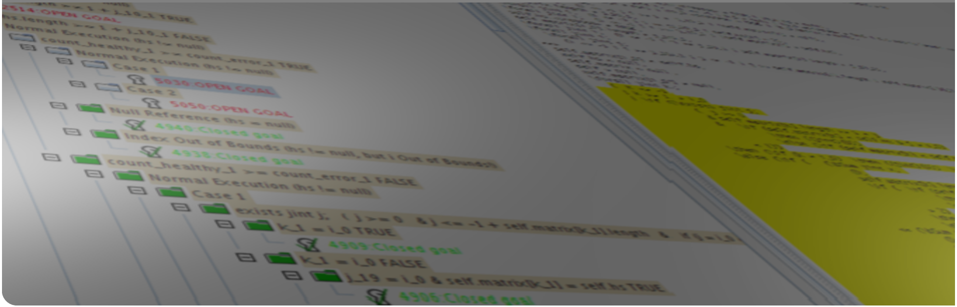
Formale Systeme

Prof. Dr. Bernhard Beckert, WS 2018/2019

Das Erfüllbarkeitsproblem

KIT – Institut für Theoretische Informatik



Das SAT Problem

oder Erfüllbarkeitsproblem

SAT

Instanz: Eine aussagenlogische Formel $F \in For0$

Frage: Ist F erfüllbar?
Gibt es eine Interpretation I mit $val_I(F) = \mathbf{W}$?

Das SAT Problem

oder Erfüllbarkeitsproblem

SAT

Instanz: Eine aussagenlogische Formel $F \in For0$

Frage: Ist F erfüllbar?
Gibt es eine Interpretation I mit $val_I(F) = \mathbf{W}$?

SAT ist ein *NP-vollständiges* Problem:

Das SAT Problem

oder Erfüllbarkeitsproblem

SAT

Instanz: Eine aussagenlogische Formel $F \in For0$

Frage: Ist F erfüllbar?
Gibt es eine Interpretation I mit $val_I(F) = \mathbf{W}$?

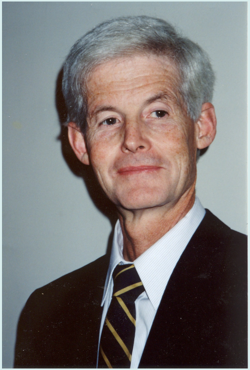
SAT ist ein *NP-vollständiges* Problem:

Gäbe es einen (deterministischen) polynomiellen Entscheidungsalgorithmus für die Erfüllbarkeit, dann wäre $NP = P$, d.h. jedes nichtdeterministisch-polynomielle Entscheidungsproblem auch deterministisch-polynomiell.

Satz von Cook

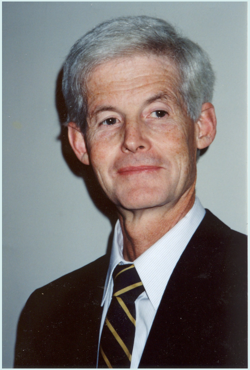
Stephen A. Cook $\star$1939

- Informatik-Professor an der Universität Toronto



Satz von Cook

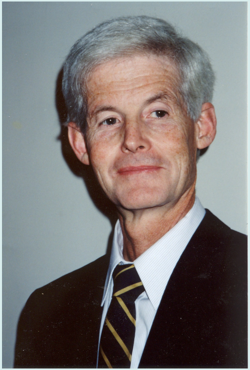
Stephen A. Cook $\star$1939



- Informatik-Professor an der Universität Toronto
- 1971: „Das Erfüllbarkeitsproblem der Aussagenlogik (SAT) ist NP-vollständig“

Satz von Cook

Stephen A. Cook $\star$1939



- Informatik-Professor an der Universität Toronto
- 1971: „Das Erfüllbarkeitsproblem der Aussagenlogik (SAT) ist NP-vollständig"
- Turing-Preisträger

Teilklassen

Das Erfüllbarkeitsproblem für Formeln

- in KNF ist NP-vollständig

Teilklassen

Das Erfüllbarkeitsproblem für Formeln

- in KNF ist NP-vollständig
- in 3-KNF ist NP-vollständig

Teilklassen

Das Erfüllbarkeitsproblem für Formeln

- in KNF ist NP-vollständig
- in 3-KNF ist NP-vollständig
- in 2-KNF ist polynomiell entscheidbar

