Algorithmen für das SAT-Problem

Skript zur Vorlesung im SS 2005
gehalten von Dr. Jan Johannsen

Version vom 26. April 2005

1 Grundlagen

1.1 Syntax der Aussagenlogik

Formeln der Aussagenlogik sind zusammengesetzt aus Variablen, von denen es eine abzählbar unendliche Menge Var gibt, mittels der logischen Zeichen $\neg, \vee, \wedge$, sie sind induktiv definiert wie folgt :

1. Die Konstanten 0 und 1 sind Formeln.
2. Jede Variable $x \in \mathrm{Var}$ ist eine Formel.
3. Ist F eine Formel, so ist auch $\neg F$ eine Formel.
4. Sind F und G Formeln, so sind auch $F \vee G$ sowie $F \wedge G$ Formeln.

Variablen bezeichnen wir im Allgemeinen mit x, y, z und Formeln mit F, G, H, beide evtl. mit Indizes. Die Menge der in F vorkommenden Variablen wird mit $V(F)$ bezeichnet und kann induktiv definiert werden durch:

$$
\begin{aligned}
V(0) &= \emptyset & V(1) &= \emptyset \\
V(x) &= \{x\} & V(\neg F) &= V(F) \\
V(F \wedge G) &= V(F) \cup V(G) & V(F \vee G) &= V(F) \cup V(G)
\end{aligned}
$$

Die Größe $|F|$ einer Formel F ist die Anzahl der in F vorkommenden Zeichen, und kann induktiv definiert werden durch:

$$
\begin{aligned}
|0| &= 1 & |1| &= 1 \\
|x| &= 1 & |\neg F| &= |F| + 1 \\
|F \wedge G| &= |F| + |G| + 1 & |F \vee G| &= |F| + |G| + 1
\end{aligned}
$$

1.2 Semantik der Aussagenlogik

Eine *Bewertung* ist eine endliche partielle Abbildung α, die Variablen $x \in \mathrm{Var}$ einen Wahrheitswert $\alpha(x) \in \{0, 1\}$ zuordnet. Wir beschreiben eine Bewertung durch eine Liste von Wertzuweisungen

$$[x_1 \leftarrow \epsilon_1, \dots, x_k \leftarrow \epsilon_k]$$

für paarweise verschiedene Variable $x_1, \dots, x_k$ und Werte $\epsilon_1, \dots, \epsilon_k \in \{0, 1\}$. Bewertungen bezeichnen wir im Allgemeinen mit α, β, γ, evtl. mit Indizes.

Der Definitionsbereich von α wird mit $\mathrm{dom}\,\alpha$ bezeichnet. Eine Bewertung α heißt *total* für F, falls sie jede in F vorkommende Variable bewertet, also $V(F) \subseteq \mathrm{dom}\,\alpha$ ist, andernfalls heißt sie *partiell.*

Der Wert $F\alpha$ einer Formel F unter der Bewertung α wird errechnet, indem jede Variable $x \in \mathrm{dom}\,\alpha$ in F durch die Konstante $\alpha(x)$ ersetzt werden, und dann gemäß der folgenden Regeln vereinfacht wird:

$$\begin{array}{lll} \neg 0 \rightsquigarrow 1 & F \wedge 0\,,\; 0 \wedge F \rightsquigarrow 0 & F \vee 0\,,\; 0 \vee F \rightsquigarrow F \\ \neg 1 \rightsquigarrow 0 & F \wedge 1\,,\; 1 \wedge F \rightsquigarrow F & F \vee 1\,,\; 1 \vee F \rightsquigarrow 1 \end{array}$$

Falls α nicht total für F ist, ist $F\alpha$ eine Formel mit $V(F\alpha) \subseteq V(F) \setminus \mathrm{dom}\,\alpha$, kann aber auch ein Wert $F\alpha \in \{0, 1\}$ sein, z.B. $(x \vee y)[x \leftarrow 1] = 1$.

Eine Bewertung β *erweitert* α, in Zeichen $\beta \supseteq \alpha$, falls $\mathrm{dom}\,\alpha \subseteq \mathrm{dom}\,\beta$ ist, und für alle $x \in \mathrm{dom}\,\alpha$ gilt $\alpha(x) = \beta(x)$.

Falls für eine Formel F und Bewertung α gilt $F\alpha = 1$, so sagen wir auch α *erfüllt* F und schreiben dafür $\alpha \models F$.

Eine Formel F heißt

- *erfüllbar*, wenn es eine Bewertung α gibt mit $\alpha \models F$.
- *gültig* oder *Tautologie*, wenn $\alpha \models F$ für jede für F totale Bewertung α gilt.

Zwei Formeln F und G heißen *äquivalent*, in Zeichen $F \equiv G$, falls für jede Bewertung α, die für F und G total ist, gilt $F\alpha = G\alpha$. Dies läßt sich auch ausdrücken wie folgt: für jede totale Bewertung α gilt $\alpha \models F$ genau dann, wenn $\alpha \models G$.

Zwei Formeln F und G heißen erfüllbarkeits-äquivalent, in Zeichen $F \approx G$, falls gilt: F ist erfüllbar genau dann wenn G ist erfüllbar.

