Automatisches Beweisen—Vertiefung
CDCL SAT Solving

Christoph Zengler

Arbeitsbereich Symbolisches Rechnen
Prof. Dr. Wolfgang Küchlin
Universität Tübingen

27. November 2012

First Order Logic

Pro

- Sehr mächtiges logisches System
- Elegante Problemformulierung (z. B. im Vergleich zu Aussagenlogik)

Contra

- semi-entscheidbar
- FOL Theorem Prover sind sehr komplexe Systeme
- Betrachtung aller möglichen Modelle

Idee!

- fixiere die Modelle auf “interessante” Interpretationen, die von Interesse sind
- z. B. Gleichheit, Lineare Arithmetik, Arrays, Pointer, Bitvektoren, . . .

SMT Solver

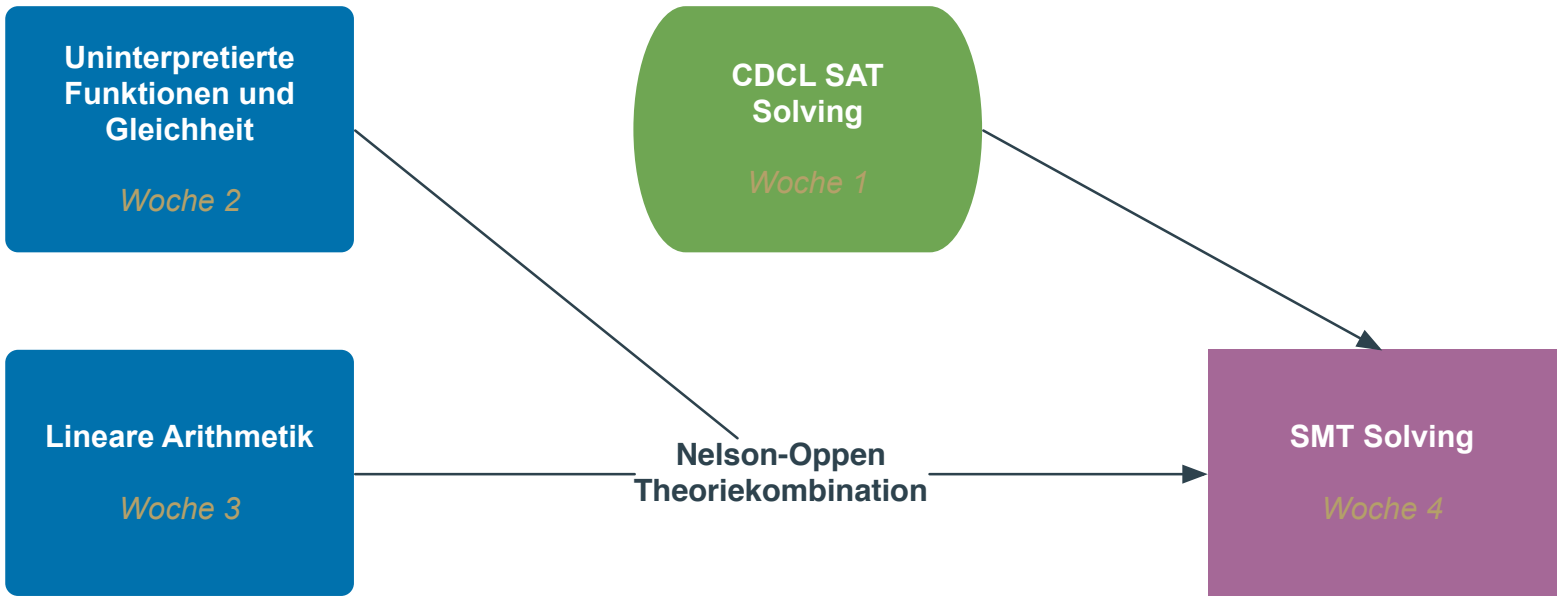
- Im Gegensatz zu FOL Beweisern werden nur bestimmte Modelle unterstützt
- Modelle, die von allgemeinem Interesse sind

BV	Bitvektoren (Software/Hardware Verifikation)
EUF	Uninterpretierte Funktionen und Gleichheit (als Abstraktionsverfahren)
LA	Lineare Arithmetik (Regelungstechnik, Naturwissenschaften)
ARR	Arrays (Speichermodellierung, Softwareverifikation)
PT	Pointer Logik (Softwareverifikation)
QE	Quantorenelimination (Alle Anwendungsbereiche)

Grundsätzliche Idee

- Benutze einen SAT Solver für das aussagenlogische Skelett
- Gib die einzelnen Teilprobleme an Solver für die jeweilige Theorie weiter
- Erlaube Wissensaustausch zwischen den Theoriesolvern

Der Fahrplan für die nächsten 5 Vorlesungen



Die Probleme bei DPLL

Problem 1: Vergesslichkeit!

Springt man über eine gewisse Grenze beim Backtracking zurück, so “vergisst” man bereits erarbeitete Information (z.B. bestimmte UP)

Idee 1

Hinzufügen dieser zusätzlichen Information zur Originalformel, so dass sie beim Backtracking nicht verloren geht

Problem 2: Wer ist schuld?

Die letzte Variable ist oft nicht verantwortlich für den aktuellen Konflikt

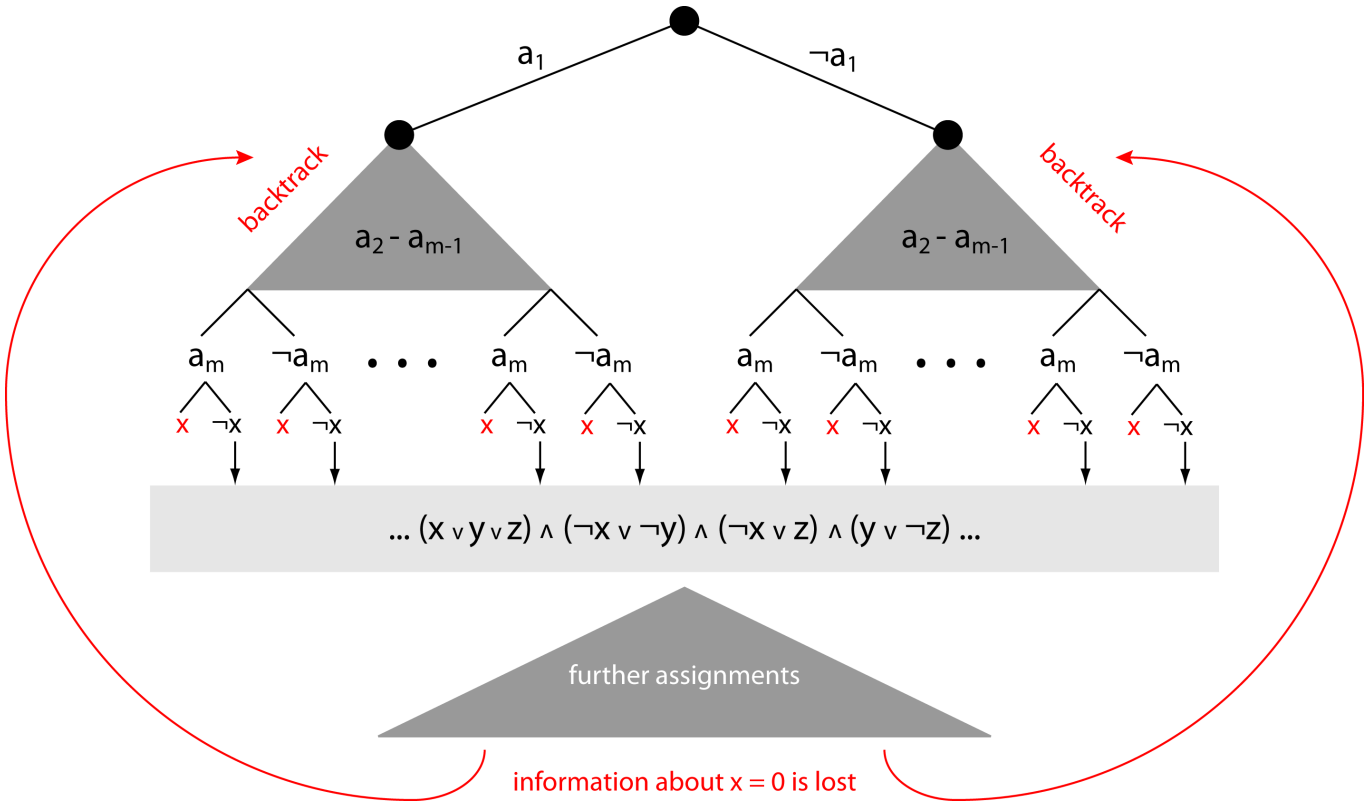
Idee 2

Backtracking auch über mehrere Entscheidungen hinweg zu einer Variable weiter oben im Entscheidungsbaum

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

Vergessen von Informationen beim Backtracking

An einer gewissen Stelle existiert die Information, dass $\chi \mapsto \texttt{false}$ gelten muss, diese geht bei zu weitem Backtracking jedoch wieder verloren



Wie realisieren wir diese Ideen?