Teilklassen

Das Erfüllbarkeitsproblem für Formeln

- in KNF ist NP-vollständig
- in 3-KNF ist NP-vollständig
- in 2-KNF ist polynomiell entscheidbar
- in DNF ist polynomiell entscheidbar ($O(n \log n)$)

Teilklassen

Das Erfüllbarkeitsproblem für Formeln

- in KNF ist NP-vollständig
- in 3-KNF ist NP-vollständig
- in 2-KNF ist polynomiell entscheidbar
- in DNF ist polynomiell entscheidbar ($O(n \log n)$)
- für Horn-Formeln ist polynomiell entscheidbar ($O(n^2)$)

Teilklassen

Das Erfüllbarkeitsproblem für Formeln

- in KNF ist NP-vollständig
- in 3-KNF ist NP-vollständig
- in 2-KNF ist polynomiell entscheidbar
- in DNF ist polynomiell entscheidbar ($O(n \log n)$)
- für Horn-Formeln ist polynomiell entscheidbar ($O(n^2)$)

Teilklassen

Das Erfüllbarkeitsproblem für Formeln

- in KNF ist NP-vollständig
- in 3-KNF ist NP-vollständig
- in 2-KNF ist polynomiell entscheidbar
- in DNF ist polynomiell entscheidbar ($O(n \log n)$)
- für Horn-Formeln ist polynomiell entscheidbar ($O(n^2)$)

Bemerkungen:

- k-KNF-Formeln sind Konjunktionen von Disjunktionen mit höchstens k Literalen.

Teilklassen

Das Erfüllbarkeitsproblem für Formeln

- in KNF ist NP-vollständig
- in 3-KNF ist NP-vollständig
- in 2-KNF ist polynomiell entscheidbar
- in DNF ist polynomiell entscheidbar ($O(n \log n)$)
- für Horn-Formeln ist polynomiell entscheidbar ($O(n^2)$)

Bemerkungen:

- k-KNF-Formeln sind Konjunktionen von Disjunktionen mit höchstens k Literalen.
- 2-KNF-Formeln heißen auch Krom-Formeln.

Horn-Formeln

Wichtigste Teilklasse mit nicht NP-vollständigem Erfüllbarkeitsproblem

Horn-Formeln

Wichtigste Teilklasse mit nicht NP-vollständigem Erfüllbarkeitsproblem

Definition

Eine aussagenlogische Formel A ist eine *Horn-Formel*, wenn

- A in KNF ist,
- jede Disjunktion in A höchstens ein positives Literal enthält

Horn-Formeln

Alternative Schreibweise:

$\neg B_1 \vee \ldots \vee \neg B_m \vee A$	$B_1 \wedge \ldots \wedge B_m \rightarrow A$
$\neg B_1 \vee \ldots \vee \neg B_m$	$B_1 \wedge \ldots \wedge B_m \rightarrow \mathbf{0}$
A	A
leere Disjunktion	$\square$

Horn-Formeln

Alternative Schreibweise:

$\neg B_1 \vee \ldots \vee \neg B_m \vee A$	$B_1 \wedge \ldots \wedge B_m \rightarrow A$
$\neg B_1 \vee \ldots \vee \neg B_m$	$B_1 \wedge \ldots \wedge B_m \rightarrow \mathbf{0}$
A	A
leere Disjunktion	$\square$

Bezeichnungen:
$B_1 \wedge \ldots \wedge B_m$: „Rumpf“ A: „Kopf“ (bei leerem Rumpf: „Fakt“)

Beispiel einer Horn-Formel

$$
\begin{array}{ll}
 & \neg P \\
\wedge & (Q \vee \neg R \vee \neg S) \\
\wedge & (\neg Q \vee \neg S) \\
\wedge & R \\
\wedge & S \\
\wedge & (\neg Q \vee P)
\end{array}
$$

Alternative Schreibweise

$$
\begin{array}{ll}
 & (P \rightarrow \mathbf{0}) \\
\wedge & (R \wedge S \rightarrow Q) \\
\wedge & (Q \wedge S \rightarrow \mathbf{0}) \\
\wedge & R \\
\wedge & S \\
\wedge & (Q \rightarrow P)
\end{array}
$$

Erfüllbarkeitsproblem für Horn-Formeln

Theorem

Für Horn-Formeln ist die Erfüllbarkeit in quadratischer Zeit entscheidbar.

