SAT-Solving und Anwendungen

Non-CNF SAT-Solving

Prof. Dr. Wolfgang Küchlin
Dipl.-Inform. Christoph Zengler
Dipl.-Inf. Andreas Kübler

Universität Tübingen

7. Juni 2011

Weshalb Non-CNF SAT-Solving?

Viele Anwendungsprobleme liegen zunächst nicht in Normalform vor

- z.B. Produktkonfigurationsformeln aus der Automobilindustrie

$\Rightarrow$ Normalformkonversion erforderlich, bevor SAT-Solving erfolgen kann

Probleme

- Effiziente Konversion führt neue (Hilfs-)Variablen ein
- Sich ständig ändernde Formeln erfordern dauernd erneute Konversionen
- Strukturverluste durch die Normalformkonversion
  - behindern die Entwicklung problemspezifischer Verzweigungsheuristiken
  - behindern die Erklärung von Ergebnissen

Idee

- SAT-Solving direkt auf nicht-normalisierten Formeln
- Spare Konversionsschritte, erhalte Struktur

Änderungen gegenüber CNF SAT-Solving

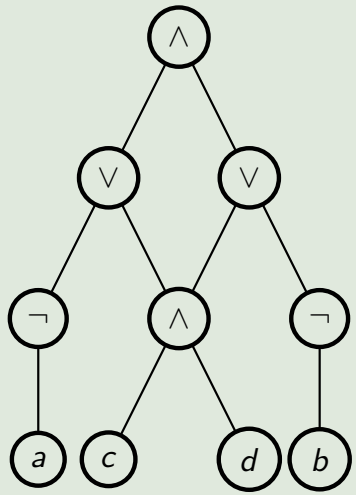
Formelrepräsentation:

- Klauseldarstellung nicht mehr möglich bzw. nicht ausreichend
- Stattdessen: DAG-Darstellung:
  - Jede Variable durch genau einen Knoten im DAG repräsentiert
  - Mehrfach vorkommende Teilformeln tauchen nur einmal im Graphen auf (Redundanzelimination/Kompaktifizierung)

  $\Rightarrow$ Einfacher Aufbau mit Hilfe einer Hashtabelle, kompakte Darstellung

Beispiel (Formelrepräsentation)

$$\begin{aligned} & (\neg a \vee (c \wedge d)) \\ \wedge \quad & (\neg b \vee (c \wedge d)) \end{aligned}$$



Änderungen gegenüber CNF SAT-Solving

Neue Inferenzregeln für Boolean Constraint Propagation:

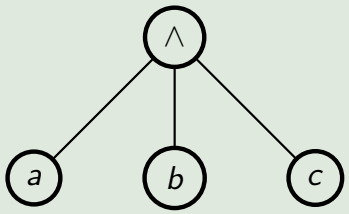
- Unitpropagation allein reicht nicht mehr aus

Beobachtung:

Korrespondenz Tseitin-Transformation — DAG-Darstellung

Ein innerer/Operator-Knoten entspricht einer Hilfsvariablen in der Tseitin-Transformation (ohne Reduktionsregel)

Beispiel



$$
\begin{aligned}
& f_\wedge \Leftrightarrow (a \wedge b \wedge c) \\
\equiv\ & (\neg f_\wedge \vee a) \wedge (\neg f_\wedge \vee b) \wedge (\neg f_\wedge \vee c) \wedge \\
& (f_\wedge \vee \neg a \vee \neg b \vee \neg c)
\end{aligned}
$$

Inferenzregeln für die Constraint Propagation

Idee

Aufgrund der o.g. Korrespondenz können auch inneren Knoten Wahrheitswerte zugewiesen und Inferenzregeln abgeleitet werden.

Beispiel (Regeln für $\wedge$)

1. Wird einem $\wedge$-Knoten der Wert **T** zugewiesen, so müssen alle seine Kinder den Wert **T** haben
2. Haben alle Kinder eines $\wedge$-Knotens den Wert **T**, so auch der $\wedge$-Knoten
3. Hat ein $\wedge$-Knoten den Wert **F** und alle Kinder bis auf eines den Wert **T**, so hat das verbliebene den Wert **F**
4. Hat eines der Kinder eines $\wedge$-Knotens den Wert **F**, so hat der $\wedge$-Knoten den Wert **F**

Ähnliche Regeln gelten für $\vee$ und $\neg$.

DPLL-Algorithmus für Non-CNF-Instanzen

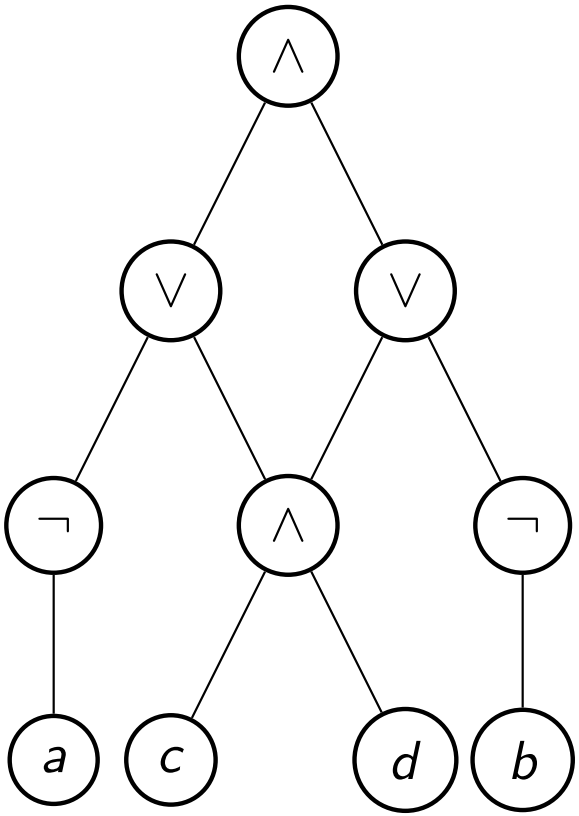
Algorithmus

Algorithm 1: DPLL

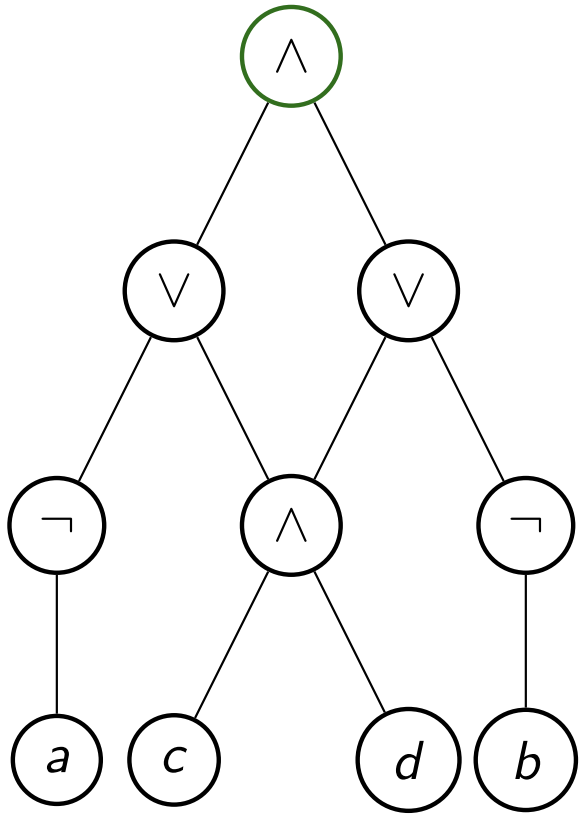
```
begin
    r ← assign(root, T, DECISION)                    /* r = Zustand des Beweisers */
    for ever do
        if r = OK then
            r ← propagate()                          /* Anwendung Inferenzregeln */
        if r = CONFLICT and restore()=UNSAT then
            return UNSAT
        ⟨v, b⟩ ← choose()                            /* Variablenselektion */
        r ← assign(v, b, DECISION)
        if r = SAT then
            return SAT
```

