

SAT-Problem und -Algorithmus (DPLL)

Sebastian Krings

12.05.2010

Hintergrund

SAT-Problem
Definitionen

Techniken

Resolution
Conditioning
Unit-Resolution

DPLL-Algorithmus

Allgemeines
DPLL-
DPLL

Ausblick

Backtracking
Conflict-Driven
Clauses

- ▶ Hintergrund, SAT-Problem, Definitionen
- ▶ Techniken für SAT Solver
- ▶ DPLL-Algorithmus

Das SAT-Problem

SAT-Problem und
-Algorithmus
(DPLL)

Sebastian Krings

Hintergrund

SAT-Problem
Definitionen

Techniken

Resolution
Conditioning
Unit-Resolution

DPLL-Algorithmus

Allgemeines
DPLL-
DPLL

Ausblick

Backtracking
Conflict-Driven
Clauses

Entscheidungsproblem: Gegeben eine aussagenlogische Formel Δ . Gibt es eine Belegung der Variablen in Δ mit true oder false, so dass Δ true ist?

- ▶ Klausel: Disjunktion über unterschiedliche Variablen
 - ▶ Schließt P und $\neg P$ in einer Klausel aus
- ▶ CNF: Konjunktion von Klauseln

Beispiel: $(A \vee B \vee \neg C) \wedge (\neg A \vee D) \wedge (B \vee C \vee D)$

Korrektheit und Erfüllbarkeit

- ▶ Die leere Klausel (keine Literale) ist inkonsistent
- ▶ Die leere CNF (keine Klauseln) ist erfüllbar

Notation als Menge

- ▶ Klausel $l_1 \vee l_2 \vee l_3 \vee \dots \vee l_m$ darstellbar als $\{l_1, l_2, l_3, \dots, l_m\}$
- ▶ CNF $a_1 \wedge a_2 \wedge \dots \wedge a_n$ darstellbar als $\{a_1, a_2, \dots, a_n\}$

Beispiel: $(A \vee B \vee \neg C) \wedge (\neg A \vee D) \wedge (B \vee C \vee D)$ darstellbar als $\{\{A, B, \neg C\}, \{\neg A, D\}, \{B, C, D\}\}$

- ▶ CNF Δ erfüllbar wenn leer: $\Delta = \emptyset$
- ▶ CNF Δ inkonsistent wenn leere Klausel enthalten:
 $\emptyset \in \Delta$
- ▶ Übliche Randbedingungen für rekursive Algorithmen

- ▶ Formel Δ impliziert Formel $\Gamma \Leftrightarrow$ jede erfüllende Belegung von Δ erfüllt auch Γ
 - ▶ Notation: $\Delta \models \Gamma$
- ▶ Für Klauseln C_i, C_j : $C_i \subseteq C_j \Rightarrow C_i \models C_j$
- ▶ Für CNF Δ : $\Delta \models \{\emptyset\} \Rightarrow \Delta$ ist unerfüllbar

- ▶ Gegeben Literal P und CNF Δ mit
 $C_i, C_j \in \Delta : P \in C_i \wedge \neg P \in C_j$
- ▶ Erlaubt Ableiten der Klausel $(C_i - \{P\}) \cup (C_j - \{\neg P\})$
- ▶ Die abgeleitete Klausel heißt auch Resolvent

- ▶ Resolution liefert keine falschen Klauseln
- ▶ Resolution liefert nicht alle ableitbaren Klauseln (incomplete)
- ▶ Dafür garantiert die leere Klausel bei Nichterfüllbarkeit (refutation complete)
- ▶ \Rightarrow Erfüllbarkeitstest: Anwenden von Resolution bis entweder die leere Klausel abgeleitet wurde oder kein Schritt mehr möglich ist.

Resolution - Beispiel

Für die CNF $\Delta = (\neg P \vee R) \wedge (\neg Q \vee R) \wedge (\neg R) \wedge (P \vee Q)$

Initiale Klauseln:

1. $\{\neg P, R\}$
2. $\{\neg Q, R\}$
3. $\{\neg R\}$
4. $\{P, Q\}$

Abgeleitete Klauseln:

5. $\{\neg P\}$ aus 1 und 3
6. $\{\neg Q\}$ aus 2 und 3
7. $\{Q\}$ aus 4 und 5
8. $\{\}$ aus 6 und 7

\Rightarrow Nicht erfüllbar.

- ▶ Gegeben Literal P und CNF Δ : Ersetze alle P durch true und alle $\neg P$ durch false
- ▶ Notation: $\Delta|L = \{\alpha - \{\neg L\} | \alpha \in \Delta, L \notin \alpha\}$
- ▶ Anwenden auf mehrere Literale: $\Delta|AB = (\Delta|A)|B$

Auswirkungen von Conditioning des Literals L auf Klausel α :

1. $L \in \alpha$: Da L ersetzt mit `true` sind diese Klauseln erfüllt. Daher für Erfüllbarkeit von $\Delta|L$ nicht mehr relevant.
2. $\neg L \in \alpha$: Da $\neg L$ ersetzt mit `false` hat das Literal keine Effekt mehr. Daher gekürzte Klausel in $\Delta|L$.
3. $L \notin \alpha, \neg L \notin \alpha$: Keine Veränderung.