Erfüllbarkeitstest

Sei $C = D_1 \wedge \ldots \wedge D_m$ eine Hornformel.
Ein Atom in C *markieren*, bedeutet, es an allen Stellen seines Auftretens in C zu markieren.

Erfüllbarkeitstest

Sei $C = D_1 \wedge \ldots \wedge D_m$ eine Hornformel.
Ein Atom in C *markieren*, bedeutet, es an allen Stellen seines Auftretens in C zu markieren.

0: Falls keine Fakten existieren: Ausgabe „erfüllbar“. STOP.
   Andernfalls: Markiere alle Fakten.

Erfüllbarkeitstest

Sei $C = D_1 \wedge \ldots \wedge D_m$ eine Hornformel.
Ein Atom in C *markieren*, bedeutet, es an allen Stellen seines Auftretens in C zu markieren.

0: Falls keine Fakten existieren: Ausgabe „erfüllbar". STOP.
   Andernfalls: Markiere alle Fakten.

1: Falls kein $B_1 \wedge \ldots \wedge B_m \rightarrow K$ existiert, so dass alle B_i im Rumpf markiert sind: Ausgabe „erfüllbar". STOP.

Erfüllbarkeitstest

Sei $C = D_1 \wedge \ldots \wedge D_m$ eine Hornformel.
Ein Atom in C *markieren*, bedeutet, es an allen Stellen seines Auftretens in C zu markieren.

0: Falls keine Fakten existieren: Ausgabe „erfüllbar“. STOP.
   Andernfalls: Markiere alle Fakten.

1: Falls kein $B_1 \wedge \ldots \wedge B_m \to K$ existiert, so dass alle B_i im Rumpf markiert sind: Ausgabe „erfüllbar“. STOP.

   Falls ein $B_1 \wedge \ldots \wedge B_m \to \mathbf{0}$ existiert mit Kopf **0**, so dass alle Atome B_i im Rumpf markiert sind:
   Ausgabe „unerfüllbar“. STOP.

Erfüllbarkeitstest

Sei $C = D_1 \wedge \ldots \wedge D_m$ eine Hornformel.
Ein Atom in C *markieren*, bedeutet, es an allen Stellen seines Auftretens in C zu markieren.

0: Falls keine Fakten existieren: Ausgabe „erfüllbar". STOP.
Andernfalls: Markiere alle Fakten.

1: Falls kein $B_1 \wedge \ldots \wedge B_m \to K$ existiert, so dass alle B_i im Rumpf markiert sind: Ausgabe „erfüllbar". STOP.

Falls ein $B_1 \wedge \ldots \wedge B_m \to \mathbf{0}$ existiert mit Kopf **0**, so dass alle Atome B_i im Rumpf markiert sind:
Ausgabe „unerfüllbar". STOP.

Falls ein $B_1 \wedge \ldots \wedge B_m \to A$ existiert, so dass alle Atome B_i im Rumpf markiert sind aber der Kopf A nicht:
Markiere A. Gehe zu 1.

Erfüllbarkeitstest

Sei $C = D_1 \wedge \ldots \wedge D_m$ eine Hornformel.
Ein Atom in C *markieren*, bedeutet, es an allen Stellen seines Auftretens in C zu markieren.

0: Falls keine Fakten existieren: Ausgabe „erfüllbar". STOP.
   Andernfalls: Markiere alle Fakten.

1: Falls kein $B_1 \wedge \ldots \wedge B_m \rightarrow K$ existiert, so dass alle B_i im Rumpf markiert sind: Ausgabe „erfüllbar". STOP.

   Falls ein $B_1 \wedge \ldots \wedge B_m \rightarrow \mathbf{0}$ existiert mit Kopf **0**, so dass alle Atome B_i im Rumpf markiert sind:
   Ausgabe „unerfüllbar". STOP.

   Falls ein $B_1 \wedge \ldots \wedge B_m \rightarrow A$ existiert, so dass alle Atome B_i im Rumpf markiert sind aber der Kopf A nicht:
   Markiere A. Gehe zu 1.

   Andernfalls: Ausgabe „erfüllbar". STOP.

