

Sei α eine KNF-Formel, x ein Atom.

$\alpha[x/0]$ entsteht aus α durch folgende Schritte:

1. streiche x aus jeder Klausel, die das positive Literal x enthält
2. streiche jede Klausel, in der $\neg x$ vorkommt

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

\Downarrow 1. Schritt

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

\Downarrow 2. Schritt

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

- $\alpha[x/0] \equiv \alpha[x/\text{false}]$
- $\alpha[x/0]^I = \alpha^I$ für jede Belegung I mit $x^I = 0$
- da x in $\alpha[x/0]$ nicht vorkommt, gilt

$$\alpha[x/0]^I = \alpha[x/0]^J,$$

falls I, J Belegungen mit $y^I = y^J$ für alle $y \neq x$

Es gilt stets: $\alpha[x/0]^I = \alpha^J$, wobei $J = I[x := 0]$

$$y^{I[x:=0]} = \begin{cases} y^I & : \text{ falls } y \neq x \\ 0 & : \text{ falls } y = x \end{cases}$$

$\alpha[x/0]$ entsteht aus α durch folgende Schritte:

1. streiche x aus jeder Klausel $\dots \vee x \vee \dots$, die das positive Literal x enthält
2. streiche jede Klausel $\dots \vee \neg x \vee \dots$, die das negative Literal $\neg x$ enthält

$\alpha[x/1]$ entsteht aus α durch folgende Schritte:

1. streiche $\neg x$ aus jeder Klausel $\dots \vee \neg x \vee \dots$, die das negative Literal $\neg x$ enthält
2. streiche jede Klausel $\dots \vee x \vee \dots$, die das positive Literal x enthält

Sei α eine KNF-Formel und x ein Atom.

α ist erfüllbar

gdw $\alpha[x/0]$ ist erfüllbar oder $\alpha[x/1]$ ist erfüllbar

... Basis für naiven KNF-SAT-Beweiser ...
mittels Backtracking

IF α enthält die leere Klausel THEN return “false” FI

IF $\alpha = \mathbf{true}$ THEN return “true” FI

wähle ein Atom x , das in α vorkommt

IF $Sat(\alpha[x/0])$

Splitting-Regel

THEN return “true”

ELSE return $Sat(\alpha[x/1])$

FI

IF α enthält die leere Klausel THEN return "false" FI
IF $\alpha = \mathbf{true}$ THEN return "true" FI

... Kriterien, die das binäre Verzweigen mit der Splitting-Regel verhindern ...

wähle ein Atom x , das in α vorkommt

IF $\mathbf{Sat}(\alpha[x/0])$
 THEN return "true"
 ELSE return $\mathbf{Sat}(\alpha[x/1])$
FI

Einheitsklausel (engl. unit clause):

Klausel bestehend aus genau einem Literal

Sei α eine KNF-Formel, x ein Atom.

- Falls α die positive Einheitsklausel x enthält, d.h., $\alpha = \dots \wedge x \wedge \dots$, dann gilt:
 α ist erfüllbar gdw $\alpha[x/1]$ ist erfüllbar
- Falls α die negative Einheitsklausel $\neg x$ enthält, d.h., $\alpha = \dots \wedge \neg x \wedge \dots$, dann gilt:
 α ist erfüllbar gdw $\alpha[x/0]$ ist erfüllbar

IF α enthält die leere Klausel THEN return "false" FI
IF $\alpha = \mathbf{true}$ THEN return "true" FI

wende so oft wie möglich die **Unit-Regel** an,
um das binäre Verzweigen mit der Splitting-Regel
zu verhindern

wähle ein Atom x , das in α vorkommt

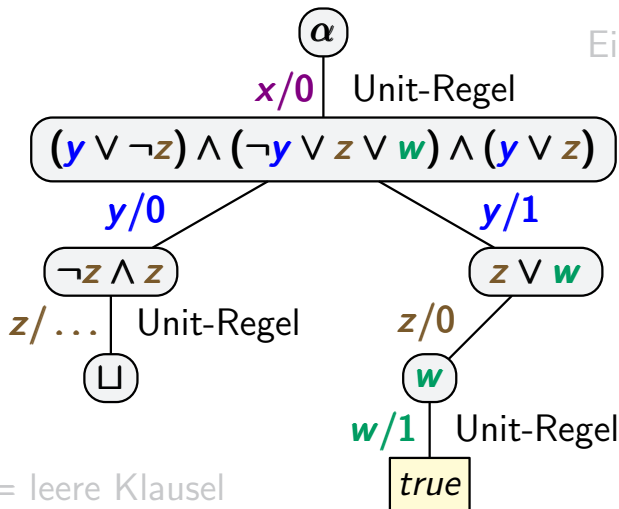
IF $Sat(\alpha[x/0])$
 THEN return "true"
 ELSE return $Sat(\alpha[x/1])$
FI

Beispiel: KNF-SAT mit Unit-Regel

168

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

↑
Einheitsklausel



Sei α eine KNF-Formel und x ein Atom, das in α wenigstens einmal vorkommt.