Conditioning - Beispiel

Für die CNF $\Delta = \{\{A, B, \neg C\}, \{\neg A, D\}, \{B, C, D\}\}$:

- ▶ $\Delta|C = \{\{A, B\}, \{\neg A, D\}\}$
- ▶ $\Delta|\neg C = \{\{\neg A, D\}, \{B, D\}\}$
- ▶ $\Delta|CA\neg D = \{\emptyset\}$
- ▶ $\Delta|\neg CD = \emptyset$

- ▶ Resolution mit Klauseln, die nur ein Literal enthalten, den so genannten “Unit-Clauses”
- ▶ Nicht mehr refutation complete
- ▶ Alle Schritte können linear in der Länge der CNF durchgeführt werden
- ▶ Liefert:
 1. Eine Menge I von Literalen, die sich in Unit-Clauses befanden oder abgeleitet wurden
 2. Eine CNF Γ , die der ursprünglichen CNF unter Conditioning der Variablen in I entspricht

- ▶ Basieren auf systematischer Suche in einem Baum
- ▶ Blätter des Baumes entsprechen Belegungen der Literale
- ▶ Suchbaum hat begrenzte Tiefe
- ▶ Tiefensuche oder iterative Tiefensuche
- ▶ in jedem Knoten Conditioning

Hintergrund

SAT-Problem
Definitionen

Techniken

Resolution
Conditioning
Unit-Resolution

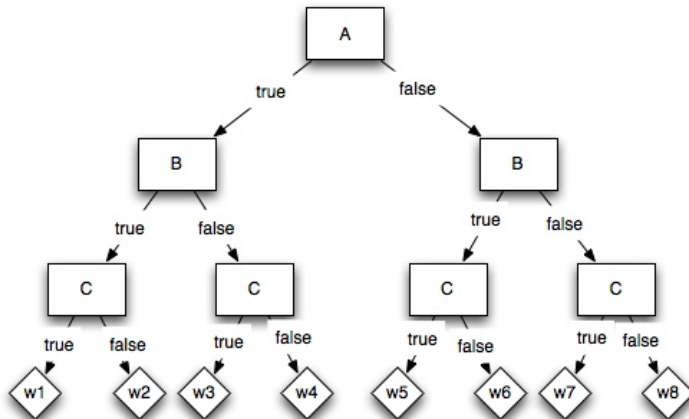
DPLL-Algorithmus

Allgemeines
DPLL-
DPLL

Ausblick

Backtracking
Conflict-Driven
Clauses

Suchbaum Beispiel



Hintergrund

SAT-Problem
Definitionen

Techniken

Resolution
Conditioning
Unit-Resolution

DPLL-Algorithmus

Allgemeines
DPLL-
DPLL

Ausblick

Backtracking
Conflict-Driven
Clauses

DPLL- - Algorithmus

```
DPLL-(CNF  $\Delta$ , depth  $d$ ) {  
  if  $\Delta = \{\}$  then  
    return  $\{\}$   
  else if  $\{\} \in \Delta$  then  
    return UNSAT  
  else if  $S = \text{DPLL}-(\Delta|P_{d+1}, d+1) \neq \text{UNSAT}$  then  
    return  $S \cup \{P_{d+1}\}$   
  else if  $S = \text{DPLL}-(\Delta|\neg P_{d+1}, d+1) \neq \text{UNSAT}$  then  
    return  $S \cup \{\neg P_{d+1}\}$   
  else  
    return UNSAT  
}
```

- ▶ Früh entstehende Konflikte werden unter Umständen erst spät erkannt
- ▶ Laufzeit hängt stark von der Reihenfolge der Literale ab
 - ▶ Algorithmen zur Wahl
 - ▶ Heuristiken

DPLL - Algorithmus

```
DPLL(CNF  $\Delta$ ) {  
  ( $I, \Gamma$ ) = UNIT-RESOLUTION( $\Delta$ )  
  if  $\Gamma = \{\}$  then  
    return  $I$   
  else if  $\{\} \in \Gamma$  then  
    return UNSAT  
  else  
    wähle Literal  $L$  in  $\Gamma$   
    if  $S = \text{DPLL}(\Gamma|L) \neq \text{UNSAT}$  then  
      return  $S \cup I \cup \{L\}$   
    if  $S = \text{DPLL}(\Gamma|\neg L) \neq \text{UNSAT}$  then  
      return  $S \cup I \cup \{\neg L\}$   
    else  
      return UNSAT  
}
```

Non-Chronological Backtracking

- ▶ DPLL nutzt chronologisches Backtracking
- ▶ Früh entstandene Unlösbarkeiten werden trotzdem durchprobiert
- ▶ Idee: Backtracking über mehrere Ebenen in einem Schritt

Non-Chronological Backtracking - Durchführung

- ▶ Bilden eines “Conflict Set” aus den Variablen die zum Widerspruch beitragen
- ▶ Backtracken zur jüngsten Zuweisung einer der Variablen
- ▶ Dabei werden alle seit dem erfolgten Zuweisungen rückgängig gemacht

Conflict-Driven Clauses

- ▶ Idee: Der Solver soll aus seinen Fehlern lernen
- ▶ Bei einem Konflikt wird eine spezielle Klausel berechnet und zur Formel hinzugefügt
- ▶ Wenn der Solver den Fehler wiederholt, schlägt diese Klausel sofort fehl

Conflict-Driven Clauses & Backtracking

- ▶ Selektives Backtracking nötig
- ▶ 2 Ansätze:
 1. far-backtracking: Conflict-Driven Clause hinzufügen, vereinfachen und von vorn suchen
 2. normales Backtracking: Conflict-Driven Clause hinzufügen und Anwenden von Non-Chronological Backtracking