Einiges zu Resolutionen anhand der Aufgaben 6 und 7

Es gibt eine Fülle von verschiedenen Resolutionen. Die bis jetzt behandelten möchte ich hier noch ein Mal kurz erläutern. Ferner möchte ich noch auf vollständige Kalküle und widerspruchsvollständige Kalküle eingehen.

Mit Hilfe von Aufgabe 7 will ich noch ein Mal die *graphische Resolution* (Definition 2.26), das *Resolutionskalkül* (Korollar 2.33: Erfüllbarkeitstest für aussagenlogische Formel), die *lineare Resolution* (Definition 3.10) und die *Input–Resolution* (Definition 3.12). Für diese vier Resolutionsmethoden ist nur eine aussagenlogische Formel in KNF vorausgesetzt.
Für den *Markierungsalgorithmus* (Satz 3.7: Der Markierungsalgorithmus oder Horn–Algorithmus) und die SLD–*Resolution* (Bemerkung 3.16) über die `PROLOG`–*Notation* (Definition 3.19: Die `PROLOG`–Notation) ist eine Hornformel von nöten, deshalb verwenden wir die erweiterte Formel aus Aufgabe 6.

Widerspruchsvollständig oder vollständig, das ist hier die Frage!

In Satz 3.17 ist die lineare Resolution als vollständig beschrieben, d. h. ein Kalkül, mit welchem sich auf jedem Fall die leere Menge $\emptyset$ aus einer Klauselmenge einer unerfüllbaren Formel resolvieren lässt. Im umgangssprachlichen Sinne ist wahrscheinlich „widerspruchsvollständig“ ein passenderes Wort.
Vier der von uns betrachteten Methoden zur Resolution sind rein (widerspruchs-)vollständig: graphische Resolution, lineare Resolution, Input–Resolution, `PROLOG`/ SLD–Resolution. Wohingegen das Resolutionskalkül und der Markierungsalgorithmus „doppelt“ vollständig sind, d. h. mit ihnen lässt sich nicht nur die Unerfüllbarkeit einer Formel beweisen, sondern gegebenenfalls auch die Erfüllbarkeit.
Da wir jedoch im Allgemeinen beweisen wollen, dass eine Formel G aus einer Formel F *immer* folgt, also dass $F \Rightarrow G$ eine Tautologie ist, zeigen wir nach Satz 2.10 die Unerfüllbarkeit von

$$\neg (F \Rightarrow G) \equiv \neg (\neg F \vee G) \equiv (F \wedge \neg G) .$$

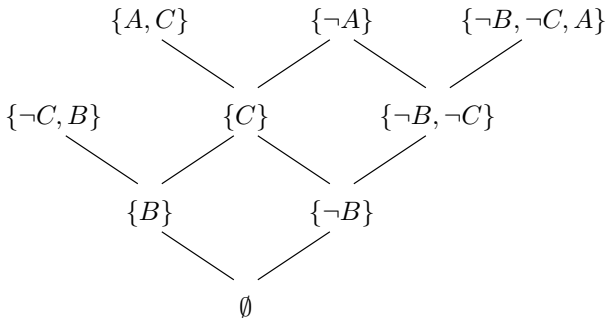
Somit reicht uns die Widerspruchsvollständigkeit völlig aus. Um genau zu sein, bringt es uns auch nichts, die Erfüllbarkeit von $F \Rightarrow G$ mittels Resolutionskalkül oder Markierungsalgorithmus zu zeigen, denn damit weisen wir nur nach, dass es *eine* Belegung α gibt mit $\alpha (F \Rightarrow G) = 1$. Über die Allgemeingültigkeit können wir nichts aussagen.

Graphische Resolution
(Definition 2.26)

Dies ist die Resolution, die der/dem Ausführenden am meisten Freiheit lässt. Man könnte sagen, dies ist eine Resolutionsmethode, die mit „scharfem Hingucken“ funktioniert. Man versucht immer die leere Menge $\emptyset$ zu resolvieren, denn diese Methode ist nur (widerspruchs-)vollständig. Bei dieser Methode fängt man erst Mal an und schaut dann, wie man weitermachen könnte. Dabei braucht man sich an keine Auswahlregeln halten.

