

Vorlesung
– Automatisches Beweisen –
Kap. 4.2.2: SAT-Solving mit dem DPLL-Verfahren

Prof. Dr. Wolfgang Kuechlin

Dipl.-Inform., Dr. sc. techn. (ETH)

**Arbeitsbereich Symbolisches Rechnen
Wilhelm-Schickard-Institut für Informatik
Fakultät für Informations- und Kognitionswissenschaften**

Universität Tübingen

**Steinbeis Transferzentrum
Objekt- und Internet-Technologien (OIT)**

**Wolfgang.Kuechlin@uni-tuebingen.de
<http://www-sr.informatik.uni-tuebingen.de>**



SR



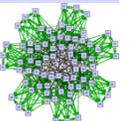
DPLL-Algorithmus

- DPLL: Davis-Putnam-Logemann-Loveland
 - Erster Algorithmus um 1960/1962, stark verbessert ab 1996.
- Entscheidungsverfahren für *SAT-Solving* Problem
 - gegeben Formel F : gibt es eine erfüllende Belegung für F ?
 - Idee: Probiere „intelligent“ eine erfüllende Belegung für F zu konstruieren, falls nötig backtracking.
- In aktueller Form das schnellste Verfahren für SAT
 - löst „gutartige“ SAT-Probleme mit Tausenden bis zu Millionen von Variablen
 - Beweise $\mathcal{F} \models F$ über $\mathcal{F} \wedge \neg F \models \perp$, also $\text{UnSAT}(\mathcal{F} \wedge \neg F)$.
 - Große Verbreitung zur Lösung industrieller (Verifikations-) Probleme, z.B. Hardware-, und Protokoll-Verifikation, KFZ-Konfiguration; Software-Verifikation im Kommen.



Grundlegender DPLL-Algorithmus

```
boolean DPLL(ClauseSet S){  
  
    //1. Bereinige S (unit constraint propagation)  
    while (S contains a unit clause { $\ell$ }) {  
        delete from S clauses containing  $\ell$ ;           // unit-subsumption  
        delete  $\neg\ell$  from all clauses in S           // unit-resolution mit subsumption  
    }  
  
    //2. Trivialfall?  
    if ( $\perp \in S$ ) return false;                          // constraint unerfüllbar  
    if (S=={}) return true;                              // nichts mehr zu erfüllen  
  
    //3. Fallunterscheidung und Rekursion  
    choose a literal  $\ell$  occurring in S;                // Heuristik (Intelligenz) gefragt!  
    if( DPLL(S  $\cup$  { $\ell$ }) ) return true;                // first recursive branch: try  $\ell := \text{true}$   
    else if ( DPLL(S  $\cup$  { $\neg\ell$ }) ) return true;        // backtracking: try  $\ell := \text{false}$   
    else return false;  
}
```



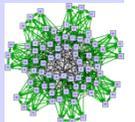
Grundlegendes zum DPLL-Algorithmus

- Keine Deduktion neuer Klauseln
 - keine dynamische Speicher-Allokation
- Algorithmus „lebt“ von der Unit-Propagation
 - UP dominiert die Laufzeit in der Praxis ($> 90\%$ UP).
 - Das muss so sein:
 - bei 100 Variablen ohne UP 2^{100} Einzelfälle
 - mit kompletter UP nur 100 Propagationen
 - Praxis-typischer Wert: 90 Propagationen, 2^{10} Einzelfälle
- Fazit für Praxis-Probleme: nur wenige Entscheidungen sind essentiell, der Rest ergibt sich als zwingende Konsequenz.



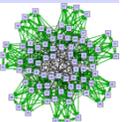
Beispiel DPLL

- $S_0 = \{\{x, y, z\}, \{\neg x, y, z\}, \{\neg x\}, \{z, \neg y\}\}$
 - unit propagation mit $\neg x$:
- $S_1 = \{\{y, z\}, \{z, \neg y\}\}$
 - Wähle z.B. y als Entscheidungsvariable:
 - Fall 1: setze $y=1$
 - $S_2 = \{\{y\}, \{y, z\}, \{z, \neg y\}\}$
 - unit propagation y ergibt $S_3 = \{z\}$, $S_4 = \{ \}$, return true.



