

Korrektheit der Transformation. Hierzu ist zu zeigen, dass die Kosten für die Konstruktion von α_{3KNF} linear in $|\alpha|$ sind und dass α_{3KNF} zu α erfüllbarkeitsäquivalent ist.

Zunächst zur Länge von α_{3KNF} . Offenbar enthält α_{3KNF} höchstens

$$3 * |\alpha_{PNF}| + 1 = \mathcal{O}(|\alpha|)$$

Klauseln. Beachte, dass die Anzahl an inneren Knoten des Syntaxbaums mit der Anzahl an Konjunktionen und Disjunktionen in α_{PNF} übereinstimmt. Die Anzahl an \wedge - und \vee -Teilformeln von α_{PNF} ist durch $|\alpha_{PNF}|$ nach oben beschränkt. Da für jeden inneren Knoten v die Formel σ_v aus drei Klauseln besteht, ist die Klauselanzahl von α_{3KNF} durch $3 \cdot |\alpha_{PNF}| + 1$ beschränkt, wobei der Summand “+1” für die Einheitsklausel v_0 steht.

Da α_{PNF} sowie der Syntaxbaum T von α_{PNF} in linearer Zeit aus α konstruiert werden können und sich α_{3KNF} aus T im Wesentlichen durch Hinschreiben der oben explizit angegebenen 3KNF-Formeln σ_v ergibt, kann α_{3KNF} aus α in linearer Zeit konstruiert werden.

Nun zur Erfüllbarkeitsäquivalenz von α und α_{3KNF} . Wie oben erwähnt, bezeichnet β_v die durch Knoten v dargestellte Teilformel von α_{PNF} . Es gilt also $\beta_v = v$ für jedes Blatt v . Ist v ein innerer Knoten, der mit dem Operator *op* beschriftet ist und dessen Söhne w und u sind, so ist $\beta_v = \beta_w \text{ op } \beta_u$. Ferner ist $\alpha_{PNF} = \beta_{v_0}$ für den Wurzelknoten v_0 .

1. Wir nehmen an, dass α erfüllbar ist und zeigen die Erfüllbarkeit von α_{3KNF} . Sei I eine erfüllende Belegung für α . Wir erweitern nun I zu einer erfüllenden Belegung J für α_{3KNF} wie folgt:

- $x_i^J \stackrel{\text{def}}{=} x_i^I$ für $1 \leq i \leq n$,
- $v^J \stackrel{\text{def}}{=} \beta_v^I$ für jeden inneren Knoten v des Syntaxbaums von α_{PNF} .

Offenbar gilt $v^J = \beta_v^I$ für jeden Knoten v des Syntaxbaums. (Für die inneren Knoten gilt dies nach Definition von J . Für die Blätter ist dies offensichtlich, da diese für Literale x_i oder $\neg x_i$ stehen und die Belegungen I und J auf $\{x_1, \dots, x_n\}$ übereinstimmen.) Zu zeigen ist nun, dass $\sigma_v^J = 1$ für jeden inneren Knoten v und dass $v_0^J = 1$.

Zunächst zum Nachweis, dass $\sigma_v^J = 1$ für alle inneren Knoten v . Wir erläutern dies am Beispiel eines AND-Knotens v , d.h., v ist mit dem Operator \wedge beschriftet. Seien w und u die beiden Söhne von v . Dann gilt:

$$\begin{aligned} v^J &= \beta_v^I = 1 \\ \text{gdw } (\beta_w \wedge \beta_u)^I &= 1 \\ \text{gdw } \beta_w^I = \beta_u^I &= 1 \\ \text{gdw } w^J = u^J &= 1 \\ \text{gdw } (w \wedge u)^J &= 1 \end{aligned}$$

und somit $\sigma_v^J = (v \leftrightarrow (w \wedge u))^J = 1$. Die Argumentation für die mit \vee beschrifteten

Knoten ist analog. Weiter ist

$$\begin{aligned}
 v_0^J &= \beta_{v_0}^I && \text{(nach Definition von J)} \\
 &= \alpha_{\text{PNF}}^I && \text{(da } \alpha_{\text{PNF}} = \beta_{v_0}\text{)} \\
 &= \alpha^I && \text{(da } \alpha \equiv \alpha_{\text{PNF}}\text{)} \\
 &= 1 && \text{(da I erfüllend für } \alpha\text{)}
 \end{aligned}$$

Also ist J eine erfüllende Belegung für $\alpha_{3\text{KNF}} = v_0 \wedge \bigwedge_{v \in \text{In}(T)} \sigma_v$.

2. Wir setzen nun die Erfüllbarkeit von $\alpha_{3\text{KNF}}$ voraus und weisen nach, dass auch α erfüllbar ist. Sei J eine erfüllende Belegung für $\alpha_{3\text{KNF}}$. Wir zeigen, dass J zugleich eine erfüllende Belegung für α ist. Hierzu benutzen wir ein induktives Argument, um folgende Aussage (*) nachzuweisen:

$$v^J = \beta_v^J \quad \text{für jeden Knoten } v \text{ von } T \quad (*)$$

Aussage (*) weisen wir durch Induktion nach der Höhe des Teilbaums von v nach.