Wir starten mit der Klauselmenge aus Aufgabe 7:

$$\mathcal{K}(F) = \{\{A, C\}, \{\neg C, B\}, \{\neg A\}, \{\neg B, \neg C, A\}\}.$$



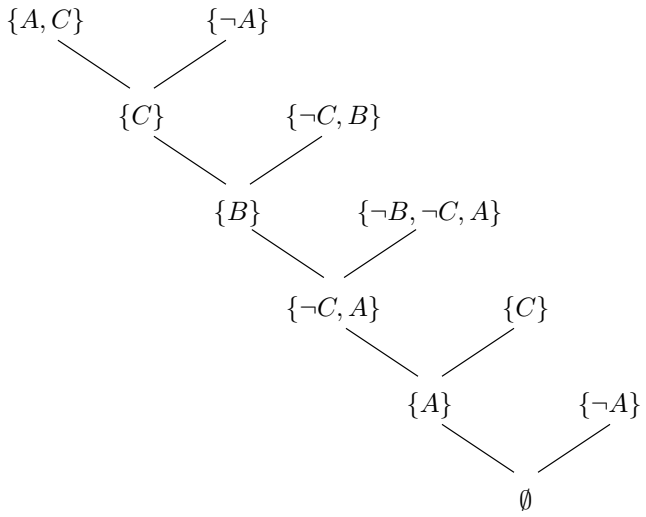
Resolutionskalkül
(Korollar 2.33)

Dieses Kalkül ermöglicht es uns sowohl die Erfüllbarkeit wie auch die Unerfüllbarkeit einer aussagenlogischen Formel F nachweisen, d. h. es ist „doppelt“ vollständig. Dazu bildet man alle möglichen Resolventen aus Elementen der Klauselmenge $\mathcal{K}(F)$ und nimmt diese dann zur Klauselmenge hinzu, man erhält $\mathrm{Res}^1(\mathcal{K}(F))$. Jetzt bildet man wieder alle möglichen Resolventen dieser Menge und nimmt sie hinzu, man erhält $\mathrm{Res}^2(\mathcal{K}(F))$, und so weiter. Dabei lässt man jedoch die den Tautologien entsprechenden Klauseln weg (vergleiche dazu das Blatt zu Bemerkung 2.29). Natürlich werden auch die mehrfach auftretenden Klauseln weggelassen. Der Algorithmus gibt „die Formel ist unerfüllbar“ aus, wenn $\emptyset \in \mathrm{Res}^n(\mathcal{K}(F))$. Wenn sich beim Abarbeiten des Algorithmus ergibt, dass $\mathrm{Res}^{n+1}(\mathcal{K}(F)) = \mathrm{Res}^n(\mathcal{K}(F))$, dann gibt er „die Formel ist erfüllbar“ aus.

$$\begin{aligned}
\mathrm{Res}^0(\mathcal{K}(F)) &= \mathcal{K}(F) = \{\{A, C\}, \{\neg C, B\}, \{\neg A\}, \{\neg B, \neg C, A\}\} \\
\mathrm{Res}^1(\mathcal{K}(F)) &= \mathcal{K}(F) \cup \{\{A, B\}, \{C\}, \{A, \neg B\}, \{A, \neg C\}, \{\neg B, \neg C\}\} \\
\mathrm{Res}^2(\mathcal{K}(F)) &= \mathrm{Res}^1(\mathcal{K}(F)) \cup \{\{A\}, \{A, \neg B\}, \{B\}, \{A, \neg C\}, \{\neg C\}, \{B\}, \{\neg B\}, \\
&\qquad \{\neg C\}, \{A, \neg C\}, \{A, \neg B\}, \{A\}, \{A, \neg C\}, \{A\}, \{\neg B\}\} \\
&= \mathrm{Res}^1(\mathcal{K}(F)) \cup \{\{A\}, \{B\}, \{\neg C\}, \{\neg B\}\} \\
\mathrm{Res}^3(\mathcal{K}(F)) &= \mathrm{Res}^2(\mathcal{K}(F)) \cup \{\emptyset, \ldots\}.
\end{aligned}$$

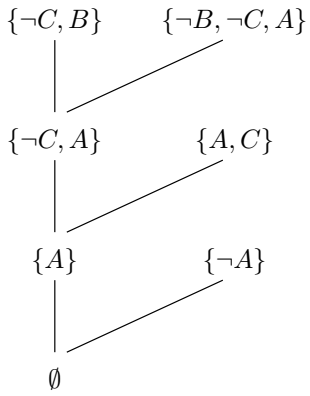
lineare Resolution
(Definition 3.10)

Diese Resolution ist im Prinzip eine Einschränkung der graphischen Resolution. Hier wird immer ein Schritt nach dem anderen durchgeführt, wobei man einen Hauptstrang beibehält. Zum Resolvieren können als Seitenklausel alle Elemente der Klauselmenge und bereits resolvierte neue Klauseln verwendet werden. In unserem Beispiel ist hier das $\{C\}$ in der vierten Resolution ein solches bereits resolviertes Ergebnis.



Input–Resolution
(Definition 3.12)

Die Input–Resolution ist eine Einschränkung der linearen Resolution, jetzt dürfen nur noch Elemente der Klauselmenge als Seitenklausel verwendet werden.