1.3 Normalformen

Ein *Literal* ist eine Variable x oder eine negierte Variable $\neg x$. Eine Variable x heißt auch *positives* Literal und eine negierte Variable $\neg x$ *negatives* Literal.

Die Literale x und $\neg x$ sind *komplementär* zueinander. Literale bezeichnen wir im Allgemeinen mit $a, b, c, \ldots$, evtl. mit Indizes.

Die Notation für Bewertungen wird wie folgt auf Literale erweitert: Ist a die Variable x, so bedeutet $[a \leftarrow \epsilon]$ dasselbe wie $[x \leftarrow \epsilon]$, und ist a die negierte Variable $\neg x$, so bedeutet $[a \leftarrow \epsilon]$ dassselbe wie $[x \leftarrow (1 - \epsilon)]$.

Negations-Normalform

Eine Formel F ist in *Negations-Normalform* (NNF), falls Negationszeichen in F nur unmittelbar vor Variablen vorkommen, also falls F der folgenden induktiven Definition genügt:

- Jedes Literal und jede Konstante ist eine Formel in NNF.
- Sind F und G Formeln in NNF, so sind auch $F \vee G$ sowie $F \wedge G$ Formeln in NNF.

Für eine Formel F in NNF definieren wir die negierte Formel $\bar{F}$ in NNF durch:

- Ist $F = 0$, so ist $\bar{F} = 1$. Ist $F = 1$, so ist $\bar{F} = 0$.
- Ist $F = x$, so ist $\bar{F} = \neg x$.
- Ist $F = \neg x$, so ist $\bar{F} = x$.
- Ist $F = G_1 \vee G_2$, so ist $\bar{F} = \bar{G}_1 \wedge \bar{G}_2$.
- Ist $F = G_1 \wedge G_2$, so ist $\bar{F} = \bar{G}_1 \vee \bar{G}_2$.

Anschaulich entsteht $\bar{F}$ aus F daurch, dass jedes Literal durch das entsprechende komplementäre Literal ersetzt wird, und die Symbole 0 und 1 sowie $\wedge$ und $\vee$ vertauscht werden.

Proposition 1. *Für jede Formel F in NNF ist $\bar{F}$ äquivalent zu $\neg F$.*

Beweis durch strukturelle Induktion nach der obigen induktiven Defintion der NNF.

- Ist $F = 0$, so ist $\bar{F} = 1 \equiv \neg 0$. Ist $F = 1$, so ist $\bar{F} = 0 \equiv \neg 1$.
- Ist $F = x$, so ist $\bar{F} = \neg x = \neg x$.
- Ist $F = \neg x$, so ist $\bar{F} = x \equiv \neg\neg x = \neg F$.
- Ist $F = G \vee H$, so ist $\bar{F} = \bar{G} \wedge \bar{H}$. Dies ist nach Induktionshypothese äquivalent zu $\neg G \wedge \neg H$, und nach dem De Morgan'schen Gesetz ist dies äquivalent zu $\neg(G \vee H) = \neg F$.

- Ist $F = G \wedge H$, so ist $\bar{F} = \bar{G} \vee \bar{H} \equiv \neg G \vee \neg H$ nach Induktionshypothese und dies ist äquivalent zu $\neg(G \wedge H) = \neg F$.

Proposition 2. *Für jede Formel F gibt es eine äquivalente Formel $n(F)$ in NNF.*

Beweis durch Induktion nach dem Formelaufbau:

- Ist $F = x$, $F = 0$ oder $F = 1$, so ist F bereits in NNF, also setze $n(F) = F$.
- Ist $F = G \circ H$ für $\circ \in \{\wedge, \vee\}$, so gibt es nach Induktionshypothese $n(G) \equiv G$ und $n(H) \equiv H$ in NNF, und wir setzen $n(F) = n(G) \circ n(H)$ und erhalten offensichtlich $n(F) = n(G) \circ n(H) \equiv G \circ H = F$.
- Ist $F = \neg G$ so gibt es nach Induktionshypothese $n(G) \equiv G$ in NNF. Definiere $n(F) = \overline{n(G)}$, nach Prop. 1 gilt dann $n(F) = \overline{n(G)} \equiv \neg n(G) \equiv \neg G = F$.

Disjunktive und Konjunktive Normalform

Definition. *Wir definieren folgende spezielle Klassen von Formeln:*

1. *Ein* Term *ist eine Konjunktion $a_1 \wedge \ldots \wedge a_k$ von Literalen.*
2. *Eine* Klausel *ist eine Disjunktion $a_1 \vee \ldots \vee a_k$ von Literalen.*
3. *Eine Formel in* disjunktiver Normalform *(DNF) ist eine Disjunktion $T_1 \vee \ldots \vee T_m$ von Termen.*
4. *Eine Formel in* konjunktiver Normalform *(KNF) ist eine Konjunktion $C_1 \wedge \ldots \wedge C_m$ von Klauseln.*

Man beachte, dass jede Formel in DNF oder KNF auch in NNF ist.