Idee 1: Lernen von Informationen

Im Konfliktfall (leere Klausel) wird der Konflikt analysiert:

1. Speichere zu jeder Variable, ob sie durch Entscheidung oder UP belegt wurde
2. Bei UP speichere die Klausel, die die UP veranlasst hat (Grund/Reason)
3. Betreibe Resolution zwischen den einzelnen Gründen um eine neue Klausel zu erhalten
4. Füge diese neue Klausel der originalen Klauselmenge hinzu

Idee 2: Nicht-chronologisches Backtracking

1. Speichere zu jeder Variablenbelegung ein Decision Level
2. Berechne das neue Backtracking Level aus der neu gelernten Klausel

Aufbau der Datenstruktur

Klauselmenge

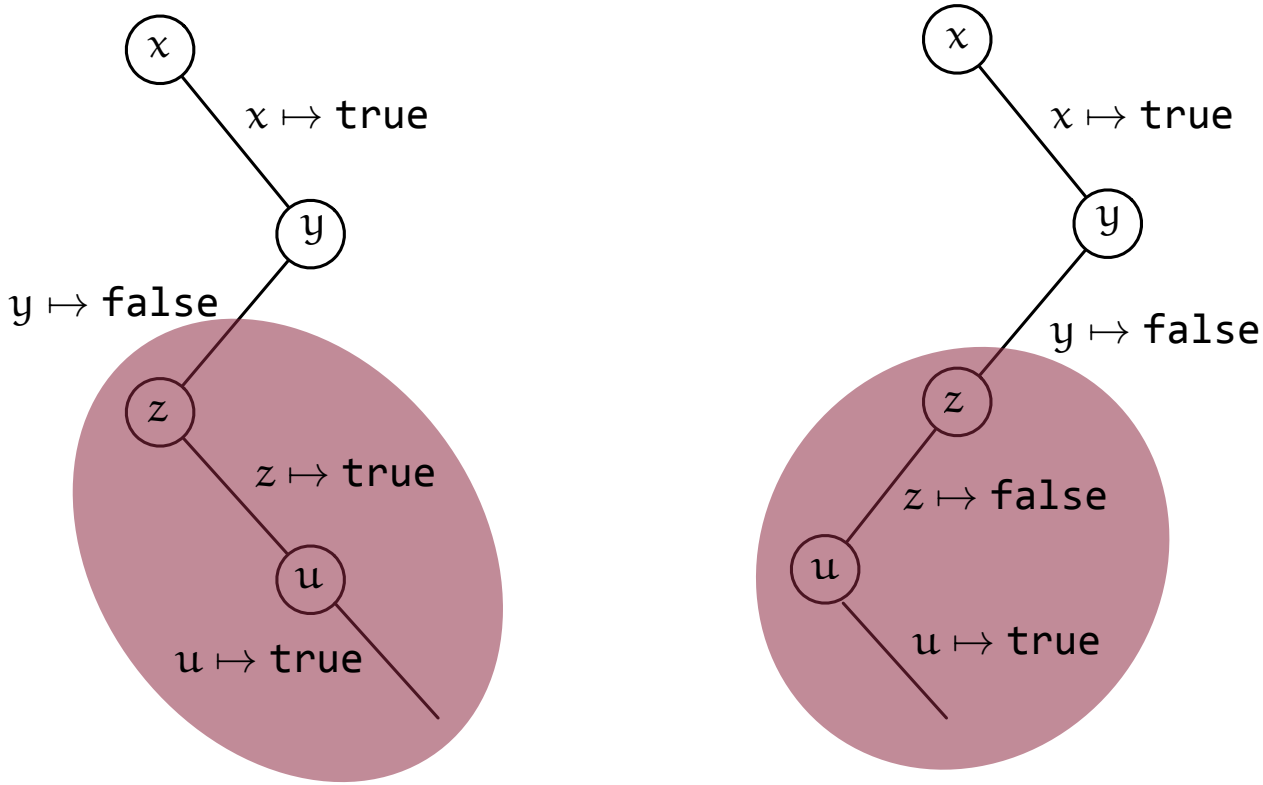
$\{\neg u, w\}$
$\{\neg u, \neg m, h, \neg w\}$
$\{\neg u, \neg f, m, \neg w\}$
$\{\neg f, \neg g, \neg h\}$
$\{y, f\}$
$\{\neg f, g\}$
$\vdots$
$\{\neg x, a\}$
$\{\neg x, b, \neg a\}$
$\{\neg x, \neg a, \neg b, c\}$
$\{\neg a, \neg d\}$
$\{\neg b, d, \neg e\}$
$\vdots$

Datenstruktur

Lev	Var	Wert	Reason
1	x	`true`	decision
	a	`true`	$\{\neg x, a\}$
	b	`true`	$\{\neg x, b, \neg a\}$
	c	`true`	$\{\neg x, \neg a, \neg b, c\}$
	d	`false`	$\{\neg a, \neg d\}$
	e	`false`	$\{\neg b, d, \neg e\}$
2	y	`false`	decision
	f	`true`	$\{y, f\}$
	g	`true`	$\{\neg f, g\}$
	h	`false`	$\{\neg f, \neg g, \neg h\}$
3	z	`true`	decision
4	u	`true`	decision
	w	`true`	$\{\neg u, w\}$
	m	`false`	$\{\neg u, \neg m, h, \neg w\}$

Konflikt: $\{\neg u, \neg f, m, \neg w\}$

Also warum Lernen?



Konflikt $\{\neg u, \neg f, m, \neg w\}$

Konflikt tritt unabhängig von der Belegung von z auf.

Lösung

Hinzufügen einer zusätzlichen Klausel um diese Information zu “lernen”.

Und welche Klausel wird hinzugefügt?

Es können verschiedene Klauseln hinzugefügt werden. Ziel ist:

- eine möglichst kurze Klausel ohne überflüssige Literale
- Klausel soll nach Backtracking unit sein (damit wird das neue Wissen sofort eingesetzt)

Zwei verschiedene Visualisierungen von Lernen:

- Resolution (schon bekannt)
- Implikationsgraph

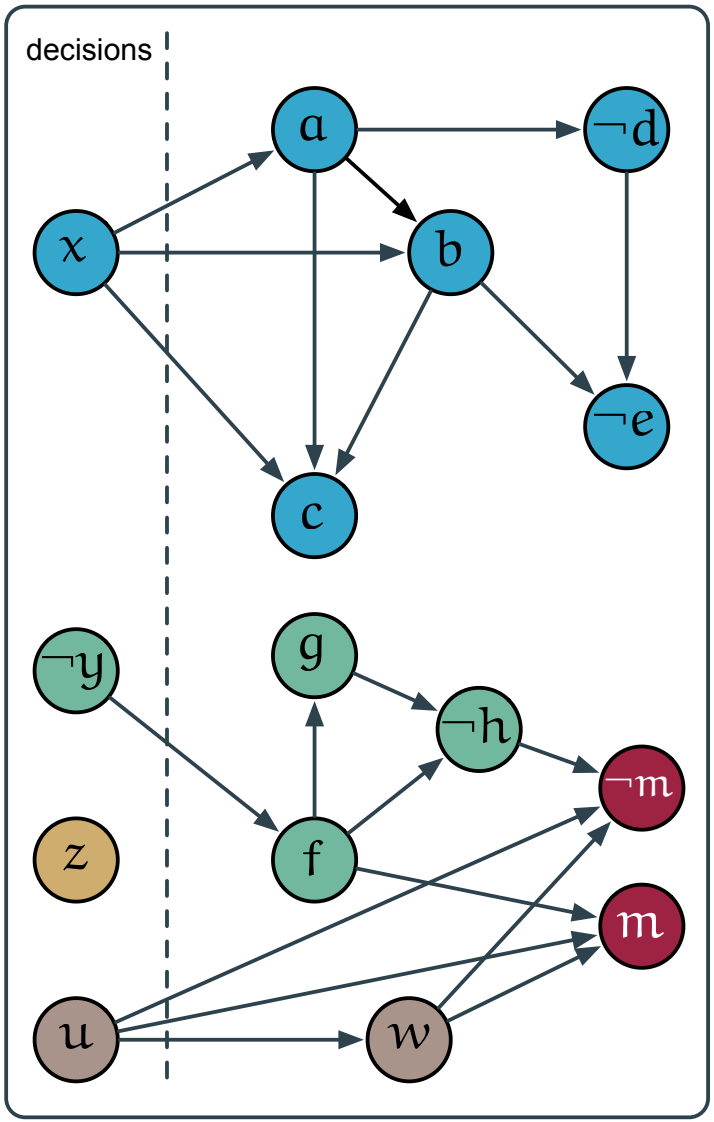
Definition (Implikationsgraph)

