SAT-Solving und Anwendungen
Der DPLL Algorithmus

Prof. Dr. Wolfgang Küchlin
Rouven Walter, M.Sc.

Universität Tübingen

23. April 2013

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 ν_0
Output: 1 oder 0
$DPLL(P, \nu_0)$

UnitPropagation(P, ν_0);
if $(\nu(P) = 1)$ **then**
  return 1
if $(\nu(P) = 0)$ **then**
  return 0
wähle eine noch nicht belegte Variable $x \in var(P)$

if $DPLL(P \cup \{\{\neg x\}\}, \nu_0 \cup [x \mapsto 0])$ **then**
  return 1
else
  return $DPLL(P \cup \{\{x\}\}, \nu_0 \cup [x \mapsto 1])$

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 $C = (x_1 \vee x_2 \vee \cdots \vee x_n)$ so dass $\nu(C) \neq 1$ und alle x_i bis auf eines interpretiert sind.

Beispiel (Unit Clause)

$(x_1 \vee \neg x_2 \vee x_3)$ mit $\{x_1 \mapsto 0, x_2 \mapsto 1\}$. Man sagt x_3 *ist unit*.

Vorgehen (solange es eine unit clause gibt):

- Belege das Literal ℓ, das unit ist, positiv
- **Unit Subsumption:** Lösche alle Klauseln in denen ℓ vorkommt (denn diese Klauseln sind damit erfüllt)
- **Unit Resolution:** Lösche ℓ aus allen Klauseln in denen es mit negativem 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: $\nu_0 = \{y \mapsto 1\}$
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: $\nu_0 = \{y \mapsto 1, x \mapsto 1\}$
7. Unit Subsumption: $P_3 = \{\{\neg x, z\}\}$
8. Unit Resolution: $P_4 = \{\{z\}\}$
9. Unit clause vorhanden? Ja $\{z\}$
10. Belege Variable entsprechend: $\nu_0 = \{y \mapsto 1, x \mapsto 1, z \mapsto 1\}$
11. Unit Subsumption: $P_5 = \{\}$

Ergebnis: P erfüllbar mit $\nu_0 = \{y \mapsto 1, x \mapsto 1, z \mapsto 1\}$

Der Algorithmus - Reminder

Input: Formel P in Aussagenlogik, Interpretation ν_0
Output: 1 oder 0
$DPLL(P, \nu_0)$

UnitPropagation(P, ν_0);
if $(\nu(P) = 1)$ **then**
  return 1
if $(\nu(P) = 0)$ **then**
  return 0
wähle eine noch nicht belegte Variable $x \in var(P)$

if $DPLL(P, \nu_0 \cup [x \mapsto 0])$ **then**
  return 1
else
  return $DPLL(P, \nu_0 \cup [x \mapsto 1])$

Berechnung von $\nu(P)$

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

- $\nu(P) = 1$ 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
- $\nu(P) = 0$ 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: $\nu_0 = \{x \mapsto 1\}$:

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

Angenommen: $\nu_0 = \{x \mapsto 1, z \mapsto 0\}$:

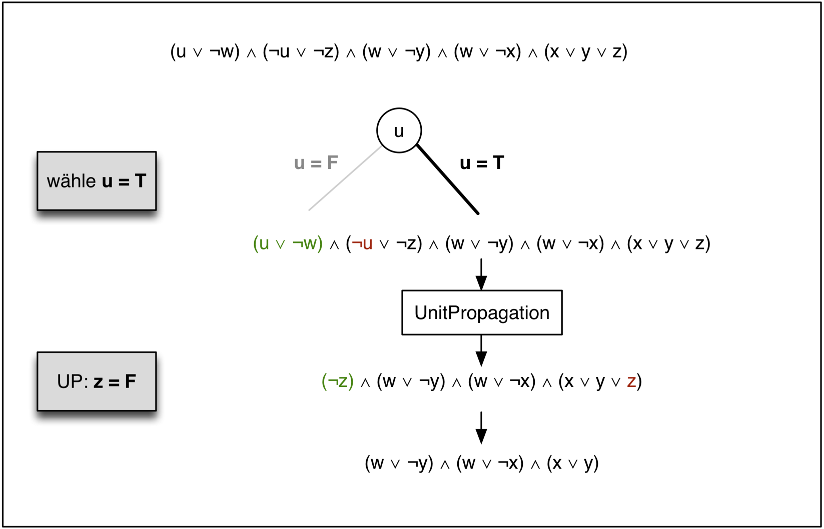
- $\{x, y, \neg z\}, \{x, \neg y\}$ und $\{\neg y, \neg z\}$ sind erfüllt
- ABER: $\{\neg x, z\}$ ist nicht erfüllt und hat keine belegbaren Variablen mehr (da x und z bereits belegt)

