Quantifizierte Boole'sche Formeln

Anna-Katharina Brünn

24. August 2020

Erklärung der Selbstständigkeit

Hiermit versichere ich, die vorliegende Bachelorarbeit/Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Wedemark, den 24. August 2020

_____

Anna-Katharina Brünn

Inhaltsverzeichnis

1 Einleitung

Diese Abschlussarbeit behandelt quantifizierte Boole'sche Formeln, also eine spezielle Form der Erweiterung der Aussagenlogik um Quantoren und quantifizierte Variablen, nach ihrer Darstellung in Kleine Bünings und Lettmanns *Aussagenlogik: Deduktion und Algorithmen*. Diese Publikation stellt damit die Primärquelle dieser Arbeit dar, die als Zusammenfassung des siebten Kapitels des genannten Buchs verstanden werden kann, dessen Inhalte aber stellenweise um weitere Quellen erweitert.

Quantifizierte Boole'sche Formeln stellen, wie bereits erwähnt, eine Erweiterung der Aussagenlogik dar. Im Verlauf dieser Arbeit wollen wir deshalb betrachten, inwiefern quantifizierte Formeln die Aussagenlogik erweitern können, welchen potentiellen Nutzen wir dadurch erlangen, ob sie neue Sichtweisen auf aus der Aussagenlogik bekannte Probleme eröffnen und ob es sinnvoll sein kann, sie anstelle von regulären aussagenlogischen Formeln zu nutzen.

Um diese Fragen zu beantworten, beschäftigen wir uns im ersten Kapitel dieser Arbeit daher mit den Grundlagen quantifizierter Formeln, beantworten die Frage, was genau quantifizierte Boole'sche Formeln sind, inwiefern sie sich von aussagenlogischen Formeln unterscheiden und was im Umgang mit ihnen zu beachten ist. Wir führen notwendige Grundbegriffe ein und modifizieren aus der Aussagenlogik bekannte Konzepte und Definitionen so, dass sie auf quantifizierte Boole'sche Formeln anwendbar sind.

Außerdem betrachten wir, wie sich die Definitionen einiger gängiger Normalformen auf quantifizierte Formeln übertragen lassen und welche Bedeutung sie für die Arbeit mit quantifizierten Formeln haben.

Danach gehen wir auf die Resolution quantifizierter Formeln ein und zeigen, dass es auch für quantifizierte Boole'sche Formeln einen widerlegungsvollständigen Resolutionskalkül gibt, um die Unerfüllbarkeit einer Formel zu zeigen. Zudem beschäftigen wir uns kurz mit einigen möglichen Restriktionen, um effizientere Resolutionswiderlegungen zu ermöglichen.

Schließlich wenden wir uns noch einigen aus der Aussagenlogik bekannten Problemen zu, die auch für quantifizerte Formeln existieren. Wir widmen uns der Frage nach der Erfüllbarkeit einer Formel, der Folgerbarkeit einer Formel von einer anderen und der Frage, wie man zwei Formeln auf ihre Äquivalenz hin überprüfen kann.

Abschließend bietet diese Arbeit noch einige weiterreichende Literaturvorschläge, die sich tiefergehend mit quantifizierten Boole'schen Formeln und bestimmten Teilbereichen aus diesem Gebiet beschäftigen.

2 Grundlagen

In diesem Kapitel werden wir uns damit beschäftigen, was quantifizierte Boole'sche Formeln sind, wie sich verhalten und was sie auszeichnet.

Bevor wir allerdings damit beginnen können, müssen für dieses und die folgenden Kapitel einige Begrifflichkeiten und Notationen festgelegt werden.

Anders als in der Aussagenlogik sind quantifizierte Formeln nicht aus herkömmlichen Atomen aufgebaut. Da die Quantoren den Atomen einen Variablencharakter verleihen, werden wir sie im Folgenden auch als solche bezeichnen.

Zudem werden wir, wie aus der Prädikatenlogik bekannt, die Begriffe freie und gebundene Variable nutzen, respektive Variablen, über die sich kein Quantor erstreckt, und solche, für die dies der Fall ist.

Entsprechend ist eine offene Formel definiert als quantifizierte Formel, die freie Variablen enthält, während in einer geschlossenen Formel alle Variablen gebunden vorkommen. Im Sinne der einfacheren Lesbarkeit werden gebundene Variablen mit den Buchstaben x und y bezeichnet, für freie Variablen werden die Buchstaben $a, b, \ldots$ genutzt. Hierbei gilt insbesondere, dass x_i für Variablen genutzt werden wird, auf die sich der Allquantor $\forall$ beziehen, y_i ist entsprechend an den Existenzquantor $\exists$ gebunden. Um quantifizierte Formeln auf den ersten Blick von aussagenlogischen Formeln zu unterscheiden, werden für Erstere als Formelbezeichner griechische Großbuchstaben verwendet, für Letztere, wie gewohnt, Kleinbuchstaben.

2.1 Quantifizierte Formeln

Quantifizierte Boole'sche Formeln sind, wie der Name bereits vermuten lässt, aussagenlogische Formeln, die durch Quantoren erweitert werden. Entsprechend lässt sich jede aussagenlogische Formel auch als quantifizierte Formel ohne gebundene Variablen verstehen, andersherum kann man allerdings nicht jede quantifizierte Formel wie eine rein aussagenlogische Formel behandeln. Dennoch werden wir sehen, dass vieles, was auf aussagenlogische Formeln zutrifft, in leicht abgewandelter Form auch für quantifizierte Formeln gegeben ist.

Bevor wir uns quantifizierte Formeln nun genauer ansehen, der Vollständigkeit halber zuerst die formale Definition:

Definition 2.1.1 (Quantifizierte Boole'sche Formeln)**.** [1] Die Menge der quantifizierten Boole'schen Formeln ist induktiv wie folgt definiert:

1. Jede aussagenlogische Formel ist eine quantifzierte Boole'sche Formel.
2. Sei Φ eine quantifizierte Boole'sche Formel und seien x bzw y aussagenlogische Variablen, dann sind auch $\exists y\ \Phi$ und $\forall x\ \Phi$ quantifizierte Boole'sche Formeln.
3. Nur gemäß 1. und 2. gebildete Formeln sind quantifizierte Boole'sche Formeln.

Punkt 1 der Definition wurde bereits erläutert, die weiteren Punkte sind selbsterklärend, weshalb ein weiteres Eingehen auf die Definition nicht notwendig ist. Im Folgenden benutzen wir die Abkürzung QBF* für die Formelklasse aller quantifizierten Boole'schen Formeln, die den obigen Definitionen genügen, und QBF $\subseteq$ QBF* bezeichnet die Teilmenge der geschlossenen Formeln.

Einführend betrachten wir nun einige Grundlagen im Zusammenhang mit quantifizierten Formeln. Im späteren Verlauf des Kapitels werden wir die folgenden Eigenschaften noch formal definieren, vorerst sollen uns allerdings diese beispielhaften, intuitiven Herleitungen für ein erstes Grundverständnis genügen.

Wie eingangs erwähnt, handelt es sich bei jeder aussagenlogischen Formel eigentlich auch um eine quantifizierte Formel ohne gebundene Variablen. Wir können also erste Eigenschaften quantifizierter Formeln darüber herleiten, dass wir bestimmte aussagenlogische Formeln mit Quantoren versehen. Sei eine Formel α mit den Atomen $x_1, x_2, x_3, \ldots x_n$ gegeben. Der nähere Aufbau der Formel, bestimmte Klammerungen, genutzte Konjunktoren, all das ist nicht von Bedeutung. Egal wie α weiter gestaltet ist, es gilt immer, dass α erfüllbar ist, wenn es eine Belegung x_i mit den Wahrheitswerten 1 oder 0 gibt, sodass die Aussage von α wahr wird. Es existieren also $x_1, \ldots x_n$ so, dass α gilt. Oder anders formuliert:
$\exists x_1, \ldots \exists x_n \alpha$ ist wahr.
Umgekehrt gilt selbstverständlich auch, dass, wenn es eine Belegung $x_1, \ldots x_n$ gibt, für die α immer falsch ist, sich dies formulieren lässt als
$\exists x_1, \ldots \exists x_n \alpha$ ist widerspruchsvoll.

[1] [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 362

Wir sehen also, dass wir die grundlegenden Eigenschaften der Erfüllbarkeit und Widersprüchlichkeit in der Aussagenlogik leicht auch als quantifizierte Formel darstellen können.

Gehen wir einen Schritt weiter, so sehen wir, dass wir auch die Äquivalenz zweier aussagenlogischer Formeln durch Formeln mit Quantoren ausdrücken können. Seien α und β zwei aussagenlogische Formeln und ihre Atome entstammen der Variablenmenge $\{x_1 \dots x_n\}$, dann sind α und β genau dann äquivalent, wenn gilt:

$$\alpha \approx \beta \Leftrightarrow \forall x_1 \dots x_n (\alpha \leftrightarrow \beta)$$

ist wahr.
Wichtig ist hier, dass, anders als bei den vorherigen Beispielen, hier der Allquantor zum Einsatz kommt. Nur dann, wenn für alle Belegungen von $x_1, \dots x_n$ das Gleiche gilt, sind α und β wirklich logisch äquivalent. Die weniger strikte Erfüllbarkeitsäquivalenz zwischen den beiden Formeln lässt sich hingegen wie folgt mit dem Existenzquantor darstellen:

$$\alpha \underset{sat}{\approx} \beta \Leftrightarrow \exists x_1 \dots x_n \alpha \leftrightarrow \exists x_1 \dots x_n \beta)$$

ist wahr.
α und β sind also erfüllbarkeitsäquivalent, solange für beide je eine Belegung von $x_1, \dots x_n$ existiert, sodass die Aussagen wahr beziehungsweise für beide falsch sind.

Äquivalenzen gibt es nicht nur zwischen zwei aussagenlogischen Formeln oder zwei quantifizierten Formeln.

Da aussagenlogische Formeln, wie bereits geschrieben, als quantifizierte Formeln ohne Quantoren, also eine Formel nur aus freien Variablen interpretiert werden können, gibt es auch Äquivalenzen zwischen aussagenlogischen Formeln und quantifizierten Formeln.

Insbesondere gilt dies für offene Formeln. Jede Teilformel aus freien Variablen lässt sich als logisch äquivalente Formel der Aussagenlogik darstellen und analog gilt auch, dass jede aussagenlogische Formel sich logisch äquivalent als Teilformel einer offenen quantifizierten Formel darstellen lässt.

Nehmen wir zum Beispiel die Formel

$$\alpha = (a \vee b \vee c) \wedge d$$

Führen wir eine neue, gebundene Variable y ein

$$\Phi = \exists y (a \vee b \vee y) \wedge (\neg y \vee c) \wedge d$$

sieht man, dass beide Formeln äquivalent sind, da immer eine Belegung von y existiert, sodass β wahr ist, wenn α mit der gleichen Belegung wahr ist. Diese Eigenschaft kann man sich insbesondere beim Umformen von Formeln zu Nutze machen.

Das obige Beispiel hat, wie man sieht, eine besondere Form. Diese ist nicht zufällig gewählt. Quantifizierte Boole'sche Formeln, wie sie in dieser Arbeit behandelt werden,

sind in der sogenannten Pränexnormalform geschrieben. Das heißt, eine Formel besteht im Prinzip aus zwei Teilen, dem Präfix, in dem die Quantoren aufgelistet sind, und dem Kern, welcher stets quantorenfrei ist. Es handelt sich also nicht bei jeder Formel mit Quantoren automatisch um eine quantifizierte Boole'sche Formel, wie sie hier in dieser Arbeit behandelt werden.
Die Formel

$$\Theta = (a \vee \neg b \vee u) \wedge \exists u(\neg u \vee c \wedge v) \vee \forall v(\neg v \wedge d)$$

ist zum Beispiel nach den bisher festgelegten Definitionen keine legitime quantifizierte Formel im Sinne dieser Arbeit.

Sie lässt sich allerdings mit einigen einfachen Schritten so umformen, dass eine äquivalente quantifizierte Formel in Pränexnormalform entsteht. Zuallererst gilt, dass keine Variable sowohl frei als auch gebunden vorkommen darf. In dem obigen Beispiel wären dies zum Beispiel u und v.

Lösen lässt sich dieses Problem, indem man gebundenen Variablen durch andere Variablennamen ersetzt, sodass Quantoren sich stets auf eindeutig benannte, unterschiedliche Variablen beziehen und keine Variable gleichzeitig innerhalb derselben Formel frei und gebunden auftritt.

$$\Theta = (a \vee \neg b \vee u) \wedge \exists y(\neg y \vee c \wedge v) \vee \forall x(\neg x \wedge d)$$

Zieht man bei dieser Formel jetzt noch alle Quantoren an den Anfang und schafft so ein Präfix und einen darauf folgenden Kern, erhält man eine quantifizierte Formel in Pränexnormalform:

$$\Theta = \exists y \forall x(a \vee \neg b \vee u) \wedge (\neg y \vee c \wedge v) \vee (\neg x \wedge d)$$

Nun da wir Präfix und Kern einer Formel definiert haben, können wir uns auch die Länge einer Formel ansehen. Diese ergibt sich, analog zur Aussagenlogik, aus der Anzahl der Variablenvorkommen. In einer Formel der Länge n kommen also exakt n-viele Variablen vor, wobei auch die Variablenvorkommen im Präfix mitgezählt werden.

Unsere Beispielformel Θ hat also die Länge 10.

Insbesondere gilt somit, dass eine quantifizierte Formel der Länge n maximal n Quantoren in ihrem Präfix haben kann.

Zusätzlich zur Form werden wir nun noch die Literalreihenfolge normieren. Dies dient in erster Linie der vereinfachten Lesbarkeit und erlaubt eine bessere maschinelle Verarbeitung der Formel. Zudem ist bei quantifizierten Formeln, deren Kern in konjunktiver Normalform ist, der Bindungsbereich der Quantoren relevant. Normiert man die Reihenfolge der Literale und beachtet man die bereits zu Anfang festgelegte Konvention, dass allquantifizierte Variablen mit x_i, existenzquantifizierte mit y_i benannt werden, so kann man viele Informationen bezüglich des Präfixes auch einfach aus dem Kern ablesen.

Wir sortieren unsere Literale aufsteigend nach ihrem Index, wobei dasjenige Literal den kleinsten Index erhält, dessen Variable im Präfix zuerst auftritt. Formal definiert erhalten wir so diese Ordnungsrelation:

Definition 2.1.2 (Literalreihenfolge in QBF*)**.** [2] Sei Φ eine quantifizierte Boole'sche und L_1 und L_2 Literale in Φ. Dann ist L_1 kleiner als L_2, in Zeichen $L_1 < L_2$, falls die Variable als L_1 im Präfix vor der Variable aus L_2 auftritt, oder die Variable von L_1 frei und die von L_2 gebunden ist.
Wir schreiben Klauseln also als $(L_1 \vee L_2 \vee \cdots \vee L_n)$, wobei kein L_i kleiner sein kann als $L_(i-1)$.

Eine solche Formel wäre zum Beispiel:

$$\Theta = \forall x_1 \exists y_1 \exists y_3 (x_1 \vee y_1 \vee y_2) \wedge y_3$$

Nicht aber:

$$\Theta = \forall x_2 \forall x_1 \exists y_1 \exists y_2 (x_2 \vee y_1 \vee y_3) \wedge y_2 \wedge x_1$$

Eine Ausnahme von dieser Regelung werden allerdings Horn-Formeln bilden. Bei diesen wird, zur besseren Übersicht, manchmal auf diese Konvention verzichtet werden.

Betrachten wir nun, wie verschiedene Belegungen für quantifizierte Formeln definiert sind.

Definition 2.1.3 (Belegungen in QBF*)**.** [3] Seien die Variablen $z_1, \ldots, z_n$ gegeben, dann ist eine Belegung $\mathfrak{J}$ eine Abbildung

$$\mathfrak{J} : z_1, \ldots, z_n \to \{0, 1\},$$

die unter Erfüllung der nachfolgenden Bedingungen zu einer Belegung $\mathfrak{J}$ der Formeln $\Phi \in$ QBF* mit $z_1, \ldots, z_n$ als freien Variablen führt.

$\mathfrak{J}(p) = \mathfrak{J}(p)$	für $p \in$ Var
$\Phi = 0$:	$\mathfrak{J}(0) = 0$
$\Phi = 1$:	$\mathfrak{J}(1) = 1$
$\Phi = \neg\Phi'$:	$\mathfrak{J}(\neg\Phi') = 1 \Leftrightarrow \mathfrak{J}(\neg\Phi') = 0$
$\Phi = \Phi_1 \vee \Phi_2$:	$\mathfrak{J}(\Phi_1 \vee \Phi_2) = 1 \Leftrightarrow \mathfrak{J}(\Phi_1) = 1$ oder $\mathfrak{J}(\Phi_2) = 1$
$\Phi = \Phi_1 \wedge \Phi_2$:	$\mathfrak{J}(\Phi_1 \wedge \Phi_2) = 1 \Leftrightarrow \mathfrak{J}(\Phi_1) = \mathfrak{J}(\Phi_2) = 1$
$\Phi = \exists y \Phi'$:	$\mathfrak{J}(\exists y \Phi') = 1 \Leftrightarrow \mathfrak{J}(\Phi'[y/0]) = 1$ oder $\mathfrak{J}(\Phi'[y/1]) = 1$
$\Phi = \forall x \Phi'$:	$\mathfrak{J}(\forall x \Phi') = 1 \Leftrightarrow \mathfrak{J}(\Phi'[x/0]) = \mathfrak{J}(\Phi'[x/1]) = 1$

[2] [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 364
[3] [KL94]Aussagenlogik: Deduktion und Algorithmen, Seite 365

Bereits am Anfang des Kapitels haben wir gesehen, dass vieles, was in der Aussagenlogik gültig ist, sich auch auf quantifizierte Formeln übertragen lässt. Die obige Definition zeigt, dass sich die Abbildungen für eine quantifizierte Formel sich kaum von jenen der Aussagenlogik unterscheidet. Es ist sogar so, dass wir einzig Abbildungen für die Quantoren $\forall$ und $\exists$ hinzufügen müssen, davon ab gibt es erstmal keinen Unterschied.

Der Grund hierfür liegt darin, dass die freien Variablen einer quantifizierten Formel, also jene, die nicht an einen Quantor gebunden sind, den Atomen einer aussagenlogischen Formel gleichen und daher auch wie solche behandelt werden können. Allerdings sind solche vielfältigen Belegungen, wie die obige Definition für Formeln mit freien Variablen sie ermöglicht, auch nur ausschließlich für offene quantifizierte Formeln möglich. Wobei auch hier die Anzahl der verschiedenen Belegungen geringer ist als bei einer aussagenlogischen Formel gleicher Länge, nämlich 2^n, wobei $n \in \mathbb{N}$ in diesem Fall die Anzahl der freien Variablen beschreibt, nicht aller Variablen.