Erfüllbarkeitstest

Beispiel

p

q

$q \wedge r \to s$

$p \to r$

$s \to 0$

Erfüllbarkeitstest

Beispiel

p

q

$q \wedge r \rightarrow s$

$p \rightarrow r$

$s \rightarrow 0$

Erfüllbarkeitstest

Beispiel

p

q

$q \wedge r \rightarrow s$

$p \rightarrow r$

$s \rightarrow 0$

Erfüllbarkeitstest

Beispiel

p

q

$q \wedge r \rightarrow s$

$p \rightarrow r$

$s \rightarrow 0$

Erfüllbarkeitstest

Beispiel

p

q

$q \wedge r \rightarrow s$

$p \rightarrow r$

$s \rightarrow 0$

Erfüllbarkeitstest

Beispiel

p

q

$q \land r \to s$

$p \to r$

$s \to 0$

Erfüllbarkeitstest

Beispiel

p

q

$q \wedge r \rightarrow s$

$p \rightarrow r$

$s \rightarrow 0$

Erfüllbarkeitstest

Beispiel

p

q

$q \wedge r \rightarrow s$

$p \rightarrow r$

$s \rightarrow 0$

Formelmenge nicht erfüllbar

Davis-Putnam-Logemann-Loveland-Verfahren

DPLL-Verfahren
Wichtigstes Verfahren zur Entscheidung des allgemeinen Erfüllbarkeitsproblems (SAT-Problem)

Davis-Putnam-Logemann-Loveland-Verfahren

DPLL-Verfahren
Wichtigstes Verfahren zur Entscheidung des allgemeinen Erfüllbarkeitsproblems (SAT-Problem)

Das zur Zeit schnellste und fast ausschließlich benutzte Verfahren

Davis-Putnam-Logemann-Loveland-Verfahren

DPLL-Verfahren
Wichtigstes Verfahren zur Entscheidung des allgemeinen Erfüllbarkeitsproblems (SAT-Problem)

Das zur Zeit schnellste und fast ausschließlich benutzte Verfahren

Eingabe: Formel in KNF

Exkurs: Klauselschreibweise

In Disjunktionen und Konjunktionen
kommt es nicht auf

- die Reihenfolge der Teilformeln an

Exkurs: Klauselschreibweise

In Disjunktionen und Konjunktionen
kommt es nicht auf

- die Reihenfolge der Teilformeln an
- die Multiplizität identischer Teilformeln an

Exkurs: Klauselschreibweise

In Disjunktionen und Konjunktionen
kommt es nicht auf

- die Reihenfolge der Teilformeln an
- die Multiplizität identischer Teilformeln an

Exkurs: Klauselschreibweise

In Disjunktionen und Konjunktionen
kommt es nicht auf

- die Reihenfolge der Teilformeln an
- die Multiplizität identischer Teilformeln an

Mengenschreibweise

Exkurs: Klauselschreibweise

In Disjunktionen und Konjunktionen
kommt es nicht auf

- die Reihenfolge der Teilformeln an
- die Multiplizität identischer Teilformeln an

Mengenschreibweise

Disjunktion als Menge: „Klausel“
KNF-Formel als Menge: „Klauselmenge“

Exkurs: Klauselschreibweise

In Disjunktionen und Konjunktionen
kommt es nicht auf

- die Reihenfolge der Teilformeln an
- die Multiplizität identischer Teilformeln an

Mengenschreibweise

Disjunktion als Menge: „Klausel"
KNF-Formel als Menge: „Klauselmenge"

Schreibweise:

$\square$ für die leere Klausel

Exkurs: Klauselschreibweise

In Disjunktionen und Konjunktionen
kommt es nicht auf

- die Reihenfolge der Teilformeln an
- die Multiplizität identischer Teilformeln an

Mengenschreibweise

Disjunktion als Menge: „Klausel"
KNF-Formel als Menge: „Klauselmenge"

Schreibweise:

$\square$ für die leere Klausel

$\emptyset$ für die leere Klauselmenge

Semantik der Klauselschreibweise

Semantik

I Interpretation, S Menge von Klauseln, C Klausel.

Semantik der Klauselschreibweise

Semantik

I Interpretation, S Menge von Klauseln, C Klausel.

