SAT-Solving und Anwendungen

CDCL SAT-Solving

Prof. Dr. Wolfgang Küchlin
Dipl. Inf. Christoph Zengler

Universität Tübingen

26. April 2011

Ausblick der letzten Vorlesung

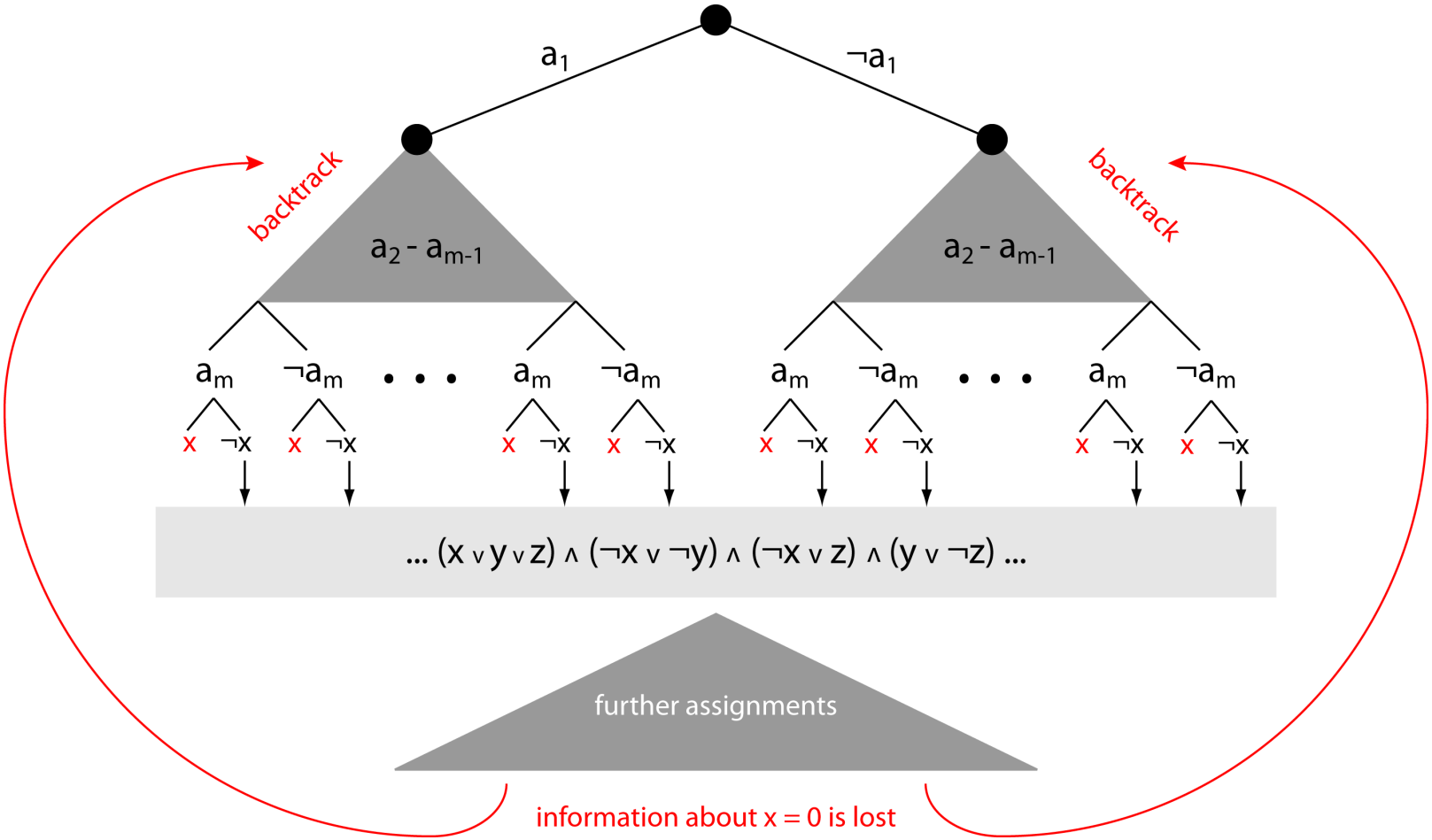
Zwei große Probleme bei DPLL:

- Vergessen von zusätzlichen Informationen beim Backtracking (Illustration auf der nächsten Folie)
  - Beobachtung 1: Springt man über eine gewisse Grenze beim Backtracking zurück, so "vergisst" man bereits erarbeitete Information (z.B. bestimmte UPs)
  - Idee 1: Hinzufügen dieser zusätzlichen Information zur Originalformel, so dass sie beim Backtracking nicht verloren geht
- Backtracking immer nur zur letzten gesetzten Variable
  - Beobachtung 2: 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 1) und **Klauselllernen** (Problem 2)

Vergessen von Informationen beim Backtracking

An einer gewissen Stelle existiert die Information, dass $x = \mathbf{F}$ 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:

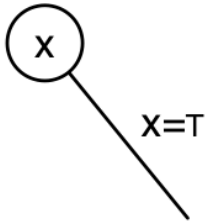
1. Speichere zu jeder Variable, ob sie durch Entscheidung oder UP belegt wurde
2. Bei UP speichere die Klausel, die die UP veranlasst hat *(Reason)*
3. Betreibe Resolution zwischen den einzelnen Reasons um eine neue Klausel zu erhalten
4. Füge diese neue Klausel der originalen Klauselmenge hinzu

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

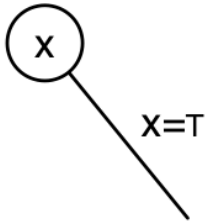
{¬u,w}
{¬u,¬m,h,¬w}
{¬u,¬f,m,¬w}
{¬f,¬g,¬h}
{y,f}
{¬f,g}
.
.
.
{¬x,a}
{¬x,b,¬a}
{¬x,¬a,¬b,c}
{¬a,¬d}
{¬b,d,¬e}
.
.
.



Level	Variable	Wert	Grund
1	x	T	decision

Aufbau der Datenstruktur

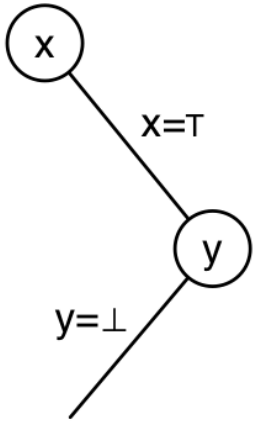
{¬u,w}
{¬u,¬m,h,¬w}
{¬u,¬f,m,¬w}
{¬f,¬g,¬h}
{y,f}
{¬f,g}
.
.
.
{¬x,a}
{¬x,b,¬a}
{¬x,¬a,¬b,c}
{¬a,¬d}
{¬b,d,¬e}
.
.
.



Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}

Aufbau der Datenstruktur

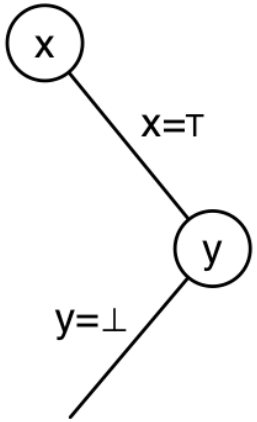
{¬u,w}
{¬u,¬m,h,¬w}
{¬u,¬f,m,¬w}
{¬f,¬g,¬h}
{y,f}
{¬f,g}
.
.
.
{¬x,a}
{¬x,b,¬a}
{¬x,¬a,¬b,c}
{¬a,¬d}
{¬b,d,¬e}
.
.
.



Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision

Aufbau der Datenstruktur

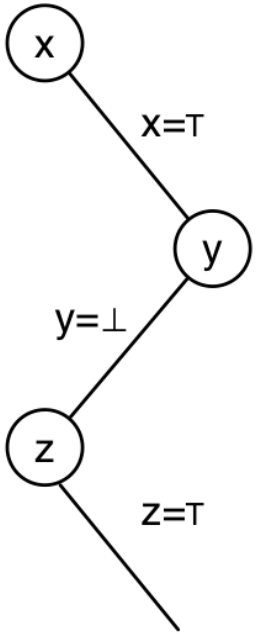
{¬u,w}
{¬u,¬m,h,¬w}
{¬u,¬f,m,¬w}
{¬f,¬g,¬h}
{y,f}
{¬f,g}
.
.
.
{¬x,a}
{¬x,b,¬a}
{¬x,¬a,¬b,c}
{¬a,¬d}
{¬b,d,¬e}
.
.
.



Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}

Aufbau der Datenstruktur

{¬u,w}
{¬u,¬m,h,¬w}
{¬u,¬f,m,¬w}
{¬f,¬g,¬h}
{y,f}
{¬f,g}
.
.
.
{¬x,a}
{¬x,b,¬a}
{¬x,¬a,¬b,c}
{¬a,¬d}
{¬b,d,¬e}
.
.
.

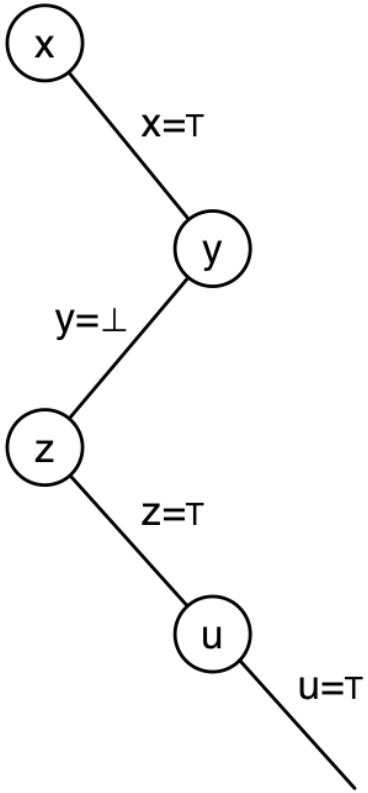


Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	⊤	decision

z kommt in den gegebenen Klauseln nicht vor, kann jedoch in ... vorkommen.

Aufbau der Datenstruktur

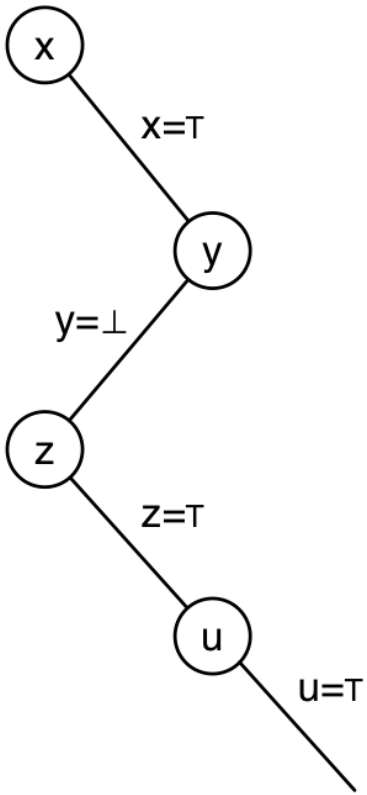
{¬u,w}
{¬u,¬m,h,¬w}
{¬u,¬f,m,¬w}
{¬f,¬g,¬h}
{y,f}
{¬f,g}
.
.
.
{¬x,a}
{¬x,b,¬a}
{¬x,¬a,¬b,c}
{¬a,¬d}
{¬b,d,¬e}
.
.
.