DPLL-Algorithmus (3)

- Heuristikbeispiele für die Literalauswahl:
 - Wähle das Literal, das am häufigsten vorkommt.
 - dann schmilzt die Formel durch Evaluation an vielen Stellen
 - Wähle ein Literal L aus einer 2er-Klausel $\{K, \neg L\}$.
 - Dann $K=1$ falls $L=1$.
 - Wähle ein Literal aus einer kürzesten Klausel.
 - dann ergibt sich bald eine Unit-Klausel
 - Teste für jedes Literal L , wie sich S mittels der unit-propagation vereinfachen lässt. Wähle dasjenige L , für das S am einfachsten wird.
 - dann schmilzt die Formel vor der nächsten Fallunterscheidung insgesamt am schnellsten



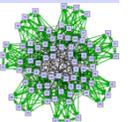
Korrektheit des Davis-Putnam-Algorithmus

- $S|_L := \{C \setminus \{\neg L\} \mid C \in S, L \in C\}$ entspricht der unit constraint propagation
 - unit Klausel $\{L\}$ subsumiert alle Klauseln C mit $L \in C$.
 - unit Klausel $\{L\}$ resolviert mit allen Klauseln $C = C_1 \cup \{\neg L\}$ zu C_1 , und C_1 wiederum subsumiert C .
- Falls $\{L\} \in S$, so ist S erfüllbar gdw. $S|_L$ erfüllbar ist.
- S erfüllbar gdw. $S \cup \{\{L\}\}$ oder $S \cup \{\{\neg L\}\}$ erfüllbar ist, für ein beliebiges Literal L .
- Termination: Anzahl n der in S vorkommenden Variablen (vor Schritt 2) nimmt bei jedem rekursiven Aufruf ab, so dass letztendlich $\square \in S$ oder $S = \{\}$ gelten muss.



Lernen im DPLL-Algorithmus

- Falls nach einer Kette von Variablenbelegungen $x_i = b_i$, $b_i \in \{0, 1\}$ die Formel $F=0$ wird ($b(F) = 0$), dann haben wir eine Nullstelle N von F gefunden.
 - N ist gegeben durch den Term $N = (\wedge\{x_i\} \wedge \{\neg y_i\})$, $b(x_i) = 1$, $b(y_i) = 0$
- Die Negation ergibt eine Klausel $C = \neg N$ in der CNF von F
 - C kann deshalb zu F hinzugefügt werden
- Eigenschaften von C
 - In C brauchen nur diejenigen Variablen aufzutauchen, die durch Entscheidungen belegt wurden (Entscheidungsvariablen)
 - die durch Propagation erzwungen belegten können wegbleiben
 - dadurch können sich kurze (=mächtige) Constraints ergeben
 - C kann auch durch Resolution hergeleitet werden, denn $F \models C$



Lernen im DPLL-Algorithmus

- Sei b eine Variablenbelegung, die im Lauf von DPLL erzeugt wird und die x belegt und y nicht belegt. Wir schreiben $b \setminus \{x\}$ für die Restriktion von b und $b \cup \{y=1\}$ für eine Erweiterung von b .
- Nutzen von $C = \neg N$:
 - Sei C mit Literal x aber ohne Literal y und $b(C)=0$; sei $b' = b \setminus \{x\}$.
 - Es ist $b(C)=0$, aber unter b' ist C die Unit-Klausel $\{x\}$.
 - Da y in C *nicht* vorkommt ist $b(C)=0$ für *jeden* Wert von y . Der Term N beschreibt also mehrere Nullstellen gleichzeitig.
 - In $b'(F \cup C)$ ist x keine Entscheidungsvariable mehr, ebensowenig mit $b' \cup \{y=1\}$ und mit $b' \cup \{y=0\}$
 - Tendenziell wird durch C eine Entscheidungsvariable durch eine UP Variable ersetzt (falls eine Belegung b' wieder auftaucht).



Lernen im DPLL-Algorithmus

➤ Nutzen von C

- Falls (z.B. nach Neustart des Verfahrens mit anderer Variablenreihenfolge) b' als Teil von einem $b'' = b' \cup \{y=1\}$ oder $b'' = b' \cup \{y=0\}$ wieder vorkommt, so wird die Belegung von x durch die Unit-Klausel $b'(C) = b''(C)$ erzwungen.
- Wiederholte Bearbeitung von Teilen des Suchbaums wird vermieden, die aufgrund derselben Ursache keine Lösung enthalten.
- Backtracking kann zum Teil unterbleiben.
- Neue Klauseln werden generiert (→ zusätzliches Wissen über die Problem Instanz)



Beispiel DPLL mit Lernen

- $S_0 = \{\{x, y\}, \{\neg y, z\}, \{\neg z, x\}\}$
 - Wähle z.B. x als Entscheidungsvariable:
 - Fall 1: setze $x=0$
 - $S_1 = \{\{\neg x\}, \{x, y\}, \{\neg y, z\}, \{\neg z, x\}\}$
 - unit propagation $\neg x$
 - $S_2 = \{\{y\}, \{\neg y, z\}, \{\neg z\}\}$
 - unit propagation y
 - $S_3 = \{\{z\}, \{\neg z\}\}$
 - unit propagation z
 - $S_4 = \{\{\}\}$, return false
- Wir lernen, dass $S_0 = 0$ falls $x=0$, also $C = \{x\}$.
 - Resolutionsbeweis von C siehe 4.2.1



Beispiel DPLL mit Lernen

- Analyse des Beispiels $S_0 = \{\{x, y\}, \{\neg y, z\}, \{\neg z, x\}\}$

| x | y | z | S_0 |
|---|---|---|-------|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

