

1. Teil: Formale Sprachen und Automaten

2. Teil: **Aussagenlogik**

- Grundbegriffe der Aussagenlogik
- Hornformeln
- SAT-Beweiser
- **Resolution**
- binäre Entscheidungsgraphen
- quantifizierte Boolesche Formeln



fasse Klauseln als Mengen von Literalen auf

$$L_1 \vee L_2 \vee \dots \vee L_k \rightsquigarrow \{L_1, L_2, \dots, L_k\}$$

$$x \vee \neg y \vee z \rightsquigarrow \{x, \neg y, z\}$$

$$z \rightsquigarrow \{z\}$$

$$\mathit{false} = \perp \rightsquigarrow \emptyset$$

fasse KNF-Formeln als Mengen von Klauseln auf

$$\kappa_1 \wedge \kappa_2 \wedge \dots \wedge \kappa_m \rightsquigarrow \{\kappa_1, \kappa_2, \dots, \kappa_m\}$$

$$(x \vee \neg y) \wedge y \wedge (\neg x \vee z) \rightsquigarrow \{\{x, \neg y\}, \{y\}, \{\neg x, z\}\}$$

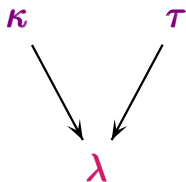
$$\mathit{true} \rightsquigarrow \emptyset$$

Seien  $\kappa$ ,  $\tau$  zwei Klauseln und  $L$  ein Literal mit  $L \in \kappa$  und  $\bar{L} \in \tau$ . Dann wird die Klausel

$$\lambda = \kappa \setminus \{L\} \cup \tau \setminus \{\bar{L}\}$$

ein Resolvent von  $\kappa$  und  $\tau$  genannt.

$\kappa$  und  $\tau$  heißen Elternklauseln von  $\lambda$ . Schreibweise:



Elternklauseln

Resolvent

Seien  $\kappa$ ,  $\tau$  zwei Klauseln und  $L$  ein Literal mit  $L \in \kappa$  und  $\bar{L} \in \tau$ . Dann wird die Klausel

$$\lambda = \kappa \setminus \{L\} \cup \tau \setminus \{\bar{L}\}$$

ein Resolvent von  $\kappa$  und  $\tau$  genannt.

Beispiel:  $\kappa = x \vee \neg y \vee z$        $\tau = \neg x \vee w$

$\{x, \neg y, z\}$                        $\{\neg x, w\}$

Klausel  $\neg y \vee z \vee w$

Sei  $\alpha$  eine KNF-Formel mit den Atomen  $x_1, \dots, x_n$ .

Resolutionsabschluss von  $\alpha$ :

$$Res^*(\alpha) = \bigcup_{i \geq 0} Res^i(\alpha) = Res^k(\alpha) \quad \text{für ein } k \leq 4^n$$

Für alle Klauseln  $\lambda$  gilt:

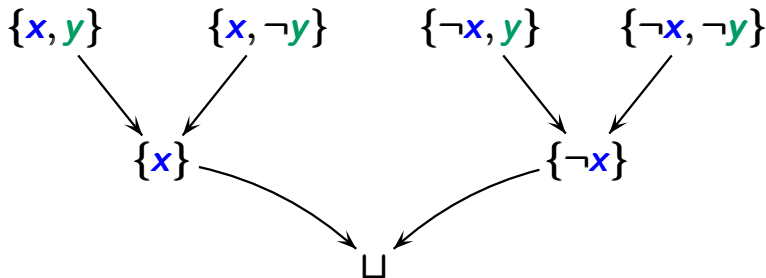
$$\lambda \in Res^*(\alpha) \quad \text{gdw} \quad \left\{ \begin{array}{l} \text{es gibt eine Herleitung} \\ \text{von } \lambda \text{ aus } \alpha \end{array} \right.$$

Insbesondere: ist  $\lambda$  aus  $\alpha$  herleitbar, so gilt  $\alpha \Vdash \lambda$

ist  $\perp$  aus  $\alpha$  herleitbar, so ist  $\alpha$  unerfüllbar.

$$\alpha = (x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

unerfüllbare KNF-Formel



Widerlegung: Herleitung der leeren Klausel

Für jede KNF-Formel  $\alpha$  gilt:

$\alpha$  unerfüllbar    gdw     $\perp \in Res^*(\alpha)$

Beweis von “ $\Leftarrow$ ”: folgt aus dem Resolventenlemma

Gilt  $\perp = \mathbf{false} \in Res^*(\alpha)$ , so gilt  $\alpha \Vdash \mathbf{false}$ .

Also ist  $\alpha$  unerfüllbar.

Beweis von “ $\Rightarrow$ ”: durch Induktion nach der Anzahl  $n$   
der in  $\alpha$  vorkommenden Atome

$\perp = \mathbf{false}$     leere Klausel

$Res^*(\alpha)$     Resolutionsabschluss von  $\alpha$



Sei  $\alpha$  eine KNF-Formel mit  $n+1$  Atomen, wobei  $n \geq 0$ .

Sei  $\alpha$  eine KNF-Formel mit  $n+1$  Atomen, wobei  $n \geq 0$ .  
Sei  $x$  ein Atom, das in  $\alpha$  vorkommt.

Sei  $\alpha$  eine KNF-Formel mit  $n+1$  Atomen, wobei  $n \geq 0$ .

Sei  $x$  ein Atom, das in  $\alpha$  vorkommt. In den KNF-Formeln

$$\alpha_0 = \alpha[x/0] \quad \text{und} \quad \alpha_1 = \alpha[x/1]$$

kommen jeweils höchstens  $n$  Atome vor.

Sei  $\alpha$  eine KNF-Formel mit  $n+1$  Atomen, wobei  $n \geq 0$ .

Sei  $x$  ein Atom, das in  $\alpha$  vorkommt. In den KNF-Formeln

$$\alpha_0 = \alpha[x/0] \quad \text{und} \quad \alpha_1 = \alpha[x/1]$$

kommen jeweils höchstens  $n$  Atome vor. Mit  $\alpha$  sind auch  $\alpha_0$  und  $\alpha_1$  unerfüllbar.

---

Splitting-Regel:

$$\alpha \text{ erfüllbar} \quad \text{gdw} \quad \left\{ \begin{array}{l} \text{wenigstens eine der Formeln} \\ \alpha[x/0] \text{ oder } \alpha[x/1] \\ \text{ist erfüllbar} \end{array} \right.$$

Sei  $\alpha$  eine KNF-Formel mit  $n+1$  Atomen, wobei  $n \geq 0$ .

Sei  $x$  ein Atom, das in  $\alpha$  vorkommt. In den KNF-Formeln

$$\alpha_0 = \alpha[x/0] \quad \text{und} \quad \alpha_1 = \alpha[x/1]$$

kommen jeweils höchstens  $n$  Atome vor. Mit  $\alpha$  sind auch  $\alpha_0$  und  $\alpha_1$  unerfüllbar. Nach Induktionsvoraussetzung gilt:

$$\perp \in \text{Res}^*(\alpha_0) \quad \text{und} \quad \perp \in \text{Res}^*(\alpha_1)$$

Sei  $\alpha$  eine KNF-Formel mit  $n+1$  Atomen, wobei  $n \geq 0$ .  
Sei  $x$  ein Atom, das in  $\alpha$  vorkommt. In den KNF-Formeln

$$\alpha_0 = \alpha[x/0] \quad \text{und} \quad \alpha_1 = \alpha[x/1]$$

kommen jeweils höchstens  $n$  Atome vor. Mit  $\alpha$  sind auch  $\alpha_0$  und  $\alpha_1$  unerfüllbar. Nach Induktionsvoraussetzung gilt:

$$\perp \in \text{Res}^*(\alpha_0) \quad \text{und} \quad \perp \in \text{Res}^*(\alpha_1)$$

Betrachte nun **Widerlegungen** für  $\alpha_0$  und  $\alpha_1$

Sei  $\alpha$  eine KNF-Formel mit  $n+1$  Atomen, wobei  $n \geq 0$ .

Sei  $x$  ein Atom, das in  $\alpha$  vorkommt. In den KNF-Formeln

$$\alpha_0 = \alpha[x/0] \quad \text{und} \quad \alpha_1 = \alpha[x/1]$$

kommen jeweils höchstens  $n$  Atome vor. Mit  $\alpha$  sind auch  $\alpha_0$  und  $\alpha_1$  unerfüllbar. Nach Induktionsvoraussetzung gilt:

$$\perp \in Res^*(\alpha_0) \quad \text{und} \quad \perp \in Res^*(\alpha_1)$$

Betrachte nun Widerlegungen für  $\alpha_0$  und  $\alpha_1$  und

- lifte die Widerlegung für  $\alpha_0$  zu einer Herleitung von  $\perp$  oder  $x$  aus  $\alpha$
- lifte die Widerlegung für  $\alpha_1$  zu einer Herleitung von  $\perp$  oder  $\neg x$  aus  $\alpha$

Sei  $\alpha$  eine KNF-Formel mit  $n+1$  Atomen, wobei  $n \geq 0$ .

Sei  $x$  ein Atom, das in  $\alpha$  vorkommt. In den KNF-Formeln

$$\alpha_0 = \alpha[x/0] \quad \text{und} \quad \alpha_1 = \alpha[x/1]$$

kommen jeweils höchstens  $n$  Atome vor. Mit  $\alpha$  sind auch  $\alpha_0$  und  $\alpha_1$  unerfüllbar. Nach Induktionsvoraussetzung gilt:

$$\sqcup \in Res^*(\alpha_0) \quad \text{und} \quad \sqcup \in Res^*(\alpha_1)$$

Betrachte nun Widerlegungen für  $\alpha_0$  und  $\alpha_1$  und

- lifte die Widerlegung für  $\alpha_0$  zu einer Herleitung von  $\sqcup$  oder  $x$  aus  $\alpha$
- lifte die Widerlegung für  $\alpha_1$  zu einer Herleitung von  $\sqcup$  oder  $\neg x$  aus  $\alpha$

Hieraus folgt:

$$\sqcup \in Res^*(\alpha)$$

Sei  $\alpha$  eine KNF-Formel mit  $n+1$  Atomen, wobei  $n \geq 0$ .

Sei  $x$  ein Atom, das in  $\alpha$  vorkommt. In den KNF-Formeln

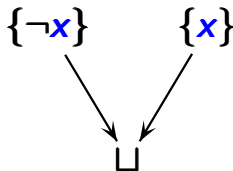
$$\alpha_0 = \alpha[x/0] \quad \text{und} \quad \alpha_1 = \alpha[x/1]$$

kommen jeweils höchstens  $n$  Atome vor. Mit  $\alpha$  sind auch  $\alpha_0$  und  $\alpha_1$  unerfüllbar. Nach Induktionsvoraussetzung gilt:

$$\perp \in \text{Res}^*(\alpha_0) \quad \text{und} \quad \perp \in \text{Res}^*(\alpha_1)$$

Betrachte nun Widerlegungen für  $\alpha_0$  und  $\alpha_1$  und

- lifte die Widerlegung für  $\alpha_0$  zu einer Herleitung von  $\perp$  oder  $x$  aus  $\alpha$
- lifte die Widerlegung für  $\alpha_1$  zu einer Herleitung von  $\perp$  oder  $\neg x$  aus  $\alpha$





$$\alpha = (x \vee \neg y \vee z) \wedge (y \vee z) \wedge (\neg x \vee z) \wedge \neg z$$

$$\alpha = (x \vee \neg y \vee z) \wedge (y \vee z) \wedge (\neg x \vee z) \wedge \neg z$$

$$\alpha_0 = \alpha[x/0]$$

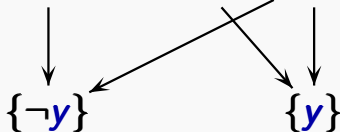
$$(\neg y \vee z) \wedge (y \vee z) \wedge \neg z$$

$$\alpha = (x \vee \neg y \vee z) \wedge (y \vee z) \wedge (\neg x \vee z) \wedge \neg z$$

$$\alpha_0 = \alpha[x/0]$$

$$(\neg y \vee z) \wedge (y \vee z) \wedge \neg z$$

$$\{\neg y, z\} \quad \{y, z\} \quad \{\neg z\}$$

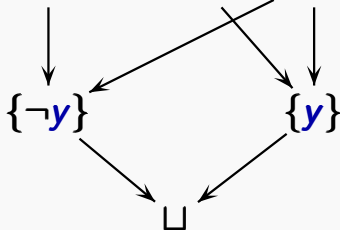


$$\alpha = (x \vee \neg y \vee z) \wedge (y \vee z) \wedge (\neg x \vee z) \wedge \neg z$$

$$\alpha_0 = \alpha[x/0]$$

$$(\neg y \vee z) \wedge (y \vee z) \wedge \neg z$$

$$\{\neg y, z\} \quad \{y, z\} \quad \{\neg z\}$$



$$\alpha_1 = \alpha[x/1]$$

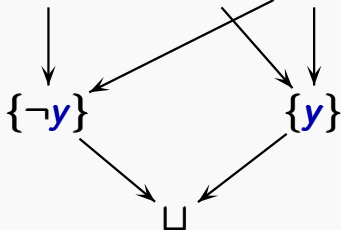
$$(y \vee z) \wedge z \wedge \neg z$$

$$\alpha = (x \vee \neg y \vee z) \wedge (y \vee z) \wedge (\neg x \vee z) \wedge \neg z$$

$$\alpha_0 = \alpha[x/0]$$

$$(\neg y \vee z) \wedge (y \vee z) \wedge \neg z$$

$\{\neg y, z\}$     $\{y, z\}$     $\{\neg z\}$



$$\alpha_1 = \alpha[x/1]$$

$$(y \vee z) \wedge z \wedge \neg z$$

$\{y, z\}$     $\{z\}$     $\{\neg z\}$



# Beispiel: Lifting von Widerlegungen

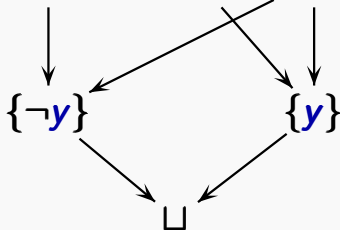
225

$$\alpha = (x \vee \neg y \vee z) \wedge (y \vee z) \wedge (\neg x \vee z) \wedge \neg z$$

$$\alpha_0 = \alpha[x/0]$$

$$(\neg y \vee z) \wedge (y \vee z) \wedge \neg z$$

$$\{\neg y, z\} \quad \{y, z\} \quad \{\neg z\}$$



$$\alpha_1 = \alpha[x/1]$$

$$(y \vee z) \wedge z \wedge \neg z$$

$$\{y, z\} \quad \{z\} \quad \{\neg z\}$$



Wiedereinfügen von  $\neg x$

# Beispiel: Lifting von Widerlegungen

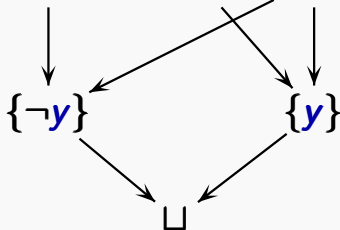
225

$$\alpha = (x \vee \neg y \vee z) \wedge (y \vee z) \wedge (\neg x \vee z) \wedge \neg z$$

$$\alpha_0 = \alpha[x/0]$$

$$(\neg y \vee z) \wedge (y \vee z) \wedge \neg z$$

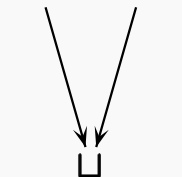
$\{\neg y, z\}$     $\{y, z\}$     $\{\neg z\}$



$$\alpha_1 = \alpha[x/1]$$

$$(y \vee z) \wedge z \wedge \neg z$$

$\{y, z\}$     $\{z\}$     $\{\neg z\}$



Wiedereinfügen von  $\neg x$

# Beispiel: Lifting von Widerlegungen

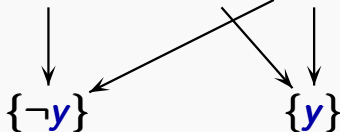
225

$$\alpha = (x \vee \neg y \vee z) \wedge (y \vee z) \wedge (\neg x \vee z) \wedge \neg z$$

$$\alpha_0 = \alpha[x/0]$$

$$(\neg y \vee z) \wedge (y \vee z) \wedge \neg z$$

$$\{\neg y, z\} \quad \{y, z\} \quad \{\neg z\}$$



$$\alpha_1 = \alpha[x/1]$$

$$(y \vee z) \wedge z \wedge \neg z$$

$$\{\neg x, z\} \quad \{\neg z\}$$



Wiedereinfügen von  $\neg x$

# Beispiel: Lifting von Widerlegungen

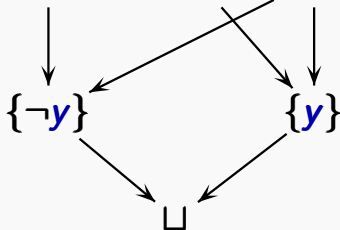
225

$$\alpha = (x \vee \neg y \vee z) \wedge (y \vee z) \wedge (\neg x \vee z) \wedge \neg z$$

$$\alpha_0 = \alpha[x/0]$$

$$(\neg y \vee z) \wedge (y \vee z) \wedge \neg z$$

$$\{\neg y, z\} \quad \{y, z\} \quad \{\neg z\}$$



Herleitung von  $\{\neg x\}$   
aus  $\alpha$

$$\{\neg x, z\} \quad \{\neg z\}$$



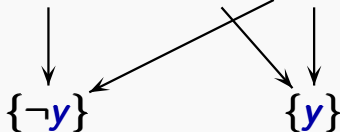
Wiedereinfügen von  $\neg x$

$$\alpha = (x \vee \neg y \vee z) \wedge (y \vee z) \wedge (\neg x \vee z) \wedge \neg z$$

$$\alpha_0 = \alpha[x/0]$$

$$(\neg y \vee z) \wedge (y \vee z) \wedge \neg z$$

$$\{\neg y, z\} \quad \{y, z\} \quad \{\neg z\}$$



Wiedereinfügen von  $x$

Herleitung von  $\{\neg x\}$   
aus  $\alpha$

$$\{\neg x, z\} \quad \{\neg z\}$$



Wiedereinfügen von  $\neg x$

# Beispiel: Lifting von Widerlegungen

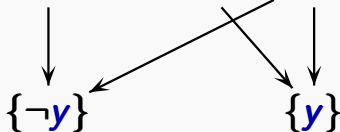
225

$$\alpha = (x \vee \neg y \vee z) \wedge (y \vee z) \wedge (\neg x \vee z) \wedge \neg z$$

$$\alpha_0 = \alpha[x/0]$$

$$(\neg y \vee z) \wedge (y \vee z) \wedge \neg z$$

$$\{\neg y, z\} \quad \{y, z\} \quad \{\neg z\}$$



Wiedereinfügen von  $x$

Herleitung von  $\{\neg x\}$   
aus  $\alpha$

$$\{\neg x, z\} \quad \{\neg z\}$$

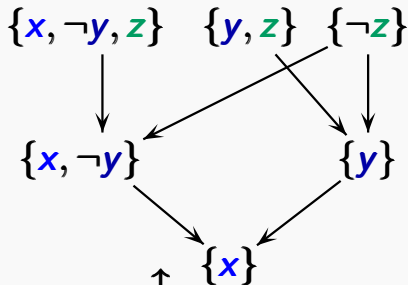


Wiedereinfügen von  $\neg x$

$$\alpha = (x \vee \neg y \vee z) \wedge (y \vee z) \wedge (\neg x \vee z) \wedge \neg z$$