Markierungsalgorithmus
(Satz 3.7)

Zuerst muss man die Voraussetzung überprüfen, ob die zu untersuchende Formel/Klauselmenge auch wirklich eine Hornformel/-klausel ist, d. h. ob in jeder Disjunktion in der KNF nur maximal ein positives Literal auftaucht oder ob in jeder Klausel nur maximal ein positives Literal vorkommt. Dann geht's los: Man markiert alle einzelnen positiven Literale in den Tatsachenklauseln (rein positive einzelne Klauseln) und jedes weitere Vorkommen dieser Literale. Entsteht dabei eine Prozedurklausel (gemischte Klausel mit genau einem positiven Literal), in der alle negativen Literale markiert sind, so markiert man das verbleibende positive Literal und sein weiteres Vorkommen, und so weiter. Es werden auch alle weiteren positiven Vorkommen dieser Literale markiert, mit diesen Klauseln (in denen das Literal positiv vorkommt) können wir dann zwar nichts weiter anfangen. Sie werden nur wegen der Vollständigkeit markiert, um nicht mehrfach die gleichen Schritte zu machen. Wichtig ist, dass man immer nur mit den positiven Literalen weitermacht. Ist z. B. in einer Klausel ein positives Literal markiert $\{\underline{A}, \neg B\}$, so dürfen wir **nicht** mit dem verbleibenden negativen Literal $\neg B$ weiterarbeiten!
Zum Schluss schaut man sich die Zielklausel (rein negativ) an. Ist sie vollständig markiert, so ist die Formel unerfüllbar, ist sie nicht vollständig markiert, so ist die Formel erfüllbar. Wenn es mehr als eine Zielklausel gibt, so reicht eine vollständig markierte Zielklausel für die Unerfüllbarkeit aus. Natürlich müssen dann für die Erfüllbarkeit auch alle Zielklauseln nicht vollständig markiert sein.
Auch dieser Algorithmus ist „doppelt" vollständig, wie schon das Resolutionskalkül. Man sollte immer die Schritte, in denen man markiert, mit angeben, um dem Leser die Möglichkeit zu geben die Abarbeitung des Algorithmus nachzuvollziehen.

Klauselmenge aus Aufgabe 6

$$\mathcal{K}(H) = \big\{ \{\neg A, \neg B\}, \{\neg A, C\}, \{\neg B, C\}, \{A, \neg C\}, \{B\} \big\}.$$

1. Schritt: Markiere B

$$\mathcal{K}(H) = \big\{ \{\neg A, \underline{\neg B}\}, \{\neg A, C\}, \{\underline{\neg B}, C\}, \{A, \neg C\}, \{\underline{B}\} \big\}$$

2. Schritt: Markiere C

$$\mathcal{K}(H) = \big\{ \{\neg A, \underline{\neg B}\}, \{\neg A, \underline{C}\}, \{\underline{\neg B}, \underline{C}\}, \{A, \underline{\neg C}\}, \{\underline{B}\} \big\}$$

3. Schritt: Markiere A

$$\mathcal{K}(H) = \big\{ \{\underline{\neg A}, \underline{\neg B}\}, \{\underline{\neg A}, \underline{C}\}, \{\underline{\neg B}, \underline{C}\}, \{\underline{A}, \underline{\neg C}\}, \{\underline{B}\} \big\}$$

Da die Zielklausel $\{\neg A, \neg B\}$ vollständig markiert ist, ist die Formel H unerfüllbar.

PROLOG/ SLD–*Resolution*
(Definition 3.19/ Bemerkung 3.16)

Um aus einer Hornformel ein PROLOG–Programm zu machen, betrachten wir die drei Arten von Klauseln in der Klauselmenge genauer:

- Tatsachenklauseln sind Klauseln, in denen nur ein positives Literal steht.
- Prozedurklauseln/Regeln enthalten ein positives Literal und eine beliebige Anzahl von negativen Literalen.
- Zielklauseln bestehen nur aus negativen Literalen.

Aus den Tatsachenklauseln werden solitäre Programmschritte. Z. B. wird aus der Klausel $\{A\}$ der Programmschritt `A`. Aus den Prozedurklauseln werden ersetzende Programmschritte, so wird z. B. aus der Klausel $\{\neg A, B\}$, die der Formel $A \Rightarrow B$ entspricht, der Programmschritt `B:-A`. Und aus der/einer der Zielklausel(n) wird der Start des Programms, so wird z. B. aus $\{\neg B\}$ der Aufruf `?:-B`. Man liest die Hornformel im Prinzip rückwärts, im Gegensatz zum Markierungsalgorithmus.