Das liegt daran, dass die Belegung einer geschlossenen Formel immer ausschließlich von den Abbildungen abhängt, die für ihre Quantoren gelten. Das führt dazu, dass geschlossene Formeln entweder immer widerspruchsvoll oder immer wahr, also Tautologien, sind. Da man jede offene quantifizierte Formel in eine Formel umformen kann, deren eine Teilformel geschlossen und deren andere Teilformel nur aus freien Variablen besteht, ergibt sich das obige Ergebnis für die Anzahl der verschiedenen Belegungen.

Da die Begriffe widerspruchsvoll und erfüllbar bereits genannt wurden, sollten sie nun auch formal für quantifizierte Formeln definiert werden. Auch hier sieht man schnell, dass die Definition sich kaum von der für die Aussagenlogik unterscheidet. Man kann sie fast schon als Erweiterung verstehen.

Definition 2.1.4 (Erfüllbarkeit)**.** [4] Eine Formel $\Phi \in QBF^*$ ist erfüllbar, genau dann, wenn es eine Belegung $\mathfrak{J}$ mit $\mathfrak{J}(\Phi) = 1$ gibt.
Eine Formel Φ heißt widerspruchsvoll genau dann, wenn für alle Belegungen $\mathfrak{J}(\Phi) = 0$ gilt.
Enthält Φ keine freien Variablen, also $\Phi \in$ QBF, dann sagen wir auch Φ ist falsch, beziehungsweise Φ ist wahr, falls Φ widerspruchsvoll beziehungsweise Φ erfüllbar ist.

Da wir hier festlegen, wann eine Formel erfüllbar ist, können wir auch direkt die Bedingungen für Folgerbarkeit und Äquivalenz einführen:

Definition 2.1.5 (Folgerbarkeit und Äquivalenz)**.** [5] Eine Formel Φ_2 folgt aus Φ_1, $(\Phi_1 \models \Phi_2)$ genau dann, wenn für alle Belegungen $\mathfrak{J}$ gilt $\mathfrak{J}(\Phi_1) = 1 \Rightarrow \mathfrak{J}(\Phi_2) = 1$.
Die Formel Φ_1 ist äquivalent zu Φ_2, $\Phi_1 \approx \Phi_2$ genau dann, wenn $(\Phi_1 \models \Phi_2)$ und $(\Phi_2 \models \Phi_1)$.

[4] [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 367
[5] [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 367

Zum Abschluss dieses Abschnitts zur Einführung quantifizierter Boole'scher Formeln betrachten wir nun noch eine geeignete Datenstruktur, um quantifizierte Formeln auch maschinell verarbeiten zu können. Die folgende Baumstruktur stellt eine Abwandlung der Struktur von Kleine Büning und Lettmann dar, wie sie in *Aussagenlogik: Deduktion und Algorithmen*[6] präsentiert wurde.

Wir wissen aus der Aussagenlogik, dass eine Formel aus den Junktoren

$$\{\vee, \wedge, \neg, \rightarrow, \leftrightarrow\}$$

besteht, die die Variablen

$$\{a, b, c, \ldots, x, y, z\}$$

miteinander verbinden. Zusätzlich haben wir die Hilfszeichen

$$\{(,)\}$$

und im Fall der quantifizierten Formeln die Quantoren.

Quantoren können wir, für eine effiziente Struktur, als weitere Junktoren auffassen. Die Menge der Junktoren erweitert sich also um zwei Elemente.

$$\{\vee, \wedge, \neg, \rightarrow, \leftrightarrow, \forall, \exists\}$$

Zusätzlich nutzen wir das Wissen, dass wir jede Formel, die in Infix-Notation geschrieben ist, auch als geschachtelte Liste in Präfix-Notation darstellen können. So ist der Übergang zu einer Baumstruktur intuitiver.

Die Formel

$$\forall x(a \vee b) \wedge x$$

sähe als geschachtelte Liste so aus

$$[\forall x[\wedge, [\vee, [a], [b],], [x]]]$$

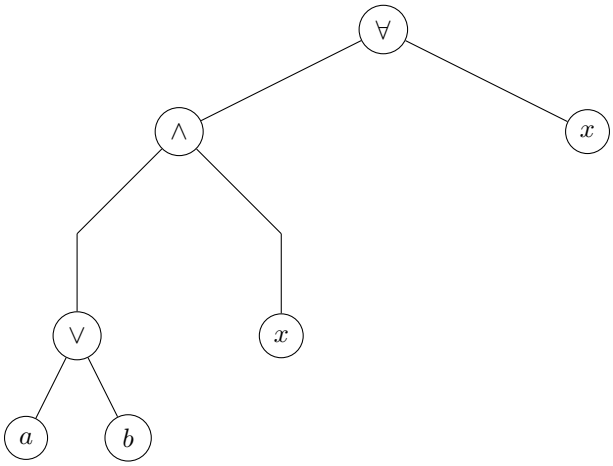
und so

$$(\forall x(\wedge(\vee a, b), x))$$

in Präfix-Notation.

Statt wie üblich von links nach rechts bearbeiten wir eine solche Liste von innen nach außen. Die Baumstruktur ist nun nur noch eine andere Darstellungsform der Liste.

[6] vgl.[KL94] Aussagenlogik: Deduktion und Algorithmen, Kapitel 2.1 Ällgmeine Datenstruktur für aussagenlogische Formeln"



Eine solche Struktur wird von oben nach unten gelesen und dann von unten nach oben aufgelöst und entspricht dem bekannten Syntaxbaum.

Für eine maschinelle Verarbeitung müssen wir nun nur noch festlegen, welche Attribute und Eigenschaften unsere Variablen und Junktoren aufweisen müssen, um die Baumstruktur eindeutig einlesbar und verarbeitbar zu machen.

Hierfür muss jede Variable mindestens einen eindeutigen Namen besitzen, einen Boole'schen Wahrheitswert und einen ebenfalls Boole'schen Hinweis, ob bereits ein Wahrheitswert gesetzt wurde und eine Komponente, die anzeigt ob die Variable frei, $\forall$-gebunden, oder $\exists$-gebunden ist.

Junktoren benötigen hingegen nur einen Bezeichner und ihre fest definierte Funktion.

Damit sind die Knoten unseres Baumes definiert, verbunden werden sie durch eine geeignete Listenstruktur, die von der jeweiligen Programmiersprache, in der die Struktur realisiert werden soll, abhängt.

Jedes Element dieser Liste besteht dann aus einer Referenz auf den Knoten, den es repräsentiert, einer Referenz auf den linken Teilbaum und einer Referenz auf den rechten Teilbaum.

Verbindet ein Junktor zwei Teilformeln, so sind der linke und rechte Teilbaum die entsprechenden Teilformeln.

Bei einer Verbindung von Teilformel und einer Variablen, enthält der linke Teilbaum die Teilformel, während wir rechts die Variable als ein Blatt unseres Baumes haben.

Ist unser Junktor ein Quantor, so ist der linke Teilbaum die quantifizierte Formel, rechts befindet sich der Verweis auf die quantifizierte Variable.

Diese Struktur ist in vielen Sprachen gut realisierbar, erlaubt eine einfache Lesbarkeit und ist, wie sich im späteren Verlauf zeigen wird, einfach zu erweitern. Zudem kann man einfach Teilformeln finden und verarbeiten. Ein Nachteil dieser Struktur ist allerdings, dass zwei Bäume nur durch vollständiges, rekursives Durchlaufen miteinander verglichen werden können.

2.2 Normalformen

Bisher haben wir uns mit quantifizierten Formeln im Allgemeinen beschäftigt. Im Verlauf dieses Abschnitts wollen wir nun einige besondere Formen von quantifizierten Formeln betrachten. Alle quantifizierten Formeln können, wie auch aussagenlogische Formeln, in ihrer Formelstruktur so verändert werden, dass man eine äquivalente Formel in einer Normalform erhält. Es gibt hierbei verschiedene, unterschiedlich definierte Normalformen, von denen wir einige im weiteren Verlauf näher betrachten wollen. Vorweg sei aber gesagt, dass eine solche Umformung in Abhängigkeit von der gewählten Normalform zwar meistens möglich ist, allerdings ist die resultierende Formel, abhängig von der Ausgangsformel und der gewählten Normalform, zum Teil erheblich länger.

In solchen Fällen ist aber auch möglich durch Einführung neuer Variablen Formeln in gewählter Normalform zu erstellen, die zur Ausgangsformel erfüllbarkeitsäquivalent sind. Diese Variante ist oftmals zu bevorzugen, da die resultierenden Formeln in ihrer Länge weit weniger stark anwachsen.
Zudem gibt es bestimmte Normalformen, bei denen die Ausgangsformel bestimmte Voraussetzungen erfüllen muss, damit eine Transformation überhaupt möglich ist. Sind diese Bedingungen nicht gegeben, so ist weder eine äquivalente noch eine erfüllbarkeitsäquivalente Umformung möglich.

Im weiteren Verlauf der Arbeit werden wir uns besonders mit zwei Normalformen beschäftigen. Zum einen die konjunktive Normalform und zum anderen Horn-Formeln, die als eine Sonderform der konjunktiven Normalform betrachtet werden können. Es existieren noch weitere Normalformen, wie zum Beispiel die Negationsnormalform oder die bereits aus Kapitel 2.1 bekannte Pränexnormalform, auf diese wollen wir allerdings nicht weiter eingehen.

Außerdem ist für die Betrachtung von Formeln in QKNF* der Präfixtyp interessant, den wir an dieser Stelle kurz vorstellen wollen.

Definition 2.2.1 (Präfixtyp)**.** [7] Aussagenlogische Formeln haben den Präfixtyp $\Sigma_0 = \Pi_0$. Das ist das leere Präfix.

Sei Φ eine quantifizierte Boole'sche Formel mit Präfixtyp Σ_n beziehungsweise Π_n, so sind die Formeln $\forall x_1 \dots x_k\ \Phi$ beziehungsweise $\exists y_1 \dots y_k\ \Phi$ für beliebiges $k\ \in \mathbb{N}$ vom Typ $\Pi_{(n+1)}$ beziehungsweise $\Sigma_{(n+1)}$.

Der Präfixtyp zeigt zum einen also, welcher Quantor im Präfix führend ist und zum anderen ist aus dem Index ersichtlich, wie viele Quantorenwechsel im Präfix stattfinden. Um dies zu verdeutlichen hier zwei kleine Beispiele.

Sei α ein quantorenfreier Kern.

Die Formel $\forall x_1\ \forall x_2\ \exists y_1\ \forall x_3\ \alpha$ hat den Präfixtyp Π_3

Die Formel $\exists y_1\ \forall x_1\ \exists y_2\ \forall x_2\ \alpha$ hat den Präfixtyp Σ_4

[7] [KL94]Aussagenlogik: Deduktion und Algorithmen, Seite 364

2.2.1 Konjunktive Normalform

Die erste Normalform, die wir in dieser Arbeit näher betrachten wollen, ist die konjunktive Normalform, kurz KNF.

Formeln in KNF bestehen aus durch Konjunktionen verbundenen Disjunktionstermen. Diese Disjunktionsterme bestehen aus Literalen, die durch Disjunktionen mit einander verbunden sind, und werden auch als Klausel bezeichnet. Wenn wir im weiteren Verlauf im Zusammenhang von Formeln in konjunktiver Normalform von Klauseln reden, werden daher immer Disjunktionsterme gemeint sein. Die Anzahl der Literale innerhalb der Klausel ist hierbei nicht relevant, ebenso kann jedes Literal sowohl negiert oder nicht negiert auftreten.

Außerdem sind alle Formeln in konjunktiver Normalform zwingend auch in Negationsnormalform, das heißt, dass alle Negationen, so welche in der Formel vorkommen, unmittelbar vor einer Variablem stehen.

Diese aus der Aussagenlogik bekannte Definition der konjunktiven Normalform lässt sich analog auf quantifizierte Formeln übertragen. Zusätzlich gilt bei quantifizierten Formeln allerdings auch, dass sie sich immer in Pränexnormalform befinden. Da für diese Arbeit bereits in Kapitel 2.1 festgelegt wurde, dass alle betrachteten quantifizierten Boole'schen Formeln sich in Pränexnormalform befinden, müssen wir uns um diese Einschränkung keine Sorgen machen.

Damit ergeben sich analog zu den bereits definierten Klassen QBF* und QBF zwei neue Klassen.

Die Klasse QKNF* $\subseteq$ QBF* beschreibt die Menge der Formeln in QBF*, die in konjunktiver Normalform sind.

Entsprechend beschreibt die Klasse QKNF $\subseteq$ QKNF* die Menge der geschlossenen Formeln in konjunktiver Normalform.

Die einfachste Transformation einer quantifizierten Formel zu einer quantifizierten Formel in konjunktiver Normalform wäre hierbei, dass wir die aus der Aussagenlogik bekannten Umformungsregeln, wie zum Beispiel die De Morgan'schen Regel, auf den Kern unserer Formel anwenden, und diesen so in eine konjunktive Normalform bringen, während wir das Präfix unberührt lassen. So bleibt der Präfixtyp unserer Formel erhalten, der Kern wächst in seiner Länge aber gegebenenfalls exponentiell. Wie bereits eingangs erwähnt, lässt sich dieses Problem der stark wachsenden Formellängen umgehen, indem wir statt die Formel äquivalent umzuformen eine erfüllbarkeitsäquivalente Umformung anstreben.

Im Folgenden wird ein Weg[8] präsentiert, um eine quantifizierte Formel zu einer quantifizierten Formel in konjunktiver Normalform umzuformen. Wir setzen voraus, dass die Formel sich in der in Kapitel 2.1 vorgestellten Baumstruktur befindet.

[8] [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 376

Input: Eine beliebige quantifizierte Formel in Baumstruktur
Output: Eine zur Anfangsformel erfüllbarkeitsäquivalente Formel in QKNF*

begin
  $\alpha :=$ Kern(Φ)
  Var := Die Menge aller Variablen einer Formel
  $\Pi :=$ Präfix(Φ)
    `// bestehend aus den Quantoren` $\forall$ `und` $\exists$
  $\Phi_{\text{KNF}} :=$ transformierter Kern in KNF
  $V :=$ Var(Φ_{KNF}) − Var(α)
    `// V ist dargestellt als Menge` $\{v_1, v_2, \ldots, v_n\}$
  $\Phi_{\text{Q-KNF}} := \Pi\ \exists v_1 \ldots \exists v_p \Phi_{\text{KNF}}$
    `// Anpassung des Präfixes`
end

Algorithmus 1: Trans-Q-KNF

Zuerst unterteilen wir unsere Formel in ihren Kern und ihr Präfix. Das liegt darin begründet, dass wir beide getrennt voneinander transformieren werden.

Den Kern transformieren wir so, als handelte es sich um eine aussagenlogische Formel. Wie genau das passiert, werden wir im weiteren Verlauf noch genauer betrachten. Sobald unser Kern in konjunktiver Normalform ist, bilden wir eine Differenzmenge. Wir betrachten die Menge der Variablen unseres Kerns in konjunktiver Normalform und entfernen aus dieser Menge alle Variablen, die bereits vor der Umformung in α vorkamen.

So erhalten wir die Menge der Variablen, die für die Transformation in eine konjunktive Normalform zur Formel hinzugefügt werden mussten. Diese Variablen binden wir nun an $\exists$-Quantoren, die wir am rechten Ende des Präfixes anfügen.

Es ist hierbei wichtig zu beachten, dass diese neuen Quantoren zwingend am inneren Ende angefügt werden müssen. Nur so kann sichergestellt werden, dass die Erfüllbarkeitsäquivalenz erhalten bleibt, da bei quantifizierten Formeln die Erfüllbarkeit auch immer davon abhängt, welche Quantoren über andere regieren.

Ein kleines Beispiel, um diesen Umstand zu verdeutlichen wäre die folgende, einfache Formel

$$\forall x(x \vee \neg x).$$

Wir können eine $\exists$-quantifizierte Variable y hinzufügen, ohne etwas an der Aussage der Formel zu verändern, aber nur, solange der $\forall$-Quantor über den $\exists$-Quantor regiert.

$$\forall x \exists y(x \vee y) \wedge (\neg x \vee \neg y)$$

Andersherum, also

$$\exists y \forall x(x \vee y) \wedge (\neg x \vee \neg y)$$

erhielten wir eine nicht erfüllbarkeitsäquivalente Formel. Im weiteren Verlauf, insbesondere im Kapitel zur Resolution quantifizierter Formeln, wird auf diese Eigenheit

quantifizierter Formeln und die Relevanz der Bindungsbereiche von Quantoren noch weiter eingegangen werden.

Entsprechend bleibt der Präfixtyp unserer neuen Formel nur dann gleich zu dem der Ausgangsformel, wenn das Präfix der Ausgangsformel auf einen Existenzquantor endete. War der letzte Quantor des Ausgangspräfixes ein Allquantor, so ändert sich der Präfixtyp, indem der Index um 1 erhöht wird.

Grundsätzlich ist es also recht unkompliziert, eine beliebige quantifizierte Formel in konjunktive Normalform zu bringen. Es ist, wie man am obigen Algorithmus sieht, nicht viel aufwendiger als die Transformation einer aussagenlogischen Formel in konjunktive Normalform, denn genau das macht den Hauptteil dieser Umwandlung aus. Deshalb wollen wir an dieser Stelle, auch wenn es eigentlich weniger mit quantifizierten Boole'schen Formeln zu tun hat, betrachten, wie man eine beliebige aussagenlogische Formel zu einer Formel in konjunktiver Normalform umformt.[9]

Man besucht alle Knoten des Formelbaums und ruft die Funktion rekursiv immer für die nächste Teilformel auf, bis man an den Blättern des Baumes ankommt. Von dort bringt man die Klauseln in die gewünschte Form, also mit Konjunktionen verbundene Disjunktionsterme, und fügt diese Stück für Stück an die neue Formel an, wobei bei Bedarf neue Variablen erzeugt werden.

Ausführliche, in einem an Pascal und C angelehnten Pseudocode geschriebene Versionen dieser Algorithmen kann man bei Kleine Büning und Lettmann in ihrer Publikation *Aussagenlogik: Deduktion und Algorithmen* auf den Seiten 31 und 376 finden.

Betrachtet man Formeln in QKNF* genauer, fällt auf, dass sie in Bezug auf ihre Quantoren und die damit verbundene Erfüllbarkeit gewisse Eigenheiten aufweist.

