Logik SS13: Zusammenfassung

Florian Pelz

11. Juni 2014

Aussagenlogik

Allgemeines

Aussagenlogische Formeln haben die Form $((\neg p \wedge q) \to r) \leftrightarrow a \vee b$.

Wir schreiben V für die Menge der aussagenlogischen Variablen (wie z.B. p oder q oben). Formeln, die nur aus einer einzigen aussagenlogischen Variablen bestehen, werden in der Aussagenlogik **atomar** genannt.

Ein **Literal** ist eine atomare aussagenlogische Formel („**positives Literal**“) oder die Negation einer atomaren aussagenlogischen Formel („**negatives Literal**“).

Wenn nicht anders angegeben haben wir als **Junctoren** $\neg, \wedge, \vee, \to, \leftrightarrow$. Wenn wir als Kurzschreibweise Klammern weglassen, dann ist dies auch die Reihenfolge, in der die Junctoren ausgewertet werden, z.b. bindet in $\neg p \vee q$ das $\neg$ stärker als das $\vee$, d.h. $\neg p \vee q \equiv (\neg p) \vee q$. Die Implication $A \to B$ wird dabei wie $\neg A \vee B$ interpretiert, d.h. $A \to B$ gilt in der Aussagenlogik auch als wahr, wenn A falsch ist. $A \leftrightarrow B$ wird als $(A \to B) \wedge (B \to A)$ interpretiert.

Eine **Aussageform** ist eine aussagenlogische Formel. Wir schreiben F für die Menge der aussagenlogischen Formeln.

Die **Tiefe** einer Formel ist die maximale Anzahl an Verschachtelungen in der Formel, z.B. hat p eine Tiefe $t(p) = 0$, während $t(p \wedge ((q \vee (q \wedge p)) \wedge p)) = 4$ ist.

Structurelle Induction ist eine Beweismethode für den Beweis von Aussagen über inductiv definierte Objecte. Als Inductionsanfang beweisen wir die Aussage für die nicht recursiven Teile der Definition, beim Inductionsschritt für die recursiven unter der Voraussetzung.

Es kann sinnvoll sein, Kurzschreibweisen für in dem Inductionsbeweis benötigte Dinge einzuführen, z.B. $\#_($(A)$ für die Anzahl öffnender Klammern in einer Formel A.

Der **Eindeutigkeitssatz** (Folien S. 17) besagt in Princip, dass nur genau gleich aufgebaute Formeln **syntactisch äquivalent** ($\equiv$) sind.

Eine **Bewertung** einer aussagenlogischen Formel ist eine Function $\varphi : F \to \mathbb{B}$, die die atomaren Formeln beliebig auf wahr (1) oder falsch (0) abbildet, und die zusammengesetzten Formeln dann entsprechend (nach den Regeln auf den Folien S.19) abbilden muss. Bewertungen können auch durch Wahrheitstafeln dargestellt werden. Wir sagen, φ erfüllt eine Formel A, falls $\varphi(A) = 1$.

Eine **Belegung** $\psi : V \to \mathbb{B}$ bildet die Variablen so ab, wie es eine entsprechende Bewertung mit den entsprechenden atomaren Aussagen tut. Für die endliche Menge der Variablen in einer (stets endlich langen) Formel als V gibt es nur endlich viele (genau $2^{|V|}$) verschiedene Belegungen und damit auch Bewertungen, die von diesen bzw. den Aussagen darüber aus auf $\mathbb{B}$ abbilden.

Eine **Tautologie** ist eine Aussage, die von allen möglichen Bewertungen erfüllt wird. Eine Aussage ist **allgemeingültig**, wenn sie eine Tautologie ist. Eine Aussage ist **erfüllbar**, wenn es eine sie erfüllende Bewertung gibt. Eine Aussage ist **widerspruchsvoll**, wenn sie nicht erfüllbar ist. TAUT ist die Menge der Tautologien. SAT ist die Menge der erfüllbaren Formeln. Eine Menge Σ von Formeln ist erfüllbar, wenn alle Formeln in der Menge von einer Belegung gleichzeitig erfüllt werden. Eine Formel A ist die **logische Folgerung** bzw. **semantische Folgerung** einer Menge Σ von Formeln, wenn jede Belegung, die Σ erfüllt, auch A erfüllt (wir schreiben $\Sigma \models A$, für ein endliches $\Sigma = \{A_1, \ldots, A_n\}$ auch $A_1, \ldots, A_n \models A$, allgemeiner schreiben wir auch z.B. etwas wie $\Sigma, A \models B$ für die logische Folgerbarkeit von B aus der Vereinigung von Σ und $\{A\}$). Wir schreiben Folg(Σ) für die Menge der logischen Folgerungen von Σ.

