

SAT-Solving und Anwendungen

Non-CNF SAT-Solving

Prof. Dr. Wolfgang Küchlin
Dipl.-Inf. Andreas Kübler
Rouven Walter, M.Sc.

Universität Tübingen

04. Juni 2013



Weshalb Non-CNF SAT-Solving?

Viele Anwendungsprobleme liegen zunächst nicht in Normalform vor

- z.B. Produktkonfigurationsformeln aus der Automobilindustrie

⇒ 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

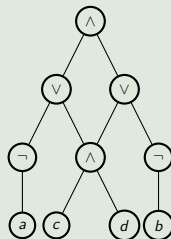
Änderungen gegenüber CNF SAT-Solving

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)
- ⇒ Einfacher Aufbau mit Hilfe einer Hashtabelle, kompakte Darstellung

Beispiel (Formelrepräsentation)

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



Änderungen gegenüber CNF SAT-Solving

Neue Inferenzregeln für Boolean Constraint Propagation:

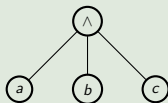
- Unitpropagation allein reicht nicht mehr aus

Beobachtung:

Korrespondenz Tseitin-Transformation — DAG-Darstellung

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

Beispiel



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

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

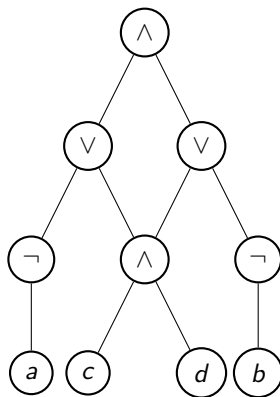
Algorithm 1: Non-CNF-DPLL

```

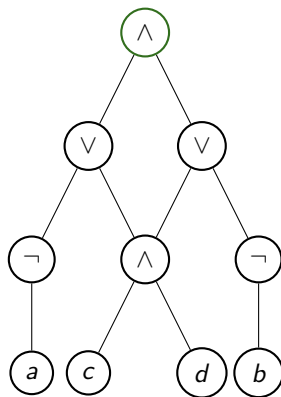
begin
   $r \leftarrow \text{assign}(\text{root}, \mathbf{T}, \text{DECISION})$                                 /*  $r$  = Zustand des Beweisers */
  for ever do
    if  $r = \text{OK}$  then
       $r \leftarrow \text{propagate}()$                                            /* Anwendung Inferenzregeln */
    if  $r = \text{SAT}$  then
      return SAT
    if  $r = \text{CONFLICT}$  then
       $r \leftarrow \text{resolveAndLearn}()$                                      /* UIP learning and backtracking */
      if  $r = \text{UNSAT}$  then
        return UNSAT
    else
       $\langle v, b \rangle \leftarrow \text{choose}()$                                /* Variablenselektion */
       $r \leftarrow \text{assign}(v, b, \text{DECISION})$ 

```

Beispiel

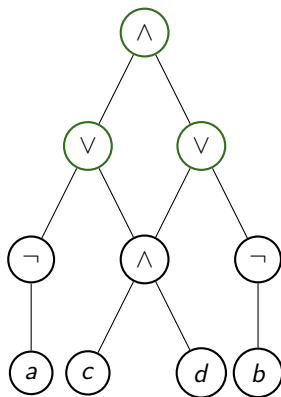


Beispiel



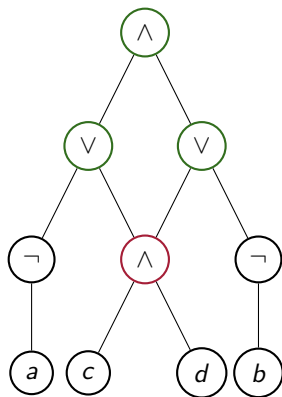
- Weise Wurzel **T** zu

Beispiel



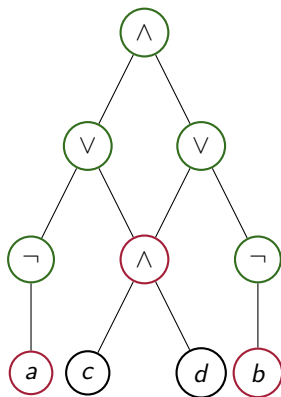
- Propagiere: $v(V_0) = v(V_1) = \mathbf{T}$