- Im Induktionsanfang sind die Blätter von T zu betrachten, da genau deren Teilbäume die Höhe 0 haben. Für die Blätter ist die Aussage $v^J = \beta_v^J$ klar, da das Blatt v mit dem Literal β_v identifiziert wird.
- Im Induktionsschritt betrachten wir einen inneren Knoten v , so dass dessen Teilbaum die Höhe h hat. Dann ist $h \geq 1$ und die Teilbäume der beiden Söhne w und u von v haben die Höhe $\leq h-1$. Die Induktionsvoraussetzung kann also auf w und u angewandt werden. Dies liefert

$$w^J = \beta_w^J \quad \text{und} \quad u^J = \beta_u^J.$$

Wir nehmen an, dass Knoten v die Beschriftung op hat, wobei $op \in \{\wedge, \vee\}$. Für die durch Knoten v dargestellte Teilformel β_v von α_{PNF} gilt also:

$$\beta_v = \beta_w op \beta_u$$

Wegen $\alpha_{3\text{KNF}}^J = 1$, sind alle Klauseln von $\alpha_{3\text{KNF}}$ unter J wahr. Daher ist auch σ_v unter J wahr. Da σ_v zu $v \leftrightarrow (w op u)$ äquivalent ist, gilt:

$$(v \leftrightarrow (w op u))^J = \sigma_v^J = 1$$

Wegen $\beta_v = \beta_w op \beta_u$ und da $w^J = \beta_w^J$ und $u^J = \beta_u^J$ (nach Induktionsvoraussetzung, siehe oben) folgt hieraus:

$$v^J = (w op u)^J \stackrel{\text{IV}}{=} (\beta_w op \beta_u)^J = \beta_v^J$$

Mit Hilfe von (*) angewandt auf den Wurzelknoten $v = v_0$ erhalten wir:

$$\begin{aligned}
 \alpha^J &= \alpha_{\text{PNF}}^J && \text{(da } \alpha \equiv \alpha_{\text{PNF}}\text{)} \\
 &= \beta_{v_0}^J && \text{(da } \beta_{v_0} = \alpha_{\text{PNF}}\text{)} \\
 &= v_0^J && \text{(Aussage (*))} \\
 &= 1 && \text{(da } v_0 \text{ Klausel von } \alpha_{3\text{KNF}} \text{ und } \alpha_{3\text{KNF}}^J = 1\text{)}
 \end{aligned}$$

Damit ist die Erfüllbarkeit von α gezeigt.

□

Beispiel 4.36 (Formel \rightsquigarrow erfüllbarkeitsäquivalente 3KNF). Unser Ausgangspunkt ist die PNF-Formel

$$\alpha = \alpha_{\text{PNF}} = ((\neg x \vee y) \wedge z) \vee (x \wedge \neg y).$$

Wir betrachten nun den Syntaxbaum für α , wobei die Blätter für die in α vorkommenden Literale stehen und die inneren Knoten für je eine der Teilformeln

$$\begin{aligned} \beta_{v_0} &= ((\neg x \vee y) \wedge z) \vee (x \wedge \neg y) = \beta_u \vee \beta_r = \alpha \\ \beta_r &= x \wedge \neg y \\ \beta_u &= (\neg x \vee y) \wedge z = \beta_w \wedge z \\ \beta_w &= \neg x \vee y \end{aligned}$$

Der Wurzelknoten ist also v_0 und stellt die gesamte Formel α dar. Der Syntaxbaum ist in Abbildung 38 skizziert, wobei die Knotenbeschriftung AND bzw. OR für \wedge bzw. \vee steht. Die für α konstruierte erfüllbarkeitsäquivalente 3KNF-Formel $\alpha_{3\text{KNF}}$ verwendet die ursprünglichen Atome x, y, z sowie zusätzlich die Atome v_0, w, u, r und hat die Gestalt:

$$\alpha_{3\text{KNF}} = v_0 \wedge \sigma_{v_0} \wedge \sigma_w \wedge \sigma_u \wedge \sigma_r,$$

wobei $\sigma_{v_0}, \sigma_w, \sigma_u$ und σ_r 3KNF-Formel bestehend aus je drei Klauseln sind, die die Semantik der Knoten v_0, w, u, r widerspiegeln, also $\sigma_{v_0} \equiv v_0 \leftrightarrow (u \vee r)$, $\sigma_u \equiv u \leftrightarrow (w \wedge z)$, $\sigma_w \equiv w \leftrightarrow (\neg x \vee y)$ und $\sigma_r \equiv r \leftrightarrow (x \wedge \neg y)$. ■

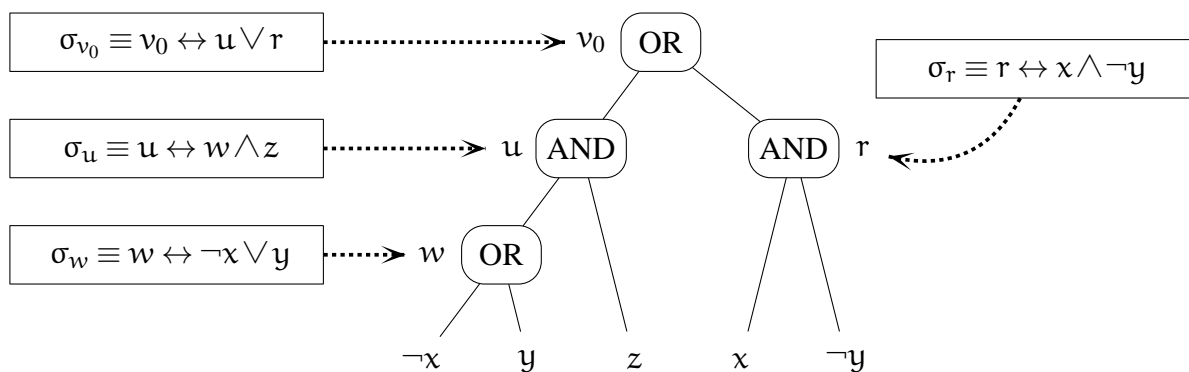


Abbildung 38: Syntaxbaum für $((\neg x \vee y) \wedge z) \vee (x \wedge \neg y)$

4.4 Resolution

Der Resolutionskalkül leistet einen entscheidenden Beitrag für die Logikprogrammierung und andere Bereiche der Informatik (KI, regelbasierte Systeme, etc.). Im Gegensatz zu den SAT-Beweisern, die wir in den vorangegangenen Abschnitten diskutiert haben, zielt Resolution auf