Satz 3. *Für jede Formel F gibt es äquivalente Formeln $\widehat{F}$ in KNF und $\check{F}$ in DNF.*

Beweis. Für eine Variable x seien die Literale $x^1 = x$ und $x^0 = \neg x$ definiert. Für $\epsilon = 0, 1$ gilt also, dass $\alpha \models x^\epsilon$ genau dann wenn $\alpha(x) = \epsilon$ ist.
Wir definieren nun die Formel $\check{F}$. Sei $V(F) = \{x_1, \ldots, x_n\}$. Für eine Bewertung $\alpha = [x_1 \leftarrow \epsilon_1, \ldots, x_n \leftarrow \epsilon_n]$ betrachte den Term

$$T(\alpha) = x_1^{\epsilon_1} \wedge \ldots \wedge x_n^{\epsilon_n} .$$

Für jede Bewertung β gilt dann: $\beta \models T(\alpha)$ genau dann, wenn $\beta \supseteq \alpha$.

Es seien nun $\alpha_1, \dots, \alpha_m$ alle Bewertungen mit $\operatorname{dom} \alpha_j = \{x_1, \dots, x_n\}$ und $\alpha_j \models F$ für $j = 1, \dots, m$. Wir definieren dann

$$\check{F} = T(\alpha_1) \vee \dots \vee T(\alpha_m)$$

und müssen zeigen, dass $\check{F} \equiv F$ ist.

Sei also β eine für F totale Bewertung mit $\beta \models F$, dann gibt es ein $j = 1, \dots, m$ so dass $\beta \supseteq \alpha_j$, also gilt $\beta \models T(\alpha_j)$, und somit $\beta \models \check{F}$.

Umgekehrt gelte $\beta \models \check{F}$, dann muss gelten $\beta \models T(\alpha_j)$ für ein $j = 1, \dots, m$, und somit $\beta \supseteq \alpha_j$, also nach Definition $\beta \models F$.

Die Formel $\widehat{F}$ in KNF erhält man nun wie folgt: Bilde zunächst die zu $\neg F$ äquivalente Formel G in DNF, $G = \check{\neg F}$. Insbesondere ist G in NNF, also definieren wir $\widehat{F} = \bar{G}$, und nach Prop. 1 ist $\widehat{F} \equiv \neg G \equiv F$. $\square$

Die im obigen Beweis konstruierte Formel $\check{F}$ hat die Größe $O(n2^n)$. Dass diese Konstruktion kann nicht wesentlich verbessert werden kann, zeigt die folgende untere Schranke:

Proposition 4. *Es gibt Formeln F_n in KNF mit $|F_n| = O(n)$, derart dass jede zu F_n äquivalente Formel G_n in DNF die Größe $|G_n| \geq n2^n$ hat.*

Beweis. Die Formeln F_n sind definiert durch

$$F_n = \bigwedge_{i=1}^{n} (x_{i,0} \vee x_{i,1}) \wedge (\neg x_{i,0} \vee \neg x_{i,1}) \ .$$

Eine Bewertung erfüllt die Formel F_n genau dann, wenn sie für jedes $i = 1, \dots, n$ *genau eine* der Variablen $x_{i,0}$ und $x_{i,1}$ mit 1 bewertet.

Sei $G_n = T_1 \vee \dots \vee T_m$ eine zu F_n äquivalente Formel in DNF. O.b.d.A. können wir annehmen, dass keiner der Terme T_j unerfüllbar ist, ansonsten könnte man diesen einfach weglassen und erhielte eine kleinere äquivalente Formel in DNF.

Wir zeigen zunächst, dass jeder Term in G_n für jedes $i = 1, \dots, n$ genau eine der Variablen $x_{i,0}$ oder $x_{i,1}$ positiv enthalten muss.

Ist nämlich T_j ein Term in G_n, der weder $x_{i,0}$ noch $x_{i,1}$ positiv enthält, dann wähle eine Bewertung $\alpha \models T$ mit $\alpha(x_{i,0}) = \alpha(x_{i,1}) = 0$. Ist andererseits T_j ein Term, der sowohl $x_{i,0}$ als auch $x_{i,1}$ positiv enthält, dann wähle eine Bewertung $\alpha \models T$ mit $\alpha(x_{i,0}) = \alpha(x_{i,1}) = 1$. In beiden Fällen gilt $\alpha \models G_n$, aber $\alpha \not\models F_n$.

Ferner zeigen wir, dass für jedes $\epsilon = (\epsilon_1, \dots, \epsilon_n) \in \{0,1\}^n$ ein Term in G_n existiert, der gerade die Variablen $x_{1,\epsilon_1}, \dots, x_{n,\epsilon_n}$ positiv enthält. Ist dies nämlich für ein $\epsilon \in \{0,1\}^n$ nicht der Fall, dann muss jeder Term in G_n eine der Variablen $x_{1,(1-\epsilon_1)}, \dots, x_{n,(1-\epsilon_n)}$ positiv enthalten. Dann wähle eine Bewertung α mit $\alpha(x_{i,\epsilon_i}) = 1$ und $\alpha(x_{i,(1-\epsilon_i)}) = 0$ für alle $i = 1, \dots, n$. Für diese gilt $\alpha \models F_n$, aber $\alpha \not\models G_n$.