Beispiel



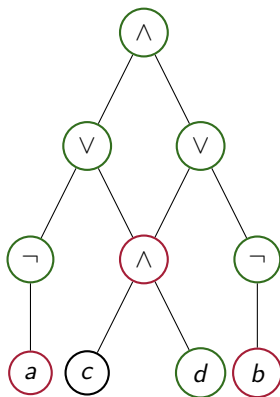
- Wähle $v(\wedge) = \mathbf{F}$

Beispiel



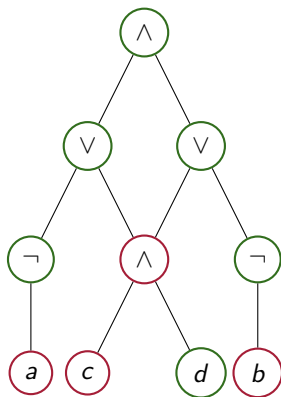
- Propagiere: $v(\neg_0) = v(\neg_1) = \mathbf{T}$, $v(a) = v(b) = \mathbf{F}$

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

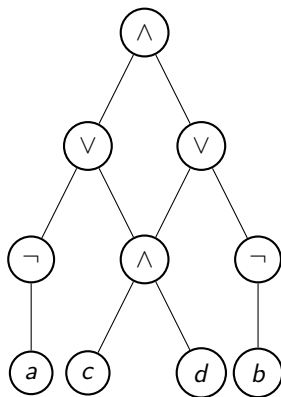
Beispiel (Don't care-Propagation)

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.

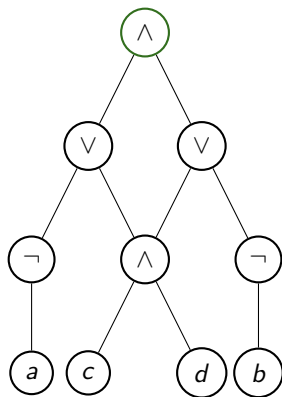
Don't care Teilformeln brauchen nicht weiter betrachtet zu werden.

- Implementierung: Watched-Literals-Schema

Beispiel: Don't care-Propagation

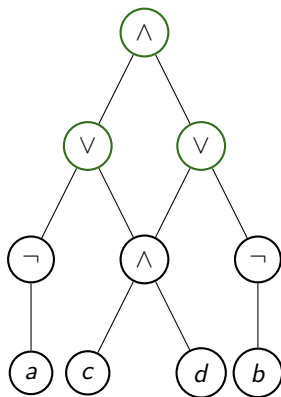


Beispiel: Don't care-Propagation



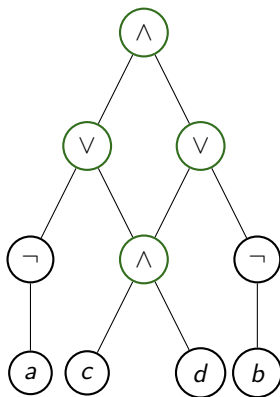
- Weise Wurzel **T** zu

Beispiel: Don't care-Propagation



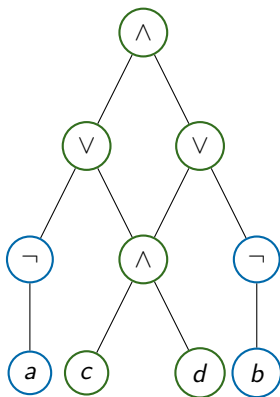
- Propagiere: $v(\vee_0) = v(\vee_1) = \mathbf{T}$

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 (durch Propagation) mit einem widersprüchlichen Wert belegt werden soll.