- Knoten sind Variablenbelegungen: x bedeutet $x \mapsto \texttt{true}$, $\neg x$ bedeutet $x \mapsto \texttt{false}$
- Kante von x nach y: x kommt in der Reason von y vor

- Decision Variables haben keine eingehenden Kanten
- War der Grund für die Implikation einer Variable y eine Klausel $\{y, x_1, ..., x_n\}$, so hat der Knoten (y) n eingehende Kanten

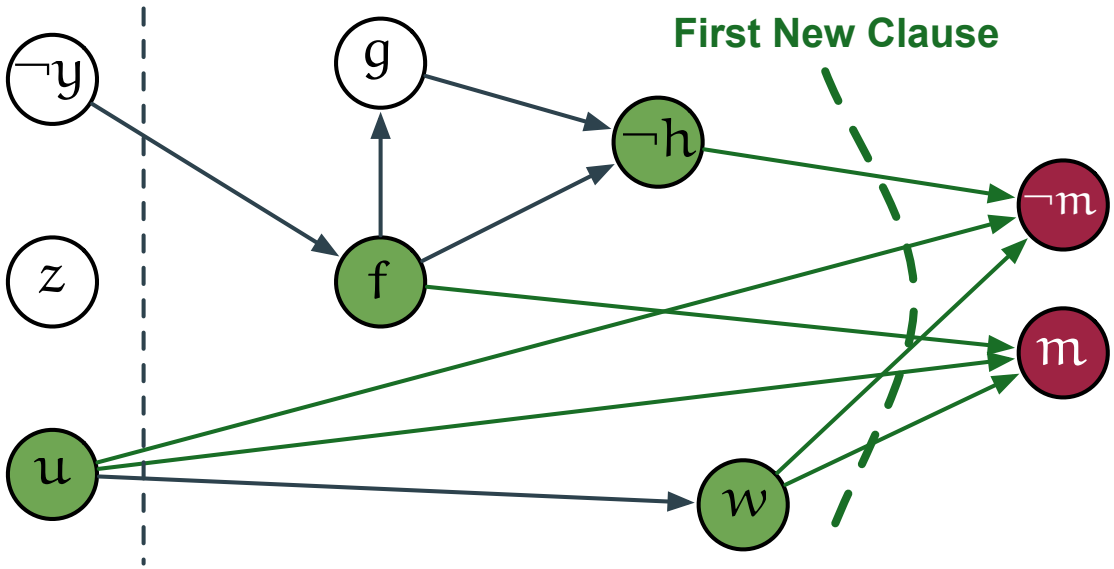
Der Implikationsgraph

Lev	Var	Wert	Reason
1	x	`true`	decision
	a	`true`	$\{\neg x, a\}$
	b	`true`	$\{\neg x, b, \neg a\}$
	c	`true`	$\{\neg x, \neg a, \neg b, c\}$
	d	`false`	$\{\neg a, \neg d\}$
	e	`false`	$\{\neg b, d, \neg e\}$
2	y	`false`	decision
	f	`true`	$\{y, f\}$
	g	`true`	$\{\neg f, g\}$
	h	`false`	$\{\neg f, \neg g, \neg h\}$
3	z	`true`	decision
4	u	`true`	decision
	w	`true`	$\{\neg u, w\}$
	m	`false`	$\{\neg u, \neg m, h, \neg w\}$
	m	`true`	$\{\neg u, \neg f, m, \neg w\}$



Und welche Klausel lernen wir jetzt?

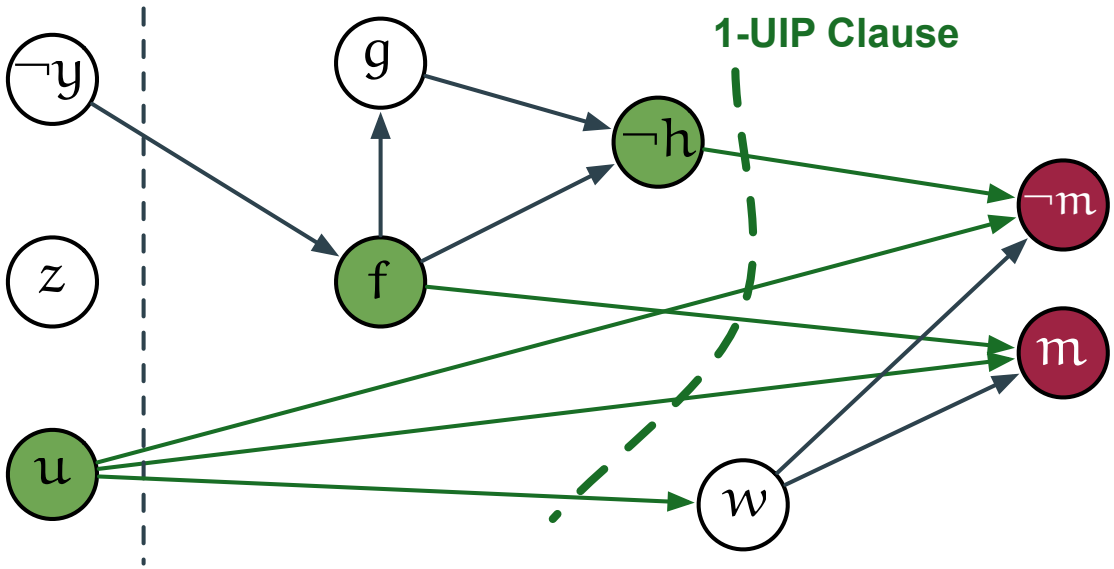
- Jeder Schnitt im Implikationsgraphen, der Entscheidungsvariablen von Konfliktvariablen trennt, ist ein gültiger Schnitt/Cut.
- Bilde eine neue Klausel aus allen Variablen, die eine ausgehende Kante durch den Cut besitzen.



Neue Klausel: $\neg(u \wedge f \wedge \neg h \wedge w) = (\neg u \vee \neg f \vee h \vee \neg w)$

Und welche Klausel lernen wir jetzt?

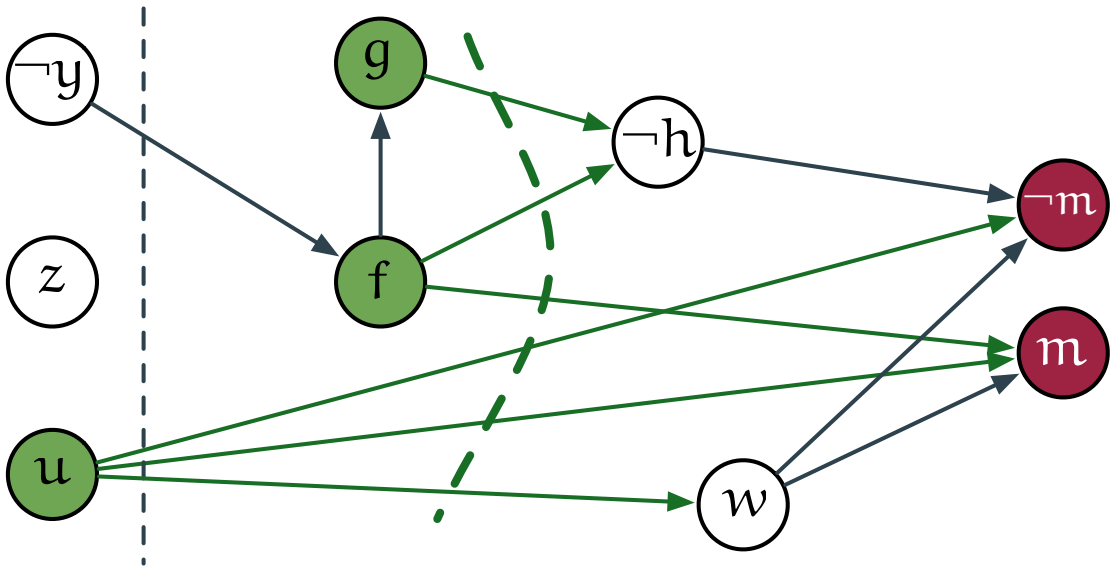
- Jeder Schnitt im Implikationsgraphen, der Entscheidungsvariablen von Konfliktvariablen trennt, ist ein gültiger Schnitt/Cut.
- Bilde eine neue Klausel aus allen Variablen, die eine ausgehende Kante durch den Cut besitzen.