Konfliktauflösung mittels *restore()*: später

Beispiel

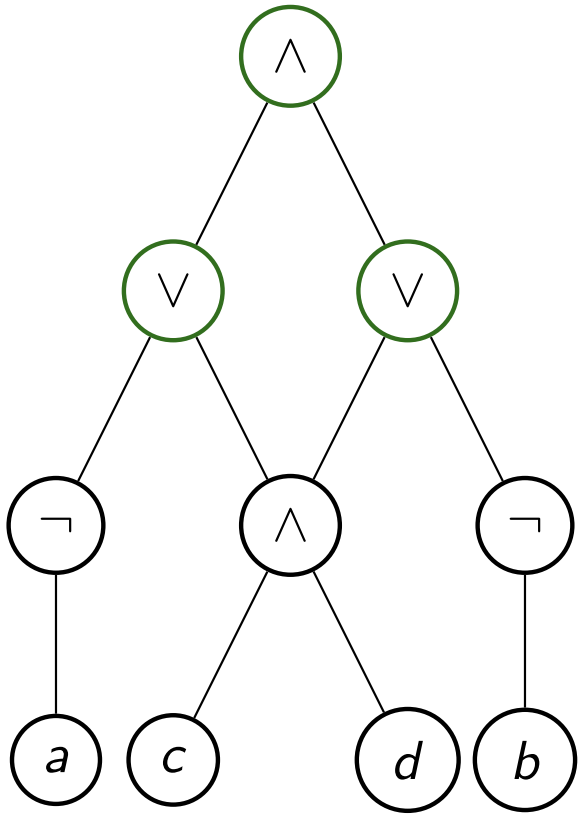


Beispiel



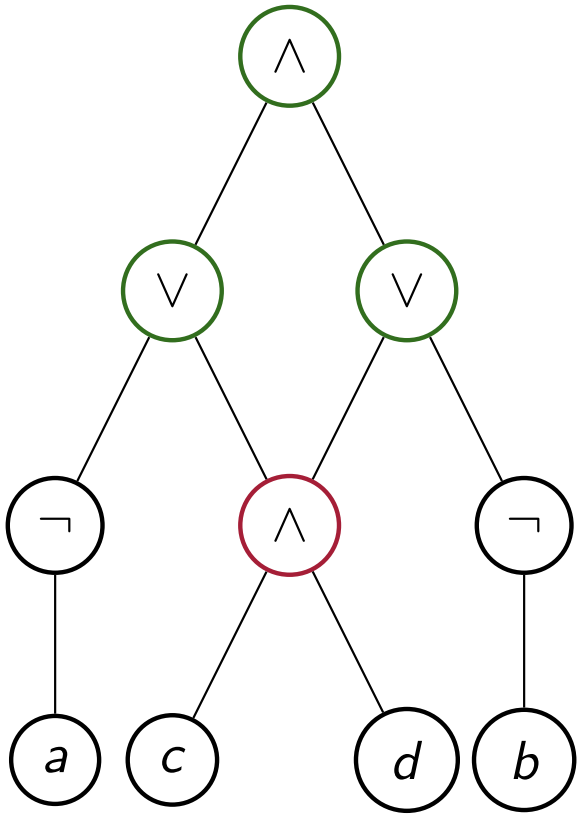
- Weise Wurzel **T** zu

Beispiel



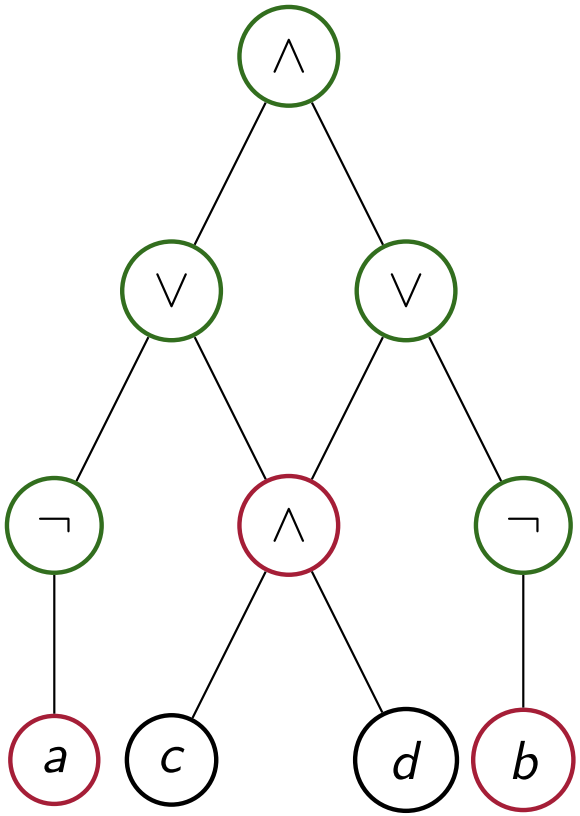
- Propagiere: $v(\vee_0) = v(\vee_1) = \mathbf{T}$

Beispiel



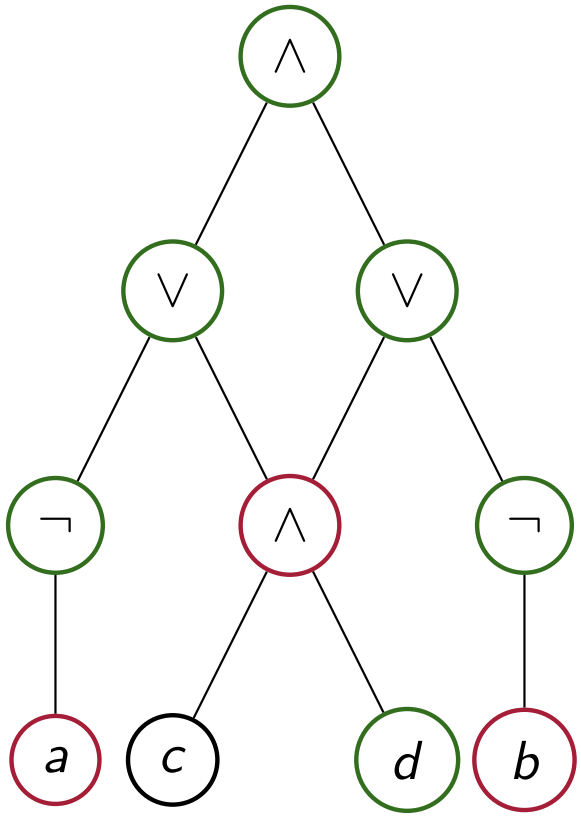
- Wähle $v(\wedge) = \mathbf{F}$

Beispiel



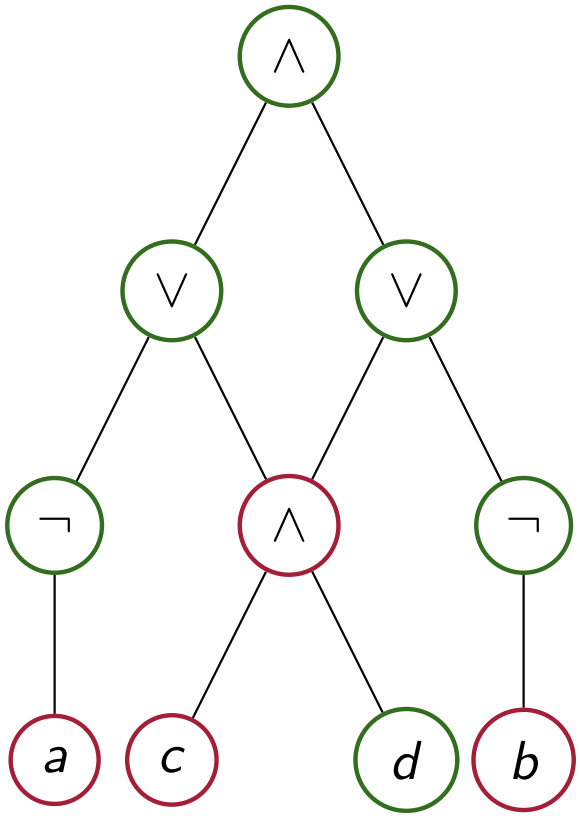
- Propagiere: $v(\neg_0) = v(\neg_1) = \mathbf{T}$, $v(a) = v(b) = \mathbf{F}$