- Eine Entscheidungsvariable wurde belegt, und die Propagation führte zu einem Konflikt.
- Wir ermitteln eine Menge von Variablenbelegungen, die für den Konflikt verantwortlich sind und dem Algorithmus nützen.
- Ausgehend von den beiden Belegungen der Konfliktvariable
 - Notieren der jeweils ursächlichen Variablenbelegungen in „NoGood“-Menge
 - Zurückverfolgen der Gründe für die Belegungen
- 1UIP-Lernen: Beende die Rückverfolgung, sobald das NoGood nur noch eine einzige Variable v auf höchster Ebene (Konfliktebene) enthält.
- Die gelernte Klausel ist die Disjunktion der negierten Variablen im NoGood (die Konjunktion dieser Variablenbelegungen soll nie mehr auftreten).
- Eine 1UIP-Klausel ermöglicht eine weitere Propagation auf der Backtrack-Ebene vor dem Konflikt und erzwingt so eine andere Belegung der Variable v .

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)

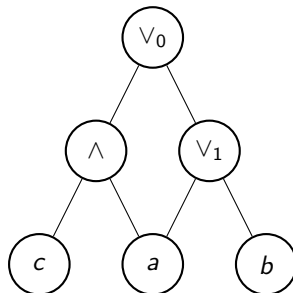
Einem Kind eines \wedge -Knotens mit Wert **F** wurde mit Grund „Parent“ der Wert **F** zugewiesen. Dann sind der \wedge -Knoten und alle Geschwisterknoten des betrachteten Kindes (haben Wert **T**) Ursachen für die Belegung.

ResolveAndLearn: Konstruktion von NoGood und gelernter Klausel

- Initialisiere das NoGood mit den Konfliktgründen.
- Solange es mehr als eine Variable auf der höchsten Entscheidungsebene gibt:
 - 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 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: die gelernte Klausel ermöglicht nun eine weitere Werte-Propagation, die den vormaligen Zuweisungswert an die UIP-Variable umkehrt.

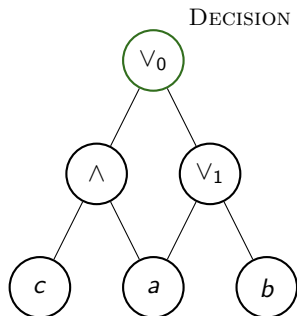
Rücksetzen des Zuweisungsstacks und Aufzeichnen des NoGood erfolgen in der Praxis gleichzeitig, Ebene für Ebene!

Beispiel: Konfliktlernen



Lv	Var	Val	Grund	Ursache
----	-----	-----	-------	---------

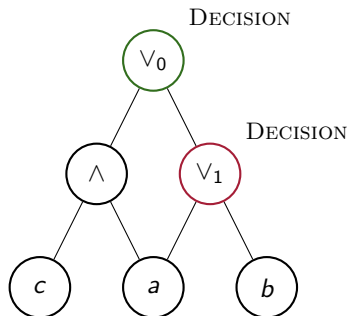
Beispiel: Konfliktlernen



Lv	Var	Val	Grund	Ursache
1	V_0	T	DECISION	

- Weise Wurzel **T** zu
- Zuweisungsstack: $V_0 = \mathbf{T@1}$

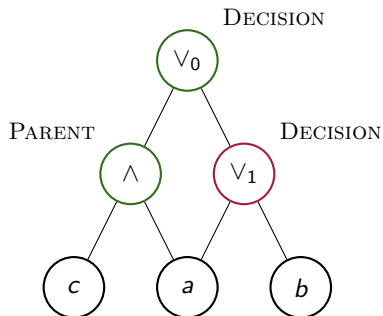
Beispiel: Konfliktlernen



Lv	Var	Val	Grund	Ursache
1	V_0	T	DECISION	
2	V_1	F	DECISION	

- Entscheidung: $v(V_1) = \mathbf{F}$
- Zuweisungsstack: $V_0 = \mathbf{T@1}$, $V_1 = \mathbf{F@2}$