den Nachweis der Unerfüllbarkeit einer KNF-Formel (oder allgemeiner Klauselmenge) ab. Wesentliche Idee der Resolution ist es, die logische Folgerbarkeit der leeren Klausel $false = \perp$ durch eine Kette von elementaren Folgerungsschritten zu belegen. Man spricht in diesem Kontext auch von einer *Widerlegung* für die Eingabeformel. Logische Folgerbarkeit $\alpha \Vdash \beta$ für beliebige Formeln α kann mittels Resolution über den Umweg einer Zerlegung von $\neg\beta$ in Klauseln $\kappa_1, \dots, \kappa_m$ durch eine Widerlegung für $\alpha \wedge \kappa_1 \wedge \dots \wedge \kappa_m$ nachgewiesen werden.

Mengenschreibweise von Klauseln und KNF-Formeln. Im Folgenden identifizieren wir Klauseln mit der betreffenden Literalmenge. Ist also $\kappa = L_1 \vee \dots \vee L_k$ eine Klausel, so wird κ als Literalmenge $\{L_1, \dots, L_k\}$ aufgefasst. Damit werden Klauseln, in denen ein Literal mehrfach vorkommt, durch äquivalente Klauseln ohne mehrfach vorkommende Literale ersetzt. Etwa $x \vee x \vee \neg y$ wird durch $\{x, x, \neg y\} = \{x, \neg y\}$ dargestellt. Ferner abstrahiert die Mengenschreibweise von der Reihenfolge der Literale. Dies ist aufgrund der Äquivalenzregeln für die Assoziativität und Kommutativität von Disjunktionen gerechtfertigt. Die leere Klausel \perp entspricht nun der leeren Menge. Analog verfahren wir mit aussagenlogischen KNF-Formeln, die wir als Klauselmengen auffassen. Ist etwa $\alpha = \kappa_1 \wedge \dots \wedge \kappa_m$, so wird α mit der Klauselmenge $\{\kappa_1, \dots, \kappa_m\}$ identifiziert.

Man beachte, dass die Mengenschreibweise für Klauseln einer Disjunktion entspricht, die Mengenschreibweise für KNF-Formeln einer Konjunktion. Daher erhalten wir gegensätzliche Interpretationen der leeren Menge:¹⁶

- Die leere Klausel (also die leere Literalmenge) \perp steht für *false* und ist daher unerfüllbar.
- Die leere Klauselmenge \emptyset steht für *true* und ist daher gültig.

Damit ergibt sich auch der Unterschied zwischen der leeren Klauselmenge \emptyset und der einelementigen Klauselmenge $\{\perp\}$. Während die leere Klauselmenge \emptyset stets gültig ist, ist $\{\perp\}$ stets unerfüllbar. Etwas gewöhnungsbedürftig sind Einheitsklauseln, für die nun die Schreibweisen $\{x\}$ und x bzw. $\{\neg x\}$ und $\neg x$ gleichwertig sind. Analoges gilt für KNF-Formeln mit nur einer Klausel. Für diese ist die Schreibweise κ (ohne Mengenklammer) gleichbedeutend mit der Mengenschreibweise $\{\kappa\}$. Im Folgenden verwenden wir die Formel- oder Mengennotation; je nachdem, welche Sichtweise einfacher ist. (An vielen Stellen ist die Wahl jedoch willkürlich.)

Die Idee des Resolutionskalküls beruht darauf, durch syntaktische Transformationen logische Schlussfolgerungen zu ziehen. Hierzu wird folgende Beobachtung eingesetzt: Sind $\kappa = x \vee \lambda_1$ und $\tau = \neg x \vee \lambda_2$ zwei Klauseln der als wahr angenommenen Grundmenge, so kann auf die logische Folgerbarkeit der Klausel $\lambda_1 \vee \lambda_2$ geschlossen werden.

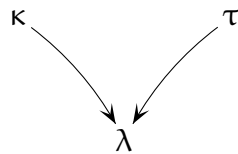
Definition 4.37 (Resolvent). Seien κ und τ Klauseln und L ein Literal, so dass $L \in \kappa$ und das negierte Literal \bar{L} in τ enthalten sind.¹⁷ Dann heißt die Klausel

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

der Resolvent von κ und τ über L . Man spricht auch von dem L -Resolventen oder kurz einem Resolventen, von κ und τ . Die Klauseln κ und τ werden in diesem Kontext auch *Elternklauseln* genannt.

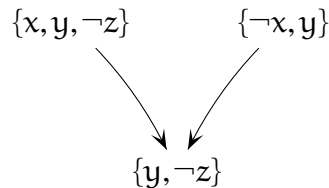
¹⁶Aus diesem Grund ist es üblich ein Sondersymbol wie \perp für die leere Klausel – anstelle des sonst üblichen Symbols \emptyset für die leere Menge – zu verwenden.

¹⁷Zur Erinnerung: $\bar{\bar{x}} = x$ und $\overline{\neg x} = x$. \bar{L} ist also dasjenige Literal, welches zu $\neg L$ äquivalent ist.



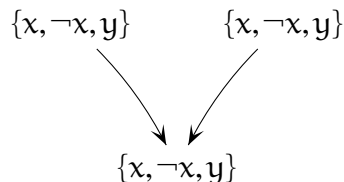
Oftmals verwendet man eine graphische Notation wie in dem Bild oben, um anzudeuten, dass die Klausel λ ein Resolvent der beiden Klauseln κ und τ ist. In Darstellungen durch gerichtete Graphen sind die Knoten mit Klauseln beschriftet. Kanten stehen für die Resolventenbildung. ■

Beispiel 4.38 (Resolventen). Z.B. ist $y \vee \neg z$ ein Resolvent aus $x \vee y \vee \neg z$ und $\neg x \vee y$.