Eine Menge ist **entscheidbar**, wenn algorithmisch in endlicher Zeit feststellbar ist, ob ein gegebenes Object in der Menge liegt oder nicht. Eine Menge ist **semi-entscheidbar**, wenn algorithmisch in endlicher Zeit für alle enthaltenen Objecte feststellbar ist, dass sie enthalten sind, während der Algorithmus für nicht enthaltene Objecte nicht terminieren muss. Dies ist genau bei den **recursiv aufzählbaren** Mengen der Fall, d.h. bei den Mengen, für die es einen Algorithmus gibt, der alle enthaltenen Objecte irgendwann (aber in endlicher Zeit) ausgibt. Ist das Complement einer Menge M semi-entscheidbar, spricht man jedoch nicht von Semi-Entscheidbarkeit von M (manche nennen die Menge M dann aber negativ semi-entscheidbar).

Die hervorgehobenen Folgerungen auf den Folien S. 28 sollte man kennen: Eine Aussage A ist allgemeingültig, wenn ihre Negation $\neg A$ widerspruchsvoll ist. $\Sigma \models A$ gilt genau dann, wenn $\Sigma \cup \{\neg A\}$ widerspruchsvoll ist. Es ist für eine endliche Formelmenge entscheidbar, ob sie erfüllbar ist. TAUT und SAT sind entscheidbar.

Der **Modus Ponens** ist die Schlussregel, dass $\{A, A \to B\} \models B$.

Die **Contraposition** einer Aussage $A \to B$ ist $\neg B \to \neg A$; sie gilt genau dann, wenn die ursprüngliche Aussage gilt, ist also **logisch äquivalent**.

Der **Compactheitssatz** besagt, dass eine Menge von aussagenlogischen Formeln genau dann erfüllbar ist, wenn es jede endliche Teilmenge ist. Die Contraposition dazu ist, dass eine Menge von aussagenlogischen Formeln genau dann unerfüllbar ist, wenn eine

endliche Teilmenge existiert, die unerfüllbar ist. Das heißt dann auch, dass $\Sigma \cup \{\neg A\}$ unerfüllbar ist, wenn eine endliche Teilmenge unerfüllbar ist, und damit, dass $\Sigma \models A$ gilt, wenn $\Sigma_0 \models A$ für eine endliche Teilmenge $\Sigma_0 \subseteq \Sigma$ gilt.

Die Eigenschaften der Junctoren bezüglich logischer Äquivalenz auf den Folien S. 32 sollte man können: Negation ist eine Involution, d.h. $\neg\neg A \models\!\!\!\models A$. Conjunction ist idempotent, d.h. $A \wedge A \models\!\!\!\models A$. Conjunction ist commutativ. Conjunction ist associativ. Disjunction ist associativ. Für Conjunction und Disjunction gelten die Distributivgesetze in beide Richtungen. Für Conjunction und Disjunction gelten die Gesetze von **De Morgan**.

Logische Äquivalenz ist, was wir eine **Congruenz** nennen, d.h. ersetzt man eine Teilformel einer Formel durch eine logisch äquivalente Teilformel, dann erhält man eine zur ganzen Formel logisch äquivalente neue Formel.

Eine Menge OP von Operatoren, d.h. von Junctoren (sowohl den grundlegenden Junctoren der Aussagenlogik als auch anderen, durch Wahrheitstafeln definierbaren Junctoren wie „false“), heißt **vollständige Operatorenmenge**, wenn es zu jeder aussagenlogischen Formel eine logisch äquivalente Formel in der Menge F(OP) der Formeln nur mit diesen Operatoren als Junctoren gibt. Wir kennen die folgenden vollständigen Operatorenmengen: $\{\neg, \to\}, \{\neg, \vee\}, \{\neg, \wedge\}, \{\neg, \vee, \wedge\}, \{\text{false}, \to\}, \{\bar{\wedge}\}$. Um zu zeigen, dass eine Operatorenmenge vollständig ist, reicht es, eine bekannte vollständige Operatorenmenge damit auszudrücken.

Eine **boolesche Function** ist eine Function, die boolesche Werte (d.h. 1 und 0) auf einen einzigen booleschen Wert abbildet. Jede aussagenlogische Formel kann als boolesche Function angesehen werden, diese nimmt die Argumente der Function als Belegung der Variablen und hat als Functionswert den Wert, den die der Belegung entsprechende Bewertung φ angewandt auf die Formel hat.

Normalformen

Normalformen für aussagenlogische Formeln, die wir kennen sollten, sind **CNF bzw. KNF** (conjunctive Normalform), d.h. eine Conjunction von Disjunctionen von Literalen, und **DNF** (disjunctive Normalform), d.h. eine Disjunction von Conjunctionen von Literalen. Die **canonische CNF oder DNF** einer Formel ist die zur Formel logisch äquivalente CNF bzw. DNF mit minimaler Länge, also z.B. ohne doppelt vorkommende Literale in einem Teilterm.