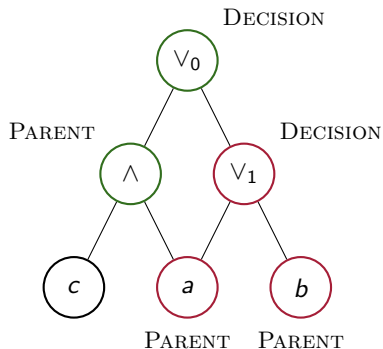
Beispiel: Konfliktlernen



Lv	Var	Val	Grund	Ursache
1	V_0	T	DECISION	
2	V_1	F	DECISION	
	\wedge	T	PARENT	$\{V_0 = \mathbf{T}, V_1 = \mathbf{F}\}$

- Implikation: $v(\wedge) = \mathbf{T}$
- Zuweisungsstack: $V_0 = \mathbf{T}@1$, $V_1 = \mathbf{F}@2$, $\wedge = \mathbf{T}@2$

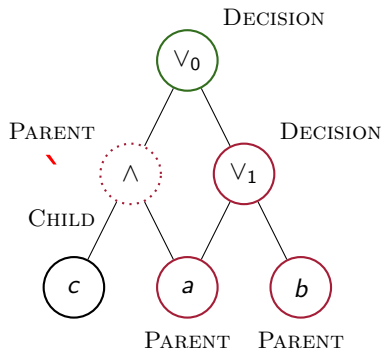
Beispiel: Konfliktlernen



Lv	Var	Val	Grund	Ursache
1	V_0	T	DECISION	
2	V_1	F	DECISION	
	\wedge	T	PARENT	$\{V_0 = \mathbf{T}, V_1 = \mathbf{F}\}$
	a	F	PARENT	$\{V_1 = \mathbf{F}\}$
	b	F	PARENT	$\{V_1 = \mathbf{F}\}$

- Implikation: $v(a)=\mathbf{F}$, $v(b)=\mathbf{F}$
- Zuweisungsstack: $V_0 = \mathbf{T}@1$, $V_1 = \mathbf{F}@2$, $\wedge = \mathbf{T}@2$, $a = \mathbf{F}@2$, $b = \mathbf{F}@2$

Beispiel: Konfliktlernen

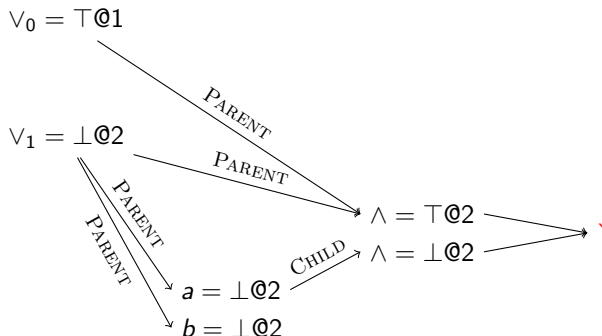


Lv	Var	Val	Grund	Ursache
1	V_0	T	DECISION	
2	V_1	F	DECISION	
	\wedge	T	PARENT	$\{V_0 = \mathbf{T}, V_1 = \mathbf{F}\}$
	a	F	PARENT	$\{V_1 = \mathbf{F}\}$
	b	F	PARENT	$\{V_1 = \mathbf{F}\}$
	\wedge	F	CHILD	$\{a = \mathbf{F}\}$

- Implikation: $v(\wedge) = \mathbf{F}$, Widerspruch!
- Zuweisungsstack: $V_0 = \mathbf{T}@1$, $V_1 = \mathbf{F}@2$, $\wedge = \mathbf{T}@2$, $a = \mathbf{F}@2$, $b = \mathbf{F}@2$

Beispiel: Implikationsgraph

(Implizit vorhandener) Implikationsgraph der Konfliktsituation:



Beispiel: Aufzeichnen des NoGood, Lernen einer Klausel

- ① Notieren der Ursachen für $\wedge = \mathbf{T@2}$, $\wedge = \mathbf{F@2}$ in NoGood-Menge:

$$\underbrace{\{V_0 = \mathbf{T@1}, V_1 = \mathbf{F@2}\}}_{\text{Grund für } \wedge = \mathbf{T@2}} \cup \underbrace{\{a = \mathbf{F@2}\}}_{\text{Grund für } \wedge = \mathbf{F@2}} = \{V_0 = \mathbf{T@1}, V_1 = \mathbf{F@2}, a = \mathbf{F@2}\}$$