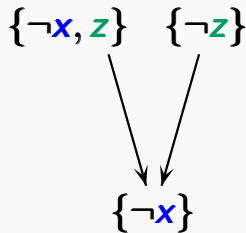
$$\alpha_0 = \alpha[x/0]$$

$$(\neg y \vee z) \wedge (y \vee z) \wedge \neg z$$



Wiedereinfügen von  $x$

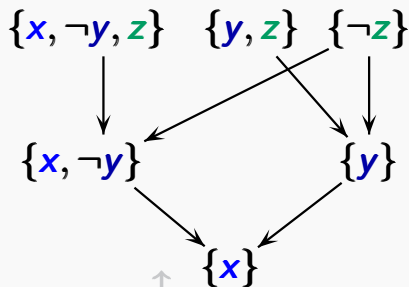
Herleitung von  $\{\neg x\}$   
aus  $\alpha$



Wiedereinfügen von  $\neg x$

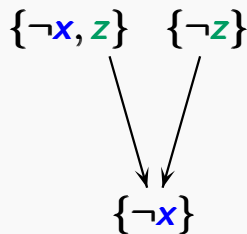
$$\alpha = (x \vee \neg y \vee z) \wedge (y \vee z) \wedge (\neg x \vee z) \wedge \neg z$$

Herleitung von  $\{x\}$   
aus  $\alpha$



Wiedereinfügen von  $x$

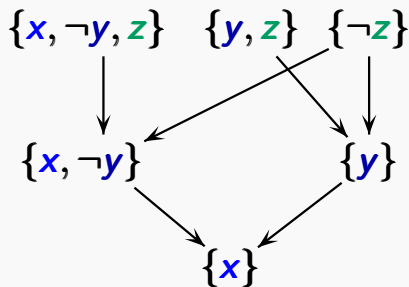
Herleitung von  $\{\neg x\}$   
aus  $\alpha$



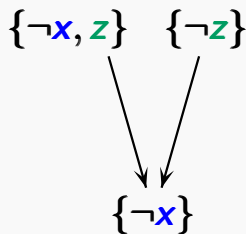
Wiedereinfügen von  $\neg x$

$$\alpha = (x \vee \neg y \vee z) \wedge (y \vee z) \wedge (\neg x \vee z) \wedge \neg z$$

Herleitung von  $\{x\}$   
aus  $\alpha$



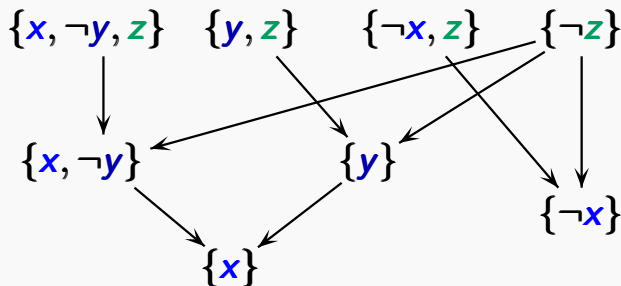
Herleitung von  $\{\neg x\}$   
aus  $\alpha$



kombiniere diese Herleitungen zu einer Widerlegung

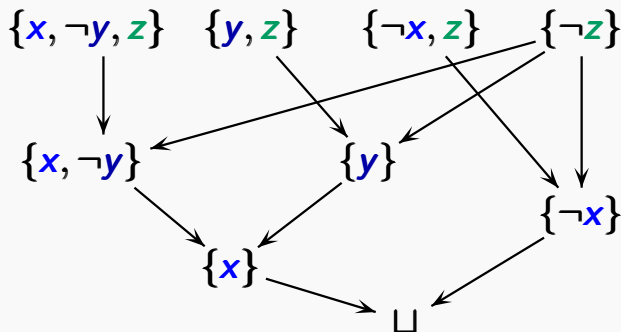
$$\alpha = (x \vee \neg y \vee z) \wedge (y \vee z) \wedge (\neg x \vee z) \wedge \neg z$$

Kombination der Herleitungen von  $\{x\}$  und  $\{\neg x\}$



$$\alpha = (x \vee \neg y \vee z) \wedge (y \vee z) \wedge (\neg x \vee z) \wedge \neg z$$

Kombination der Herleitungen von  $\{x\}$  und  $\{\neg x\}$



Widerlegung  
für  $\alpha$



Für jede KNF-Formel  $\alpha$  gilt:

$\alpha$  unerfüllbar    gdw     $\sqcup \in Res^*(\alpha)$

Anwendung für KNF-SAT-Beweiser:

gegeben:    KNF-Formel  $\alpha$

Methode:    berechne  $Res^*(\alpha)$  und  
              prüfe, ob  $\sqcup \notin Res^*(\alpha)$

$\sqcup = false$     leere Klausel

$Res^*(\alpha)$     Resolutionsabschluss von  $\alpha$



gegeben: KNF-Formel  $\alpha$

Aufgabe: prüfe mittels Resolution, ob  $\alpha$  erfüllbar ist

gegeben: KNF-Formel  $\alpha$

Aufgabe: prüfe mittels Resolution, ob  $\alpha$  erfüllbar ist

```
 $\alpha_0 := \alpha; \quad i := 0;$ 
```

```
REPEAT
```

```
     $\alpha_{i+1} := \alpha_i \cup \text{Res}(\alpha_i); \quad i := i + 1$ 
```

```
UNTIL ...
```

gegeben: KNF-Formel  $\alpha$

Aufgabe: prüfe mittels Resolution, ob  $\alpha$  erfüllbar ist

```
 $\alpha_0 := \alpha; \quad i := 0;$ 
```

```
REPEAT
```

```
     $\alpha_{i+1} := \alpha_i \cup \text{Res}(\alpha_i); \quad i := i + 1$ 
```

```
UNTIL  $\perp \in \alpha_i$  oder ...
```

gegeben: KNF-Formel  $\alpha$

Aufgabe: prüfe mittels Resolution, ob  $\alpha$  erfüllbar ist

$\alpha_0 := \alpha; \quad i := 0;$

REPEAT

$\alpha_{i+1} := \alpha_i \cup \text{Res}(\alpha_i); \quad i := i + 1$

UNTIL  $\perp \in \alpha_i$  oder  $\alpha_{i-1} = \alpha_i$

gegeben: KNF-Formel  $\alpha$

Aufgabe: prüfe mittels Resolution, ob  $\alpha$  erfüllbar ist

```
 $\alpha_0 := \alpha; \quad i := 0;$ 
```

```
REPEAT
```

```
     $\alpha_{i+1} := \alpha_i \cup \text{Res}(\alpha_i); \quad i := i + 1$ 
```

```
UNTIL  $\perp \in \alpha_i$  oder  $\alpha_{i-1} = \alpha_i$ 
```

```
IF  $\perp \in \alpha_i$  THEN gib "nein,  $\alpha$  ist unerfüllbar" aus
```

```
    ELSE ...
```

```
FI
```

gegeben: KNF-Formel  $\alpha$

Aufgabe: prüfe mittels Resolution, ob  $\alpha$  erfüllbar ist

```
 $\alpha_0 := \alpha; \quad i := 0;$ 
```

```
REPEAT
```

```
     $\alpha_{i+1} := \alpha_i \cup \text{Res}(\alpha_i); \quad i := i + 1$ 
```

```
UNTIL  $\perp \in \alpha_i$  oder  $\alpha_{i-1} = \alpha_i$ 
```

```
IF  $\perp \in \alpha_i$  THEN gib "nein,  $\alpha$  ist unerfüllbar" aus
```

```
    ELSE gib "ja,  $\alpha$  ist erfüllbar" aus
```

```
FI
```

gegeben: KNF-Formel  $\alpha$  mit Atomen  $x_1, \dots, x_n$

Aufgabe: prüfe mittels Resolution, ob  $\alpha$  erfüllbar ist

```
 $\alpha_0 := \alpha; \quad i := 0;$ 
```

```
REPEAT
```

```
     $\alpha_{i+1} := \alpha_i \cup \text{Res}(\alpha_i); \quad i := i + 1$ 
```

```
UNTIL  $\perp \in \alpha_i$  oder  $\alpha_{i-1} = \alpha_i$ 
```

Anzahl an Iterationen:  $\exp(n)$  im schlimmsten Fall

gegeben: KNF-Formel  $\alpha$  mit Atomen  $x_1, \dots, x_n$

Aufgabe: prüfe mittels Resolution, ob  $\alpha$  erfüllbar ist

```
 $\alpha_0 := \alpha; \quad i := 0;$ 
```

```
REPEAT
```

```
     $\alpha_{i+1} := \alpha_i \cup \text{Res}(\alpha_i); \quad i := i + 1$ 
```

```
UNTIL  $\perp \in \alpha_i$  oder  $\alpha_{i-1} = \alpha_i$ 
```

Anzahl an Iterationen:  $\exp(n)$  im schlimmsten Fall

Beispiel: Pigeonhole-Formeln ... (ohne Beweis)

gegeben: KNF-Formel  $\alpha$  mit Atomen  $x_1, \dots, x_n$

Aufgabe: prüfe mittels Resolution, ob  $\alpha$  erfüllbar ist

```
 $\alpha_0 := \alpha; \quad i := 0;$ 
```

```
REPEAT
```

```
     $\alpha_{i+1} := \alpha_i \cup \text{Res}(\alpha_i); \quad i := i + 1$ 
```

```
UNTIL  $\perp \in \alpha_i$  oder  $\alpha_{i-1} = \alpha_i$ 
```

Anzahl an Iterationen:  $\exp(n)$  im schlimmsten Fall

aber effizienter für spezielle KNF-Typen

gegeben: KNF-Formel  $\alpha$  mit Atomen  $x_1, \dots, x_n$

Aufgabe: prüfe mittels Resolution, ob  $\alpha$  erfüllbar ist

```
 $\alpha_0 := \alpha; \quad i := 0;$ 
```

```
REPEAT
```

```
     $\alpha_{i+1} := \alpha_i \cup \text{Res}(\alpha_i); \quad i := i + 1$ 
```

```
UNTIL  $\perp \in \alpha_i$  oder  $\alpha_{i-1} = \alpha_i$ 
```

Anzahl an Iterationen:  $\exp(n)$  im schlimmsten Fall

aber effizienter für spezielle KNF-Typen, z.B. **2KNF**

2KNF-Formeln: KNF-Formeln mit höchstens  
**zwei Literalen** pro Klausel

Beispiele:

$$\neg z \wedge (y \vee z) \wedge (x \vee \neg y) \wedge (\neg x \vee \neg y)$$

$\perp$  KNF-Formel bestehend aus der leeren Klausel

**true** KNF-Formel bestehend aus **0** Klausel

2KNF-Formeln: KNF-Formeln mit höchstens zwei Literalen pro Klausel

Alle Resolventen von 2KNF-Formeln sind Klauseln mit **höchstens zwei Literalen**.

Beispiele:

$$\neg z \wedge (y \vee z) \wedge (x \vee \neg y) \wedge (\neg x \vee \neg y)$$

$\perp$  KNF-Formel bestehend aus der leeren Klausel

**true** KNF-Formel bestehend aus **0** Klausel

2KNF-Formeln: KNF-Formeln mit höchstens zwei Literalen pro Klausel

Alle Resolventen von 2KNF-Formeln sind Klauseln mit **höchstens zwei Literalen**.

 $\{\neg z\}$  $\{y, z\}$  $\{x, \neg y\}$  $\{\neg x, \neg y\}$

2KNF-Formeln: KNF-Formeln mit höchstens zwei Literalen pro Klausel

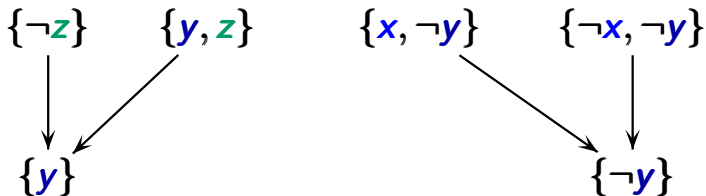
Alle Resolventen von 2KNF-Formeln sind Klauseln mit **höchstens zwei Literalen**.

 $\{\neg z\}$  $\{y, z\}$  $\{x, \neg y\}$  $\{\neg x, \neg y\}$ 

$\{\neg y\}$

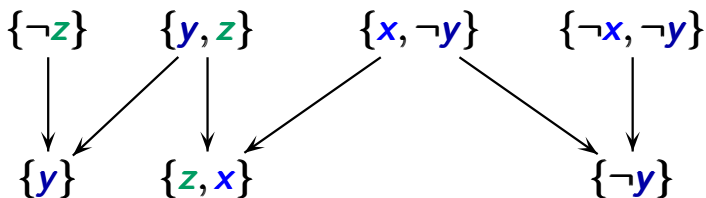
2KNF-Formeln: KNF-Formeln mit höchstens zwei Literalen pro Klausel

Alle Resolventen von 2KNF-Formeln sind Klauseln mit **höchstens zwei Literalen**.



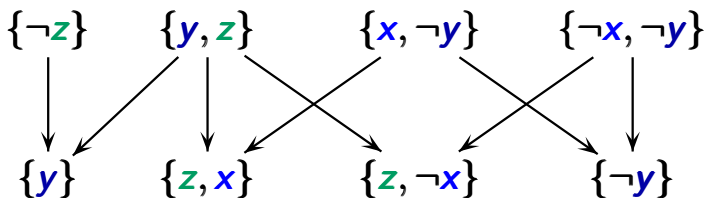
2KNF-Formeln: KNF-Formeln mit höchstens zwei Literalen pro Klausel

Alle Resolventen von 2KNF-Formeln sind Klauseln mit **höchstens zwei Literalen**.



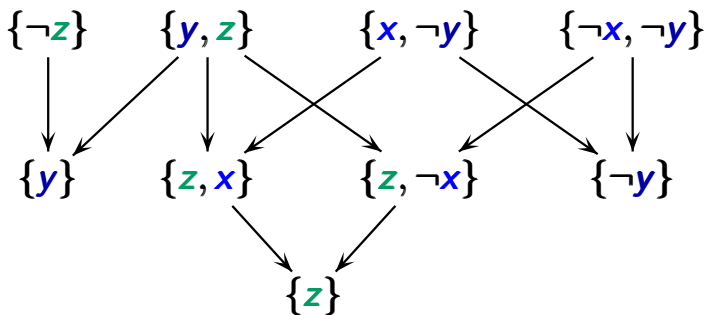
2KNF-Formeln: KNF-Formeln mit höchstens zwei Literalen pro Klausel

Alle Resolventen von 2KNF-Formeln sind Klauseln mit **höchstens zwei Literalen**.



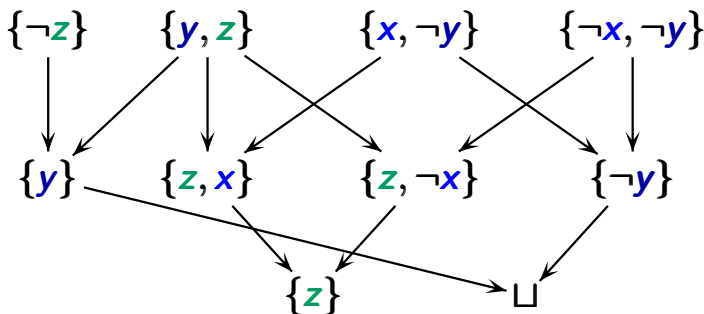
2KNF-Formeln: KNF-Formeln mit höchstens zwei Literalen pro Klausel

Alle Resolventen von 2KNF-Formeln sind Klauseln mit **höchstens zwei Literalen**.



2KNF-Formeln: KNF-Formeln mit höchstens zwei Literalen pro Klausel

Alle Resolventen von 2KNF-Formeln sind Klauseln mit **höchstens zwei Literalen**.



2KNF-Formeln: KNF-Formeln mit höchstens zwei Literalen pro Klausel

Alle Resolventen von 2KNF-Formeln sind Klauseln mit höchstens zwei Literalen.

Sei  $\alpha$  eine 2KNF-Formel mit den Atomen  $x_1, \dots, x_n$ .

Was ist die maximale Anzahl an Resolventen von  $\alpha$  ?