Die Klausel $y \vee \neg z$ ergibt sich durch Streichen des positiven Literals x aus der Klausel $x \vee y \vee \neg z$ und des negativen Literals $\neg x$ aus $\neg x \vee y$ und anschließende disjunktive Verknüpfung der verkürzten Klauseln. Die Klauseln $x \vee y \vee \neg z$ und $x \vee \neg z \vee w$ haben jedoch keine Resolventen, da sie keine zueinander komplementäre Literale enthalten. ■

Durch Resolventenbildung kann die leere Klausel \square entstehen; nämlich dann, wenn der Resolvent der beiden Einheitsklauseln x und $\neg x$ gebildet wird. Zwei Klauseln können durchaus auch zwei oder mehrere Resolventen haben; nämlich dann, wenn sie mehrere zueinander komplementäre Literale enthalten. Z.B. sind $x \vee \neg x \vee z$ und $y \vee \neg y \vee z$ Resolventen der Klauseln $x \vee y \vee z$ und $\neg x \vee \neg y$. In solchen Fällen sind jedoch alle Resolventen tautologisch. Resolventen einer tautologischen Klausel sind zwar möglich, aber langweilig. Z.B.:



Das folgende Lemma zeigt, dass Resolventen einer KNF-Formel logische Folgerungen sind.

Lemma 4.39 (Resolventenlemma (1. Teil)). Sei α eine KNF-Formel und λ ein Resolvent zweier Klauseln von α . Dann gilt $\alpha \models \lambda$.

Beweis. Sei $\lambda = \kappa \setminus \{x\} \cup \tau \setminus \{\neg x\}$, wobei κ, τ Klauseln von α sind und $x \in \kappa, \neg x \in \tau$. Sei I ein Modell für α . Zu zeigen ist, dass $\lambda^I = 1$; also dass eines der Literale von λ den Wahrheitswert 1 unter I hat.

1. Fall: $x^I = 1$. Wegen $\neg x \in \tau$ und $\tau^I = 1$ gibt es ein Literal $L \in \tau \setminus \{\neg x\}$, welches unter I den Wahrheitswert 1 hat. Dieses Literal L ist auch in λ enthalten.

2. Fall: $x^I = 0$. In diesem Fall liegt eine zum ersten Fall analoge Situation für κ vor. Wegen $\kappa^I = 1$ gibt es ein anderes in κ vorkommendes Literal $L \neq x$, das unter I den Wahrheitswert 1 hat. Dieses Literal L liegt in λ . □

Lemma 4.40. *Ist α eine KNF-Formel, so gilt $\alpha \wedge \lambda \equiv \alpha$ für jeden Resolventen λ zweier Klauseln von α .*

Beweis. Sei λ ein Resolvent von α . Es ist klar, dass jedes Modell für $\alpha \wedge \lambda$ zugleich ein Modell für α ist. Umgekehrt gilt $\lambda^I = 1$ für jedes Modell I für α , da $\alpha \models \lambda$ (Resolventenlemma). Also ist I ein Modell für $\alpha \wedge \lambda$. \square

Bezeichnung 4.41 (Resolventenmenge, Resolutionsabschluss). Sei α eine KNF-Formel. Dann bezeichnet die Resolventenmenge

$$Res(\alpha) = \{ \lambda : \lambda \text{ ist Resolvent zweier Klauseln in } \alpha \}$$

diejenige Menge bestehend aus allen Resolventen, die aus den Klauseln von α gebildet werden können. Der Resolutionsabschluss $Res^*(\alpha)$ von α ergibt sich durch sukzessive Resolventenbildung:

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

wobei $Res^0(\alpha) = \alpha$ und $Res^{i+1}(\alpha) = Res^i(\alpha) \cup Res(Res^i(\alpha))$ für $i = 1, 2, \dots$ \blacksquare

Beispiel 4.42 (Resolutionsabschluss). Für $\alpha = (x \vee \neg y \vee z) \wedge (y \vee z) \wedge (\neg x \vee z) \wedge \neg z$ erhalten wir (wobei die Klauseln als Literalmenge angegeben werden):

$$Res^0(\alpha) = \{ \{x, \neg y, z\}, \{y, z\}, \{\neg x, z\}, \{\neg z\} \}$$

$$Res^1(\alpha) = Res^0(\alpha) \cup \{ \{x, z\}, \{\neg y, z\}, \{x, \neg y\}, \{y\}, \{\neg x\} \}$$

$$Res^2(\alpha) = Res^1(\alpha) \cup \{ \{z\}, \{x\}, \{\neg y\} \}$$

$$Res^3(\alpha) = Res^2(\alpha) \cup \{ \square \}$$

und $Res^k(\alpha) = Res^3(\alpha) = Res^*(\alpha)$ für alle $k \geq 4$. \blacksquare

Offenbar gilt $Res^0(\alpha) \subseteq Res^1(\alpha) \subseteq Res^2(\alpha) \subseteq \dots \subseteq Res^*(\alpha) \subseteq 2^{\mathfrak{L}}$, wobei \mathfrak{L} die Menge aller Literale bezeichnet, die in wenigstens einer der Klauseln von α vorkommen. Da \mathfrak{L} (und somit auch die Potenzmenge $2^{\mathfrak{L}}$) endlich ist, ist die Folge der Resolventenmengen $Res^i(\alpha)$ ab einem gewissen Index stationär. Wir halten diese Beobachtung in folgendem Lemma fest:

Lemma 4.43 (Endlichkeit des Resolutionsabschlusses). *Für jede KNF-Formel α gibt es eine natürliche Zahl k , so dass*

$$Res^k(\alpha) = Res^{k+1}(\alpha) = Res^{k+2}(\alpha) = \dots = Res^*(\alpha).$$

Eine sehr grobe obere Schranke für die in Lemma 4.43 genannte Zahl k ist $|2^{\mathfrak{L}}| = 2^{2^n} = 4^n$, wobei n für die Anzahl an Aussagensymbolen steht, die in α vorkommen.