Beispiel: Aufzeichnen des NoGood, Lernen einer Klausel

- ① Notieren der Ursachen für $\wedge = \mathbf{T@2}$, $\wedge = \mathbf{F@2}$ in NoGood-Menge:

$$\underbrace{\{V_0 = \mathbf{T@1}, V_1 = \mathbf{F@2}\}}_{\text{Grund für } \wedge = \mathbf{T@2}} \cup \underbrace{\{a = \mathbf{F@2}\}}_{\text{Grund für } \wedge = \mathbf{F@2}} = \{V_0 = \mathbf{T@1}, V_1 = \mathbf{F@2}, a = \mathbf{F@2}\}$$

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

Beispiel: Aufzeichnen des NoGood, Lernen einer Klausel

- ① Notieren der Ursachen für $\wedge = \mathbf{T@2}$, $\wedge = \mathbf{F@2}$ in NoGood-Menge:

$$\underbrace{\{V_0 = \mathbf{T@1}, V_1 = \mathbf{F@2}\}}_{\text{Grund für } \wedge = \mathbf{T@2}} \cup \underbrace{\{a = \mathbf{F@2}\}}_{\text{Grund für } \wedge = \mathbf{F@2}} = \{V_0 = \mathbf{T@1}, V_1 = \mathbf{F@2}, a = \mathbf{F@2}\}$$

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

$$\begin{aligned} & (\{V_0 = \mathbf{T@1}, V_1 = \mathbf{F@2}, a = \mathbf{F@2}\} \setminus \{a = \mathbf{F@2}\}) \cup \{V_1 = \mathbf{F@2}\} \\ &= \{V_0 = \mathbf{T@1}, V_1 = \mathbf{F@2}\} \end{aligned}$$

\Rightarrow 1UIP NoGood gefunden (mit UIP $V_1 = \mathbf{F@2}$), lerne daraus neue Klausel

Beispiel: Aufzeichnen des NoGood, Lernen einer Klausel

- ① Notieren der Ursachen für $\wedge = \mathbf{T@2}$, $\wedge = \mathbf{F@2}$ in NoGood-Menge:

$$\underbrace{\{V_0 = \mathbf{T@1}, V_1 = \mathbf{F@2}\}}_{\text{Grund für } \wedge = \mathbf{T@2}} \cup \underbrace{\{a = \mathbf{F@2}\}}_{\text{Grund für } \wedge = \mathbf{F@2}} = \{V_0 = \mathbf{T@1}, V_1 = \mathbf{F@2}, a = \mathbf{F@2}\}$$

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

$$\begin{aligned} & (\{V_0 = \mathbf{T@1}, V_1 = \mathbf{F@2}, a = \mathbf{F@2}\} \setminus \{a = \mathbf{F@2}\}) \cup \{V_1 = \mathbf{F@2}\} \\ &= \{V_0 = \mathbf{T@1}, V_1 = \mathbf{F@2}\} \Rightarrow \text{Klausel: } ([\neg V_0] \vee [V_1]) \end{aligned}$$

\Rightarrow 1UIP NoGood gefunden (mit UIP $V_1 = \mathbf{F@2}$), lerne daraus neue Klausel

Beispiel: Aufzeichnen des NoGood, Lernen einer Klausel

- ① Notieren der Ursachen für $\wedge = \mathbf{T@2}$, $\vee = \mathbf{F@2}$ in NoGood-Menge:

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

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

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

\Rightarrow UIP NoGood gefunden (mit UIP $\vee_1 = \mathbf{F@2}$), lerne daraus neue Klausel

- ④ Backtracking-Ebene ist BE=1 (höchste Ebene in $\{\vee_0 = \mathbf{T@1}\}$). Mache $\wedge = \mathbf{T@2}$ und $\vee_1 = \mathbf{F@2}$ rückgängig, setze Level auf BE=1 herunter