2KNF-Formeln: KNF-Formeln mit höchstens zwei Literalen pro Klausel

Alle Resolventen von 2KNF-Formeln sind Klauseln mit höchstens zwei Literalen.

Sei  $\alpha$  eine 2KNF-Formel mit den Atomen  $x_1, \dots, x_n$ .

Was ist die maximale Anzahl an Resolventen von  $\alpha$  ?

Antwort:  $\mathcal{O}(n^2)$

2KNF-Formeln: KNF-Formeln mit höchstens zwei Literalen pro Klausel

Alle Resolventen von 2KNF-Formeln sind Klauseln mit höchstens zwei Literalen.

Sei  $\alpha$  eine 2KNF-Formel mit den Atomen  $x_1, \dots, x_n$ .

Was ist die maximale Anzahl an Resolventen von  $\alpha$  ?

Antwort:  $\mathcal{O}(n^2)$ , genauer  $2n(2n-1) + 2n + 1$

2KNF-Formeln: KNF-Formeln mit höchstens zwei Literalen pro Klausel

Alle Resolventen von 2KNF-Formeln sind Klauseln mit höchstens zwei Literalen.

Sei  $\alpha$  eine 2KNF-Formel mit den Atomen  $x_1, \dots, x_n$ .

Was ist die maximale Anzahl an Resolventen von  $\alpha$  ?

Antwort:  $\mathcal{O}(n^2)$ , genauer  $2n(2n-1) + 2n + 1$

↑  
leere Klausel

2KNF-Formeln: KNF-Formeln mit höchstens zwei Literalen pro Klausel

Alle Resolventen von 2KNF-Formeln sind Klauseln mit höchstens zwei Literalen.

Sei  $\alpha$  eine 2KNF-Formel mit den Atomen  $x_1, \dots, x_n$ .

Was ist die maximale Anzahl an Resolventen von  $\alpha$  ?

Antwort:  $\mathcal{O}(n^2)$ , genauer  $2n(2n-1) + 2n + 1$

↑  
Einheitsklausel  
 $x_j$  oder  $\neg x_j$

2KNF-Formeln: KNF-Formeln mit höchstens zwei Literalen pro Klausel

Alle Resolventen von 2KNF-Formeln sind Klauseln mit höchstens zwei Literalen.

Sei  $\alpha$  eine 2KNF-Formel mit den Atomen  $x_1, \dots, x_n$ .

Was ist die maximale Anzahl an Resolventen von  $\alpha$  ?

Antwort:  $\mathcal{O}(n^2)$ , genauer  $2n(2n-1) + 2n + 1$

↑

Klauseln  $L_1 \vee L_2$ , wobei  $L_1 \neq L_2$  und  
 $L_1, L_2 \in \{x_1, \dots, x_n, \neg x_1, \dots, \neg x_n\}$

2KNF-Formeln: KNF-Formeln mit höchstens zwei Literalen pro Klausel

Alle Resolventen von 2KNF-Formeln sind Klauseln mit höchstens zwei Literalen.

Sei  $\alpha$  eine 2KNF-Formel mit den Atomen  $x_1, \dots, x_n$ .

Was ist die maximale Anzahl an Resolventen von  $\alpha$  ?

Antwort:  $\mathcal{O}(n^2)$ , genauer  $2n(2n-1) + 2n + 1$

---

Anzahl der Iterationen des Resolutionsalgorithmus ist für 2KNF-Formeln durch  $\mathcal{O}(n^2)$  beschränkt

3KNF-Formeln: KNF-Formeln mit höchstens  
drei Literalen pro Klausel

3KNF-Formeln: KNF-Formeln mit höchstens drei Literalen pro Klausel

Ist die Anzahl an Iterationen für 3KNF-Formeln durch  $\mathcal{O}(n^3)$  beschränkt ?

3KNF-Formeln: KNF-Formeln mit höchstens drei Literalen pro Klausel

Ist die Anzahl an Iterationen für 3KNF-Formeln durch  $\mathcal{O}(n^3)$  beschränkt ?

Antwort: nein

3KNF-Formeln: KNF-Formeln mit höchstens drei Literalen pro Klausel

Ist die Anzahl an Iterationen für 3KNF-Formeln durch  $\mathcal{O}(n^3)$  beschränkt ?

Antwort: nein, denn Resolventen von 3KNF-Formeln können vier oder mehr Literale haben

3KNF-Formeln: KNF-Formeln mit höchstens drei Literalen pro Klausel

Ist die Anzahl an Iterationen für 3KNF-Formeln durch  $\mathcal{O}(n^3)$  beschränkt ?

Antwort: nein, denn Resolventen von 3KNF-Formeln können vier oder mehr Literale haben

$\{x, y, z\}$

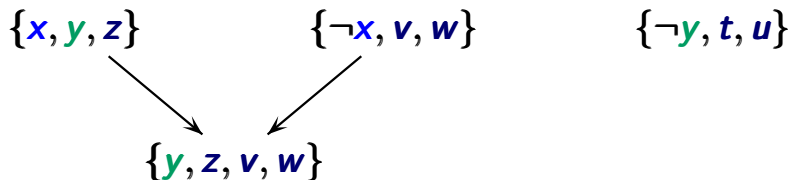
$\{\neg x, v, w\}$

$\{\neg y, t, u\}$

3KNF-Formeln: KNF-Formeln mit höchstens drei Literalen pro Klausel

Ist die Anzahl an Iterationen für 3KNF-Formeln durch  $\mathcal{O}(n^3)$  beschränkt ?

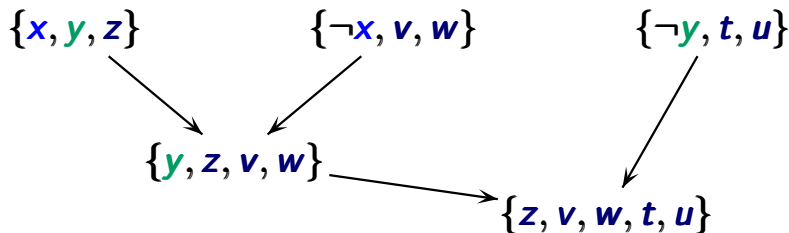
Antwort: nein, denn Resolventen von 3KNF-Formeln können vier oder mehr Literale haben



3KNF-Formeln: KNF-Formeln mit höchstens drei Literalen pro Klausel

Ist die Anzahl an Iterationen für 3KNF-Formeln durch  $\mathcal{O}(n^3)$  beschränkt ?

Antwort: nein, denn Resolventen von 3KNF-Formeln können vier oder mehr Literale haben





Für jede KNF-Formel  $\alpha$  gilt:

$\alpha$  unerfüllbar    gdw     $\sqcup \in Res^*(\alpha)$

Folgerung: Resolution ist korrekt und vollständig  
als **Widerlegungskalkül**

$\sqcup = \textit{false}$     leere Klausel

$Res^*(\alpha)$     Resolutionsabschluss von  $\alpha$

Für jede KNF-Formel  $\alpha$  gilt:

$\alpha$  unerfüllbar    gdw     $\sqcup \in Res^*(\alpha)$

Folgerung: Resolution ist korrekt und vollständig  
als **Widerlegungskalkül**

aber: Resolution als Kalkül für **logische Folgerbarkeit**  
ist zwar korrekt, aber unvollständig

$\sqcup = \textit{false}$     leere Klausel

$Res^*(\alpha)$     Resolutionsabschluss von  $\alpha$

Für jede KNF-Formel  $\alpha$  gilt:

$\alpha$  unerfüllbar    gdw     $\perp \in Res^*(\alpha)$

Folgerung: Resolution ist korrekt und vollständig  
als Widerlegungskalkül

aber: Resolution als Kalkül für logische Folgerbarkeit  
ist zwar korrekt, aber unvollständig



Resolventenlemma:

$\alpha \Vdash \lambda$  für jede Klausel  $\lambda \in Res^*(\alpha)$

Für jede KNF-Formel  $\alpha$  gilt:

$\alpha$  unerfüllbar    gdw     $\perp \in Res^*(\alpha)$

Folgerung: Resolution ist korrekt und vollständig  
als Widerlegungskalkül

aber: Resolution als Kalkül für logische Folgerbarkeit  
ist zwar korrekt, aber unvollständig

$\alpha \Vdash \lambda$  und  $\lambda \notin Res^*(\alpha)$  ist möglich

Für jede KNF-Formel  $\alpha$  gilt:

$\alpha$  unerfüllbar    gdw     $\sqcup \in Res^*(\alpha)$

Folgerung: Resolution ist korrekt und vollständig  
als Widerlegungskalkül

aber: Resolution als Kalkül für logische Folgerbarkeit  
ist zwar korrekt, aber **unvollständig**

Beispiel:  $\alpha = y \wedge \neg y \wedge x$  ist unerfüllbar.

Für jede KNF-Formel  $\alpha$  gilt:

$\alpha$  unerfüllbar    gdw     $\sqcup \in \text{Res}^*(\alpha)$

Folgerung: Resolution ist korrekt und vollständig  
als Widerlegungskalkül

aber: Resolution als Kalkül für logische Folgerbarkeit  
ist zwar korrekt, aber **unvollständig**

Beispiel:  $\alpha = y \wedge \neg y \wedge x$  ist unerfüllbar.

Also gilt  $\alpha \Vdash \neg x$

Da  $\alpha$  unerfüllbar ist, gilt  
 $\alpha \Vdash \beta$  für jede Formel  $\beta$ .

Für jede KNF-Formel  $\alpha$  gilt:

$\alpha$  unerfüllbar    gdw     $\perp \in Res^*(\alpha)$

Folgerung: Resolution ist korrekt und vollständig  
als Widerlegungskalkül

aber: Resolution als Kalkül für logische Folgerbarkeit  
ist zwar korrekt, aber **unvollständig**

Beispiel:  $\alpha = y \wedge \neg y \wedge x$  ist unerfüllbar.

Also gilt  $\alpha \models \neg x$ , aber  $\{\neg x\} \notin Res^*(\alpha) = \alpha \cup \{\perp\}$

Für jede KNF-Formel  $\alpha$  gilt:

$\alpha$  unerfüllbar    gdw     $\sqcup \in \text{Res}^*(\alpha)$

Folgerung: Resolution ist korrekt und vollständig  
als Widerlegungskalkül

aber: Resolution als Kalkül für logische Folgerbarkeit  
ist zwar korrekt, aber unvollständig

dennoch: Resolution einsetzbar für **Folgerungstest**

gegeben: KNF-Formel  $\alpha$ , Klausel  $\lambda$

Aufgabe: prüfe mittels Resolution, ob  $\alpha \Vdash \lambda$

gegeben: KNF-Formel  $\alpha$ , Klausel  $\lambda$

Aufgabe: prüfe mittels Resolution, ob  $\alpha \Vdash \lambda$

Sei  $\lambda = L_1 \vee L_2 \vee \dots \vee L_k$ , wobei  $L_1, \dots, L_k$  Literale.

gegeben: KNF-Formel  $\alpha$ , Klausel  $\lambda$

Aufgabe: prüfe mittels Resolution, ob  $\alpha \models \lambda$

Sei  $\lambda = L_1 \vee L_2 \vee \dots \vee L_k$ , wobei  $L_1, \dots, L_k$  Literale.

$\alpha \models \lambda$  gdw  $\alpha \wedge \neg \lambda$  ist unerfüllbar

gegeben: KNF-Formel  $\alpha$ , Klausel  $\lambda$

Aufgabe: prüfe mittels Resolution, ob  $\alpha \models \lambda$

Sei  $\lambda = L_1 \vee L_2 \vee \dots \vee L_k$ , wobei  $L_1, \dots, L_k$  Literale.

$\alpha \models \lambda$  gdw  $\alpha \wedge \neg \lambda$  ist unerfüllbar

gdw  $\alpha \wedge \overline{L_1} \wedge \overline{L_2} \wedge \dots \wedge \overline{L_k}$  unerfüllbar

$\overline{L} = \begin{cases} \neg x & : \text{ falls } L = x \\ x & : \text{ falls } L = \neg x \end{cases}$  komplementäres Literal

gegeben: KNF-Formel  $\alpha$ , Klausel  $\lambda$

Aufgabe: prüfe mittels Resolution, ob  $\alpha \models \lambda$

Sei  $\lambda = L_1 \vee L_2 \vee \dots \vee L_k$ , wobei  $L_1, \dots, L_k$  Literale.

$\alpha \models \lambda$  gdw  $\alpha \wedge \neg \lambda$  ist unerfüllbar

gdw  $\alpha \wedge \overline{L_1} \wedge \overline{L_2} \wedge \dots \wedge \overline{L_k}$  unerfüllbar

gdw  $\sqcup \in \text{Res}^*(\alpha \wedge \overline{L_1} \wedge \dots \wedge \overline{L_k})$

$\overline{L} = \begin{cases} \neg x & : \text{ falls } L = x \\ x & : \text{ falls } L = \neg x \end{cases}$  komplementäres Literal

Für jede nicht-tautologische KNF-Formel  $\alpha$  und jede nicht-tautologische Klausel  $\lambda$  gilt:

$$\alpha \Vdash \lambda \quad \text{gdw} \quad \dots$$

Für jede nicht-tautologische KNF-Formel  $\alpha$  und jede nicht-tautologische Klausel  $\lambda$  gilt:

$$\alpha \Vdash \lambda \quad \text{gdw} \quad \left\{ \begin{array}{l} \text{es gibt eine Klausel } \lambda' \in \text{Res}^*(\alpha) \\ \text{mit } \lambda' \subseteq \lambda \end{array} \right.$$

(ohne Beweis)

Für jede nicht-tautologische KNF-Formel  $\alpha$  und jede nicht-tautologische Klausel  $\lambda$  gilt:

$$\alpha \Vdash \lambda \quad \text{gdw} \quad \left\{ \begin{array}{l} \text{es gibt eine Klausel } \lambda' \in \text{Res}^*(\alpha) \\ \text{mit } \lambda' \subseteq \lambda \end{array} \right. \quad (\text{ohne Beweis})$$

Die Aussage des Resolutionssatzes folgt mit  $\lambda = \sqcup$ :

$\sqcup =$  leere Klausel = leere Literalmenge

Für jede nicht-tautologische KNF-Formel  $\alpha$  und jede nicht-tautologische Klausel  $\lambda$  gilt:

$$\alpha \Vdash \lambda \quad \text{gdw} \quad \left\{ \begin{array}{l} \text{es gibt eine Klausel } \lambda' \in \text{Res}^*(\alpha) \\ \text{mit } \lambda' \subseteq \lambda \end{array} \right.$$

(ohne Beweis)

Die Aussage des Resolutionssatzes folgt mit  $\lambda = \perp$ :

$$\alpha \text{ ist unerfüllbar} \quad \text{gdw} \quad \alpha \Vdash \text{false} = \perp$$

$\perp$  = leere Klausel = leere Literalmenge

Für jede nicht-tautologische KNF-Formel  $\alpha$  und jede nicht-tautologische Klausel  $\lambda$  gilt:

$$\alpha \Vdash \lambda \quad \text{gdw} \quad \left\{ \begin{array}{l} \text{es gibt eine Klausel } \lambda' \in \text{Res}^*(\alpha) \\ \text{mit } \lambda' \subseteq \lambda \end{array} \right.$$

(ohne Beweis)

Die Aussage des Resolutionssatzes folgt mit  $\lambda = \sqcup$ :

$$\begin{aligned} \alpha \text{ ist unerfüllbar} & \quad \text{gdw} \quad \alpha \Vdash \text{false} = \sqcup \\ & \quad \text{gdw} \quad \sqcup \in \text{Res}^*(\alpha) \end{aligned}$$

$\sqcup =$  leere Klausel  $=$  leere Literalmenge

Für jede nicht-tautologische KNF-Formel  $\alpha$  und jede nicht-tautologische Klausel  $\lambda$  gilt:

$$\alpha \Vdash \lambda \quad \text{gdw} \quad \left\{ \begin{array}{l} \text{es gibt eine Klausel } \lambda' \in \text{Res}^*(\alpha) \\ \text{mit } \lambda' \subseteq \lambda \end{array} \right. \quad (\text{ohne Beweis})$$

Die Aussage des Resolutionsatzes folgt mit  $\lambda = \perp$ :

$$\begin{aligned} \alpha \text{ ist unerfüllbar} & \quad \text{gdw} \quad \alpha \Vdash \text{false} = \perp \\ & \quad \text{gdw} \quad \perp \in \text{Res}^*(\alpha) \end{aligned}$$

... falsch für tautologische KNF-Formeln oder Klauseln

Für jede nicht-tautologische KNF-Formel  $\alpha$  und jede nicht-tautologische Klausel  $\lambda$  gilt:

$$\alpha \Vdash \lambda \quad \text{gdw} \quad \left\{ \begin{array}{l} \text{es gibt eine Klausel } \lambda' \in \text{Res}^*(\alpha) \\ \text{mit } \lambda' \subseteq \lambda \end{array} \right.$$

(ohne Beweis)

falsch für tautologische KNF-Formeln:

$$\alpha = \text{true} = \bigwedge_{i \in \emptyset} \dots \hat{=} \text{leere Klauselmeng}$$

Für jede nicht-tautologische KNF-Formel  $\alpha$  und jede nicht-tautologische Klausel  $\lambda$  gilt:

$$\alpha \Vdash \lambda \quad \text{gdw} \quad \left\{ \begin{array}{l} \text{es gibt eine Klausel } \lambda' \in \text{Res}^*(\alpha) \\ \text{mit } \lambda' \subseteq \lambda \end{array} \right. \quad (\text{ohne Beweis})$$

falsch für tautologische KNF-Formeln:

$$\alpha = \text{true} = \bigwedge_{i \in \emptyset} \dots \hat{=} \text{leere Klauselmenge}$$

$$\text{Res}^*(\alpha) = \emptyset,$$

Für jede nicht-tautologische KNF-Formel  $\alpha$  und jede nicht-tautologische Klausel  $\lambda$  gilt:

$$\alpha \Vdash \lambda \quad \text{gdw} \quad \left\{ \begin{array}{l} \text{es gibt eine Klausel } \lambda' \in \text{Res}^*(\alpha) \\ \text{mit } \lambda' \subseteq \lambda \end{array} \right.$$

(ohne Beweis)

falsch für tautologische KNF-Formeln:

$$\alpha = \text{true} = \bigwedge_{i \in \emptyset} \dots \hat{=} \text{leere Klauselmeng}$$

$$\text{Res}^*(\alpha) = \emptyset, \quad \text{aber z.B. } \alpha \Vdash \lambda \text{ für } \lambda = \underbrace{x \vee \neg x}_{\text{gültig}}$$

Für jede nicht-tautologische KNF-Formel  $\alpha$  und jede nicht-tautologische Klausel  $\lambda$  gilt:

$$\alpha \Vdash \lambda \quad \text{gdw} \quad \left\{ \begin{array}{l} \text{es gibt eine Klausel } \lambda' \in \text{Res}^*(\alpha) \\ \text{mit } \lambda' \subseteq \lambda \end{array} \right.$$

(ohne Beweis)

falsch für tautologische Klauseln

Für jede nicht-tautologische KNF-Formel  $\alpha$  und jede nicht-tautologische Klausel  $\lambda$  gilt:

$$\alpha \Vdash \lambda \quad \text{gdw} \quad \left\{ \begin{array}{l} \text{es gibt eine Klausel } \lambda' \in \text{Res}^*(\alpha) \\ \text{mit } \lambda' \subseteq \lambda \end{array} \right. \quad (\text{ohne Beweis})$$

falsch für tautologische Klauseln, z.B.

