SAT-Solving und Anwendungen

Der DPLL Algorithmus

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

Universität Tübingen

26. Oktober 2017

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, entscheidet die Erfüllbarkeit einer Formel F.
- Meist auf CNF benutzt, geht aber auch allgemein (non-CNF)
- Grundidee steckt bis heute in fast allen SAT Solvern
- Grundidee: Sukzessiver Aufbau einer Belegung ν mit dem Ziel $\nu(F) = 1$. Sofortige Vereinfachung von F nach jeder einzelnen Variablenbelegung. Bei Misserfolg Backtracking zu alternativen Variablenbelegungen.
  - Vereinfachungen: Unit Propagation (Unit Resolution + Unit Subsumption)
- Automatisches Beweisen von $F \models G$ über (Nicht-)Erfüllbarkeit.
  - Zum Beweis von $\models F \rightarrow G$ bzw. $\models \neg F \vee G$, suche Gegenbeispiel durch Erfüllbarkeitsprüfung $\texttt{SAT}(\neg(\neg F \vee G)) = \texttt{SAT}(F \wedge \neg G)$
  - Der Beweis ist erbracht, falls $\texttt{SAT}(F \wedge \neg G) = \mathit{false}$

Der DPLL-Algorithmus (Textfassung)

Diese Fassung eignet sich gut für Papier+Bleistift. Belegungen $x \mapsto 1$ bzw. $x \mapsto 0$ werden durch Hinzufügen von *Unit Klauseln* $\{x\}$ bzw. $\{\neg x\}$ zu P erzwungen. P wird durch Streichungen vereinfacht: gestrichen werden erfüllte Klauseln in P sowie nicht erfüllte Literale in jeder Klausel.

Input: Formula P in CNF
Output: 1 or 0
$DPLL(P)$

$P \leftarrow$UnitPropagation(P);
if $\underline{(P = \emptyset)}$ **then** // No constraint left to satisfy
  return $\underline{1}$
if $\underline{(\emptyset \in P)}$ **then** // Unsatisfiable constraint
  return $\underline{0}$

Choose a variable $x \in var(P)$

if $\underline{DPLL(P \cup \{\{\neg x\}\})}$ **then** // Try x=0
  return $\underline{1}$
else // Try x=1
  return $\underline{DPLL(P \cup \{\{x\}\})}$

Unit Propagation (Textfassung)

Unit Clause, Unit Literal

Eine Klausel ist *unit* wenn sie nur genau ein Literal ℓ enthält. ℓ ist dann ein *Unit Literal*.

Idee: Die Existenz einer Unit Klausel $\{x\}$ in P erzwingt die Belegung $x \mapsto 1$ (bzw. $\{\neg x\}$ erzwingt $x \mapsto 0$) in allen erfüllenden Belegungen von P.

Algorithmus: Unit Propagation

Solange es in P eine Unit Clause C mit einem Unit Literal ℓ gibt:

- **Unit Subsumption:** Lösche alle Klauseln in denen ℓ vorkommt, inklusive der unit clause.
  - Diese Klauseln sind damit erfüllt und müssen im Weiteren nicht mehr betrachtet werden.
  - Nach Löschen der letzten Klausel ist P erfüllt.
- **Unit Resolution with Subsumption:** Lösche $\neg\ell$ aus allen Klauseln
  - Mit Hilfe von $\neg\ell$ kann eine Klausel nicht mehr erfüllt werden.
  - Nach Löschen des letzten Literals in C ist C eine *leere Klausel* und kann nicht mehr erfüllt werden. Dann kannt auch P nicht mehr erfüllt werden.

Reminder: Subsumption and Resolution

Subsumption

Klausel C subsumiert Klausel D, falls $C \subseteq D$.
Falls $C \subseteq D$, dann $C \models D$. Falls $C \models D$, dann $\{C, D\} \equiv \{C\}$.