Die Formel G_n enthält also mindestens $m \geq 2^n$ Terme, von denen jeder mindestens n Literale hat, also ist $|G_n| \geq n2^n$. □

Korollar 5. *Es gibt Formeln F'_n in DNF mit $|F'_n| = O(n)$, derart dass jede zu F_n äquivalente Formel G_n in KNF die Größe $|G_n| \geq n2^n$ hat.*

Beweis. Sei $F'_n := \bar{F}_n$ für die Formeln F_n aus Proposition 4. Diese Formeln sind in DNF, und wenn es Formeln $G_n \equiv F'_n$ in KNF mit $G_n < n2^n$ gäbe, dann wäre auch $\bar{G}_n \equiv F_n$ eine Formel in DNF mit $|\bar{G}_n| = |G_n| < n2^n$, im Widerspruch zu Proposition 4. □

Im folgenden werden vor allem Formeln in KNF für uns von Interesse sein. Für eine solche Formel F legen wir die folgenden Bezeichnungen fest:

n	die Anzahl der Variablen in F,
m	die Anzahl der Klauseln in F,
k	die maximale Anzahl von Literalen in einer Klausel in F.

Ferner identifizieren wir zur Vereinfachung der Notation Formeln in KNF mit Mengen von Klauseln. Eine Menge M von Klauseln stehe also für die Formel $F_M := \bigwedge_{C \in M} C$ und umgekehrt. Außerdem bedeute $F \setminus C$ für eine Klausel C dasselbe wie $F \setminus \{C\}$.

1.4 Das Problem SAT und seine Komplexität

Das Entscheidungsproblem gSAT ist definiert wie folgt:

Gegeben:	Eine Formel F der Aussagenlogik.
Frage:	Ist F erfüllbar?

Ohne Beweis zitieren wir das folgende zentrale Resultat:

Satz 6 (Cook 1971). *gSAT ist NP-vollständig.*

Satz 7. *Für jede Formel F gibt es eine erfüllbarkeits-äquivalente Formel $F' \approx F$ in KNF mit $|F'| = O(|F|)$. Weiter ist F' aus F in polynomieller Zeit berechenbar.*

Beweis. Wir definieren für jede Teilformel G von F eine Variable x_G und eine Menge F_G von Klauseln wie folgt:

- Ist G eine Variable x, so ist $x_G = x$ und $F_G = \emptyset$.
- Ist $G = H \vee H'$, so ist x_G eine neue Variable, und F_G enthält Klauseln, die ausdrücken, dass $x_G \leftrightarrow (x_H \vee x_{H'})$ ist, nämlich $(\neg x_G \vee x_H \vee x_{H'})$, $(\neg x_H \vee x_G)$ und $(\neg x_{H'} \vee x_G))$.
- Ist $G = H \wedge H'$, so ist x_G eine neue Variable, und F_G enthält die Klauseln $(\neg x_H \vee \neg x_{H'} \vee x_G)$, $(\neg x_G \vee x_H)$ und $(\neg x_G \vee x_{H'}))$.
- Ist $G = \neg H$, so ist x_G eine neue Variable, und F_G enthält die Klauseln $(x_G \vee x_H)$ und $(\neg x_G \vee \neg x_H)$.

Die Formel F' ist nun definiert als $\bigwedge_G F_G \wedge x_F$. Es bleibt zu zeigen, dass $F' \approx F$ ist.

Sei also $\alpha \models F'$. Durch Induktion nach dem Aufbau von F zeigen wir, dass für jede Teilformel G von F gilt $\alpha(x_G) = G\alpha$. Insbesondere ist also $F\alpha = \alpha(x_F)$, und wegen des letzten Konjunktionsglieds muss $\alpha(x_F) = 1$ sein, somit gilt $\alpha \models F$.

Der Induktionsanfang $G = x$ ist trivial. Sei also z.B. $G = H \vee H'$. Nach Induktionshypothese ist $\alpha(x_H) = H\alpha$ und $\alpha(x_{H'}) = H'\alpha$. Die Klauseln F_G erzwingen nun, dass $\alpha(x_G) = \alpha(x_H) \vee \alpha(x_{H'})$ ist, also folgt $\alpha(x_G) = H\alpha \vee H'\alpha = G\alpha$. Die übrigen Fälle sind analog.

Sei umgekehrt $\alpha \models F$. Dann wird eine Bewertung $\alpha' \supseteq \alpha$ definiert durch $\alpha'(x_G) = G\alpha$ für jede Teilformel G von F. Dann rechnet man leicht nach, dass $\alpha' \models F_G$ für alle Formeln F_G gilt. Da $F\alpha = 1$ ist, gilt auch $\alpha'(x_F) = 1$, also ist insgesamt $F'\alpha' = 1$, also $\alpha' \models F'$. □

Das Entscheidungsproblem SAT ist der folgende Spezialfall von gSAT:

Gegeben:	Eine Formel F in KNF.
Frage:	Ist F erfüllbar?