- x kommt **nur positiv** in α vor, falls das negative Literal $\neg x$ in keiner Klausel von α enthalten ist.
- x kommt **nur negativ** in α vor, falls das positive Literal x in keiner Klausel von α enthalten ist.

Kommt x in α nur positiv vor, so gilt:

α erfüllbar gdw $\alpha[x/1]$ erfüllbar

Kommt x in α nur negativ vor, so gilt:

α erfüllbar gdw $\alpha[x/0]$ erfüllbar

IF α enthält die leere Klausel THEN return “false” FI

IF $\alpha = \mathbf{true}$ THEN return “true” FI

so oft wie möglich: wende die **Unit-Regel** oder
die **Pure-Literal-Regel** an

wähle ein Atom x , das in α vorkommt

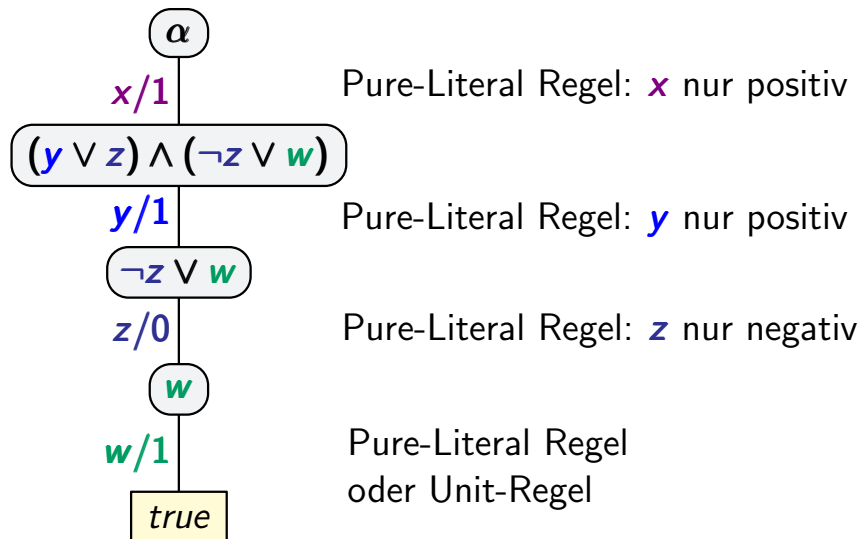
IF $Sat(\alpha[x/0])$

THEN return “true”

ELSE return $Sat(\alpha[x/1])$

FI

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



$$\begin{aligned}\kappa &= L_1 \vee \dots \vee L_n \hat{=} \{L_1, \dots, L_n\} \\ \lambda &= L'_1 \vee \dots \vee L'_m \hat{=} \{L'_1, \dots, L'_m\}\end{aligned} \quad \text{Klauseln}$$

Falls $\kappa \subseteq \lambda$, so gilt:

$$\kappa \Vdash \lambda \quad \text{und} \quad \kappa \wedge \lambda \equiv \kappa$$

Ist α eine KNF-Formel der Form

$$\alpha = \dots \wedge \kappa \wedge \lambda \wedge \dots,$$

wobei κ Teilklausel von λ ist, dann gilt:

$$\alpha \equiv \alpha \setminus \{\lambda\} = \dots \wedge \kappa \wedge \dots$$

α nach Streichen der Klausel λ

IF α enthält die leere Klausel THEN return “false” FI

IF $\alpha = \mathbf{true}$ THEN return “true” FI

so oft wie möglich: wende die **Unit-Regel** oder
die **Pure-Literal-Regel** an

vereinfache α mit der **Subsumierungsregel**

wähle ein Atom x , das in α vorkommt

IF $Sat(\alpha[x/0])$

THEN return “true”

