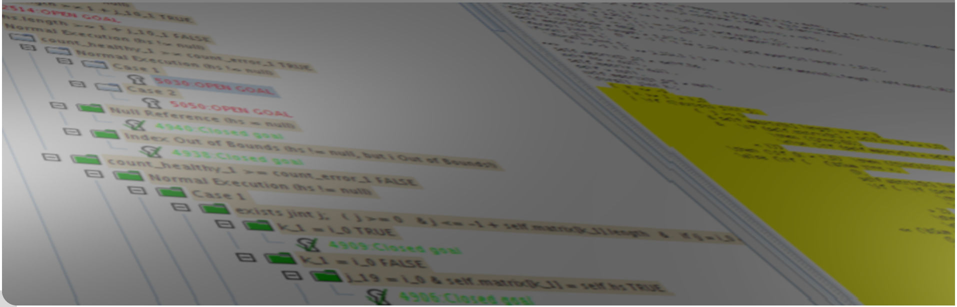
Formale Systeme

Das Erfüllbarkeitsproblem

Prof. Dr. Peter H. Schmitt

KIT – Institut für Theoretische Informatik



Das SAT Problem

oder Erfüllbarkeitsproblem

SAT

Instanz: Eine aussagenlogische Formel $F \in For0$

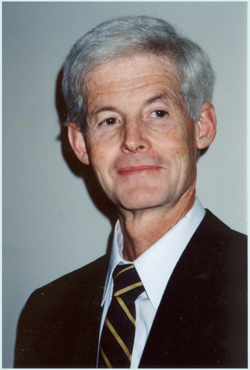
Frage: Ist F erfüllbar?
Gibt es eine Interpretation I mit $val_I(F) = \mathbf{W}$?

SAT ist ein *NP-vollständiges* Problem:

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

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 A

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

Bemerkungen:

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

Davis–Putnam–Logemann–Loveland

- Das **Davis–Putnam–Logemann–Loveland** (DPLL) Verfahren ist das zur Zeit schnellste und fast ausschließlich benutzte Verfahren in implementierten Erfüllbarkeitstests.
- Im Englischen nennt man solche Programme *SAT solver*.
- Das DPLL Verfahren benötigt als Eingabe eine KNF, also eine Konjunktion von Klauseln.
- Da es in einer Klausel nicht auf die Reihenfolge der Literale ankommt und auch das Weglassen mehrfach auftretender Literale die logische Äquivalenz erhält, werden Klauseln als Mengen von Literalen repräsentiert.
- Dasselbe gilt für Konjunktionen von Klauseln.

Notation

1. Eine Klausel K ist eine Menge von Literalen.
2. Eine KNF ist eine Menge von Klauseln.
3. $\Box$ steht für die leere Klausel.
4. $\emptyset$ für die leere Menge von Klauseln.

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. $I(\Box) = \mathbf{F}$.

DPLL Pseudocode

if $S = \emptyset$ **then** DPLL(S) $= 1$; **stop**
if $\Box \in S$ **then** DPLL(S) $= 0$; **stop**
if S enthält Einerklausel
    then choose Einerklausel $K \in S$;
        DPLL(S) $=$ DPLL($\text{red}_K(S)$);
    else choose $P \in \text{atom}(S)$
        DPLL(S) $= \mathbf{max}\{\text{DPLL}(S_P), \text{DPLL}(S_{\neg P})\}$;

- $\text{atom}(S) = \bigcup_{C \in S} \text{atom}(C)$,
  wobei $\text{atom}(C) = \{P \in \Sigma \mid P \in C \text{ oder } \neg P \in C\}$
- $\text{red}_{\{L\}}(S) = \{\text{red}_{\{L\}}(C) \mid L \not\in C\}$
  $\text{red}_{\{L\}}(C) = \{L' \mid L' \in C \text{ und } L' \neq \bar{L}\} = C \setminus \{\bar{L}\}$
- $S_P = S \cup \{\{P\}\}$ und $S_{\neg P} = S \cup \{\{\neg P\}\}$.

Der DPLL Algorithmus ist, wie man sieht, rekursiv.

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

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

DPLL Beispiel

$$
\begin{array}{l}
S_{1,0}: \\
P_1 \vee P_3 \\
\neg P_1 \vee \neg P_4 \\
\neg P_1 \vee P_3 \\
\neg P_1 \vee \neg P_3 \vee P_4 \\
P_1 \vee \neg P_3 \\
P_1
\end{array}
$$

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

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.

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

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.
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

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 Pseudocode - Terminierung

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

Beweismethode
Finde ein Maß $m(S) \in \mathbb{N}$ für Klauselmengen S mit

1. stets $0 \leq m(S)$
2. $m(red_K(S)) < m(S)$
3. $m(S_P) < m(S)$, $m(S_{\neg P}) < m(S)$

$a_u(S) = \{P \mid \{P\} \in S \text{ oder } \{\neg P\} \in S\}$
$a_{nu}(S) = atom(S) \setminus a_u(S)$
$m(S) = card(a_u(S)) + 2 * card(a_{nu}(S))$

DPLL Pseudocode - Korrektheit

falls DPLL(S) $= 1$ dann ist S erfüllbar
falls S erfüllbar dann DPLL(S) $= 1$.

if $S = \emptyset$ **then** DPLL(S) $= 1$; **stop**
if $\Box \in S$ **then** DPLL(S) $= 0$; **stop**
if S enthält Einerklausel
    then choose Einerklausel $K \in S$;
        DPLL(S) $=$ DPLL($\mathrm{red}_K(S)$);
    else choose $P \in \mathrm{atom}(S)$
        DPLL(S) $=$ **max**$\{$DPLL(S_P), DPLL($S_{\neg P}$)$\}$;

- Induktion nach $n = m(S)$.
- Für $n = 0$ gilt $S = \emptyset$ oder $S = \{\Box\}$
- Ist $\{L\} \in S$ dann gilt:
  S ist erfüllbar gdw $\mathrm{red}_{\{L\}}(S)$ ist erfüllbar
- Für $P \in \mathrm{atom}(S)$ gilt
  S ist erfüllbar gdw S_P oder $S_{\neg P}$ ist erfüllbar.

Beobachtung

Für jede Klauselmenge S und Atom P gilt

- $\mathit{red}_{\{\ P\}}(S) = S[1/P]$
- $\mathit{red}_{\{\neg P\}}(S) = S[0/P]$