Aus Satz 7 folgt unmittelbbar, dass die Erfüllbarkeit von Formeln in KNF ebenso schwierig ist wie für allgemeine Formeln.

Korollar 8. *SAT ist NP-vollständig.*

Im Gegensatz dazu ist es für Formeln in DNF sehr einfach zu entscheiden, ob sie erfüllbar sind. Eine solche Formel F ist nämlich genau dann erfüllbar, wenn es einen Term in F gibt, der keine komplementären Literale a und $\bar{a}$ enthält. Daraus erhält man aber keinen effizienten Algorithmus für SAT, da die Umwandlung von Formeln in DNF aufwändig sein kann, wie Prop. 4 zeigt.

3-SAT

Definition. *Eine* k*-Klausel für* $k \in \mathbb{N}$ *ist eine Disjunktion von* k *Literalen.*

Eine Formel $F = C_1 \wedge \ldots \wedge C_m$ *in KNF ist in* k*-KNF für* $k \in \mathbb{N}$*, falls jede Klausel* C_j *eine* k'*-Klausel für* $k' \leq k$ *ist.*

Die k-KNF ist im Gegensatz zur vollen KNF keine echte Normalform, da nicht jede Formel in diese Form gebracht werden kann:

Proposition 9. *Für jedes* k *gibt es eine Formel* F *in* $(k+1)$*-KNF, für die es keine äquivalente Formel in* k*-KNF gibt.*

Beweis. Sei $F = x_1 \vee \ldots \vee x_{k+1}$. Angenommen, G sei eine zu F äquivalente Formel in k-KNF, also $G = C_1 \wedge \ldots \wedge C_m$. O.B.d.A. ist C_1 nicht tautologisch, und C_1 enthält höchstens k Variablen, also o.B.d.A. die Variable x_{k+1} nicht. Wähle also eine Bewertung α so, dass $\alpha(C_1) = 0$ ist, aber $\alpha(x_{k+1}) = 1$. Dann gilt $\alpha \models F$, aber $\alpha \not\models G$, im Widerspruch zur Annahme. $\square$

Das Entscheidungsproblem k-SAT ist der folgende Spezialfall von SAT:

Gegeben:	Eine Formel F in k-KNF.
Frage:	Ist F erfüllbar?

Die Probleme k-SAT für $k \geq 3$ sind ebenso schwierig wie das allgemeine Problem, wie die folgende Reduktion zeigt.

Proposition 10. *Für jede Formel* F *in KNF gibt es eine erfüllbarkeitsäquivalente Formel* F' *in 3-KNF . Weiter ist* F' *aus* F *in polynomieller Zeit konstruierbar.*

Beweis. Für jede Klausel $C = a_1 \vee \ldots \vee a_{k+1}$ wird eine Formel C' in 3-KNF wie folgt konstruiert: Seien $y_1, \ldots, y_k$ neue Variablen, dann ist C' die Formel

$$(a_1 \vee y_1) \wedge (\bar{y}_1 \vee a_2 \vee y_2) \wedge \ldots \wedge (\bar{y}_{k-1} \vee a_k \vee y_k) \wedge (\bar{y}_k \vee a_{k+1})$$

Nun gilt: Falls $\alpha \models C$, so gibt es ein $\alpha' \supseteq \alpha$ mit $\alpha' \models C'$. Ist $\alpha(a_i) = 1$, so definiere $\alpha'(y_j) = 1$ für alle $j < i$ und $\alpha'(y_j) = 0$ für alle $j \geq i$, und man prüft leicht nach, dass dann $\alpha \models C'$ gilt.

Andererseits: ist $\beta \models C'$, so auch $\beta \models C$. Denn angenommen $\beta(a_i) = 0$ für alle $i = 1, \ldots, k+1$, dann muss $\beta(y_1) = 1$ sein, damit die erste Klausel in C' erfüllt ist. Dann muss aber auch $\beta(y_2) = 1$ sein, damit die zweite Klausel erfüllt ist.

Analog schließt man für alle i, dass $\beta(y_{i+1}) = 1$ sein muss, falls $\beta(y_i) = 1$ ist. Es muss aber $\beta(y_k) = 0$ sein, damit die letzte Klausel in C' erfüllt ist. Durch Induktion ergibt sich ein Widerspruch, also muss $\beta(a_i) = 1$ für mindestens ein $i = 1, \ldots, k+1$ sein, und somit $\beta \models C$.

Für eine Formel $F = C_1 \wedge \ldots \wedge C_m$ in KNF wird nun die Formel F' definiert als $C_1' \wedge \ldots \wedge C_m'$. Nun läßt sich nach dem Obigen jede Bewertung $\alpha \models F$ zu einer Bewertung $\alpha' \models F'$ erweitern, und jede Bewertung $\beta \models F'$ erfüllt auch F, also sind F und F' erfüllbarkeits-äquivalent. $\square$

Die Konstruktion ist eine polynomielle Reduktion, also folgt:

Korollar 11. *3-SAT ist NP-vollständig.*

1.5 Einfache Spezialfälle

Wir betrachten einige Spezialfälle von Formeln, für die es effiziente Algorithmen für das Erfüllbarkeitsproblem gibt.