Neue Klausel: $\neg(u \wedge f \wedge \neg h) = (\neg u \vee \neg f \vee h)$

Und welche Klausel lernen wir jetzt?

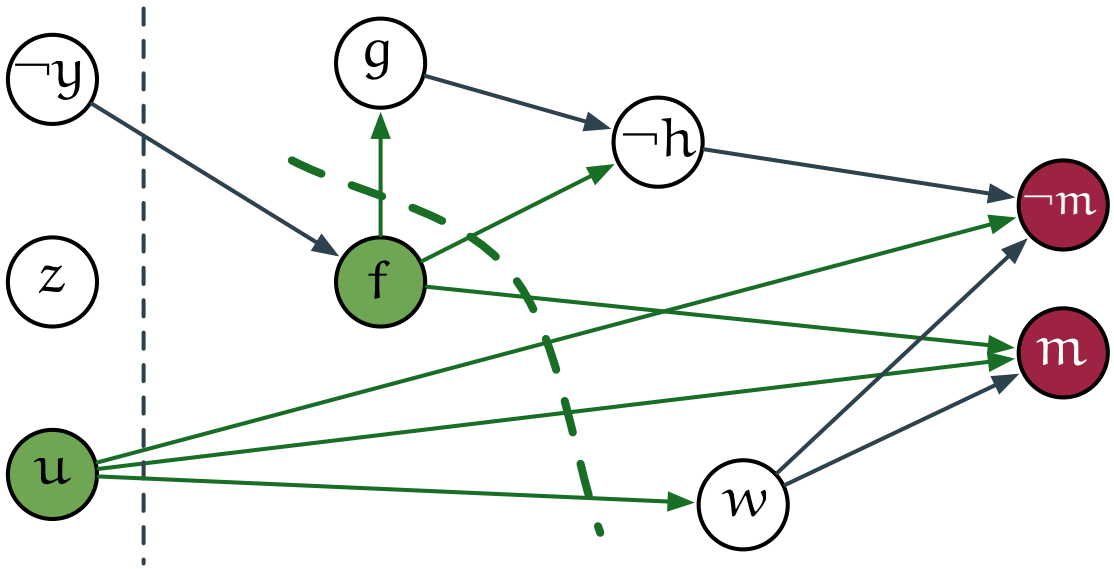
- Jeder Schnitt im Implikationsgraphen, der Entscheidungsvariablen von Konfliktvariablen trennt, ist ein gültiger Schnitt/Cut.
- Bilde eine neue Klausel aus allen Variablen, die eine ausgehende Kante durch den Cut besitzen.



Neue Klausel: $\neg(u \wedge f \wedge g) = (\neg u \vee \neg f \vee \neg g)$

Und welche Klausel lernen wir jetzt?

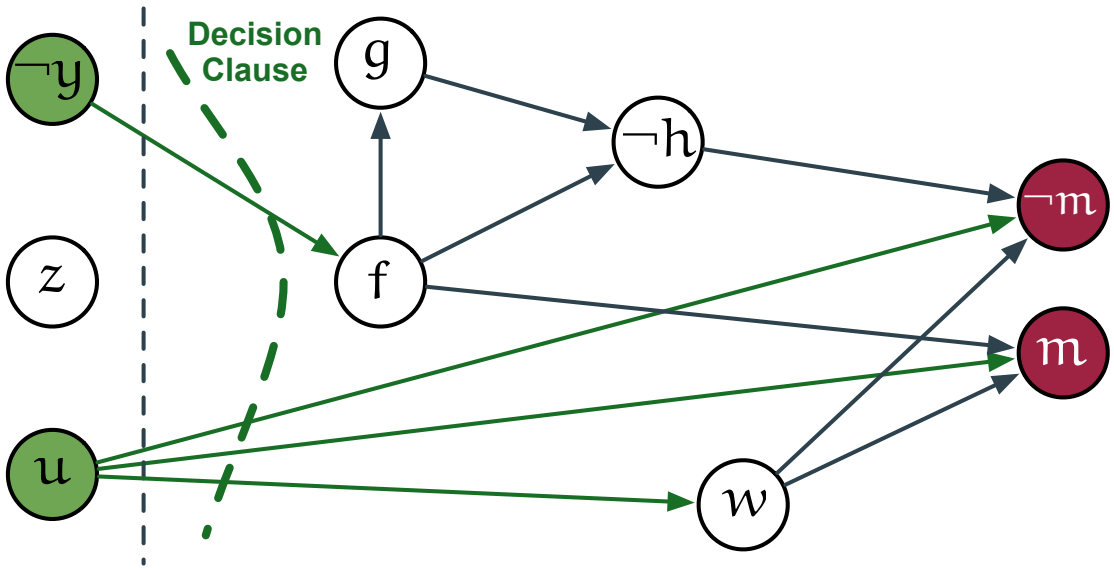
- Jeder Schnitt im Implikationsgraphen, der Entscheidungsvariablen von Konfliktvariablen trennt, ist ein gültiger Schnitt/Cut.
- Bilde eine neue Klausel aus allen Variablen, die eine ausgehende Kante durch den Cut besitzen.



Neue Klausel: $\neg(u \wedge f) = (\neg u \vee \neg f)$

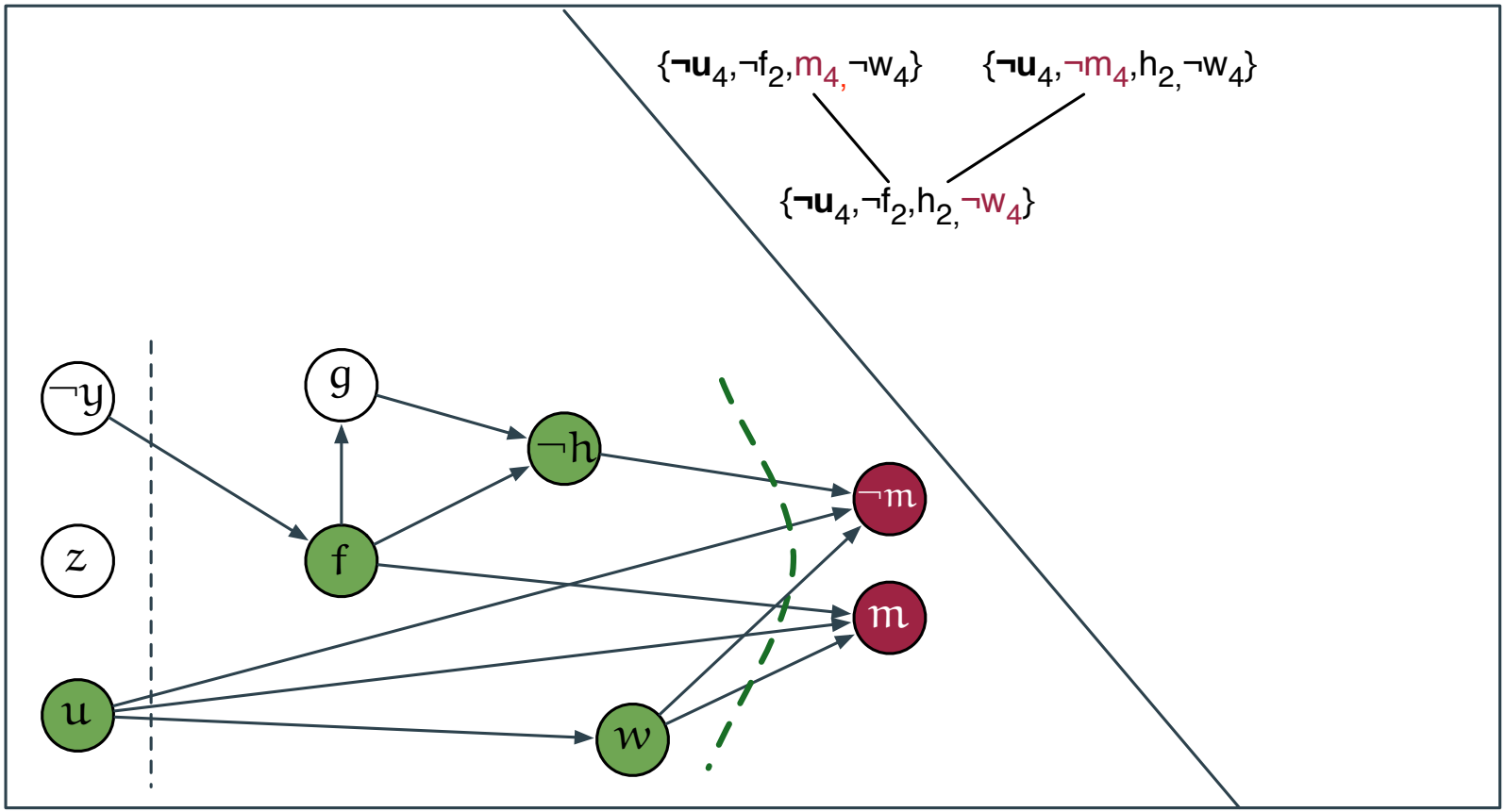
Und welche Klausel lernen wir jetzt?