Beispiel



- Wähle $v(d) = \mathbf{T}$

Beispiel



- Propagiere: $v(c) = \mathbf{F}$

Integration moderner Techniken

Durch Erweiterung der DAG-Datenstruktur um zusätzliche Informationen lassen sich in CNF-Solvern geläufige Techniken implementieren:

- Schnelle BCP durch Watched Literals
- Conflict driven clause learning

Darüber hinaus können auf dem DAG don't care Werte propagiert werden:

Regel: Don't care-Propagation

Hängt der Wert eines Knotens nicht mehr vom Wert einer der Teilformeln ab, so kann der Teilformel der Wert $*$ (don't care) zugewiesen werden

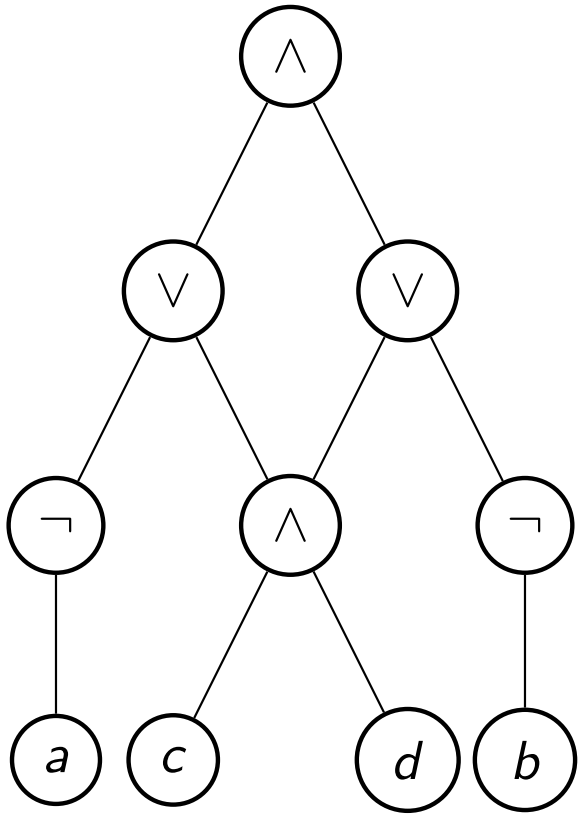
Beispiel (Don't care-Propagation)

Ist ein $\wedge$ ($\vee$) -Knoten mit **F** (**T**) beschriftet und hat eines seiner Kinder den Wert **F** (**T**) so hängt der Wert des Knotens nicht von seinen übrigen Kindern ab.

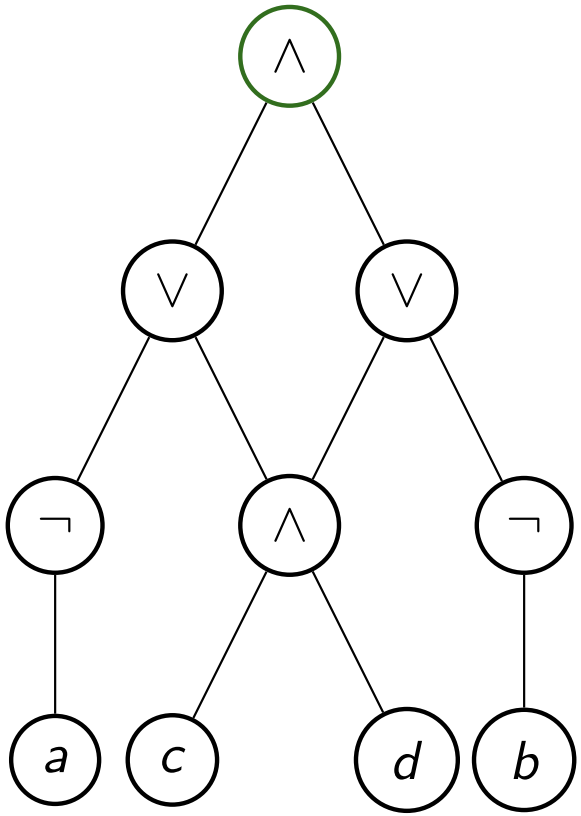
Don't care Teilformeln brauchen nicht weiter betrachtet zu werden.