Level	Variable	Wert	Grund
1	x	T	decision
	a	T	{¬x,a}
	b	T	{¬x,b,¬a}
	c	T	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	T	{y,f}
	g	T	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	T	decision
4	u	T	decision

Aufbau der Datenstruktur

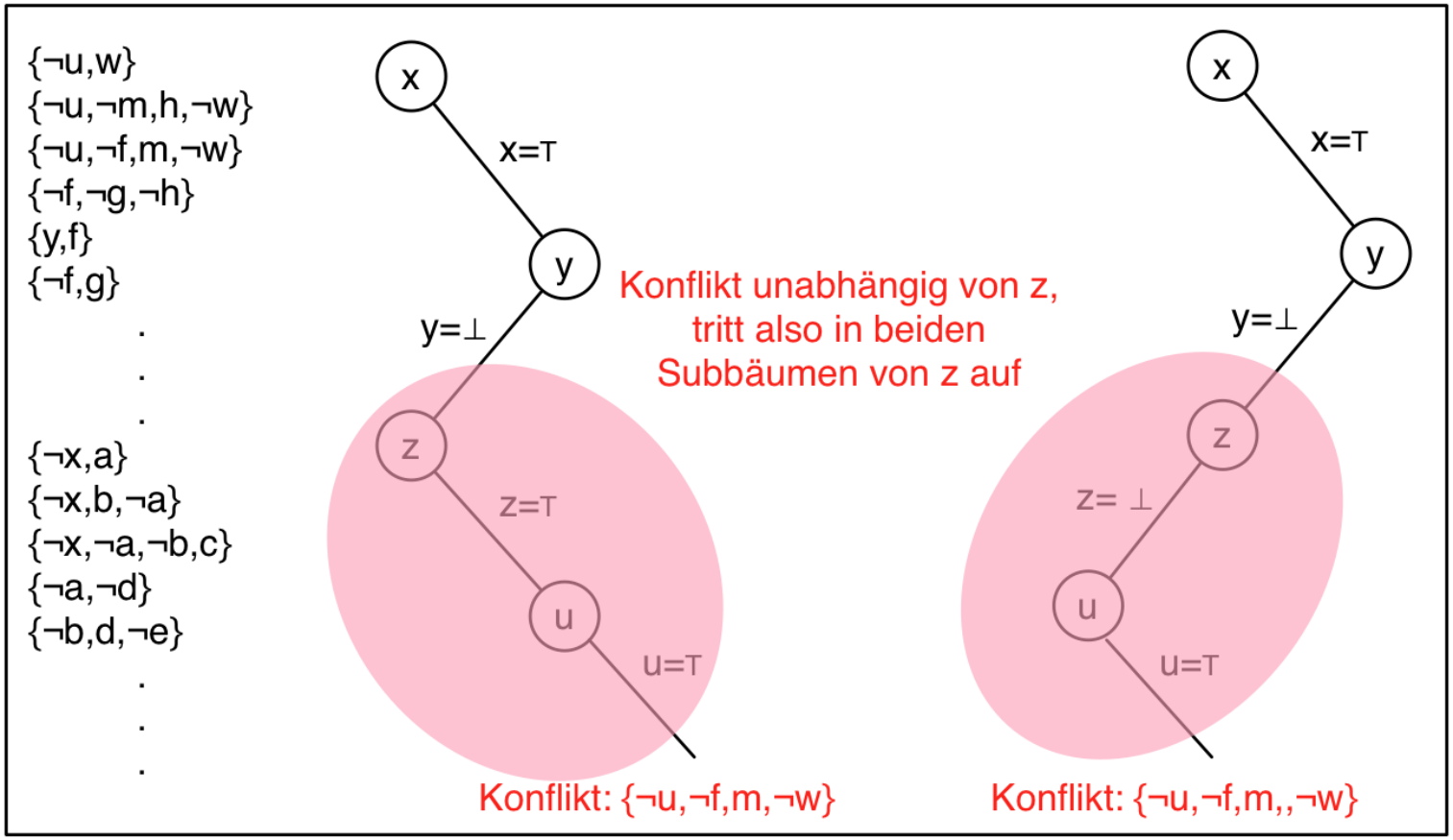
{¬u,w}
{¬u,¬m,h,¬w}
{¬u,¬f,m,¬w}
{¬f,¬g,¬h}
{y,f}
{¬f,g}
.
.
.
{¬x,a}
{¬x,b,¬a}
{¬x,¬a,¬b,c}
{¬a,¬d}
{¬b,d,¬e}
.
.
.



Konflikt: {¬u,¬f,m,¬w}

Level	Variable	Wert	Grund
1	x	T	decision
	a	T	{¬x,a}
	b	T	{¬x,b,¬a}
	c	T	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	T	{y,f}
	g	T	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	T	decision
4	u	T	decision
	w	T	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}

Warum Lernen?



Problem: Bei Backtracking über z hinaus gehen die Informationen über u verloren

Lösung: Hinzufügen einer zusätzlichen Klausel um diese Information zu "lernen"

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:

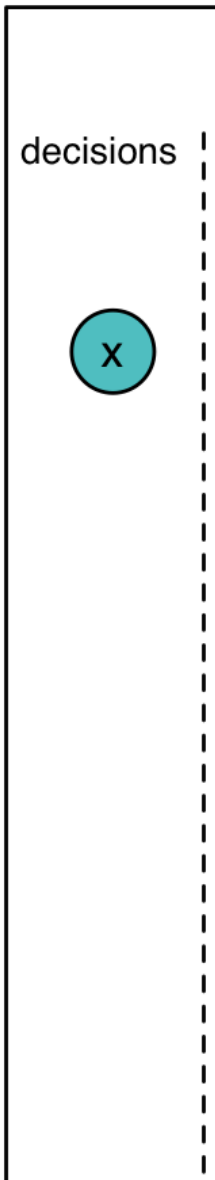
- Resolution (schon bekannt)
- Implikationsgraph
  - Knoten sind Variablenbelegungen (x bedeutet, $x \mapsto \top$, $\neg x$ bedeutet $x \mapsto \bot$)
  - Kante von A nach B bedeutet, dass A ein Grund war, dass B belegt wurde
  - Decision Variables haben keine eingehenden Kanten
  - War der Grund für die Implikation einer Variable y eine Klausel $\{y, x_1, ..., x_n\}$, so hat der Knoten (y) n eingehende Kanten

Aufbau des Implikationsgraphen

decisions

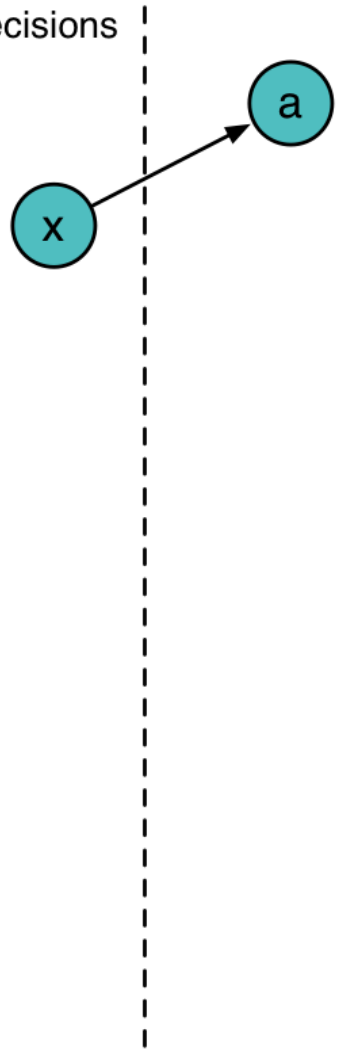
Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	⊤	decision
4	u	⊤	decision
	w	⊤	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}
	m	⊤	{¬u,¬f,m,¬w}

Aufbau des Implikationsgraphen



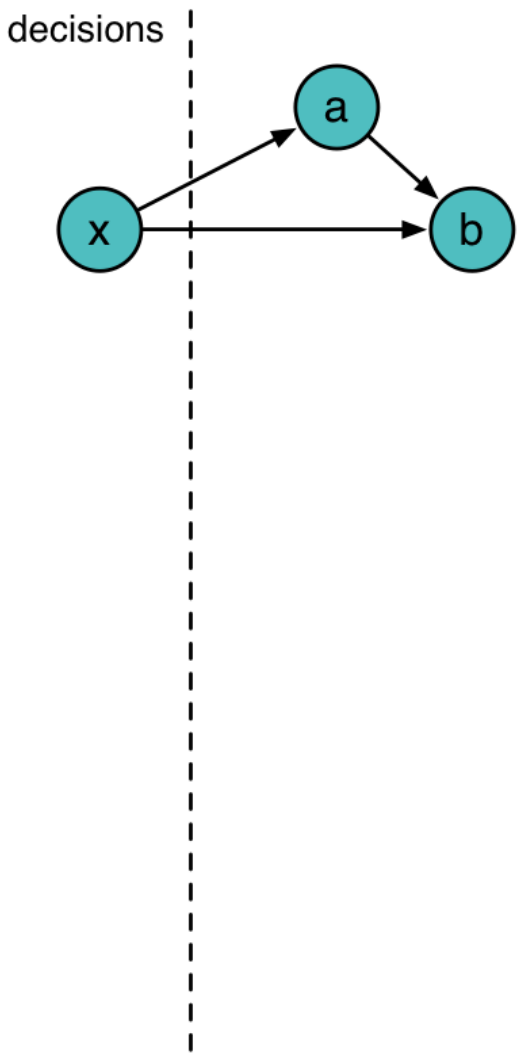
Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	⊤	decision
4	u	⊤	decision
	w	⊤	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}
	m	⊤	{¬u,¬f,m,¬w}

Aufbau des Implikationsgraphen



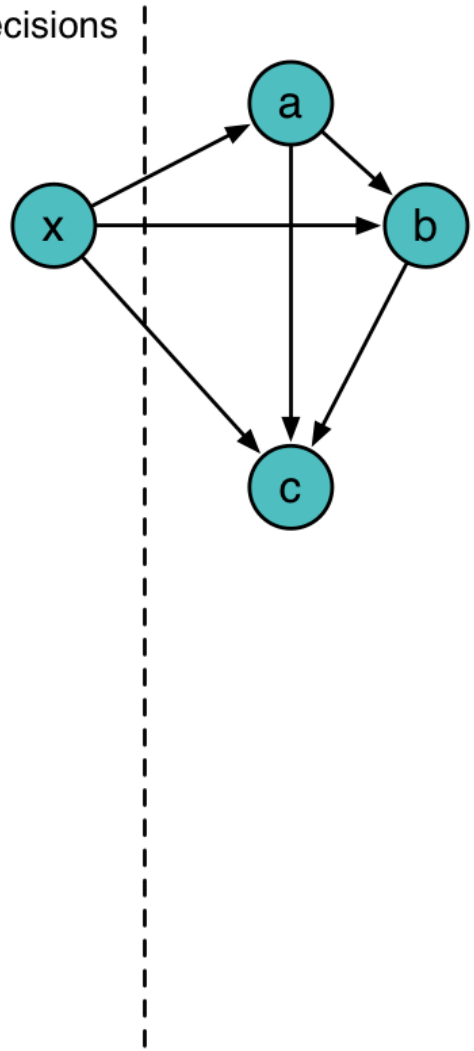
Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	⊤	decision
4	u	⊤	decision
	w	⊤	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}
	m	⊤	{¬u,¬f,m,¬w}

Aufbau des Implikationsgraphen



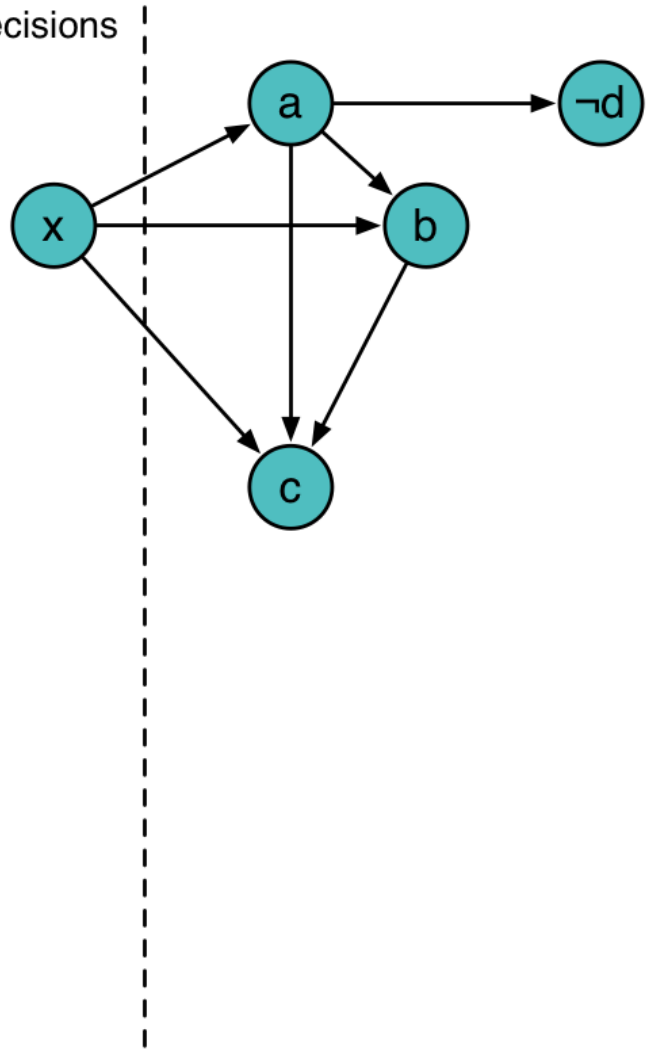
Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	⊤	decision
4	u	⊤	decision
	w	⊤	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}
	m	⊤	{¬u,¬f,m,¬w}