Erweitern wir eine aussagenlogische Formel α in konjunktiver Normalform um einen Existenzquantor, also $\exists y\ \alpha$, ergeben sich folgende Auswirkungen für die Belegung, nämlich, dass $\mathfrak{J}(\exists y\ \alpha) = 1$ genau dann gilt, wenn $\mathfrak{J}(\alpha[y/1]) = 1$ oder $\mathfrak{J}(\alpha[y/0]) = 1$ gilt. Das heißt α ist dann erfüllbar, wenn alle Klauseln erfüllbar sind, die y nicht enthalten und zusätzlich entweder die Menge der Klauseln erfüllbar ist, die y enthalten, oder die Menge der Klauseln die $\neg y$ enthalten. Klauseln die sowohl y als auch $\neg y$ sind tautologisch und immer erfüllbar.

Erweitern wir α stattdessen um einen Allquantor, also $\forall x\ \alpha$, zeigt sich, dass α nur erfüllbar ist, wenn jede Klausel, mit Ausnahme der generell erfüllbaren tautologischen Klauseln, erfüllbar ist. Die Erfüllbarkeit dieser Klauseln muss unabhängig von x gegeben sein, weshalb sich ergibt, dass die Vorkommen von x dann gestrichen werden können.

Dieser Umstand wird formal durch das folgende Lemma beschrieben:

Lemma 2.2.1. [10] Sei $\Phi = Q_1 z_1 \dots Q_n z_n\ \forall x_1 \dots \forall x_n \alpha$ eine quantifizierte KNF-Formel, in der keine Klauseln vorkommen, die nur Literale aus $x_1, \dots, x_n, \neg x_1, \dots, \neg x_n$ enthalten oder die tautologisch sind aufgrund eines Vorkommens von x_i und $\neg x_i$ für ein

[9] vgl.[KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 31

[10] [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 379

Input: Eine beliebige aussagenlogische Formel in Negationsnormalform
Output: Eine erfüllbarkeitsäquivalente aussagenlogische Formel in konjunktiver Normalform

```
1  begin
2      γ:= NULL ;                                          // neue Zielformel
3      while der Baum der Ausgangsformel unbesuchte Knoten hat do
4          gehe zum nächsten Knoten
5          if Knoten = Kind_links ∧ Kind_rechts then
6              return Trans-KNF(Kind_links) ∧ Trans-KNF(Kind_rechts)
7          end
8          if Knoten = Kind_links ∨ Kind_rechts then
9              if Kind_links = NKind_links ∧ NKind_rechts then
10                 σ1 :=Trans-KNF(NKind_links)
                    σ2 :=Trans-KNF(NKind_rechts)
11                 A := neue Variable, nicht der Variablenname
12                 if γ = NULL then
13                     γ := (¬A ∨ σ1) ∧ (¬A ∨ σ2)
14                 else
15                     γ := γ ∧ (¬A ∨ σ1) ∧ (¬A ∨ σ2)
16                 end
17                 return A ∨ Trans-KNF(σ2)
18             end
19             else if Kind_rechts = NKind_links ∧ NKind_rechts then
20                 Analog zum Block für das linke Kind des Knotens
21             end
22             else
23                 return Trans-KNF(Kind_links) ∨ Trans-KNF(Kind_rechts)
24             end
25         end
26     end
27     return γ
28 end
```

Algorithmus 2: Trans-KNF

$1 \leq i \leq n$. Dann gilt: $\Phi \approx Q_1 z_1 \ldots Q_n z_n \alpha^*$, wobei jede Klausel aus α^* durch Streichung aller Vorkommen von x_i und $\neg x_i$ für $1 \leq i \leq n$ aus einer Klausel von α entstanden ist.

Beweis. Es ist zu zeigen, dass für jede Formel $\forall x\ \alpha$ in konjunktivier Normalform eine erfüllbarkeitsäquivalente Formel β, ebenfalls in konjunktiver Normalform, gibt, die aus α entsteht indem man alle Vorkommen einer bestimmten Variablen, hier x, aus der Ausgangsformel streicht.

Sei α eine aussagenlogische Formel in konjunktiver Normalform, die den Anforderungen des Lemmas genügt, und β eine aussagenlogische Formel, die aus der Streichung aller Vorkommen der Variablen x aus α entstanden ist.

1. $\alpha = (a_1 \vee \cdots \vee a_n) \approx \beta = (a_1 \vee \cdots \vee a_n)$
2. $\alpha = (a_1 \vee \cdots \vee a_n \vee x) \approx \beta = (a_1 \vee \cdots \vee a_n) \quad$ für $x = 0$
3. $\alpha = (a_1 \vee \cdots \vee a_n \vee \neg x) \approx \beta = (a_1 \vee \cdots \vee a_n) \quad$ für $x = 1$

Aus Definition 2.1.3 wissen wir, dass eine solche Belegung durch den Allquantor $\forall$ realisiert werden kann. Daraus ergibt sich also:

$$\forall x\ \alpha \approx \beta$$

□

Dieser Umstand führt, zusammen mit der folgenden Beobachtung dazu, dass wir über die Möglichkeit verfügen, Formeln in QKNF* zu vereinfachen. In Formeln, deren innerster Quantor der Allquantor ist, können wir diesen, und die an ihn gebundenen Variablen, einfach eliminieren, da es keinen Existenzquantor gibt, über den er Präzedenz hat. Diese Eigenschaft lässt sich wie folgt zusammenfassen:

Korollar. [11] Sei Φ eine Formel in QKNF*. Dann gibt es eine Formel $\Psi \in$ QKNF* mit folgenden Eigenschaften:

1. $\Phi \approx \Psi$
2. Enthält Φ keine tautologischen Klauseln, so enthält der Kern von Ψ keine im aussagenlogischen Sinne tautologischen Klauseln.
3. In keiner Klausel von Ψ kommt ein Literal mehrfach vor.
4. Für jede allquantifizierte Variable x_i einer Klausel existiert in derselben Klausel eine existenzquantifizierte Variable y_j mit $x < y$
5. Das Präfix von Ψ enthält nur Quantoren über Variablen, die auch im Kern von Ψ auftreten.

[11] [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 380

Beweis. Die oben genannten Eigenschaften seien hier exemplarisch zu zeigen: Sei $\Pi_{x_1,\ldots x_{n-2},y_1,\ldots,y_{n-1}}$ ein beliebiges Präfix und α ein quantorenfreier Kern in konjunktiver Normalform. Die Formel Φ sei nun

$$\Pi_{x_1,\ldots x_{n-2},y_1,\ldots,y_{n-1}} \forall x_{n-1} \exists y_n \forall x_n \ (\alpha) \wedge (\neg x_{n-1} \vee y_n) \wedge (y_n \vee x_n)$$

Aus Lemma 2.2.1 ergibt sich dann folgende Formel Ψ

$$\Pi_{x_1,\ldots x_{n-2},y_1,\ldots,y_{n-1}} \forall x_{n-1} \exists y_n (\alpha) \wedge (\neg x_{n-1} \vee y_n) \wedge y_n$$

□

Wir können also festhalten, dass wir die Normalform QKNF* noch weiter normieren können, indem wir festlegen, dass der innerste Quantor des Präfix immer ein Existenzquantor ist. Eine solche Normierung ist schnell durchzuführen und vereinfacht sowie verkürzt Formeln, weshalb in den folgenden Kapiteln, wenn von Formeln in QKNF* die Rede ist, von normierten Formeln in QKNF* ausgegangen werden wird, außer es wird explizit erwähnt, dass dem nicht so ist.

2.2.2 Quantifizierte Horn-Formeln

Eine weitere interessante und häufig gebrauchte Klasse der Normalformeln sind die Horn-Formeln. Eine Horn-Formel ist dabei eine Formel in konjunktiver Normalform, die sich dadurch auszeichnet, dass in jeder Klausel höchstens ein positives Literal vorkommt. Eine solche Klausel bezeichnet man entsprechend auch als Horn-Klausel.

Wir erweitern also unsere die Unterklassen von QBF* um zwei weitere Klassen.

Zum einen haben wir die Klasse QHORN* $\subseteq$ QBF*, die Menge der quantifzierten Hornformeln und zum anderen QHORN $\subseteq$ QHORN*, die Menge der geschlossenen, quantifizierten Hornformeln.

Diese Klasse QHORN* weist, ähnlich wie ihr aussagenlogisches Äquivalent, praktische Eigenschaften auf. Das Erfüllbarkeitsproblem zum Beispiel lässt sich in QHORN*, wie wir später noch sehen werden, in maximal quadratischer Zeit lösen. Ebenso werden wir uns damit beschäftigen, welche Bedeutung Horn-Formeln im Bereich der Resolution haben.

Allerdings teilt sie sich mit der Klasse noch eine weitere, weniger wünschenswerte Eigenschaft, nämlich die, dass die Klasse QHORN* im Vergleich zu QBF* nur sehr wenige Formeln beinhaltet, da es keine Möglichkeit gibt, eine beliebige quantifizierte Formel in eine quantifizierte Horn-Formel zu transformieren.

Oft kann es aber auch hilfreich sein, wenn Formeln in ihrer Struktur einer Horn-Formel zumindest sehr ähnlich sind. Im Kapitel zur Resolution quantifizierter Formeln werden wir eine solche Klasse vorstellen, nämlich die Klasse QEHORN*, die Klasse der erweiterten quantifizierten Horn-Formeln. Sie ist, wie wir sehen werden, eine echte Oberklasse der Horn-Formeln und umfasst damit mehr Formeln als QHORN* und bietet hinsichtlich der Resolution bessere Möglichkeiten, als es für Formeln die nur in QKNF* liegen der Fall wäre.

Theoretisch lässt sich auch jede quantifizierte Horn-Formel zu einer aussagenlogischen Formel umformen.

Wir führen uns noch einmal in Erinnerung, wenn wir die Formel $\Phi = \exists y(y, z)$ nehmen, dann könnten wir äquivalent auch schreiben $\Phi = (1, z) \vee (0, z)$. Auch für $\forall$-quantifzierte Variablen gibt es eine solche Äquivalenz, sei $\Phi = \forall x(x, z)$, dann können wir äquivalent auch schreiben $\Phi = (1, z) \wedge (0, z)$.

Wollte man nun also sämtliche $\forall$-Variablen aus einer Formel entfernen, wäre der einfachste Weg eine Erweiterung der Formel, also jedes Vorkommen der Variable durch die oben genannte Formel zu ersetzen, indem man zwei Kopien der Ursprungsformel erstellt und in einer davon die zu ersetzende Variable durch 0 und in der anderen durch 1 ersetzt. Diesen Schritt müsste für jede $\forall$-Variable durchgeführt werden. Die Länge der Zielformel würde zusätzlich wachsen, da man $\exists$-Variablen, die von $\forall$-Variablen abhängen ebenfalls berücksichtigen müsste, sodass gegebenenfalls weitere Kopien angehängt werden müssen, um die Äquivalenz zu erhalten.

Aus diesem Grund ist es für gewöhnlich nicht ratsam, quantifizierte Formeln zu aussagenlogischen Formeln transformieren zu wollen, im Sonderfall der quantifizierten Horn-Formeln gibt es aber eine Möglichkeit die $\forall$-quantifizieren Variablen effizient zu entfernen und so eine Zielformel zu erhalten, deren Kern wie eine aussagenlogische Formel behandelt werden kann.

Kleine Büning und Bubeck zeigen in ihrer Veröffentlichung *Models and quantifier elimination for quantified Horn formulas*, dass es möglich ist, eine quantifizierte Horn-Formel so umzuwandeln, dass sie keine $\forall$-Variablen mehr enthält. Eine solche Transformation ist möglich in höchsten quadratischer Zeit $\mathcal{O}(rn)$, wobei n die Länge der Ausgangsformel ist und r die Anzahl $\forall$-Variablen. Eine quadratische Laufzeit erhalten wir hierbei genau dann, wenn $r = n$, also alle Variablen der Ausgangsformel $\forall$-Variablen sind. Zudem hängt auch die Länge der Zielformel von der Ausgangsformel und der Anzahl der $\forall$-Variablen ab und wächst ebenfalls höchstens quadratisch.

Da wir uns im Folgenden mit solchen Formeln, die keine $\forall$-Variablen enthalten, beschäftigen wollen, führen wir an dieser Stelle noch eine weitere Klasse quantifizierter Formeln ein, $\exists$HORN. Für bessere Übersichtlichkeit schreiben wir, wenn eine quantifizierte Horn-Formel ohne $\forall$-Variablen vorliegt $\Phi \in \exists$HORN.

Definition 2.2.2 ($\exists$HORN)**.** [12] Eine Formel $\Phi \in \exists$HORN ist eine quantifizierte Formel, in deren Kern φ nur Klauseln in Horn-Struktur vorliegen und die ausschließlich aus $\exists$-quantifizierten und freien Variablen besteht.

Eine solche Formel Φ hat die Form $\Phi = \exists y_1 \dots \exists y_n \varphi$. Es sind keine $\forall$-Literale vorhanden.

Der folgende, von Bubeck und Kleine Büning übernommene, Algorithmus erzeugt für jede Formel $\Phi \in$ QHORN* eine äquivalente Formel $\Phi' \in \exists$HORN.[13]

Der gezeigte Algorithmus basiert darauf, dass für quantifizierte Horn-Formeln keine vollständige Erweiterung aller $\forall$-Variablen notwendig ist, da die Erfüllbarkeit der Formel

[12] [BK08]
[13] [BK08]

Input: Eine beliebige quantifizierte Horn-Formel Φ mit
$\Phi(z) = \forall X_1 \exists Y_1 \ldots \forall X_r \exists Y_r \ \phi(X_1, \ldots, X_r, Y_1, \ldots, Y_r, z)$,
$X_i = x_{i,1}, \ldots, x_{i,n_i}$ und $Y_i = y_{i,1}, \ldots, y_{i,m_i}$

Output: Der Kern φ einer zu Φ äquivalenten Formel $\Phi' \in \exists$HORN

```
1  begin
2      φ = ∅
3      for i = 1; i <= r; i++ do
4          for j = 1, j = n_i, j++ do
5              A_{x_{i,j}} = 1
6          end
7      end
8      for i = 1; i <= r; i++ do
9          for j = 1; j <= n_i; j++ do
10         end
11         A_{x_{i,j}} = 0
12         for k = i; k <= r; k++ do
13             for l = 1; l <= m_k; l++ do
14             end
15             y'_{k,l} = neue ∃-Variable
16         end
17         φ = φ ∪ φ[x/A_x, y/y']
18         A_{x_{i,j}} = 1  for l = 1; l <= m_i; l++ do
19             y'_{i,l} = neue ∃-Variable
20         end
21     end
22     φ = φ ∪ φ[x/A_x, y/y']
23 end
```

Algorithmus 3: Transform-$\exists$HORN

bereits erhalten bleibt, wenn maximal eine ∀-Variable durch 0 und alle anderen durch 1 ersetzt werden. Man muss also nicht alle möglichen Modelle betrachten, sondern nur jene, für die höchstens eine ∀-Variable durch 0 ersetzt wurde. Aus diesen Teilmodellen lässt sich dann der Kern einer neuen Formel zusammensetzen, die äquivalent zur Ausgangsformel ist, aber dadurch, dass sie weniger mögliche Belegungen abbildet, weitaus kürzer ist, als nach der herkömmlichen Erweiterungsmethode.

Diesen Umstand nutzt der in der Abbildung gezeigte Algorithmus aus, in dem er in der Hauptschleife immer genau eine ∀-Variable auf falsch setzt und alle anderen auf wahr und für alle von dieser Variable abhängigen ∃-Variablen neue, unabhängige ∃-Variablen in die Formel einfügt. Jeder Durchlauf dieser Schleife erzeugt eine Teilformel. Diese Teilformel wird an den bisherigen Kern der Zielformel angehängt, sodass am Ende die gewünschte Zielformel entsteht.

Beweise für die Äquivalenz von Ziel- und Ausgangsformel sowie eine genauere Herleitung von teilweisen Erweiterungsmodellen quantifizierter Formeln werden an dieser Stelle nicht betrachtet, können aber bei Bubeck und Kleine Büning nachgelesen werden.

3 Resolution quantifizierter Formeln

Nachdem wir quantifizierte Boole'sche Formeln und einige für sie definierte Normalformen betrachtet haben, wollen wir uns nun einem weiteren wichtigen Thema zuwenden. Der Widerlegung der Erfüllbarkeit von Formeln der Klasse QKNF* durch Resolution.

Diese Herangehensweise, um die Unerfüllbarkeit einer Formel zu zeigen ist bereits aus der Aussagenlogik bekannt und wird hier nun erweitert, um auch auf quantifizierte Formeln zuzutreffen.

Bevor wir allerdings diese Erweiterungen einführen, kommt ein kurzer Rückblick darauf, was Resolution ist.[1]

Um eine Resolution durchzuführen, benötigen wir zwei Klauseln in konjunktiver Normalform und eine Variable, wobei diese Variable in einer der beiden Klauseln positiv, und in der anderen negiert auftreten muss. Also beispielsweise die Klauseln A und B, mit der Variablen c, sodass $c \in A$ und $\neg c \in B$. Sind zwei solche Klauseln vorhanden, können wir, mit Hilfe der Resolutionsregel, eine neue Klausel $D = (A \setminus c \cup B \setminus \neg c)$ bilden. Dies geschieht, indem wir die beiden Vorkommen von c streichen, da die Erfüllbarkeit der zu resolvierenden Klauseln nicht von c abhängen kann. Wäre $c = 1$ so ist zwar A erfüllt, B aber nicht. Für $\neg c$ wäre B erfüllt, A aber nicht. Da aber für die Erfüllbarkeit der Formel beide Klauseln erfüllt sein müssen, müssen sie offensichtlich unabhängig von c erfüllbar sein.

Nachdem wir also beide Vorkommen von c gestrichen haben, bilden wir die Vereinigung der Restklauseln. Eine solche Klausel D nennt man Resolvente.

Grafisch würde man eine solche Resolutionsableitung zum Beispiel so darstellen:

$$\begin{array}{ccc} a \vee c & & b \vee \neg c \\ & \diagdown \quad \diagup & \\ & a \vee b & \end{array}$$

Diese Resolvente kann selbst auch wieder in einer Resolution genutzt werden, um eine neue Resolvente zu erzeugen. Diesen Prozess kann man so lange durchführen, bis kein Resolutionsschritt innerhalb einer Formel mehr möglich ist, oder man zur leeren Klausel gelangt. Dieses Verfahren nennt man auch Resolutionsableitung. Erreicht man hierbei die leere Klausel schreibt man dies auch als $\alpha \underset{Res}{\vdash} \square$. Dies ist dann die Resolutionswiderlegung der Formel und die Formel ist unerfüllbar.