Beispiel: Aufzeichnen des NoGood, Lernen einer Klausel

- ① Notieren der Ursachen für $\wedge = \mathbf{T@2}$, $\wedge = \mathbf{F@2}$ in NoGood-Menge:

$$\underbrace{\{V_0 = \mathbf{T@1}, V_1 = \mathbf{F@2}\}}_{\text{Grund für } \wedge = \mathbf{T@2}} \cup \underbrace{\{a = \mathbf{F@2}\}}_{\text{Grund für } \wedge = \mathbf{F@2}} = \{V_0 = \mathbf{T@1}, V_1 = \mathbf{F@2}, a = \mathbf{F@2}\}$$

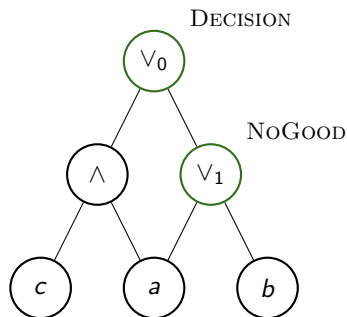
- ② Mache $b = \mathbf{F@2}$ rückgängig, b ist nicht im NoGood notiert, also nicht am Konflikt beteiligt
- ③ Mache $a = \mathbf{F@2}$ rückgängig: a ist am Konflikt beteiligt, ersetze a durch seine Ursache:

$$\begin{aligned} & (\{V_0 = \mathbf{T@1}, V_1 = \mathbf{F@2}, a = \mathbf{F@2}\} \setminus \{a = \mathbf{F@2}\}) \cup \{V_1 = \mathbf{F@2}\} \\ &= \{V_0 = \mathbf{T@1}, V_1 = \mathbf{F@2}\} \Rightarrow \text{Klausel: } ([\neg V_0] \vee [V_1]) \end{aligned}$$

\Rightarrow UIP NoGood gefunden (mit UIP $V_1 = \mathbf{F@2}$), lerne daraus neue Klausel

- ④ Backtracking-Ebene ist $BE=1$ (höchste Ebene in $\{V_0 = \mathbf{T@1}\}$). Mache $\wedge = \mathbf{T@2}$ und $V_1 = \mathbf{F@2}$ rückgängig, setze Level auf $BE=1$ herunter
- ⑤ Es ist $V_0 = \mathbf{T@1}$, also ist gelernte Klausel Unit ($[\mathbf{F}] \vee [V_1]$) auf $BE=1$. Zustand ist OK, kehre zurück und propagiere $V_1 = \mathbf{T@1}$ mit Grund NoGood.

Beispiel: Konfliktauflösung

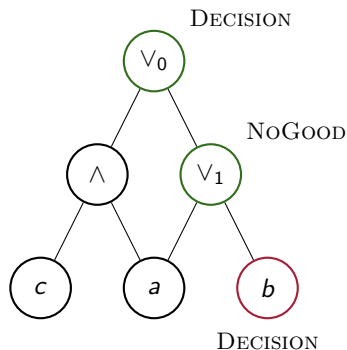


NoGood: $([V_0 = \mathbf{F}] \vee [V_1 = \mathbf{T}])$

Lv	Var	Val	Grund	Ursache
1	V_0	\mathbf{T}	DECISION	
	V_1	\mathbf{T}	NoGood	$\{V_0 = \mathbf{T}\}$

- Implikation: $v(V_1) = \mathbf{T}$
- Zuweisungsstack: $V_0 = \mathbf{T}@1, V_1 = \mathbf{T}@1$

Beispiel: Konfliktauflösung

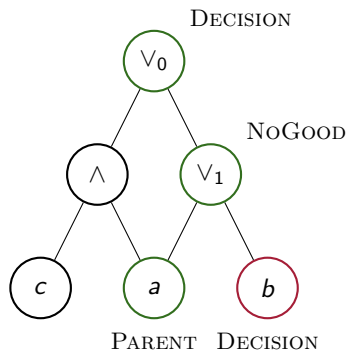


NoGood: $((V_0 = \mathbf{F}) \vee (V_1 = \mathbf{T}))$

Lv	Var	Val	Grund	Ursache
1	V_0	\mathbf{T}	DECISION	
	V_1	\mathbf{T}	NoGOOD	$\{V_0 = \mathbf{T}\}$
2	b	\mathbf{F}	DECISION	