- Implementierung: Watched-Literals-Schema

Beispiel: Don't care-Propagation

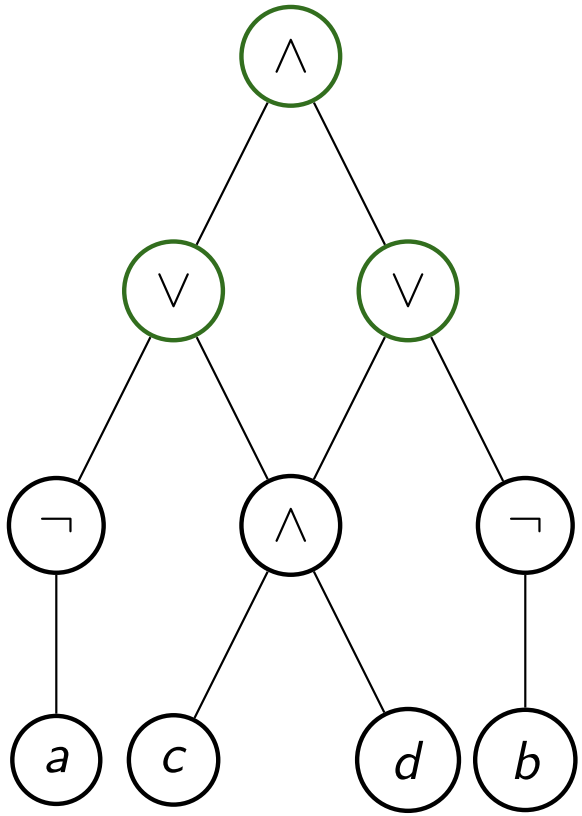


Beispiel: Don't care-Propagation



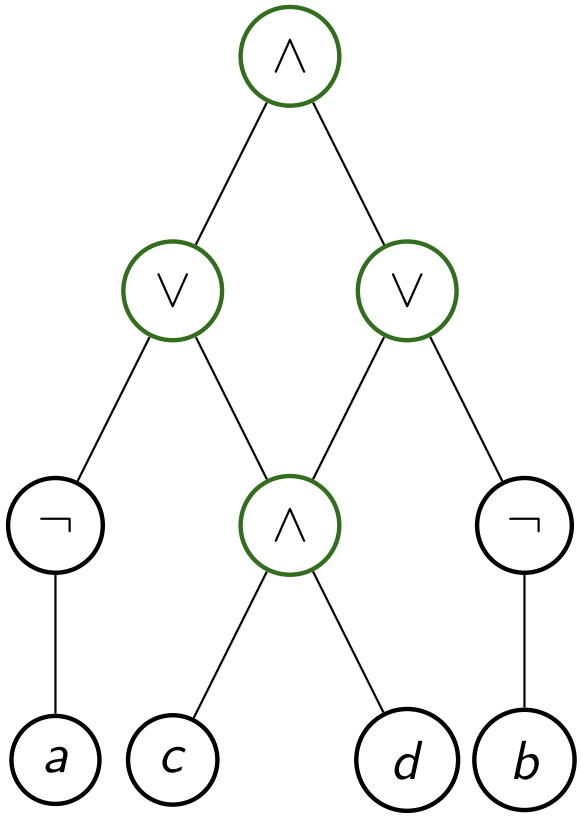
- Weise Wurzel **T** zu

Beispiel: Don't care-Propagation



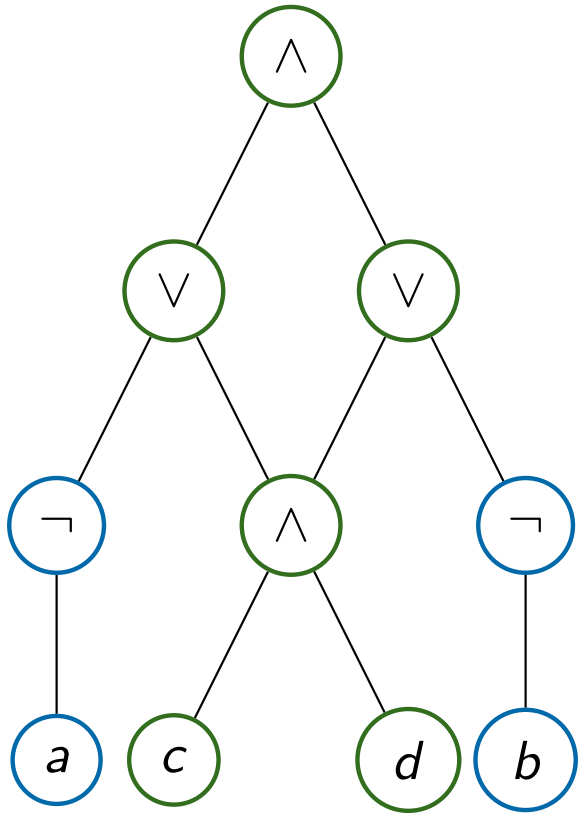
- Propagiere: $v(\vee_0) = v(\vee_1) = \mathbf{T}$

Beispiel: Don't care-Propagation



- Wähle $v(\wedge) = \mathbf{T}$

Beispiel: Don't care-Propagation



- Propagiere: $v(\neg_0) = v(\neg_1) = v(a) = v(b) = *$, $v(c) = v(d) = \mathbf{T}$

Grundidee des Konfliktlernens

Erinnerung: Ein Konflikt tritt auf, wenn ein Knoten bereits belegt ist und mit einem widersprüchlichen Wert belegt werden soll.

- Wir ermitteln eine Menge von Variablenbelegungen, die für den Konflikt verantwortlich sind und dem Algorithmus nützen.
- Ausgehend von den beiden Belegungen der Konfliktvariablen
  - Zurückverfolgen der Gründe für die Belegungen
  - Notieren der jeweils ursächlichen Variablenbelegungen in „dynamischem NoGood“
- 1UIP-Lernen: Beende die Rückverfolgung, sobald das NoGood nur noch eine einzige (Entscheidungs-)Variable auf höchster Ebene enthält.
- Die gelernte Klausel ist die Disjunktion der negierten Variablen im NoGood (die Konjunktion dieser Variablenbelegungen soll nie mehr auftreten).
- Eine 1UIP-Klausel erzwingt in der Folge eine andere Belegung der letzten Entscheidungsvariable vor dem Konflikt.

Gründe für Belegungen

- Merke bei der Belegung eines Knotens den **Grund** für die Belegung
  - Grund besteht aus: Typ des Grundes + ggf. andere Variablen, die die Belegung erzwingen (unmittelbar ursächliche Variablen).
- Tritt ein Konflikt auf, so liegen zwei Gründe vor:
  1. Grund der bisherigen Belegung
  2. Grund für die Belegung, die den Konflikt verursacht
- Typisierung der Gründe:

Decision	Setzen der Variable aufgrund einer Entscheidung/Variablenselektion
Parent	Setzen der Variable aufgrund des Wertes eines Elternknotens
Child	Setzen der Variable aufgrund des Wertes eines Kindknotens
NoGood	Setzen der Variable aufgrund eines gelernten NoGood

Gründe für Belegungen

Informationen für die Berechnung einer Menge ursächlicher Knoten für eine Wertzuweisung:

- Typ des Grundes
- Wert des Elternknotens bzw. der Kindknoten
- Werte der Geschwisterknoten

Beispiel (Child)

Einem $\vee$-Knoten wurde mit Grund „Child“ ein Wert zugewiesen. Ist der Wert des Kindes **T**, so ist das Kind ursächlich für die Zuweisung des Wertes **T** an den $\vee$-Knoten, andernfalls sind alle Kinder ursächlich für die Zuweisung des Wertes **F** an den $\vee$-Knoten.

Beispiel (Parent)

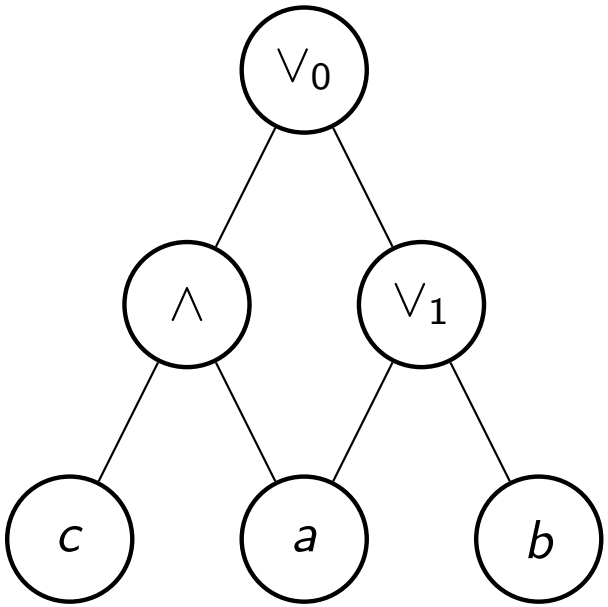
Einem Kind eines $\wedge$-Knotens mit Wert **F** wurde mit Grund „Parent“ der Wert **F** zugewiesen. Dann sind der $\wedge$-Knoten und alle Geschwisterknoten des betrachteten Kindes (haben Wert **T**) Ursachen für die Belegung.

Konstruktion des NoGood

- Initialisiere das NoGood mit den Konfliktvariablen. (Es sind mindestens 2 Variablen auf höchster Ebene vorhanden).
- Solange es geht:
  - Identifiziere eine Variable höchster Entscheidungsebene im NoGood, die keine Entscheidungsvariable ist.
  - Ersetze diese Variable durch die Variablenmenge, die für ihre Belegung unmittelbar ursächlich war.
- Konstruiere die gelernte Klausel aus dem NoGood. Darin gibt es eine einzige (Entscheidungs-)Variable (UIP) auf größter Ebene.
- Die Backtrack-Ebene BE ist die numerisch größte Ebene aller Variablen im NoGood außer der UIP.
- Falls BE=0 ist das Endergebnis UNSAT.
- Entferne vom Zuweisungsstack alle Ebenen größer als BE.
- Fahre auf der Backtrack-Ebene fort: statt der vormals nötigen Entscheidungs-Zuweisung an die UIP-Variable ermöglicht die gelernte Klausel nun eine weitere Werte-Propagation, die den vormaligen Zuweisungswert an die UIP-Variable umkehrt.

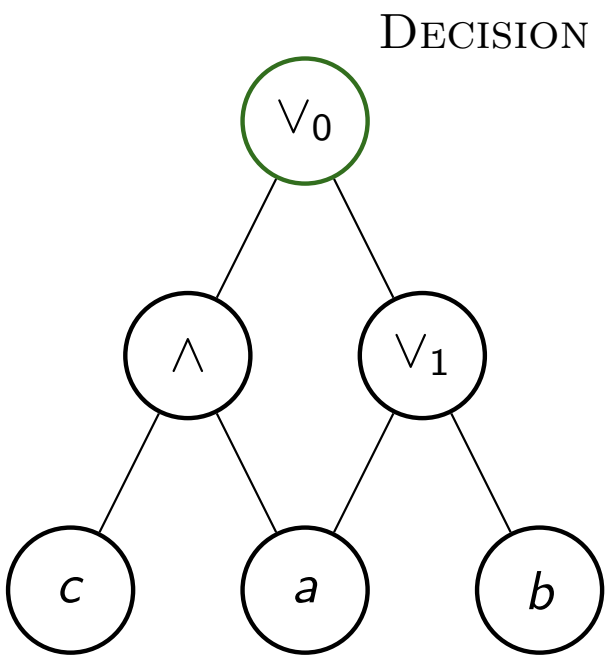
Rücksetzen des Zuweisungsstacks und Aufzeichnen des NoGood erfolgen in der Praxis gleichzeitig, Ebene für Ebene!