Durch Induktion nach i kann man mit Hilfe des Resolventenlemmas zeigen, dass alle Klauseln in $Res^i(\alpha)$ logische Folgerungen von α sind. Hieraus folgt die Korrektheit der Resolution als Kalkül für logische Folgerbarkeit:

Lemma 4.44 (Resolventenlemma (2. Teil)). Sei α eine KNF-Formel. Dann gilt $\alpha \models \lambda$ für alle $\lambda \in \text{Res}^*(\alpha)$.

Insbesondere ist die Resolution als Verfahren für den Nachweis der Unerfüllbarkeit einer KNF-Formel korrekt. Man spricht auch von Widerlegungskorrektheit:

Corollar 4.45 (Widerlegungskorrektheit). Falls $\perp \in \text{Res}^*(\alpha)$, so ist α unerfüllbar.

Beweis. Falls $\perp \in \text{Res}^*(\alpha)$, so gilt $\alpha \models \perp = \text{false}$ (zweiter Teil des Resolventenlemmas). Hieraus folgt die Unerfüllbarkeit von α . \square

Wir werden später sehen, dass auch die Umkehrung gilt. Zunächst geben wir jedoch eine alternative Charakterisierung des Resolutionsabschlusses an.

Definition 4.46 ((Resolutions-)Herleitung). Sei α eine KNF-Formel, aufgefasst als Klauselmengemenge, und λ eine Klausel. Eine *Resolutionsherleitung* (kurz *Herleitung*) von λ aus α ist eine Folge von Klauseltripeln

$$\langle \kappa_1, \tau_1, \lambda_1 \rangle \langle \kappa_2, \tau_2, \lambda_2 \rangle \dots \langle \kappa_m, \tau_m, \lambda_m \rangle,$$

so dass $\lambda_m = \lambda$ und so dass für alle $i \in \{1, 2, \dots, m\}$ gilt:

$$\lambda_i \text{ ist Resolvent von } \kappa_i \text{ und } \tau_i, \text{ und } \kappa_i, \tau_i \in \alpha \cup \{\lambda_1, \lambda_2, \dots, \lambda_{i-1}\}.$$

m wird die Länge der Herleitung genannt. Da die jeweils ersten beiden Klauseln κ_i, τ_i der Herleitungstriple lediglich die Funktion haben anzugeben, woraus die hergeleitete Klausel λ_i entstanden ist, verzichtet man oft auf deren Angabe und nennt die Klauselfolge $\lambda_1 \lambda_2 \dots \lambda_m$ eine Herleitung von $\lambda_m = \lambda$ aus α . Eine Klausel λ ist aus α herleitbar, wenn entweder $\lambda \in \alpha$ oder wenn es eine Herleitung der Länge $m \geq 1$ für λ aus α gibt. Alle Klauseln aus α werden als in Herleitungen der Länge 0 erzeugbar angesehen. \blacksquare

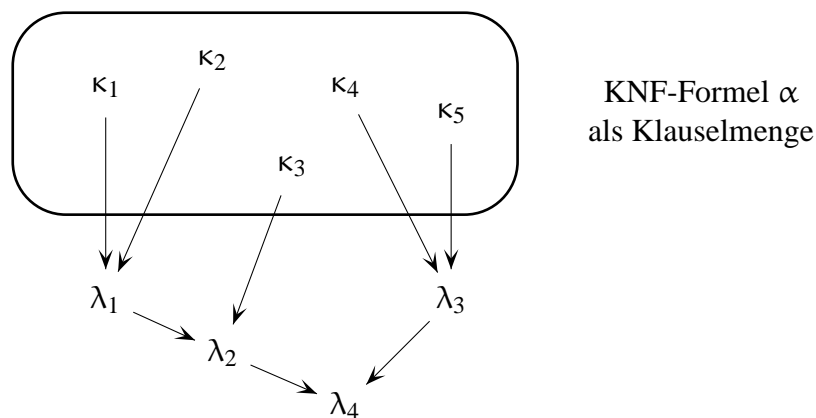


Abbildung 39: Graphische Darstellung einer Resolutionsherleitung

Abbildung 39 zeigt wie Herleitungen als azyklische Digraphen skizziert werden können. Die in einer Herleitung benötigten Klauseln aus α haben keine Vorgänger. Alle über ein oder mehrere Resolutionsschritte hergeleiteten Klauseln haben genau zwei Vorgängerknoten; nämlich die beiden Elternklauseln. Das Bild in Abbildung 39 könnte etwa für folgende Herleitung stehen:

$$\langle \kappa_1, \kappa_2, \lambda_1 \rangle \quad \langle \lambda_1, \kappa_3, \lambda_2 \rangle \quad \langle \kappa_4, \kappa_5, \lambda_3 \rangle \quad \langle \lambda_2, \lambda_3, \lambda_4 \rangle$$

Beispiel 4.47 (Resolutionsherleitung). Ist $\alpha = (x \vee y) \wedge (\neg x \vee \neg z) \wedge (y \vee z)$, so ist