- Entscheidung: $v(b) = \mathbf{F}$
- Zuweisungsstack: $V_0 = \mathbf{T}@1, V_1 = \mathbf{T}@1, b = \mathbf{F}@2$

Beispiel: Konfliktauflösung

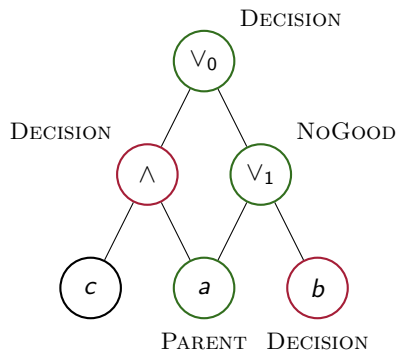


NoGood: $([V_0 = \mathbf{F}] \vee [V_1 = \mathbf{T}])$

Lv	Var	Val	Grund	Ursache
1	V_0	\mathbf{T}	DECISION	
	V_1	\mathbf{T}	NoGOOD	$\{V_0 = \mathbf{T}\}$
2	b	\mathbf{F}	DECISION	
	a	\mathbf{T}	PARENT	$\{V_1 = \mathbf{T}, b = \mathbf{F}\}$

- Implikation: $v(a) = \mathbf{T}$
- Zuweisungsstack: $V_0 = \mathbf{T}@1, V_1 = \mathbf{T}@1, b = \mathbf{F}@2, a = \mathbf{T}@2$

Beispiel: Konfliktauflösung

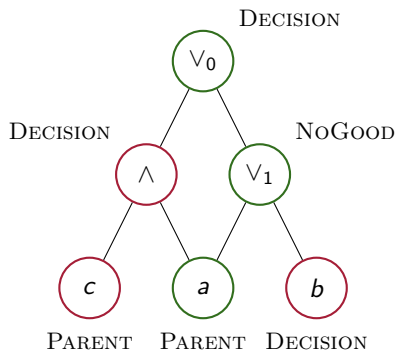


NoGood: $([V_0 = \mathbf{F}] \vee [V_1 = \mathbf{T}])$

Lv	Var	Val	Grund	Ursache
1	V_0	\mathbf{T}	DECISION	
	V_1	\mathbf{T}	NOGOOD	$\{V_0 = \mathbf{T}\}$
2	b	\mathbf{F}	DECISION	
	a	\mathbf{T}	PARENT	$\{V_1 = \mathbf{T}, b = \mathbf{F}\}$
3	\wedge	\mathbf{F}	DECISION	

- Entscheidung: $v(\wedge) = \mathbf{F}$
- Zuweisungsstack: $V_0 = \mathbf{T}@1, V_1 = \mathbf{T}@1, b = \mathbf{F}@2, a = \mathbf{T}@2, \wedge = \mathbf{F}@3$

Beispiel: Konfliktauflösung



NoGood: $([V_0 = \mathbf{F}] \vee [V_1 = \mathbf{T}])$

Lv	Var	Val	Grund	Ursache
1	V_0	\mathbf{T}	DECISION	
	V_1	\mathbf{T}	NoGOOD	$\{V_0 = \mathbf{T}\}$
2	b	\mathbf{F}	DECISION	
	a	\mathbf{T}	PARENT	$\{V_1 = \mathbf{T}, b = \mathbf{F}\}$
3	\wedge	\mathbf{F}	DECISION	
	c	\mathbf{F}	PARENT	$\{\wedge = \mathbf{F}, a = \mathbf{T}\}$

- Implikation: $v(c) = \mathbf{F}$
- Z.stack: $V_0 = \mathbf{T}@1, V_1 = \mathbf{T}@1, b = \mathbf{F}@2, a = \mathbf{T}@2, \wedge = \mathbf{F}@3, c = \mathbf{F}@3$

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 , ...) möglich
- Implementierung problemspezifischer Heuristiken

Nachteile:

- Implementierung komplizierter
- Kompliziertere Datenstruktur (DAG)