Eine Disjunction von Literalen nennen wir **Clausel**. Eine Clausel nennen wir **positive Clausel**, wenn sie nur positive Literale beinhaltet; eine Clausel nur mit negativen Literalen nennen wir **negative Clausel**. Eine Clausel mit höchstens einem positiven Literal nennen wir **Horn-Clausel**. Eine Clausel aus höchstens k Literalen nennen wir k **-Clausel**. Eine 1-Clausel nennen wir **Unit-Clausel**. Eine CNF aus k-Clauseln nennen wir k **-CNF** (bzw. k **-KNF**). Das entsprechende Erfüllbarkeitsproblem für Formeln in k-CNF nennen wir dementsprechend k**-SAT**. Clauseln können wir auch als Menge der

enthaltenen Literale schreiben, z.b. $\{p, \neg q, \neg r\}$ für $p \vee \neg q \vee \neg r$. Eine CNF läßt sich dann als Menge von Clauseln darstellen.

Zudem gibt es die **Negationsnormalform** (**NNF**): Dabei verwenden wir nur die Operatoren $\neg$, $\wedge$ und $\vee$ und fordern, dass $\neg$ nur vor atomaren Formeln steht. Wir können eine Formel, die nur aus diesen Operatoren besteht, in Negationsnormalform bringen, indem wir mit den Gesetzen von De Morgan jedes $\neg$ schrittweise weiter nach innen „ziehen“.

CNF, DNF und NNF von einer Formel sind logisch äquivalent zur ursprünglichen Formel.

Als **duale Formel** $d(A)$ zu einer Formel A in $F(\{\neg, \wedge, \vee\})$ bezeichnen wir die Formel, die A entspricht, nachdem alle $\wedge$ und $\vee$ darin vertauscht wurden. Zum Beispiel ist $d(\neg p \vee (q \wedge p)) = \neg p \wedge (q \vee p)$. Für jede A erfüllende Bewertung ist die Bewertung zur genau umgekehrten Belegung eine $d(A)$ nicht erfüllende Bewertung (analog zu den De Morganschen Gesetzen). Damit sind die Tautologien genau die Formeln, deren duale Formeln unerfüllbar sind. Erfüllbare Formeln sind genau die, deren duale Formeln keine Tautologien sind.

Deductive Systeme

Ein **deductives System** oder **Calcül** ist ein Alphabet mit einer entscheidbaren Menge F von Formeln darüber zusammen mit einer entscheidbaren Menge von Axiomen aus dieser Formelmenge F und einer entscheidbaren Menge von Schlussregeln der Form $\frac{A_1,\ldots,A_n}{A}$ mit $A_1, \ldots, A_n, A \in F$. Wir schreiben $\mathcal{F}(Ax, R)$ für das deductive System mit der Axiomenmenge Ax und der Regelmenge R und einer implicit sinnvoll angenommenen Menge von Formeln. Die Menge der Axiome kann durch eine Auflistung der zugehörigen Axiome und Axiomenschemata gegeben werden. Ein **Axiomenschema** ist eine Formel wie ein Axiom, aber mit Variablen, die durch beliebige Formeln des deductiven Systems ersetzt werden dürfen. Somit beschreibt ein Axiomenschema eine Menge von Axiomen.

Ein **Theorem** ist eine Formel, die zur Menge der zulässigen Formeln eines deductiven Systems gehört und für die es einen Beweis im deductiven System gibt, über den sie **hergeleitet** wird, d.h. in dem sie vorkommt. Ein **Beweis** ist eine Folge von Aussagen, die entweder Axiome sind oder Formeln, die über Schlussregeln aus vorherigen Folgengliedern folgen. Wir schreiben $\vdash_{\mathcal{F}} A$ falls die Formel A aus dem deductiven System $\mathcal{F}$ herleitbar ist, d.h. falls es einen Beweis dafür gibt. Die Menge der Beweise ist damit entscheidbar. Dann ist A ein Theorem. Wir schreiben $T(\mathcal{F})$ für die Menge der Theoreme von $\mathcal{F}$.

Ein **abgekürzter Beweis** ist wie ein Beweis, nur dass Beweisglieder auch als gegeben angenommen werden dürfen, d.h. Voraussetzungen bzw. Annahmen bzw. Hypothesen sind, und dann nicht aus den darüber stehenden Beweisgliedern direct über die Schlussregeln des Calcüls folgen müssen. Wir schreiben $\Sigma \vdash_{\mathcal{F}} A$ falls es einen abgekürzten Beweis

gibt, bei dem alle Voraussetzungen in der Formelmenge Σ liegen. Wir sagen, A **folgt syntactisch** aus Σ. Dabei ist $\text{Folg}_{\mathcal{F}}(\Sigma)$ die Menge solcher Formeln A für das jeweilige Σ.

Für $\vdash_{\mathcal{F}}$ schreiben wir auch nur $\vdash$, wenn es klar ist, auf welches deductive System $\mathcal{F}$ wir uns beziehen.

Ist $\Sigma \subseteq \Gamma$, dann ist $\text{Folg}(\Sigma) \subseteq \text{Folg}(\Gamma)$. Folgt etwas aus einer Teilmenge von Σ, dann folgt es auch aus Σ, Beweise lassen sich somit zusammensetzen. Insbesondere ist die Menge der Theoreme des deductiven Systems immer in Σ enthalten.

