Algorithmen für das SAT-Problem

Skript zur Vorlesung im WS 02/03
gehalten von Dr. Jan Johannsen

Version vom 27. Januar 2003

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 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:

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

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

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

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

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

für paarweise verschiedene Variable $x_1, \ldots, x_k$ und Werte $\epsilon_1, \ldots, \epsilon_k \in \{0, 1\}$. 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, 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 vor Variablen vorkommen, also falls F der folgenden induktiven Definition genügt:

- Jedes Literal 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 $F = x$, 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 $\bar{F} = \bar{G}_1 \vee \bar{G}_2$.

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

Beweis durch Induktion nach der obigen induktiven Defintion der NNF.

- 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 $\tilde{F}$ in NNF.*

Beweis durch Induktion nach dem Formelaufbau:

- Ist $F = x$, so ist F bereits in NNF, also setze $\tilde{F} = x$.

- Ist $F = G \circ H$ für $\circ \in \{\wedge, \vee\}$, so gibt es nach Induktionshypothese $\tilde{G} \equiv G$ und $\tilde{H} \equiv H$ in NNF, und wir setzen $\tilde{F} = \tilde{G} \circ \tilde{H}$ und erhalten offensichtlich $\tilde{F} \equiv G \circ H = F$.

- Ist $F = \neg G$ so gibt es nach Induktionshypothese $\tilde{G} \equiv G$ in NNF. Definiere $\tilde{F} = \tilde{\tilde{G}}$, nach Prop. 1 gilt dann $\tilde{F} \equiv \neg G = F$.

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 DNF und* $\check{F}$ *in KNF.*

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 $\widehat{F}$. 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 $\alpha_1, \ldots, \alpha_m$ genau diejenigen Bewertungen der Variablen $x_1, \ldots, x_n$ mit $\alpha_j \models F$ für $j = 1, \ldots, m$. Wir definieren dann

$$\widehat{F} = T(\alpha_1) \vee \ldots \vee T(\alpha_m)$$

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

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

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

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 $F \setminus \{C\}$.

Definition. *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 *höchstens* k *Literale enthält.*

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

Proposition 4. *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 $G = C_1 \wedge \ldots \wedge C_m$. Die Klausel 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. $\square$

1.4 Das Problem SAT und seine Komplexität

Das Entscheidungsproblem SAT ist definiert wie folgt:

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

Ohne Beweis zitieren wir das folgende zentrale Resultat:

Satz 5. *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.

Proposition 6. *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 $i = 1, \ldots, n$ *genau eine* der Variablen $x_{i,0}$ und $x_{i,1}$ mit 1 bewertet.

Sei G_n eine zu F_n äquivalente Formel in DNF. Wir zeigen zunächst, dass jeder Term in G_n für jedes $i = 1, \ldots, n$ genau eine der Variablen $x_{i,0}$ oder $x_{i,1}$ positiv enthalten muss.

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

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

3-SAT

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 7. *Für jede Formel F in KNF gibt es eine erfüllbarkeits-äquivalente Formel $\check{F}$ in 3-KNF . Weiter ist $\check{F}$ aus F in polynomieller Zeit konstruierbar.*

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

Andererseits: ist $\beta \models \check{C}$, 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 $\check{F}$ definiert als $\check{C}_1 \wedge \ldots \wedge \check{C}_m$. Nun läßt sich nach dem Obigen jede Bewertung $\alpha \models F$ zu einer Bewertung $\alpha' \models \check{F}$ erweitern, und jede Bewertung $\beta \models \check{F}$ erfüllt auch F, also sind F und $\check{F}$ erfüllbarkeits-äquivalent. □

Da die Konstruktion eine polynomielle Reduktion von SAT auf 3-SAT ist, folgt somit:

Korollar 8. *3-SAT ist* **NP***-vollständig.*

1.5 Einfache Spezialfälle

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

2-SAT

Im Kontrast zu Korollar 8 ist das Problem 2-SAT einfach:

Satz 9. *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 Einerklausel 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
        α(b) := 1 für b ∈ [a_j]
        α(b) := 0 für b ∈ [ā_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 bestimmt und topologisch angeordnet werden können, ist diese Konstruktion und der Test auf Erfüllbarkeit in linearer Zeit 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).

Horn-Formeln

Definition. *Eine Klausel heißt*

- positiv *(*negativ*), wenn sie nur positive (negative) Literale enthält,*
- Einerklausel, *wenn sie aus genau einem Literal besteht,*
- 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 10. *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)
  α := []
  while Fα enthält eine positive Einerklausel
      wähle eine solche x
      α := α ∪ [x ← 1]
  if α′ ⊨ 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 Einerklausel x_1 in $F\alpha$ vor, und somit setzt der Algorithmus $\alpha(x_1) = 1$. Dann gilt aber auch $\alpha' \models C$.

Wir zeigen nun, jede Bewertung β, die alle definiten Klauseln in F erfüllt, eine Erweiterung $\beta(x) \supseteq \alpha$ 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 Einerklausel 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 β, die alle definiten Klauseln in F erfüllt, $\beta(C) = 0$. Damit ist F unerfüllbar.

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

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