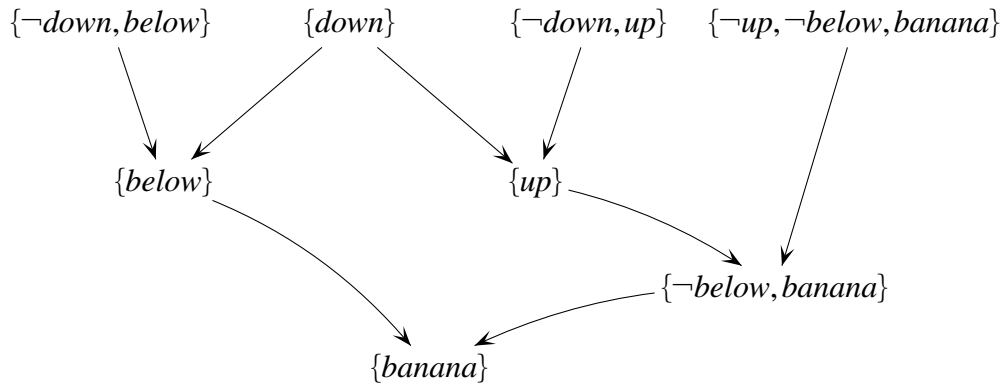
$$\langle \{x, y\}, \{\neg x, \neg z\}, \{y, \neg z\} \rangle \quad \langle \{y, z\}, \{y, \neg z\}, \{y\} \rangle$$

eine Herleitung der Einheitsklausel $\{y\}$ aus α . Diese hat die Länge 2. ■

Beispiel 4.48 (Resolutionsherleitung (Affen-Bananen-Problem)). Wir kehren zu dem Affen-Bananen-Problem zurück. Ausgangspunkt ist die KNF-Formel $\alpha = \kappa_1 \wedge \kappa_2 \wedge \kappa_3 \wedge \kappa_4$ mit den Hornklauseln $\kappa_1 = \neg \text{down} \vee \text{below}$, $\kappa_2 = \neg \text{down} \vee \text{up}$, $\kappa_3 = \neg \text{up} \vee \neg \text{below} \vee \text{banana}$ und der Einheitsklausel $\kappa_4 = \text{down}$. Die folgende Skizze stellt die Resolutionsherleitung

$$\langle \kappa_2, \kappa_4, \lambda_1 \rangle \quad \langle \kappa_1, \kappa_4, \lambda_2 \rangle \quad \langle \kappa_3, \lambda_1, \lambda_3 \rangle \quad \langle \lambda_2, \lambda_3, \lambda_4 \rangle$$

dar, wobei $\lambda_1 = \{\text{up}\}$, $\lambda_2 = \{\text{below}\}$, $\lambda_3 = \{\neg \text{below}, \text{banana}\}$ und $\lambda_4 = \{\text{banana}\}$. Aus dem zweiten Teil des Resolventenlemmas folgt die für den Affen erfreuliche Aussage, dass er die Bananen erreichen kann. ■



Bemerkung 4.49 (Irrelevante Resolutionsschritte). Es ist klar, dass in einer Herleitung alle Resolutionsschritte weggelassen werden können, die für die Generierung der letzten Klausel λ_m nicht benötigt werden. Z.B. ist

$$\langle \{x, y\}, \{\neg x\}, \{y\} \rangle \quad \langle \{z, \neg w\}, \{w\}, \{z\} \rangle \quad \langle \{y\}, \{\neg y\}, \sqcup \rangle$$

eine Herleitung aus $\alpha = (x \vee y) \wedge \neg x \wedge \neg y \wedge (z \vee \neg w) \wedge w$, die zu

$$\langle \{x, y\}, \{\neg x\}, \{y\} \rangle \quad \langle \{y\}, \{\neg y\}, \sqcup \rangle$$

verkürzt werden kann, da zur Herleitung der leeren Klausel die im zweiten Resolutionsschritt generierte Klausel $\{z\}$ nicht benötigt wird. In diesem Sinn kann man stets annehmen, dass Herleitungen unverkürzbar sind. ■

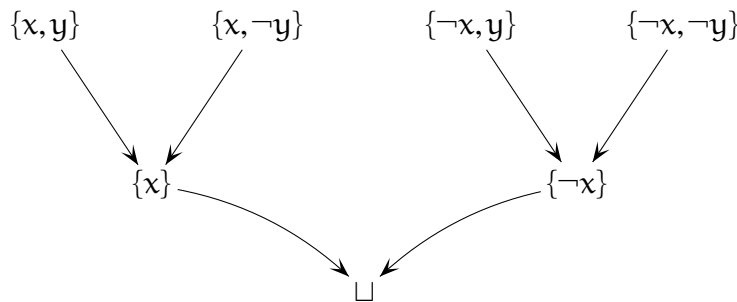
Durch Induktion über die Länge m einer Herleitung kann gezeigt werden, dass alle herleitbaren Klauseln in $Res^*(\alpha)$ liegen. Umgekehrt hat jede Klausel in $Res^*(\alpha)$ eine Herleitung aus α . Diese Aussage kann nachgewiesen werden, indem man durch Induktion nach i zeigt, dass es zu jeder Klausel in $Res^i(\alpha)$ eine Herleitung aus α gibt. Wir erhalten damit folgende Aussage:

Lemma 4.50 (Resolutionsabschluss und Herleitbarkeit). $Res^*(\alpha)$ ist die Menge aller Klauseln, für die es eine Herleitung aus α gibt.

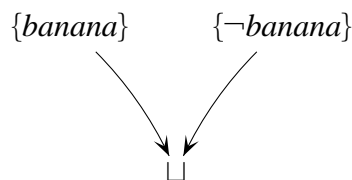
Aus Corollar 4.45 auf Seite 167 und Lemma 4.50 ergibt sich, dass jede Klauselmenge α , aus welcher die leere Klausel herleitbar ist, widersprüchlich (d.h. unerfüllbar) ist. Dies erklärt folgende Bezeichnung:

Definition 4.51 ((Resolutions-)Widerlegung). Eine *Resolutionswiderlegung* (kurz *Widerlegung*) für α bezeichnet eine Resolutionsherleitung der leeren Klausel \sqcup aus α . ■

Folgende Abbildung zeigt eine Widerlegung für die unerfüllbare KNF-Formel $(x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$.