Beispiel: Konfliktlernen



Lv	Var	Val	Grund	Ursache

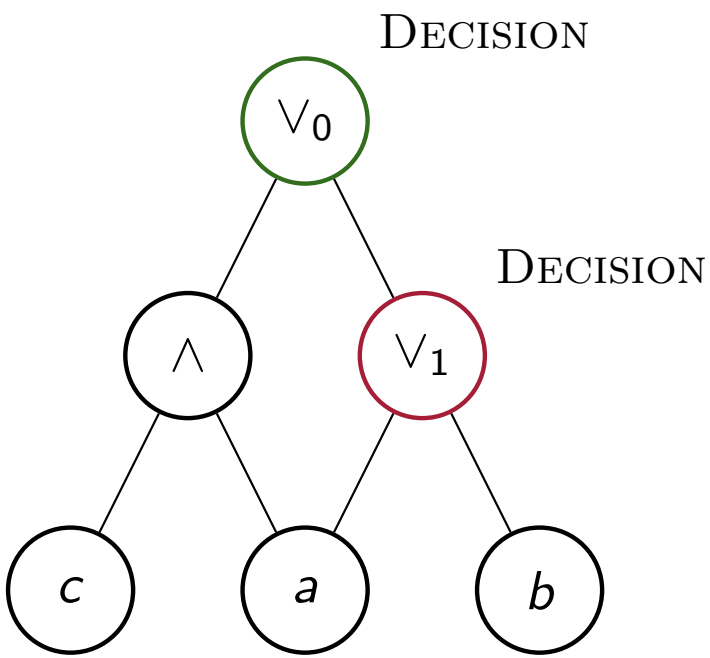
Beispiel: Konfliktlernen



Lv	Var	Val	Grund	Ursache
1	$\vee_0$	**T**	DECISION	

- Weise Wurzel **T** zu
- Zuweisungsstack: $\vee_0 = \mathbf{T}@1$

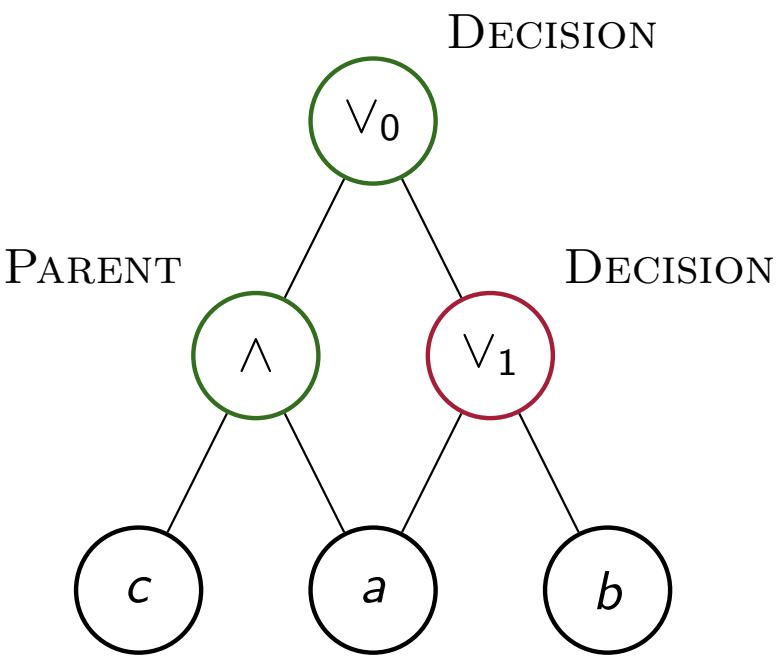
Beispiel: Konfliktlernen



Lv	Var	Val	Grund	Ursache
1	$\vee_0$	**T**	Decision	
2	$\vee_1$	**F**	Decision	

- Entscheidung: $v(\vee_1) = \mathbf{F}$
- Zuweisungsstack: $\vee_0 = \mathbf{T}@1$, $\vee_1 = \mathbf{F}@2$

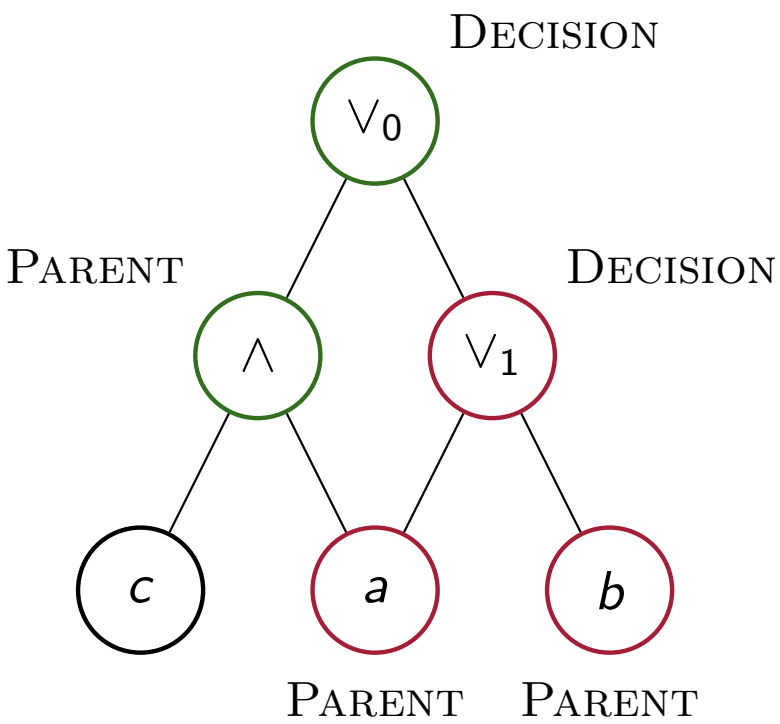
Beispiel: Konfliktlernen



Lv	Var	Val	Grund	Ursache
1	$\vee_0$	**T**	Decision	
2	$\vee_1$	**F**	Decision	
	$\wedge$	**T**	Parent	$\{\vee_0 = \mathbf{T}, \vee_1 = \mathbf{F}\}$

- Implikation: v($\wedge$)=**T**
- Zuweisungsstack: $\vee_0 = \mathbf{T}@1$, $\vee_1 = \mathbf{F}@2$, $\wedge = \mathbf{T}@2$

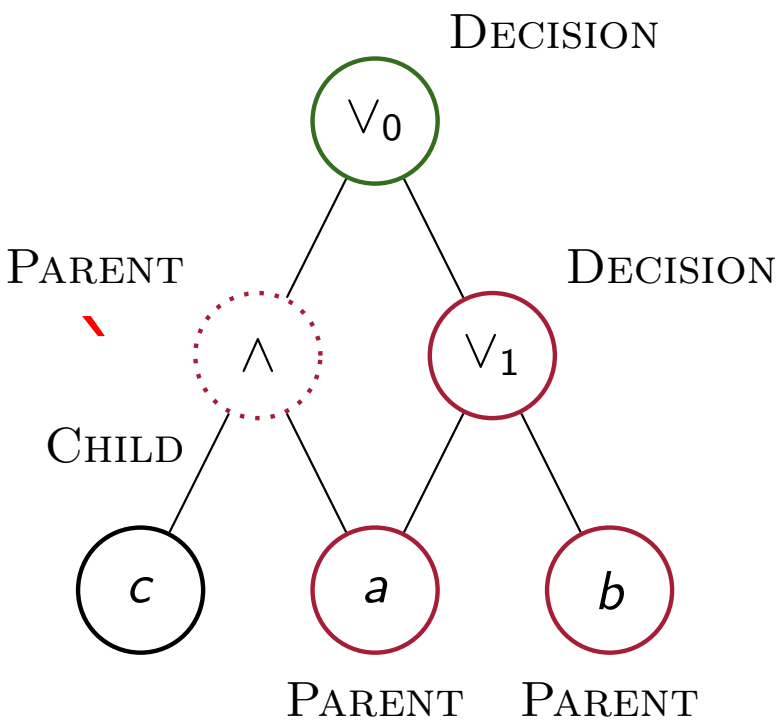
Beispiel: Konfliktlernen



Lv	Var	Val	Grund	Ursache
1	$\vee_0$	**T**	Decision	
2	$\vee_1$	**F**	Decision	
	$\wedge$	**T**	Parent	$\{\vee_0 = \mathbf{T}, \vee_1 = \mathbf{F}\}$
	a	**F**	Parent	$\{\vee_1 = \mathbf{F}\}$
	b	**F**	Parent	$\{\vee_1 = \mathbf{F}\}$