Horn-Formeln

Definition. *Eine Klausel heißt*

- positiv *(*negativ*), wenn sie nur positive (negative) Literale enthält,*
- Horn-Klausel*, wenn sie höchstens ein positives Literal enthält.*
- definit*, wenn sie genau ein positives Literal enthält.*

Jede Horn-Klausel ist also entweder definit oder negativ.

Eine Horn-Formel *ist eine Konjunktion von Horn-Klauseln, also eine Formel in KNF, in der jede Klausel eine Horn-Klausel ist.*

Horn-SAT ist der folgende Spezialfall von SAT:

Gegeben:	Eine Horn-Formel F.
Frage:	Ist F erfüllbar?

Satz 12. *Horn-SAT ist polynomialer Zeit* $O(nm)$ *entscheidbar.*

Beweis. Der folgende Algorithmus entscheidet, ob eine eingegebene Horn-Formel F erfüllbar ist, und gibt gegebenenfalls eine erfüllende Bewertung aus.

Horn-Sat(F)

$\alpha := []$
while $F\alpha$ enthält eine positive 1-Klausel
  wähle eine solche x
  $\alpha := \alpha \cup [x \leftarrow 1]$
if $\alpha' \models F$
  then return α'
  else return unerfüllbar

wobei α' definiert ist durch

$$\alpha'(x) = \begin{cases} 1 & \alpha(x) = 1 \\ 0 & x \notin \operatorname{dom} \alpha \end{cases} .$$

Wir zeigen zuerst, dass für jede definite Klausel $C = x_1 \vee \bar{x}_2 \vee \ldots \vee \bar{x}_k$ in F gilt $\alpha' \models C$. Ist $\alpha'(x_i) = 0$ für ein $i = 2, \ldots, k$, dann gilt $\alpha' \models C$. Andernfalls kommt zu dem Zeitpunkt, wenn alle $x_2, \ldots, x_k$ in $\operatorname{dom} \alpha$ sind, die 1-Klausel x_1 in $F\alpha$ vor, und somit setzt der Algorithmus $\alpha(x_1) = 1$. Dann gilt aber auch $\alpha' \models C$. Es bezeichne nun F_D die Menge der definiten Klauseln in F.

Wir zeigen nun, dass jede Bewertung $\beta \models F_D$ eine Erweiterung $\beta \supseteq \alpha$ ist. Andernfalls sei x die erste Variable, für die der Algorithmus $\alpha(x) = 1$ setzt, und für die $\beta(x) = 0$ ist. Zum Zeitpunkt, wo $\alpha(x) = 1$ gesetzt wird, kommt die 1-Klausel x in $F\alpha$ vor, also gibt es eine definite Klausel $C = x \vee \bar{y}_1 \vee \ldots \vee \bar{y}_k$ in F, so dass die Variablen $y_1, \ldots, y_k$ bereits in $\operatorname{dom} \alpha$ sind. Nach Wahl von x ist $\beta(y_1) = \ldots = \beta(y_k) = 1$, also ist $\beta \not\models C$.

Schließlich zeigen wir, dass F unerfüllbar ist, falls $\alpha' \not\models F$, woraus sofort die Korrektheit des Algorithmus folgt. Falls $\alpha' \not\models F$, gibt es eine negative Klausel $C = \bar{y}_1 \vee \ldots \vee \bar{y}_k$ in F mit $\alpha' \not\models C$. Also sind $y_1, \ldots, y_k \in \operatorname{dom} \alpha$, und daher ist für jede Bewertung $\beta \models F_D$ auch $\beta(C) = 0$. Damit ist F unerfüllbar.

Offenbar ist die Zahl der Schleifendurchläufe durch n beschränkt, und in jedem Schleifendurchlauf wird nach einer positiven 1-Klausel gesucht. Durch Verwendung geeigneter Datenstrukturen (s.u.) ist dies in Zeit $O(m)$ möglich, der gesamte Zeitaufwand ist also $O(nm)$. □

Insbesondere liegt also Horn-SAT in der Klasse P, und ist sogar vollständig für diese Klasse.

2-SAT

Im Kontrast zu Korollar 11 ist das Problem 2-SAT einfach:

Satz 13. *2-SAT ist in linearer Zeit* $O(m)$ *entscheidbar.*

Beweis. Für jede Formel F in 2-KNF wird ein gerichteter Graph $G(F)$ definiert wie folgt:

- Die $2n$ Knoten von $G(F)$ sind die Literale x und $\neg x$ für jede Variable $x \in V(F)$.
- In $G(F)$ gibt es eine Kante $a \to b$, wenn die Klausel $\bar{a} \vee b$ in F vorhanden ist.
- Außerdem gibt es die Kanten $\bar{a} \to a$ für jede 1-Klausel a.

