

# SAT-Solving und Anwendungen

## Der DPLL Algorithmus

Prof. Dr. Wolfgang Küchlin  
Dipl. Inform. Christoph Zengler

Universität Tübingen

06. Mai 2009

# Übersicht

- **DPLL:** Davis-Putnam-Logemann-Loveland
  - M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.
  - M. Davis, G. Logemann and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.
- Grundlegender Algorithmus
- Meist auf CNF benutzt, geht aber auch allgemein (non-CNF)
- Grundidee steckt bis heute in fast allen SAT Solvern
- **Grundidee:** Fallunterscheidung und Vereinfachungen
- Vereinfachungen:
  - Unit Propagation (Unit Resolution + Unit Subsumption)

# Der DPLL-Algorithmus

**Input:** Formel  $P$  in Aussagenlogik, Interpretation  $\nu_0$

**Output:** **true** oder **false**

$DPLL(P, \nu_0)$

UnitPropagation( $P, \nu_0$ );

**if** ( $\nu(P) = \mathbf{T}$ ) **then**

└ **return true**

**if** ( $\nu(P) = \mathbf{F}$ ) **then**

└ **return false**

wähle eine noch nicht belegte Variable  $x \in \text{var}(P)$

**if**  $DPLL(P \cup \{\{\neg x\}\}, \nu_0 \cup [x \mapsto \mathbf{F}])$  **then**

└ **return T**

**else**

└ **return**  $DPLL(P \cup \{\{x\}\}, \nu_0 \cup [x \mapsto \mathbf{T}])$

# Unit Propagation

Idee bei CNF: Wenn in einer Klausel  $C$  nur noch eine Variable unbelegt ist und  $C$  unerfüllt ist, belege diese letzte Variable so, dass  $C$  erfüllt wird.

## Unit clause

Eine *unit clause* ist eine Klausel  $C = (x_1 \vee x_2 \vee \dots \vee x_n)$  so dass  $\nu(C) \neq \mathbf{T}$  und alle  $x_i$  bis auf eines interpretiert sind.

## Beispiel (Unit clause)

$(x_1 \vee \neg x_2 \vee x_3)$  mit  $\{x_1 \mapsto \mathbf{F}, x_2 \mapsto \mathbf{T}\}$ . Man sagt  $x_3$  ist *unit*.

**Vorgehen (solange es eine unit clause gibt):**

- Belege das Literal  $\ell$ , das unit ist, positiv
- **Unit Subsumption:** Lösche alle Klauseln in denen  $\ell$  vorkommt (denn diese Klauseln sind damit erfüllt)
- **Unit Resolution:** Lösche  $\ell$  aus allen Klauseln in denen es mit negiertem Vorzeichen vorkommt

# Unit Propagation - Beispiel

$$P = \{\{\neg x, y, \neg z\}, \{\neg x, z\}, \{\neg y, x\}, \{y\}\}$$

## Beispiel (Unit Propagation)

- ① Unit clause vorhanden? Ja:  $\{y\}$
  - ② Belege Variable entsprechend:  $\nu_0 = \{y \mapsto \mathbf{T}\}$
  - ③ Unit Subsumption: lösche Klauseln, in denen  $y$  vorkommt:  $P_1 = \{\{\neg x, z\}, \{\neg y, x\}\}$
  - ④ Unit Resolution: lösche  $\neg y$  aus allen Klauseln:  $P_2 = \{\{\neg x, z\}, \{x\}\}$
  - ⑤ Unit clause vorhanden? Ja:  $\{x\}$
  - ⑥ Belege Variable entsprechend:  $\nu_0 = \{y \mapsto \mathbf{T}, x \mapsto \mathbf{T}\}$
  - ⑦ Unit Subsumption:  $P_3 = \{\{\neg x, z\}\}$
  - ⑧ Unit Resolution:  $P_4 = \{\{z\}\}$
  - ⑨ Unit clause vorhanden? Ja  $\{z\}$
  - ⑩ Belege Variable entsprechend:  $\nu_0 = \{y \mapsto \mathbf{T}, x \mapsto \mathbf{T}, z \mapsto \mathbf{T}\}$
  - ⑪ Unit Subsumption:  $P_5 = \{\}$
- Ergebnis:  $P$  erfüllbar mit  $\nu_0 = \{y \mapsto \mathbf{T}, x \mapsto \mathbf{T}, z \mapsto \mathbf{T}\}$