Dieser kleine Rückblick soll genügen, um die Resolution aussagenlogischer Formeln kurz zusammenzufassen. Beschäftigen wir uns nun mit der Resolution quantifizierter

[1] inspiriert durch [VK19] die Veranstaltung "Logik und Formale Systeme"

Formeln. Es fällt vielleicht auf, dass zwischen der in Kapitel 2.2.1 vorgestellten Normierung von Formeln in QKNF und der Resolution gewisse Parallelen vorhanden sind. In beiden Fällen wird eine Variable aus einer Klausel entfernt, um so eine neue Klausel zu bilden. Um diese Parallele zu verdeutlichen, sei die Formel Φ wie folgt gegeben

$$\Phi = \forall x \ (y \vee z \vee x).$$

Wir könnten diese Formel auch alternativ darstellen als

$$\Phi' = (y \vee z \vee x) \wedge (y \vee z \vee \neg x).$$

Die Resolvente dieser beiden Klauseln wäre

$$\Psi = (y \vee z)$$

und ist, wie wir aus Lemma 2.2.1 wissen, erfüllbarkeitsäquivalent zu Φ.

Die Streichung nicht regierender, also im Präfix ganz rechts stehender, Allquantoren und ihrer Variablen ist also in gewisser Weise auch eine Form der Resolution und kann so auch für die Resolutionswiderlegung einer Formel genutzt werden.

Wir können die Resolutionsregel auch auf existenzquantifizierte Variablen anwenden. Hierfür sind keine Abwandlungen nötig und die so resultierenden Resolventen können wie in der Aussagenlogik weiter genutzt werden. Zusätzlich kann es passieren, dass diese Resolventen allquantifizierte Variablen haben, die durch den Wegfall der existenzquantifizierten Variable nun auch gestrichen werden können.

Außerdem können wir festhalten, dass eine Klausel, die nur allquantifizierte Variablen enthält und keine Tautologie ist, dafür sorgt, dass die Formel Φ, in der eine solche Klausel enthalten ist, immer widerspruchsvoll ist. Dies liegt in der Definition des Allquantors $\forall$ begründet und sorgt dafür, dass wir Klauseln, die nur aus allquantifizierten Variablen bestehen auch durch die leere Klausel ersetzen können.

Damit kommen wir zur Definition der Resolution quantifizierter Formeln, wobei wir vorher zwei neue Schreibweisen einführen wollen, um den Text konziser zu halten.

Wir schreiben $\forall$-Literal, wenn wir das Literal einer $\forall$ gebundenen Variablen beschreiben wollen. Analog ergibt sich das $\exists$-Literal, wobei dieses auch das Literal einer freien Variablen beschreiben kann.

Eine $\forall$-Klausel ist eine Klausel, die nur $\forall$-Literale enthält.

Definition 3.1 (Q-Resolution)**.** [2] Sei $\Phi = \Pi\alpha$ eine Formel in QKNF* mit Kern α und Präfix Π. Seien α_1 und α_2 nicht tautologische Klauseln von α, wobei α_1 das $\exists$-Literal y enthält und α_2 das Literal $\neg y$. Eine Q-Resolvente σ aus α_1 und α_2 erhält man durch Anwendung der Schritte 1 bis 3:

[2] [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 383

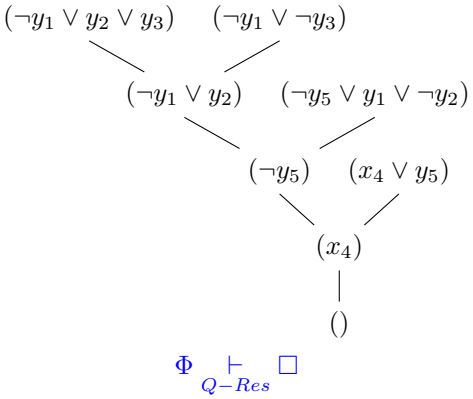
1. Eliminiere alle Vorkommen von $\forall$-Literalen in α_i, die nicht kleiner sind als jedes in α_i vorkommendes $\exists$-Literal$(i = 1, 2)$. Die Ergebnisklauseln sind α_1' und α_2'.
2. Eliminiere die Vorkommen von y aus α_1' und die von $\neg y$ aus α_2'. Wir erhalten α_1'' und α_2''.
3. $\sigma = \alpha_1'' \vee \alpha_2''$.

Um die Resolution einer quantifizierten Formel zu beschreiben, schreiben wir

$$\Phi \underset{Q-Res}{\vdash} \sigma.$$

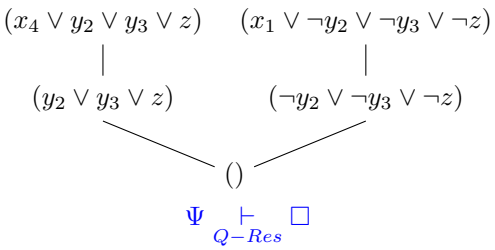
Im Folgenden soll die obige Definition durch einige Beispiele veranschaulicht werden.

1. $\Phi = \exists y_1 \exists y_2 \exists y_3 \forall x_4 \exists y_5 (x_4 \vee y_5) \wedge (\neg y_1 \vee y_2 \vee y_3) \wedge (\neg y_5 \vee y_1 \vee \neg y_2) \wedge (\neg y_1 \vee \neg y_3)$



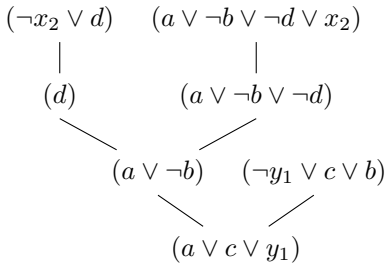
$\Phi \underset{Q-Res}{\vdash} \square$

2. $\Psi = \forall x_1 \exists y_2 \exists y_3 \forall x_4 (x_1 \vee \neg y_2 \vee \neg y_3 \vee \neg z) \wedge (x_4 \vee y_2 \vee y_3 \vee z)$



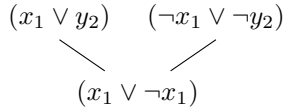
$\Psi \underset{Q-Res}{\vdash} \square$

3. $\Theta = \exists y_1 \forall x_2 (a \vee \neg b \vee \neg d \vee x_2) \wedge (\neg y_1 \vee c \vee b) \wedge (\neg x_2 \vee d)$



$\Theta \underset{Q-Res}{\vdash} (a \vee c \vee y_1)$, Θ ist also erfüllbar

4. $\Xi = \forall x_1 \exists y_2 (x_1 \vee y_2) \wedge (\neg x_1 \vee \neg y_2)$



$\Xi \underset{Q-Res}{\vdash} (x_1 \vee \neg x_1)$, was eine Tautologie ist

Es ist bekannt, dass der Resolutionskalkül für die Aussagenlogik widerlegungsvollständig ist, das heißt, dass eine Formel unerfüllbar ist, genau dann, wenn man in Folge der Resolutionsableitung die leere Klausel erhält, es also eine Resolutionswiderlegung für diese Formel gibt.

Dass diese wichtige Eigenschaft der Resolution auch für die Resolutionsdefinition für quantifizierte Formeln in QKNF* gilt, zeigt der folgende Beweis.

Beweis der Widerlegungsvollständigkeit der Q-Resolution für die Klasse QKNF.* [3] Wir werden im Folgenden einen Vollständigkeitsbeweis so führen, dass wir zeigen, dass für jede Formel in QKNF* die nicht erfüllbar ist die leere Klausel als letzter Resolutionsschritt erreicht werden kann.

Dafür gehen wir erst einmal davon aus, dass alle betrachteten Formeln geschlossen sind.

Eine solche Annahme ist möglich, da wir für jede offene quantifizierte Formel die freien Variablen an einen $\exists$-Quantor binden können, den wir am linken, also vorderen, Ende des Präfixes anfügen, ohne, dass sich dabei die Erfüllbarkeit der Formel ändert.

Induktiv können wir dann wie folgt über die Präfixlänge n zeigen, dass sich für alle unerfüllbaren Formeln die leere Klausel herleiten lässt.

Es gelte, dass die betrachteten Formeln falsch sind. Es ergeben sich nun vier verschiedene Fälle, die zu betrachten sind. Zwei Fälle für $n = 1$, und zwei Fälle für $n > 1$. Diese

[3] vgl.[KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 386f

Fallunterscheidungen ergeben sich daraus, dass wir zwischen $\exists$- und $\forall$-Quantoren unterscheiden müssen. Der Grund hierfür liegt auf der Hand, wenn man sich im Folgenden die beiden Fälle für ein Präfix aus nur einem Quantor ansieht.

Sei $n = 1$ und sei $\Phi = \forall x_1(a_1 \vee \cdots \vee a_m)$.

Da Φ geschlossen ist, ist x_1 die einzige Variable. Alle Klauseln a_i enthalten also entweder x_1, oder sind bereits leer. Wir können sämtliche tautologischen Klauseln ignorieren und aus der Formel entfernen. Da vorausgesetzt ist, dass die Formel falsch ist, bleibt mindestens eine nicht tautologische Klausel, die nur $\forall$-gebundene Variablen, nämlich x_1, enthält. Eine solche Klausel ist immer unerfüllbar und wird im Folgenden zur Vereinheitlichung mit der leeren Klausel gleichgesetzt. Es folgt $\Phi \underset{Q-Res}{\vdash} \square$.

Sei $n = 1$ und sei $\Phi = \exists y_1(a_1 \vee \cdots \vee a_m)$.

In diesem Fall können wir unsere quantifizierte Formel so behandeln, als handle es sich um eine aussagenlogische Formel, da wir zu Beginn bereits gesehen haben, dass es für die Erfüllbarkeit einer Formel keinen Unterschied zwischen einer $\exists$-Variablen und einer freien Variablen gibt, solange kein $\forall$-Quantor weiter links steht als der $\exists$-Quantor der entsprechenden Variablen. Daraus folgt, dass die Q-Resolution der aussagenlogischen Resolution entspricht, es folgt also $\Phi \underset{Q-Res}{\vdash} \square$.

Sei $n > 1$ und sei $\Phi = \exists y_1 \ldots (a_1 \vee \cdots \vee a_m)$.

Nun seien Φ_1 und Φ_0 zwei Formeln, in denen jedes Vorkommen von y_1 in Φ durch 1 beziehungsweise 0 ersetzt wurde, während der Rest der Formel beibehalten wird. Mit diesen derart verkürzten Formeln und der Induktionsvoraussetzung für $n - 1$ können wir $\Phi_0 \underset{Q-Res}{\vdash} \sigma_0$ und $\Phi_1 \underset{Q-Res}{\vdash} \sigma_1$ herleiten, wobei σ_0 beziehungsweise σ_1 nicht-tautologische $\forall$-Klauseln oder die leere Klausel sind.

Da wir die kleinste Variable y_1 aus Φ ersetzt haben, können alle Resolutionsschritte, die wir in Φ_0 beziehungsweise Φ_1 durchführen, auch für Φ durchgeführt werden, da die ersetzten Literale y_1 und $\neg y_1$ die Resolutionsschritte für Φ_0 beziehungsweise Φ_1 nicht beeinflussen und da y_1 die kleinste Variable ist, gibt es auch keine $\forall$-Literale, deren mögliche Streichung von y_1 abhinge.

Führen wir diese Resolutionsschritte also für Φ aus, erhalten wir entweder
$\Phi \underset{Q-Res}{\vdash} \sigma_0$ oder $\Phi \underset{Q-Res}{\vdash} \sigma_1$
oder
$\Phi \underset{Q-Res}{\vdash} y_1 \vee \sigma_0$ und $\Phi \underset{Q-Res}{\vdash} \neg y_1 \vee \sigma_1$.

Im letztgenannten Fall ist y_1 das kleinste Literal, das heißt, die Quantoren aller möglichen $\forall$-Literale stehen rechts vom Quantor von y_1, und das heißt, dass wir $\sigma_1$und σ_0 streichen können, wir führen dann einen letzten Resolutionsschritt mit den verbliebenen Klauseln y_1 und $\neg y_1$ durch und kommen so zu $\Phi \underset{Q-Res}{\vdash} \square$.

Sei $n > 1$ und sei $\Phi = \forall x_1 \ldots (a_1 \vee \cdots \vee a_m)$.

Wir führen die gleiche Ersetzung durch, wie im vorherigen Fall, erhalten also wieder Φ_1 und Φ_0, wobei dieses Mal selbstverständlich x_1 ersetzt wird.

Da Φ falsch ist, muss Φ_0 oder Φ_1 falsch sein. Da beide Fälle analog zueinander zu behandeln wären, betrachten wir hier nur einen Fall genauer.

Sei also Φ_1 falsch. Wir erhalten also $\Phi_1 \underset{Q-Res}{\vdash} \sigma_1$, wobei σ_1 wieder eine nicht tautologische $\forall$-Klausel ist.

Da für diese Resolutionsschritte nur Klauseln relevant sind, die entweder lediglich $\neg x_1$ oder überhaupt keine Vorkommen von x_1 aufweisen, können alle auf Φ_1 vorgenommenen Resolutionsschritte auch für Φ vorgenommen werden und wir erhalten

$\Phi \underset{Q-Res}{\vdash} \sigma_1$ oder $\Phi \underset{Q-Res}{\vdash} \neg x_1 \vee \sigma_1$.

Für den ersten Fall ist die Herleitung der leeren Klausel beziehungsweise einer dazu gleichbedeutenden nicht-tautologischen $\forall$-Klausel offensichtlich, im zweiten Fall erhalten wir eine ebenfalls nicht-tautologische $\forall$-Klausel, weil σ_1 bereits eine nicht-tautologische $\forall$-Klausel ist und nicht x_1 enthalten kann und somit kann $\neg x_1 \vee \sigma_1$ ebenfalls nicht tautologisch sein, wir erhalten also auch wieder $\Phi \underset{Q-Res}{\vdash} \square$.

Wir haben also für alle möglichen Fälle gezeigt, dass eine vollständige Resolutionswiderlegung möglich ist, wenn die Formel falsch ist.

Abschließend gilt es noch zu zeigen, dass sich aus keiner erfüllbaren Formel die leere Klausel beziehungsweise eine nicht-tautologische $\forall$-Klausel ableiten lässt.

Gelte nun also $\Phi \underset{Q-Res}{\vdash} \sigma$ für eine nicht-tautologische $\forall$-Klausel σ.

Sei nun $\Phi = \forall x_1 \exists y_2 \ldots \forall x_k (a_1 \wedge \cdots \wedge a_m)$ die quantifizierte Formel, die wir betrachten. Wenn wir die nicht-tautologische $\forall$-Klausel σ aus der Formel ableiten können, dann gilt $\Phi \approx \forall x_1 \exists y_2 \ldots \forall x_k (a_1 \wedge \cdots \wedge a_m \wedge \sigma)$, was allerdings offensichtlich falsch ist.

$\square$

Bisher haben wir die Resolution quantifizierter Formeln definiert und ihre Widerlegungsvollständigkeit bewiesen. Beides gilt allgemein für alle Formeln aus QKNF*. Nun wollen wir uns eine bestimmte Untergruppe der Resolution genauer ansehen, die nicht mehr für alle Formeln aus QKNF* gilt, sondern nur für spezielle Teilklassen.

Aus der Aussagenlogik ist bekannt, dass man Resolution mit Restriktionen versehen kann. Da eine Resolutionableitung nicht deterministisch ist, man kann theoretisch mit jeden zwei ableitbaren Klauseln beginnen, aber nicht alle Wege führen effizient zum gewünschten Ziel, ist es notwendig gewisse Strategien zu definieren, die die Auswahl der abzuleitenden Klauseln bestimmt. Bekannte Strategien sind zum Beispiel die Stützmengenrestriktion, Lineare Resolution, Davis-Putnam-Resolutin oder Unit-Resolution.

Da die Zweckmäßigkeit von Restriktionen sich analog zur Aussagenlogik auch auf quantifizierte Formeln übertragen lässt, wollen wir uns im Folgenden eine solche ansehen. Hierbei handelt es sich um eine Übertragung der oben bereits erwähnten Unit-Resolution, die im Folgenden so definiert werden wird, dass sie auf quantifizierte Formeln anwendbar ist. Wir nennen diese Restriktion dann Q-Unit-Restriktion.

Definition 3.2 (Q-Unit-Resolution)**.** [4] Die Q-Unit-Resolution ist die Q-Resolution mit

[4] [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 387

der Einschränkung, dass in einer der beteiligten Elternklauseln nur genau ein ∃-Literal vorkommt.

Eine solche Klausel nennen wir ∃-Unit-Klausel. Ist β Resolvente aus α_1 und α_2 für die Q-Unit-Resolution, so schreiben wir α_1,α_2 qunitres β.

Selbstverständlich ist diese Resolution noch immer korrekt, da die Restriktion keine andere Auswirkung hat, als die Auswahl der abzuleitenden Klauseln zu verändern. Ebenso selbstverständlich ist die allerdings in QKNF* nicht widerlegungsvollständig, da unerfüllbare Formeln, die keine ∃-Unit-Klauseln enthalten, oder nicht mehr enthalten, nicht zur leeren Klausel abgeleitet werden können. Die folgende Formel Ψ zum Beispiel liegt in QKNF*, ist unerfüllbar, aber über die Q-Unit-Resolution nicht ableitbar:

$$\Psi = \forall x_1 \exists y_2 \exists y_3 \forall x_4 (x_1 \vee \neg y_2 \vee \neg y_3 \vee \neg z) \wedge (x_4 \vee y_2 \vee y_3 \vee z)$$

Dieses Problem umgehen wir, indem wir eine neue Klasse von Formeln einführen, für die die Q-Unit-Resolution nicht nur korrekt, sondern auch widerlegungsvollständig ist. Diese neue Klasse ist die Klasse QEHORN*, die Klasse der quantifizierten erweiterten Horn-Formeln.

Definition 3.3 (QEHORN*)**.** [5] Eine Formel $\Phi = \forall x_1 \; \exists y_1 \ldots \forall x_k (a_1 \wedge \cdots \wedge a_m)$ mit freien Variablen y_0 ist eine quantifizierte erweiterte Horn-Formel (Extended-Horn-Formel), falls für jede Klausel a_i der ∃-Teil der Klausel eine Horn-Klausel ist, das heißt die Klausel nach Elimination der ∀-Literale in der Klasse der aussagenlogischen Horn-Formeln liegt.

Die Klasse dieser Formeln wird als QEHORN* bezeichnet.