Beispiel 4.52 (Resolutionswiderlegung (Affen-Bananen-Problem)). Wir betrachten nun die Klauselmenge $\alpha \cup \{\neg banana\}$, wobei α wie in Beispiel 4.48 gegeben ist. Die negative Einheitsklausel $\{\neg banana\}$ steht für die Negation der Frage nach der Erreichbarkeit der Banane. Die in Beispiel 4.48 angegebene Herleitung kann nun um den Resolutionsschritt



erweitert werden. Man erhält somit eine Resolutionswiderlegung für $\alpha \cup \{\neg banana\}$. Hieraus folgt zunächst die Unerfüllbarkeit von $\alpha \cup \{\neg banana\}$ und somit $\alpha \models banana$. ■

Da aus der Existenz einer Widerlegung (also einer Herleitung von \sqcup) auf die Unerfüllbarkeit der betreffenden Klauselmenge geschlossen werden kann, ist Resolution als Kalkül für den Nachweis der Unerfüllbarkeit von KNF-Formeln *korrekt*. Der nun folgende Hauptsatz dieses Abschnitts sichert die *Vollständigkeit* der Resolution für den Nachweis der Unerfüllbarkeit zu, indem er belegt, dass auch die Umkehrung gilt; also, dass die leere Klausel aus α herleitbar ist, sofern α unerfüllbar ist. Man spricht daher auch von *Widerlegungsvollständigkeit*.

Satz 4.53 (Resolutionssatz der Aussagenlogik). Sei α eine KNF-Formel. Dann ist α genau dann unerfüllbar, wenn $\sqcup \in Res^*(\alpha)$.

Beweis. Die Korrektheit der Resolution (die Implikation " \Leftarrow ") wird durch Corollar 4.45 (Seite 167) belegt. Wir zeigen nun die Widerlegungsvollständigkeit des Resolutionskalküls. Wir

nehmen die Unerfüllbarkeit von α an und zeigen, dass die leere Klausel \sqcup über 0 oder mehrere Resolutionsschritte aus α resolviert werden kann. Diese Aussage beweisen wir durch Induktion nach der Anzahl n an Aussagensymbolen, die in α vorkommen.

Der Induktionsanfang ist klar, da für $n = 0$ eine KNF-Formel vorliegt, die keine Aussagensymbole enthält und daher entweder zu *false* oder zu *true* äquivalent ist. Da die Unerfüllbarkeit von α vorausgesetzt wird, ist nur der erste Fall möglich. Daher gilt $\sqcup \in \alpha \in Res^*(\alpha)$. Nun zum Induktionsschritt $n \implies n+1$. Wir nehmen an, dass eine unerfüllbaren KNF-Formel α mit $n+1$ Aussagensymbolen vorliegt. Sei x eines der in α vorkommenden Aussagensymbole. Die wie in Bezeichnung 4.24 auf Seite 148 definierten KNF-Formeln

$$\alpha_0 = \alpha[x/0], \quad \alpha_1 = \alpha[x/1]$$

sind aus höchstens n Aussagensymbolen aufgebaut. Mit α sind auch α_0 und α_1 unerfüllbar (Splitting-Regel, siehe Lemma 4.27, Seite 150). Daher liefert die Induktionsvoraussetzung angewandt auf α_0 und α_1 , dass $\sqcup \in Res^*(\alpha_0)$ und $\sqcup \in Res^*(\alpha_1)$. Wir zeigen nun, dass

$$\{\sqcup, \{x\}\} \cap Res^*(\alpha) \neq \emptyset \quad (*)$$

Die Idee für den Nachweis dieser Aussage beruht darauf, die Resolutionsschritte, die zur Herleitung von \sqcup aus α_0 führten, zu einer Herleitung der leeren Klausel \sqcup oder der positiven Einheitsklausel $\{x\}$ aus α zu liften. Hierzu verfahren wir wie folgt. Falls $\sqcup \in \alpha_0$, so gilt entweder $\sqcup \in \alpha$ oder $\{x\} \in \alpha$. In diesem Fall ist Aussage (*) offenbar erfüllt. Im Folgenden nehmen wir $\sqcup \notin \alpha_0$ an und weisen Aussage (*) nach. Wegen $\sqcup \in Res^*(\alpha_0)$ gibt es eine Resolutionswiderlegung

$$\langle \kappa_1, \tau_1, \lambda_1 \rangle \langle \kappa_2, \tau_2, \lambda_2 \rangle \dots \langle \kappa_m, \tau_m, \lambda_m \rangle$$

der Länge $m \geq 1$. Jede der Klauseln κ von α_0 ist entweder eine Klausel von α oder ist durch Streichen des positiven Literals x aus einer Klausel von α entstanden. Dies erlaubt es, eine Resolutionsherleitung

$$\langle \kappa'_1, \tau'_1, \lambda'_1 \rangle \langle \kappa'_2, \tau'_2, \lambda'_2 \rangle \dots \langle \kappa'_m, \tau'_m, \lambda'_m \rangle$$

aus α wie folgt zu definieren. Für $i = 1, \dots, m$ sind die Klauseln κ'_i wie folgt definiert.

- Ist κ_i eine Klausel von α_0 und α , so setzen wir $\kappa'_i = \kappa_i$.
- Ist $\kappa_i \in \alpha_0 \setminus \alpha$, so ist κ_i aus einer Klausel $\kappa \in \alpha$, welche das positive Literal x enthält, durch Streichen von x entstanden. Wir kehren nun – durch Wiedereinfügen von x – zur ursprünglichen Klausel von α zurück. Hierzu setzen wir $\kappa'_i = \kappa = \kappa_i \cup \{x\}$.
- Ist $\kappa_i \notin \alpha_0$, so ist $\kappa_i = \lambda_j$ für ein $j \in \{1, \dots, i-1\}$. In diesem Fall setzen wir $\kappa'_i = \lambda'_j$.