# Der Algorithmus - Reminder

**Input:** Formel  $P$  in Aussagenlogik, Interpretation  $\nu_0$

**Output:** **true** oder **false**

$DPLL(P, \nu_0)$

UnitPropagation( $P, \nu_0$ );

**if** ( $\nu(P) = \mathbf{T}$ ) **then**

**return true**

**if** ( $\nu(P) = \mathbf{F}$ ) **then**

**return false**

wähle eine noch nicht belegte Variable  $x \in \text{var}(P)$

**if**  $DPLL(P, \nu_0 \cup [x \mapsto \mathbf{F}])$  **then**

**return T**

**else**

**return**  $DPLL(P, \nu_0 \cup [x \mapsto \mathbf{T}])$

# Berechnung von $\nu(P)$

Im Falle der CNF muss man  $\nu(P)$  nicht kompliziert mit Regeln berechnen:

- $\nu(P) = \mathbf{T}$  gdw. jede einzelne Klausel von  $P$  erfüllt ist
  - man speichert zu jeder Klausel, welche Literale sie gerade erfüllen
  - ist diese Menge nicht leer, ist die Klausel erfüllt
- $\nu(P) = \mathbf{F}$  gdw. es gibt eine leere Klausel *empty clause* in der Klauselmenge
- Leere Klausel: Alle Variablen in der Klausel sind belegt, aber die Klausel ist unerfüllt
  - Es gibt dann keine Möglichkeit mehr, die Klausel zu erfüllen (da alle Variablen belegt)
  - Damit ist die gesamte Formel  $P$  nicht erfüllbar
  - Man speichert zu jeder Klausel, wie viele unbelegte Variablen in ihr sind.
  - Ist diese Anzahl gleich 0 und die Menge der erfüllenden Literale ist leer, so handelt es sich um eine *empty clause*

# Berechnung von $\nu(P)$ - Beispiele

$$P = \{\{x, y, \neg z\}, \{\neg x, z\}, \{x, \neg y\}, \{\neg y, \neg z\}\}$$

## Beispiel (Berechnung von $\nu(P)$ )

Angenommen:  $\nu_0 = \{x \mapsto \mathbf{T}\}$ :

- $\{x, y, \neg z\}$  und  $\{x, \neg y\}$  sind erfüllt
- $\{\{x, y, \neg z\}, \{\neg x, z\}, \{x, \neg y\}, \{\neg y, \neg z\}\}$

Angenommen:  $\nu_0 = \{x \mapsto \mathbf{T}, z \mapsto \mathbf{F}\}$ :

- $\{x, y, \neg z\}, \{x, \neg y\}$  und  $\{\neg y, \neg z\}$  sind erfüllt
- **ABER:**  $\{\neg x, z\}$  ist nicht erfüllt und hat keine belegbaren Variablen mehr (da  $x$  und  $z$  bereits belegt)

→  $\{\neg x, z\}$  ist empty clause

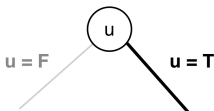
Angenommen:  $\nu_0 = \{x \mapsto \mathbf{T}, z \mapsto \mathbf{T}, y \mapsto \mathbf{F}\}$ :

- Alle Klauseln sind erfüllt →  $P$  ist erfüllt

# Visualisierung von DPLL

$$(u \vee \neg w) \wedge (\neg u \vee \neg z) \wedge (w \vee \neg y) \wedge (w \vee \neg x) \wedge (x \vee y \vee z)$$

wähle  $u = T$



$$(u \vee \neg w) \wedge (\neg u \vee \neg z) \wedge (w \vee \neg y) \wedge (w \vee \neg x) \wedge (x \vee y \vee z)$$