$\alpha = (x \vee \neg y) \wedge (x \vee \neg z)$  erfüllbar, nicht gültig

$\text{Res}^*(\alpha) = \alpha$ , aber z.B.  $\alpha \Vdash \lambda$  für  $\lambda = x \vee \neg x$

↑  
tautologische Klausel

Für jede nicht-tautologische KNF-Formel  $\alpha$  und jede nicht-tautologische Klausel  $\lambda$  gilt:

$$\alpha \Vdash \lambda \quad \text{gdw} \quad \left\{ \begin{array}{l} \text{es gibt eine Klausel } \lambda' \in \text{Res}^*(\alpha) \\ \text{mit } \lambda' \subseteq \lambda \end{array} \right. \quad (\text{ohne Beweis})$$

Spezialfall: Sei  $\alpha$  erfüllbar und  $\lambda$  eine Einheitsklausel, d.h.,  $\lambda = L$  für ein Literal  $L$ .

Für jede nicht-tautologische KNF-Formel  $\alpha$  und jede nicht-tautologische Klausel  $\lambda$  gilt:

$$\alpha \Vdash \lambda \quad \text{gdw} \quad \left\{ \begin{array}{l} \text{es gibt eine Klausel } \lambda' \in \text{Res}^*(\alpha) \\ \text{mit } \lambda' \subseteq \lambda \end{array} \right. \quad (\text{ohne Beweis})$$

Spezialfall: Sei  $\alpha$  erfüllbar und  $\lambda$  eine Einheitsklausel, d.h.,  $\lambda = L$  für ein Literal  $L$ . Dann gilt:

$$\alpha \Vdash L \quad \text{gdw} \quad L \in \text{Res}^*(\alpha)$$



gegeben: KNF-Formel  $\alpha$ , Klausel  $\lambda$

Aufgabe: prüfe mittels Resolution, ob  $\alpha \models \lambda$

gegeben: KNF-Formel  $\alpha$ , Klausel  $\lambda$

Aufgabe: prüfe mittels Resolution, ob  $\alpha \models \lambda$

$\alpha_0 := \alpha; \quad i := 0;$

REPEAT

$\alpha_{i+1} := \alpha_i \cup \text{Res}(\alpha_i); \quad i := i + 1$

gegeben: KNF-Formel  $\alpha$ , Klausel  $\lambda$

Aufgabe: prüfe mittels Resolution, ob  $\alpha \models \lambda$

$\alpha_0 := \alpha; \quad i := 0;$

REPEAT

$\alpha_{i+1} := \alpha_i \cup \text{Res}(\alpha_i); \quad i := i + 1$

UNTIL es gibt eine Klausel  $\lambda' \in \alpha_i$  mit  $\lambda' \subseteq \lambda$   
oder ...

gegeben: KNF-Formel  $\alpha$ , Klausel  $\lambda$

Aufgabe: prüfe mittels Resolution, ob  $\alpha \Vdash \lambda$

$\alpha_0 := \alpha; \quad i := 0;$

REPEAT

$\alpha_{i+1} := \alpha_i \cup \text{Res}(\alpha_i); \quad i := i + 1$

UNTIL es gibt eine Klausel  $\lambda' \in \alpha_i$  mit  $\lambda' \subseteq \lambda$   
oder  $\alpha_{i-1} = \alpha_i$

gegeben: KNF-Formel  $\alpha$ , Klausel  $\lambda$

Aufgabe: prüfe mittels Resolution, ob  $\alpha \models \lambda$

```
 $\alpha_0 := \alpha; \quad i := 0;$ 
```

```
REPEAT
```

```
     $\alpha_{i+1} := \alpha_i \cup \text{Res}(\alpha_i); \quad i := i + 1$ 
```

```
UNTIL es gibt eine Klausel  $\lambda' \in \alpha_i$  mit  $\lambda' \subseteq \lambda$   
      oder  $\alpha_{i-1} = \alpha_i$ 
```

```
IF es gibt eine Klausel  $\lambda' \in \alpha_i$  mit  $\lambda' \subseteq \lambda$ 
```

```
    THEN gib "ja" aus
```

```
    ELSE gib "nein" aus
```

```
FI
```



$$\alpha = (x \vee y \vee \neg z) \wedge (\neg x \vee y \vee w) \wedge (z \vee w) \wedge v$$

Gilt  $\alpha \Vdash \lambda$ , wobei  $\lambda = y \vee w \vee \neg v$  ?

$$\alpha = (x \vee y \vee \neg z) \wedge (\neg x \vee y \vee w) \wedge (z \vee w) \wedge v$$

Gilt  $\alpha \models \lambda$ , wobei  $\lambda = y \vee w \vee \neg v$ ?

$\{x, y, \neg z\}$

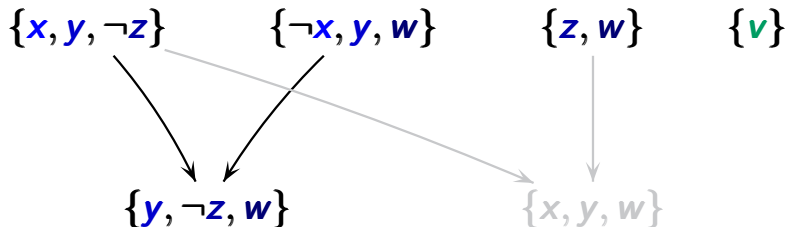
$\{\neg x, y, w\}$

$\{z, w\}$

$\{v\}$

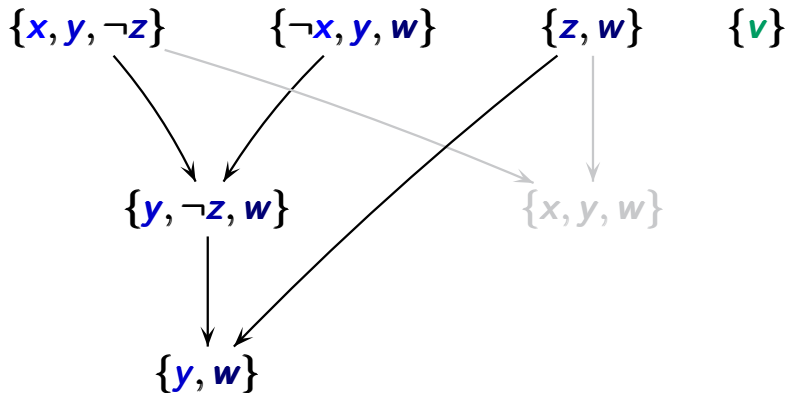
$$\alpha = (x \vee y \vee \neg z) \wedge (\neg x \vee y \vee w) \wedge (z \vee w) \wedge v$$

Gilt  $\alpha \models \lambda$ , wobei  $\lambda = y \vee w \vee \neg v$ ?



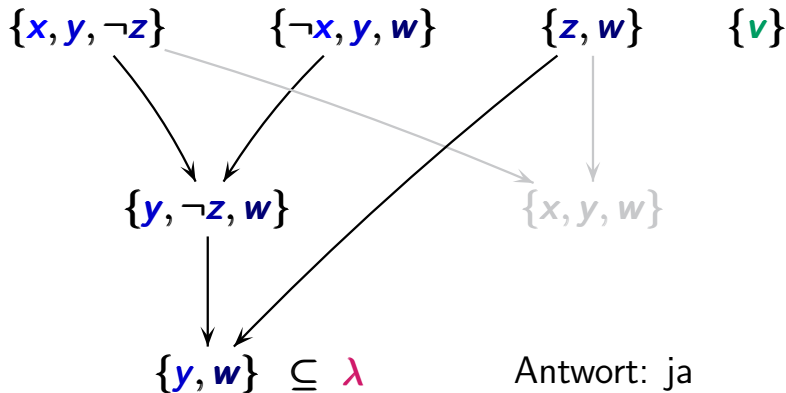
$$\alpha = (x \vee y \vee \neg z) \wedge (\neg x \vee y \vee w) \wedge (z \vee w) \wedge v$$

Gilt  $\alpha \models \lambda$ , wobei  $\lambda = y \vee w \vee \neg v$ ?



$$\alpha = (x \vee y \vee \neg z) \wedge (\neg x \vee y \vee w) \wedge (z \vee w) \wedge v$$

Gilt  $\alpha \models \lambda$ , wobei  $\lambda = y \vee w \vee \neg v$ ?





angewandt auf eine KNF-Formel  $\alpha$  über  $x_1, \dots, x_n$

Erfüllbarkeitstest: gilt  $\perp \notin Res^*(\alpha)$  ?

Folgerungstest: gilt  $\alpha \Vdash \lambda$  für gegebene Klausel  $\lambda$  ?

Laufzeit:

- $\exp(n)$  im schlimmsten Fall
- polynomielle Zeitbeschränkung nur für spezielle KNF-Typen, z.B. 2KNF

angewandt auf eine KNF-Formel  $\alpha$  über  $x_1, \dots, x_n$

Erfüllbarkeitstest: gilt  $\perp \notin Res^*(\alpha)$  ?

Folgerungstest: gilt  $\alpha \Vdash \lambda$  für gegebene Klausel  $\lambda$  ?

Laufzeit:

- $\exp(n)$  im schlimmsten Fall
- polynomielle Zeitbeschränkung nur für spezielle KNF-Typen, z.B. 2KNF

zur Steigerung der Effizienz: **Resolutionsstrategien**

angewandt auf eine KNF-Formel  $\alpha$  über  $x_1, \dots, x_n$

Erfüllbarkeitstest: gilt  $\perp \notin Res^*(\alpha)$  ?

Folgerungstest: gilt  $\alpha \Vdash \lambda$  für gegebene Klausel  $\lambda$  ?

Laufzeit:

- $\exp(n)$  im schlimmsten Fall
- polynomielle Zeitbeschränkung nur für spezielle KNF-Typen, z.B. 2KNF

zur Steigerung der Effizienz: **Resolutionsstrategien**, z.B.

syntaktische Restriktionen für Resolventenbildung,  
um die Anzahl an möglichen Resolventen einzuschränken

angewandt auf eine KNF-Formel  $\alpha$  über  $x_1, \dots, x_n$

Erfüllbarkeitstest: gilt  $\perp \notin Res^*(\alpha)$  ?

Folgerungstest: gilt  $\alpha \Vdash \lambda$  für gegebene Klausel  $\lambda$  ?

Laufzeit:

- $\exp(n)$  im schlimmsten Fall
- polynomielle Zeitbeschränkung nur für spezielle KNF-Typen, z.B. 2KNF

zur Steigerung der Effizienz: **Resolutionsstrategien**, z.B.

N-Resolution	lineare Resolution	u.a.
P-Resolution	Input-Resolution	

angewandt auf eine KNF-Formel  $\alpha$  über  $x_1, \dots, x_n$

Erfüllbarkeitstest: gilt  $\perp \notin Res^*(\alpha)$  ?

Folgerungstest: gilt  $\alpha \Vdash \lambda$  für gegebene Klausel  $\lambda$  ?

Laufzeit:

- $\exp(n)$  im schlimmsten Fall
- polynomielle Zeitbeschränkung nur für spezielle KNF-Typen, z.B. 2KNF

zur Steigerung der Effizienz: **Resolutionsstrategien**, z.B.

N-Resolution

lineare Resolution

u.a.

P-Resolution

Input-Resolution

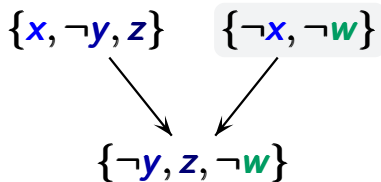


N-Resolutionsschritt:

wie gewöhnlicher Resolutionsschritt, aber eine der Elternklauseln muss aus **negativen Literalen** bestehen

N-Resolutionsschritt:

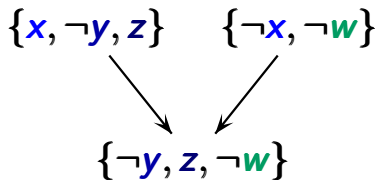
wie gewöhnlicher Resolutionsschritt, aber eine der Elternklauseln muss aus **negativen Literalen** bestehen



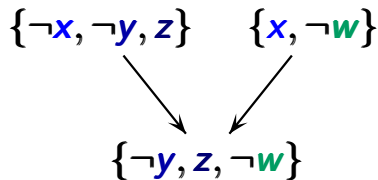
N-Resolutionsschritt

N-Resolutionsschritt:

wie gewöhnlicher Resolutionsschritt, aber eine der Elternklauseln muss aus **negativen Literalen** bestehen



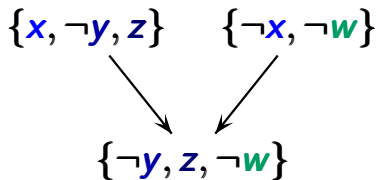
N-Resolutionsschritt



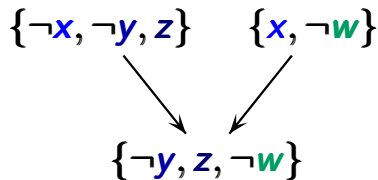
kein N-Resolutionsschritt

N-Resolutionsschritt:

wie gewöhnlicher Resolutionsschritt, aber eine der Elternklauseln muss aus negativen Literalen bestehen



N-Resolutionsschritt



kein N-Resolutionsschritt

N-Herleitung: Herleitung  $\langle \kappa_1, \tau_1, \lambda_1 \rangle \dots \langle \kappa_m, \tau_m, \lambda_m \rangle$ ,  
so dass  $\kappa_1, \dots, \kappa_m$  nur aus negativen  
Literalen bestehen

N-Widerlegung: N-Herleitung der leeren Klausel

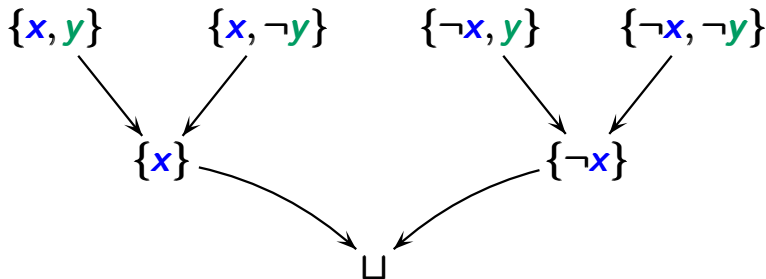
$$\alpha = (x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

unerfüllbare KNF-Formel

N-Widerlegung: N-Herleitung der leeren Klausel

$$\alpha = (x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

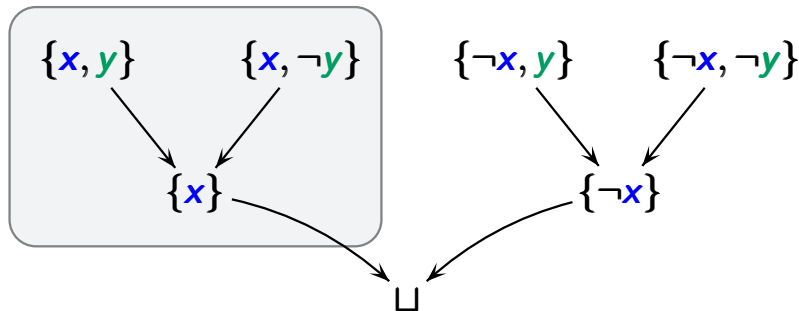
unerfüllbare KNF-Formel



Widerlegung, aber keine N-Widerlegung

$$\alpha = (x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

unerfüllbare KNF-Formel



Widerlegung, aber keine N-Widerlegung

$$\alpha = (x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

unerfüllbare KNF-Formel

$\{x, y\}$

$\{x, \neg y\}$

$\{\neg x, y\}$

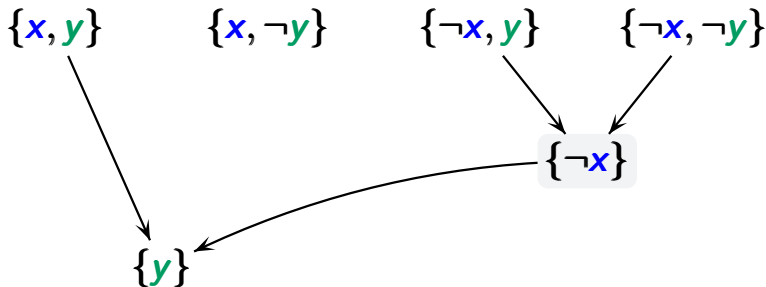
$\{\neg x, \neg y\}$

$\{\neg x\}$



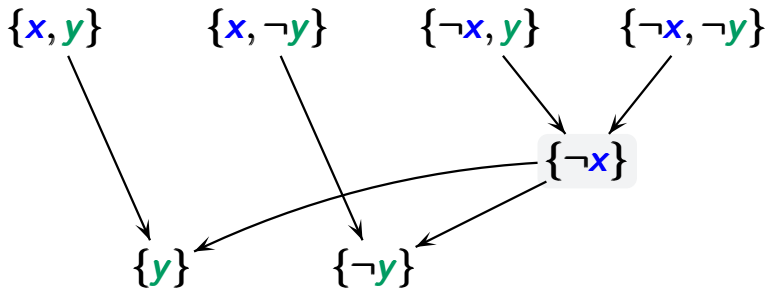
$$\alpha = (x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

unerfüllbare KNF-Formel



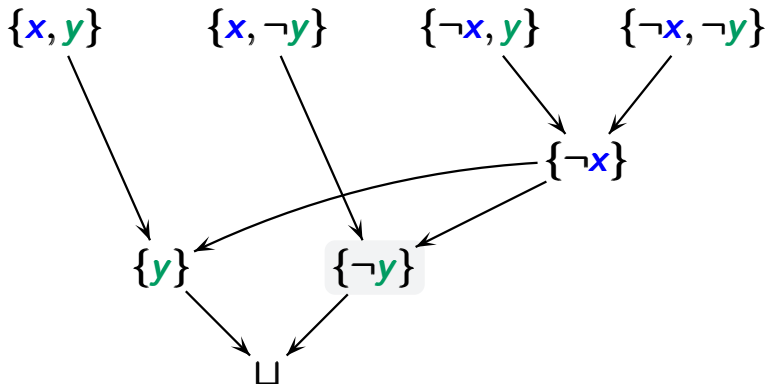
$$\alpha = (x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

unerfüllbare KNF-Formel



$$\alpha = (x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

unerfüllbare KNF-Formel





Sei  $\alpha$  eine KNF-Formel.