- Implikation: v(a)=**F**, v(b)=**F**
- Zuweisungsstack: $\vee_0 = \mathbf{T}@1$, $\vee_1 = \mathbf{F}@2$, $\wedge = \mathbf{T}@2$, $a = \mathbf{F}@2$, $b = \mathbf{F}@2$

Beispiel: Konfliktlernen

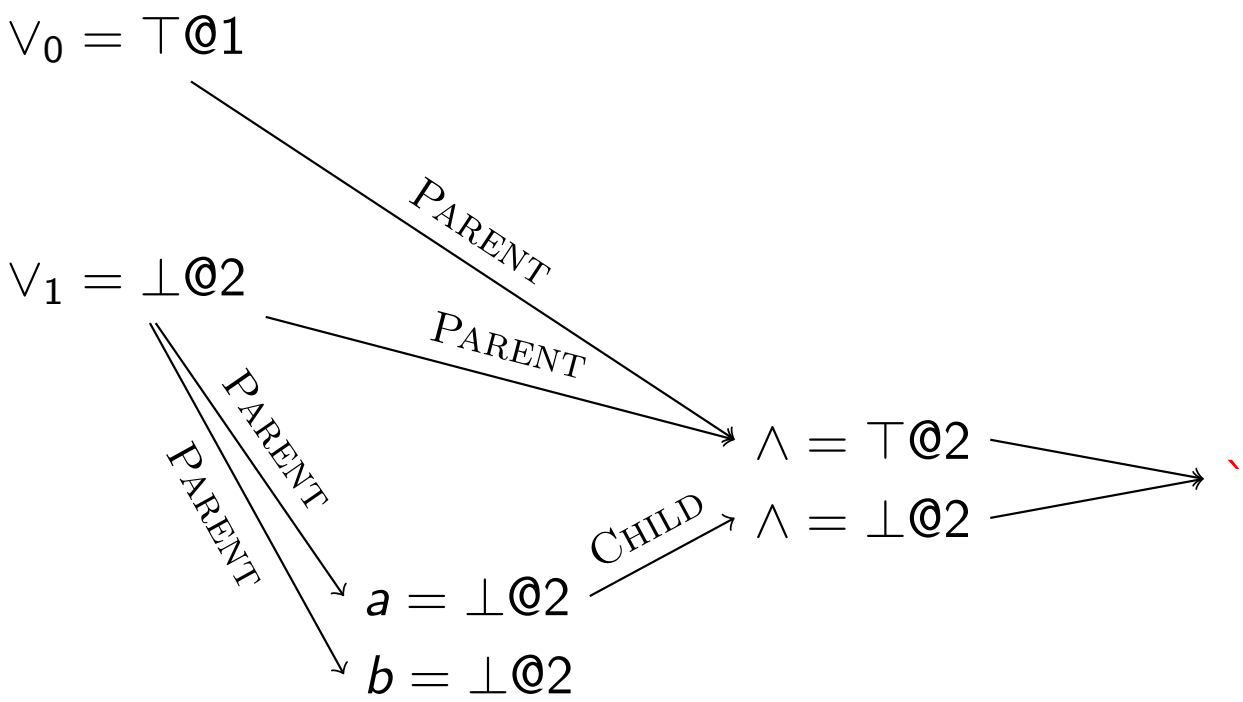


Lv	Var	Val	Grund	Ursache
1	$\vee_0$	**T**	Decision	
2	$\vee_1$	**F**	Decision	
	$\wedge$	**T**	Parent	$\{\vee_0 = \mathbf{T}, \vee_1 = \mathbf{F}\}$
	a	**F**	Parent	$\{\vee_1 = \mathbf{F}\}$
	b	**F**	Parent	$\{\vee_1 = \mathbf{F}\}$
	$\wedge$	**F**	Child	$\{a = \mathbf{F}\}$

- Implikation: v($\wedge$)=**F**, Widerspruch!
- Zuweisungsstack: $\vee_0 = \mathbf{T}@1$, $\vee_1 = \mathbf{F}@2$, $\wedge = \mathbf{T}@2$, $a = \mathbf{F}@2$, $b = \mathbf{F}@2$

Beispiel: Implikationsgraph

(Implizit vorhandener) Implikationsgraph der Konfliktsituation:



Beispiel: Aufzeichnen des NoGood

1. Notieren der Ursachen für $\wedge = \mathbf{T}@2$, $\wedge = \mathbf{F}@2$:

$$\underbrace{\{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2\}}_{\text{Grund für } \wedge = \mathbf{T}@2} \cup \underbrace{\{a = \mathbf{F}@2\}}_{\text{Grund für } \wedge = \mathbf{F}@2} = \{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2, a = \mathbf{F}@2\}$$

Beispiel: Aufzeichnen des NoGood

1. Notieren der Ursachen für $\wedge = \mathbf{T}@2$, $\wedge = \mathbf{F}@2$:

$$\underbrace{\{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2\}}_{\text{Grund für } \wedge = \mathbf{T}@2} \cup \underbrace{\{a = \mathbf{F}@2\}}_{\text{Grund für } \wedge = \mathbf{F}@2} = \{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2, a = \mathbf{F}@2\}$$

2. Mache $b = \mathbf{F}@2$ rückgängig, b ist nicht notiert, also nicht am Konflikt beteiligt

Beispiel: Aufzeichnen des NoGood

1. Notieren der Ursachen für $\wedge = \mathbf{T}@2$, $\wedge = \mathbf{F}@2$:

$$\underbrace{\{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2\}}_{\text{Grund für } \wedge = \mathbf{T}@2} \cup \underbrace{\{a = \mathbf{F}@2\}}_{\text{Grund für } \wedge = \mathbf{F}@2} = \{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2, a = \mathbf{F}@2\}$$

2. Mache $b = \mathbf{F}@2$ rückgängig, b ist nicht notiert, also nicht am Konflikt beteiligt
3. Mache $a = \mathbf{F}@2$ rückgängig: a ist am Konflikt beteiligt, ersetze a durch seine Ursache:

$$\begin{aligned} & (\{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2, a = \mathbf{F}@2\} \setminus \{a = \mathbf{F}@2\}) \cup \{\vee_1 = \mathbf{F}@2\} \\ = \quad & \{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2\} \end{aligned}$$

$\Rightarrow$ UIP gefunden ($\vee_1 = \mathbf{F}@2$), schreibe gegenwärtiges NoGood

Beispiel: Aufzeichnen des NoGood

1. Notieren der Ursachen für $\wedge = \mathbf{T}@2$, $\wedge = \mathbf{F}@2$:

$$\underbrace{\{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2\}}_{\text{Grund für } \wedge = \mathbf{T}@2} \cup \underbrace{\{a = \mathbf{F}@2\}}_{\text{Grund für } \wedge = \mathbf{F}@2} = \{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2, a = \mathbf{F}@2\}$$

2. Mache $b = \mathbf{F}@2$ rückgängig, b ist nicht notiert, also nicht am Konflikt beteiligt
3. Mache $a = \mathbf{F}@2$ rückgängig: a ist am Konflikt beteiligt, ersetze a durch seine Ursache:

$$\begin{aligned} & (\{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2, a = \mathbf{F}@2\} \setminus \{a = \mathbf{F}@2\}) \cup \{\vee_1 = \mathbf{F}@2\} \\ = \quad & \{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2\} \quad \Rightarrow \text{NoGood: } ([\vee_0 = \mathbf{F}] \vee [\vee_1 = \mathbf{T}]) \end{aligned}$$