Es gilt durch die Definition der Ableitung in einem deductiven System, dass Beweise immer endlich lang sind, und damit, dass $\Sigma \vdash A$ genau dann gilt, wenn es eine endliche Teilmenge $\Sigma_0 \subseteq \Sigma$ gibt mit $\Sigma_0 \vdash A$ (analog zum Compactheitssatz).

Eine Menge Σ heißt **consistent** bezüglich eines deductiven Systems, wenn für keine daraus syntactisch folgende Aussage auch deren Negation $\neg A$ syntactisch folgt. Consistenz bei deductiven Systemen zur syntactischen Folgerung ist analog zu der Erfüllbarkeit bei der Semantik. Eine inconsistente Menge hat somit auch eine inconsistente endliche Teilmenge, bzw. eine Menge ist genau dann consistent, wenn jede endliche Teilmenge consistent ist (analog zum Compactheitssatz).

Die semantische Version des **Deductionstheorems** besagt, dass $\Sigma, A \models B$ genau dann gilt, wenn $\Sigma \models (A \to B)$ gilt. Dementsprechend ist zum Beispiel $A \models B$ äquivalent zu $\models A \to B$ und $A \models\!\models B$ äquivalent zu $\models A \leftrightarrow B$. Die syntactische Version macht die analoge Aussage für $\vdash_{\mathcal{F}}$ statt $\models$, gilt aber nicht für jedes deductive System $\mathcal{F}$.

Ein deductives System $\mathcal{F}$ heißt **correct**, wenn $(\vdash_{\mathcal{F}} A) \Rightarrow (\models A)$ gilt, d.h. wenn nur Tautologien Theoreme des Systems sind. $\mathcal{F}$ heißt **vollständig**, wenn $(\models A) \Rightarrow (\vdash_{\mathcal{F}} A)$ gilt, d.h. wenn sich alle im System darstellbaren Tautologien über $\mathcal{F}$ herleiten lassen. Dabei ist A eine Formel aus der Menge der im deductiven System zulässigen Formeln. In einem correcten und vollständigen System sind die consistenten Formeln somit genau die erfüllbaren Formeln.

Gilt $\Sigma \vdash A$, so ist $\{\Sigma, \neg A\}$ inconsistent. Die Umkehrung gilt, wenn das System vollständig ist, auch. Dadurch werden viele Beweise bedeutend einfacher.

Das System $\mathcal{F}_0$

Ein deductives System für Formeln der Operatormenge $\{\neg, \to\}$, welches wir kennen müssen, ist $\mathcal{F}_0$. Es verwendet den Modus Ponens als einzige Schlussregel und als Axiome die der folgenden drei Axiomenschemata von Jan Łukasiewicz:

Ax1 $A \to (B \to A)$
Ax2 $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$
Ax3 $(\neg A \to \neg B) \to (B \to A)$

Ax1 kann unter anderem benutzt werden, um aus bewiesenen Aussagen Implicationen mit diesen als Consequenz als schwächere Aussagen herzuleiten. Mit Ax2 können wir Klammern nach vorne ziehen und Implicationen $A \to C$ (eventuell in Combination mit Tautologien aus Ax1 als B und $A \to B$) herleiten. Mit Ax3 können wir die Contraposition mit weniger Negationen zu Aussagen bilden; dies kann hilfreich bei negierten Aussagen sein.

Wir haben einige Theoreme, siehe dazu die Folien S. 52.

Für $\mathcal{F}_0$ gilt das Deductionstheorem: Gilt $\Gamma \vdash A \to B$, dann gilt auch $\Gamma, A \vdash A \to B$ und damit auch mit Modus Ponens $\Gamma, A \vdash B$. Für die andere Richtung betrachte man einen Beweis von B aus Γ und A in $\mathcal{F}_0$. Für jeden Beweisschritt kann für die entsprechende Formel per Induction gezeigt werden, dass $\Gamma \vdash A \to B$ gilt. Dabei müssen die drei Fälle unterschieden werden, dass $B \in \Gamma$ oder $B \in Ax$ ist, dass $B \equiv A$, und dass B über zwei Formeln C und $C \to B$ mit Modus Ponens hergeleitet wurde; in letzterem Fall benutzen wir die Inductionsvoraussetzung.

Ein Beispiel für einen Beweis in $\mathcal{F}_0$ aus der Fragestunde zur Zwischenclausur zu $\vdash_{\mathcal{F}_0} B \to ((\neg A \to \neg B) \to A)$ über $B, (\neg A \to \neg B) \vdash_{F_0} \to A$:

$B_0 \equiv (\neg A \to \neg B) \to (B \to A)$	Ax3
$B_1 \equiv (\neg A \to \neg B)$	Vor
$B_2 \equiv B \to A$	MP(B_0,B_1)
$B_3 \equiv B$	Vor
$B_4 \equiv A$	MP(B_2,B_3)

$\mathcal{F}_0$ ist correct, da alle Axiome Tautologien sind, die logischen Folgerungen von Tautologien nur Tautologien sind und mit dem Modus Ponens aus Formeln nur deren logische Folgerungen herleitbar sind.