$$Res^*(\alpha) = \bigcup_{i \geq 0} Res^i(\alpha)$$

$$NRes^*(\alpha) = \bigcup_{i \geq 0} NRes^i(\alpha)$$

Sei  $\alpha$  eine KNF-Formel.

$$Res^*(\alpha) = \bigcup_{i \geq 0} Res^i(\alpha)$$

Menge aller Klauseln,  
für die es eine  
**Herleitung** aus  $\alpha$  gibt

$$NRes^*(\alpha) = \bigcup_{i \geq 0} NRes^i(\alpha)$$

Menge aller Klauseln,  
für die es eine  
**N-Herleitung** aus  $\alpha$  gibt

Sei  $\alpha$  eine KNF-Formel.

$$Res^*(\alpha) = \bigcup_{i \geq 0} Res^i(\alpha) \quad NRes^*(\alpha) = \bigcup_{i \geq 0} NRes^i(\alpha)$$

Es gilt stets:  $\alpha \subseteq NRes^*(\alpha) \subseteq Res^*(\alpha)$

↑  
alle N-Resolventen sind Resolventen

Sei  $\alpha$  eine KNF-Formel.

$$Res^*(\alpha) = \bigcup_{i \geq 0} Res^i(\alpha) \quad NRes^*(\alpha) = \bigcup_{i \geq 0} NRes^i(\alpha)$$

Es gilt stets:  $\alpha \subseteq NRes^*(\alpha) \subseteq Res^*(\alpha)$

Folgerung: Korrektheit der N-Resolution

$$\alpha \Vdash \lambda \quad \text{für alle } \lambda \in NRes^*(\alpha)$$

Sei  $\alpha$  eine KNF-Formel.

$$Res^*(\alpha) = \bigcup_{i \geq 0} Res^i(\alpha) \quad NRes^*(\alpha) = \bigcup_{i \geq 0} NRes^i(\alpha)$$

Es gilt stets:  $\alpha \subseteq NRes^*(\alpha) \subseteq Res^*(\alpha)$

Folgerung: Korrektheit der N-Resolution

$$\alpha \Vdash \lambda \quad \text{für alle } \lambda \in NRes^*(\alpha)$$

Insbesondere: Ist  $\perp \in NRes^*(\alpha)$ , so ist  $\alpha$  unerfüllbar.

Sei  $\alpha$  eine KNF-Formel.

$$Res^*(\alpha) = \bigcup_{i \geq 0} Res^i(\alpha) \quad NRes^*(\alpha) = \bigcup_{i \geq 0} NRes^i(\alpha)$$

Es gilt stets:  $\alpha \subseteq NRes^*(\alpha) \subseteq Res^*(\alpha)$

Folgerung: Korrektheit der N-Resolution

$$\alpha \Vdash \lambda \quad \text{für alle } \lambda \in NRes^*(\alpha)$$

Insbesondere: Ist  $\perp \in NRes^*(\alpha)$ , so ist  $\alpha$  unerfüllbar.

Hauptsatz zur N-Resolution: (ohne Beweis)

$$\alpha \text{ unerfüllbar} \quad \text{gdw} \quad \perp \in NRes^*(\alpha)$$

$$\alpha = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (x \vee \neg y \vee \neg w) \wedge \\ (x \vee \neg z \vee w) \wedge (\neg z \vee \neg w)$$

$$\alpha = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (x \vee \neg y \vee \neg w) \wedge \\ (x \vee \neg z \vee w) \wedge (\neg z \vee \neg w)$$

viele Resolventen, aber nur **zwei N-Resolventen**

$$\alpha = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (x \vee \neg y \vee \neg w) \wedge \\ (x \vee \neg z \vee w) \wedge (\neg z \vee \neg w)$$

↑  
einzige Klausel bestehend  
aus negativen Literalen

viele Resolventen, aber nur **zwei N-Resolventen**

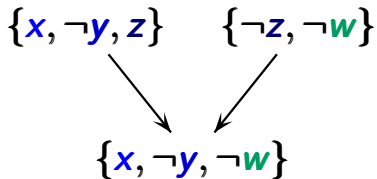
$$\alpha = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (x \vee \neg y \vee \neg w) \wedge \\ (x \vee \neg z \vee w) \wedge (\neg z \vee \neg w)$$

↑  
einzige Klausel bestehend  
aus negativen Literalen

viele Resolventen, aber nur zwei N-Resolventen

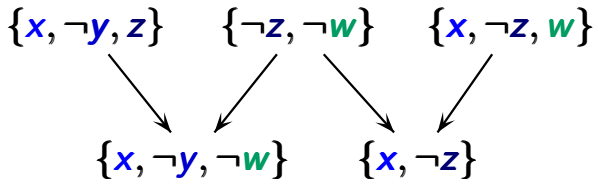
$$\alpha = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (x \vee \neg y \vee \neg w) \wedge (x \vee \neg z \vee w) \wedge (\neg z \vee \neg w)$$

↑  
einzige Klausel bestehend  
aus negativen Literalen



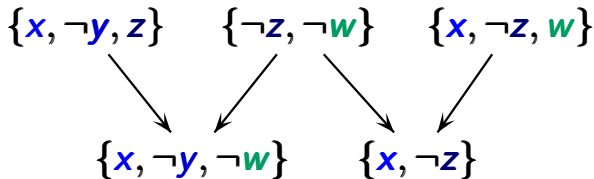
$$\alpha = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (x \vee \neg y \vee \neg w) \wedge \\ (x \vee \neg z \vee w) \wedge (\neg z \vee \neg w)$$

↑  
einzige Klausel bestehend  
aus negativen Literalen



$$\alpha = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (x \vee \neg y \vee \neg w) \wedge (x \vee \neg z \vee w) \wedge (\neg z \vee \neg w)$$

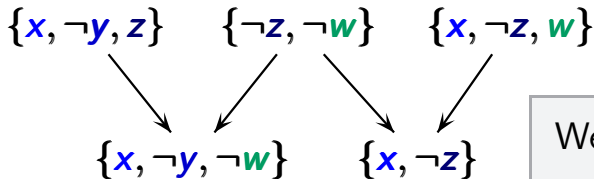
$\uparrow$   
 einzige Klausel bestehend  
 aus negativen Literalen



$$NRes^*(\alpha) = \alpha \cup \{x \vee \neg y \vee \neg w, x \vee \neg z\}$$

$$\alpha = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (x \vee \neg y \vee \neg w) \wedge (x \vee \neg z \vee w) \wedge (\neg z \vee \neg w)$$

$\uparrow$   
 einzige Klausel bestehend  
 aus negativen Literalen



Wegen  $\perp \notin NRes^*(\alpha)$   
 ist  $\alpha$  erfüllbar.

$$NRes^*(\alpha) = \alpha \cup \{x \vee \neg y \vee \neg w, x \vee \neg z\}$$

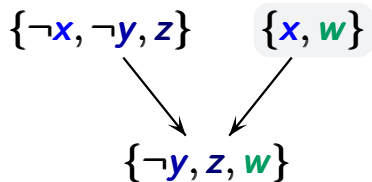
- N-Resolution
- P-Resolution
- lineare Resolution
- Input-Resolution
- ⋮

P-Resolutionsschritt:

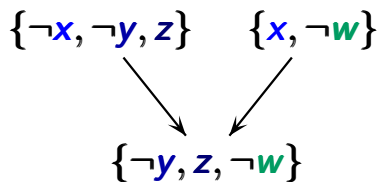
wie gewöhnlicher Resolutionsschritt, aber eine der Elternklauseln muss aus **positiven Literalen** bestehen

P-Resolutionsschritt:

wie gewöhnlicher Resolutionsschritt, aber eine der Elternklauseln muss aus **positiven Literalen** bestehen



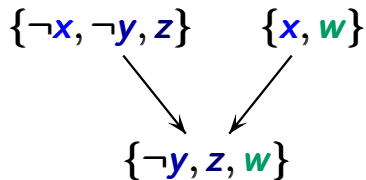
P-Resolutionsschritt



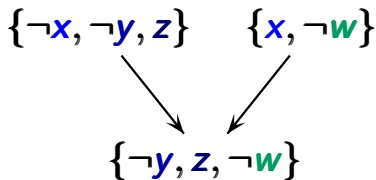
kein P-Resolutionsschritt

P-Resolutionsschritt:

wie gewöhnlicher Resolutionsschritt, aber eine der Elternklauseln muss aus positiven Literalen bestehen



P-Resolutionsschritt



kein P-Resolutionsschritt

P-Herleitung: Herleitung  $\langle \kappa_1, \tau_1, \lambda_1 \rangle \dots \langle \kappa_m, \tau_m, \lambda_m \rangle$ ,  
so dass  $\kappa_1, \dots, \kappa_m$  nur aus positiven  
Literalen bestehen

$$\alpha = (x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

unerfüllbare KNF-Formel

$$\alpha = (x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

unerfüllbare KNF-Formel

$$\{x, y\}$$

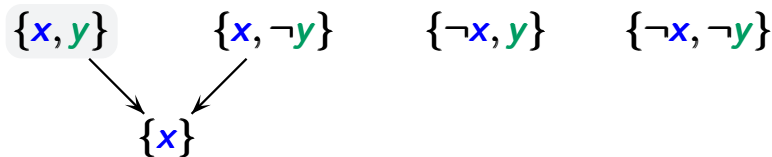
$$\{x, \neg y\}$$

$$\{\neg x, y\}$$

$$\{\neg x, \neg y\}$$

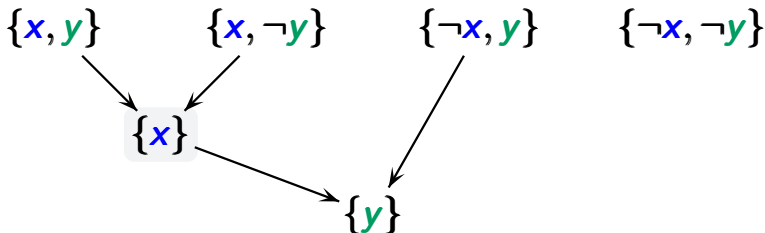
$$\alpha = (x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

unerfüllbare KNF-Formel



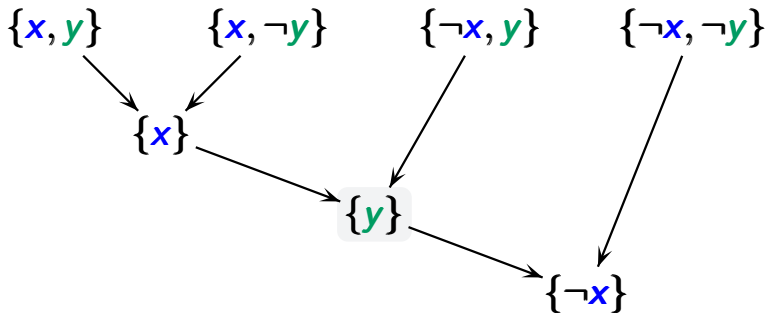
$$\alpha = (x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

unerfüllbare KNF-Formel



$$\alpha = (x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

unerfüllbare KNF-Formel





... wie für die N-Resolution ...

Für jede KNF-Formel  $\alpha$  gilt:  $\alpha \subseteq PRes^*(\alpha) \subseteq Res^*(\alpha)$

↑  
Menge aller Klauseln, die aus  $\alpha$  durch  
eine P-Herleitung erzeugbar sind

Für jede KNF-Formel  $\alpha$  gilt:  $\alpha \subseteq PRes^*(\alpha) \subseteq Res^*(\alpha)$

Menge aller Klauseln, die aus  $\alpha$  durch  
eine P-Herleitung erzeugbar sind

Folgerung: Korrektheit der P-Resolution

$\alpha \Vdash \lambda$  für alle  $\lambda \in PRes^*(\alpha)$

Für jede KNF-Formel  $\alpha$  gilt:  $\alpha \subseteq PRes^*(\alpha) \subseteq Res^*(\alpha)$

↑  
Menge aller Klauseln, die aus  $\alpha$  durch  
eine P-Herleitung erzeugbar sind

Folgerung: Korrektheit der P-Resolution

$\alpha \models \lambda$  für alle  $\lambda \in PRes^*(\alpha)$

Hauptsatz zur P-Resolution: (ohne Beweis)

$\alpha$  unerfüllbar gdw  $\perp \in PRes^*(\alpha)$

- N-Resolution
- P-Resolution
- lineare Resolution
- Input-Resolution
- ⋮

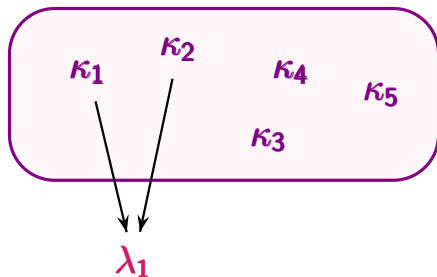




KNF-Formel  $\alpha$   
als Klauselmenge

lineare Herleitung: Herleitung der Gestalt

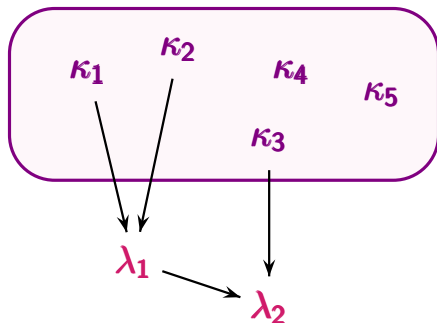
$$\langle \tau_0, \tau_1, \lambda_1 \rangle \langle \lambda_1, \tau_2, \lambda_2 \rangle \langle \lambda_2, \tau_3, \lambda_3 \rangle \dots \langle \lambda_{m-1}, \tau_m, \lambda_m \rangle$$



KNF-Formel  $\alpha$   
als Klauselmenge

lineare Herleitung: Herleitung der Gestalt

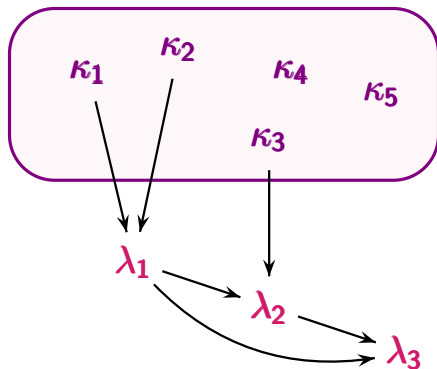
$$\langle \tau_0, \tau_1, \lambda_1 \rangle \langle \lambda_1, \tau_2, \lambda_2 \rangle \langle \lambda_2, \tau_3, \lambda_3 \rangle \dots \langle \lambda_{m-1}, \tau_m, \lambda_m \rangle$$



KNF-Formel  $\alpha$   
als Klauselmenge

lineare Herleitung: Herleitung der Gestalt

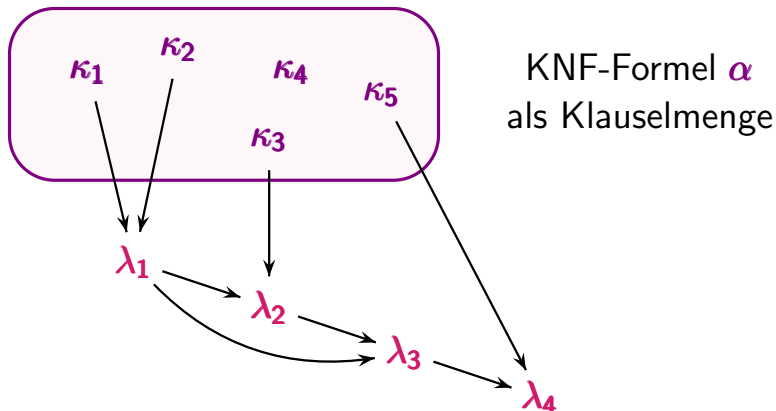
$$\langle \tau_0, \tau_1, \lambda_1 \rangle \langle \lambda_1, \tau_2, \lambda_2 \rangle \langle \lambda_2, \tau_3, \lambda_3 \rangle \dots \langle \lambda_{m-1}, \tau_m, \lambda_m \rangle$$