In analoger Weise definieren wir die Klausel τ'_i . Die Klausel λ'_i ist als Resolvent von κ'_i und τ'_i definiert. Ist etwa

$$\lambda_i = \kappa_i \setminus \{y\} \cup \tau_i \setminus \{\neg y\}$$

mit $y \in \kappa_i$ und $\neg y \in \tau_i$, so setzen wir

$$\lambda'_i = \kappa'_i \setminus \{y\} \cup \tau'_i \setminus \{\neg y\}.$$

Es ist nun leicht zu sehen, dass $\langle \kappa'_1, \tau'_1, \lambda'_1 \rangle \dots, \langle \kappa'_m, \tau'_m, \lambda'_m \rangle$ eine Herleitung aus α ist und dass

$$\lambda'_i = \lambda_i \text{ oder } \lambda'_i = \lambda_i \cup \{x\}, \quad i = 1, \dots, m.$$

Mit $i = m$ folgt, dass λ'_m entweder die leere Klausel ist oder die Einheitsklausel $\{x\}$. Eine analoge Argumentation zeigt, dass

$$\{\sqcup, \{\neg x\}\} \cap \text{Res}^*(\alpha) \neq \emptyset. \quad (**)$$

Da die leere Klausel \sqcup ein Resolvent der Einheitsklauseln $\{x\}$ und $\{\neg x\}$ ist, folgt aus (*) und (**) die Herleitbarkeit der leeren Klausel. Also $\sqcup \in \text{Res}^*(\alpha)$. \square

Beispiel 4.54 (Lifting der Herleitung). Wir illustrieren die im Beweis des Resolutionsatzes beschriebene Technik zum Liften von Herleitungen aus $\alpha[x/b]$ zu Herleitungen aus α an zwei einfachen Beispielen. Sei

$$\alpha = \underbrace{(x \vee \neg y \vee z)}_{=\kappa'_1} \wedge \underbrace{(y \vee z)}_{=\tau'_1=\tau_1} \wedge (\neg x \vee z) \wedge \underbrace{\neg z}_{=\kappa'_2=\kappa_2}$$

Dann ist $\alpha_0 = \alpha[x/0] = \underbrace{(\neg y \vee z)}_{=\kappa_1} \wedge \underbrace{(y \vee z)}_{=\tau_1} \wedge \underbrace{\neg z}_{=\kappa_2}$. Z.B. ist

$$\begin{aligned} \langle \kappa_1, \tau_1, \lambda_1 \rangle &= \langle \{\neg y, z\}, \{y, z\}, \{z\} \rangle \\ \langle \kappa_2, \tau_2, \lambda_2 \rangle &= \langle \{\neg z\}, \underbrace{\{z\}}_{=\lambda_1}, \sqcup \rangle \end{aligned}$$

eine Widerlegung für α_0 . Diese wird zu einer Herleitung der Einheitsklausel $\{x\}$ aus α geliftet:

$$\begin{aligned} \langle \kappa'_1, \tau'_1, \lambda'_1 \rangle &= \langle \underbrace{\{x, \neg y, z\}}_{=\kappa_1 \cup \{x\}}, \underbrace{\{y, z\}}_{=\tau_1 \in \alpha}, \underbrace{\{x, z\}}_{=\lambda_1 \cup \{x\}} \rangle \\ \langle \kappa'_2, \tau'_2, \lambda'_2 \rangle &= \langle \underbrace{\{\neg z\}}_{=\kappa_2 \in \alpha}, \underbrace{\{x, z\}}_{=\lambda'_1}, \underbrace{\{x\}}_{=\lambda_2 \cup \{x\}} \rangle. \end{aligned}$$

Ebenso kann für $\alpha_1 = \alpha[x/1] = (y \vee z) \wedge z \wedge \neg z$ verfahren werden. Für α_1 haben wir eine Widerlegung der Länge 1 bestehend aus dem Tripel $\langle \{z\}, \{\neg z\}, \sqcup \rangle$. Wiedereinfügen des gestrichenen Literals $\neg x$ in die erste Elternklausel liefert die Herleitung

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

der Einheitsklausel $\{\neg x\}$ aus α . Wir setzen nun die beiden Herleitungen für $\{x\}$ und $\{\neg x\}$ aus α zu einer Widerlegung für α zusammen:

$$\langle \{x, \neg y, z\}, \{y, z\}, \{x, z\} \rangle \langle \{\neg z\}, \{x, z\}, \{x\} \rangle \langle \{\neg x, z\}, \{\neg z\}, \{\neg x\} \rangle \langle \{x\}, \{\neg x\}, \sqcup \rangle$$

Man beachte, dass es möglich ist, durch das Liften von Widerlegungen für $\alpha[x/b]$ sofort eine Widerlegung für α zu erhalten. Z.B. trifft dies auf die Formel

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

zu. Für diese besteht

$$\alpha_0 = \alpha[x/0] = (y \vee z) \wedge (\neg y \vee z) \wedge \neg z$$

nur aus Klauseln, die bereits in α enthalten sind. Liften der Widerlegung

$$\langle \{y, z, \{\neg z\}, \{y\} \rangle \langle \{\neg y, z, \{\neg z\}, \{\neg y\} \rangle \langle \{y\}, \{\neg y\}, \perp \rangle$$

für α_0 zu einer Herleitung für α bringt keinerlei Veränderungen, da die Elternklauseln der ersten beiden Resolutionsschritte in α liegen. ■