1. $val_I(S) = \begin{cases} \mathbf{W} & \text{falls für alle } C \in S \text{ gilt: } val_I(C) = \mathbf{W} \\ \mathbf{F} & \text{sonst} \end{cases}$

Semantik der Klauselschreibweise

Semantik

I Interpretation, S Menge von Klauseln, C Klausel.

1. $$val_I(S) = \begin{cases} \mathbf{W} & \text{falls für alle } C \in S \text{ gilt: } val_I(C) = \mathbf{W} \\ \mathbf{F} & \text{sonst} \end{cases}$$
2. $$val_I(C) = \begin{cases} \mathbf{W} & \text{falls ein } L \in C \text{ existiert mit } val_I(L) = \mathbf{W} \\ \mathbf{F} & \text{sonst} \end{cases}$$

Semantik der Klauselschreibweise

Semantik

I Interpretation, S Menge von Klauseln, C Klausel.

1. $val_I(S) = \begin{cases} \mathbf{W} & \text{falls für alle } C \in S \text{ gilt: } val_I(C) = \mathbf{W} \\ \mathbf{F} & \text{sonst} \end{cases}$
2. $val_I(C) = \begin{cases} \mathbf{W} & \text{falls ein } L \in C \text{ existiert mit } val_I(L) = \mathbf{W} \\ \mathbf{F} & \text{sonst} \end{cases}$
3. $I(\emptyset) = \mathbf{W}$.

Semantik der Klauselschreibweise

Semantik

I Interpretation, S Menge von Klauseln, C Klausel.

1. $val_I(S) = \begin{cases} \mathbf{W} & \text{falls für alle } C \in S \text{ gilt: } val_I(C) = \mathbf{W} \\ \mathbf{F} & \text{sonst} \end{cases}$
2. $val_I(C) = \begin{cases} \mathbf{W} & \text{falls ein } L \in C \text{ existiert mit } val_I(L) = \mathbf{W} \\ \mathbf{F} & \text{sonst} \end{cases}$
3. $I(\emptyset) = \mathbf{W}$.
4. $I(\square) = \mathbf{F}$.

DPLL-Verfahren (Pseudocode)

procedure DPLL(Klauselmenge S)

if $S = \emptyset$ **then return** 1;
if $\square \in S$ **then return** 0;
if S enthält Einerklausel
    then choose Einerklausel $\{L\} \in S$;
        return DPLL($\mathrm{red}_{\{L\}}(S)$);
    else choose $P \in \mathrm{atom}(S)$
        return $\max\{\mathrm{DPLL}(S_P), \mathrm{DPLL}(S_{\neg P})\}$;

DPLL-Verfahren (Pseudocode)

procedure DPLL(Klauselmenge S)
if $S = \emptyset$ **then return** 1;
if $\square \in S$ **then return** 0;
if S enthält Einerklausel
  then choose Einerklausel $\{L\} \in S$;
    return DPLL($\mathrm{red}_{\{L\}}(S)$);
  else choose $P \in \mathrm{atom}(S)$
    return $\max\{\mathrm{DPLL}(S_P), \mathrm{DPLL}(S_{\neg P})\}$;

- $\mathrm{atom}(S) = \bigcup_{C \in S} \mathrm{atom}(C)$,
  wobei $\mathrm{atom}(C) = \{P \in \Sigma \mid P \in C \text{ oder } \neg P \in C\}$

DPLL-Verfahren (Pseudocode)

procedure DPLL(Klauselmenge S)
if $S = \emptyset$ **then return** 1;
if $\Box \in S$ **then return** 0;
if S enthält Einerklausel
  then choose Einerklausel $\{L\} \in S$;
    return DPLL($\text{red}_{\{L\}}(S)$);
  else choose $P \in \text{atom}(S)$
    return $\max\{\text{DPLL}(S_P), \text{DPLL}(S_{\neg P})\}$;

- $\text{atom}(S) = \bigcup_{C \in S} \text{atom}(C)$,
  wobei $\text{atom}(C) = \{P \in \Sigma \mid P \in C \text{ oder } \neg P \in C\}$
- $\text{red}_{\{L\}}(S) = \{\text{red}_{\{L\}}(C) \mid C \in S \text{ mit } L \notin C\}$

  $\text{red}_{\{L\}}(C) = C \setminus \{\overline{L}\}$, wobei $\overline{A} = \neg A$ und $\overline{\neg A} = A$ für $A \in \Sigma$