- Jeder Schnitt im Implikationsgraphen, der Entscheidungsvariablen von Konfliktvariablen trennt, ist ein gültiger Schnitt/Cut.
- Bilde eine neue Klausel aus allen Variablen, die eine ausgehende Kante durch den Cut besitzen.

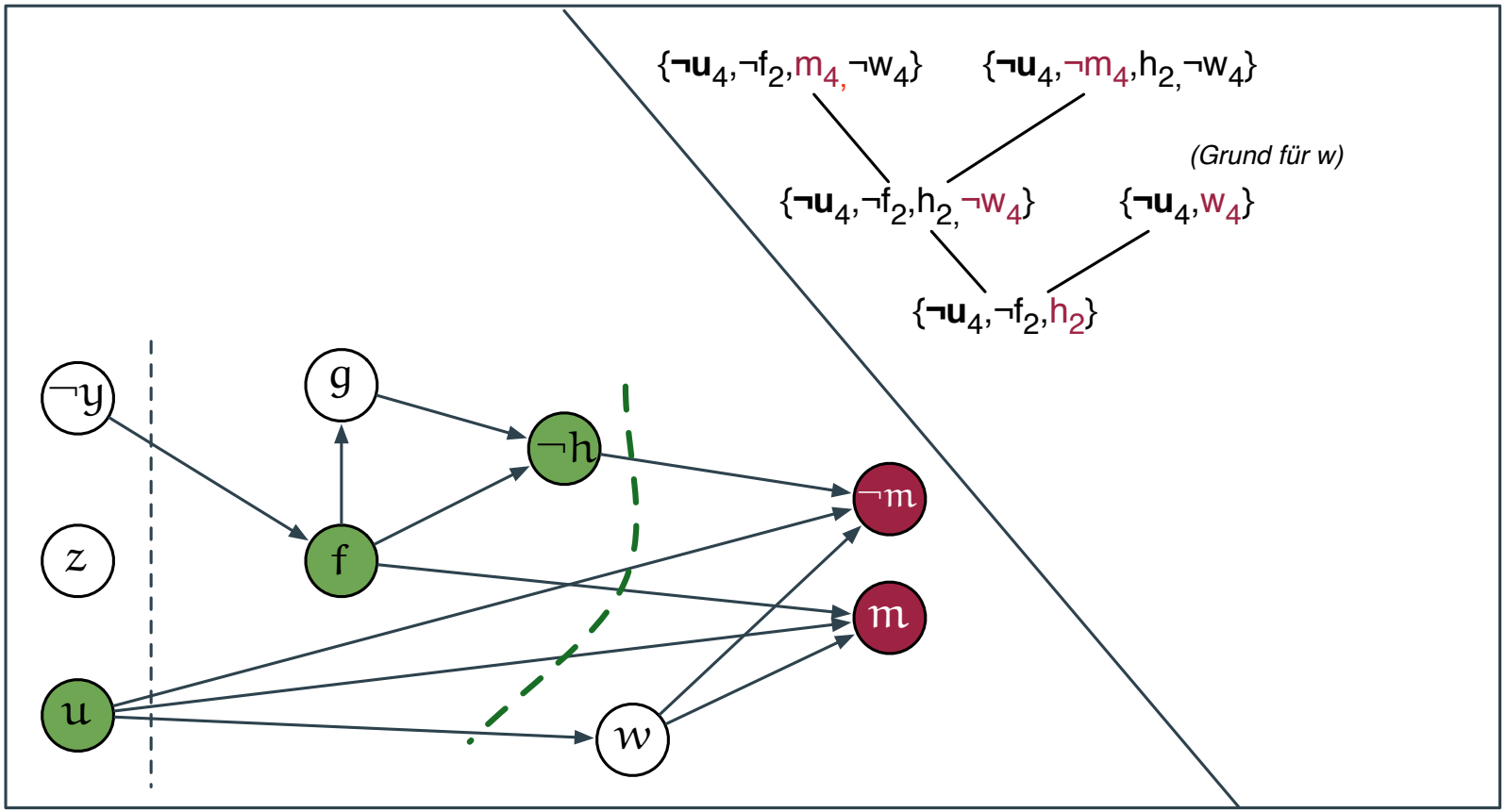


Neue Klausel: $\neg(u \wedge \neg y) = (\neg u \vee y)$

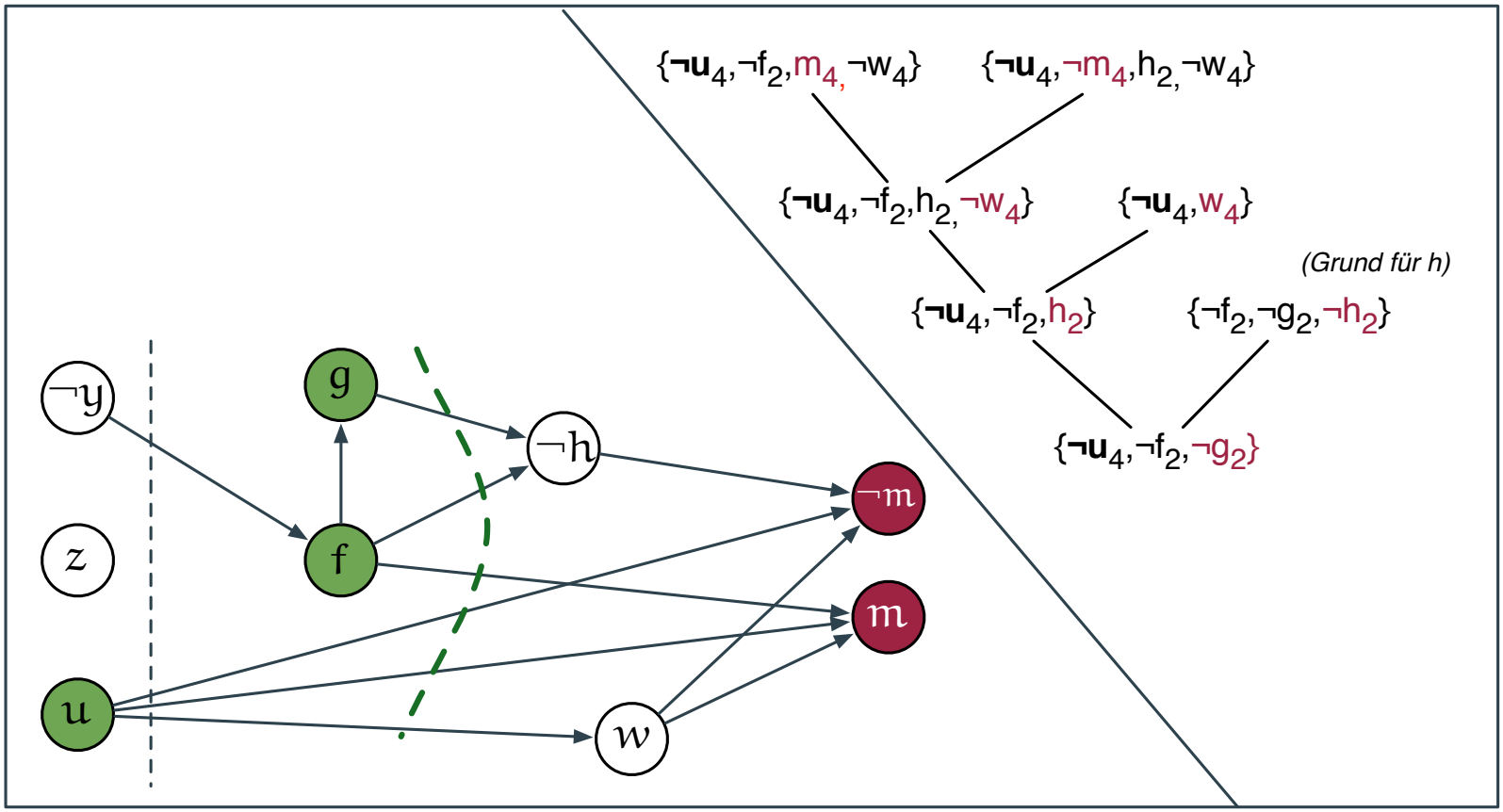
Zusammenhang zwischen Resolution und dem Implikationsgraphen