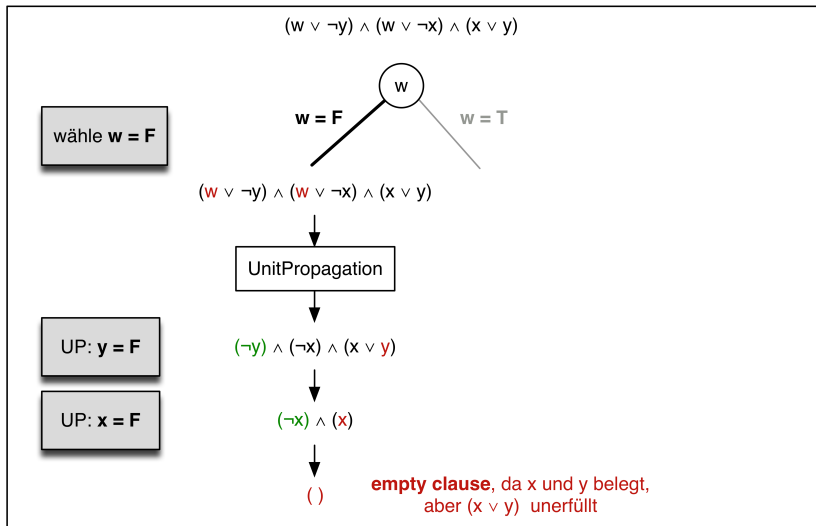
UnitPropagation

UP:  $z = F$

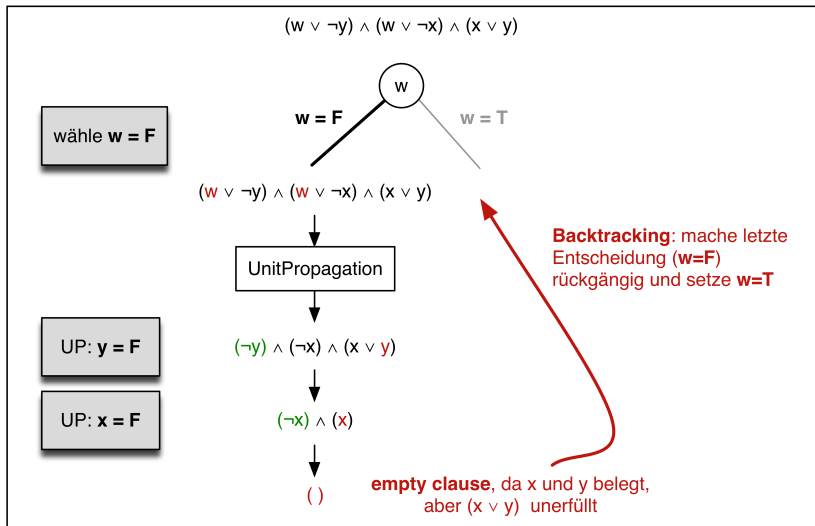
$$(\neg z) \wedge (w \vee \neg y) \wedge (w \vee \neg x) \wedge (x \vee y \vee z)$$

$$(w \vee \neg y) \wedge (w \vee \neg x) \wedge (x \vee y)$$

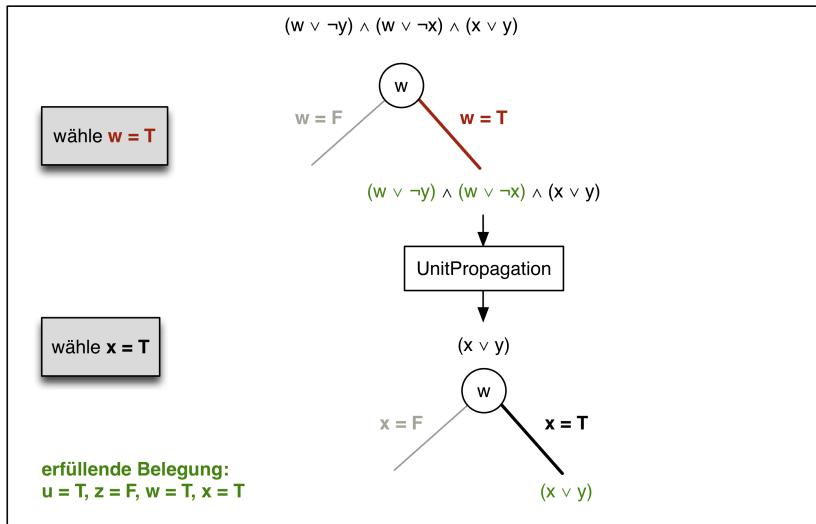
# Visualisierung von DPLL



# Visualisierung von DPLL



# Visualisierung von DPLL



# Laufzeit von DPLL

- Im worst case müssen für  $n$  Variablen alle  $2^n$  Belegungen durchprobiert werden
- **ABER:** In der Praxis ist dies so gut wie nie der Fall
- wichtige Faktoren:
  - schnelle UP (moderne SAT-Solver verbrauchen bis zu 95% ihrer Zeit mit UP)
  - gute Auswahl für die nächste Variable (Verzweigungsheuristik)