Aufbau des Implikationsgraphen



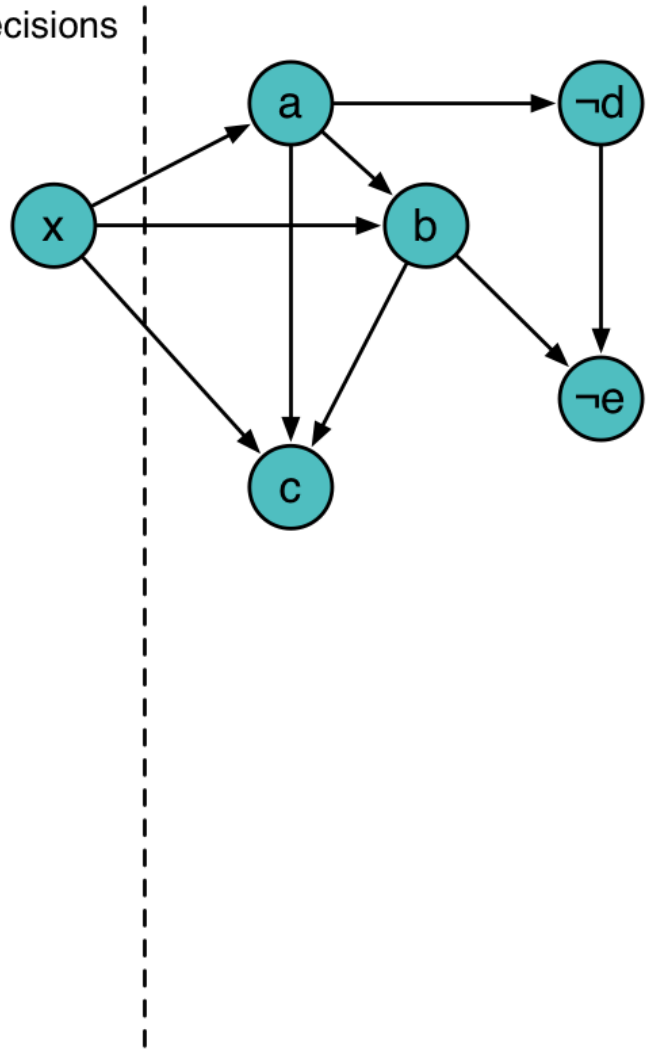
Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	⊤	decision
4	u	⊤	decision
	w	⊤	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}
	m	⊤	{¬u,¬f,m,¬w}

Aufbau des Implikationsgraphen



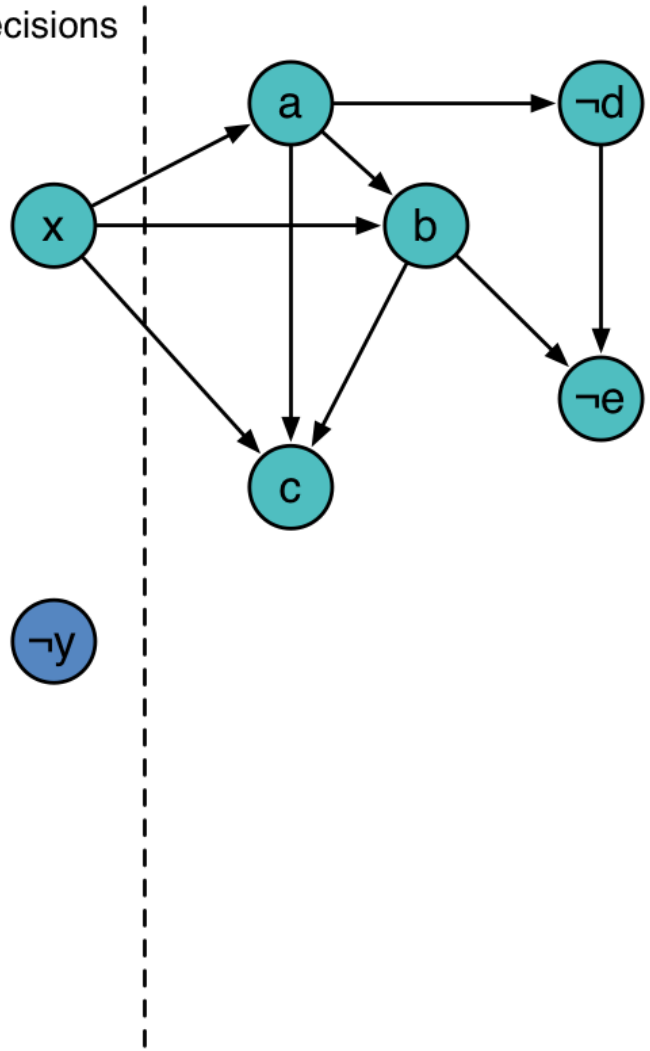
Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	⊤	decision
4	u	⊤	decision
	w	⊤	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}
	m	⊤	{¬u,¬f,m,¬w}

Aufbau des Implikationsgraphen



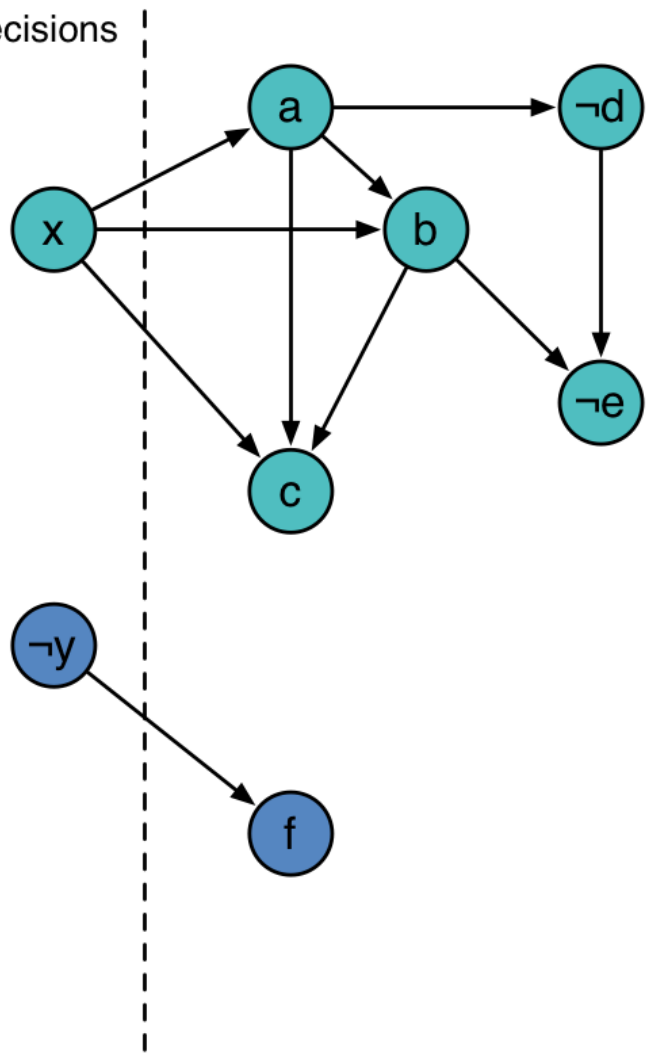
Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	⊤	decision
4	u	⊤	decision
	w	⊤	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}
	m	⊤	{¬u,¬f,m,¬w}

Aufbau des Implikationsgraphen



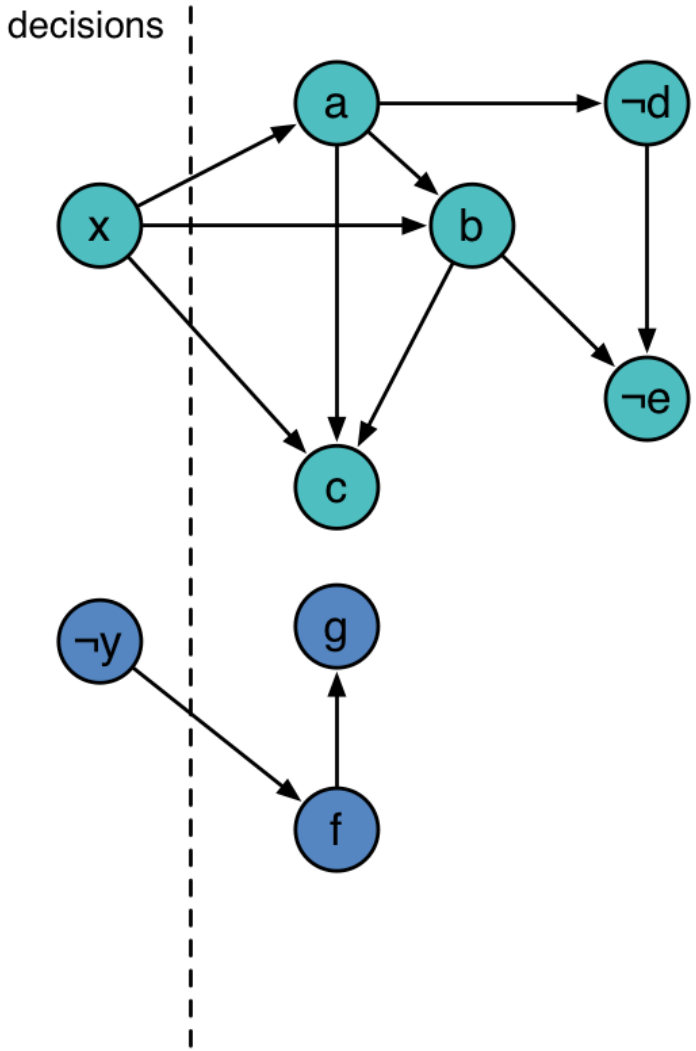
Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	⊤	decision
4	u	⊤	decision
	w	⊤	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}
	m	⊤	{¬u,¬f,m,¬w}

Aufbau des Implikationsgraphen



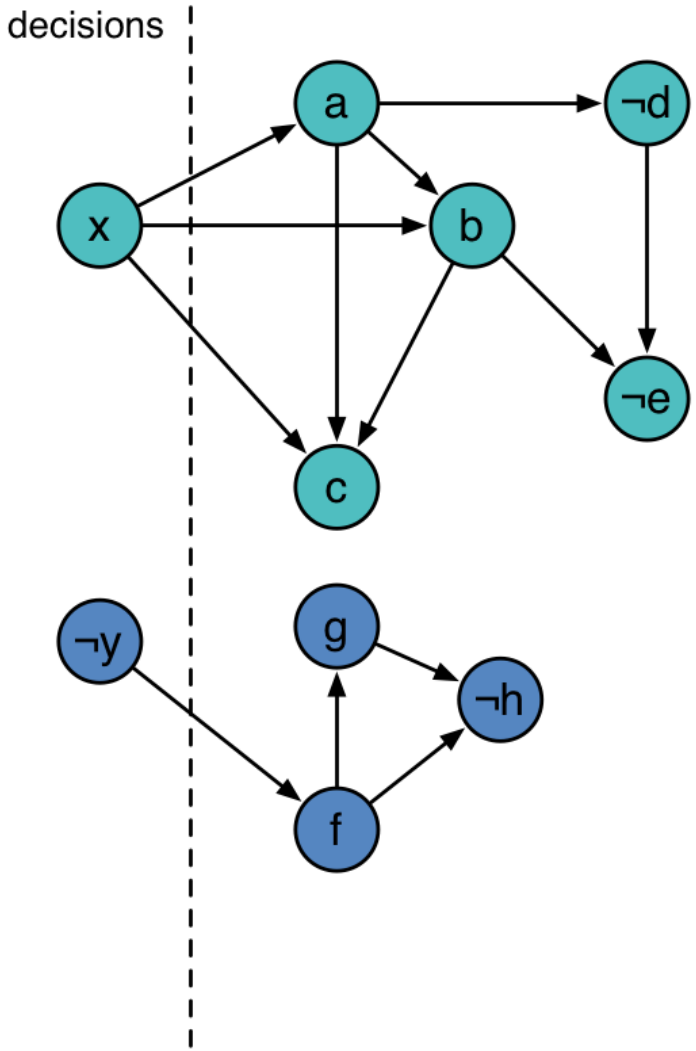
Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	⊤	decision
4	u	⊤	decision
	w	⊤	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}
	m	⊤	{¬u,¬f,m,¬w}