Resolution

Aus den Elternklauseln $C \cup \{\neg x\}$ und $D \cup \{x\}$ folgt logisch die Resolvente $C \cup D$.

Unit Resolution

Aus den Elternklauseln $C \cup \{\neg x\}$ und $\{x\}$ folgt logisch die Resolvente C.

Unit Resolution and Subsumption

Aus den Elternklauseln $C = C' \cup \{\neg x\}$ und $\{x\}$ folgt logisch die Resolvente C', und diese subsumiert C.
Also gilt $\{C' \cup \{\neg x\}, \{x\}\} \equiv \{C', C, \{x\}\} \equiv \{C', \{x\}\}$.

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&Subsumption: 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&Subsumption: $P_4 = \{\{z\}\}$
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 DPLL-Algorithmus (Fassung für den Computer)

Die Textfassung ist ineffizient, da auf jeder Rekursionsebene eine weitgehend identische Kopie der Eingabeformel existiert. Für den Computer benutzt man im Prinzip nur eine einzige Kopie der Formel P, aber jede Rekursionsebene hat eine eigene Belegung ν. Ob eine Klausel unit ist muss durch Auswertung ermittelt werden, ebenso der Wert von P (möglich: $\nu(P) = 1, \nu(P) = 0, \nu(P) = \mathit{unknown}$).

Input: Formula P in CNF, valuation $\nu = \emptyset$
Output: Satisfying assignment ν or 0
$DPLL(P, \nu)$

$\nu \leftarrow$UnitPropagation(P, ν);
if $\underline{(\nu(P) = 1)}$ **then**
  return $\underline{\nu}$
if $\underline{(\nu(P) = 0)}$ **then**
  return $\underline{0}$
Choose a variable $x \in \mathit{var}(P)$ // $\nu(P) = \mathit{unknown}$
if $\underline{DPLL(P, \nu \cup \{\{x \mapsto 0\}\}) \neq 0)}$ **then**
  return $\underline{\nu}$
else
  return $\underline{DPLL(P, \nu \cup \{\{x \mapsto 1\}\})}$

Unit Propagation (Fassung für den Computer)

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

Unit Clause

Eine Klausel $C = (\ell_1 \vee \ell_2 \vee \cdots \vee \ell_n)$ ist unit unter einer Belegung ν, falls unter ν noch genau ein Literal unbelegt ist und alle anderen Literale mit 0 belegt sind.

Beispiel (Unit Clause)

$C = (x_1 \vee \neg x_2 \vee x_3)$ mit $\nu = \{x_1 \mapsto 0, x_2 \mapsto 1\}$. C ist eine *Unit Clause* und x_3 das *Unit Literal*.