KNF-Formel  $\alpha$   
als Klauselmenge

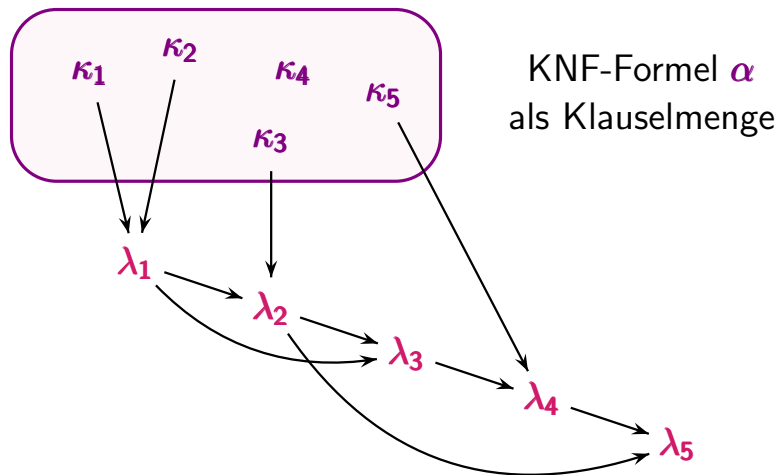
lineare Herleitung: Herleitung der Gestalt

$$\langle \tau_0, \tau_1, \lambda_1 \rangle \langle \lambda_1, \tau_2, \lambda_2 \rangle \langle \lambda_2, \tau_3, \lambda_3 \rangle \dots \langle \lambda_{m-1}, \tau_m, \lambda_m \rangle$$



lineare Herleitung: Herleitung der Gestalt

$$\langle \tau_0, \tau_1, \lambda_1 \rangle \langle \lambda_1, \tau_2, \lambda_2 \rangle \langle \lambda_2, \tau_3, \lambda_3 \rangle \dots \langle \lambda_{m-1}, \tau_m, \lambda_m \rangle$$

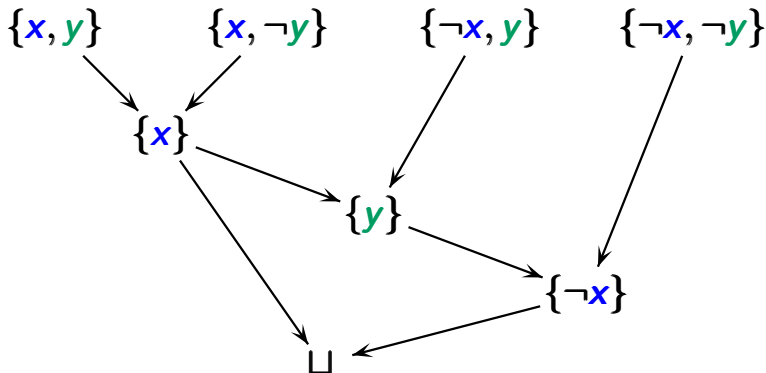


lineare Herleitung: Herleitung der Gestalt

$$\langle \tau_0, \tau_1, \lambda_1 \rangle \langle \lambda_1, \tau_2, \lambda_2 \rangle \langle \lambda_2, \tau_3, \lambda_3 \rangle \dots \langle \lambda_{m-1}, \tau_m, \lambda_m \rangle$$

$$\alpha = (x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

unerfüllbare KNF-Formel





Für jede KNF-Formel  $\alpha$  und Klausel  $\lambda$  gilt:

Ist  $\lambda$  aus  $\alpha$  durch eine **lineare Herleitung** herleitbar, so gilt  $\alpha \Vdash \lambda$ .



gilt für jede Resolutionsherleitung,  
also auch für lineare Herleitungen

Für jede KNF-Formel  $\alpha$  und Klausel  $\lambda$  gilt:

Ist  $\lambda$  aus  $\alpha$  durch eine lineare Herleitung herleitbar, so gilt  $\alpha \Vdash \lambda$ .

es gibt eine lineare  
Widerlegung für  $\alpha$  }  $\implies$   $\alpha$  ist unerfüllbar

↑  
lineare Herleitung  
der leeren Klausel  
 $\perp = \textit{false}$

Für jede KNF-Formel  $\alpha$  und Klausel  $\lambda$  gilt:

Ist  $\lambda$  aus  $\alpha$  durch eine lineare Herleitung herleitbar, so gilt  $\alpha \Vdash \lambda$ .

Widerlegungsvollständigkeit der linearen Resolution:

es gibt eine lineare  
Widerlegung für  $\alpha$  } gdw  $\alpha$  ist unerfüllbar

lineare Herleitung  
der leeren Klausel  
 $\perp = \textit{false}$

(ohne Beweis)

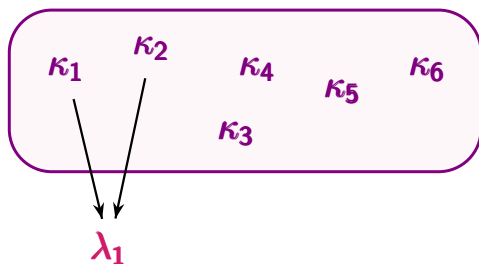




KNF-Formel  $\alpha$   
als Klauselmenge

Input-Herleitung aus  $\alpha$ :

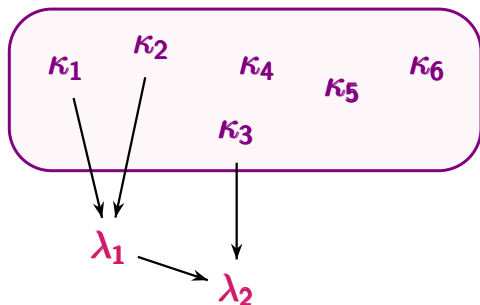
lineare Herleitung, so dass in jedem Resolutionsschritt eine Klausel aus  $\alpha$  als Elternklausel eingesetzt wird



KNF-Formel  $\alpha$   
als Klauselmenge

Input-Herleitung aus  $\alpha$ :

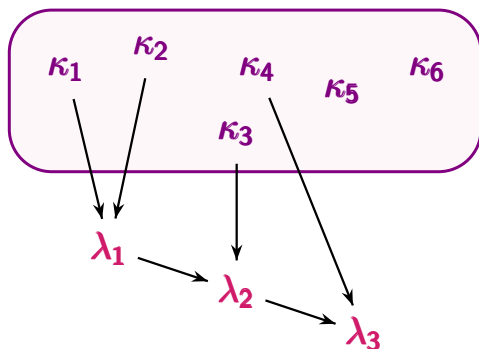
lineare Herleitung, so dass in jedem Resolutionsschritt eine Klausel aus  $\alpha$  als Elternklausel eingesetzt wird



KNF-Formel  $\alpha$   
als Klauselmenge

Input-Herleitung aus  $\alpha$ :

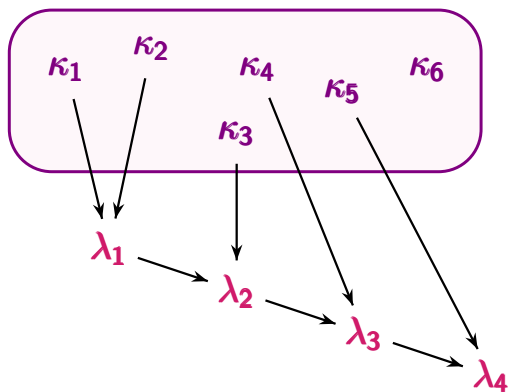
lineare Herleitung, so dass in jedem Resolutionsschritt eine Klausel aus  $\alpha$  als Elternklausel eingesetzt wird



KNF-Formel  $\alpha$   
als Klauselmenge

Input-Herleitung aus  $\alpha$ :

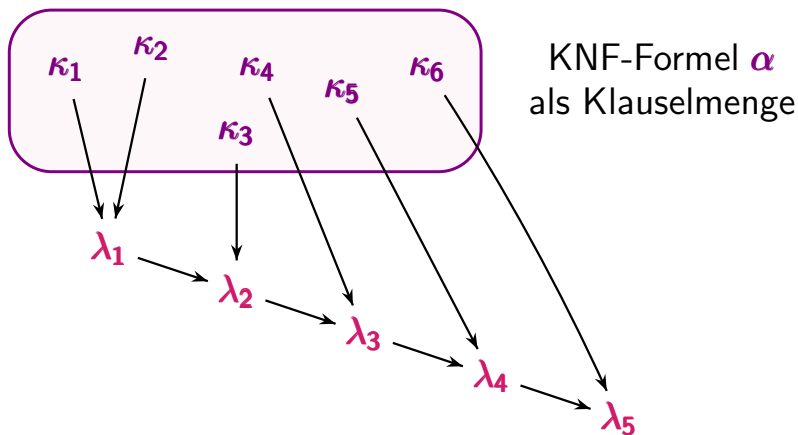
lineare Herleitung, so dass in jedem Resolutionsschritt eine Klausel aus  $\alpha$  als Elternklausel eingesetzt wird



KNF-Formel  $\alpha$   
als Klauselmenge

Input-Herleitung aus  $\alpha$ :

lineare Herleitung, so dass in jedem Resolutionsschritt eine Klausel aus  $\alpha$  als Elternklausel eingesetzt wird



Input-Herleitung aus  $\alpha$ :

lineare Herleitung, so dass in jedem Resolutionsschritt eine Klausel aus  $\alpha$  als Elternklausel eingesetzt wird



Die Input-Resolution ist korrekt, aber unvollständig.

Die Input-Resolution ist korrekt, aber unvollständig.

Korrektheit: Ist  $\perp$  durch eine Input-Herleitung aus  $\alpha$  herleitbar, so ist  $\alpha$  unerfüllbar.

Die Input-Resolution ist korrekt, aber unvollständig.

Korrektheit: Ist  $\perp$  durch eine Input-Herleitung aus  $\alpha$  herleitbar, so ist  $\alpha$  unerfüllbar.

Unvollständigkeit: es gibt unerfüllbare KNF-Formeln, die keine Input-Widerlegung haben

Die Input-Resolution ist korrekt, aber unvollständig.

Korrektheit: Ist  $\sqcup$  durch eine Input-Herleitung aus  $\alpha$  herleitbar, so ist  $\alpha$  unerfüllbar.

Unvollständigkeit: es gibt unerfüllbare KNF-Formeln, die keine Input-Widerlegung haben, z.B.

$$\alpha = (x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

ist unerfüllbar, aber hat keine Input-Widerlegung

Die Input-Resolution ist korrekt, aber unvollständig.

Korrektheit: Ist  $\sqcup$  durch eine Input-Herleitung aus  $\alpha$  herleitbar, so ist  $\alpha$  unerfüllbar.

Unvollständigkeit: es gibt unerfüllbare KNF-Formeln, die keine Input-Widerlegung haben, z.B.

$$\alpha = (x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

ist unerfüllbar, aber hat keine Input-Widerlegung

denn: alle Resolventen  $\lambda = \kappa \setminus \{L\} \cup \tau \setminus \{\bar{L}\}$

mit  $\kappa \in \alpha$  enthalten mindestens ein Literal

Hornformel: KNF-Formel bestehend aus Hornklauseln

Hornklausel: höchstens ein positives Literal

Resolventen von **Hornklauseln** sind **Hornklauseln**.

Hornformel: KNF-Formel bestehend aus Hornklauseln

Hornklausel: höchstens ein positives Literal

Resolventen von Hornklauseln sind Hornklauseln.

Beweis: Seien  $\kappa, \tau$  Hornklauseln mit  $x \in \kappa$  und  $\neg x \in \tau$ .

$$\lambda = \underbrace{\kappa \setminus \{x\}} \cup \tau \setminus \{\neg x\} \text{ ist eine Hornklausel}$$

Klausel bestehend aus  
negativen Literalen

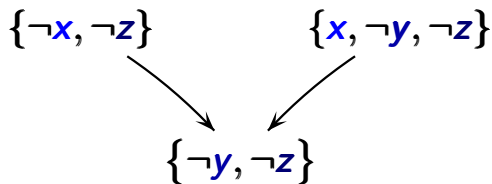
Hornformel: KNF-Formel bestehend aus Hornklauseln

Hornklausel: höchstens ein positives Literal

Resolventen von HornklauseIn sind HornklauseIn.

Beweis: Seien  $\kappa, \tau$  HornklauseIn mit  $x \in \kappa$  und  $\neg x \in \tau$ .

$\lambda = \kappa \setminus \{x\} \cup \tau \setminus \{\neg x\}$  ist eine HornklauseI



Resolventen von Hornklauseln sind Hornklauseln.

Beweis: Seien  $\kappa, \tau$  Hornklauseln mit  $x \in \kappa$  und  $\neg x \in \tau$ .


$\lambda = \kappa \setminus \{x\} \cup \tau \setminus \{\neg x\}$  ist eine Hornklausel

Zielklausel

$\{\neg x, \neg z\}$

Regel

$\{x, \neg y, \neg z\}$



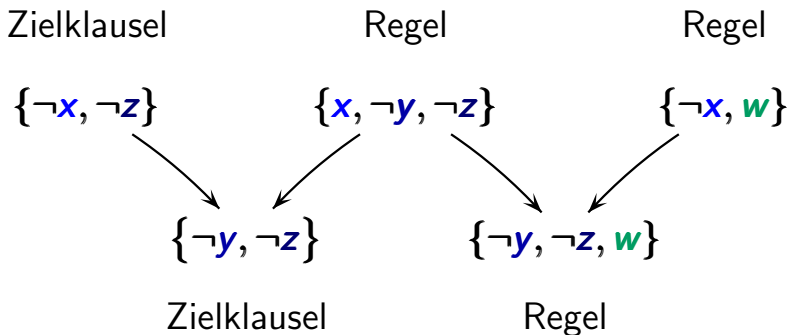
$\{\neg y, \neg z\}$

Zielklausel

Resolventen von Hornklauseln sind Hornklauseln.

Beweis: Seien  $\kappa, \tau$  Hornklauseln mit  $x \in \kappa$  und  $\neg x \in \tau$ .

$\lambda = \kappa \setminus \{x\} \cup \tau \setminus \{\neg x\}$  ist eine Hornklausel



Resolventen von Hornklauseln sind Hornklauseln.

Beweis: Seien  $\kappa, \tau$  Hornklauseln mit  $x \in \kappa$  und  $\neg x \in \tau$ .

$\lambda = \kappa \setminus \{x\} \cup \tau \setminus \{\neg x\}$  ist eine Hornklausel

Resolventenbildung nur möglich, wenn wenigstens eine der Elternklauseln

- eine Regel oder Zielklausel ist
- eine Regel oder ein Faktum ist

Resolventen von Hornklauseln sind Hornklauseln.

Beweis: Seien  $\kappa, \tau$  Hornklauseln mit  $x \in \kappa$  und  $\neg x \in \tau$ .