Ein Beispiel für eine solche Formel wäre

$$\Phi = \exists y_1 \forall x_2 \forall x_3 (x_2 \vee y_1 \vee \neg z_1) \wedge (x_2 \vee x_3 \vee z_2)$$

aber nicht

$$\Psi = \forall x_1 \forall x_2 \exists y_3 (x_1 \vee y_3 \vee \neg z_1) \wedge (x_1 \vee x_2 \vee z_2 \vee z_3).$$

(Beweis der Widerlegungsvollständigkeit der Q-Unit-Resolution für die Klasse QEHORN).* [6]

Da es sich bei der Q-Unit-Resolution um eine Sonderform der Unit-Resolution handelt, deren Widerlegungsvollständigkeit für die Aussagenlogik bekannt ist, müssen wir an dieser Stelle nur zeigen, dass Φ widerspruchsvoll $\Rightarrow \Phi \underset{Q-U-Res}{\vdash} \square$ gilt.

Zu Beginn dieses Kapitels haben wir bereits betrachtet, wie man einen Vollständigkeitsbeweis so führen kann, dass man zeigt, dass jede nicht erfüllbare Formel sich zwingend zur leeren Klausel ableiten lässt.

Andersherum kann man auch zeigen, dass jede erfüllbare Formel sich nicht zur leeren Klausel ableiten lässt. Wir zeigen also indirekt

$$(\text{nicht } \Phi \underset{Q-U-Res}{\vdash} \square) \Rightarrow \Phi \text{ ist wahr.}$$

[5] [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 388

[6] vgl.[KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 389f

Wir gehen wieder davon aus, dass unsere Formel Φ geschlossen ist. Außerdem gilt $\Phi \in$QEHORN*. Es gelte: $nicht\Phi \underset{Q-U-Res}{\vdash} \square$.

Sei $\Phi = \forall x_1 \exists y_1 \ldots \forall x_k \exists y_k (a_1 \wedge \cdots \wedge a_m)$ mit $x_1 = x_1 \ldots x_{n_1}$ und $y_1 = y_1 \ldots y_{m_1}$ sowie $x_{i+1} = x_{n_i+1} \ldots x_{n_{i+1}}$ und $y_{i+1} = y_{m_i+1} \ldots y_{m_{i+1}}$ für $1 \leq i \leq k-1$.

Sei nun

$$\mathfrak{U}_\Phi := (X \vee W) \mid W \text{ ist ein } \exists\text{-Literal, } X \text{ eine Disjkt. von } \forall\text{-Literalen, } \Phi \underset{Q-Res}{\vdash} (X \vee W).$$

$\mathfrak{U}_\Phi$ ist die Menge der $\exists$-Unit-Klauseln, die wir aus Φ herleiten können. Über diese Menge kann nun ein Belegungsschema für Φ konstruiert werden, das heißt, wir konstruieren in Abhängigkeit der Quantorenreihenfolge Teilbelegungen $\mathfrak{J}_{a_1,\ldots,a_{n_i}}$, die am Ende zu Gesamtbelegungen erweitert werden.

Für unsere Teilbelegungen gilt, dass sie alle erfüllend sein müssen. Deshalb kann für jede $\exists$-Variable aus Φ nur entweder eine positive oder eine negative $\exists$-Unit-Klausel in $\mathfrak{U}$ enthalten sein, wenn die dazugehörigen $\forall$-Anteile der Klausel nicht tautologisch sind. Gäbe es beide $\exists$-Unit-Klausel in $\mathfrak{U}$, hätten die $\forall$-Anteile keine komplementären Literale und wir erhielten über die Resolution die leere Klausel, was den Voraussetzungen widerspräche.

Außerdem gilt, dass wenn es für $y_{m_{i-1}}$ mit $(1 \leq t \leq m_i - m_{i-1})$ und $(1 \leq i \leq k)$ eine $\exists$-Unit-Klausel $(X \vee y_{m_{i-1}}) \in \mathfrak{U}_\Phi$ oder $(X \vee \neg y_{m_{i-1}}) \in \mathfrak{U}_\Phi$ gibt, dann ist X falsch für eine Zuweisung $(a_1 \ldots a_{n_i}) \in \{0,1\}^{n_1}$ für die $\forall$-Variablen $x_1, \ldots, x_{n_i}$.

Entsprechend definieren wir also

$$\mathfrak{J}_{a_1,\ldots,a_{n_1}}(y_{m_{i-1}+t}) := 1 \text{ bzw. } \mathfrak{J}_{a_1,\ldots,a_{n_1}}(y_{m_{i-1}+t}) := 0.$$

Unsere Gesamtbelegungen $\mathfrak{J}_{a_1,\ldots,a_{n_k}}$ sind, wie bereits erwähnt, jeweils eine Erweiterung der Teilbelegung, also zum Beispiel $\mathfrak{J}_{a_1,\ldots,a_{n_k}}(x_i) = a_i$ oder $\mathfrak{J}_{a_1,\ldots,a_{n_k}}(y_{m_{i-1}+t}) = \mathfrak{J}_{a_1,\ldots,a_{n_i}}(y_{m_{i-1}+t})$.

Mit Induktion über die Anzahl der $\exists$-Literale einer Formel, dass jede Klausel $a_1, \ldots a_m$ mit jeder Gesamtbelegung erfüllt wird.

Hierzu sei φ eine Klausel in Φ mit t-vielen $\exists$-Literalen.

Ist $t = 1$, dann liegt unsere Klausel bereits in $\mathfrak{U}$ und die Aussage ist somit wahr.

Ist $t > 1$, dann gibt es in φ ein negatives $\exists$-Literal $\neg y_{m_{i-1}+t}$, weil Φ in QEHORN* liegt. Dann sei X der $\forall$-Anteil von φ und X' der $\forall$-Anteil aus X, sodass alle Literale aus X' vor $y_{m_{i-1}+t}$ liegen.

Wir nehmen nun an, dass φ für eine mögliche Bewertung $\mathfrak{J}_{a_1,\ldots,a_{n_k}}$ falsch ist.

Für eine Belegung $\mathfrak{J}_{a_1,\ldots,a_{n_k}}$ gilt dann $\mathfrak{J}_{a_1,\ldots,a_{n_k}}(X') = \mathfrak{J}_{a_1,\ldots,a_{n_k}}(\neg y_{m_{i-1}+t}) = 0$. Es gibt also eine $\exists$-Unit-Klausel $X'' \vee y_{m_{i-1}+t} \in \mathfrak{U}$ mit $\mathfrak{J}_{a_1,\ldots,a_{n_k}}(X'') = 0$, sodass X' und X'' keine komplementären Literale enthalten und die Q-Unit-Resolution somit auf $(X'' \vee y_{m_{i-1}+t})$ und φ anwendbar ist.

So erhalten wir eine Klausel φ', mit $t-1$ $\exists$-Literalen, also einem weniger als noch bei φ. Diese Klausel wird ebenfalls für die Belegung $\mathfrak{J}_{a_1,\ldots,a_{n_k}}$ falsch. Sei nun die Formel Φ'

so, dass wir die Formel Φ um die Klausel φ' erweitern. Das Anfügen dieser Resolvente verändert nichts am Wahrheitswert, Φ' ist also dann wahr, wenn Φ wahr ist.

Wenden wir nun die Induktionsvoraussetzung auf Φ' an, erhalten wir die Aussage, dass für Φ' für alle Belegungen der Kern wahr ist. Diese Aussage widerspricht sich mit der Annahme, dass φ falsch ist, die Annahme ist also falsch.

Da wir gezeigt haben, dass Φ unter den genannten Bedingungen immer wahr sein muss, lässt sich mit der Q-Unit-Resolution aus einer erfüllbaren Formel keine leere Klausel ableiten, sie ist also für Formeln in QEHORN* widerlegungsvollständig. □

Abschließend wollen wir noch eine zweite Restriktion betrachten, eine weitere Verfeinerung der bereits vorgestellten Q-Unit-Resolution, die positive Q-Unit-Resolution, auch Q-Pos-Unit-Resolution genannt.

Wie der Name bereits andeutet, handelt es sich hierbei um eine Q-Unit-Resolution, wobei für jeden Resolutionsschritt Unit-Klauseln mit positivem ∃-Literal genutzt werden.

Definition 3.4 (Q-Pos-Unit-Resoultion)**.** [7] Die Q-Pos-Unit-Resolution $\underset{Q-PU-Res}{\vdash}$ ist eine Q-Resolution, mit der weiteren Einschränkung, dass bei jedem Resolutionsschritt eine der beteiligten Elternklauseln in der Form $X \vee y$ sein muss, wobei y ein positives ∃-Literal und X eine Disjunktion von ∀-Literalen ist. Eine solche Klausel nennt man positive ∃-Unit-Klausel.

Beweis. [8] Es genügt erneut zu zeigen, dass mit der Q-Pos-Unit-Resolution für Formeln aus QEHORN* die leere Klausel abgeleitet werden kann, genau dann, wenn die Formel nicht erfüllbar ist. Sei $\Phi \in$QEHORN* also falsch, dann gilt

$$\Phi \underset{Q-PU-Res}{\vdash} \square.$$

Wie auch in den vorherigen Beweisen zur Widerlegungsvollständigkeit gehen wir davon aus, dass Φ geschlossen ist, da freie Variablen mit einem ∃-Quantor gebunden werden können.

Wir definieren nun zu diesem Zweck mehrere Mengen.

Die Menge $\mathbf{Neg}(\Phi)$, die Menge aller Klauseln aus Φ, die ausschließlich negative ∃ Literale enthalten.

Zusätzlich noch die Menge $\mathbf{Rest}(\Phi)$, die wir definieren als $\mathbf{Rest}(\Phi) := \Phi \setminus \mathrm{Neg}(\Phi)$.

Außerdem die Menge $\mathbf{P\text{-}Unit}(\Phi)$, die wie folgt definiert ist:

$$\mathbf{P\text{-}Unit}(\Phi) := \{X \vee Y \mid \mathrm{Rest}(\Phi) \underset{Q-U-Res}{\vdash} (X \vee Y), Y \text{ ist ein pos. } \exists\text{-Literal, } X \text{ besteht aus } \forall\text{-Disjkt.}\}$$

Wir stellen fest, dass jede Klausel $(X \vee Y) \in \mathbf{P\text{-}Unit}(\Phi)$ vorkommt, bereits mit der Q-Pos-Unit-Resolution aus der Menge der Klauseln aus $\mathbf{Rest}(\Phi)$ herleitbar ist, also $\mathbf{Rest}(\Phi) \underset{Q-PU-Res}{\vdash} (X \vee Y)$.

[7] [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 390

[8] vgl.[KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 391f

Außerdem gilt, dass wenn Φ nicht erfüllbar ist, wir mit der einfachen Q-Unit-Resolution die leere Klausel herleiten können, also $\Phi \underset{Q-U-Res}{\vdash} \square$.

Damit dies möglich ist, muss es eine Klausel $\varphi \in \mathbf{Neg}(\Phi)$ und zusätzlich Klauseln $\varphi_1, \ldots \varphi_n \in$**P-Unit**$(\Phi)$ geben, mit denen diese Resolutionsschritte die zur leeren Klausel führen durchgeführt werden können. Also $\varphi, \varphi_1, \ldots \varphi_n \underset{Q-PU-Res}{\vdash} \square$.

Daraus folgt $\Phi \underset{Q-PU-Res}{\vdash} \square$. $\square$

Nachdem wir in diesem Kapitel drei Arten der Q-Resolution kennengelernt haben, wollen wir uns in den folgenden Kapiteln mit bekannten Problemen beschäftigen, zu deren Lösung diese Methoden oft genutzt werden.

4 Erfüllbarkeit Quantifizierter Boole'scher Formeln

Im vorangegangenen Kapitel haben wir uns mit dem Resolutionskalkül befasst, einem Mittel, um zu zeigen, dass eine Formel unerfüllbar ist.

Die Frage, ob eine Formel erfüllbar ist oder nicht, gehört zu den wichtigsten Fragen im Bereich der formalen Logik. Für allgemeine aussagenlogische Formeln gilt das Erfüllbarkeitsproblem als NP-vollständig, es ist also mit bisherigen Mitteln nicht effizient lösbar. Für bestimmte Unterklassen, wie zum Beispiel Horn-Formeln oder Formeln in disjunktiver Normalform, gibt es zwar effiziente Algorithmen, die eine Lösung finden, es gibt aber keine effizienten, und in manchen Fällen auch überhaupt keine, Transformationen von allgemeinen aussagenlogischen Formeln zu Formeln aus diesen Unterklassen.

Analog zur Aussagenlogik gibt es das Erfüllbarkeitsproblem, auch kurz SAT (von engl. satisfiability) genannt, auch für quantifizierte Boole'sche Formeln, häufig abgekürzt als QSAT.

Für eine Klasse K von quantifizierten Boole'schen Formeln ist das Erfüllbarkeitsproblem durch die folgende Menge definiert:[1]

$$\{QSAT = \sigma \in K \mid \text{ es gibt eine Belegung } \mathfrak{J} \text{ für } \sigma \text{ mit } \mathfrak{J}(\sigma) = 1\}$$

In Kapitel 2 haben wir bereits gesehen, dass die Anzahl möglicher Belegungen für Formeln in QBF* geringer ist, als für aussagenlogische Formeln mit gleicher Variablenanzahl. Während bei einer aussagenlogischen Formel die Anzahl aller Variablen ausschlaggebend für die Anzahl möglicher Belegungen ist, hängt die Anzahl möglicher Belegungen bei für quantifizierte Boole'sche Formeln lediglich von der Anzahl freier Variablen ab.

Diese Besonderheit quantifizierter Boole'scher Formeln bringt allerdings einen weit geringeren Vorteil, als man denken mag. Die Anzahl möglicher Belegungen ist zwar geringer und vermindert so den zum Lösen des Problems nötigen Aufwand, allerdings gibt es keine effiziente Möglichkeit den Wahrheitswert gebundener Variablen zu bestimmen.

Wir erinnern uns, eine Formel $\forall x\ \Phi$ ist erfüllbar genau dann, wenn die Formel Φ sowohl für $x = 0$ erfüllbar ist, als auch für $x = 1$.

Für eine Formel $\exists y\ \Phi$ gilt, dass die Formel erfüllbar ist, wenn Φ erfüllbar ist entweder für $y = 1$ oder $y = 0$.

[1] [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 368

Man muss eine zu überprüfende Formel also für jede quantifizierte Variable anhand dieser Belegungskriterien auf ihre Erfüllbarkeit testen. Da die Reihenfolge der Quantoren relevant ist, da Abhängigkeiten zwischen Quantoren entstehen, muss dieser Vorgang für jeden Quantoren einzeln vorgenommen werden, beginnend mit dem, der am äußersten links steht.

Der Aufwand um die Erfüllbarkeit einer geschlossenen Formel zu bestimmen wächst also exponentiell zur Länge des Präfixes.

Wir sehen also, dass sich weder für eine Formel komplett ohne freie Variablen, noch für eine, die nur aus freien Variablen besteht, und damit einer aussagenlogischen Formel entspräche, in Polynomialzeit bestimmen lässt, ob sie erfüllbar ist oder nicht.

Allerdings konnte gezeigt werden, dass das Erfüllbarkeitsproblem für quantifizierte Boole'sche Formeln in polynomiellem Platz gelöst werden kann. Das Problem QSAT liegt somit in PSPACE und es wurde sogar gezeigt, dass es ein für diese Klasse vollständiges Problem ist.

Einen Beweis hierfür findet man zum Beispiel in M. Sipsers *Introduction to the Theory of Computation* in seinem Kapitel über Platzkomplexität. Dieser Beweis wird zwar für geschlossene quantifizierte Formeln geführt, aber wie bereits in dieser Arbeit gezeigt, kann jede offene quantifizierte Formel entsprechend transformiert werden, dass alle freien Variablen einem Quantor zugeordnet wurden. Im folgenden sehen wir einen Algorithmus, der die Erfüllbarkeit einer quantifizierten Boole'schen Formel in polynomiellen Platz löst, er wurde sinngemäß dem oben genannten Buch entnommen.

Beweisidee für die PSPACE-Vollständigkeit von QSAT. [2] Um die PSPACE-Vollständigkeit zu zeigen, muss zum einen gezeigt werden, dass das Problem auch wirklich in PSPACE liegt, und zum anderen, dass es PSPACE-schwer ist, das heißt, jedes Problem in PSPACE lässt sich in polynomieller Zeit auf das zu beweisende Problem reduzieren.

Zuerst wollen wir für QSAT anhand des gezeigten Algorithmus'[3] die Zugehörigkeit zur Klasse PSPACE zeigen.

Wir sehen, dass wir den Algorithmus rekursiv für jede Variable der zu untersuchenden Formel aufrufen, wobei auch immer nur der Wert dieser einen Variable gespeichert wird. Für eine Formel Φ ist die Rekursionstiefe also die Anzahl der Variablen $n \in \Phi$. Unser Platzbedarf ist $n + 1$, wobei die konstante 1 bei wachsendem n immer weniger ins Gewicht fällt, wir können also sagen das Problem QSAT hat einen Platzbedarf von $\mathcal{O}(n)$, also einen linearen Platzbedarf.

Damit ist QSAT in PSPACE.

Die PSPACE-Schwere des Problems wollen wir an dieser Stelle nicht ausführlich beweisen, stattdessen wollen wir nur eine Idee skizzieren, wie ein solcher Beweis aussehen könnte.

[2] vgl. [Sip97] Seite 311ff
[3] vgl. [Sip97] Seite 312

Sei A eine beliebige Sprache, die sich von einer Turingmaschine M in n^k Schritten überprüfen lässt, wobei k eine beliebige Konstante sei.

Diese Sprache A müsste dann in polynomieller Zeit auf QSAT reduziert werden, indem man eine Eingabe w auf einer geschlossenen quantifzierten Boole'schen Formel Φ abbildet. Hierbei gilt, dass Φ genau dann erfüllbar ist, wenn M w akzeptiert. Die Reduktion im Detail kann zum Beispiel bei M. Sipsers nachgelesen werden. □

Input: Eine beliebige geschlossene quantifizierte Formel Φ
Output: True oder False, in Abhängigkeit der Erfüllbarkeit der Formel

```
1  begin
2  |  Π :=das Präfix von Φ
3  |  Ψ := die Formel Φ, ohne das erste Element des Präfix
4  |  α := der Kern von Φ
5  |  if Π = ∅ then
6  |  |  überprüfe ob α wahr oder falsch ist        // Bei einem leeren Präfix
   |  |   besteht die Formel ausschließlich aus Konstanten
7  |  end
8  |  if Φ = ∃y Ψ then
9  |  |  Zweimaliger rekursiver Aufruf des Algorithmus mit Ψ
10 |  |  einmal Ψ[y/0] und einmal mit Ψ[y/1]
11 |  |  if Ψ_{y_0} or Ψ_{y_1} = True then
12 |  |  |  return True
13 |  |  else
14 |  |  |  return False
15 |  |  end
16 |  end
17 |  if Φ = ∀x Ψ then
18 |  |  Zweimaliger rekursiver Aufruf des Algorithmus mit Ψ
19 |  |  einmal Ψ[x/0] und einmal mit Ψ[x/1]
20 |  |  if Ψ_{x_0} and Ψ_{x_1} = True then
21 |  |  |  return True
22 |  |  else
23 |  |  |  return False
24 |  |  end
25 |  end
26 end
```

Algorithmus 4: QSAT

Nachdem wir nun gesehen haben, wie das Erfüllbarkeitsproblem für quantifizierte Boole'sche Formeln definiert ist und sich allgemein verhält, wollen wir im Folgenden betrachten, wie sich das Erfüllbarkeitsproblem für die Unterklassen QKNF* und QHORN* verhält und ob es Unterschiede zu der allgemeineren Form gibt.