DPLL-Verfahren (Pseudocode)

procedure DPLL(Klauselmenge S)
if $S = \emptyset$ **then return** 1;
if $\square \in S$ **then return** 0;
if S enthält Einerklausel
  then choose Einerklausel $\{L\} \in S$;
    return DPLL($\mathrm{red}_{\{L\}}(S)$);
  else choose $P \in \mathrm{atom}(S)$
    return $\max\{\mathrm{DPLL}(S_P), \mathrm{DPLL}(S_{\neg P})\}$;

- $\mathrm{atom}(S) = \bigcup_{C \in S} \mathrm{atom}(C)$,
  wobei $\mathrm{atom}(C) = \{P \in \Sigma \mid P \in C \text{ oder } \neg P \in C\}$
- $\mathrm{red}_{\{L\}}(S) = \{\mathrm{red}_{\{L\}}(C) \mid C \in S \text{ mit } L \notin C\}$

  $\mathrm{red}_{\{L\}}(C) = C \setminus \{\overline{L}\}$, wobei $\overline{A} = \neg A$ und $\overline{\neg A} = A$ für $A \in \Sigma$
- $S_P = S \cup \{\{P\}\}$ und $S_{\neg P} = S \cup \{\{\neg P\}\}$.

DPLL Beispiel

Wir beginnen mit der Klauselmenge S

$$\begin{array}{ll} P_1 \vee P_2 \vee P_3 & \neg P_1 \vee P_2 \vee \neg P_4 \\ \neg P_1 \vee P_3 & \neg P_1 \vee \neg P_3 \vee P_4 \\ P_1 \vee \neg P_3 & \neg P_2 \end{array}$$

DPLL Beispiel

Wir beginnen mit der Klauselmenge S

$$\begin{array}{ll} P_1 \vee P_2 \vee P_3 & \neg P_1 \vee P_2 \vee \neg P_4 \\ \neg P_1 \vee P_3 & \neg P_1 \vee \neg P_3 \vee P_4 \\ P_1 \vee \neg P_3 & \neg P_2 \end{array}$$

Beim ersten Aufruf von DPLL(S) wird das Unterprogramm $red_{\{\neg P_2\}}(S)$ aufgerufen.

DPLL Beispiel

Wir beginnen mit der Klauselmenge S

$$
\begin{array}{ll}
P_1 \vee P_2 \vee P_3 & \neg P_1 \vee P_2 \vee \neg P_4 \\
\neg P_1 \vee P_3 & \neg P_1 \vee \neg P_3 \vee P_4 \\
P_1 \vee \neg P_3 & \neg P_2
\end{array}
$$

Beim ersten Aufruf von DPLL(S) wird das Unterprogramm $red_{\{\neg P_2\}}(S)$ aufgerufen und liefert S_1:

$$
\begin{array}{ll}
P_1 \vee P_3 & \neg P_1 \vee \neg P_4 \\
\neg P_1 \vee P_3 & \neg P_1 \vee \neg P_3 \vee P_4 \\
P_1 \vee \neg P_3 &
\end{array}
$$

rot	=	ganze Klausel entfernt
dunkelgrün	=	Literal aus Klausel entfernt
blau	=	unverändert

DPLL Beispiel

$$\begin{array}{ll} P_1 \vee P_3 & \neg P_1 \vee \neg P_4 \\ \neg P_1 \vee P_3 & \neg P_1 \vee \neg P_3 \vee P_4 \\ P_1 \vee \neg P_3 & \end{array}$$

S_1 enthält keine Einerklausel.

DPLL Beispiel

$$\begin{array}{ll} P_1 \lor P_3 & \neg P_1 \lor \neg P_4 \\ \neg P_1 \lor P_3 & \neg P_1 \lor \neg P_3 \lor P_4 \\ P_1 \lor \neg P_3 & \end{array}$$

S_1 enthält keine Einerklausel. Die Variable P_1 wird gewählt und DPLL($S_{1,0}$) und DPLL($S_{1,1}$) werden aufgerufen.