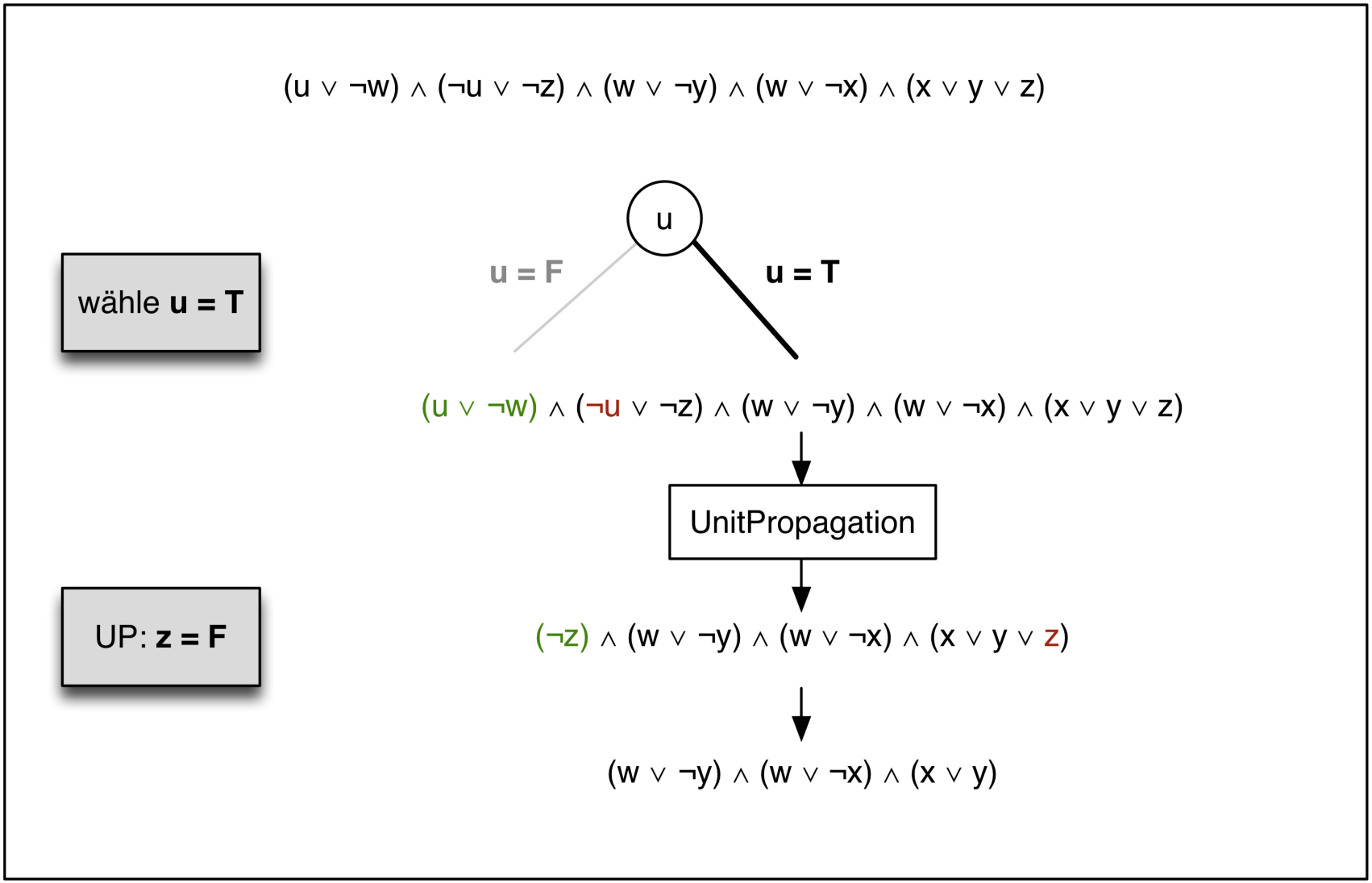
Algorithmus: Unit Propagation

Solange es unter ν eine Unit Clause C mit einem noch unbelegten Unit Literal ℓ gibt:

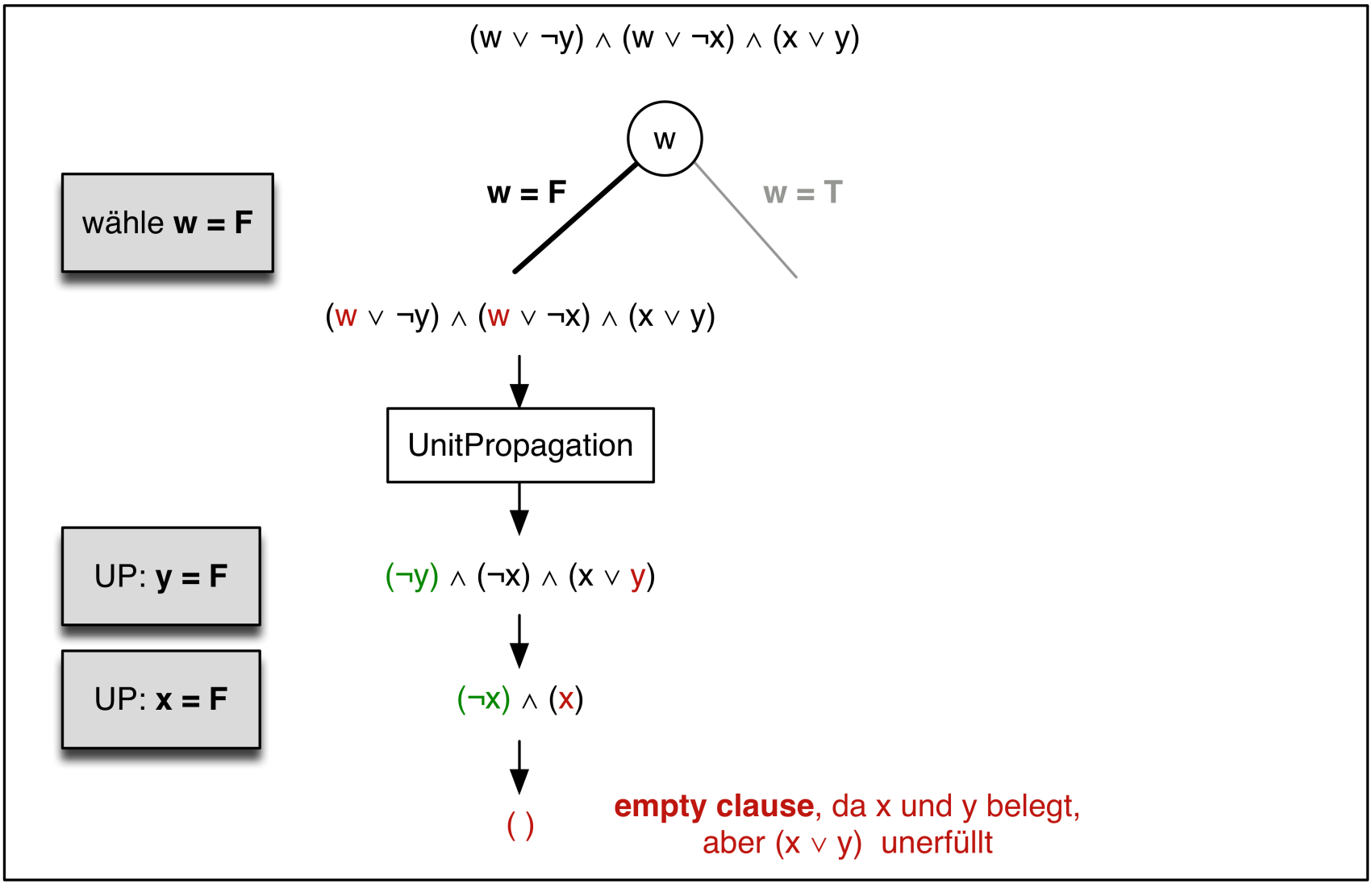
- Sei x_i die Variable in ℓ. Erweitere ν um eine Belegung von x_i so, dass $\nu(\ell) = 1$.

Unit Subsumption: geschieht implizit bei der Berechnung von $\nu(P)$.
Unit Resolution with Subsumption: geschieht implizit bei der Berechnung von $\nu(P)$ und von Unit Clauses.