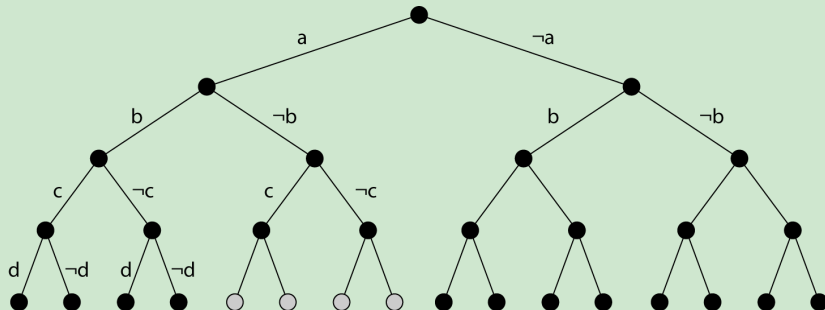
## Verzweigungsheuristik

- Es gibt keine beweisbar “gute” Heuristik um die nächste Variable auszuwählen
- Jedoch gibt es viele Ansätze, die sich in der Praxis bewährt haben
- Kenngrößen: Anzahl der Vorkommen der Variable, Anzahl der Vorkommen in kleinen Klauseln, Aktivität der Variable

# Verzweigungsheuristiken

## Beispiel (Variablenauswahl)

Wählt man im folgenden Beispiel als erste Variable  $a = \mathbf{F}$  so steigt man in den falschen Teilbaum ab und muss bis ganz nach oben backtracken um wieder in den richtigen Teilbaum zu gelangen



# Verzweigungsheuristiken - 1

- $numpos(x)$  = Anzahl der positiven Vorkommen der Variable  $x$  in unerfüllten Klauseln
- $numneg(x)$  = Anzahl der negativen Vorkommen der Variable  $x$  in unerfüllten Klauseln

## DLCS - Dynamic Largest Combined Sum

Wähle als nächstes Variable  $x$  mit der größten Summe:

$$numpos(x) + numneg(x)$$

## DLIS - Dynamic Largest Individual Sum

Wähle als nächste Variable  $x$  mit dem größten  $numpos(x)$  oder  $numneg(x)$

Idee von DLCS und DLIS: Variablen die besonders oft vorkommen haben mehr Einfluss auf das Gesamtergebnis als Variablen, die selten vorkommen.

# Verzweigungsheuristiken - 2

## MOM - **M**aximum **o**ccurrence in clauses of **m**inimal size

Wähle die Variable, deren Vorkommen in kurzen Klauseln maximal ist.

Idee von MOM:

- Kurze Klauseln haben mehr Aussagekraft als lange Klauseln, da sie schneller zu UP und somit zu neuen Variablenbelegungen führen
- Variablen die besonders oft in kurzen Klauseln vorkommen haben großen Einfluss auf das Gesamtergebnis

## Aktivitätsheuristiken

Wähle die Variable, die am aktivsten ist.

- Aktivität kann verschieden definiert sein
- Häufig: Variable, die kürzlich in vielen Konflikten (empty clauses) vorkam

Idee: Variablen, die oft in Konflikten vorkommen spielen eine herausragende Rolle und sollten zuerst belegt werden

# Ausblick

Zwei große Probleme bei DPLL:

- Backtracking immer nur zur letzten gesetzten Variable
  - **Beobachtung:** Die letzte Variable ist oft nicht verantwortlich für den aktuellen Konflikt
  - **Idee:** **Backtracking** auch über mehrere Entscheidungen hinweg zu einer Variable weiter oben im Entscheidungsbaum
- Vergessen von zusätzlichen Informationen beim Backtracking (Illustration auf der nächsten Folie)
  - **Beobachtung:** Springt man über eine gewisse Grenze beim Backtracking zurück, so “vergisst” man bereits erarbeitete Information (z.B. bestimmte UPs)
  - **Idee:** Hinzufügen dieser zusätzlichen Information zur Originalformel, so dass sie beim Backtracking nicht verloren geht

**Resultat: SAT-Solver mit nichtchronologischem Backtracking**  
(Problem 1) und **Klausellernen** (Problem 2)

# Vergessen von Informationen beim Backtracking

An einer gewissen Stelle existiert die Information, dass  $x = \mathbf{F}$  gelten muss, diese geht bei zu weitem Backtracking jedoch wieder verloren