Aufbau des Implikationsgraphen



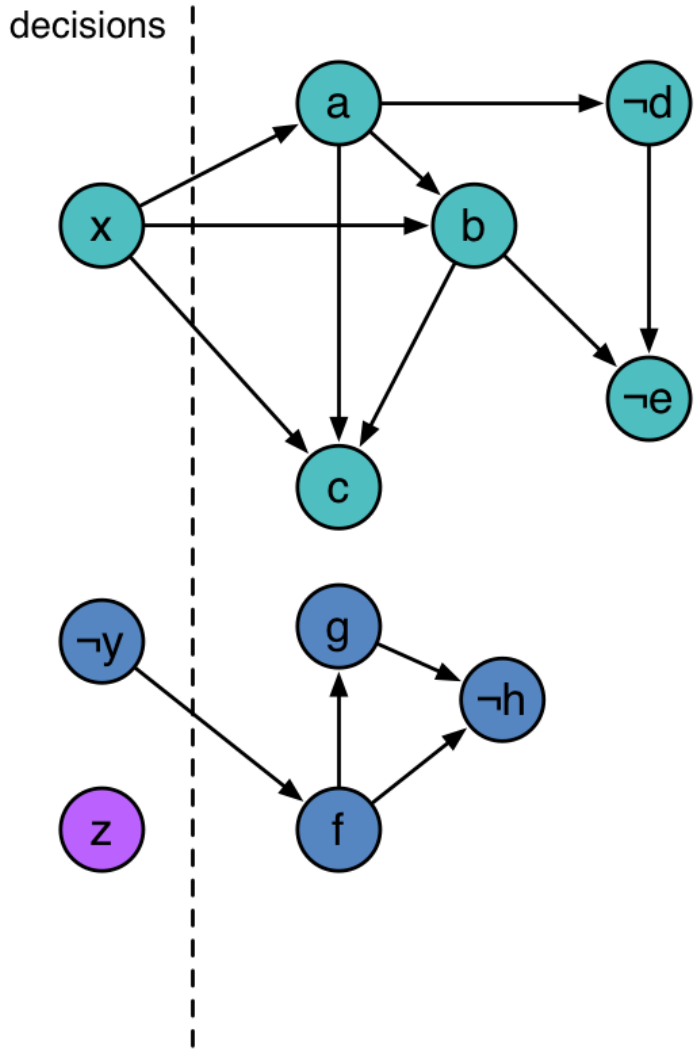
Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	⊤	decision
4	u	⊤	decision
	w	⊤	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}
	m	⊤	{¬u,¬f,m,¬w}

Aufbau des Implikationsgraphen



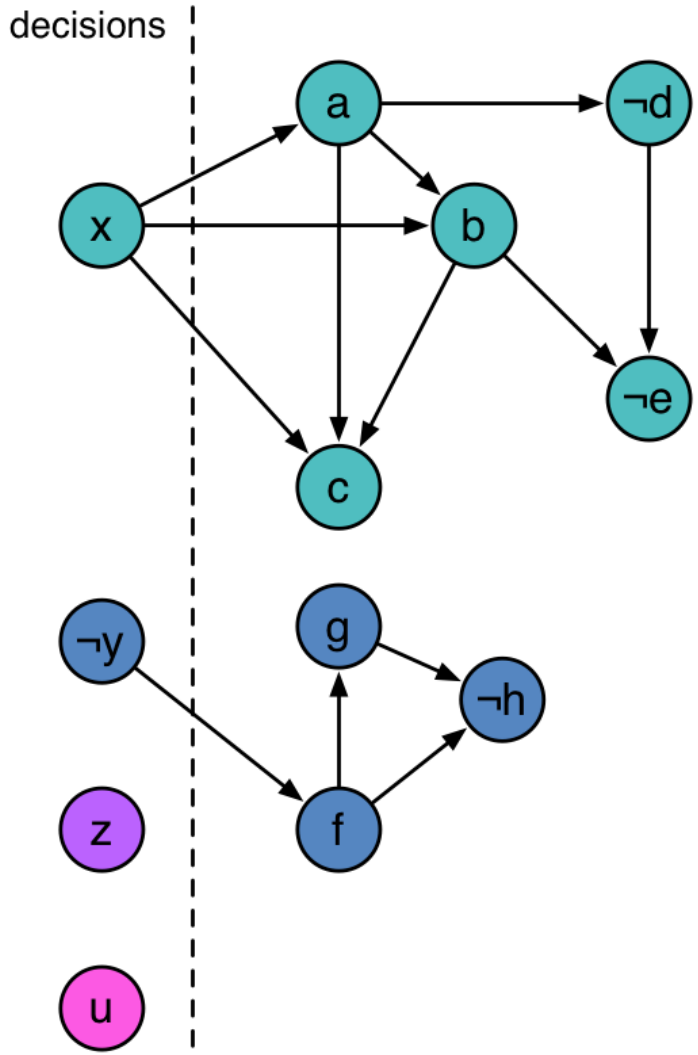
Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	⊤	decision
4	u	⊤	decision
	w	⊤	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}
	m	⊤	{¬u,¬f,m,¬w}

Aufbau des Implikationsgraphen



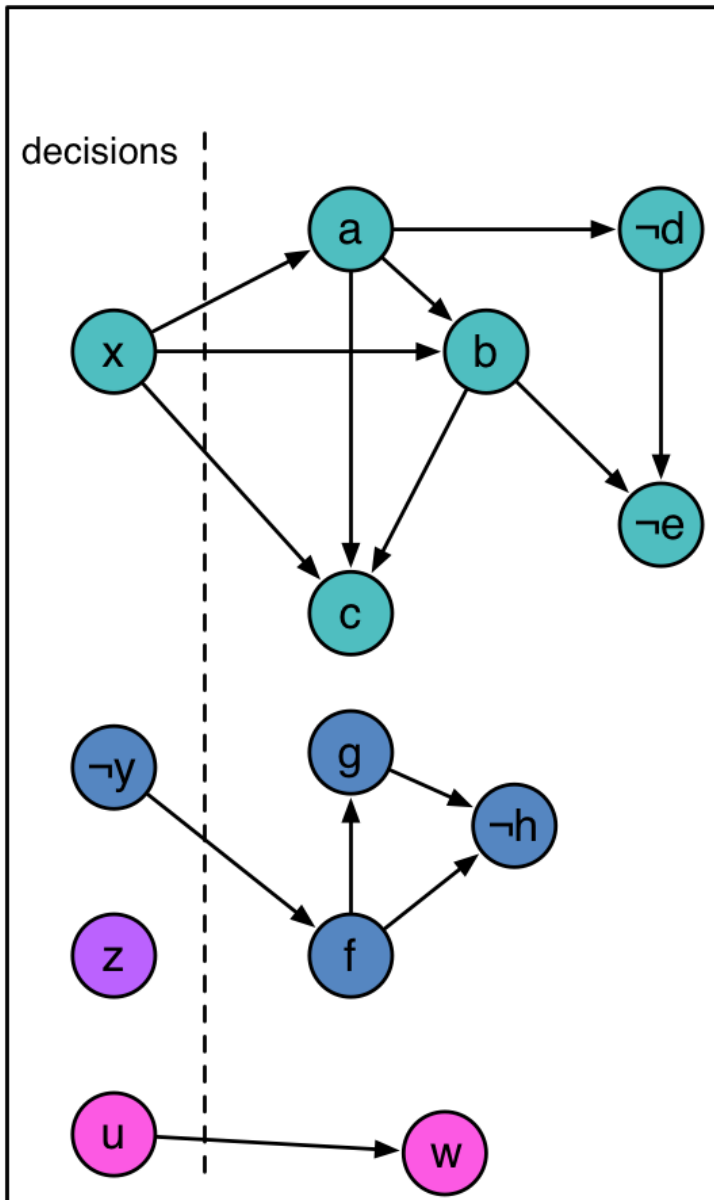
Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	⊤	decision
4	u	⊤	decision
	w	⊤	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}
	m	⊤	{¬u,¬f,m,¬w}

Aufbau des Implikationsgraphen



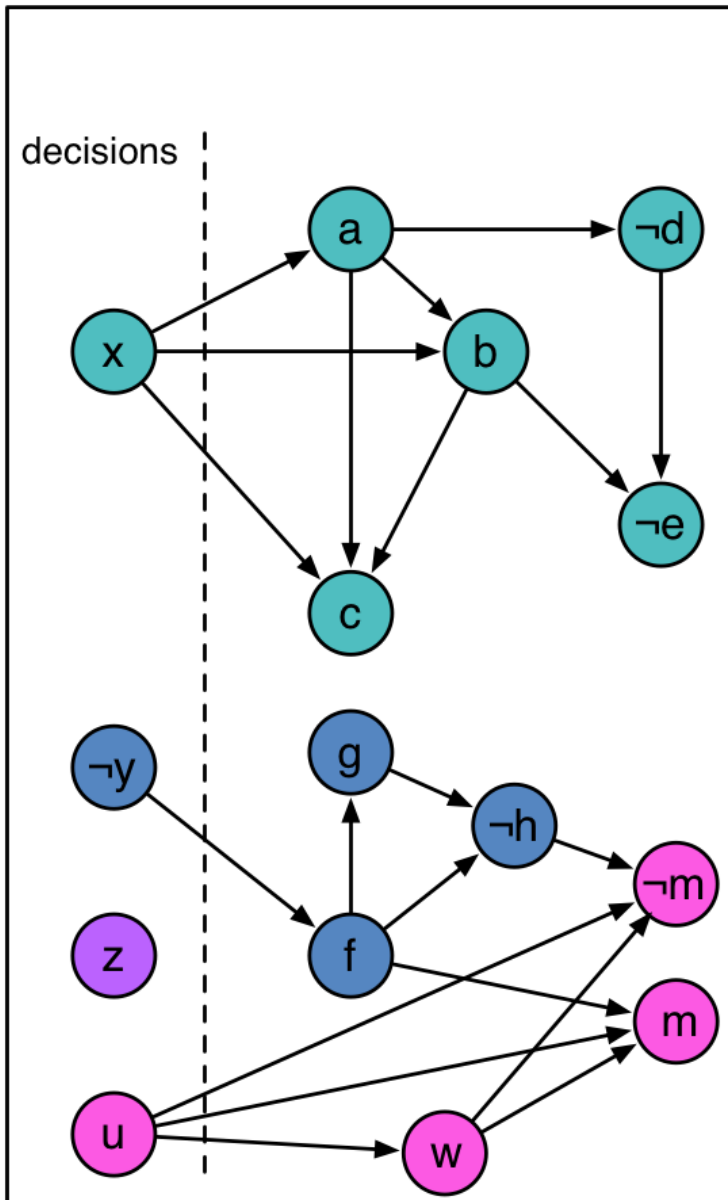
Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	⊤	decision
4	u	⊤	decision
	w	⊤	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}
	m	⊤	{¬u,¬f,m,¬w}

Aufbau des Implikationsgraphen



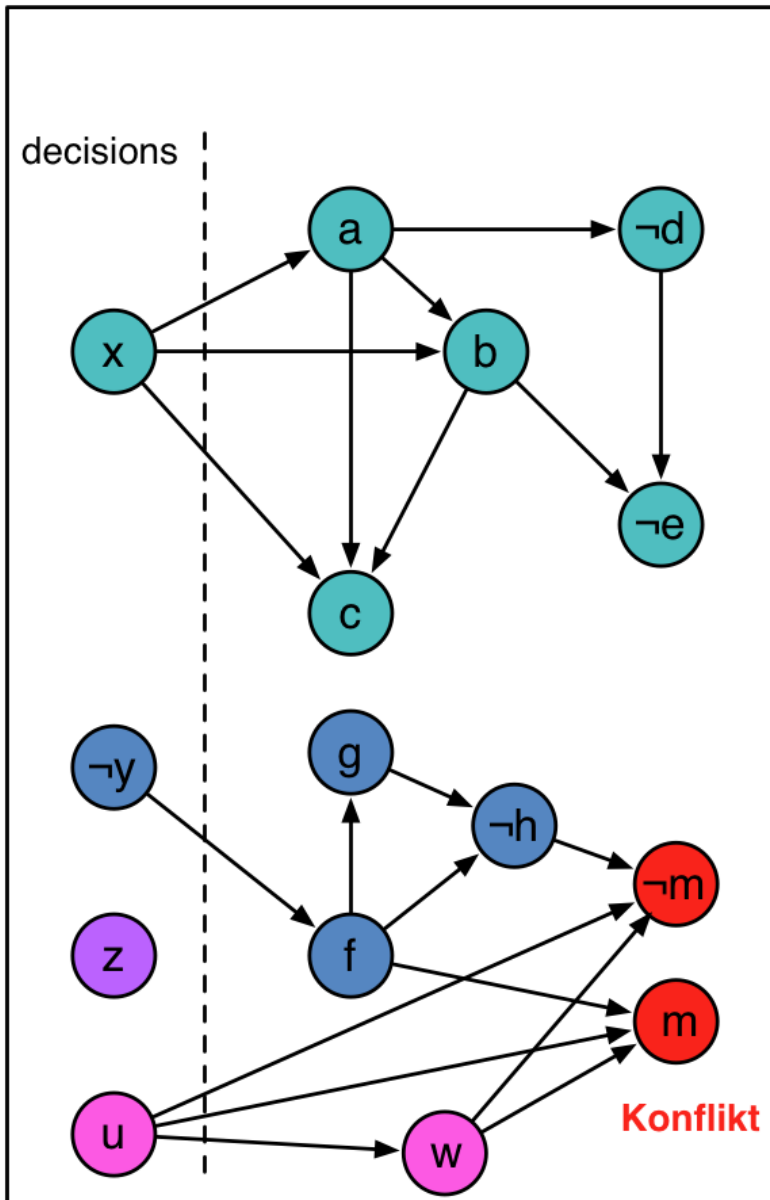
Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	⊤	decision
4	u	⊤	decision
	w	⊤	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}
	m	⊤	{¬u,¬f,m,¬w}