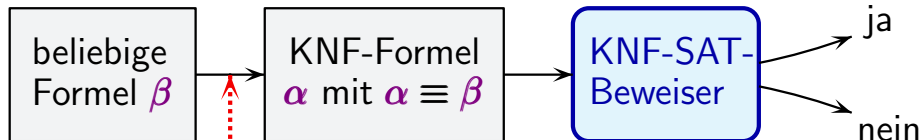
ELSE return $Sat(\alpha[x/1])$

FI

DP-Algorithmus ist ein KNF-SAT-Beweiser, d.h.,
löst das Problem:

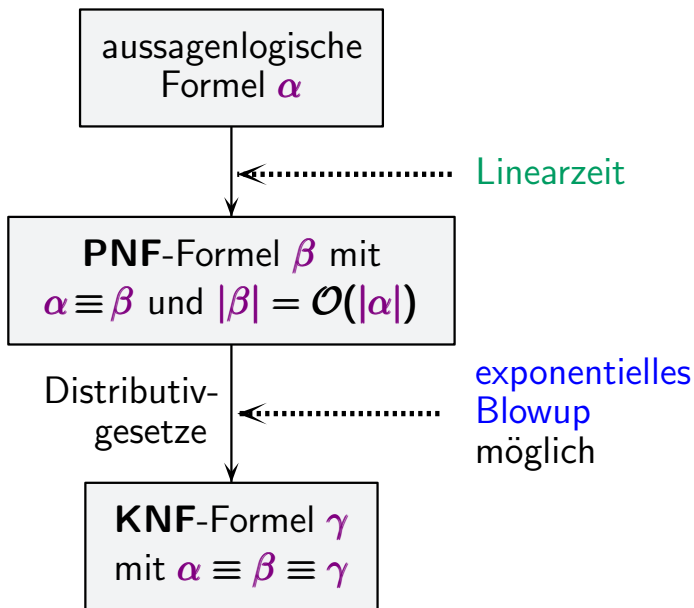
gegeben: KNF-Formel α

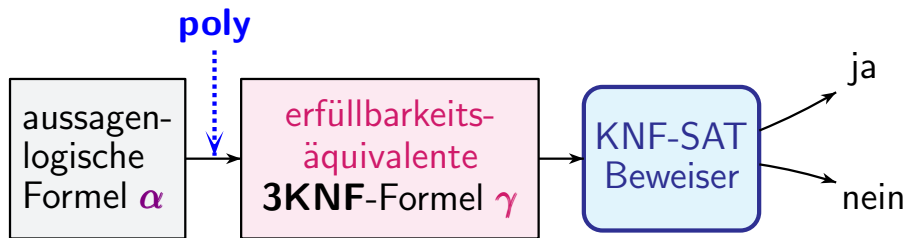
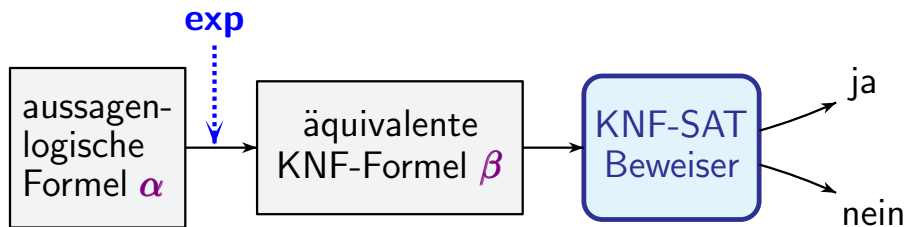
gefragt: ist α erfüllbar ?



exponentielles
Blowup
möglich

z.B. DP-Algorithmus





$$|\gamma| = \mathcal{O}(|\alpha|)$$

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

Beispiele:

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

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

$\perp = \textit{false}$

← 1 Klausel mit 0 Literalen

\textit{true}

← KNF-Formel mit 0 Klauseln

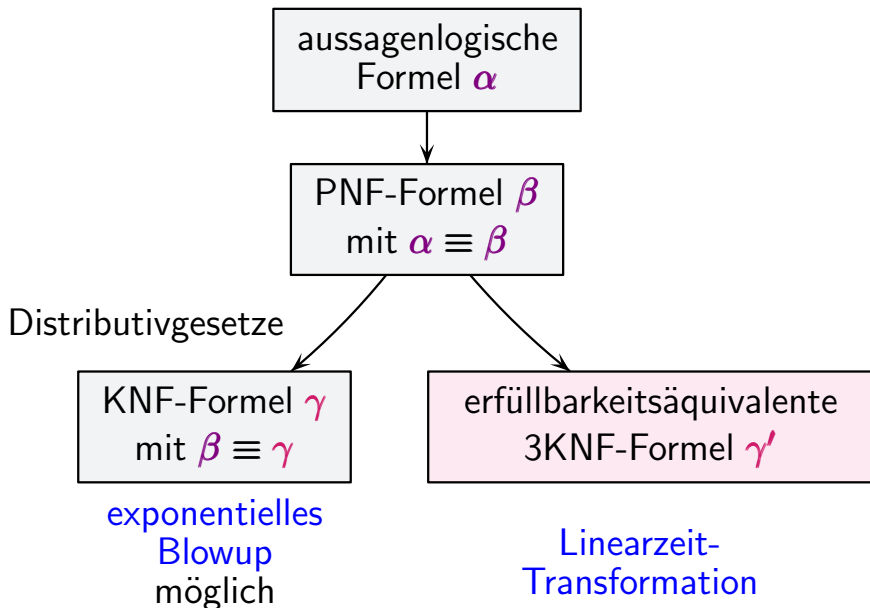
Formeln α , β heißen erfüllbarkeitsäquivalent, falls
entweder α und β erfüllbar sind
oder α und β unerfüllbar sind.

z.B. x und $\neg x$ sind erfüllbarkeitsäquivalent

Frage: wieviele Erfüllbarkeitsäquivalenzklassen gibt es ?

Antwort: **2**

1. die Klasse der erfüllbaren Formeln
2. die Klasse der unerfüllbaren Formeln



1. Schritt: erstelle eine zu α äquivalente PNF-Formel α_{PNF} und vereinfache α_{PNF} gemäß der Regeln

$$\beta \wedge \text{true} \equiv \text{true} \wedge \beta \equiv \beta$$

$$\beta \vee \text{true} \equiv \text{true} \vee \beta \equiv \text{true}$$

+ analoge
Regeln
für *false*

Falls $\alpha_{\text{PNF}} \in \{\text{true}, \text{false}\}$, dann gib $\gamma = \alpha_{\text{PNF}}$ als gesuchte 3KNF-Formel zurück.

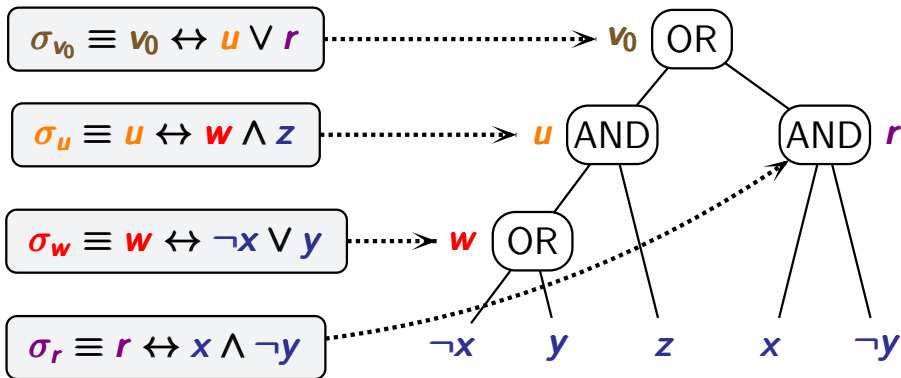
Andernfalls ist α_{PNF} eine PNF-Formel, in der die Konstanten *true* und *false* nicht vorkommen.
Fahre mit dem 2. Schritt fort.

2. Schritt: ...

1. Schritt: erstelle eine zu α äquivalente PNF-Formel α_{PNF} , in der *true* und *false* nicht vorkommen
2. Schritt: konstruiere den **Syntaxbaum** von α_{PNF}
3. Schritt: erstelle anhand des Syntaxbaums eine zu α_{PNF} erfüllbarkeitsäquivalente 3KNF-Formel

3KNF für $\alpha_{\text{PNF}} = ((\neg x \vee y) \wedge z) \vee (x \wedge \neg y)$

355



3KNF-Formel α_{3KNF} mit den Atomen x, y, z, w, u, v_0, r

$$\alpha_{\text{3KNF}} = v_0 \wedge \underbrace{\sigma_{v_0} \wedge \sigma_u \wedge \sigma_r \wedge \sigma_w}_{\text{3KNF-Formeln, jeweils bestehend aus 3 Klauseln}}$$

Einheitsklausel