4.1 Erfüllbarkeit für QKNF*

Im Kapitel zur Resolution haben wir bereits den Resolutionskalkül als Möglichkeit kennengelernt, die Erfüllbarkeit einer Formel in konjunktiver Normalform indirekt zu bestimmen.

Lässt sich die Formel erfolgreich durch Resolution widerlegen, so ist sie nicht erfüllbar. Im Umkehrschluss können wir also auch sagen, dass eine Formel, die sich nicht durch Resolution widerlegen lässt, erfüllbar sein muss.

Auch wenn dieses Verfahren ebenfalls keine Lösung in Polynomialzeit liefert, basieren die meisten modernen Lösungswege für das reguläre Erfüllbarkeitsproblem auf der Resolution. Der vermutlich bekannteste Algorithmus zur Entscheidung über die Unerfüllbarkeit einer Formel in konjunktiver Normalform ist das Davis-Putnam-Verfahren, benannt nach Martin Davis und Hillary Putnam.

Modifiziert man dieses Verfahren so, dass quantifizierte Variablen berücksichtigt werden, kann es auch für quantifizierte Formeln angewandt werden. Dies gilt auch für die erweiterte Form dieses Verfahrens, den Davis-Putnam-Logemann-Loveland-Algorithmus, kurz DPLL genannt, der auch heute noch von vielen Forschern, sowohl für aussagenlogische als auch quantifzierte Formeln, weiter verfeinert und erweitert wird, mit dem Ziel, für möglichst viele Formeln einen effizienten Weg zur Lösung von Erfüllbarkeitsproblemen zu finden.

Ein Beispiel für einen sogenannten SAT-Solver aus dem Bereich der quantifizierten Boole'schen Formeln, der auf Basis von DPLL arbeitet, wäre das Verfahren QUAFFLE aus dem Jahr 2002. Bei diesem Verfahren wird der DPLL-Algorithmus zusätzlich um konfliktbasiertes Lernen erweitert. Näheres hierzu kann in dem von L. Zhang und S. Malik veröffentlichtem Paper *Conflict Driven Learning in a Quantified Boolean Satisfiability Solver*[4] nachgelesen werden.

An dieser Stelle muss allerdings angemerkt werden, dass es durchaus Unterklassen in QKNF* gibt, für die das Erfüllbarkeitsproblem sehr wohl in polynomieller Zeit lösbar ist. Im Folgenden wollen wir uns eine Unterklasse ansehen, für die eine Lösung nicht nur in polynomieller Zeit, sondern sogar in linearer Zeit vorhanden ist, die Klasse Q-2-KNF*.

Bevor wir uns allerdings mit der Frage der Erfüllbarkeit beschäftigen, wollen wir kurz betrachten, was genau Q-2-KNF* genau heißt. Hierbei handelt es sich um Formeln, die in QKNF* liegen, sich aber dadurch auszeichnen, dass jede Klausel der Formel maximal die Länge 2 hat, also in jeder Klausel höchstens zwei Variablen auftreten, die durch eine Disjunktion miteinander verbunden sind.

Formeln die in Q-2-KNF* liegen sind für dieses Kapitel nicht nur interessant, weil sie sich zeiteffizient lösen lassen, sondern auch, weil der Weg zu dieser Lösung eine

[4] [ZM02]

weitere mögliche Herangehensweise abseits von Resolutionswiderlegung und einfachem Brute-Force zeigt. Um das Erfüllbarkeitsproblem für Q-2-KNF* zu lösen, bedient man sich nämlich der Graphentheorie, von der einige Elemente hier kurz erläutert werden sollen.

Ein Graph G besteht aus einer Menge von Knoten V, die durch Kanten E miteinander verbunden sind. Eine Kante ist dabei definiert durch $\{v_1, v_2\} \in V$, die Knoten, die sie miteinander verbindet.

Einen Graphen bezeichnen wir als gerichtet, wenn die durch eine Kante verbundenen Knotenpaare geordnet sind und nicht beliebig vertauscht werden können. In einem gerichteten Graphen ist die Kante (v_1, v_2) nicht identisch zur Kante (v_2, v_1). Bei einem ungerichteten Graphen würden beide Tupel dieselbe Kante beschreiben. Eine gerichtete Kante kann zum besseren Verständnis auch als $v_1 \to v_2$ geschrieben werden.

Als starke Zusammenhangkomponente S eines gerichteten Graphen bezeichnen wir einen Teilgraphen, sodass innerhalb dieses Teilgraphen jeder Knoten von jedem anderen Knoten aus erreicht werden kann.

Um nun die Erfüllbarkeit einer logischen Formel durch einen Graphen zeigen zu können, müssen wir den sogenannten assoziierten Graphen definieren.

Definition 4.1.1 (Assoziierter Graph)**.** [5]

Für eine Klausel $\Phi_i = (L_{i,1} \vee \cdots \vee L_{i,k_i})$ mit $k_1 > 1$ ist der assoziierte Graph $G(\Phi_i)$ ein gerichteter Graph, bestehend aus der Knotenmenge $V(\Phi_i) := \{L_{i,1}, \ldots, L_{i,k_i}, \neg L_{i,1}, \ldots, \neg L_{i,k_i}\}$, wobei doppelte Negationssymbole gestrichen werden, und der Kantenmenge $E(\Phi) := \{\neg L_{i,r} \to L_{i,s} \mid 1 \leq r \neq s \leq k_i\}$, wobei der Pfeil $\to$ die Kantenrichtung angibt.

Für jede Unit-Klausel L bilden wir die Kante $(\neg L \to L)$.

Der zu $\Phi = \Phi_1, \ldots, \Phi_n$ assoziierte Graph $G(\Phi)$ ist die Vereinigung der assoziierten Graphen $G(\Phi_1), \ldots, G(\Phi_n)$ der einzelnen Klauseln von Φ:

$$G(\Phi) := \left(\bigcup_{i=1}^{n} V(\Phi_i), \bigcup_{i=1}^{n} E(\Phi_i) \right)$$

Der Einfachheit halber werden wir als Namen für die Knoten unseres assoziierten Graphen mit das jeweilige Literal nutzen, welches der Knoten abbildet.

Können wir von einem Knoten L_1 einen Knoten L_2 erreichen, so sagen wir, dass es einen Pfad zwischen diesen beiden Knoten gibt. Ein solcher Pfad hat die Länge n, wobei n hier die Anzahl der besuchten Knoten zwischen L_1 und L_2 ist. Einen solchen Pfad beschreiben wir, genau wie eine gerichtete Kante, mit $L_1 \to L_2$.

Zusätzlich werden wir uns im Folgenden erneut nur mit den assoziierten Graphen geschlossener Formeln beschäftigen, da es für die Erfüllbarkeit der Formel keinen Unterschied macht, ob eine Variable frei oder $\exists$-gebunden ist.

Außerdem werden wir uns im Folgenden eine Eigenschaft von assoziierten Graphen zu Nutze machen, die hier kurz gezeigt werden soll. Es gilt, dass für jede Q-Resolvente

[5] [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 78

$(L_1 \vee L_2)$ aus Φ auch Pfade zwischen den entsprechenden Knoten, also $(\neg L_1 \to L_2)$ und $(\neg L_2 \to L_1)$ in $G(\Phi)$ gibt.

Außerdem kann man, da nur die Erfüllbarkeit einer Formel betrachtet wird, auch umgekehrt sagen, dass zu jedem Pfad auch eine entsprechende Q-Resolvente existiert. Selbst wenn eine solche Resolvente $(\neg L_1 \vee L_2)$ nicht in der ursprünglichen Formel existiert, kann sie hinzugefügt werden, ohne dass der Wahrheitswert der Formel sich ändert, was wir im folgenden Lemma festhalten.

Lemma 4.1.1. [6] Sei $\Phi = \forall x_1 \exists y_2 \ldots \forall x_{k-1} \exists y_k \ (\phi_1 \wedge \cdots \wedge \phi_m) \in$Q-2-KNF* und sei $G(\Phi)$ der assoziierte Graph zu dieser Formel.

1. Seien $Z_1 \to x$ und $x \to Z_2$ zwei Kanten in $G(\Phi)$ mit den Literalen Z_1, Z_2 und einer $\forall$-Variable x. Dann gilt

   $$\Phi \text{ ist wahr} \quad \Leftrightarrow \quad \forall x_1 \exists y_2 \ldots \forall x_{k-1} \exists y_k (\phi_1 \wedge \cdots \wedge \phi_m \wedge (\neg Z_1 \vee Z_2)) \text{ ist wahr.}$$

2. Sei $L_1 \to L_2$ ein Weg in $G(\Phi)$. Dann gilt

   $$\Phi \text{ ist wahr} \quad \Leftrightarrow \quad \forall x_1 \exists y_2 \ldots \forall x_{k-1} \exists y_k (\phi_1 \wedge \cdots \wedge \phi_m \wedge (\neg L_1 \vee L_2)) \text{ ist wahr.}$$

Beweis. Der erste Teil des Lemmas ist einfach zu zeigen. Die zu den Kanten $Z_1 \to x$ und $x \to Z_2$ entsprechenden Klauseln sind $(\neg Z_1 \vee x)$ und $(\neg x \vee Z_2)$.

Fügt man einer Formel, in der diese beiden Klauseln enthalten sind die Klausel $(\neg Z_1 \vee Z_2)$ hinzu, ändert sich, wie man sehen kann, die Belegung nicht.

$$\mathfrak{J}((\neg Z_1 \vee x) \wedge (\neg x \vee Z_2)) = 1 \Leftrightarrow \mathfrak{J}((\neg Z_1 \vee x) \wedge (\neg x \vee Z_2) \wedge (\neg Z_1 \vee Z_2)) = 1$$

Nachdem so der erste Teil des Lemmas gezeigt wurde, ergibt sich der zweite Teil fast von selbst.

Die in der ersten Hälfte des Lemmas beschriebenen Kanten bilden einen Pfad der Länge 1 und somit unseren Induktionsanfang.

Wir haben gezeigt, dass die Aussage für einen Pfad der Länge 1 wahr ist. Für Pfade einer Länge > 1 können wir für jeden Schritt die notwendigen Klauseln ergänzen und danach wieder reduzieren und erhalten so für eine erfüllbare Formel auch immer einen Pfad im assoziierten Graphen. □

Das Lemma 4.1.1 zeigt, dass es zwischen der Erfüllbarkeit einer Formel und den Pfaden innerhalb ihres assoziierten Graphen einen Zusammenhang gibt. Deshalb können wir nun mithilfe dieses Lemmas folgende Aussagen über die Erfüllbarkeit einer Formel anhand ihres assoziierten Graphen treffen:

Lemma 4.1.2. [7] Sei $\Phi \in$ Q-2-KNF* und $G(\Phi)$ der assoziierte Graph. Dann ist Φ nicht wahr genau dann, wenn eine der folgenden Bedingungen gilt:

[6] [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 400
[7] [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 401

1. Es gibt eine $\exists$-Variable y und es gibt eine starke Zusammenhangkomponente S von $G(\Phi)$ mit $y, \neg y \in S$.

2. Es gibt eine $\forall$-Variable x und eine $\exists$-Variable y mit $y < x$, für die es eine starke Zusammenhangkomponente S von $G(\Phi)$ gibt, in der ein Literal über x sowie ein Literal über y enthalten sind.

3. Es gibt zwei $\forall$-Literale L_1 und L_2 über unterschiedliche Variablen oder über die gleiche Variable, die sowohl negiert als auch nicht negiert auftritt, für die ein Pfad von L_1 zu L_2 in $G(\Phi)$ existiert.

Anstatt also eine Resolutionswiderlegung direkt auf die Formel anzuwenden, überprüfen wir stattdessen den Zusammenhang des assoziierten Graphen. So erhalten wir effizientere Lösungsmöglichkeiten, da die Frage zum Beispiel durch Erweiterung der Frage, ob es einen Pfad zwischen zwei Knoten eines Graphen gibt, auch bekannt als PATH, gelöst werden kann. Weitere Möglichkeiten zum Finden von Zusammenhangskomponenten wären beispielsweise der Algorithmus von Tarjan zur Bestimmung starker Zusammenhangskomponenten oder der Kosajaru-Sharir-Algorithmus.

Nun gilt es noch zu zeigen, dass die oben aufgestellten Bedingungen auch wirklich gelten. Der folgende Beweis ist sinngemäß aus *Aussagenlogik: Deduktion und Algorithmen* von Kleine Büning und Lettmann übernommen.

Beweis (Lemma 4.1.2). [8] Im Folgenden wollen wir einen zweiteiligen Beweis führen. Zum einen zeigen wir, dass wenn eine Formel Φ falsch ist, auch immer mindestens eine der drei oben genannten Bedingungen gilt. Zum anderen zeigen wir, dass wenn eine der drei oben genannten Bedingungen gilt, eine Formel Φ auch immer falsch ist. Womit wir zu dem Ergebnis kommen, dass eine Formel Φ genau dann falsch ist, wenn mindestens eine der oben genannten Bedingungen erfüllt ist.

Zuerst zeigen wir, dass wenn Φ falsch ist, mindestens eine der oben genannten Bedingungen erfüllt ist. Dies tun wir über mögliche Resolutionsschritte, da wir wissen, dass wenn Φ falsch ist, es immer auch eine Resolutionswiderlegung $\Phi \underset{Q-Res}{\vdash} \square$ gibt.

Außerdem lassen sich Resolutionswiderlegungen für Q-2-KNF*-Formeln immer zu linearen Ableitungen umformen, das heißt, jeder Resolutionsschritt nutzt die im vorherigen Schritt geformte Resolvente. Hat eine Formel zudem nur eine einzige Ausgangsklausel mit einem $\forall$-Literal, kann diese in einer linearen Resolution zuletzt genutzt werden.

So können wir, mit Hilfe der Induktion über die Anzahl der Resolutionsschritte, zu folgenden Aussagen für zwei $\exists$-Literale L_1 und L_2 kommen:

1. $\Phi \underset{Q-Res}{\vdash} (L_1 \vee L_2)$ $\Rightarrow$ Es gibt Pfade $\neg L_1 \to L_2$ und $\neg L_2 \to L_1$ in $G(\Phi)$.
2. $\Phi \underset{Q-Res}{\vdash} (L_1 \vee X)$ $\Rightarrow$ Es gibt Pfade $\neg L_1 \to X$ und $\neg X \to L_1$ in $G(\Phi)$.
3. $\Phi \underset{Q-Res}{\vdash} (L_1)$ $\Rightarrow$ Es gibt einen Pfade $\neg L_1 \to L_1$ in $G(\Phi)$.

[8] vgl.[KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 401ff

Zu Beginn haben wir bereits festgelegt, dass Φ falsch ist und der letzte Resolutionsschritt somit immer zur leeren Klausel führt. Daher können folgende Fälle auftreten: Da Φ falsch ist, gibt es eine Herleitung der leeren Klausel, also können die folgenden Fälle auftreten.

1. $\Phi \underset{Q-Res}{\vdash} y, \neg y \overset{1}{\underset{Q-Res}{\vdash}} \square$

   Dann gibt es in $G(\Phi)$ die Pfade $\neg y \to y$ und $y \to \neg y$. Also liegen y und $\neg y$ in derselben starken Zusammenhangskomponente von $G(\Phi)$ und Bedingung 1 ist erfüllt.

2. $\Phi \underset{Q-Res}{\vdash} (y \vee X), \neg y \overset{1}{\underset{Q-Res}{\vdash}} \square$

   Dann gibt es in $G(\Phi)$ die Pfade $\neg X \to y$, $y \to \neg y$ und $\neg y \to X$ und Bedingung 3 ist erfüllt.

3. $\Phi \underset{Q-Res}{\vdash} (y \vee X_1), (\neg y \vee X_2) \overset{1}{\underset{Q-Res}{\vdash}} (X_1 \vee X_2) \overset{1}{\underset{Q-Res}{\vdash}} \square$

   Dann gibt es in $G(\Phi)$ die Pfade $\neg X_1 \to y$, $\neg y \to X_1$ und $\neg X_2 \to \neg y$, $y \to X_2$ und $\neg X_1 \to X_2$ und $\neg X_2 \to X_1$. Falls $X_1 \neq X_2$ gilt, dann erfüllt dieser Fall Bedingung 3.

   Gilt $X_1 = \neg X_2$ gibt es zwei Möglichkeiten, die beide dazu führen, dass Bedingung 2 erfüllt wird. Entweder es gilt $(y < x_1)$, wobei x_1 eine Variable ist, die in X_1 ist, und Bedingung 2 ist offensichtlich erfüllt. Ansonsten wissen wir, dass im Kern von Φ eine Klausel mit einer $\exists$-Variablen y' auftauchen muss, sodass $(y' < x_1)$. Dann gäbe es einen der beiden möglichen Resolutionsschritte $\Phi \underset{Q-Res}{\vdash} (y' \vee X_1), (\neg y' \vee X_2)$ beziehungsweise $\Phi \underset{Q-Res}{\vdash} (\neg y' \vee X_1), (y' \vee X_2)$ und Bedingung 2 wäre wieder erfüllt. Wäre dies nicht der Fall, dann gäbe es eine $\forall$-gebundene Variable, über die eine $\exists$-gebundene Variable regiert, die nicht über Resolution auflösbar wäre, sodass keine vollständige Resolutionswiderlegung möglich wäre, welche zu Beginn des Beweises als gegeben vorausgesetzt wurde.

Für die weiteren möglichen Fälle, die lediglich Vertauschungen der $\exists$-Literale darstellen, lassen diese Begründungen sich analog führen.

Da wir nun gezeigt haben, dass, wenn Φ falsch ist, eine der oben genannten Bedingungen gültig ist, zeigen wir im nächsten Teil, dass Φ auch falsch sein muss, wenn eine der Bedingungen gilt.