Aufbau des Implikationsgraphen



Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	⊤	decision
4	u	⊤	decision
	w	⊤	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}
	m	⊤	{¬u,¬f,m,¬w}

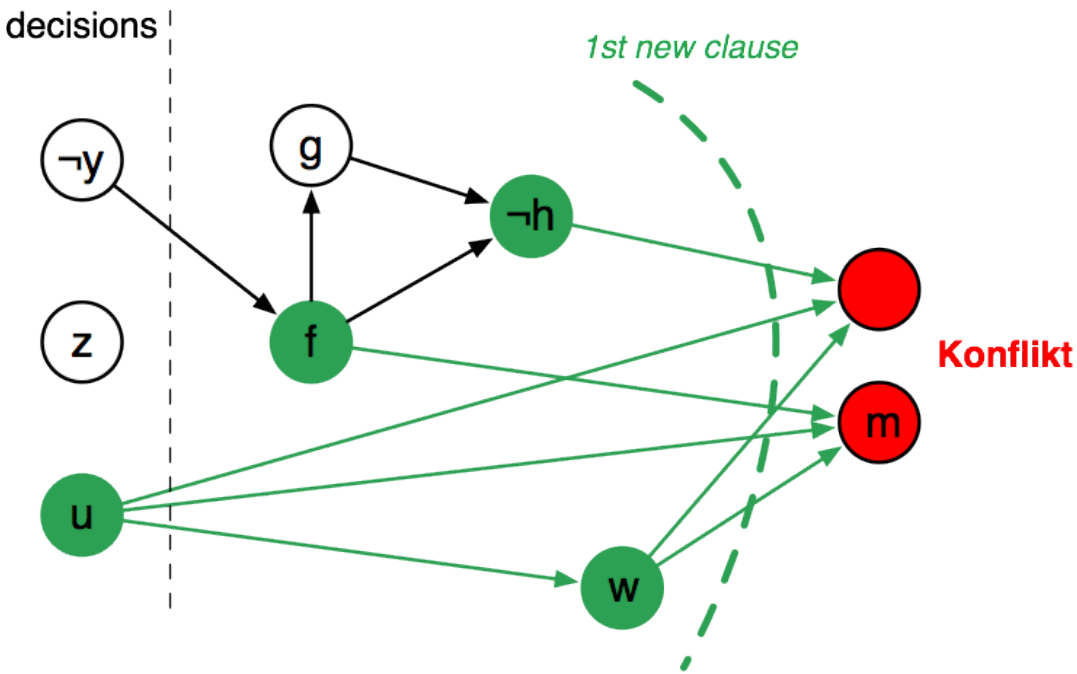
Aufbau des Implikationsgraphen



Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	⊤	decision
4	u	⊤	decision
	w	⊤	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}
	m	⊤	{¬u,¬f,m,¬w}

Und welche Klausel lernen wir jetzt?

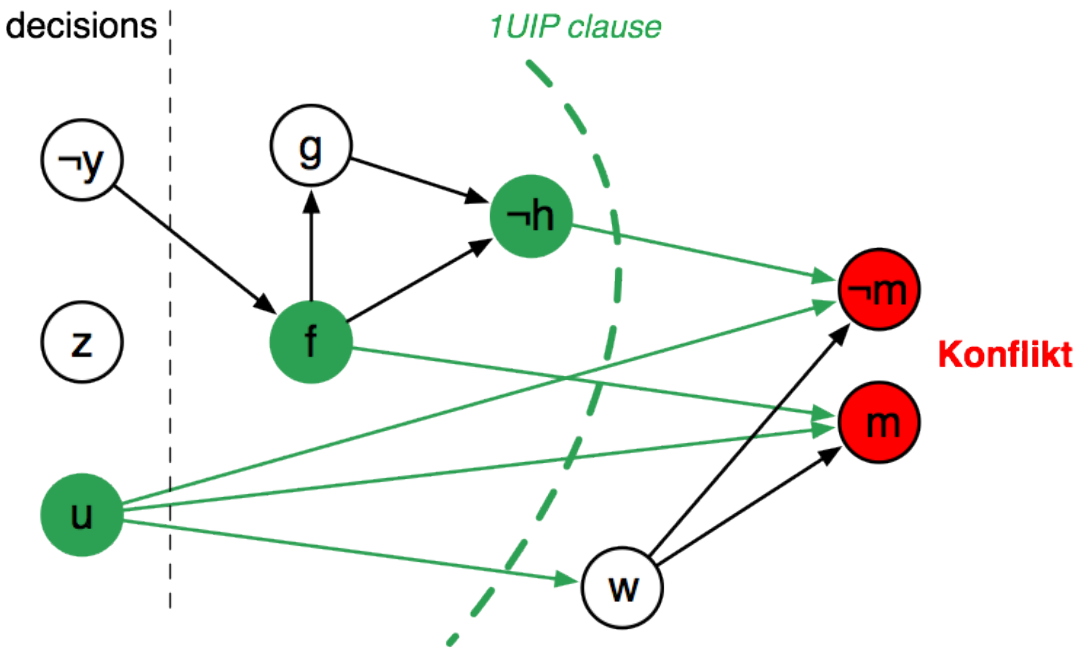
- Jeder Schnitt im Implikationsgraphen, der Entscheidungsvariablen von Konfliktvariablen trennt, ist ein gültiger *Cut*.
- Bilde eine neue Klausel aus allen Variablen, die eine ausgehende Kante durch den Cut besitzen.



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?

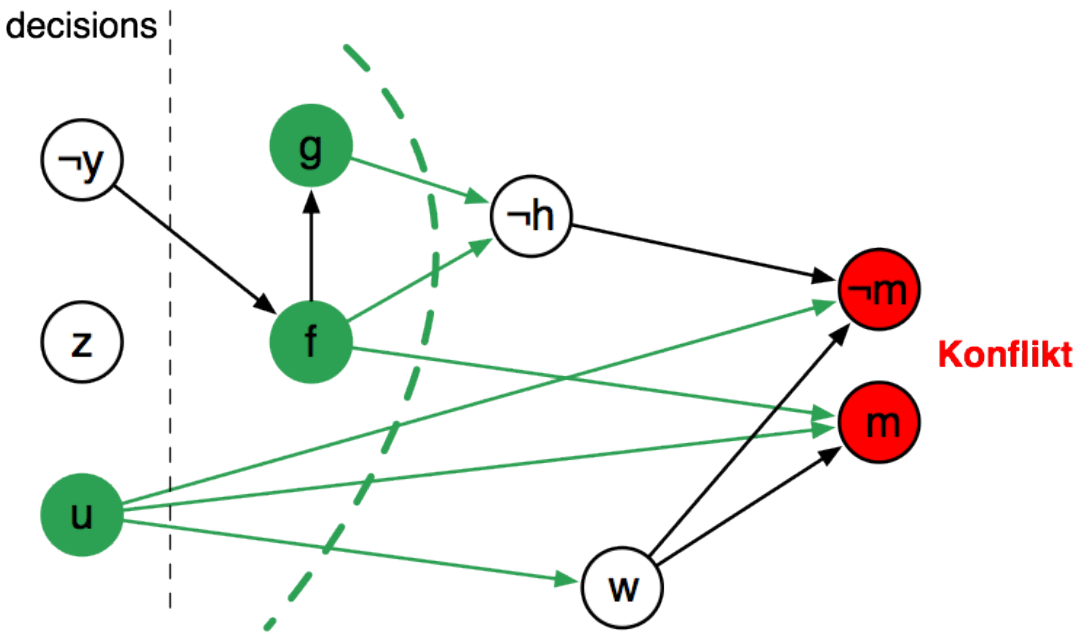
- Jeder Schnitt im Implikationsgraphen, der Entscheidungsvariablen von Konfliktvariablen trennt, ist ein gültiger *Cut*.
- Bilde eine neue Klausel aus allen Variablen, die eine ausgehende Kante durch den Cut besitzen.



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

Und welche Klausel lernen wir jetzt?

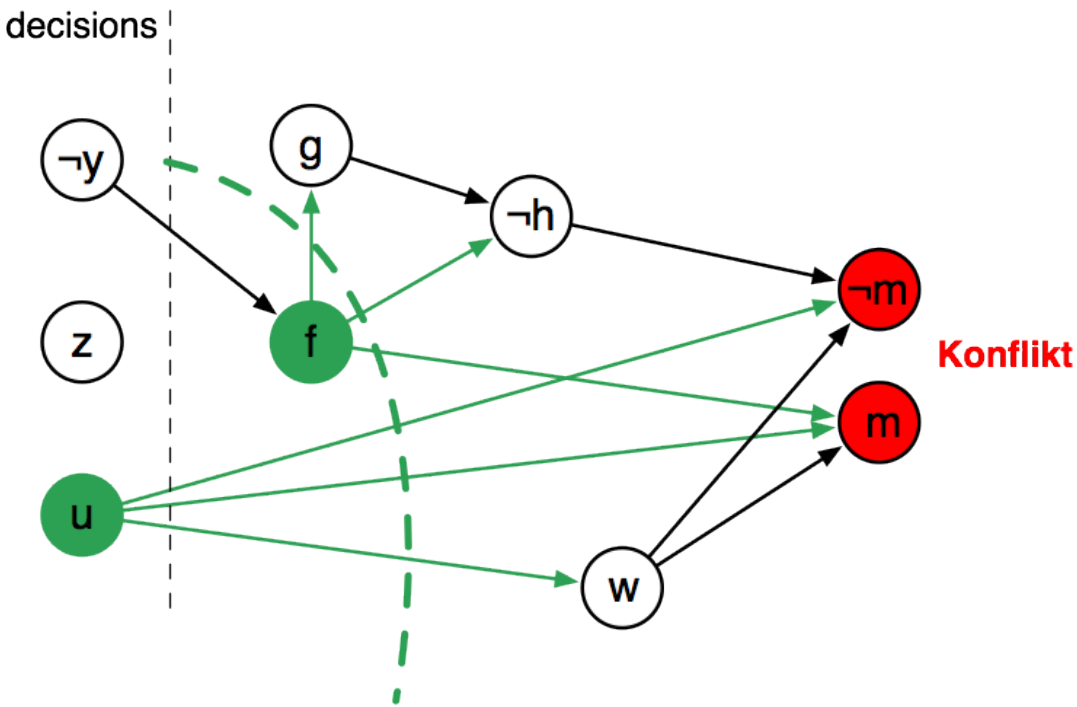
- Jeder Schnitt im Implikationsgraphen, der Entscheidungsvariablen von Konfliktvariablen trennt, ist ein gültiger *Cut*.
- Bilde eine neue Klausel aus allen Variablen, die eine ausgehende Kante durch den Cut besitzen.



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

Und welche Klausel lernen wir jetzt?