$\Rightarrow$ UIP gefunden ($\vee_1 = \mathbf{F}@2$), schreibe gegenwärtiges NoGood

Beispiel: Aufzeichnen des NoGood

1. Notieren der Ursachen für $\wedge = \mathbf{T}@2$, $\wedge = \mathbf{F}@2$:

$$\underbrace{\{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2\}}_{\text{Grund für } \wedge = \mathbf{T}@2} \cup \underbrace{\{a = \mathbf{F}@2\}}_{\text{Grund für } \wedge = \mathbf{F}@2} = \{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2, a = \mathbf{F}@2\}$$

2. Mache $b = \mathbf{F}@2$ rückgängig, b ist nicht notiert, also nicht am Konflikt beteiligt
3. Mache $a = \mathbf{F}@2$ rückgängig: a ist am Konflikt beteiligt, ersetze a durch seine Ursache:

$$\begin{aligned} & (\{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2, a = \mathbf{F}@2\} \backslash \{a = \mathbf{F}@2\}) \cup \{\vee_1 = \mathbf{F}@2\} \\ = \quad & \{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2\} \quad \Rightarrow \text{NoGood: } ([\vee_0 = \mathbf{F}] \vee [\vee_1 = \mathbf{T}]) \end{aligned}$$

   $\Rightarrow$ UIP gefunden ($\vee_1 = \mathbf{F}@2$), schreibe gegenwärtiges NoGood
4. Mache $\wedge = \mathbf{T}@2$ und $\vee_1 = \mathbf{F}@2$ rückgängig, setze Level herunter

Beispiel: Aufzeichnen des NoGood

1. Notieren der Ursachen für $\wedge = \mathbf{T}@2$, $\wedge = \mathbf{F}@2$:

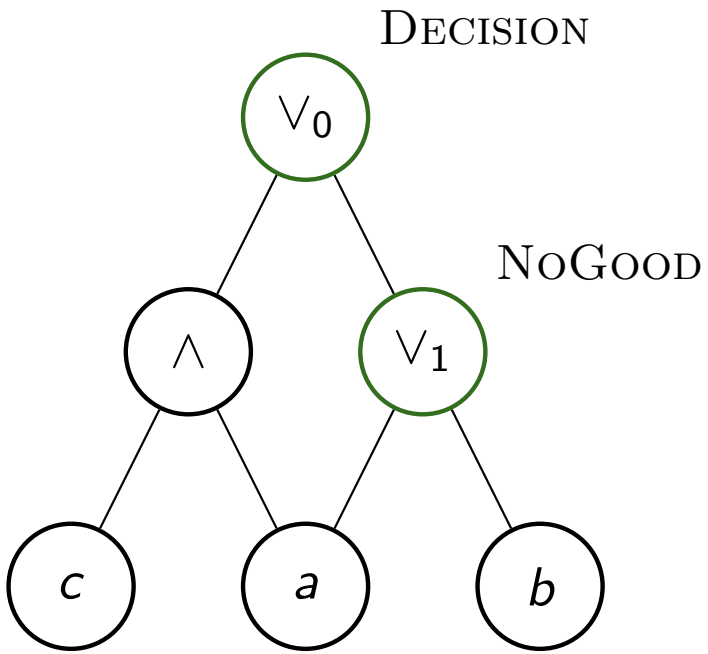
$$\underbrace{\{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2\}}_{\text{Grund für } \wedge = \mathbf{T}@2} \cup \underbrace{\{a = \mathbf{F}@2\}}_{\text{Grund für } \wedge = \mathbf{F}@2} = \{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2, a = \mathbf{F}@2\}$$

2. Mache $b = \mathbf{F}@2$ rückgängig, b ist nicht notiert, also nicht am Konflikt beteiligt
3. Mache $a = \mathbf{F}@2$ rückgängig: a ist am Konflikt beteiligt, ersetze a durch seine Ursache:

$$\begin{aligned} & (\{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2, a = \mathbf{F}@2\} \setminus \{a = \mathbf{F}@2\}) \cup \{\vee_1 = \mathbf{F}@2\} \\ = \quad & \{\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{F}@2\} \quad \Rightarrow \text{NoGood: } ([\vee_0 = \mathbf{F}] \vee [\vee_1 = \mathbf{T}]) \end{aligned}$$

   $\Rightarrow$ UIP gefunden ($\vee_1 = \mathbf{F}@2$), schreibe gegenwärtiges NoGood
4. Mache $\wedge = \mathbf{T}@2$ und $\vee_1 = \mathbf{F}@2$ rückgängig, setze Level herunter
5. Letzte Entscheidung ($\vee_1 = \mathbf{F}@2$) war notiert, kehre zurück

Beispiel: Konfliktauflösung

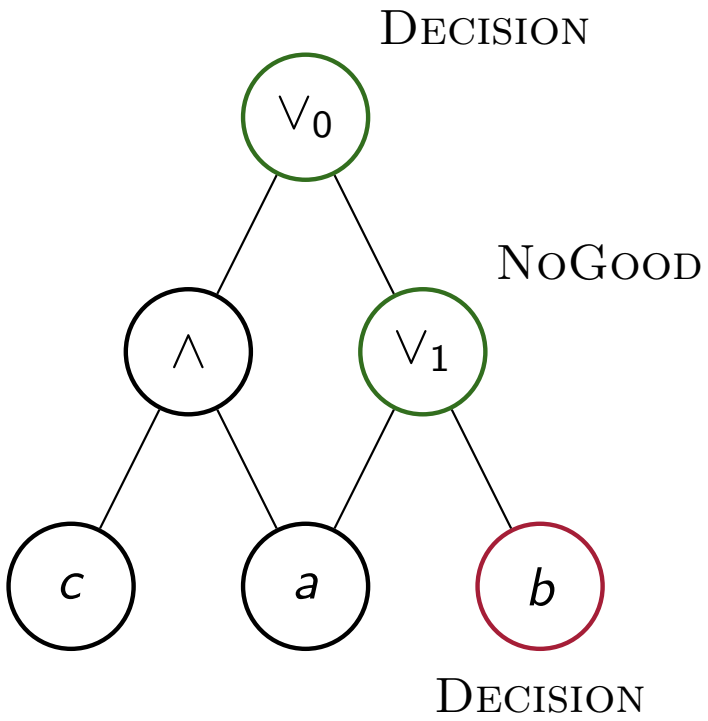


NoGood: $([\vee_0 = \mathbf{F}] \vee [\vee_1 = \mathbf{T}])$

Lv	Var	Val	Grund	Ursache
1	$\vee_0$	**T**	DECISION	
	$\vee_1$	**T**	NOGOOD	$\{\vee_0 = \mathbf{T}\}$

- Implikation: $v(\vee_1) = \mathbf{T}$
- Zuweisungsstack: $\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{T}@1$