In unserem Fall ergibt sich aus der Klauselmenge

$$\mathcal{K}(H) \quad = \quad \big\{ \{\neg A, \neg B\}, \{\neg A, C\}, \{\neg B, C\}, \{A, \neg C\}, \{B\} \big\}.$$

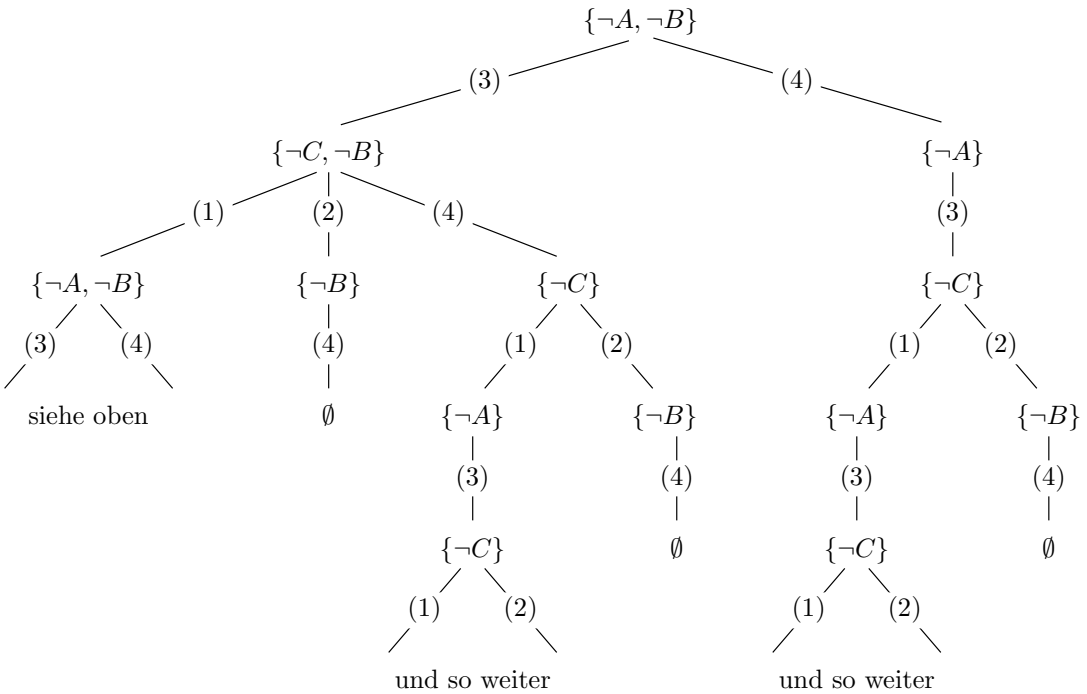
das folgende PROLOG–Programm:

(1) `C:-A.` (3) `A:-C.` `?:-A,-B.`

(2) `C:-B.` (4) `B.`

Wir bilden jetzt den SLD–Baum. Die SLD–Resolution ist eine Input–Resolution bei der mit der Zielklausel gestartet wird und dann immer mit den Programmklauseln (Prozedurklauseln und Tatsachenklauseln) weiter resolviert wird. In einem SLD–Baum werden immer alle Möglichkeiten zu resolvieren durch verschiedene Äste dargestellt. Auf oder neben den jeweiligen Zweig ist dann die entsprechende Programmnummer zu notieren. Wenn man jedoch eine Ahnung hat, wie die Resolution verlaufen könnte, so macht man nur eine kurze Input–Resolution.

- SLD–Baum:



Benutzt man keine weitere Auswahlstrategie, sondern immer die erstmögliche Resolution, so ergibt sich folgendes Bild.

Beim Abarbeiten mit der *breadth–first*–Strategie, also der Breitensuche, arbeitet man den Baum in folgender Reihenfolge, bis zum Erreichen der leeren Menge, ab:

$$(3),\ (4),\ (1),\ (2),\ (4),\ (3),\ (3),\ (4),\ (4).$$

Benutzt man die *depth–first*–Strategie, also die Tiefensuche, so kommt man sofort in eine Endloschleife:

$$(3),\ (1),\ (3),\ (1),\ (3),\ (1),\ \ldots\ .$$

- Input–Resolution:

Endlosschleife

$\{\neg A, \neg B\}$
|
(3)
|
$\{\neg C, \neg B\}$
|
(1)
|
$\{\neg A, \neg B\}$
⋮
(3)/(1)
⋮

Unerfüllbarkeit

$\{\neg A, \neg B\}$
|
(3)
|
$\{\neg C, \neg B\}$
|
(2)
|
$\{\neg B\}$
|
(4)
|
$\emptyset$