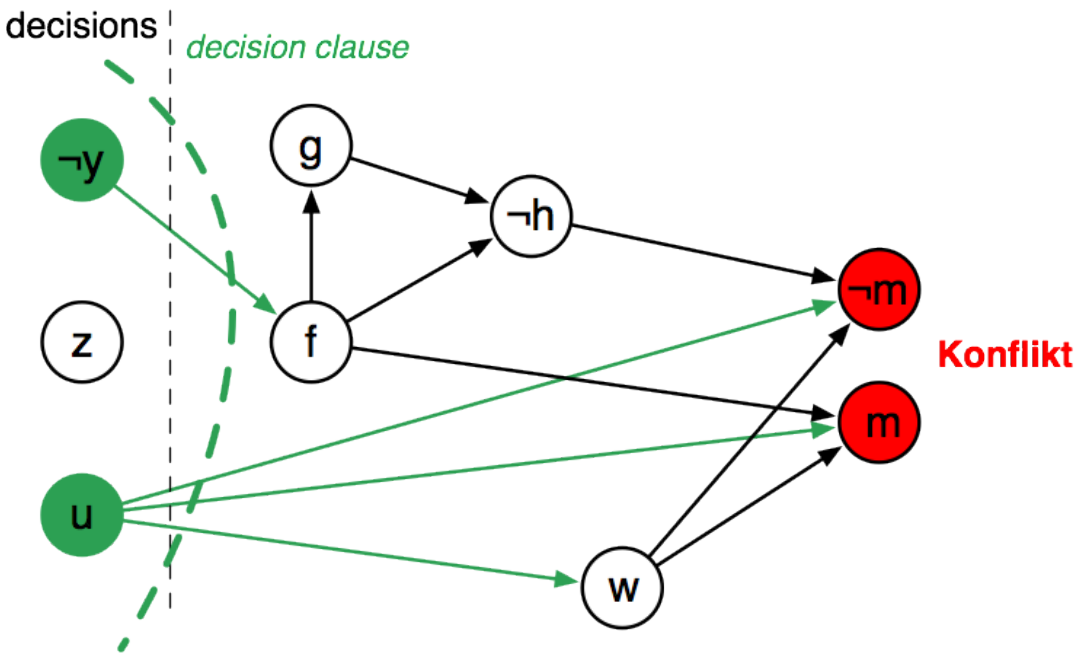
- Jeder Schnitt im Implikationsgraphen, der Entscheidungsvariablen von Konfliktvariablen trennt, ist ein gültiger *Cut*.
- Bilde eine neue Klausel aus allen Variablen, die eine ausgehende Kante durch den Cut besitzen.



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 *Cut*.
- Bilde eine neue Klausel aus allen Variablen, die eine ausgehende Kante durch den Cut besitzen.



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

Wiederholung: Resolution — 1

Definition (Resolution)

Kommt ein Literal λ positiv in einer Klausel c_1 und negativ in einer Klausel c_2 vor, so kann man die Resolvente

$$r = c_1 \setminus \{\lambda\} \cup c_2 \setminus \{\neg\lambda\}$$

berechnen.

Beispiel (Resolution)

- $c_1 = \{a, \neg b, c, \neg d\}$
- $c_2 = \{\neg a, e, \neg f\}$

Resolvente: $r = \{\neg b, c, \neg d, e, \neg f\}$

Bemerkung: Es darf immer nur **genau ein Literal** *resolviert werden.*

Wiederholung: Resolution — 2

Bedeutung der Resolvente

Für eine Klauselmenge C und eine Resolvente r zweier Klauseln $c_1, c_2 \in C$ gilt

$$C \equiv C \cup \{r\}.$$

Beweis[1]**:**
Sei α eine beliebige Belegung für C. Dann ist α auch eine Belegung für $C \cup \{r\}$ (da $\text{var}(C) = \text{var}(C \cup \{r\})$). Angenommen $\alpha \models C$, dann muss auch für alle Klauseln $c \in C$ gelten, dass $\alpha \models c$.

Die Resolvente r der beiden Klauseln $c_1, c_2 \in C$ hat die Form

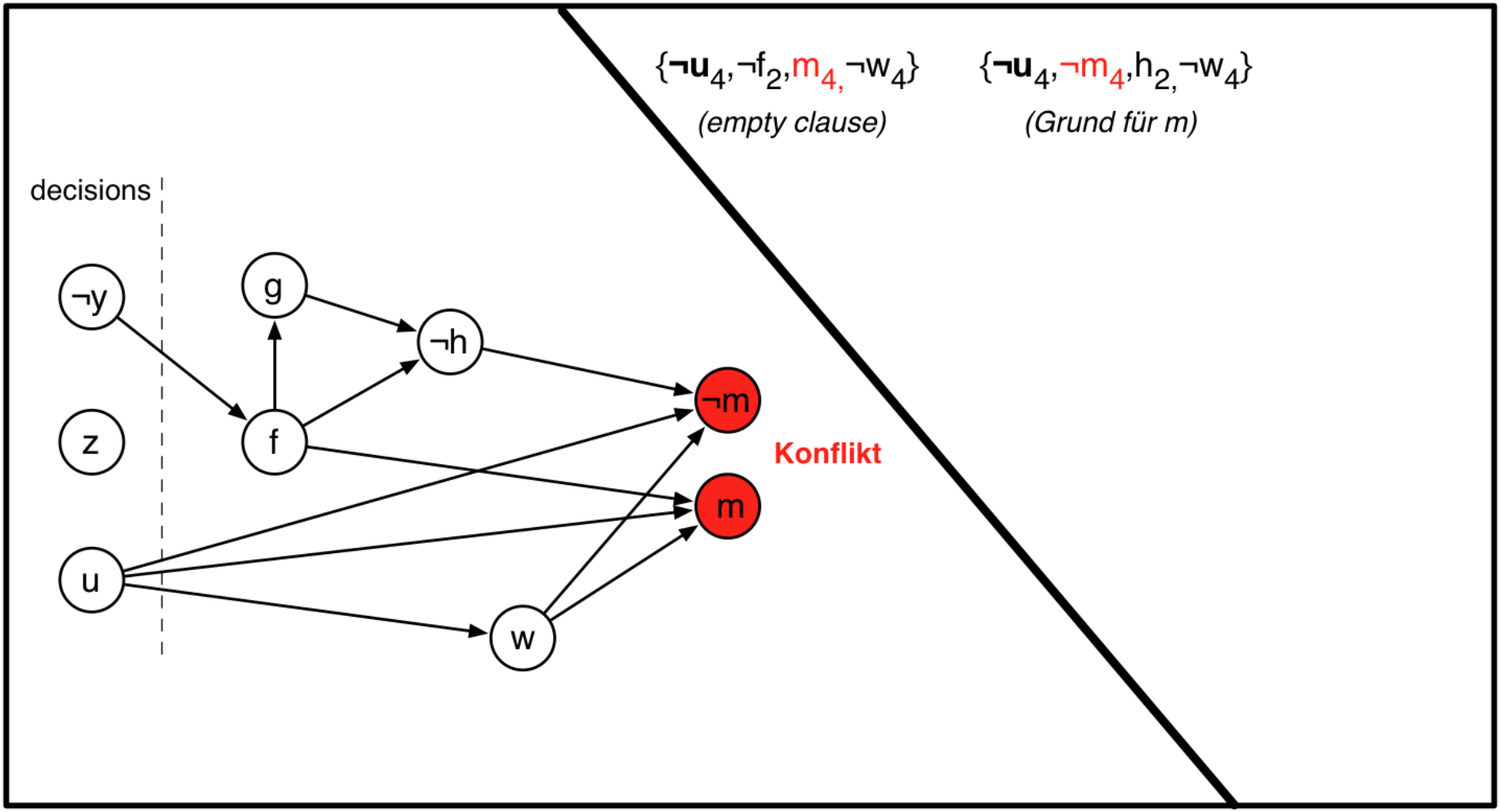
$$c_1 \setminus \{\lambda\} \cup c_2 \setminus \{\neg\lambda\}.$$

Wir betrachten zwei Fälle:

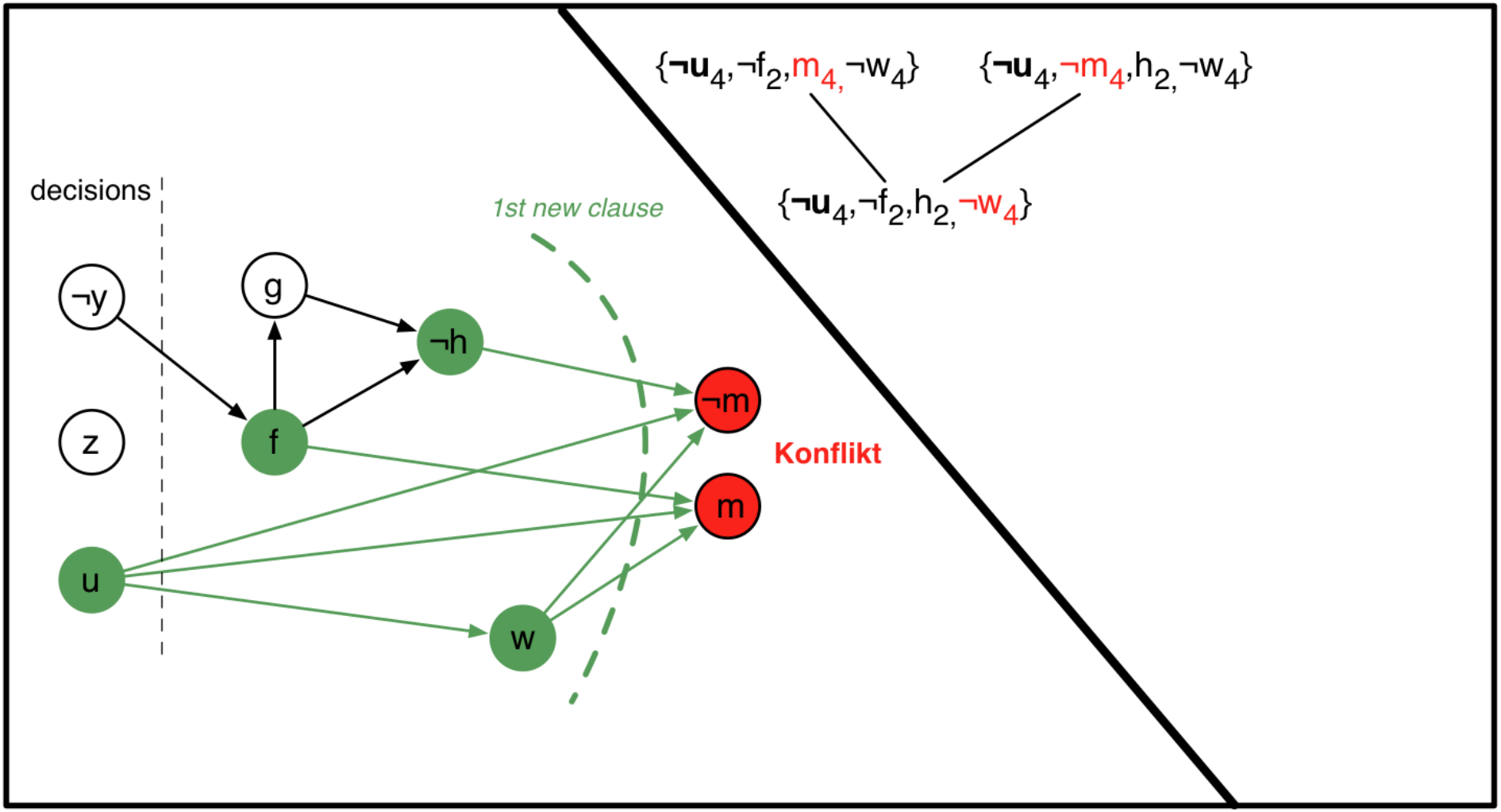
1. $\alpha \models \lambda$: Aus $\alpha \models c_2$ und $\alpha \not\models \neg\lambda$ folgt $\alpha \models (c_2 \setminus \{\neg\lambda\})$ und daher $\alpha \models r$
2. $\alpha \not\models \lambda$: Aus $\alpha \models c_1$ folgt $\alpha \models (c_1 \setminus \{\lambda\})$ und daher $\alpha \models r$.