- Aus den Variablenbelegungen im vorangehenden Lauf von DPLL lernen wir: es wurde die Nullstelle $N=(x=0, y=1, z=1)$ getroffen
- Da hierbei nur x eine Entscheidungsvariable war, lernen wir weiter, dass N Teil eines Clusters aus 4 Nullstellen ist
- Zukünftige Treffer in den anderen Nullstellen des Clusters werden durch die Zusatzklausel $C = \{x\}$ vermieden



Verbessertes Lernen im DPLL-Algorithmus

- Das Lernen aus Belegungen der Entscheidungsvariablen führt i.A. zu sehr langen (und wenig nützlichen) Klauseln
 - da i.A. schon sehr viele Entscheidungen getroffen wurden
 - Verbesserung: Lerne ausgehend von Fehler-Klausel F
 - F schlägt nur wegen einer Unit-Propagation fehl
 - Ausgehend von einer Entscheidung wurde eine Klausel R (sog. „reason“-Klausel) zur Unit und dadurch wurde ein Literal $L = \text{true}$ erzwungen
 - Dadurch wurde in F ein Literal $L' = \text{false}$.
 - Also sind L und L' komplementäre Literale
 - Also haben R und F eine Resolvente ohne dieses Literal
 - Nun iteriere diesen Prozess ...
-



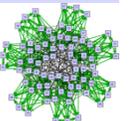
Verbessertes Lernen im DPLL-Algorithmus

- Angenommen, F enthält die letzte Entscheidungsvariable x sowie einige Literale, die als Folge der letzten Entscheidung durch UP gesetzt wurden.
- Ausgehend von F
 - ersetze in F sukzessive durch Resolution alle $UP(x)$ -Literale.
 - Die so gelernte Klausel K enthält nur noch
 - die letzte Entscheidungsvariable x
 - eine Teilmenge E der vorhergehenden Entscheidungsvariablen
 - durch UP als Folge der Entscheidungen aus E gesetzte Literale
 - andere Literale kann es nicht geben, denn diese wären unbelegt.
- Momentan ist $b(K)=\text{false}$



Verbessertes Lernen im DPLL-Algorithmus

- Nun hebt man in b alle Belegungen der letzten Entscheidungsebene auf
 - also die Entscheidung x und die Belegung aller $UP(x)$ -Variablen
- Durch Aufheben der Belegungen wird K eine unit
 - Durch die unit K wird die Alternativbelegung von x erzwungen
 - ersetzt das backtracking
 - Bem.: falls x nicht in F vorkommt, lässt man eine $UP(x)$ -Variable u in K übrig, und diese übernimmt die Rolle von x .
- Die so gelernten Klauseln sind viel kürzer und mächtiger
- Das verbesserte Lernen war der große Durchbruch für DPLL i.d. Praxis



Beispiel DPLL mit verbessertem Lernen

- $S_0 = \{\{x, y\}, \{\neg y, z\}, \{\neg z, x\}\}$
 - Wähle z.B. x als Entscheidungsvariable:
 - Fall 1: setze $x=0$
 - $S_1 = \{\{\neg x\}, \{x, y\}, \{\neg y, z\}, \{\neg z, x\}\}$
 - unit propagation $\neg x$ (Reason für $\neg x$ ist Entscheidung $x=0$)
 - $S_2 = \{\{y\}, \{\neg y, z\}, \{\neg z\}\}$
 - unit propagation y (Reason für y ist $R_1=\{x, y\}$).
 - $S_3 = \{\{z\}, \{\neg z\}\}$.
 - unit propagation z (Reason für z ist $R_2=\{\neg y, z\}$)
 - $S_4 = \{\{\}\}$, return false (Fehler-Klausel ist $F=\{\neg z, x\}$)
- Es ist $b(\{\neg z, x\}) = 0$. Resolution mit $R_2 = \{\neg y, z\}$ ergibt $\{\neg y, x\}$, weitere Resolution mit $R_1 = \{x, y\}$ ergibt $K=\{x\}$.



Beispiel DPLL mit verbessertem Lernen

- Analyse des Beispiels $S_0 = \{\{x, y\}, \{\neg y, z\}, \{\neg z, x\}\}$

| x | y | z | S_0 |
|---|---|---|-------|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

- Aus der Fehlerklausel $F = \{\neg z, x\}$ im Lauf von DPLL lernen wir: es wurde der Nullstellen-Cluster $N = (x=0, z=1)$ getroffen
- Nach Res. von F mit $R_2 = \{\neg y, z\}$ lernen wir weiter, dass auch $N' = (x=0, y=1)$ ein "benachbartes" Nullstellencluster ist.
- Nach Res. mit $R_1 = \{x, y\}$ lernen wir das Cluster $N'' = (x=0)$.
- Zukünftige Treffer in den anderen Nullstellen des Clusters werden durch die Zusatzklausel $C = \{x\}$ vermieden

