jk} aus $\{x_1, ..., x_n\} \cup \{\neg x_1, ..., \neg x_n\}$.

Der Formel F wird nun ein Graph auf die folgende Weise zugeordnet:

$G = (V, E)$, $V = \{(1,1), (1,2), (1,3), ..., (m,1), (m,2), (m,3)\}$, $E = \{ \{(i,p), (j,q)\} \mid i \neq j \wedge z_{ip} \neq \neg z_{jq}\}$

Anmerkung: Die Vervielfachung der Literale ist nur zur leichteren Formulierung notwendig...)

Als Beispiele Graphen zu:

- $$(x \vee y) \wedge (\neg x \vee y) \wedge (x \vee \neg y)$$
- $$x \wedge y \wedge (\neg x \vee \neg y)$$

Außerdem ordnen wir der Formel F die Zahl $k := m$ zu.

Dann ist F durch eine Belegung erfüllbar

- genau dann, wenn es in jeder Klausel ein Literal gibt, das den Wert 1 erhält, z.B. $z_{1p_1}, ..., z_{mp_m}$,
- genau dann, wenn es Literale $z_{1p_1}, ..., z_{mp_m}$ gibt mit $z_{ip_i} \neq \neg z_{jp_j}$ für $i \neq j$
- genau dann, wenn es Knoten $(1, p_1), (2, p_2), ..., (m, p_m)$ in G gibt, die paarweise verbunden sind,
- genau dann, wenn es in G eine Clique der Größe $k = m$ gibt.

Satz 14.4. ***KNOTENÜBERDECKUNG** ist NP-vollständig.*

Beweis: Man kann **CLIQUE** leicht nach **KNOTENÜBERDECKUNG** reduzieren: Der Graph $G = (V, E)$ und die Zahl k werden abgebildet auf den Komplementgraphen $G' = (V, V^2 \setminus E)$ und die Zahl $|V| - k$.

Satz 14.5. ***RUCKSACK** ist NP-vollständig.*

Zum Beweis reduzieren wir **3KNF-SAT** auf **RUCKSACK**. Sei F eine Formel in **3KNF**. Wie oben können wir annehmen, dass jede Klausel in F genau drei Literale enthält:

$$F = (z_{11} \vee z_{12} \vee z_{13}) \wedge ... \wedge (z_{m1} \vee z_{m2} \vee z_{m3})$$

Dabei sind die z_{jk} aus $\{x_1, ..., x_n\} \cup \{\neg x_1, ..., \neg x_n\}$.

Wir müssen nun Zahlen $a_1, ..., a_k$ und b angeben, so dass die Summe der a_i gerade b ergibt. Die Zahl b ist gegeben durch $4...41...1$ (m-mal die Zahl 4, n-mal die 1 im Dezimalsystem).

Die Menge $\{a_1, ..., a_k\}$ der Zahlen a_i setzt sich aus vier verschiedenen Klassen von Zahlen zusammen: Zahlen $v_1, ..., v_n$, Zahlen $v'_1, ..., v'_n$, Zahlen $c_1, ..., c_m$ und Zahlen $d_1, ..., d_m$.

Die Zahlen v_k werden folgendermaßen definiert. Geschrieben als Dezimalzahl wie die Zahl b steht an der i-ten Position (von links) der Zahl v_k die Anzahl des Vorkommens der Variablen x_k in positiver Form in der Klausel i. Im hinteren Ziffernblock wird der Index k der Variablen $(1...n)$ in der Stellencodierung angezeigt: die erste Stelle wird 1 für x_1, die zweite für x_2 etc., die anderen bleiben 0. Analog werden dann Zahlen v'_k für das negative Vorkommen der Variablen x_k in der Klausel i definiert.

Die Zahlen c_j und d_j für $j = 1, .., m$ sind analog aufgebaut und haben im ersten Ziffernblock an der $j-$ten Stelle einfach eine 1 bzw. eine 2.

Beispiel:

$$F = (x_1 \vee \neg x_3 \vee x_5) \wedge (\neg x_1 \vee x_4 \vee x_5) \wedge (\neg x_2 \vee \neg x_2 \vee \neg x_5)$$

mit $m = 3$, $n = 5$ und

$v_1 = 100\,10000$	$v'_1 = 010\,10000$
$v_2 = 000\,01000$	$v'_2 = 002\,01000$
$v_3 = 000\,00100$	$v'_3 = 100\,00100$
$v_4 = 010\,00010$	$v'_4 = 000\,00010$
$v_5 = 110\,00001$	$v'_5 = 001\,00001$
$c_1 = 100\,00000$	$d_1 = 200\,00000$
$c_2 = 010\,00000$	$d_2 = 020\,00000$
$c_3 = 001\,00000$	$d_3 = 002\,00000$

Nun zeigen wir folgendes: Wenn F eine erfüllende Belegung besitzt, so lässt sich eine Auswahl der Zahlen treffen, die sich zu b aufsummiert. Dabei nehme man v_k bzw. v'_k in die Auswahl auf, falls $x_k = 1$ bzw. $x_k = 0$ in der Belegung gilt. Dazu beachte man, dass zu jedem Index nur v_k oder v'_k, aber nicht beide auftreten können. Daher ergibt die Summe der letzten n Ziffern immer genau $1..1$. Schauen wir nun die ersten m Ziffern an. Sie spiegeln das Vorkommen von x_k bzw. $\neg x_k$ in den m Klauseln wider. In den einzelnen Ziffern summieren sie sich zu 1, 2 oder 3 auf. Ergänzt man dies in geeigneter Weise mit c_j und/oder d_j, so ergibt sich in jeder Ziffer gerade 4.