[1] Theorem 7 aus *Logic for computer scientists*, Uli Furbach, `http://bit.ly/cY6dwS`

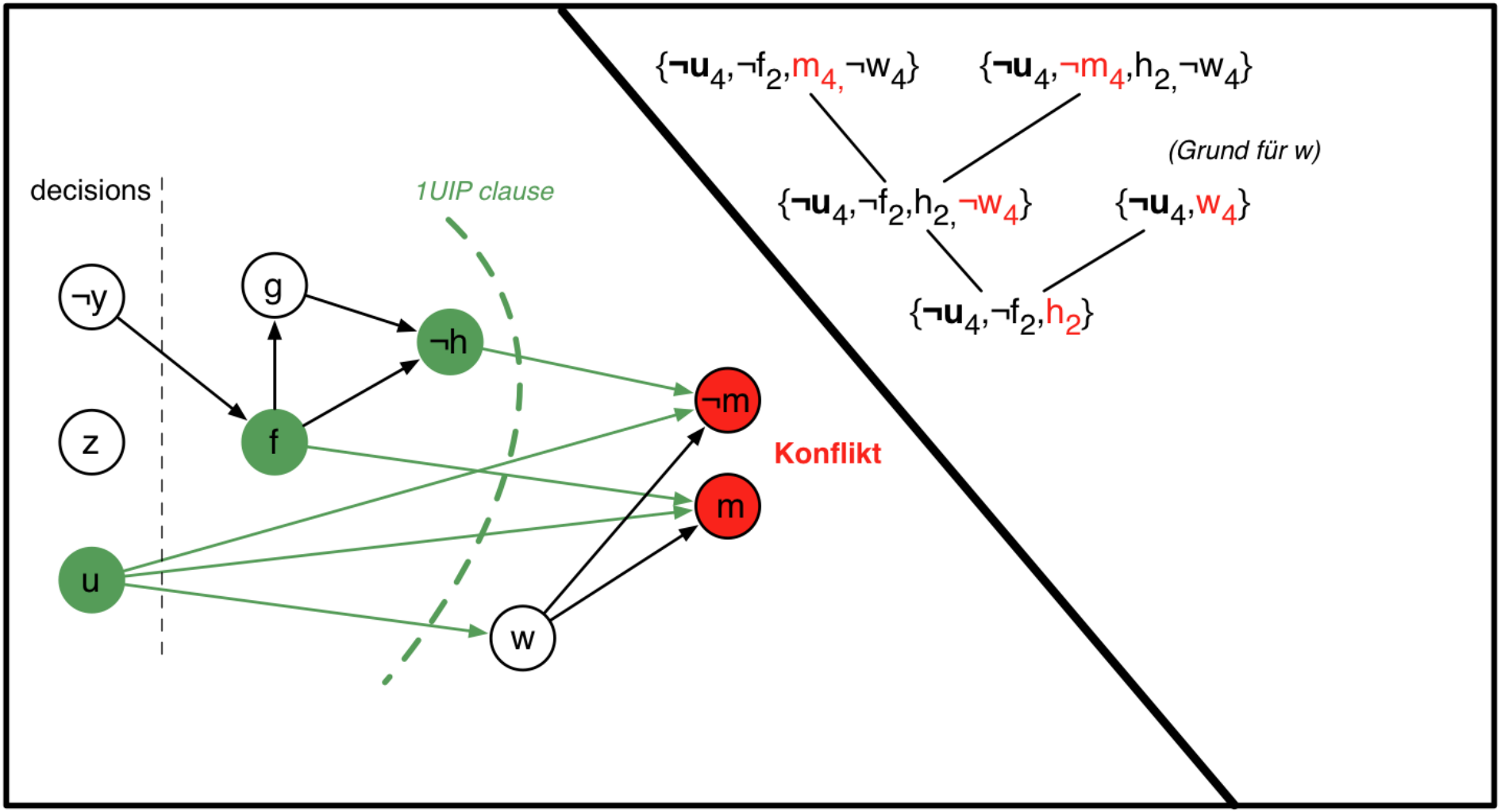
Zusammenhang zwischen Resolution und dem Implikationsgraphen



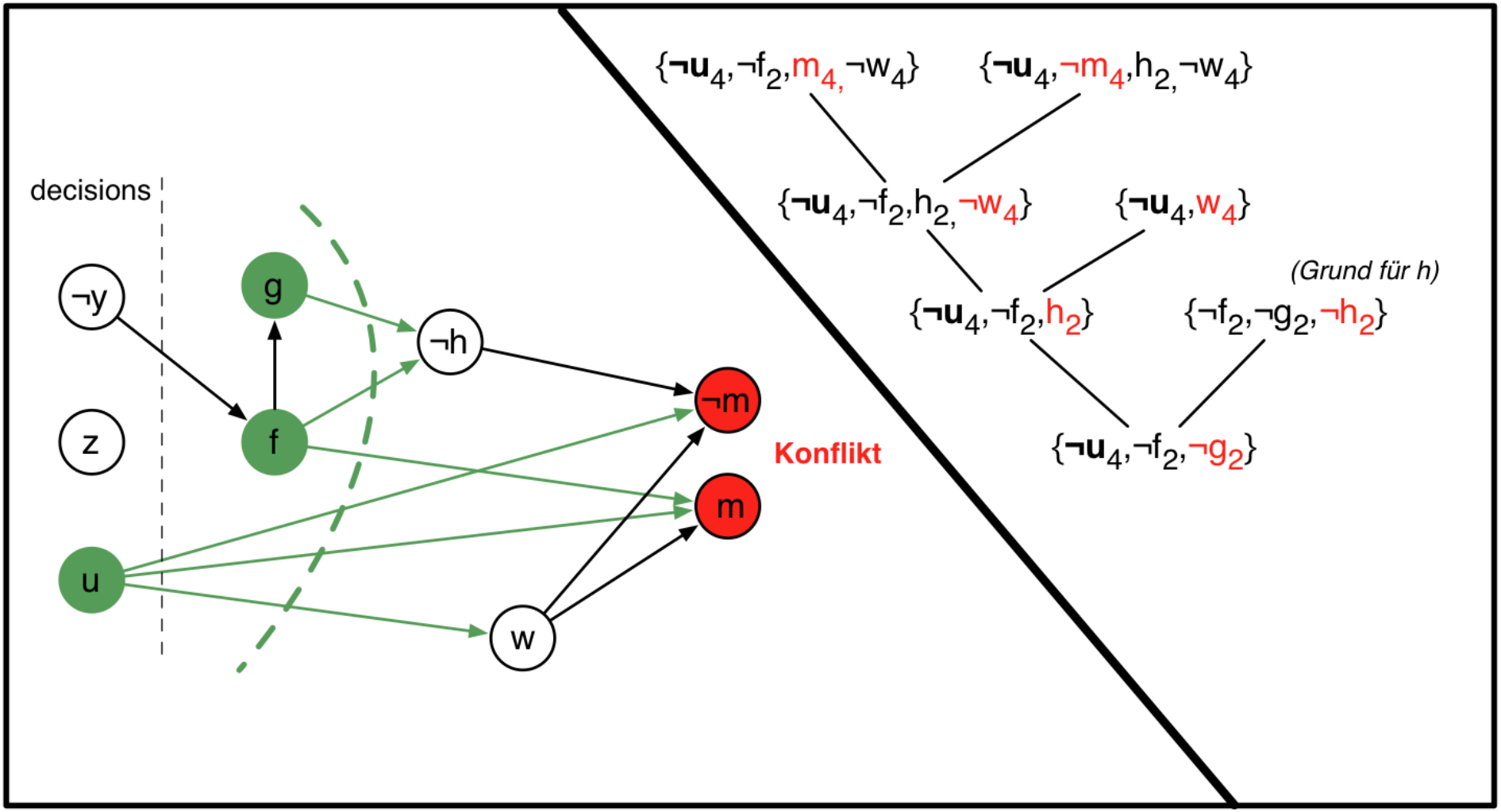
Zusammenhang zwischen Resolution und dem Implikationsgraphen



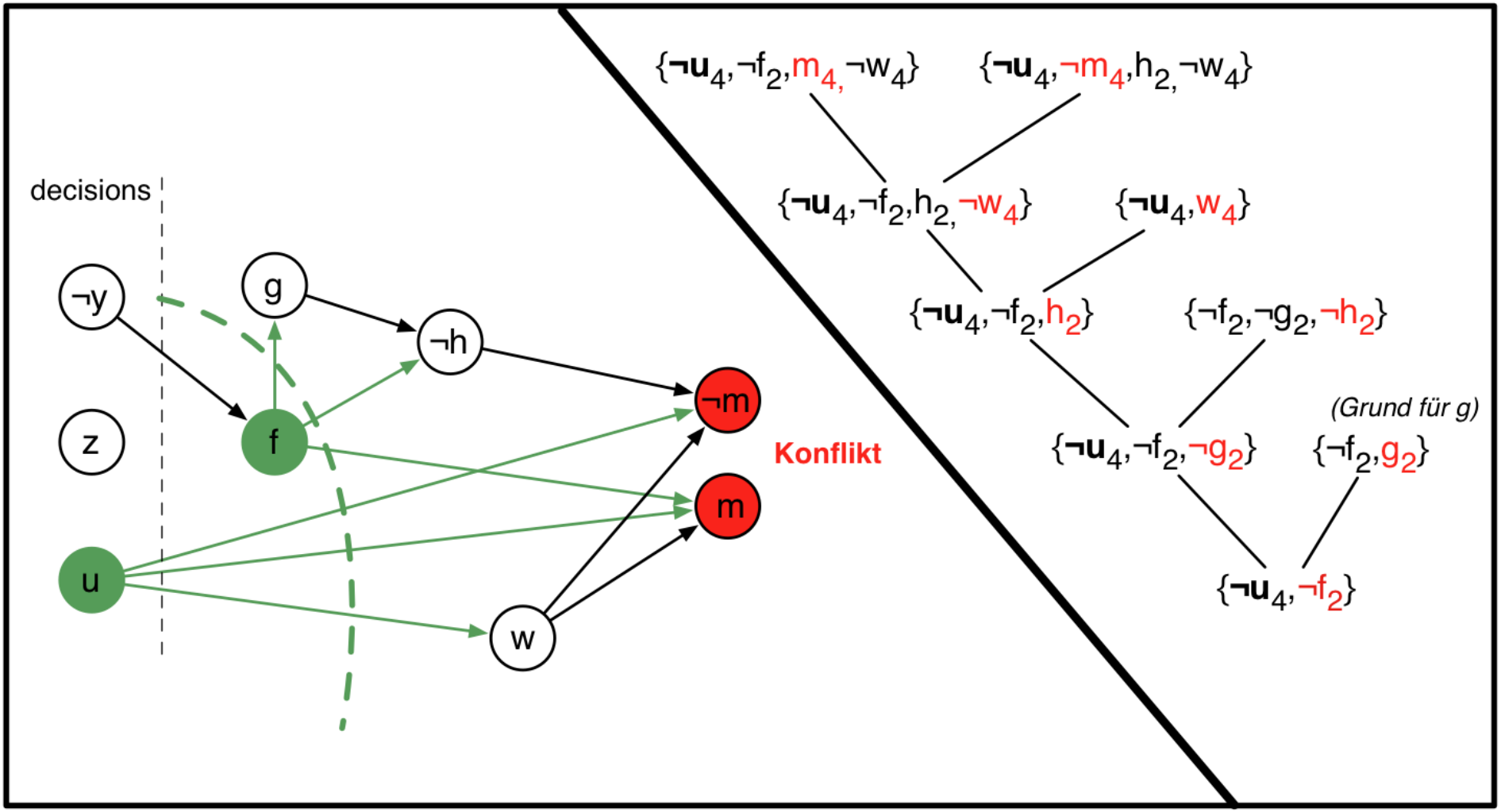
Zusammenhang zwischen Resolution und dem Implikationsgraphen



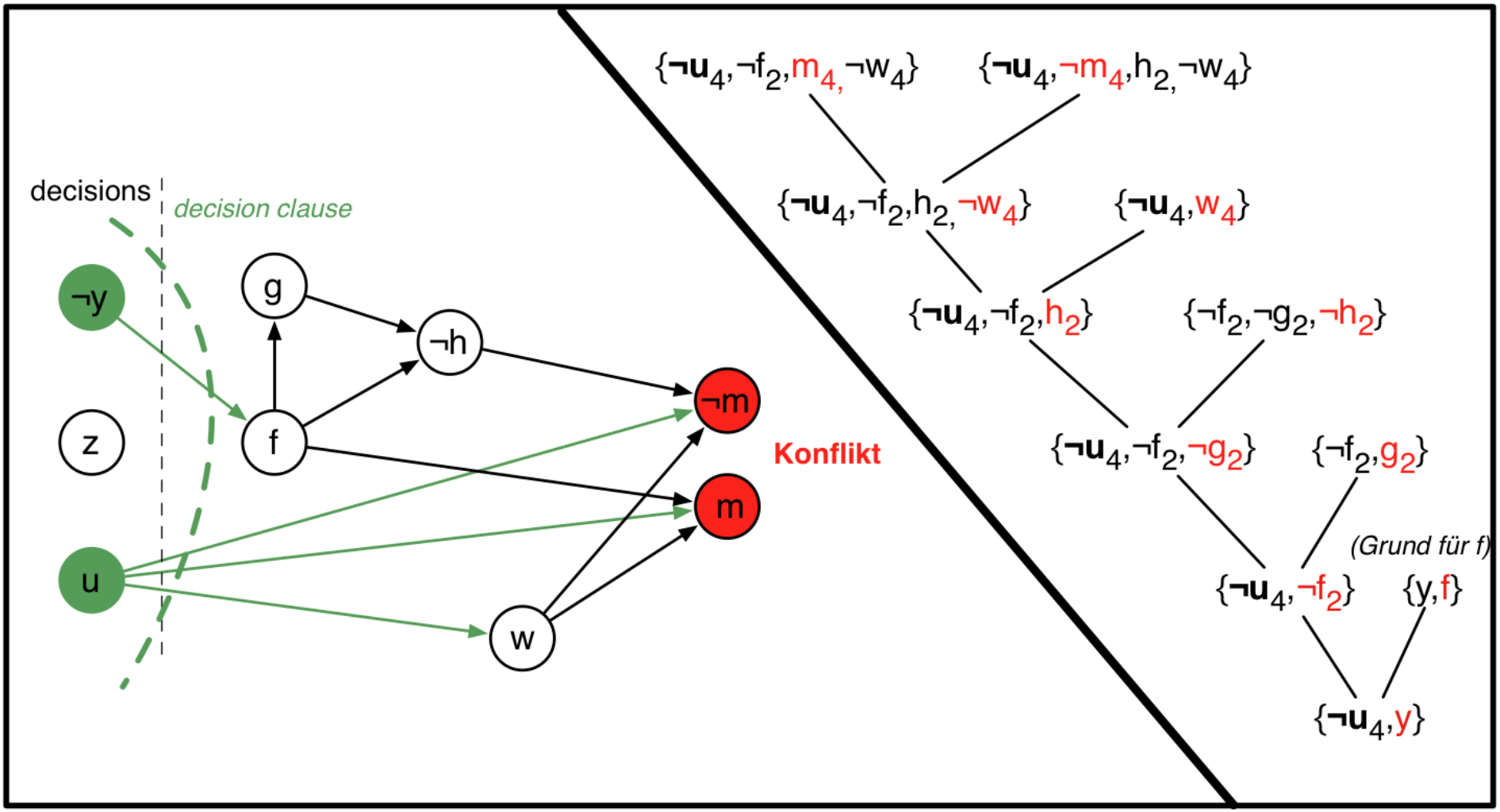
Zusammenhang zwischen Resolution und dem Implikationsgraphen