Gelte Bedingung 1:

Seien $y, \neg y \in S$, dann gibt es Pfade $y \to \neg y$ und $\neg y \to y$ in S. Die Formel Φ ist, nach Lemma 4.1.1 genau dann wahr, wenn $\Phi' := \forall x_1 \exists y_2 \ldots \exists y_k (a_1 \wedge \cdots \wedge a_m \wedge (y \vee \neg y))$ wahr ist, da diese Klausel die entsprechenden Pfade repräsentiert. Da die Formel Φ' falsch ist, muss Φ auch falsch sein.

Gelte Bedingung 2:

Sei $y < x$ und $x, y \in S$, dann gibt es Pfade $x \to y$ und $y \to x$ in S. Daraus folgt Φ ist genau dann wahr, wenn $\Phi' = \forall x_1 \ldots \exists y_k(a_1 \wedge \cdots \wedge a_m \wedge (y \vee \neg x) \wedge (\neg y \vee x))$ wahr ist, da Φ, nach Lemma 4.1.1 die beiden letztgenannten Klauseln enthält, da sie Kanten des assoziierten Graphen sind. Die letzten beiden Klauseln führen aber dazu, dass die Formel falsch wird, da $\exists y \forall x (y \vee \neg x) \wedge (\neg y \vee x)$ nicht erfüllbar ist. Also ist Φ falsch.

Gelte Bedingung 3:

Seien L_1 und L_2 zwei $\forall$-Literale und $L_1 \to L_2$ ein Pfad in $G(\Phi)$. Nach Lemma 4.1.1 kann $(\neg L_1 \vee L_2)$ zum Kern von Φ ergänzt werden, also $\Phi = \forall x_1 \ldots \exists y_k(a_1 \wedge \cdots \wedge a_m \wedge (\neg L_1 \vee L_2))$ ohne die Erfüllbarkeit zu verändern.

Die hieraus resultierende Formel ist aber nicht erfüllbar, wenn L_1 und L_2 unterschiedliche Variablen beschreiben, oder Variablen mit unterschiedlichen Vorzeichen, wenn also zum Beispiel $L_1 = x_i$ und $L_2 = x_j$ ist, da die Klausel $\neg x_i \vee x_j$ nicht erfüllbar ist, oder wenn $L_1 = x_i$ und $L_2 = \neg x_i$, da die hieraus resultierende Klausel $\neg x_i \vee \neg x_i$ zu $\neg x_i$ zusammengefasst werden kann und offensichtlich unerfüllbar ist.

Also muss Φ unerfüllbar sein, da die vorgenommene Modifikation an der Erfüllbarkeit der Formel nichts verändert.

Ist also eine der drei Bedingungen erfüllt, ist eine Formel Φ falsch. □

Damit ist gezeigt, dass man über den assoziierten Graph einer Formel aus Q-2-KNF* ihre Erfüllbarkeit bestimmen kann und das Erfüllbarkeitsproblem für diese Formelklasse somit in polynomieller, sogar linearer, Zeit lösbar ist, da es für die Analyse gerichteter Graphen Algorithmen mit entsprechend geringem Zeitaufwand gibt und die Erstellung eines assoziierten Graphen zu einer Formel auch nur linear viel Zeit in Abhängigkeit der Länge der Formel benötigt.

4.2 Erfüllbarkeit für QHORN*

Zum Schluss wollen wir nun noch das Erfüllbarkeitsproblem für die Klasse QHORN* betrachten.

Wir haben in Kapitel 3 die Q-Resolution, die Q-Unit-Resolution und die Q-Pos-Unit-Resolution kennengelernt und für alle drei gezeigt, dass sie für die Klasse QEHORN* widerlegungsvollständig ist. Wir erinnern uns, dass QEHORN* die Klasse der erweiterten Hornformeln ist, das heißt, dass alle Formeln die in QHORN* sind, auch in QEHORN* vorkommen.

Dies bedeutet, dass, auch wenn wir die Widerlegungsvollständigkeit der oben genannten Resolutionsmethoden nicht ausdrücklich für QHORN* gezeigt haben, wir dennoch davon ausgehen können, dass sie gilt, da wir sie für QEHORN* gezeigt haben.

Wir wissen also, dass die Q-Unit-Resolution und auch die Q-Pos-Unit-Resolution für quantifizierte Horn-Formeln widerlegungsvollständig sind. Diesen Umstand werden wir uns im Folgenden zunutze machen, denn einer der effizientesten Weg die Erfüllbarkeit

einer quantifizierten Hornformel zu überprüfen ist mit Hilfe der positiven Q-Unit-Resolution.

Wir betrachten erneut lediglich geschlossene Formeln und schließen solche, die nicht geschlossen sind, vor der Betrachtung ab. Außerdem überprüfen wir unsere Formel vorher auf tautologische Klauseln und Klauseln, die nur aus $\forall$-Literalen und damit unerfüllbar sind. Ist eines von beidem enthalten, sortieren wir die Formel aus, da wir anhand dieser Klauseln bereits ein Urteil über ihre Erfüllbarkeit fällen können und weitere Betrachtungen nicht notwendig sind. Die Formel Φ hat also die Form $\Phi = \exists y_0 \forall x_1 \exists y_1 \dots \forall x_{k-1} \exists y_k (a_1 \wedge \dots \wedge a_m)$.

Wir teilen nun alle Klauseln unserer Formel eine der drei folgenden Gruppen zu. Eine solche Einteilung erleichtert nicht nur das Verständnis, sondern auch die Überprüfung, da für bestimmte Anzeichen einer nicht erfüllbaren Formel nicht mehr jedes Mal die komplette Formel durchlaufen werden muss, es reicht sich auf eine bestimmte Menge an Klauseln zu konzentrieren. Da wir quantifizierte Horn-Formeln betrachten, wissen wir zudem, dass jede Klausel maximal ein positives Literal enthalten kann, womit wir die folgenden Gruppen erhalten.

Die erste Gruppe bezeichnen wir als P_Φ, die Menge aller Klauseln, in denen ein positives $\exists$-Literal auftaucht.

Die zweite Gruppe, wir nennen sie N_Φ^+, ist die Menge der Klauseln, die ein positives $\forall$-Literal enthalten.

Die letzte Gruppe ist folglich die Menge der Klauseln, die kein positives Literal enthalten, die Gruppe N_Φ.

Wir können nun über diese Mengen die Erfüllbarkeit von Φ überprüfen, denn Φ ist genau dann nicht erfüllbar, wenn es eine Klausel φ gibt, die in der Vereinigung von N_Φ^+ und N_Φ auftritt, also $\varphi \in N_\Phi \cup N_\Phi^+$, sodass die daraus resultierende Formel $\Phi = \exists y_0 \forall x_1 \exists y_1 \dots \forall x_{k-1} \exists y_k (P_\Phi \wedge \varphi)$ falsch ist.

Diese Problematik lässt sich mit Hilfe der Q-Pos-Unit-Resolution lösen, wobei es zwei Möglichkeiten gibt.

Zum einen kann die Klausel φ in der Menge N_Φ liegen. In diesem Fall können wir ohne weitere Vorkehrungen die Q-Pos-Unit-Resolution anwenden, nachdem wir die Formel dahingehend vereinfacht haben, alle $\forall$-Literale zu streichen. Da $N_\Phi \cup P_\Phi$ keine positiven $\forall$-Literale enthält, kann es nach keinem Resolutionsschritt zu einer tautologischen $\forall$-Klausel kommen, da dafür ein positives und ein negatives Vorkommen des gleichen $\forall$-Literals notwendig wäre, die $\forall$-Literale dieser Formel haben also auf ihre Erfüllbarkeit keinen Einfluss. Erhalten wir nach dem letzten Resolutionsschritt die leere Klausel wissen wir, Φ ist nicht erfüllbar. Erhalten wir nicht die leere Klausel wissen wir umgekehrt auch, dass Φ erfüllbar ist.

Die andere Möglichkeit ist, dass die Klausel φ in N_Φ^+ liegt. In diesem Fall könnten im Verlauf der Resolution tautologische Klauseln auftreten. Sei unser positives $\forall$-Literal x_i, dann kann auch nur x_i zum Bilden einer tautologischen Klausel genutzt werden, indem wir als Resolvente $x_i \vee \neg x_i$ erhielten. Alle anderen $\forall$-Literale die in $P_\Phi \wedge \varphi$ auftauchen

können also gestrichen werden.

Seien nun y_j die $\exists$-Variablen unserer Formel, dann können wir für jedes y_j zwei verschiedene, mögliche positive Unit-Klauseln bilden, nämlich y_j und $y_j \vee x_i$. Wir wissen aus Kapitel 3, dass unsere Formel $\Phi i = \exists y_0 \forall x_1 \exists y_1 \ldots \forall x_{k-1} \exists y_k (P_\Phi \wedge \varphi)$ in diesem Fall also genau dann falsch ist, wenn wir mit Hilfe der Q-Pos-Unit-Resolution aus N_Φ^+ Unit-Klauseln der Form y_j herleiten können, sodass es eine positive Unit-Klausel für jede in φ vorkommende $\exists$-Variable gibt, denn dann lassen sich entweder die leere Klausel oder eine nicht-tautologische $\forall$-Klausel x_i beziehungsweise $\neg x_i$ herleiten.

Diese gesuchten Unit-Klauseln können wir für alle $y_j < x_i$ herleiten, ohne, dass wir x_i beachten müssen, da der dazugehörige Quantor in diesen Fällen nicht regierend ist. Sobald dies geschehen ist, können wir die Klauseln für $y_j > x_i$ mit den bereits vorhandenen Klauseln und den Klauseln aus P_Φ die keine $\forall$-Literale enthalten berechnen. Danach müssen wie die Menge dieser herleitbaren positiven Unit-Klauseln nur noch mit den $\exists$-Variablen von φ vergleichen und können so eine Entscheidung über die Erfüllbarkeit von Φ treffen.

Würde man diese Vorgänge algorithmisch umsetzen, so erhielte man eine Laufzeit von $\mathcal{O}(rn)$, wobei r die Anzahl der positiven $\forall$-Literale ist und n die Länge der Formel.

Dies ergibt sich daraus, dass zwar die meisten benötigten Vorgänge in linearer Zeit zu bearbeiten sind, wir aber für den Fall dass φ in N_Φ^+ liegt, eine Schleife bilden müssen, die so oft aufgerufen wird, wie es zu überprüfende Klauseln mit positivem $\forall$-Literal gibt.

Es gibt alternativ die Möglichkeit, die zu untersuchende quantifizierte Horn-Formel zum Beispiel mit dem in Kapitel 2 vorgestellten Algorithmus von Bubeck und Kleine Büning in eine rein $\exists$-quantifizierte Formel zu übertragen.

In Kapitel 2 haben wir bereits gesehen, dass eine solche Transformation in maximal quadratischer Zeit möglich ist und eine Formel in ebenfalls maximal quadratischer Länge im Verhältnis zur Ausgangsformel generiert.

Eine solche Formel $\Phi \in \exists\text{HORN}$ kann in Bezug auf ihre Erfüllbarkeit behandelt werden wie eine aussagenlogische Horn-Formel, für die es effiziente Algorithmen zur Klärung der Erfüllbarkeit gibt, zum Beispiel den Markierungsalgorithmus.

Dieses Verfahren hat zwar ebenfalls eine Laufzeit von $\mathcal{O}(rn)$, ist aber einfacher praktisch umzusetzen als das zuvor beschriebene.

5 Konsequenz und Äquivalenz

Abschließend wollen wir noch zwei weitere Probleme betrachten. Dem Problem der Folgerbarkeit, auch Konsequenz genannt, die Frage, ob eine Formel semantisch aus einer anderen Formel folgt, und der Äquivalenz, die Frage ob zwei Formeln unter denselben Belegungen wahr oder falsch werden. Wir haben diese beiden Begriffe bereits in Kapitel 2 eingeführt, wollen uns nun aber näher mit ihnen beschäftigen.

Man kann diese Probleme wie folgt für eine Klasse K quantifizierter Formeln als Mengen zusammenfassen:[1]

$$\text{Das Konsequenzproblem: } conQBF = \{(\Phi, \Psi) \in K \times K \mid \Phi \models \Psi\}$$

und

$$\text{Das Äquivalenzproblem: } equiQBF = \{(\Phi, \Psi) \in K \times K \mid \Phi \approx \Psi\}$$

Logische Äquivalenz kann auch über Folgerbarkeit dargestellt werden, denn zwei Formeln Φ und Ψ sind äquivalent genau dann, wenn $\Phi \models \Psi$ und $\Psi \models \Phi$.

Diese alternative Darstellung der Äquivalenz macht es offensichtlich, dass wir, wenn wir einen effizienten Weg haben, zu bestimmen, ob eine Formel aus einer anderen folgt, wir über diesen Weg auch effizient bestimmen können, ob zwei Formeln äquivalent sind.

Außerdem können wir beobachten, dass es einen Zusammenhang zwischen Konsequenz und Erfüllbarkeit gibt.

Lemma 5.0.1. [2] Seien Φ und Ψ zwei Formeln, dann gilt $\Phi \models \Psi$, genau dann, wenn $(\Phi \wedge \neg\Psi)$ widerspruchsvoll, also nicht erfüllbar ist.

Beweis. [3] Gelte $\Phi \models \Psi$, dann gilt für alle Belegungen, dass die Belegung $\mathfrak{J}(\Phi) = 1$ auch $\mathfrak{J}(\Psi) = 1$ impliziert.

Entsprechend gilt auch für alle Belegungen $\mathfrak{J}(\neg\Phi \vee \Psi) = 1$. Ist $\mathfrak{J}(\Phi) = 0$, ist diese Aussage offensichtlich wahr, für $\mathfrak{J}(\Phi) = 1$ ist sie ebenfalls wahr, weil daraus $\mathfrak{J}(\Psi) = 1$ folgt.

Daraus folgt $\mathfrak{J}(\neg(\neg\Phi \vee \Psi)) = 0$, was nach den de Morganschen Gesetzen auch als $\mathfrak{J}(\Phi \wedge \neg\Psi) = 0$ geschrieben werden kann .

Daraus folgt

$$(\Phi \wedge \neg\Psi) \text{ ist widerspruchsvoll } \Leftrightarrow \Phi \models \Psi.$$

□

[1] [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 368
[2] [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 9
[3] vgl.[KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 9

Dieser Zusammenhang lässt bereits erahnen, dass es für Formeln aus QBF* und QKNF* keine effiziente Lösung für die Frage nach der Folgerung gibt. Wir wissen aus dem vorherigen Kapitel, dass es keine effiziente Möglichkeit gibt, die Erfüllbarkeit für alle Formeln dieser Klassen zu bestimmen. Da die Erfüllbarkeit einer Formel für die Folgerbarkeit, und ebenso die Äquivalenz, eine große Rolle spielt, sieht man schnell, dass ohne eine effiziente Lösung für das Erfüllbarkeitsproblem, das Beantworten der Frage nach der Folgerbarkeit recht umständlich wird.

Fände man dennoch einen effizienten Weg, so könnte man diesen nutzen, um auch die Erfüllbarkeit einer Formel effizient zu bestimmen, was wir im Fall für QBF* und QKNF* im vorangegangenen Kapitel bereits ausgeschlossen haben.

Allerdings haben wir im vorherigen Kapitel auch gesehen, dass das Erfüllbarkeitsproblem für die Klassen QHORN* und Q-2-KNF* effizient lösbar ist. Im Folgenden wollen wir also untersuchen, ob es für diese Klassen effiziente Lösungen des Konsequenzproblems, und daraus folgend auch des Äquivalenzproblems, gibt.

Zu diesem Zweck wollen wir an dieser Stelle zwei neue Begriffe einführen, die $\mathfrak{R}$-Folgerung und die $\mathfrak{R}$-Äquivalenz.

Definition 5.1 ($\mathfrak{R}$-Folgerung und $\mathfrak{R}$-Äquivalenz)**.** [4] Sei $\mathfrak{R}$ eine nicht-leere Menge von Variablen.

Dann heißen zwei Formeln α und β $\mathfrak{R}$-äquivalent, geschrieben $\alpha \overset{\mathfrak{R}}{\approx} \beta$, wenn für jede Klausel π mit Variablen aus $\mathfrak{R}$ gilt:

$$\alpha \models \pi \quad \Leftrightarrow \quad \beta \models \pi.$$

Außerdem ist die Formel β eine $\mathfrak{R}$-Folgerung von α, geschrieben $\alpha \overset{\mathfrak{R}}{\models} \beta$, genau dann, wenn für alle Klauseln π mit Variablen aus $\mathfrak{R}$ gilt:

$$\beta \models \pi \quad \Rightarrow \quad \alpha \models \pi.$$

Eine Formel β ist also eine $\mathfrak{R}$-Folgerung von α, wenn für jede Klausel, die nur Variablen aus $\mathfrak{R}$ enthält, folgt, dass sie aus α folgerbar ist, wenn sie aus β folgerbar ist.

Wenn die Mengen der aus α und β folgerbaren Klauseln, die nur Variablen aus $\mathfrak{R}$ enthalten, identisch sind, dann sprechen wir also von $\mathfrak{R}$-Äquivalenz.

Man sieht direkt, dass es sich hierbei um weniger strikte Folgerungs- und Äquivalenzbegriffe handelt. Da wir aber mehr vergleichen als nur die reine Erfüllbarkeit zweier Formeln, haben wir mit dem Begriff der $\mathfrak{R}$-Äquivalenz eine Möglichkeit einen Zusammenhang zwischen zwei Formeln zu beschreiben, der stärker ist als die Erfüllbarkeitsäquivalenz.

Betrachten wir α und β jetzt nicht als aussagenlogische Formeln, sondern als Kerne von quantifizierten Formeln, erhalten wir die folgenden Zusammenhänge:

[4] [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 267

Anmerkung. Die Formeln α_1 und α_2 seien Formeln in konjunktiver Normalform und die Menge $\mathfrak{R}$ sei definiert als $\mathfrak{R} \subseteq var(\alpha_1) \cup var(\alpha_2)$.

Mit $\{y_1, \ldots, y_n\} := (var(\alpha_1) \cup var(\alpha_2)) \setminus \mathfrak{R}$ gilt dann:

$$\alpha_1 \overset{\mathfrak{R}}{\models} \alpha_2 \Leftrightarrow \exists y_1 \ldots \exists y_n \alpha_1 \models \exists y_1 \ldots \exists y_n \alpha_2$$

und

$$\alpha_1 \overset{\mathfrak{R}}{\approx} \alpha_2 \Leftrightarrow \exists y_1 \ldots \exists y_n \alpha_1 \approx \exists y_1 \ldots \exists y_n \alpha_2$$

Da wir bereits wissen, dass es möglich ist alle Formeln aus QHORN* zu Formeln in ∃HORN umzuwandeln, ist dieser Zusammenhang für uns äußerst nützlich. Wir stellen fest, dass wir die Folgerbarkeit und Äquivalenz für quantifizierte Horn-Formeln also direkt aus der $\mathfrak{R}$-Folgerbarkeit bzw -Äquivalenz ihres Kerns ableiten können.