$\lambda = \kappa \setminus \{x\} \cup \tau \setminus \{\neg x\}$  ist eine Hornklausel

Resolventenbildung nur möglich, wenn wenigstens eine der Elternklauseln

- eine Regel oder Zielklausel ist
- eine Regel oder ein Faktum ist

$\{\neg x, \neg y\}$

$\{\neg x\}$

zwei Zielklauseln:  
kein Resolvent

$\{x\}$

$\{y\}$

zwei Fakten:  
kein Resolvent



Sei  $\alpha$  eine Hornformel. Dann gilt:

$\alpha$  ist unerfüllbar

gdw es gibt eine Widerlegung von  $\alpha$ , so dass  
in jedem Resolutionsschritt eine **Zielklausel**  
als Elternklausel eingesetzt wird

Sei  $\alpha$  eine Hornformel. Dann gilt:

$\alpha$  ist unerfüllbar

gdw es gibt eine Widerlegung von  $\alpha$ , so dass  
in jedem Resolutionsschritt eine Zielklausel  
als Elternklausel eingesetzt wird

Widerlegungsvollständigkeit der N-Resolution

Sei  $\alpha$  eine Hornformel. Dann gilt:

$\alpha$  ist unerfüllbar

gdw es gibt eine Widerlegung von  $\alpha$ , so dass  
in jedem Resolutionsschritt eine Zielklausel  
als Elternklausel eingesetzt wird

gdw es gibt eine Widerlegung von  $\alpha$ , so dass  
in jedem Resolutionsschritt ein Faktum  
als Elternklausel eingesetzt wird

Widerlegungsvollständigkeit der N-Resolution

Sei  $\alpha$  eine Hornformel. Dann gilt:

$\alpha$  ist unerfüllbar

gdw es gibt eine Widerlegung von  $\alpha$ , so dass  
in jedem Resolutionsschritt eine **Zielklausel**  
als Elternklausel eingesetzt wird

gdw es gibt eine Widerlegung von  $\alpha$ , so dass  
in jedem Resolutionsschritt ein **Faktum**  
als Elternklausel eingesetzt wird

Widerlegungsvollständigkeit der N-Resolution  
und der **P-Resolution**



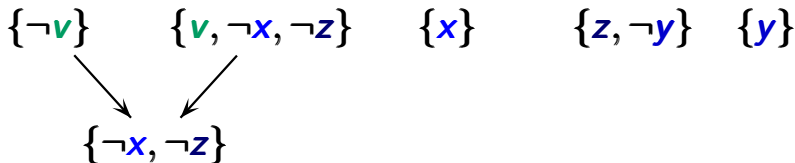
$$(v \rightarrow \text{false}) \wedge (x \wedge z \rightarrow v) \wedge x \wedge (y \rightarrow z) \wedge y$$

$$(v \rightarrow \text{false}) \wedge (x \wedge z \rightarrow v) \wedge x \wedge (y \rightarrow z) \wedge y$$

↑  
einzige Zielklausel

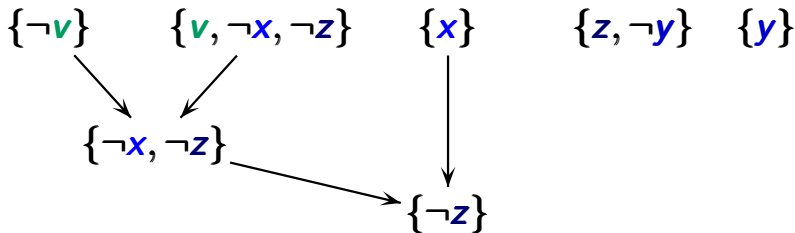
$$(v \rightarrow \text{false}) \wedge (x \wedge z \rightarrow v) \wedge x \wedge (y \rightarrow z) \wedge y$$

↑  
einzige Zielklausel



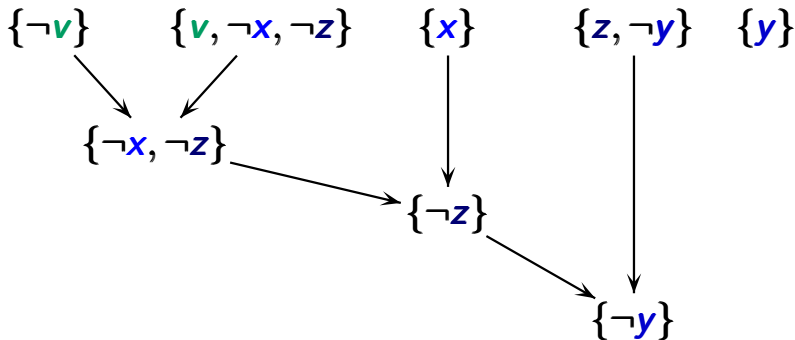
$$(v \rightarrow \text{false}) \wedge (x \wedge z \rightarrow v) \wedge x \wedge (y \rightarrow z) \wedge y$$

↑  
einzige Zielklausel



$$(v \rightarrow \text{false}) \wedge (x \wedge z \rightarrow v) \wedge x \wedge (y \rightarrow z) \wedge y$$

↑  
einzige Zielklausel

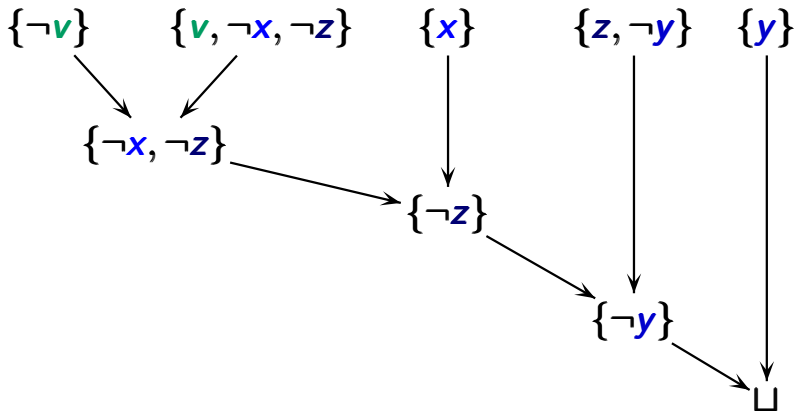


# Beispiel: N-Resolution für Hornformeln

835

$$(v \rightarrow \text{false}) \wedge (x \wedge z \rightarrow v) \wedge x \wedge (y \rightarrow z) \wedge y$$

↑  
einzige Zielklausel



$$(v \rightarrow \text{false}) \wedge (x \wedge z \rightarrow v) \wedge x \wedge (y \rightarrow z) \wedge y$$

P-Resolution für Hornformeln:

verwende in jedem Resolutionsschritt ein **Faktum**  
(positive Einheitsklausel) als Elternklausel

$$(v \rightarrow \text{false}) \wedge (x \wedge z \rightarrow v) \wedge \boxed{x} \wedge (y \rightarrow z) \wedge \boxed{y}$$

$\uparrow$  Fakt                       $\uparrow$  Fakt

P-Resolution für Hornformeln:

verwende in jedem Resolutionsschritt ein **Faktum**  
(positive Einheitsklausel) als Elternklausel

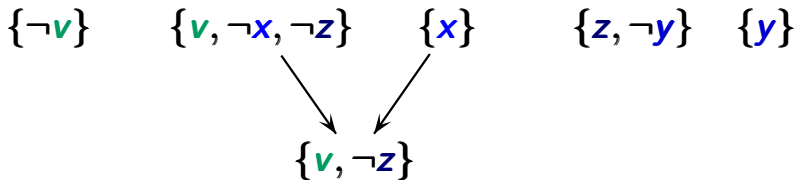
$$(v \rightarrow \text{false}) \wedge (x \wedge z \rightarrow v) \wedge x \wedge (y \rightarrow z) \wedge y$$

$\uparrow$  Fakt                       $\uparrow$  Fakt

$$\{\neg v\} \quad \{v, \neg x, \neg z\} \quad \{x\} \quad \{z, \neg y\} \quad \{y\}$$

$$(v \rightarrow \text{false}) \wedge (x \wedge z \rightarrow v) \wedge x \wedge (y \rightarrow z) \wedge y$$

$\uparrow$  Fakt                       $\uparrow$  Fakt











Die **Input-Resolution** ist für Hornformeln widerlegungsvollständig.

d.h.,  $\alpha$  ist genau dann unerfüllbar wenn es eine Widerlegung gibt, so dass in jedem Resolutionsschritt eine Klausel aus  $\alpha$  als Elternklausel eingesetzt wird

Die **Input-Resolution** ist für Hornformeln widerlegungsvollständig.

d.h.,  $\alpha$  ist genau dann unerfüllbar wenn es eine Widerlegung gibt, so dass in jedem Resolutionsschritt eine Klausel aus  $\alpha$  als Elternklausel eingesetzt wird

Warum ???

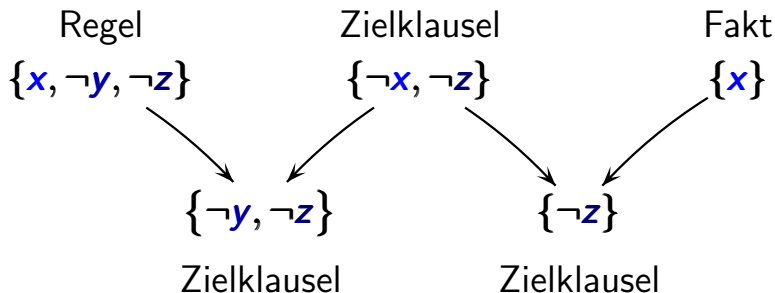
Die **Input-Resolution** ist für Hornformeln widerlegungsvollständig.

d.h.,  $\alpha$  ist genau dann unerfüllbar wenn es eine Widerlegung gibt, so dass in jedem Resolutionsschritt eine Klausel aus  $\alpha$  als Elternklausel eingesetzt wird

## Warum ???

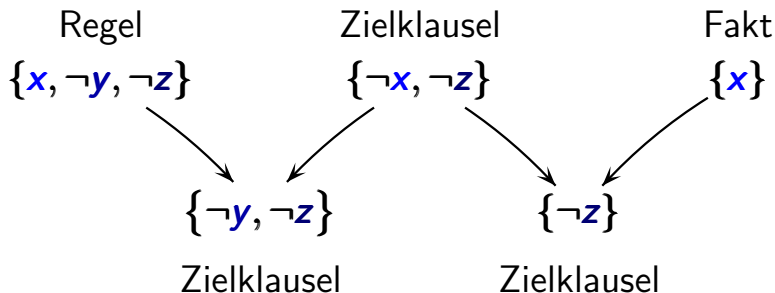
Jede **N-Herleitung** aus einer Hornformel kann als **Input-Herleitung** aufgefasst werden.

Die **Input-Resolution** ist für Hornformeln widerlegungsvollständig.



Jede **N-Herleitung** aus einer Hornformel kann als **Input-Herleitung** aufgefasst werden.

Die **Input-Resolution** ist für Hornformeln widerlegungsvollständig.

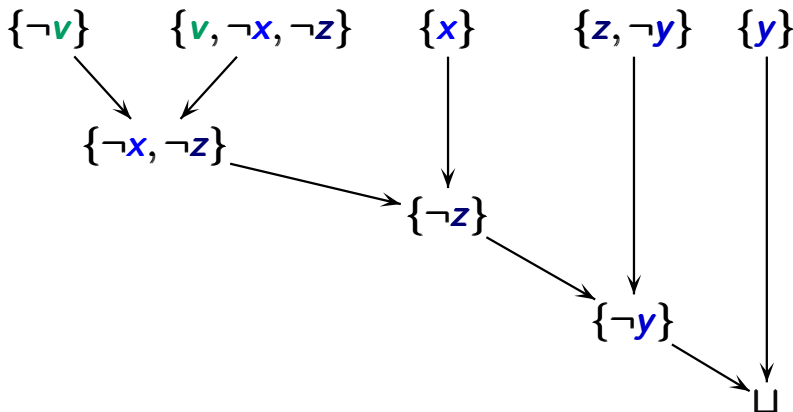


**N-Resolventen** aus Hornklauseln sind Zielklauseln.

Und diese sind nur mit einer **Regel** oder einem **Faktum** der Eingabeformel resolvierbar.

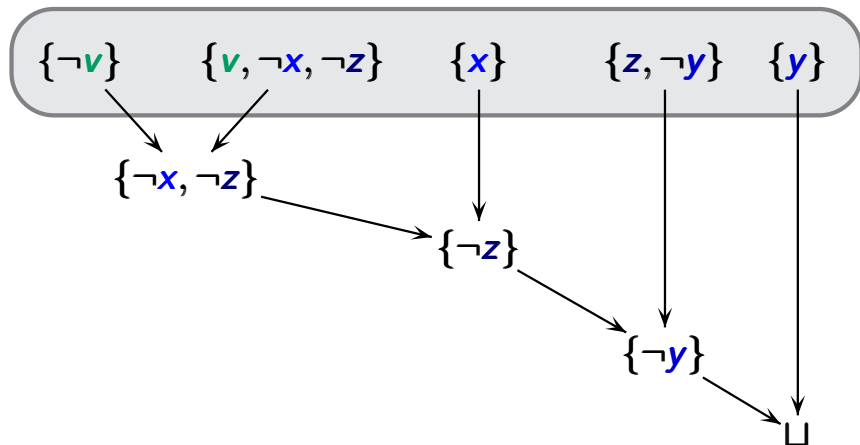
$$(v \rightarrow \text{false}) \wedge (x \wedge z \rightarrow v) \wedge x \wedge (y \rightarrow z) \wedge y$$

N-Widerlegung



$$(v \rightarrow \text{false}) \wedge (x \wedge z \rightarrow v) \wedge x \wedge (y \rightarrow z) \wedge y$$

N-Widerlegung ist zugleich **Input-Widerlegung**



1. **Teil:** Formale Sprachen und Automaten
2. **Teil:** **Aussagenlogik**
  - Grundbegriffe der Aussagenlogik
  - Hornformeln
  - SAT-Beweiser
  - Resolution
  - binäre Entscheidungsgraphen
  - quantifizierte Boolesche Formeln

1. Teil: Formale Sprachen und Automaten

2. Teil: **Aussagenlogik**

- Grundbegriffe der Aussagenlogik
- Hornformeln
- SAT-Beweiser
- Resolution
- ~~binäre Entscheidungsgraphen~~
- ~~quantifizierte Boolesche Formeln~~



Datenstruktur zur Darstellung von **Schaltfunktionen**

↑  
Semantik von  
aussagenlogischen  
Formeln

Datenstruktur zur Darstellung von Schaltfunktionen

$f$  : Menge der Belegungen für  $AP \rightarrow \{0, 1\}$

Datenstruktur zur Darstellung von Schaltfunktionen

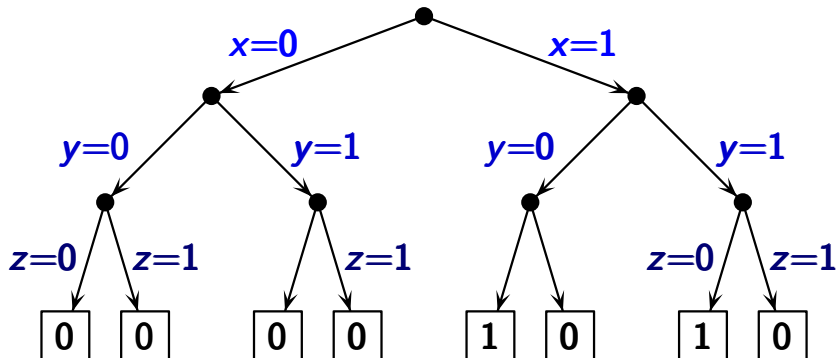
$f$  : Menge der Belegungen für  $AP \rightarrow \{0, 1\}$

Idee: Kompaktifizierung des Entscheidungsbaums

Datenstruktur zur Darstellung von Schaltfunktionen, z.B.

$$f : (\{x, y, z\} \rightarrow \{0, 1\}) \rightarrow \{0, 1\}, \quad f(x, y, z) = x \wedge \neg z$$

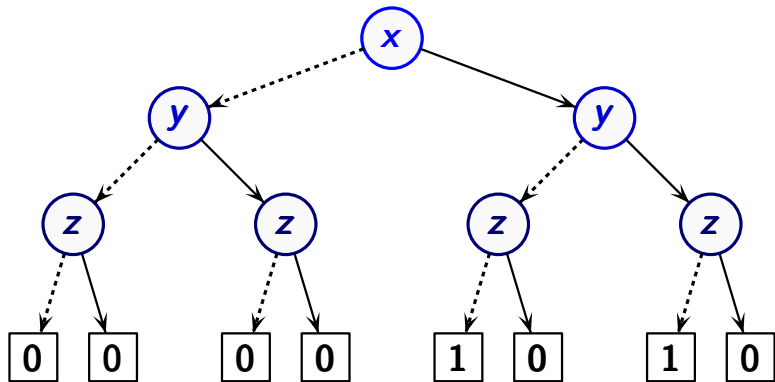
Idee: Kompaktifizierung des Entscheidungsbaums



Datenstruktur zur Darstellung von Schaltfunktionen, z.B.

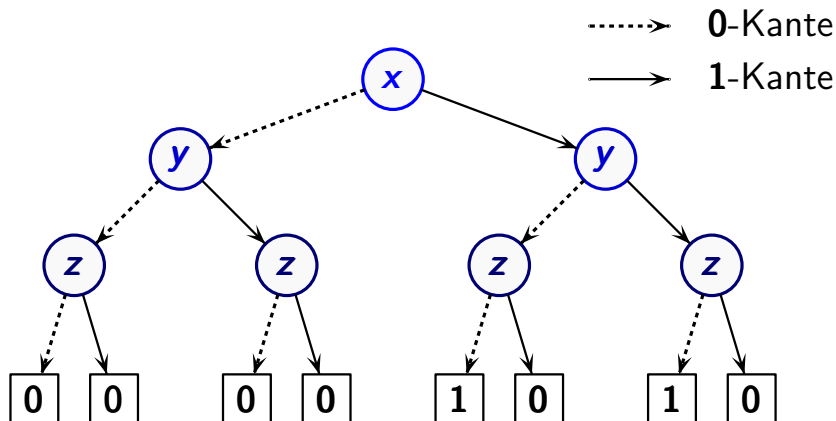
$$f : (\{x, y, z\} \rightarrow \{0, 1\}) \rightarrow \{0, 1\}, \quad f(x, y, z) = x \wedge \neg z$$

Idee: Kompaktifizierung des Entscheidungsbaums



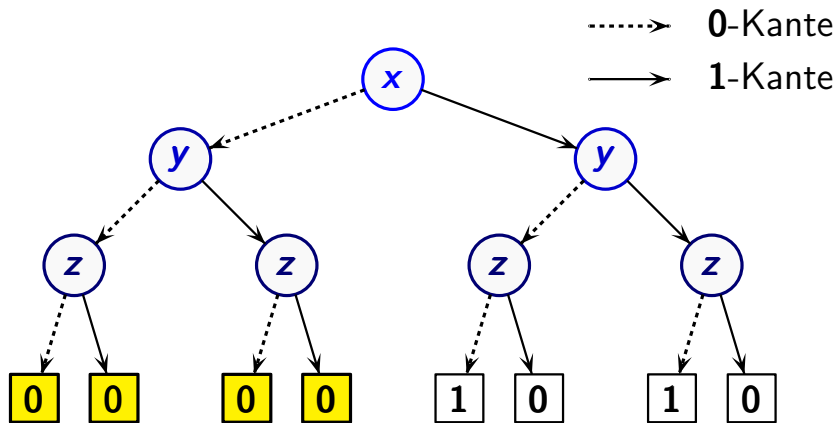
Datenstruktur zur Darstellung von Schaltfunktionen, z.B.

$$f : (\{x, y, z\} \rightarrow \{0, 1\}) \rightarrow \{0, 1\}, \quad f(x, y, z) = x \wedge \neg z$$