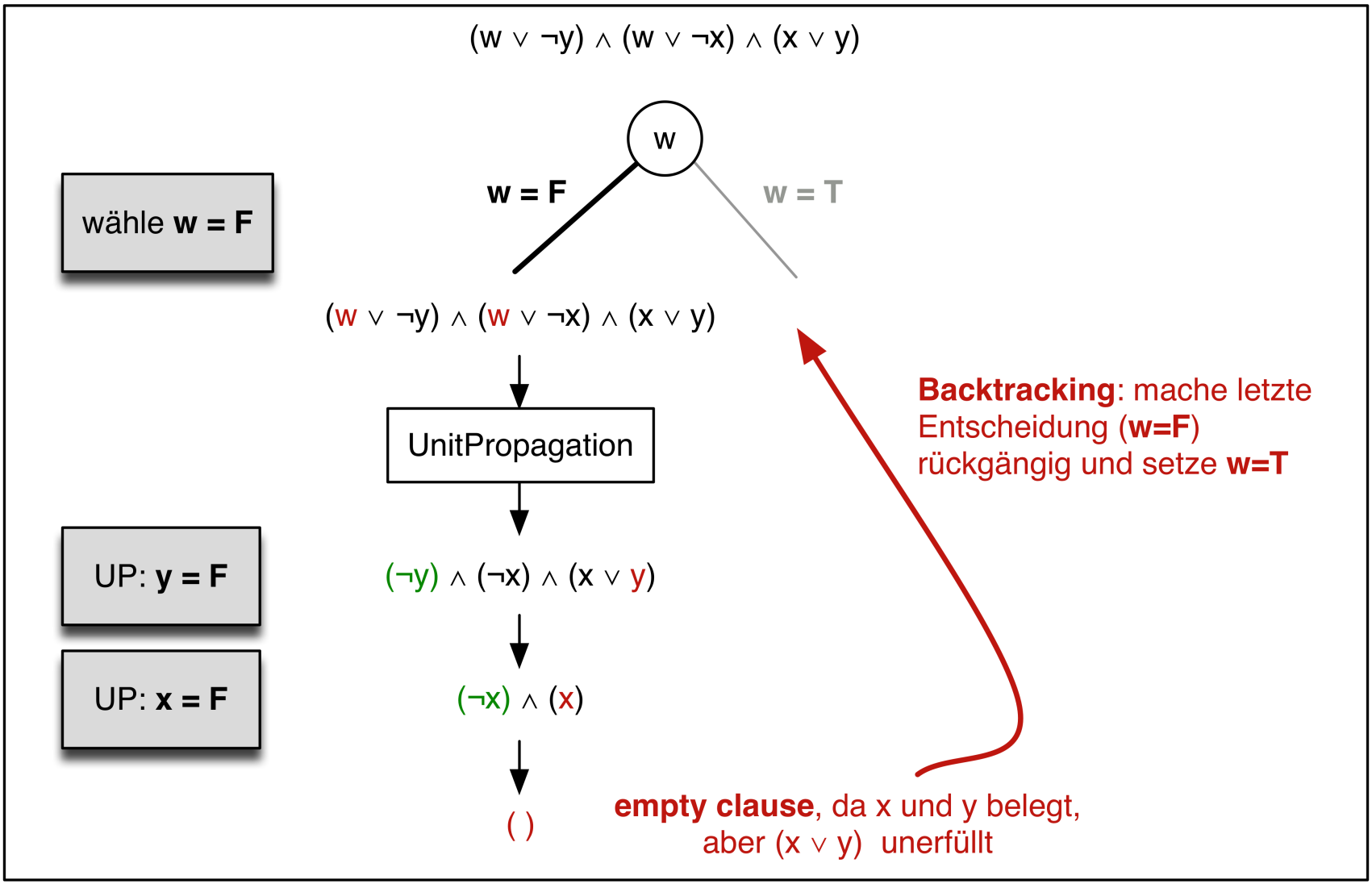
Visualisierung von DPLL



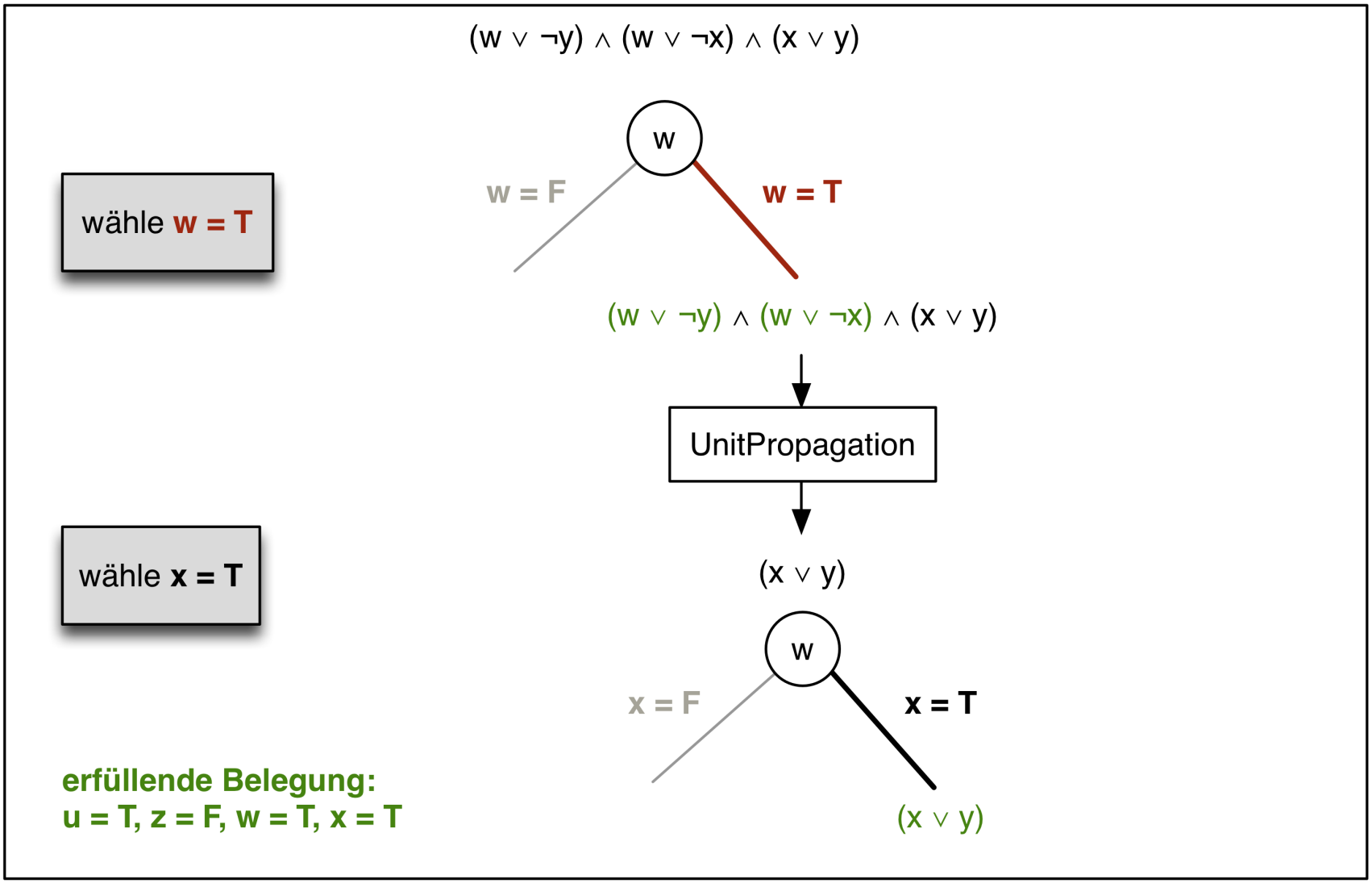
Visualisierung von DPLL



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)