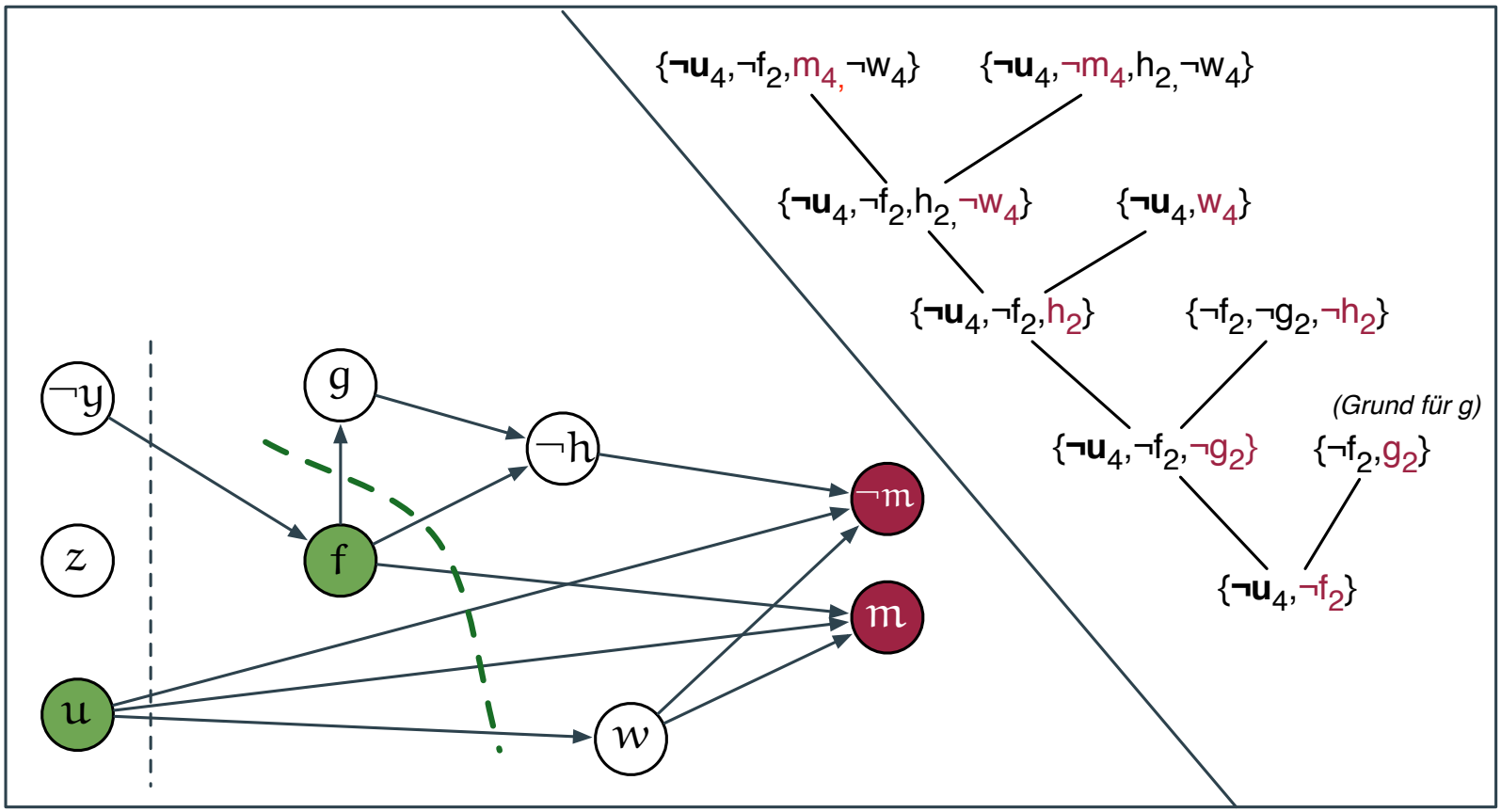
Zusammenhang zwischen Resolution und dem Implikationsgraphen



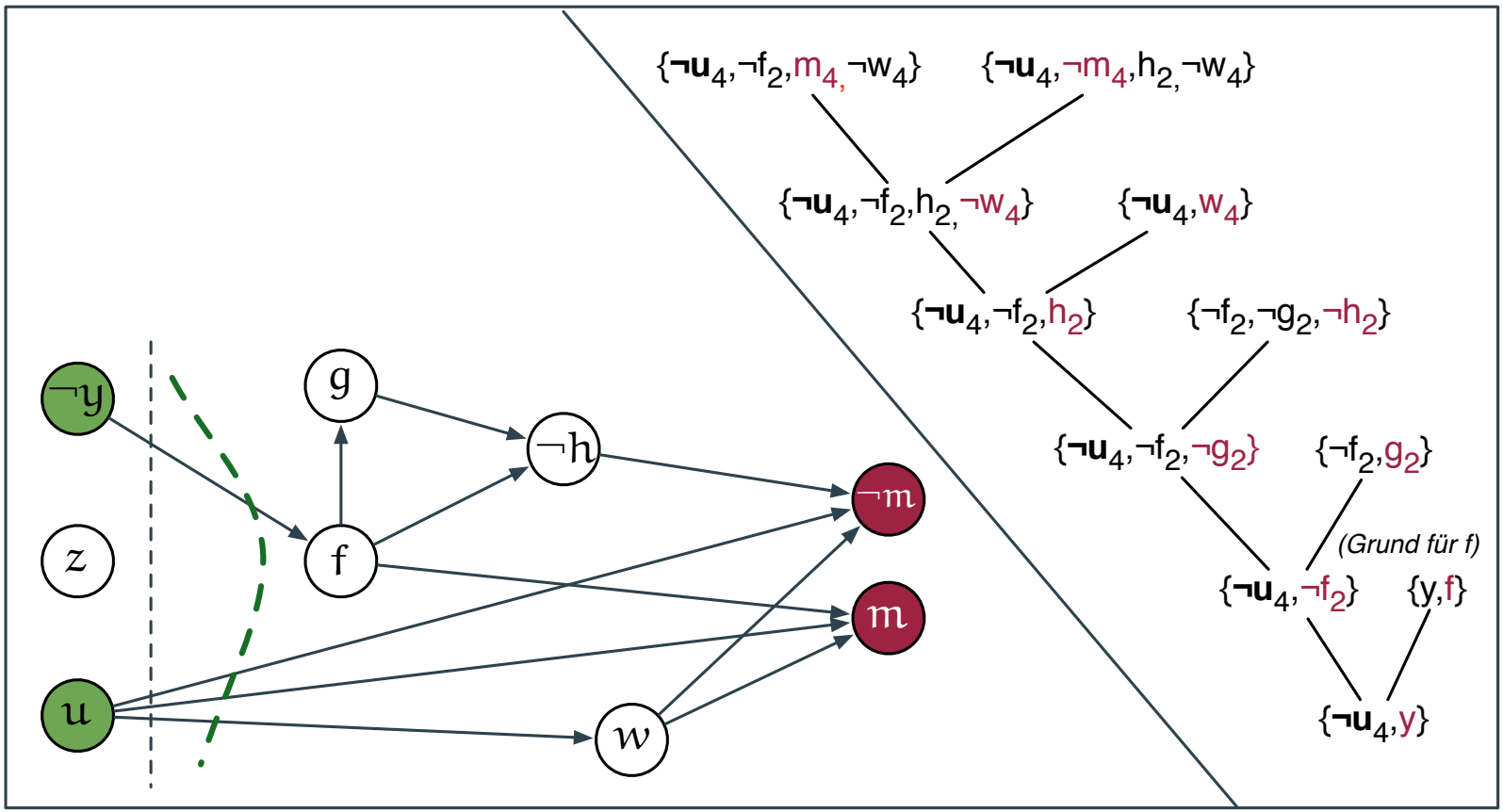
Zusammenhang zwischen Resolution und dem Implikationsgraphen



Zusammenhang zwischen Resolution und dem Implikationsgraphen



Zusammenhang zwischen Resolution und dem Implikationsgraphen



Verschiedene Lernstrategien

Abhängig davon, wie lange man Resolution betreibt bzw. wo man den Implikationsgraphen schneidet, ergeben sich verschiedene neue Klauseln:

- **Decision Clause:** enthält nur Entscheidungsvariablen
- **First New Cut Clause:** enthält auf der einen Seite nur die beiden Konfliktvariablen
- **1-UIP Clause (Unique Implication Point):** In der neuen Klausel ist genau eine Variable auf höchstem Decision Level. Diese Klausel wird nach Backtracking eine Unit Klausel.