$S_{1,0}$:	$S_{1,1}$:
$P_1 \lor P_3$	$P_1 \lor P_3$
$\neg P_1 \lor \neg P_4$	$\neg P_1 \lor \neg P_4$
$\neg P_1 \lor P_3$	$\neg P_1 \lor P_3$
$\neg P_1 \lor \neg P_3 \lor P_4$	$\neg P_1 \lor \neg P_3 \lor P_4$
$P_1 \lor \neg P_3$	$P_1 \lor \neg P_3$
P_1	$\neg P_1$

DPLL Beispiel

$S_{1,0}$:
$P_1 \vee P_3$
$\neg P_1 \vee \neg P_4$
$\neg P_1 \vee P_3$
$\neg P_1 \vee \neg P_3 \vee P_4$
$P_1 \vee \neg P_3$
P_1

$red_{\{P_1\}}(S_{1,0})$

DPLL Beispiel

$S_{1,0}$:
$P_1 \vee P_3$
$\neg P_1 \vee \neg P_4$
$\neg P_1 \vee P_3$
$\neg P_1 \vee \neg P_3 \vee P_4$
$P_1 \vee \neg P_3$
P_1

$red_{\{P_1\}}(S_{1,0})$: Die Klauseln werden gestrichen.
Das Literal wird entfernt.

$$
\begin{aligned}
red_{\{P_1\}}(S_{1,0}) &= S_{2,0} \\
&= \{\neg P_4, P_3, \neg P_3 \vee P_4\}
\end{aligned}
$$

DPLL Beispiel

$$S_{2,0} = \{\neg P_4, P_3, \neg P_3 \vee P_4\}$$

Der Aufruf von $red_{\{P_3\}}(S_{2,0})$ liefert

$$\{\neg P_4, P_4\}$$

DPLL Beispiel

$$S_{2,0} = \{\neg P_4, P_3, \neg P_3 \vee P_4\}$$

Der Aufruf von $red_{\{P_3\}}(S_{2,0})$ liefert

$$\{\neg P_4, P_4\}$$

Dann:

$$red_{\{P_4\}}(\{P_4, \neg P_4\}) = \{\Box\}$$

woraus die Unerfüllbarkeit von $S_{1,0}$ folgt.

DPLL Beispiel

$S_{1,0}$:	$S_{1,1}$:
$P_1 \vee P_3$	$P_1 \vee P_3$
$\neg P_1 \vee \neg P_4$	$\neg P_1 \vee \neg P_4$
$\neg P_1 \vee P_3$	$\neg P_1 \vee P_3$
$\neg P_1 \vee \neg P_3 \vee P_4$	$\neg P_1 \vee \neg P_3 \vee P_4$
$P_1 \vee \neg P_3$	$P_1 \vee \neg P_3$
P_1	$\neg P_1$

Jetzt kommt die Abarbeitung von DPLL($S_{1,1}$) an die Reihe.

DPLL Beispiel

$S_{1,1}$:
$P_1 \vee P_3$
$\neg P_1 \vee \neg P_4$
$\neg P_1 \vee P_3$
$\neg P_1 \vee \neg P_3 \vee P_4$
$P_1 \vee \neg P_3$
$\neg P_1$

$red_{\{\neg P_1\}}(S_{1,1})$ entfernt die Klauseln, in denen $\neg P_1$ vorkommt ...

DPLL Beispiel

$$S_{1,1}:$$
$$P_1 \vee P_3$$
$$P_1 \vee \neg P_3$$

... und streicht in den restlichen P_1.

DPLL Beispiel

$$S_{1,1}:$$
$$P_1 \vee P_3$$
$$P_1 \vee \neg P_3$$

... und streicht in den restlichen P_1.
Das liefert

$$\{P_3, \neg P_3\}$$

DPLL Beispiel

$$S_{1,1}:$$
$$P_1 \vee P_3$$
$$P_1 \vee \neg P_3$$

... und streicht in den restlichen P_1.
Das liefert

$$\{P_3, \neg P_3\}$$

woraus im nächsten Schritt

$$\{\Box\}$$

entsteht,

DPLL Beispiel

$$S_{1,1}:$$
$$P_1 \vee P_3$$
$$P_1 \vee \neg P_3$$

... und streicht in den restlichen P_1.
Das liefert

$$\{P_3, \neg P_3\}$$

woraus im nächsten Schritt

$$\{\square\}$$

entsteht,
woraus die Unerfüllbarkeit von $S_{1,1}$ und damit insgesamt die Unerfüllbarkeit von S folgt.