Jede Zweierklausel $a \vee b$ entspricht also zwei Kanten $\bar{a} \to b$ und $\bar{b} \to a$.
Wir zeigen die folgende Behauptung:

> Ist b von a in $G(F)$ erreichbar, und ist $\alpha \models F$ mit $\alpha(a) = 1$, dann ist auch $\alpha(b) = 1$.

Es genügt, dies für den Fall zu zeigen, wo eine Kante $a \to b$ vorhanden ist, die allgemeine Behauptung folgt induktiv. Für jede soche Kante ist aber eine Klausel $\bar{a} \vee b$ in F vorhanden, die nicht erfüllt ist, wenn $\alpha(a) = 1$ und $\alpha(b) = 0$ ist.

Bezeichne nun $[a]$ für jedes Literal a die starke Zusammenhangskomponente von $G(F)$, in der der Knoten a liegt. Es gilt also: ist $\alpha \models F$, und $[a] = [b]$, so ist $\alpha(a) = \alpha(b)$. Daher ist F unerfüllbar, falls $[x] = [\bar{x}]$ für eine Variable x ist. Umgekehrt gilt: ist $[x] \neq [\bar{x}]$ für jede Variable x, dann ist F erfüllbar. Eine Bewertung $\alpha \models F$ wird wie folgt konstruiert:

Seien $[a_1], \ldots, [a_r]$ die starken Zusammenhangskomponenten in umgekehrter topologischer Ordnung (des induzierten gerichteten azyklischen Graphen).

for $j := 1$ to r do
  falls die Literale in $[a_j]$ noch unbewertet sind
    $\alpha(b) := 1$ für $b \in [a_j]$
    $\alpha(b) := 0$ für $b \in [\bar{a}_j]$

Nach der Voraussetzung ist diese Bewertung wohldefiniert. Offenbar erfüllt α alle Klauseln, die Kanten innerhalb einer Komponente entsprechen.

Sei $a \to b$ also eine Kante zwischen zwei Komponenten, und nehmen wir $\alpha \not\models \bar{a} \vee b$ an. Dann müssen die Literale in $[a]$ mit 1 und die in $[b]$ mit 0 bewertet worden sein. Deshalb liegt in der topologischen Ordnung $[\bar{a}]$ vor $[a]$ und $[\bar{b}]$ hinter $[b]$, und daher auch $[\bar{b}]$ hinter $[\bar{a}]$. Dies ist ein Widerspruch, da $G(F)$ auch die Kante $\bar{b} \to \bar{a}$ enthält.

Da die starken Zusammenhangskomponenten eines Graphen in linearer Zeit $O(n + m)$ bestimmt und topologisch angeordnet werden können, ist diese Konstruktion und der Test auf Erfüllbarkeit in linearer Zeit $O(m)$ möglich. $\square$

Der Algorithmus zeigt sogar, dass das Problem 2-SAT in der Komplexitätsklasse NL liegt, also von einem nichtdeterministischen Algorithmus mit logarithmischem Speicherplatzbedarf gelöst werden kann. Tatsächlich ist 2-SAT auch vollständig für NL (unter einem geeigneten schwachen Reduktionsbegriff).

SAT(2)

Eine Formel F in KNF ist in KNF(l) für $l \in \mathbb{N}$, wenn jede Variable in F höchstens l mal vorkommt. Das Problem SAT(l) ist das Erfüllbarkeitsproblem SAT für Formeln in KNF(l).

Proposition 14. *SAT(3) ist NP-vollständig.*

Beweis. Wir geben eine polynomielle Reduktion von SAT auf SAT(3) an. Sei F eine Formel in KNF. Für jede Variable x, die in F l mal vorkommt, wählen wir l neue Variablen $x_1, \ldots, x_l$, und ersetzen das i-te Vorkommen von x in F durch x_i, für $1 \leq i \leq l$. Das Ergebnis dieser Ersetzungen sei F'.

Damit F' zu F erfüllbarkeits-äquivalent wird, müssen wir erzwingen, dass die Variablen $x_1, \ldots, x_l$ immer gleich bewertet werden. Dazu fügen wir für jede Variable $x \in V(F)$ die Klauseln E_x hinzu:

$$E_x := (\bar{x}_1 \vee x_2) \wedge \ldots \wedge (\bar{x}_{l-1} \vee x_l) \wedge (\bar{x}_l \vee x_1)$$

Definiere nun die Formel $F'' := F' \wedge \bigwedge_{x \in V(F) E_x}$. Offenbar ist F'' in KNF(3), denn jede Variable x_i kommt in F'' dreimal vor: einmal in F', und zweimal in E_x.

Jede Bewertung $\alpha \models F$ lässt sich nun zu einer $\alpha' \models F''$ umbauen, indem man für jede Variable x setzt $\alpha'(x_i) = \alpha(x)$ für $1 \leq i \leq l$.