Trotz dieses Zusammenhangs gibt es allerdings keinen effizienten Weg diese Fragen für Formeln aus QHORN* zu beantworten, da sowohl die $\mathfrak{R}$-Folgerbarkeit als auch die $\mathfrak{R}$-Äquivalenz für aussagenlogische Horn-Formeln coNP-vollständig ist, wie Kleine Büning und Lettmann im fünften Kapitel ihres Buches *Aussagenlogik: Deduktion und Algorithmen* zeigen.

Satz. Die $\mathfrak{R}$-Äquivalenz ist für Horn-Formeln coNP-vollständig.

Wir wollen an dieser Stelle den dort erbrachten Beweis lediglich kurz skizzieren.

Beweissskizze. [5] Da eine Klausel π mit $var(\pi) \subseteq \mathfrak{R}$ auf einem nicht-deterministischen Weg bestimmt werden kann und die Folgerbarkeit dieser Klausel aus zwei Formeln α und β in Linearzeit entschieden werden kann, sehen wir direkt, dass das Problem der $\mathfrak{R}$-Äquivalenz in coNP liegt.

Zusätzlich ist es coNP-schwer, was über eine Reduktion auf ein Komplement eines NP-vollständigen Problems gezeigt werden kann.

Das in diesem Beweis gewählte Problem ist das Komplement von Monotone-3-SAT, der Frage ob $(\alpha \wedge \beta)$ erfüllbar ist, wobei beide genannten Formeln in 3-KNF liegen und α nur positive Literale enthält, β nur negative. Das Komplement stellt entsprechend die Frage danach, ob $(\alpha \wedge \beta)$ nicht erfüllbar ist.

Für die Reduktion wird jetzt jeder Formel $(\alpha \wedge \beta) \in$ 3-KNF zwei aussagenlogische Horn-Formeln σ_1 und σ_2 zugewiesen, sowie eine Menge $\mathfrak{R} \subseteq var(\sigma_1 \cup \sigma_2)$, sodass die folgende Aussage gilt:

$$\sigma_1 \overset{\mathfrak{R}}{\approx} \sigma_2 \quad \Leftrightarrow \quad \sigma_2 \overset{\mathfrak{R}}{\models} \sigma_1 \quad \Leftrightarrow \quad \alpha \wedge \beta \notin \text{SAT}$$

[5] vgl.[KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 268

Diese Aussage gilt es zu zeigen, um die coNP-schwere der $\mathfrak{R}$-Äquivalenz für aussagenlogische Horn-Formeln zu zeigen. Daraus ergibt sich dann ebenfalls die coNP-Vollständigkeit der $\mathfrak{R}$-Folgerbarkeit. Ebenso können wir damit aufgrund der obigen Anmerkung die coNP-Vollständigkeit der beiden Probleme für quantifizierte Horn-Formeln zeigen. □

Wir sehen also, dass es für quantifizierte Horn-Formeln keine effiziente Lösung des Konsequenzproblems oder des Äquivalenzproblems gibt, obwohl wir die Erfüllbarkeit von quantifizierten Horn-Formeln effizient bestimmen können.

Es bleibt nun noch die Frage, ob dies auch für Formeln in Q-2-KNF* gilt, oder ob es zumindest für diese von uns betrachtete Formelklasse möglich ist, eine effiziente Lösung zu finden.

Um diese Frage einfacher beantworten zu können, wollen wir zuerst eine Methode vorstellen, um die zu untersuchenden Formeln zu vereinfachen.

Wir haben bereits im Kapitel über quantifizierte Horn-Formeln gesehen, dass es eine Möglichkeit gibt, jede quantifizierte Horn-Formel zu einer rein $\exists$-quantifizierten Horn-Formel zu transformieren. Diese Möglichkeit besteht für erfüllbare Formeln in Q-2-KNF* ebenfalls.

Im Folgenden sehen wir einen Algorithmus, $\exists$-2-KNF*, der in höchstens quadratischer Zeit eine erfüllbare Formel $\Phi \in$ Q-2-KNF* zu einer äquivalenten Formel Φ' umwandelt, die keine $\forall$-quantifizierten Variablen mehr enthält.[6]

Dieser Algorithmus arbeitet in $n \cdot z$ Schritten, wobei n die Länge der Formel ist, und z die Anzahl der freien Variablen. Wir haben also $\mathcal{O}(zn)$, eine höchstens quadratische Laufzeit.

Da wir die Erfüllbarkeit von Formeln in Q-2-KNF* in linearer Zeit bestimmen können, ist der hierfür nötige Zeitaufwand zu vernachlässigen und wir können ohne zusätzlichen Zeitaufwand davon ausgehen, dass sowohl Ein- als auch Ausgabe erfüllbare Formeln in Q-2-KNF* sind. Wir können also aus Formeln in Q-2-KNF* zeiteffizient die $\forall$-quantifizierten Variablen entfernen und erhalten eine Formel in Q-2-KNF*, die nur $\exists$-quantifzierte und freie Variablen enthält.

Für die Frage nach der Folgerbarkeit zweier Formeln $\Phi_1, \Phi_2 \in$Q-2-KNF* können wir dank dieser Transformation die Kerne der Formeln wie aussagenlogische Formeln behandeln und das Problem über die $\mathfrak{R}$-Folgerbarkeit lösen. Da die $\mathfrak{R}$-Folgerbarkeit für Formeln in 2-KNF in quadratischer Zeit lösbar ist und die Transformation einer beliebigen Formel in Q-2-KNF* ebenfalls höchstens quadratische Zeit benötigt, ist es also möglich die Frage der Folgerbarkeit einer Formel in Q-2-KNF* aus einer anderen Formel aus dieser Klasse, und damit auch die Frage nach der Äquivalenz zweier solcher Formeln, zeiteffizient zu beantworten.

Satz. Die $\mathfrak{R}$-Folgerbarkeit ist für aussagenlogische Formeln in 2-KNF in quadratischer Zeit lösbar.

[6] vgl. [KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 406

Input: Eine beliebige, erfüllbare Formel $\Phi \in$ Q-2-KNF*
Output: Eine zu Φ äquivalente Formel $\Psi \in$ Q-2-KNF* ohne $\forall$-quantifizierten Variablen

```
1  begin
2      freeVar :=die Menge der freien Variablen z_1, ... z_r von Φ
3      φ := Menge der Klauseln φ_1, ..., φ_m von Φ
4      𝔘 := ∅
5      for i = 1; i <= r; i++ do
6          if SAT-Q-2-KNF(Φ ∧ ¬z_i)= false then
7              𝔘 = 𝔘 ∪ {z_i}
8          else
9              𝔘 = 𝔘 ∪ {¬z_i}
10         end
11     end
12     for i = 1; i <= m; i++ do
13         if φ_i ∩ 𝔘 ≠ ∅ then
14             entferne φ_i aus Φ
15         else
16             X := ∀-Literale der Klausel
17             Y := ∃-Literale der Klausel
18             Z :=Literale über freie Variablen der Klausel
19             if φ_i = (Z ∨ X) then
20                 φ_i = Z
21             end
22             if (φ_i = (Y ∨ X)) and var(Y) < var(X) then
23                 φ_i = Y
24             end
25             if z ∈ 𝔘 and ¬z ∈ φ_i then
26                 entferne alle Literale über ¬z aus φ_i
27             end
28             if beliebige ∀-Variable x ∈ φ_i then
29                 entferne φ_i
30             end
31         end
32     end
33     if Φ = ∅ then              // Φ ist hier die resultierende Formel
34         Φ := z_1 ∨ ¬z_1
35     end
36     return (Φ ∪ 𝔘)
37 end
```

Algorithmus 5: $\exists$-2-KNF*

Beweis. [7]

Zuerst halten wir fest, dass das Erfüllbarkeitsproblem für Formeln in 2-KNF in linearer Zeit lösbar ist, das heißt wir können uns darauf beschränken nur erfüllbare Formeln zu betrachten, ohne, dass dies den Zeitaufwand beeinflusst.

Außerdem gilt für zwei Formeln α und β, dass $\mathfrak{R} = \{Z_1, \ldots, Z_n\} \subseteq var(\alpha) \cup var(\beta)$.

Um zu zeigen, dass wir die $\mathfrak{R}$-Folgerbarkeit in maximal quadratischer Zeit bestimmen können, betrachten wir eine zu $\alpha \overset{\mathfrak{R}}{\models} \beta$ äquivalente Aussage, aus der wir weitere Aussagen ableiten können, nämlich $\beta \models L_1 \vee L_2 \Rightarrow \alpha \models L_1 \vee L_2$ für alle Literale $L_1, L_2 \in lit(\mathfrak{R})$.

Sei nun für eine Formel Δ und ein Literal L die Menge $\mathfrak{U}$ wie folgt definiert:

$$\mathfrak{U}(\Delta, L_1) := \{L_2 \mid L_2 \in lit(\Delta), \Delta \wedge L_1 \models L_2\}$$

Dann können wir die obige Aussage auch äquivalent als

$$\mathfrak{U}(\beta, L) \ \cap \ lit(\mathfrak{R}) \subseteq \mathfrak{U}(\alpha, L) \cap lit(\mathfrak{R})$$

darstellen. Das heißt, dass wenn β $\mathfrak{R}$-folgerbar aus α ist, die Menge der aus β herleitbaren Unitklauseln, die ein Literal über eine Variable in $\mathfrak{R}$ enthalten, eine Teilmenge der aus α herleitbaren Unitklauseln, die ebenfalls in $\mathfrak{R}$ vorkommen, ist.

Um das zu zeigen, liegt es nahe, zum einen die Menge der jeweiligen Unitklauseln und danach die Menge der übrigen Klauseln mit exakt zwei Literalen, im Folgenden 2-Klauseln, zu betrachten.

Die Menge der Unitklauseln, die sich direkt aus α beziehungsweise β herleiten lassen, lässt sich in quadratischer Zeit bestimmen, indem wir den assoziierten Graphen mit einem geeigneten Algorithmus auf folgerbare Unitklauseln hin überprüfen.

Wenn $Unit(\beta) \not\subseteq Unit(\alpha)$ gilt, dann können wir ohne weitere Betrachtungen sofort schließen, dass $\alpha \overset{\mathfrak{R}}{\not\models} \beta$ gilt.

Andernfalls müssen wir zusätzlich die 2-Klauseln aus α und β betrachten. Hierbei gehen wir davon aus, dass aus beiden Formeln die gleichen Unit-Klauseln über $\mathfrak{R}$ herleitbar sind.

Dann betrachten wir im Folgenden die Formeln α' und β', die sich ergeben, wenn wir α, beziehungsweise β, um die jeweiligen Unit-Klauseln kürzen, was ebenfalls nicht mehr als quadratische Zeit in Anspruch nimmt.

Für diese Formeln α' und β' bilden wir nun die assoziierten Graphen $G(\alpha')$ und $G(\beta')$ und überprüfen für jedes Literal $L \in \mathfrak{R}$, ob die folgerbaren Literale aus α' eine Obermenge der folgerbaren Literale aus β' bilden, wir also die 2-Klauseln bestimmen, die $\neg L$ enthalten.

Dafür testen wir einfach die Erreichbarkeit der jeweiligen Knoten von L aus innerhalb der assoziierten Graphen $G(\alpha')$ und $G(\beta')$. Anschließend überprüfen wir, ob die Menge von L aus erreichbaren Knoten aus $G(\beta')$, geschrieben $reach(G(\beta'), L)$, in

[7] vgl.[KL94] Aussagenlogik: Deduktion und Algorithmen, Seite 279f

$reach(G(\alpha'), L)$, also der Menge der von L aus erreichbaren Knoten in $G(\alpha')$, enthalten ist.

Wenn für alle Literale $L \in (lit(\mathfrak{R}) \setminus Unit(\alpha))$ gilt

$$(reach(G(\beta'), L) \cap lit(\mathfrak{R})) \setminus Unit(\alpha) \subseteq reach(G(\alpha'), L) \cap lit(\mathfrak{R}),$$

dann gilt genau dann auch $\alpha \overset{\mathfrak{R}}{\models} \beta$.

Da jede dieser Operationen in maximal quadratischer Zeit durchführbar ist, erhalten wir also eine Lösung für die $\mathfrak{R}$-Folgerbarkeit für Formeln in 2-KNF in maximal quadratischer Zeit. Gleiches gilt für $\mathfrak{R}$-Äquivalenz, da wir in diesem Fall die Folgerbarkeit einfach in beide Richtungen überprüfen.

□

Wir haben nun gezeigt, dass es eine Möglichkeit gibt, die Folgerbarkeit, und somit auch die Äquivalenz, für Formeln aus Q-2-KNF* über die $\mathfrak{R}$-Folgerbarkeit des Kerns zu bestimmen und, dass dies in höchstens quadratischer Zeit möglich ist.

Es ist also möglich, die Folgerbarkeit und Äquivalenz für Formeln in Q-2-KNF* zeiteffizient zu bestimmen, und Q-2-KNF* stellt somit ein Beispiel für eine Unterklasse von Formeln in QBF* dar, für die diese Probleme effizient lösbar sind dar.

6 Fazit

Wir haben uns im Verlauf dieser Arbeit damit auseinandergesetzt, wie quantifizierte Boole'sche Formeln definiert sind, was sie auszeichnet und worin sie sich von der klassischen Aussagenlogik unterscheiden. So haben wir festgestellt, dass es für jede aussagenlogische Formel die Möglichkeit gibt, sie in eine äquivalente quantifizierte Formel umzuwandeln und, dass es ebenfalls möglich ist, jede quantifizierte Formel in eine aussagenlogische Formel umzuwandeln. Wir haben die Pränexnormalform kennengelernt, die quantifizierte Formeln in ein Präfix und einen Kern teilt und konnten so schnell erkennen, dass der Kern einer quantifizierten Formel große Ähnlichkeiten zu einer aussagenlogischen Formel aufweist und, dass eine quantifizierte Formel ohne Präfix, also nur aus freien Variablen bestehend, identisch zu einer aussagenlogischen Formel ist.

Als Beispiele für mögliche Unterklassen der quantifizierten Formeln haben wir uns exemplarisch mit den quantifizierten Entsprechungen der Horn-Formeln und den Formeln in konjunktiver Normalform beschäftigt. Dabei haben wir nicht nur ihre Definition betrachtet, sondern in späteren Kapiteln auch, wie die Probleme der Erfüllbarkeit und Folgerbarkeit für diese Unterklassen im Vergleich zu quantifizierten Boole'schen Formeln zu lösen sind.

Hierbei haben wir auch gesehen, dass es nicht für jedes Problem eine effiziente Lösung gibt und dass es für Probleme, die in der Aussagenlogik nicht effizient lösbar sind, auch für ihre quantifizierten Entsprechungen keine effiziente Lösung gibt. So konnten wir zwar die Widerlegungsvollständigkeit des Resolutionskalküls für quantifizierte Boole'sche Formeln zeigen, aber, ähnlich wie in der Aussagenlogik, reicht dies noch nicht, um einen effizienten Lösungsweg des Erfüllbarkeitsproblems für alle quantifizierte Formeln zu haben.

Obwohl quantifizierte Boole'sche Formeln uns also keine Möglichkeit bieten, logische Probleme zeiteffizienter zu lösen, so stellen sie doch in vielen Fällen eine Platzersparnis dar und bieten so eine kompaktere Alternative zur Aussagenlogik.

Es ist selbstverständlich, dass eine Arbeit wie diese nur an der Oberfläche eines derart umfangreichen Themas kratzen kann. Man kann sich weitaus tiefgehender mit quantifizierten Boole'schen Formeln beschäftigen, als es hier geschehen ist, schließlich ist diese Arbeit auch lediglich als Einführung in dieses Thema zu verstehen.

Einige tiefere Einblicke in die Thematik bietet zum Beispiel das bereits in dieser Arbeit erwähnte Paper *Models and quantifier elimination for quantified Horn formulas* von Bubeck und Kleine Büning, das sich mit der Eliminierung von $\forall$-quantifizierten Variablen in quantifizierten Horn-Formeln beschäftigt. Ebenfalls von diesen Autoren ist *Bounded Universal Expansion for Preprocessing QBF* erschienen, ein Paper, in dem es

darum geht, quantifizierte Boole'sche Formeln und die Arbeit mit ihnen effizienter zu gestalten.

Resolution and Expressiveness of Subclasses of Quantified Boolean Formulas and Circuits von Bubeck, Kleine Büning und Zhao beschäftigt sich mit der hier bereits eingeführten Q-Unit-Resolution und führt eine neue Erweiterung, die b-Unit-Resolution ein.

Zudem behandeln Kleine Büning und Zhao in *On Models for Quantified Boolean Formulas* die Möglichkeiten des Model Checkings im Bereich der quantifizierten Boole'schen Formeln.

Dies ist alles nur ein kleiner Ausblick, der aber deutlich zeigt, dass die Thematik der quantifizierten Boole'schen Formeln sehr umfangreich ist und viele Möglichkeiten zur Weiterbildung bietet, die aber unmöglich alle in dieser Arbeit hätten Platz finden können.

Abschließen lässt sich also sagen, dass es sich bei quantifizierten Boole'schen Formeln um ein spannendes Themenfeld handelt und quantifizierte Boole'sche Formeln in der praktischen Anwendung oftmals eine konzise und teils intuitivere Alternative zu der bekannten Aussagenlogik bieten.

Literaturverzeichnis

[BK08] Uwe Bubeck and Hans Kleine Büning. Models and quantifier elimination for quantified horn formulas. *Discret. Appl. Math.*, 156(10):1606–1622, 2008.

[KL94] Hans Kleine Büning and Theodor Lettmann. *Aussagenlogik - Deduktion und Algorithmen*. Leitfäden und Monographien der Informatik. Teubner, 1994.

[Sip97] Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.

[VK19] Heribert Vollmer and Thorsten Kluge. Logik und formale systeme. Skript zur Veranstaltung Logik und Formale Systeme, Juli 2019.

[ZM02] Lintao Zhang and Sharad Malik. Conflict driven learning in a quantified boolean satisfiability solver. In Lawrence T. Pileggi and Andreas Kuehlmann, editors, *Proceedings of the 2002 IEEE/ACM International Conference on Computer-aided Design, ICCAD 2002, San Jose, California, USA, November 10-14, 2002*, pages 442–449. ACM / IEEE Computer Society, 2002.