Beispiel: Konfliktauflösung

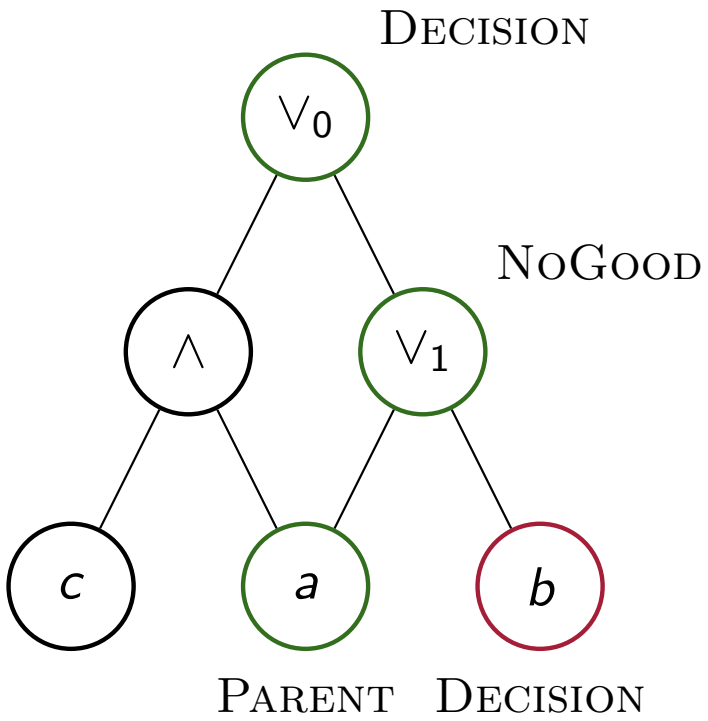


NoGood: $([\vee_0 = \mathbf{F}] \vee [\vee_1 = \mathbf{T}])$

Lv	Var	Val	Grund	Ursache
1	$\vee_0$	**T**	Decision	
	$\vee_1$	**T**	NoGood	$\{\vee_0 = \mathbf{T}\}$
2	b	**F**	Decision	

- Entscheidung: $v(b) = \mathbf{F}$
- Zuweisungsstack: $\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{T}@1, b = \mathbf{F}@2$

Beispiel: Konfliktauflösung

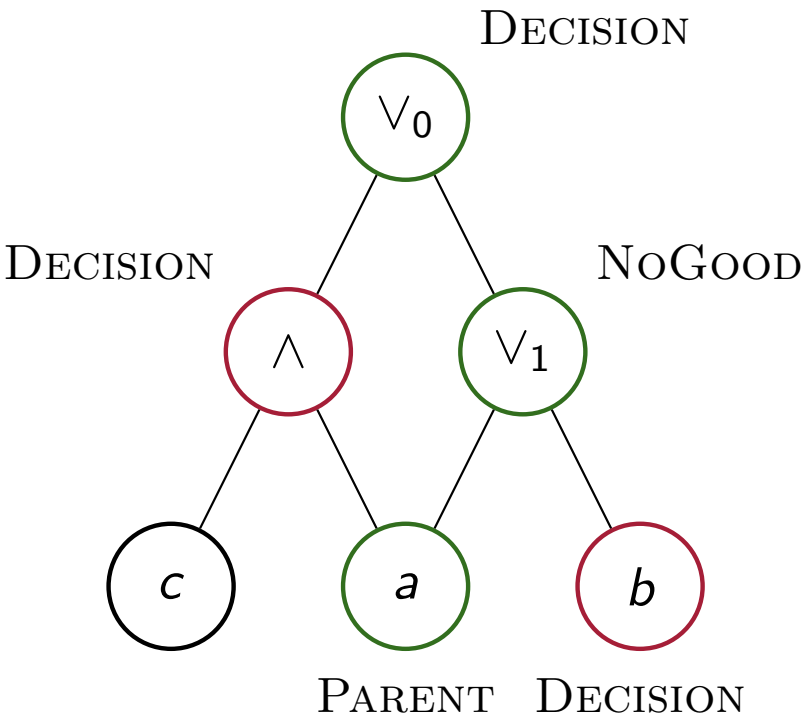


NoGood: $([\vee_0 = \mathbf{F}] \vee [\vee_1 = \mathbf{T}])$

Lv	Var	Val	Grund	Ursache
1	$\vee_0$	**T**	Decision	
	$\vee_1$	**T**	NoGood	$\{\vee_0 = \mathbf{T}\}$
2	b	**F**	Decision	
	a	**T**	Parent	$\{\vee_1 = \mathbf{T}, b = \mathbf{F}\}$

- Implikation: $v(a) = \mathbf{T}$
- Zuweisungsstack: $\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{T}@1, b = \mathbf{F}@2, a = \mathbf{T}@2$

Beispiel: Konfliktauflösung

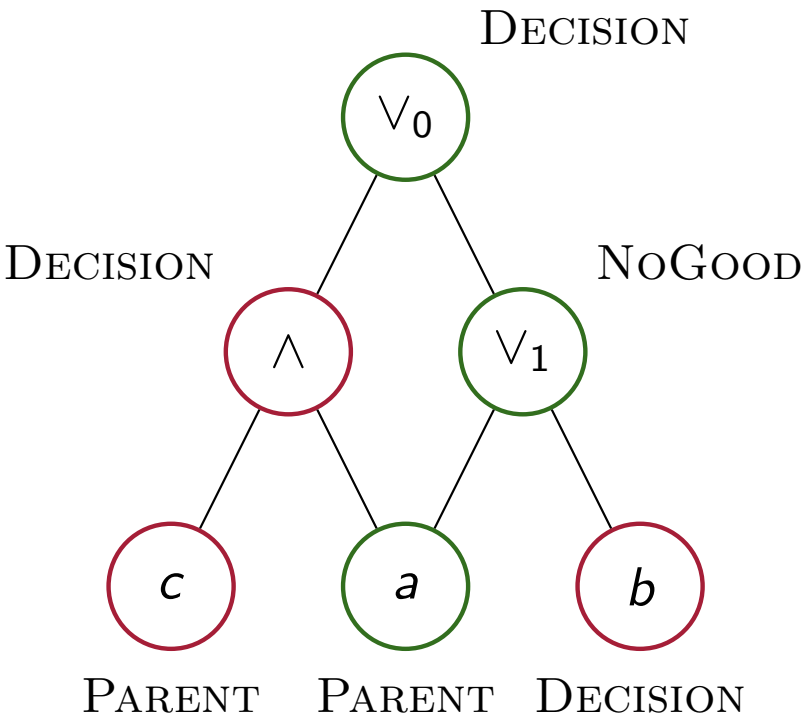


NoGood: $([\vee_0 = \mathbf{F}] \vee [\vee_1 = \mathbf{T}])$

Lv	Var	Val	Grund	Ursache
1	$\vee_0$	**T**	Decision	
	$\vee_1$	**T**	NoGood	$\{\vee_0 = \mathbf{T}\}$
2	b	**F**	Decision	
	a	**T**	Parent	$\{\vee_1 = \mathbf{T}, b = \mathbf{F}\}$
3	$\wedge$	**F**	Decision	

- Entscheidung: $v(\wedge) = \mathbf{F}$
- Zuweisungsstack: $\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{T}@1, b = \mathbf{F}@2, a = \mathbf{T}@2, \wedge = \mathbf{F}@3$

Beispiel: Konfliktauflösung



NoGood: $([\vee_0 = \mathbf{F}] \vee [\vee_1 = \mathbf{T}])$

Lv	Var	Val	Grund	Ursache
1	$\vee_0$	**T**	Decision	
	$\vee_1$	**T**	NoGood	$\{\vee_0 = \mathbf{T}\}$
2	b	**F**	Decision	
	a	**T**	Parent	$\{\vee_1 = \mathbf{T}, b = \mathbf{F}\}$
3	$\wedge$	**F**	Decision	
	c	**F**	Parent	$\{\wedge = \mathbf{F}, a = \mathbf{T}\}$

- Implikation: $v(c) = \mathbf{F}$
- Z.stack: $\vee_0 = \mathbf{T}@1, \vee_1 = \mathbf{T}@1, b = \mathbf{F}@2, a = \mathbf{T}@2, \wedge = \mathbf{F}@3, c = \mathbf{F}@3$

Zusammenfassung

Nutzen von Non-CNF-SAT-Solving:

- Keine CNF-Konversion erforderlich
- Erhält Formelstruktur

Praktische Umsetzung:

- Wichtige State-of-the-Art-Techniken aus CNF-Solvern können auf Non-CNF-Solver übertragen werden
  - Schnelle BCP
  - Conflict driven clause learning

Weitergehende Vorteile:

- Implementierung weiterer Operatoren ($\Rightarrow, \Leftrightarrow, \oplus, \ldots$) möglich
- Implementierung problemspezifischer Heuristiken

Nachteile:

- Implementierung komplizierter
- Kompliziertere Datenstruktur (DAG)