Aus den einer die Aussage erfüllenden Belegung entsprechenden Literalen in einer beliebigen Aussage läßt sich die Aussage leicht herleiten, z.B. $x \vdash x \to x$, da $\vdash x \to (x \to x)$ gilt und wir den Modus Ponens haben und $\neg x \vdash x \to x$, da $\vdash \neg x \to (\neg x \to \neg x)$ gilt, wir mit Modus Ponens $\neg x \to \neg x$ herleiten können, und mit Ax3 davon die Contraposition $x \to x$ bilden können.

$\mathcal{F}_0$ ist vollständig. Zum Beweis der Vollständigkeit betrachten wir für eine Tautologie alle $2^{|V|}$ erfüllenden Literalcombinationen. Wir können nun jeweils entsprechende Herleitungen finden. Mit dem Deductionstheorem können wir nun zwei Herleitungen, die sich nur in einem Literal im Antecedens unterscheiden, verschmelzen, indem wir bei beiden das Literal auf die rechte Seite bringen. Mit dem Theorem $(A \to B) \to ((\neg A \to B) \to B)$ können wir dann wieder die Tautologie aus einem Literal weniger herleiten. Dies wiederholen wir, bis wir sie aus keinem Literal herleiten können, und so gezeigt haben, dass sie eine Tautologie ist.

Königs Lemma

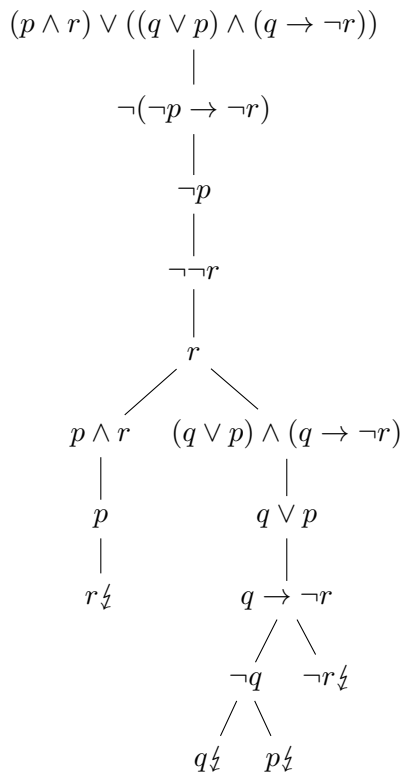
Der **Ausgangsgrad** eines Baumes ist die größte Anzahl an Kindknoten, die ein Knoten im Baum hat. Ein **Pfad** in einem Baum ist eine Folge von Knoten, wobei jeder Knoten außer dem ersten ein Kind oder Elternteil seines Vorgängers ist, also mit seinem Vorgänger verbunden ist. Bei uns wird zudem vorausgesetzt, dass in einem Pfad kein Knoten mehrmals vorkommt. **Königs Lemma** besagt, dass ein unendlicher Baum mit endlichem Ausgangsgrad einen unendlichen (d.h. unendlich langen) Pfad haben muss.

Stellt man sich zum Beispiel ein Dateisystem wie FAT vor, bei dem jede Datei einen kurzen im Verzeichnis eindeutigen Dateinamen mit einer Länge von höchstens 12 Zeichen haben muss, die jeweils in einem Byte codiert werden können. Es gibt also nur endlich viele mögliche Dateinamen. Man kann sich ein solches Dateisystem als Baum vorstellen, bei dem Verzeichnisse und Dateien ebenso wie die Wurzel bzw. Root (geschrieben z.B. als `C:\`) die Knoten sind. Pfade im Baum haben dabei zum Beispiel die Form `C:\WINDOWS\SYSTEM32` (sie müssen aber nicht von der Wurzel ausgehen). Der Baum hat aufgrund der endlichen Anzahl und der Eindeutigkeit von Dateinamen einen endlichen Ausgangsgrad. Pfade sind nicht unendlich, weil das Dateisystem auf einem Datenträger mit endlicher Größe gespeichert sein muss. Damit handelt es sich nach der Contraposition von Königs Lemma nicht um einen unendlichen Baum.

Mit Königs Lemma läßt sich zum Beispiel mit entsprechender Modellierung als Baum auch zeigen, wie lang eine IP-Adresse (als Folge von Bytes) maximal sein müßte, um beliebig viele Geräte zu identifizieren, oder ob es nur endlich viele Formeln mit einer maximalen Tiefe n gibt.