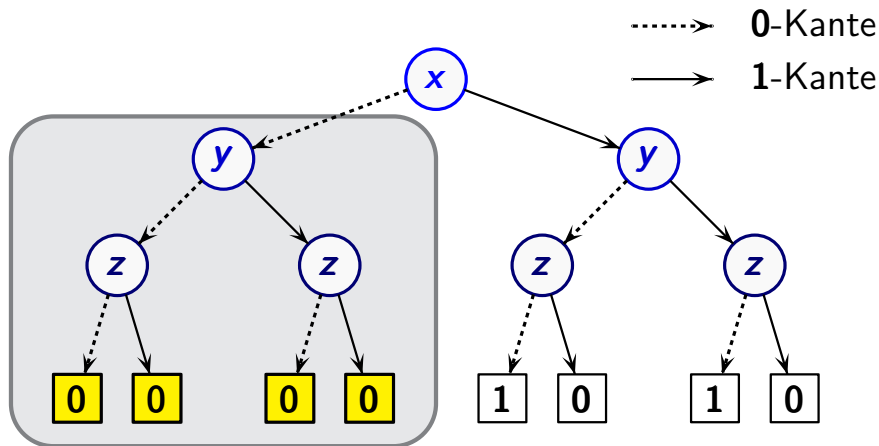
Datenstruktur zur Darstellung von Schaltfunktionen, z.B.

$$f : (\{x, y, z\} \rightarrow \{0, 1\}) \rightarrow \{0, 1\}, \quad f(x, y, z) = x \wedge \neg z$$



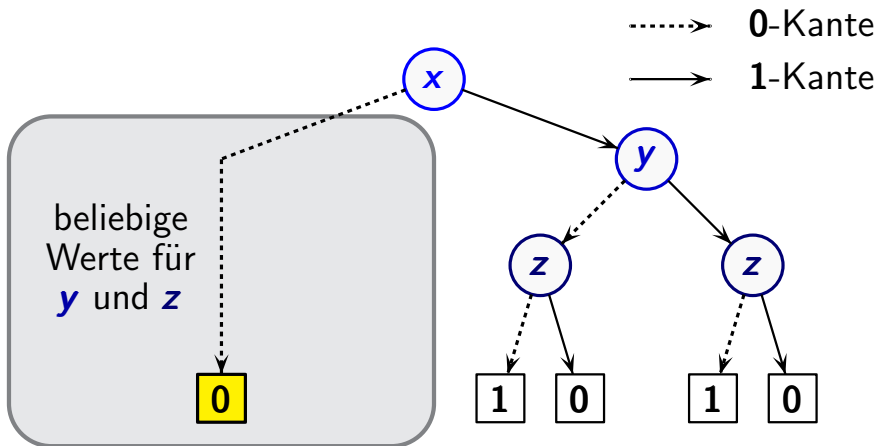
Datenstruktur zur Darstellung von Schaltfunktionen, z.B.

$$f : (\{x, y, z\} \rightarrow \{0, 1\}) \rightarrow \{0, 1\}, \quad f(x, y, z) = x \wedge \neg z$$



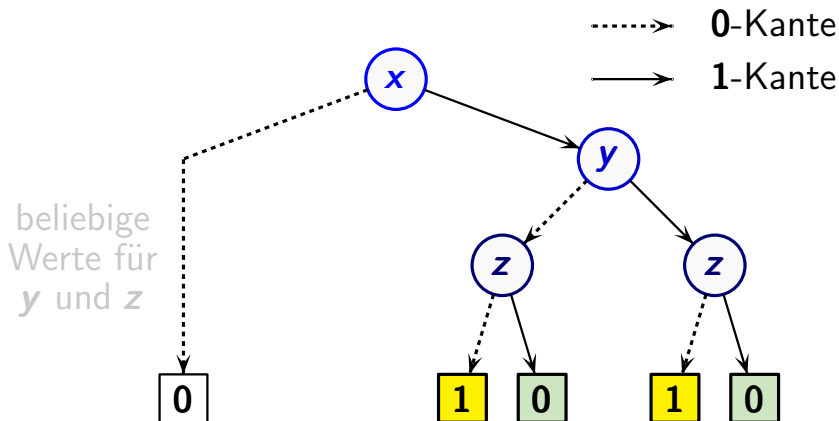
Datenstruktur zur Darstellung von Schaltfunktionen, z.B.

$$f : (\{x, y, z\} \rightarrow \{0, 1\}) \rightarrow \{0, 1\}, \quad f(x, y, z) = x \wedge \neg z$$



Datenstruktur zur Darstellung von Schaltfunktionen, z.B.

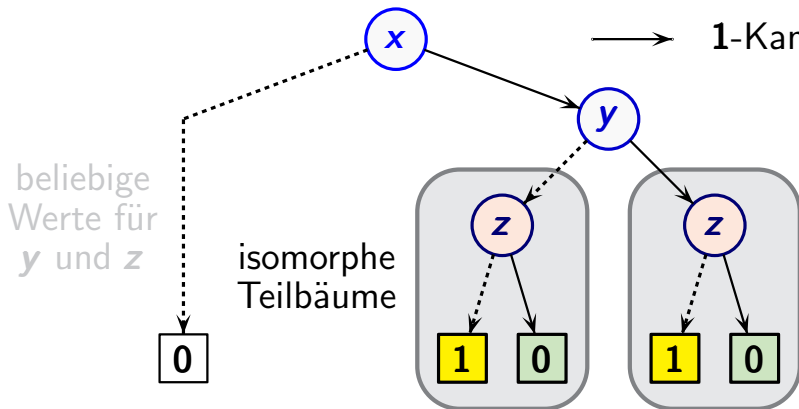
$$f : (\{x, y, z\} \rightarrow \{0, 1\}) \rightarrow \{0, 1\}, \quad f(x, y, z) = x \wedge \neg z$$



Datenstruktur zur Darstellung von Schaltfunktionen, z.B.

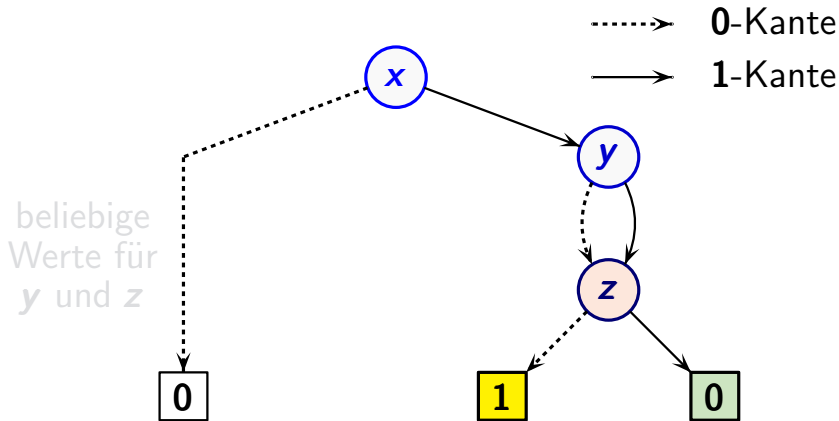
$$f : (\{x, y, z\} \rightarrow \{0, 1\}) \rightarrow \{0, 1\}, \quad f(x, y, z) = x \wedge \neg z$$

-----> 0-Kante  
-----> 1-Kante



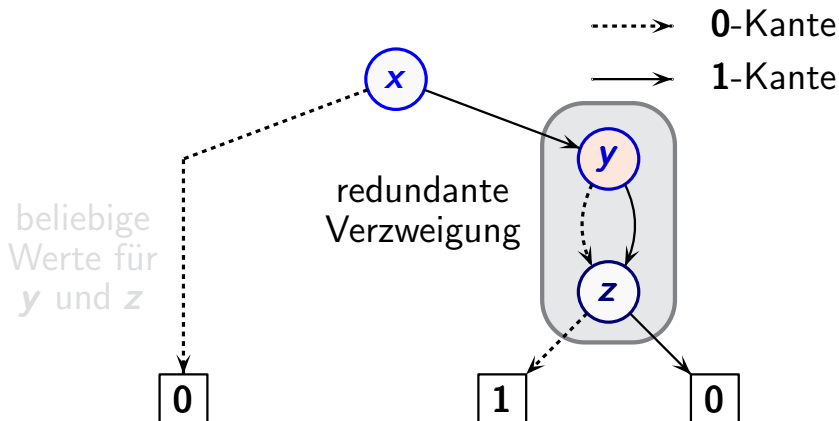
Datenstruktur zur Darstellung von Schaltfunktionen, z.B.

$$f : (\{x, y, z\} \rightarrow \{0, 1\}) \rightarrow \{0, 1\}, \quad f(x, y, z) = x \wedge \neg z$$



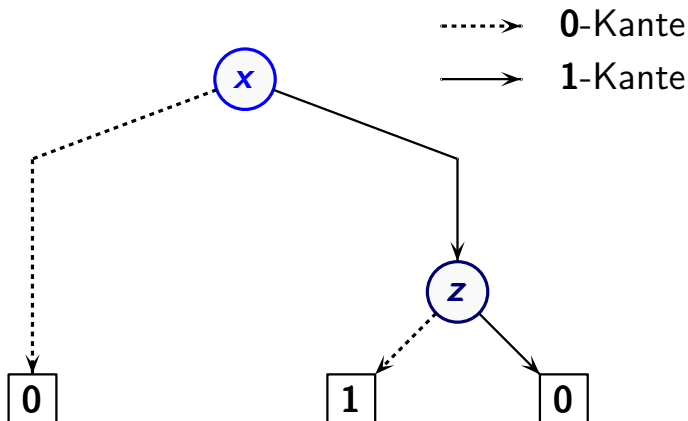
Datenstruktur zur Darstellung von Schaltfunktionen, z.B.

$$f : (\{x, y, z\} \rightarrow \{0, 1\}) \rightarrow \{0, 1\}, \quad f(x, y, z) = x \wedge \neg z$$



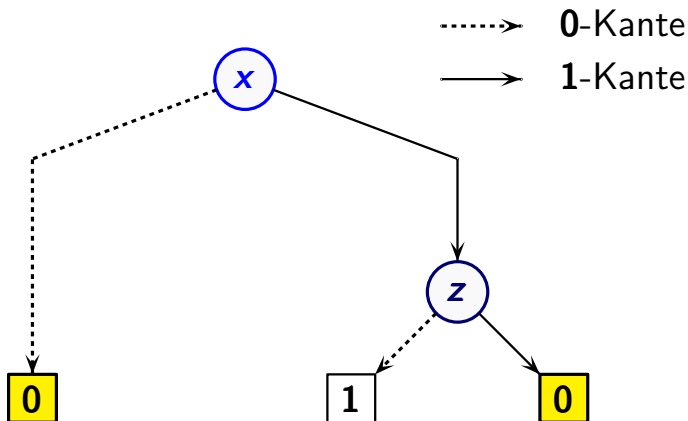
Datenstruktur zur Darstellung von Schaltfunktionen, z.B.

$$f : (\{x, y, z\} \rightarrow \{0, 1\}) \rightarrow \{0, 1\}, \quad f(x, y, z) = x \wedge \neg z$$



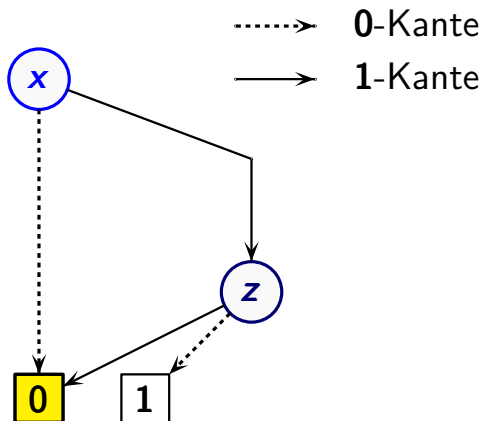
Datenstruktur zur Darstellung von Schaltfunktionen, z.B.

$$f : (\{x, y, z\} \rightarrow \{0, 1\}) \rightarrow \{0, 1\}, \quad f(x, y, z) = x \wedge \neg z$$



Datenstruktur zur Darstellung von Schaltfunktionen, z.B.

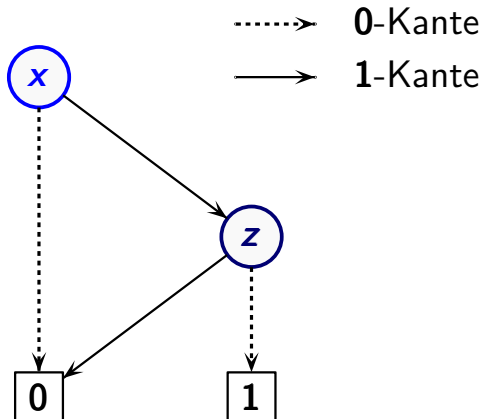
$$f : (\{x, y, z\} \rightarrow \{0, 1\}) \rightarrow \{0, 1\}, \quad f(x, y, z) = x \wedge \neg z$$



Datenstruktur zur Darstellung von Schaltfunktionen, z.B.

$$f : (\{x, y, z\} \rightarrow \{0, 1\}) \rightarrow \{0, 1\}, \quad f(x, y, z) = x \wedge \neg z$$

reduzierter  
geordneter binärer  
Entscheidungsgraph





Erweiterung der Aussagenlogik um Quantoren

$\exists x. \alpha$  “es gibt ein  $x \in \{0, 1\}$ , so dass  $\alpha$  gilt”

$\forall x. \alpha$  “für alle  $x \in \{0, 1\}$  gilt  $\alpha$ ”

Erweiterung der Aussagenlogik um Quantoren

$\exists x. \alpha$  “es gibt ein  $x \in \{0, 1\}$ , so dass  $\alpha$  gilt”

$\forall x. \alpha$  “für alle  $x \in \{0, 1\}$  gilt  $\alpha$ ”

Ausdrucksstärke wie Aussagenlogik, denn:

$$\exists x. \alpha \equiv \alpha[x/\text{false}] \vee \alpha[x/\text{true}]$$

$$\forall x. \alpha \equiv \alpha[x/\text{false}] \wedge \alpha[x/\text{true}]$$

aber oftmals kompaktere Darstellungen



zur Darstellung von Eigenschaften relationaler  
Strukturen, z.B. Graphen

zur Darstellung von Eigenschaften relationaler Strukturen, z.B. Graphen

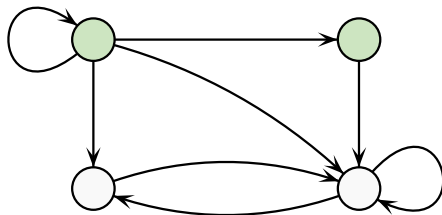
- Prädikatenlogik

$$\exists x. (\text{grün}(x) \wedge \forall y. \text{kante}(x, y))$$

“es gibt einen grünen Knoten  $x$ , für den es zu jedem Knoten  $y$  eine Kante von  $x$  nach  $y$  gibt”

$\forall$  “für alle”

$\exists$  “es gibt ein”



zur Darstellung von Eigenschaften relationaler Strukturen, z.B. Graphen

- Prädikatenlogik (Logik erster Ordnung)

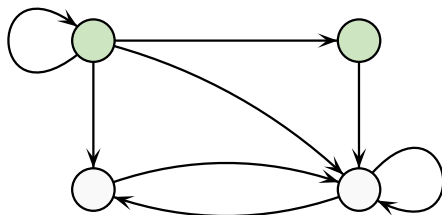
$$\exists x. (\text{grün}(x) \wedge \forall y. \text{kante}(x, y))$$

“es gibt einen grünen Knoten  $x$ , für den es zu jedem Knoten  $y$  eine Kante von  $x$  nach  $y$  gibt”

Quantoren über Individuen einer festen Grundmenge

$\forall$  “für alle”

$\exists$  “es gibt ein”



zur Darstellung von Eigenschaften relationaler Strukturen, z.B. Graphen

- Prädikatenlogik (Logik erster Ordnung)

$$\exists x. (\text{grün}(x) \wedge \forall y. \text{kante}(x, y))$$

- Logik zweiter Ordnung

$$(\forall P. P(x) \leftrightarrow P(y)) \longrightarrow x = y$$

Leibniz Axiom für Gleichheit

↑  
Quantifizierung über alle Mengen von Individuen

zur Darstellung von Eigenschaften relationaler Strukturen, z.B. Graphen

- Prädikatenlogik (Logik erster Ordnung)

$$\exists x. (\text{grün}(x) \wedge \forall y. \text{kante}(x, y))$$

- Logik zweiter Ordnung

$$(\forall P. P(x) \leftrightarrow P(y)) \longrightarrow x = y$$

Leibniz Axiom für Gleichheit

Vorlesungen **Theoretische Informatik und Logik**  
und **Advanced Logics**

- modale Logik

$\langle a \rangle \alpha$  “es gibt einen  $a$ -Nachfolger, für den  $\alpha$  gilt”

$[a] \alpha$  “für alle  $a$ -Nachfolger gilt  $\alpha$ ”

- modale Logik
  - $\langle a \rangle \alpha$  “es gibt einen  $a$ -Nachfolger, für den  $\alpha$  gilt”
  - $[a] \alpha$  “für alle  $a$ -Nachfolger gilt  $\alpha$ ”
- temporale Logik
  - $\diamond \alpha$  “ $\alpha$  gilt irgendwann in der Zukunft”
  - $\square \alpha$  “ $\alpha$  gilt von nun an immer”

- modale Logik
  - $\langle a \rangle \alpha$  “es gibt einen  $a$ -Nachfolger, für den  $\alpha$  gilt”
  - $[a] \alpha$  “für alle  $a$ -Nachfolger gilt  $\alpha$ ”
- temporale Logik
  - $\diamond \alpha$  “ $\alpha$  gilt irgendwann in der Zukunft”
  - $\square \alpha$  “ $\alpha$  gilt von nun an immer”

Anwendung: Verifikation paralleler Systeme

Vorlesung “Model Checking”

THE END

Lernraum: Anmeldung siehe VL-Webseite

Fr, 3. Februar	14:50-16:20	INF/E10
----------------	-------------	---------

Mo, 6. Februar	10:00-11:30	INF/E08
----------------	-------------	---------

		INF/E09
--	--	---------

		INF/E10
--	--	---------

Probeklausur:

- siehe Webseite am **Di, 31.01. (heute)** nach der VL
- Besprechung in der letzten VL am **Do, 02.02.**