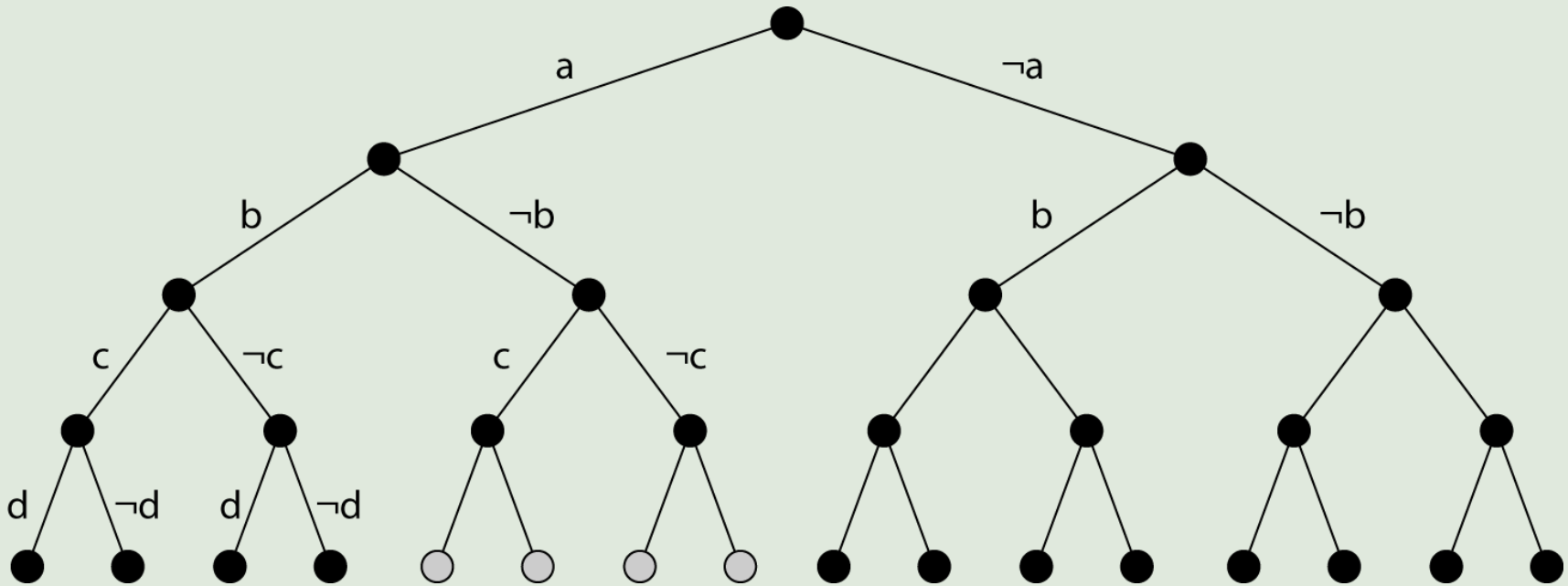
Verzweigungsheuristik

- 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

Drei große Probleme bei DPLL:

- Rekursiv, hohe Wahrscheinlichkeit für „Out Of Memory“
- Vergessen von zusätzlichen Informationen beim Backtracking (Illustration auf der nächsten Folie)
  - Beobachtung 1a: Springt man über eine gewisse Grenze beim Backtracking zurück, so „vergisst“ man bereits erarbeitete Information (z.B. bestimmte UPs)
  - Beobachtung 1b: Information zur zukünftigen Vermeidung bereits getroffener Nullstellen lässt sich als Klauseln speichern.
  - Idee 1: Hinzufügen dieser zusätzlichen Information zur Originalformel, so dass sie beim Backtracking nicht verloren geht
- Backtracking immer nur zur letzten durch Entscheidung gesetzten Variable
  - Beobachtung 2: Die letzte Variable ist oft nicht verantwortlich für den aktuellen Konflikt, sondern dieser wurde schon durch frühere Entscheidungen verursacht.
  - Idee 2: Backtracking auch über mehrere Entscheidungen hinweg zu einer Variable weiter oben im Entscheidungsbaum

Resultat: Iterativer SAT-Solver mit **Klauselllernen** (Idee 1) und **nicht-chronologischem Backtracking** (Idee 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