Zusammenhang zwischen Resolution und dem Implikationsgraphen



Zusammenhang zwischen Resolution und dem Implikationsgraphen



Verschiedene Strategien zum Lernen neuer Klauseln

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
- 1UIP Clause (Unique Implication Point): In der neuen Klausel ist genau eine Variable auf höchstem Decision Level. Diese Klausel wird nach Backtracking eine Unit Klausel.

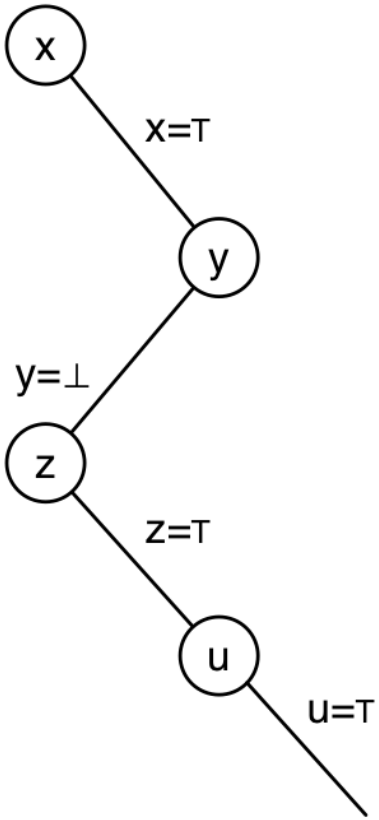
Einsetzen der 1UIP Klausel

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

Auflösung des Beispiels mit der 1UIP Clause

{¬u,w}
{¬u,¬m,h,¬w}
{¬u,¬f,m,¬w}
{¬f,¬g,¬h}
{y,f}
{¬f,g}
.
.
.
{¬x,a}
{¬x,b,¬a}
{¬x,¬a,¬b,c}
{¬a,¬d}
{¬b,d,¬e}
.
.
.

{¬u,¬f,h} *(Neue Klausel)*

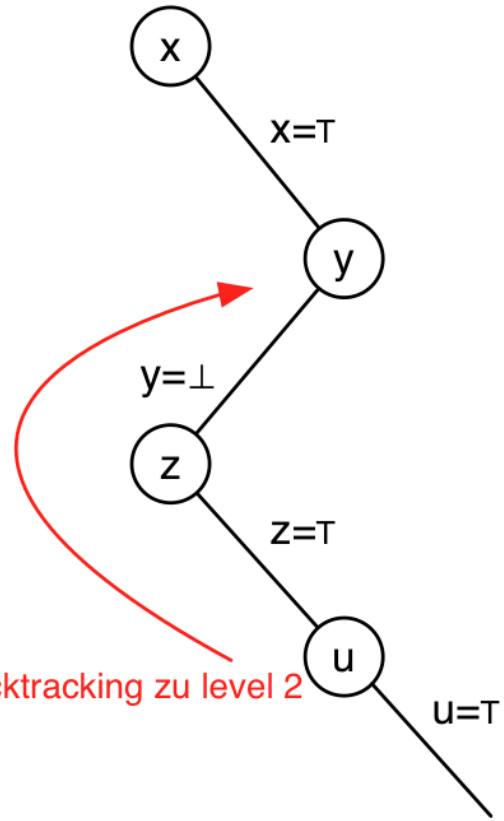


Konflikt: {¬u,¬f,m,¬w}

Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	⊤	decision
4	u	⊤	decision
	w	⊤	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}

Auflösung des Beispiels mit der 1UIP Clause

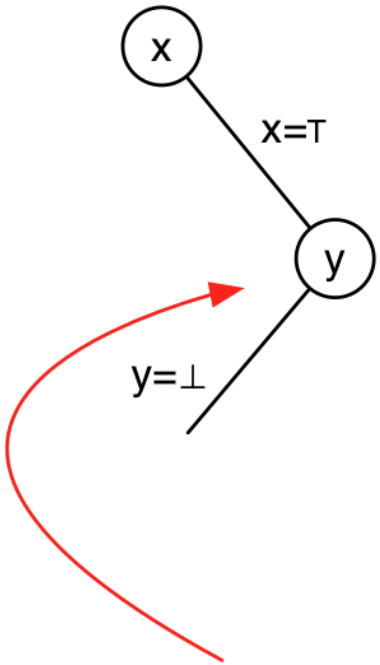
{¬u,w}
{¬u,¬m,h,¬w}
{¬u,¬f,m,¬w}
{¬f,¬g,¬h}
{y,f}
{¬f,g}
.
.
.
{¬x,a}
{¬x,b,¬a}
{¬x,¬a,¬b,c}
{¬a,¬d}
{¬b,d,¬e}
.
.
.
{¬u,¬f,h} *(Neue Klausel)*



Level	Variable	Wert	Grund
1	x	T	decision
	a	T	{¬x,a}
	b	T	{¬x,b,¬a}
	c	T	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	T	{y,f}
	g	T	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
3	z	T	decision
4	u	T	decision
	w	T	{¬u,w}
	m	⊥	{¬u,¬m,h,¬w}

Auflösung des Beispiels mit der 1UIP Clause

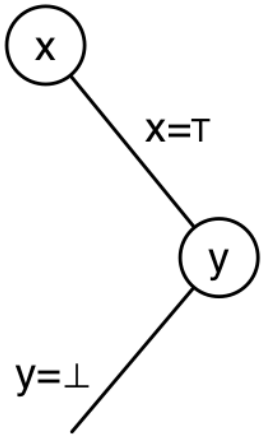
{¬u,w}
{¬u,¬m,h,¬w}
{¬u,¬f,m,¬w}
{¬f,¬g,¬h}
{y,f}
{¬f,g}
.
.
.
{¬x,a}
{¬x,b,¬a}
{¬x,¬a,¬b,c}
{¬a,¬d}
{¬b,d,¬e}
.
.
.
{¬u,¬f,h} *(Neue Klausel)*



Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}

Auflösung des Beispiels mit der 1UIP Clause

{¬u,w}
{¬u,¬m,h,¬w}
{¬u,¬f,m,¬w}
{¬f,¬g,¬h}
{y,f}
{¬f,g}
.
.
.
{¬x,a}
{¬x,b,¬a}
{¬x,¬a,¬b,c}
{¬a,¬d}
{¬b,d,¬e}
.
.
.
{¬u,¬f,h} *(Neue Klausel)*



Level	Variable	Wert	Grund
1	x	⊤	decision
	a	⊤	{¬x,a}
	b	⊤	{¬x,b,¬a}
	c	⊤	{¬x,¬a,¬b,c}
	d	⊥	{¬a,¬d}
	e	⊥	{¬b,d,¬e}
2	y	⊥	decision
	f	⊤	{y,f}
	g	⊤	{¬f,g}
	h	⊥	{¬f,¬g,¬h}
	u	⊥	**{¬u,¬f,¬h}**

Nach UP von u sind alle gegebenen Klauseln erfüllt

Der CDCL Algorithmus

Algorithm 1: sat(C)

Input: Clauseset C
Output: `SAT` or `UNSAT`

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

Der Konfliktanalyse Algorithmus

Algorithm 2: analyzeConf(ec, C)

Input: an empty clause ec, the set of clauses C
Output: the backtracking level
if *current decision level* $==$ *0* **then**
  return -1
lv = the last assigned variable before the conflict clause;
reason = the reason of lv;
newclause = resolve(ec,reason) ; `// Erste Resolution`
while *the stop criterion for newclause is not met* **do**
  cv = chooseLiteral(newclause);
  reason = the reason of cv;
  newclause = resolve(newclause,reason) ; `// Weitere Resolution`
$C = C \cup \{\text{newclause}\}$; `// Füge neue Klausel hinzu`
level = computeBacktrackLevel(newclause) ; `// Berechne Backtrack Level`
return *level*;

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

Korrektheit des CDCL Algorithmus — 1[2]

Theorem (Erfüllbarkeit einer Formel)

Ist die Ausgabe von $\mathrm{sat}(C) = \mathtt{SAT}$*, so ist die Klauselmenge* C *erfüllbar.*

Beweis.

Der Algorithmus gibt nur dann `SAT` zurück, wenn eine erfüllende Belegung α gefunden wurde. Da die neu gelernten Klauseln die Semantik der Originalformel nicht verändern, muss α eine erfüllende Belegung für die originale Formel sein. □

[2] Die folgenden Beweise finden sich in: L. Zhang, and S. Malik. **Validating SAT solvers using an independent resolution-based checker: Practical implementations and other applications.** In Proceedings of DATE '03. 2003.

Korrektheit des CDCL Algorithmus — 2

Theorem (Unerfüllbarkeit einer Formel)

Ist die Ausgabe von $\text{sat}(C) = \texttt{UNSAT}$*, so ist die Klauselmenge* C *eine Kontradiktion.*

Beweis.

Der Algorithmus gibt nur dann `UNSAT` zurück, wenn ein Konflikt auf Decision Level 0 gefunden wurde. Von diesem Konflikt ausgehend kann man nun wie in der `while`-Schleife im Algorithmus *analyzeConf* Resolutionen ausführen. Auf Level 0 gibt es keine Entscheidungsvariable (alle Variablen wurden durch UP belegt). D.h. im Speziellen muss es zu jeder Variable eine Reason geben. Da in jedem Durchlauf der `while`-Schleife eine Variable aus `newclause` durch Resolution entfernt wird und es zu jeder Variable eine Reason gibt, endet man nach spätestens n Schritten (Anzahl der Variablen in der Konfliktklausel) bei der leeren Klausel. D.h. die leere Klausel ist aus der Formel durch Resolution erzeugbar und daher muss die Formel unerfüllbar sein. □

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

Termination des CDCL Algorithmus — 1

Theorem (Termination des Algorithmus)

Der Algorithmus sat(C) *terminiert für jede beliebige Eingabeklauselmenge* C.

Vorbereitung des Beweises — 1

Sei $k(l)$ die Anzahl der Variablen, die auf Level l belegt wurden. Sei n die 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.

Der Wert von f steigt während des Solving Prozesses monoton an. Solange kein Konflikt auftritt ist dies offensichtlich, da in jedem Schritt neue Variablen auf dem höchsten Decision Level belegt werden, ohne dass ältere Variablenbelegungen verändert werden.

Tritt ein Konflikt auf, erfolgt ein Backtracking zu einem kleineren Decision Level und auf diesem Level wird mindestens eine neue Variable belegt. Aus der Ungleichung (1) folgt, dass auch in diesem Fall der Wert von f ansteigt — ungeachtet der Variablenbelegungen, die während des Backtracking Prozesses rückgängig gemacht wurden.

Da f nur eine endliche Anzahl von verschiedenen Werten annehmen kann, muss der Algorithmus also terminieren. □

Partielle Korrektheit + Termination $\Rightarrow$ Totale Korrektheit