Einsetzen der 1-UIP Klausel

- Hinzufügen der neuen Klausel zur Klauselmenge
- Backtracking zu höchstem Level der neuen Klausel, das $<$ als das aktuelle Decision Level ist
- Unit Propagation (Bei 1-UIP ist die neue Klausel nach dem Backtracking immer unit, d.h. mindestens eine UP wird ausgeführt)

Auflösung des Beispiels mit 1-UIP

$\{\neg u, w\}$
$\{\neg u, \neg m, h, \neg w\}$
$\{\neg u, \neg f, m, \neg w\}$
$\{\neg f, \neg g, \neg h\}$
$\{y, f\}$
$\{\neg f, g\}$
$\vdots$
$\{\neg x, a\}$
$\{\neg x, b, \neg a\}$
$\{\neg x, \neg a, \neg b, c\}$
$\{\neg a, \neg d\}$
$\{\neg b, d, \neg e\}$
$\vdots$

Lev	Var	Wert	Reason
1	x	true	decision
	a	true	$\{\neg x, a\}$
	b	true	$\{\neg x, b, \neg a\}$
	c	true	$\{\neg x, \neg a, \neg b, c\}$
	d	false	$\{\neg a, \neg d\}$
	e	false	$\{\neg b, d, \neg e\}$
2	y	false	decision
	f	true	$\{y, f\}$
	g	true	$\{\neg f, g\}$
	h	false	$\{\neg f, \neg g, \neg h\}$
3	z	true	decision
4	u	true	decision
	w	true	$\{\neg u, w\}$
	m	false	$\{\neg u, \neg m, h, \neg w\}$

Konflikt: $\{\neg u, \neg f, m, \neg w\}$

Auflösung des Beispiels mit 1-UIP

$\{\neg u, w\}$
$\{\neg u, \neg m, h, \neg w\}$
$\{\neg u, \neg f, m, \neg w\}$
$\{\neg f, \neg g, \neg h\}$
$\{y, f\}$
$\{\neg f, g\}$
$\vdots$
$\{\neg x, a\}$
$\{\neg x, b, \neg a\}$
$\{\neg x, \neg a, \neg b, c\}$
$\{\neg a, \neg d\}$
$\{\neg b, d, \neg e\}$
$\vdots$
$\{\neg u, \neg f, h\}$

Lev	Var	Wert	Reason
1	x	true	decision
	a	true	$\{\neg x, a\}$
	b	true	$\{\neg x, b, \neg a\}$
	c	true	$\{\neg x, \neg a, \neg b, c\}$
	d	false	$\{\neg a, \neg d\}$
	e	false	$\{\neg b, d, \neg e\}$
2	y	false	decision
	f	true	$\{y, f\}$
	g	true	$\{\neg f, g\}$
	h	false	$\{\neg f, \neg g, \neg h\}$
3	z	true	decision
4	u	true	decision
	w	true	$\{\neg u, w\}$
	m	false	$\{\neg u, \neg m, h, \neg w\}$

Konflikt: $\{\neg u, \neg f, m, \neg w\}$

1. Lerne die 1-UIP Klausel: $\{\neg u, \neg f, h\}$

Auflösung des Beispiels mit 1-UIP

$\{\neg u, w\}$
$\{\neg u, \neg m, h, \neg w\}$
$\{\neg u, \neg f, m, \neg w\}$
$\{\neg f, \neg g, \neg h\}$
$\{y, f\}$
$\{\neg f, g\}$
$\vdots$
$\{\neg x, a\}$
$\{\neg x, b, \neg a\}$
$\{\neg x, \neg a, \neg b, c\}$
$\{\neg a, \neg d\}$
$\{\neg b, d, \neg e\}$
$\vdots$
$\{\neg u, \neg f, h\}$

Lev	Var	Wert	Reason
1	x	true	decision
	a	true	$\{\neg x, a\}$
	b	true	$\{\neg x, b, \neg a\}$
	c	true	$\{\neg x, \neg a, \neg b, c\}$
	d	false	$\{\neg a, \neg d\}$
	e	false	$\{\neg b, d, \neg e\}$
2	y	false	decision
	f	true	$\{y, f\}$
	g	true	$\{\neg f, g\}$
	h	false	$\{\neg f, \neg g, \neg h\}$

1. Lerne die 1-UIP Klausel: $\{\neg u, \neg f, h\}$
2. Backjumping zu Level 2 (kleinstes Level in der 1-UIP < 4)

Auflösung des Beispiels mit 1-UIP

$\{\neg u, w\}$
$\{\neg u, \neg m, h, \neg w\}$
$\{\neg u, \neg f, m, \neg w\}$
$\{\neg f, \neg g, \neg h\}$
$\{y, f\}$
$\{\neg f, g\}$
$\vdots$
$\{\neg x, a\}$
$\{\neg x, b, \neg a\}$
$\{\neg x, \neg a, \neg b, c\}$
$\{\neg a, \neg d\}$
$\{\neg b, d, \neg e\}$
$\vdots$
$\{\neg u, \neg f, h\}$

Lev	Var	Wert	Reason
1	x	true	decision
	a	true	$\{\neg x, a\}$
	b	true	$\{\neg x, b, \neg a\}$
	c	true	$\{\neg x, \neg a, \neg b, c\}$
	d	false	$\{\neg a, \neg d\}$
	e	false	$\{\neg b, d, \neg e\}$
2	y	false	decision
	f	true	$\{y, f\}$
	g	true	$\{\neg f, g\}$
	h	false	$\{\neg f, \neg g, \neg h\}$
	u	false	$\{\neg u, \neg f, h\}$

1. Lerne die 1-UIP Klausel: $\{\neg u, \neg f, h\}$
2. Backjumping zu Level 2 (kleinstes Level in der 1-UIP < 4)
3. 1-UIP ist nun unit und UP auf u kann ausgeführt werden

Auflösung des Beispiels mit 1-UIP

$\{\neg u, w\}$
$\{\neg u, \neg m, h, \neg w\}$
$\{\neg u, \neg f, m, \neg w\}$
$\{\neg f, \neg g, \neg h\}$
$\{y, f\}$
$\{\neg f, g\}$
$\vdots$
$\{\neg x, a\}$
$\{\neg x, b, \neg a\}$
$\{\neg x, \neg a, \neg b, c\}$
$\{\neg a, \neg d\}$
$\{\neg b, d, \neg e\}$
$\vdots$
$\{\neg u, \neg f, h\}$

Lev	Var	Wert	Reason
1	x	true	decision
	a	true	$\{\neg x, a\}$
	b	true	$\{\neg x, b, \neg a\}$
	c	true	$\{\neg x, \neg a, \neg b, c\}$
	d	false	$\{\neg a, \neg d\}$
	e	false	$\{\neg b, d, \neg e\}$
2	y	false	decision
	f	true	$\{y, f\}$
	g	true	$\{\neg f, g\}$
	h	false	$\{\neg f, \neg g, \neg h\}$
	u	false	$\{\neg u, \neg f, h\}$

1. Lerne die 1-UIP Klausel: $\{\neg u, \neg f, h\}$
2. Backjumping zu Level 2 (kleinstes Level in der 1-UIP < 4)
3. 1-UIP ist nun unit und UP auf u kann ausgeführt werden
4. Alle Klauseln sind nun erfüllt

Der CDCL Algorithmus

Algorithmus: cdcl(C)

Eingabe: Klauselmenge C
Ausgabe: SAT wenn C erfüllbar ist, UNSAT sonst

```
level = 0 ;                                   // Setze initiales Level auf 0
β = ∅ ;                                       // Alle Variablen sind unbelegt
while true do
    UP(C, β) ;                                          // Unit Propagation
    if C contains an empty clause e then
        level = analyzeConf(e, C) ;                     // Lerne neue Klausel
        if level == −1 then
            return UNSAT
        backtrack(level) ;                                  // Backtracking
    else
        if eval(β, C) then
            return SAT
        level = level + 1 ;                              // Erhöhe Level, ...
        choose x ∉ β ;                             // ... wähle neue Variable,
        β = β ∪ [x ↦ true] ;                          // ... belege Variable
```

Der Konfliktanalyse Algorithmus

Algorithmus: analyzeConf(e, C)

Eingabe: Leere Klausel e, Klauselmenge C
Ausgabe: Das neue Backtracking Level

---

if *current decision level* == *0* **then**
  return -1
lv = the last assigned variable;
newclause = `resolve`(e,`reason`(lv)) ; `// Erste Resolution`
while *the stop criterion for newclause is not met* **do**
  cv = `chooseLiteral`(newclause);
  newclause = `resolve`(newclause,`reason`(cv)) ; `// Resolutionen`
C = C $\cup$ {newclause} ; `// Füge neue Klausel hinzu`
level = `computeBacktrackLevel`(newclause) **return** *level*;

- **Stopkriterium bei 1UIP:** Nur noch eine Variable auf höchstem Decision Level l_h
- **Backtracklevel:** max{level | level $< l_h$}

Korrektheit des CDCL Algorithmus—1

Theorem (Erfüllbarkeit einer Formel)

Gilt $\texttt{cdcl}(C) = \texttt{SAT}$*, so ist die Klauselmenge* C *erfüllbar.*

Beweis.