DPLL

Theorem

DPLL

Theorem

1. Der DPLL Algorithmus terminiert für jede Eingabe.

DPLL

Theorem

1. Der DPLL Algorithmus terminiert für jede Eingabe.
2. Der DPLL Algorithmus ist korrekt und vollständig.

DPLL

Theorem

1. Der DPLL Algorithmus terminiert für jede Eingabe.
2. Der DPLL Algorithmus ist korrekt und vollständig.

DPLL

Theorem

1. Der DPLL Algorithmus terminiert für jede Eingabe.
2. Der DPLL Algorithmus ist korrekt und vollständig.
    2.1 aus DPLL(S) $= 1$ folgt, daß S erfüllbar ist und

DPLL

Theorem

1. Der DPLL Algorithmus terminiert für jede Eingabe.
2. Der DPLL Algorithmus ist korrekt und vollständig.
   2.1 aus $\mathrm{DPLL}(S) = 1$ folgt, daß S erfüllbar ist und
   2.2 ist S erfüllbar, dann gilt $\mathrm{DPLL}(S) = 1$.

DPLL

Theorem

1. Der DPLL Algorithmus terminiert für jede Eingabe.
2. Der DPLL Algorithmus ist korrekt und vollständig.
   2.1 aus DPLL(S) $= 1$ folgt, daß S erfüllbar ist und
   2.2 ist S erfüllbar, dann gilt DPLL(S) $= 1$.

DPLL

Theorem

1. Der DPLL Algorithmus terminiert für jede Eingabe.
2. Der DPLL Algorithmus ist korrekt und vollständig.
   2.1 aus DPLL(S) $= 1$ folgt, daß S erfüllbar ist und
   2.2 ist S erfüllbar, dann gilt DPLL(S) $= 1$.

Beweisidee:

Terminierung:

- Bei jeder Reduktion fällt ein Atom weg

DPLL

Theorem

1. Der DPLL Algorithmus terminiert für jede Eingabe.
2. Der DPLL Algorithmus ist korrekt und vollständig.
   2.1 aus $\mathrm{DPLL}(S) = 1$ folgt, daß S erfüllbar ist und
   2.2 ist S erfüllbar, dann gilt $\mathrm{DPLL}(S) = 1$.

Beweisidee:

Terminierung:

- Bei jeder Reduktion fällt ein Atom weg

DPLL

Theorem

1. Der DPLL Algorithmus terminiert für jede Eingabe.
2. Der DPLL Algorithmus ist korrekt und vollständig.
   2.1 aus DPLL(S) $= 1$ folgt, daß S erfüllbar ist und
   2.2 ist S erfüllbar, dann gilt DPLL(S) $= 1$.

Beweisidee:

Terminierung:

- Bei jeder Reduktion fällt ein Atom weg

Korrektheit/Vollständigkeit

- Jeder Schritt führt zu einer erfüllbarkeitsäquivalenten Klauselmenge

DPLL

Theorem

1. Der DPLL Algorithmus terminiert für jede Eingabe.
2. Der DPLL Algorithmus ist korrekt und vollständig.
   2.1 aus $\text{DPLL}(S) = 1$ folgt, daß S erfüllbar ist und
   2.2 ist S erfüllbar, dann gilt $\text{DPLL}(S) = 1$.

Beweisidee:

Terminierung:

- Bei jeder Reduktion fällt ein Atom weg

Korrektheit/Vollständigkeit

- Jeder Schritt führt zu einer erfüllbarkeitsäquivalenten Klauselmenge
- $\emptyset$ ist erfüllbar

DPLL

Theorem

1. Der DPLL Algorithmus terminiert für jede Eingabe.
2. Der DPLL Algorithmus ist korrekt und vollständig.
   2.1 aus DPLL(S) $= 1$ folgt, daß S erfüllbar ist und
   2.2 ist S erfüllbar, dann gilt DPLL(S) $= 1$.

Beweisidee:

Terminierung:

- Bei jeder Reduktion fällt ein Atom weg

Korrektheit/Vollständigkeit

- Jeder Schritt führt zu einer erfüllbarkeitsäquivalenten Klauselmenge
- $\emptyset$ ist erfüllbar
- $\{\Box\}$ ist unerfüllbar