Aussagenlogische Tableaus

Eine andere Methode, um systematisch herzuleiten, ob eine logische Folgerung gilt, sind **Tableaus**. Mit diesen kann die Erfüllbarkeit einer aussagenlogischen Formel geprüft werden. Wir schreiben T_A für die Menge aller Tableaus zu einer einzelnen Formel A und T_Σ für die Menge aller Tableaus zu einer Formelmenge Σ. Ein Beispiel aus der Fragestunde zur Zwischenclausur für ein Tableau $\tau \in T_\Gamma$ zum Zeigen von $(p \wedge r) \vee ((q \vee p) \wedge (q \to \neg r)) \vdash \neg p \to \neg r$ mit $\Gamma = \{(p \wedge r) \vee ((q \vee p) \wedge (q \to \neg r)), \neg(\neg p \to \neg r)\}$:



Es wird versucht, die gemeinsame Erfüllbarkeit einer Menge aussagenlogischer Formeln zu zeigen. Dazu wird ein Baum (das Tableau) entwickelt, indem zunächst alle Formeln der Menge auf den selben Zweig untereinander geschrieben und verbunden werden. Dann werden diese in Teilformen aufgespalten: Conjunctionen resultieren im Hinzufügen der conjugierten Aussagen zum selben Ast am Ende des Astes, Disjunctionen im Hinzufügen der disjugierten Aussagen in verzweigten Ästen. Andere Junctoren müssen entsprechend vereinfacht werden. Näher an der Wurzel befindliche Formeln müssen nicht zuerst aufgespalten bzw. vereinfacht werden, es können auch andere zuerst gewählt werden (dabei ist es oft sinnvoll, erst Conjunctionen aufzuspalten; dies wurde in der Fragestunde zur Zwischenclausur gesagt, entspricht aber nicht dem Algorithmus auf den Folien). Ist kein weiteres Aufspalten und Vereinfachen mehr möglich, ist die Formel genau dann erfüllbar, wenn auf dem Pfad von einem Blatt zur Wurzel keine Widersprüche in Literalen vorkommen; durch die jetzt vorhandene Darstellung in Literalform ist dies einfach zu erkennen. Stellt sich schon vorher heraus, dass ein Ast widersprüchliche Literale enthält, müssen nicht unbedingt alle Formeln darin fertig aufgespalten werden - dabei soll allerdings gelten, dass für jede Formel entweder beide oder keine Teilformel hinzugefügt wurde, nicht nur eine, damit nachvollzogen werden kann, was wir gemacht haben.

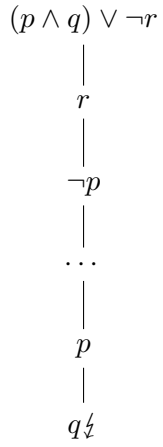
Formeln, die zur Einführung neuer Formeln in den Ast ohne ein Verzweigen des Astes führen, nennen wir α**-Formeln**; Formeln, bei denen der Ast aufgespalten wird, heißen β**-Formeln**. Siehe die Folien S. 66 für eine Auflistung.

Wir bezeichnen einen Ast eines Tableaus als **vollständiger Ast**, wenn kein weiteres Erzeugen von Formeln mehr für diesen Ast möglich ist. Wir bezeichnen ein Tableau als **vollständiges Tableau**, wenn all seine Äste vollständig sind. Es gibt zu jeder Formelmenge ein vollständiges Tableau. Einen Ast mit widersprüchlichen Literalen nennen wir **abgeschlossen**, einen nicht abgeschlossenen Ast nennen wir **offen**. Ein Tableau nur mit abgeschlossenen Ästen nennen wir ebenso abgeschlossen. Verwenden wir diese Bezeichnungen für Formelmengen, so meinen wir damit die entsprechenden zugehörigen Äste, ebenso sagen wir zu den Formelmengen von Ästen auch einfach Äste.

Vollständige, offene Formelmengen nennen wir **Hintikka-Mengen**. Nach dem **Lemma von Hintikka** ist ein vollständiger Ast genau dann erfüllbar, wenn er offen ist. Aus den Literalen eines vollständigen, offenen Asts läßt sich eine die Formelmenge erfüllende Bewertung ablesen. Ein vollständiges Tableau nur mit offenen Ästen basiert also auf allgemeingültigen Formeln.

Da eine logische Folgerung $\Sigma \models A$ genau dann gilt, wenn $\Sigma \cup \{\neg A\}$ unerfüllbar ist, kann mit einem solchen Erfüllbarkeitsprüfer auch die logische Folgerung gezeigt werden. Alternativ kann man für eine endliche Formelmenge diese auch mit dem Deductionstheorem in eine einzelne Formel umwandeln und dann auf Allgemeingültigkeit prüfen. Die Tableau-Methode ist correct und vollständig.

Es gibt auch unendliche Tableaus über unendlichen Formelmengen. In Tableaus ist der Ausgangsgrad endlich (höchstens 2). Damit hat ein unendliches Tableau nach Königs Lemma einen unendlichen (wiederholungsfreien) Pfad; mit dem gegebenen Verfahren kann das Tableau also nie vollständig werden und für unendliche Tableaus semientscheidet das Verfahren die Unerfüllbarkeit nur; der Compactheitssatz kann als Folge davon angesehen werden (eine unendliche Formelmenge ist genau dann unerfüllbar, wenn der Algorithmus terminiert, d.h. wenn er irgendwann einen Widerspruch findet). Ein Beispiel für ein unendliches Tableau zu $\{(p \wedge q) \vee \neg r, r, \neg p, \dots\}$:



Davis-Putnam-Algorithmus

Es gibt verschiedene auf einem Verfahren von Martin Davis und Hilary Putnam aufbauende Algorithmen zum Prüfen der Erfüllbarkeit von Formeln in NNF. Unser Algorithmus functioniert wie folgt: Zuerst wird die **Pure-Literal-Regel**, dass ein Symbol, was nie negiert vorkommt, überall durch eine Constante 1 und ein Symbol, was immer negiert vorkommt, überall durch eine Constante 0 ersetzt werden kann, so oft wie möglich angewendet. Ist die Formel in CNF und kann die Pure-Literal-Regel nicht mehr benutzt werden, so wird die **Unit-Regel**, dass eine in der Conjunction allein stehende atomare Formel (d.h. ein Literal in einer positiven Unit-Clausel) überall durch 1 und eine negiert allein stehende atomare Formel (ein Literal in einer negativen Unit-Clausel) überall durch 0 ersetzt werden darf, so angewendet, bis wir wieder die Pure-Literal-Regel nutzen können. Ist die Formel in CNF und hilft weder die Pure-Literal-Regel noch die Unit-Regel, so wird die **Subsumption-Regel**, dass alle Clauseln für die auch eine Teilclausel in der Conjunction steht gestrichen werden dürfen (dabei werden außerdem noch Clauseln gestrichen, in denen Literale mit der gleichen atomaren Aussage sowohl positiv als auch negativ vorkommen), angewendet und danach, wenn keine andere Regelanwendung durch Anwendung der Subsumption-Regel möglich wird, wird die **Splitting-Regel** benutzt: Der actuelle Ast im Baum der bisherigen Vereinfachungen wird aufgespalten, indem für eine Variable unserer Wahl einmal 1 und einmal 0 angenommen wird. Es wäre besser, wenn wir dazuschreiben, welche Regel wir anwenden. Ein Beispiel zur Formel aus den Folien:

$$
\begin{array}{l}
\neg p \vee ((\neg q \vee r) \wedge (q \vee s) \wedge \neg r \wedge \neg s \wedge (p \vee q)) \\
\qquad\qquad\qquad\qquad\quad \diagup\, p = 1 \qquad \diagdown\, p = 0 \text{ (split)} \\
(\neg q \vee r) \wedge (q \vee s) \wedge \neg r \wedge \neg s \qquad\quad 1 \\
\qquad\qquad\quad \Big| \; \neg r = 1 \text{ (unit)} \\
\qquad\quad \neg q \wedge (q \vee s) \wedge \neg s \\
\qquad\qquad\quad \Big| \; \neg s = 1 \text{ (unit)} \\
\qquad\qquad \neg q \wedge q \\
\qquad\qquad\quad \Big| \; \neg q = 1 \text{ (unit)} \\
\qquad\qquad\quad\; 0
\end{array}
$$

Ein Beispiel aus der Zwischenclausur-Fragestunde mit anderer Notation ($\top$ steht für eine immer wahre Constante, $\bot$ für eine immer falsche):

$$
\begin{array}{c}
p \wedge (\neg p \vee q \vee \neg r) \wedge (q \vee \neg r \vee s) \wedge (\neg q \vee r \vee \neg s) \\
\Big| \; \varphi(p) = 1 \text{ (unit)} \\
(q \vee \neg r) \wedge (q \vee \neg r \vee s) \wedge (\neg q \vee r \vee \neg s) \\
\Big| \; \text{(subsumption)} \\
(q \vee \neg r) \wedge (\neg q \vee r \vee \neg s) \\
\Big| \; \varphi(s) = 0 \text{ (pure)} \\
q \vee \neg r \\
\Big| \; \varphi(q) = 1 \text{ (pure)} \\
\top
\end{array}
$$

Wir schreiben $A[p/0]$ für die Formel, die der Formel A entspricht, nachdem dort alle$p$ durch die Constante 0 ersetzt wurden. Analog schreiben wir $A[p/1]$ für die **Substitution** von p durch 1. Wir sehen 0 als die **leere Clausel** an und interpretieren sie als falsch, während wir 1 als **leere Formel** ansehen und als wahr interpretieren. Keines davon ist allerdings eine aussagenlogische Formel nach unserer Definition.

Das Davis-Putnam-Verfahren ist wesentlich effectiver, wenn die Formel in CNF ist. Wenn wir für eine in der Clausur gegebene Formel mit Davis-Putnam zeigen sollen, dass sie keine Tautologie ist, können wir sie dafür negieren und mit Davis-Putnam zeigen, dass es eine die Negation erfüllende Bewertung gibt; dabei erhalten wir eventuell auch eine schneller zu vereinfachende Formel in CNF.