1. Algorithmus gibt SAT zurück, gdw. $\texttt{eval}(\beta, C)$
2. Neu gelernte Klauseln ändern die Semantik der originalen Klauselmenge C nicht

$\Rightarrow$ β muss erfüllende Belegung für die originale Klauselmenge sein

□

Korrektheit des CDCL Algorithmus—2

Theorem (Unerfüllbarkeit einer Formel)

Gilt `cdcl`(C) = `UNSAT`, *so ist die Klauselmenge* C *eine Kontradiktion.*

Beweis.

1. Algorithmus gibt `UNSAT` zurück, gdw. Konflikt auf Level 0
2. Von diesem Konflikt ausgehend kann man Resolventen berechnen (vgl. Algorithmus `analyzeConflict`)
   - Auf Level 0 gibt es keine Entscheidungsvariablen

   $\Rightarrow$ Es gibt zu jeder Variable eine Reason
   - Pro Durchlauf der Schleife wird eine Variable eliminiert

   $\Rightarrow$ Nach spätestens n Schritten ist eine leere Klausel erzeugt (n = Anzahl der Literale in der Konfliktklausel)
3. Leere Klausel ist direkt aus der Formel per Resolution erzeugbar

$\Rightarrow$ C ist unerfüllbar

□

Korrektheit der Ausgabe des Algorithmus: **Partielle Korrektheit!**

Termination des CDCL Algorithmus—1

Theorem (Termination des Algorithmus)

Der Algorithmus $\texttt{cdcl}(C)$ *terminiert für jede beliebige Eingabeklauselmenge* C.

Vorbereitung des Beweises—1

- $k(l)$: Anzahl der Variablen, die auf Level l insgesamt belegt wurden
- n: Anzahl der Variablen der Originalformel.
- Auf jedem Decision Level (außer Level 0) muss mindestens eine Variable belegt werden

Folgende Aussagen gelten offensichtlich:

1. $\forall l\,[(0 \leq l \leq n) \rightarrow k(l) \leq n]$
2. $\forall l\,[(l > n) \rightarrow k(l) = 0]$
3. $\sum_{l=0}^{n} k(l) = n$

Termination des CDCL Algorithmus—2

Vorbereitung des Beweises — 2

Wir betrachten die Funktion

$$f = \sum_{l=0}^{n} \frac{k(l)}{(n+1)^l}.$$

Diese Funktion bildet “eine Art lexikographische Ordnung”: Für zwei Variablenbelegungen α und β der selben Formel gilt $f_\alpha > f_\beta$ gdw.

1. ein Decision Level d mit $0 \leq d < n$ existiert, in dem $k_\alpha(d) > k_\beta(d)$ gilt, und
2. für alle Decision Levels l mit $0 \leq l < d$ gilt, dass $k_\alpha(l) = k_\beta(l)$.

Intuitiv gewichtet diese Funktion Variablenbelegungen auf kleinen Decision Levels höher. Die folgende Ungleichung hält:

$$\frac{1}{(n+1)^j} > \frac{n}{(n+1)^{j+1}} \tag{1}$$

D.h. von zwei verschiedene Variablenbelegungen hat diejenige den größeren Wert von f, bei der die Belegungen auf die kleineren Decision Levels konzentriert sind.

Termination des CDCL Algorithmus—3

Beweis.

$\mathtt{f}$ steigt während Solving Prozess streng monoton an

1. Kein Konflikt: in jedem Schritt werden neue Variablen auf höchstem Level belegt
2. Konflikt: Backtracking zu kleinerem Level, aber dort sicher eine zusätzliche Belegung (1-UIP Klausel ist dann unit)
   - Aus Ungleichung (1) folgt, dass auch in diesem Fall $\mathtt{f}$ streng monoton steigt

Da $\mathtt{f}$ nur eine endliche Anzahl von verschiedenen Werten annehmen kann, muss der Algorithmus also terminieren. □

Partielle Korrektheit + Termination $\Rightarrow$ Totale Korrektheit

Aktivitätsheuristiken

Wähle die Variable, die am aktivsten ist.

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

Idee!

- Variablen, die oft in Konflikten vorkommen, spielen eine herausragende Rolle und sollten zuerst belegt werden
- Variablen haben verschieden großen Einfluss auf eine Formel (Bsp: Softwareverifikation, Variable die über einen großen `if-then-else`-Branch entscheidet beeinflusst Ergebnis mehr als andere Variablen)

Beispiel für Aktivitätsheuristik:

- **VSIDS** (variable state independent decaying sum)

VSIDS

Berechnung:

- Jede Variable bekommt eine Aktivität act zugewiesen
- Initial ist act Anzahl der Vorkommen der Variable
- Für jede gelernte Klausel $\{x_1, x_2, \ldots x_n\}$: erhöhe act für alle x_i um konstanten Betrag c (1)
- Teile periodisch alle Aktivitäten durch einen Faktor f (2)
- Wähle Variable mit höchster Aktivität

Erklärung:

(1) Aktivität einer Variable wird höher, je häufiger sie in Konflikten auftaucht

(2) Aktivität aller Variablen wird von Zeit zu Zeit verringert, d.h. Konflikte die in der nahen Vergangenheit aufgetreten sind, werden bevorzugt

Damit:

- Wahl der Variablen, die **aktuell** (im aktuellen Suchbaum) am **häufigsten in Konflikten** vorkam

Erweiterung der Aktivität auf Klauseln

⚠ Problem!

- Während eines CDCL Durchgangs werden viele neue Klauseln gelernt
- Viele dieser Klauseln werden im Folgenden nicht mehr benutzt

$\Rightarrow$ Klauselmenge bläht sich unnötig auf

Lösung: Clause Deletion

- Bewerte neu gelernte Klauseln ebenfalls mit Aktivität
- Initiale Aktivität einer neu gelernten Klausel ist eine Konstante c
- Jedes mal, wenn die Klausel in der Berechnung einer neuen Klausel vorkommt (d.h. in der Resolution benutzt wird), wird die Aktivität erhöht
- Periodisch werden alle Aktivitäten verringert
- Periodisch werden alle Klauseln mit geringer Aktivität (unter einem vorher bestimmten Schwellenwert) gelöscht

Gefahr von lokalen Minima

⚠ Lokale Minima

Mit VSIDS besteht die Gefahr des “Festfressens” in einem lokalen Minimum (1):

- Heuristik wählt immer aktuelle Konfliktvariablen aus, jedoch besteht ein größerer Konflikt ganz am Anfang des Suchbaums
- Bei einer großen Anzahl von Variablen kann ein Verlassen des lokalen Minimums sehr lange dauern

Lösung: Restarts

Der SAT Solver wird zurückgesetzt und neu gestartet, d.h.

- Alle Variablenbelegungen werden rückgängig gemacht
- Alle Aktivitäten von Variablen und Klauseln werden auf initialen Wert gesetzt
- **Aber:** Gelernte Klauseln bleiben erhalten
- Kriterium für Restart: Anzahl der gelernten Klauseln
- Um Termination zu gewährleisten: Erhöhen dieses Restart-Schwellenwertes nach jedem Restart

Zusammenfassung

Alles, was man für einen effizienten state-of-the-art SAT Solver braucht (so zu finden in z.B. MiniSAT):

1. Klauselllernen
2. Nicht-chronologisches Backtracking
3. Gute Auswahlheuristiken (z.B. VSIDS)
4. Effiziente Unit Propagation (z. B. watched literals)
5. Restarts mit Clause Deletions

Fazit:

- Im Gegensatz zu anderen Problemen (z.B. lineare Optimierung mit Simplex) kann SAT Solving auf wenige Kernpunkte reduziert werden.
- Einer der besten aktuellen SAT Solver - MiniSAT - kommt mit $<$ 1000 Zeilen C++ Code aus.

Literaturtipps

Literaturhinweis

- *Joao Marques-Silva, Ines Lynce & Sharad Malik.* **Chapter 4—CDCL Solvers in Handbook of Satisfiability.** IOS Press, 2009.
- *Lintao Zhang & Sharad Malik.* **Validating SAT solvers using an independent resolution-based checker: Practical implementations and other applications.** Proceedings of DATE '03. 2003.

Web Links

- `http://minisat.se/` — Kleiner und gut dokumentierter state-of-the-art SAT Solver in C++
- `http://www.sat4j.org/` — SAT Solver in Java, der z. B. in Eclipse eingesetzt wird
- `http://www.satlive.org/` — Wichtigste Quelle für SAT News, Konferenzen, Bücher, Paper, Benchmarks, Jobs, ...