$\rightarrow$ $\{\neg x, z\}$ ist empty clause

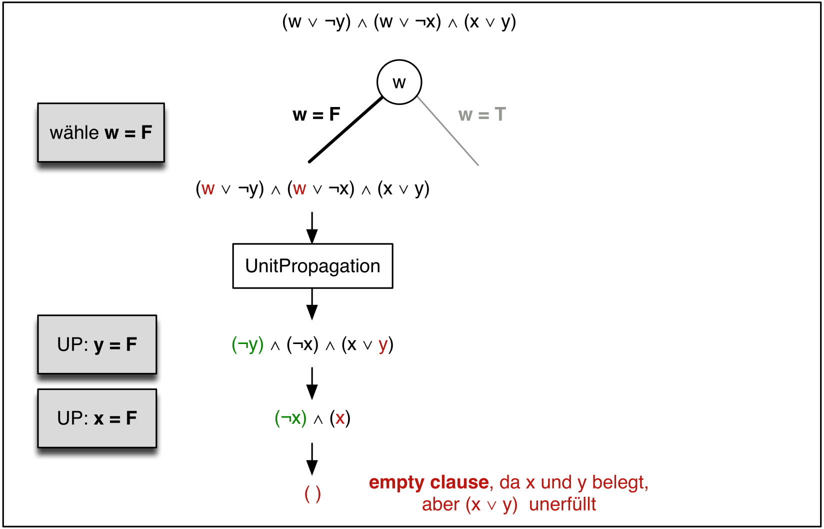
Angenommen: $\nu_0 = \{x \mapsto 1, z \mapsto 1\ y \mapsto 0\}$:

- Alle Klauseln sind erfüllt $\rightarrow$ P ist erfüllt

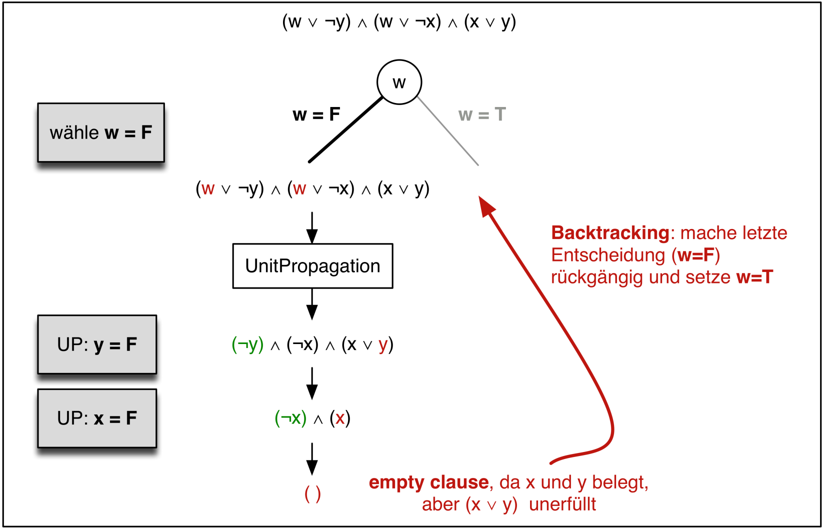
Visualisierung von DPLL



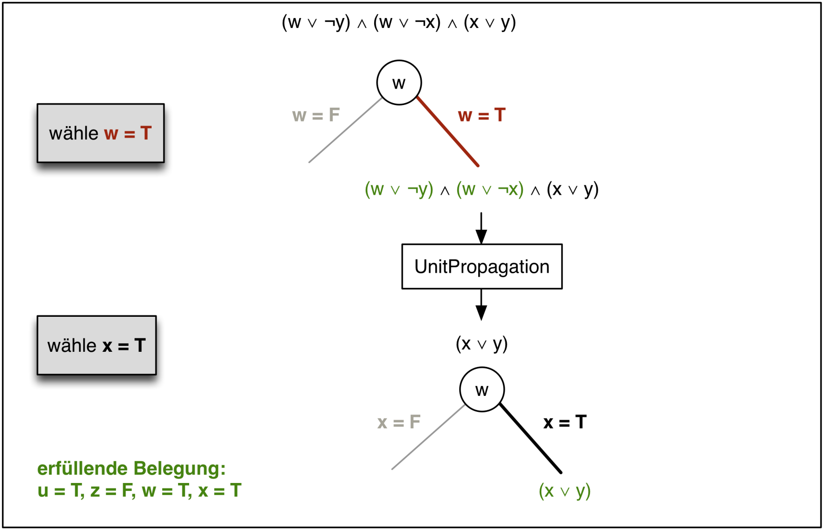
Visualisierung von DPLL



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)

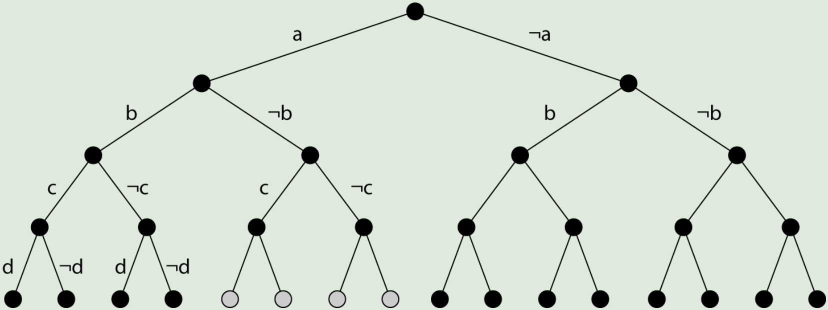
Verzweigungsheuristik

- 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 $a = 0$ 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.
- Durch die Belegung von häufig vorkommenden Variablen vereinfacht sich die Formel am stärksten.

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: Aus Konflikten lernen wir neue Klauseln (s.u.). Variablen, die oft in Konflikten vorkommen spielen eine herausragende Rolle, da sie das Lernen fördern.

Ausblick

Zwei große Probleme bei DPLL:

- Backtracking immer nur zur letzten durch Entscheidung gesetzten Variable
  - Beobachtung: Die letzte Entscheidungsvariable ist oft garnicht verantwortlich für den aktuellen Konflikt, sondern dieser wurde schon durch frühere Entscheidungen verursacht.
  - 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 $x = 0$ gelten muss, diese geht bei zu weitem Backtracking jedoch wieder verloren