Umgekehrt ist $\beta \models F''$, so müssen die Variabeln x_i für ein $x \in V(F)$ alle den gleichen Wert bekommen, da $\beta \models E_x$. Also definiere $\beta'(x) := \beta(x_1)$, dann folgt $\beta' \models F$, da ja $\beta \models F'$. $\square$

Satz 15. *SAT(2) ist in linearer Zeit $O(n)$ entscheidbar.*

Beweis. Ein *markierter Graph* $G = (V, E, M)$ ist ein ungerichteter Multigraph (V, E) (es kann also mehrere Kanten zwischen zwei Knoten geben) mit einer ausgezeichneten Teilmenge $M \subseteq V$ vom *markierten* Knoten. Eine Zusammenhangskomponente von G heiße *markiert*, falls sie mindestens einen markierten Knoten enthält.

Für eine Formel F in KNF(2) definiere einen markierten Graphen $G(F)$ wie folgt:

- $G(F)$ hat m Knoten, einen v_C für jede Klausel C in F.
- Falls die Klauseln C und D komplementäre Literale x und $\bar{x}$ enthalten, so gibt es eine Kante e_x zwischen v_C und v_D.
- Enthält C ein Literal a, derart dass $\bar{a}$ in F nicht vorkommt (ein *reines* Literal), so ist v_C markiert.

Man beachte, dass es zwischen v_C und v_D mehrere Kanten geben kann, falls C und D mehrere Paare komplementärer Literale enthalten, z.B. $C = (x \vee \bar{y})$ und $D = (\bar{x} \vee y \vee z)$.

Eine Bewertung der Variablen in F kann man nun so auffassen, dass sie den Kanten in $G(F)$ eine Richtung gibt: die Kante e_x wird so gerichtet, dass sie von derjenigen Klausel, die das mit 1 bewertete Literal aus $x, \bar{x}$ enthält, zu der Klausel, die das mit 0 bewertete Literal enthält, zeigt. Eine Klausel C ist dann erfüllt, wenn in dieser Ausrichtung v_C eine ausgehende Kante hat. Da Klauseln mit reinen Literalen stets erfüllt werden können, gilt die folgende offensichtliche Behauptung:

Proposition 16. *Eine Formel F in KNF(2) ist genau dann erfüllbar, genau dann wenn die Kanten in $G(F)$ so gerichtet werden können, dass es keine unmarkierte Senke gibt.*

Dies führt uns zum folgenden Lemma:

Lemma 17. *Eine Formel F in KNF(2) ist genau dann erfüllbar, wenn jede Zusammenhangskomponente in $G(F)$ markiert ist oder einen Kreis enthält.*

Beweis. Es reicht zu zeigen, dass die Bedingung des Lemmas äquivalent zu der Bedingung in Proposition 16 ist. Sie ist offensichtlich notwendig, da in einem endlichen gerichteten Graphen jeder Weg in einer Senke oder einem Kreis enden muss.

Ferner ist die Bedingung auch hinreichend: wir konstruieren eine passende Ausrichtung der Kanten für jede Zusammenhangskomponente, die entweder markiert ist oder einen Kreis enthält.

In einer markierten Komponente wird eine Ausrichtung wie folgt vorgenommen:

- Betrachte einen Spannbaum T der Komponente.
- Jede Kante in T wird so gerichtet, dass sie von dem von v weiter entfernten zum näher an v liegenden Knoten führt.

Die übrigen Kanten können dann beliebig gerichtet werden, da jeder Knoten außer der Wurzel v bereits eine ausgehende Kante in T hat.

In einer Komponente K , die einen Kreis $v_1, \dots, v_k, v_{k+1} = v_1$ enthält, wird eine Ausrichtung wie folgt vorgenommen:

- Zuerst werden die Kanten in diesem Kreis um diesen herum gerichtet, also von v_1 zu v_2, ..., von v_{k-1} zu v_k und von v_k zu v_1.
- Sei E' die Menge der Kanten in K ohne diejenigen, die die Knoten $v_1, \dots, v_k$ verbinden (also ohne den Kreis und seine Sehnen).
- Für jedes $1 \leq i \leq k$ sei K_i die Menge der Knoten, die von v_i, aber keinem v_j für $j < i$ aus in E' erreichbar sind.
- Für jedes i sei T_i ein Spannbaum des induzierten Graphen auf K_i. Die Kanten in T_i werden wie oben auf die Wurzel v_i hin gerichtet wie oben.

Auf diese Weise erhält jeder Knoten in K eine ausgehende Kante. Alle übrigen Kanten können dann beliebig gerichtet werden. $\square$

Die Bedingung in Lemma 17 läßt sich leicht mittels Tiefensuche überprüfen, etwa mit dem folgenden Algorithmus:

```
solange noch unarkierte Knoten in G
    wähle den kleinsten unmarkierten Knoten v
    führe Tiefensuche ab v durch
    falls keinen markierten Knoten und keine rückläufige Kante gefunden
        return FALSE
return TRUE
```

Daher kann man die Erfüllbarkeit in Zeit $O(n + m)$ testen, dies ist $O(n)$ da $m \leq 2n$ ist. $\square$

Tatsächlich liegt das Problem SAT(2) sogar in der Komplexitätsklasse L, kann also von einem deterministischen Algorithmus mit logarithmischem Speicherplatzbedarf gelöst werden, und ist auch vollständig für L (wiederum unter einem geeigneten schwachen Reduktionsbegriff).