Sei nun umgekehrt eine Auswahl von Zahlen getroffen derart, dass die Summe gerade b ergibt. Dann muss diese Auswahl für jeden Index k aus $\{1, \dots n\}$ entweder v_k oder v'_k enthalten. Da man durch Summation von c_j und d_j allein auf höchstens den Wert 3 an der Stelle j (von links) kommt, muss auch für jeden Index j aus $\{1, \dots, m\}$ (also jeden Index einer Klausel) mindestens eine Zahl v_k oder v'_k ausgewählt sein, so dass die Summe von c_j und d_j in der Ziffer j zu 4 ergänzt wird. Das zugehörige Literal muss dann in der Klausel auftreten. Also erhält man aus der Auswahl v_k oder v'_k eine erfüllende Belegung der Formel F.

Sei andererseits

$$\sum_{a \in A} a = b = \underbrace{4...4}_{m}\underbrace{1...1}_{n}$$

Mit $c_1 + ... + c_m + d_1 + ... + d_m = \underbrace{3...3}_{m}\underbrace{0...0}_{n}$ gilt

- entweder v_k in A oder v'_k in A für $1 \leq k \leq n$
- v_k, v'_k nicht beide gleichzeitig in A!
- Summe der v_k, v'_k in A: $z_1...z_m \underbrace{1...1}_{n}$ mit $z_i \geq 0$

Also: F erfüllt durch Belegung mit

$$x_k = 1 \Leftrightarrow v_k \in A$$

Satz 14.6. ***PARTITION*** *ist* $\mathbf{NP}$*-vollständig.*

Es gibt eine elementare Reduktion von **RUCKSACK** auf **PARTITION**. Sei $(a_1, ..., a_k, b)$ ein Rucksackproblem und $M = a_1 + ... + a_k$, dann bilden wir ab:

$$(a_1, ..., a_k, b) \mapsto (a_1, ..., a_k, M - b + 1, b + 1)$$

Ist die Indexmenge I aus $\{1, ..., k\}$ eine Lösung des Problems **RUCKSACK**, so ist $I \cup \{k + 1\}$ eine Lösung von **PARTITION**, denn es gilt

$$\sum_{I} a_i + M - b + 1 = b + M - b + 1 = M - b + b + 1 = \sum_{CI} a_i + b + 1$$

Dabei bezeichnet CI das Komplement von I.

Gibt es andererseits eine Lösung J von **PARTITION**, so können die beiden neuen Indizes $k + 1$ und $k + 2$ nicht beide in J oder beide im Komplement von J liegen, da die Summe der Zahlen mit Indizes in dieser Menge dann automatisch mindestens $M - b + 1 + b + 1 = M + 2$ wäre, also größer wäre als die Summe der Zahlen mit Indizes in der anderen Menge. Diejenige unter den beiden Indexmengen J und 'Komplement von J', die den Index $k + 1$ enthält, ist ohne diesen Index eine Lösung von **RUCKSACK**.

Satz 14.7. ***BIN-PACKING*** *ist* $\mathbf{NP}$*-vollständig.*

Es gibt eine sehr einfache Reduktion von **PARTITION** auf **BIN-PACKING**: $(a_1, ..., a_k)$ sei dabei die Eingabe für **PARTITION**.

Falls $\sum_{1 \leq i \leq k} a_i$ ungerade ist, kann es keine Lösung geben, dann wählen wir eine eine Eingabe für **BIN-PACKING**, die keine Lösung hat.

Ansonsten setzen wir die Behältergröße mit $b = \sum_{1 \leq i \leq k} a_i / 2$ an und wählen 2 Behälter und die Zahlen $(a_1, ..., a_k)$.

Für die verbleibenden Reduktionen in dem Diagramm oben, mit denen man die folgenden vier Sätze beweist, sei auf Schöning verwiesen.

Satz 14.8. *Das* ***GERICHTETE HAMILTON-KREIS*** *Problem* GHK *ist* $\mathbf{NP}$*-vollständig.*

Satz 14.9. *Das* ***UNGERICHTETE HAMILTON-KREIS*** *Problem* UHK *ist* $\mathbf{NP}$*-vollständig.*

Satz 14.10. *Das* ***TRAVELING SALESMAN****-Problem* TSP *ist* $\mathbf{NP}$*-vollständig.*

Satz 14.11. *Das* ***Färbbarkeits****problem ist* $\mathbf{NP}$*-vollständig.*

Zum Abschluss wollen wir noch einige Probleme untersuchen, die mit formalen Sprachen zu tun haben.

Satz 14.12. *Das Wortproblem für monotone Grammatiken ist* $\mathbf{NP}$*-hart.*

Es sei angemerkt, dass nich bekannt ist, ob das Wortproblem für monotone Grammatiken in **NP** liegt. Tatsächlich ist das Problem sogar **PSPACE**-vollständig. Das heißt, erstens ist es mit polynomialem Speicherplatz lösbar, und zweitens können alle mit polynomialem Speicherplatz lösbaren Problemen auf dieses Problem polynomial reduziert werden. Man kann leicht sehen, dass **NP** in **PSPACE** enthalten ist. Ob die Umkehrung gilt, ist unbekannt. Es wird vermutet, dass **NP** eine echte Teilmenge von **PSPACE** ist.

Satz 14.13. *Das Problem* ***Regulär-Inäquivalenz****, für zwei reguläre Ausdrücke festzustellen, ob sie inäquivalent sind, d.h. ob die zugehörigen Sprachen nicht identisch sind, ist* $\mathbf{NP}$*-hart.*