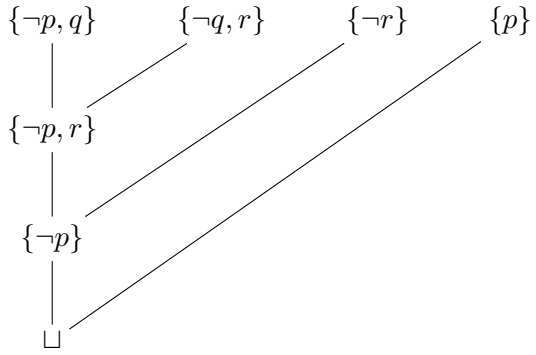
Aussagenlogische Resolution

Resolution ist eine Schlussregel, dass wir für eine Formel in CNF aus zwei Clauseln, in der eine atomare Aussage einmal negiert als negatives und einmal nicht negiert als positives Literal vorkommt, die Vereinigung beider Clauseln ohne die atomare Aussage herleiten können. Zum Beispiel:

Aus	
$\{p$	$, \neg q, r, s\}$
und	
$\{\neg p$	$, \neg q, \neg r, t\}$
können wir	
$\{$	$\neg q, r, s, \neg r, t\}$
herleiten.	

Die neu hergeleitete Clausel nennen wir **Resolvente**. Wir können aus einer Formel A durch eventuell wiederholte Resolution also Clauseln herleiten. Für eine solche Clausel C schreiben wir dann $A \underset{\text{Res}}{\vdash} C$. Diese Art der Herleitung ist als Calcül aber nur correct

und nicht vollständig (d.h. wir können nicht alle semantischen Folgerungen so herleiten, sondern nur die Folgerungen, die Resolventen sind). Resolution ist allerdings **widerlegungsvollständig**, d.h. für einen Widerspruch können wir stets die leere Clausel $\sqcup$ herleiten. Wir können also indirect logische Folgerungen der Form $\Sigma \models A$ zeigen, indem wir die Unerfüllbarkeit der Menge $\{\Sigma, \neg A\}$ aus dem Antecedens Σ und der negierten Folgerung $\neg A$ zeigen als $\{\Sigma, \neg A\} \underset{\text{Res}}{\vdash} \sqcup$. Eine Herleitung über Resolution läßt sich als gerichteter acyclischer Graph darstellen:



Prädicatenlogik

Allgemeines

Die **Prädicatenlogik erster Stufe** (First-Order Logic) ist eine Erweiterung der Aussagenlogik. In der Prädicatenlogik beziehen wir uns auf einen **Datenbereich** bzw. eine **Domäne** D. Dabei handelt es sich um eine Menge von möglichen Werten, die die Variablen in einer prädicatenlogischen Formel annehmen können. Zum Beispiel ist $x + 3 = 8 \wedge x < 5$ eine prädicatenlogische Formel, für die wir x als Symbol für eine Variable definieren können. In der Prädicatenlogik können wir auch **Prädicats- und Functionssymbole** definieren. Prädicatssymbole werden als ein Prädicat gedeutet, das Werte aus dem Datenbereich als Argumente bekommt und einen booleschen Wert zurückgibt; Functionssymbole werden als Function gedeutet, die Werte aus dem Datenbereich bekommt, aber auch wieder einen Wert aus dem Datenbereich zurückgibt. Im vorherigen Beispiel ist $<$ ein Prädicatssymbol, denn es gibt einen Wahrheitswert zurück, während $+$ ein Functionssymbol ist. Beide sind zweistellig - die **Stelligkeit** bzw. Ärität oder Arität ist die Anzahl der Argumente, die ein Prädicat oder eine Function entgegennimmt. Wir schreiben Functionen und Prädicate in der Regel in der Form $<(x, 5)$, bei $<$ gilt die Infix-Notation $x < 5$ bei uns nur als Kurzschreibweise dafür, die Formel müßte also eigentlich so aussehen: $+(x, 3) = 8 \wedge <(x, 5)$. Eine nullstellige Function heißt auch Constante. Im obigen Beispiel sind 3, 8 und 5 Constanten. Die aussagenlogischen Variablen entsprechen in der Prädicatenlogik den nullstelligen Prädicaten. Einen Ausdruck, der einen Wert aus dem Datenbereich hat, nennen wir Term; gibt ein Ausdruck einen Wahrheitswert zurück, handelt es sich um eine Formel. $=$ ist kein Prädicatssymbol, sondern

ein fester Bestandteil der Prädicatenlogik mit der üblichen Bedeutung. Ein weiterer Bestandteil der Prädicatenlogik sind die beiden Quantoren $\forall$ (der **Allquantor**) und $\exists$ (der **Existenzquantor**), die jeweils eine einzige Variable binden können. Nicht durch einen Quantor gebunden vorkommende Variablen („**gebundene Variable**“) nennen wir **freie Variablen**. Die Vorkommen von Variablen in der Formel sind damit entweder frei oder gebunden im **Geltungsbereich** eines Quantors.

Um zu bestimmen, ob eine prädicatenlogische Formel erfüllt ist oder nicht, muss man zunächst festlegen, welche Symbole Functionen und Prädicate sind - die übrigen sind dann Variablen. Dafür definieren wir zusätzlich zur Formel noch eine **Signatur**, welche ein Paar aus einer entscheidbaren Menge von Functions- und einer entscheidbaren Menge von Prädicatssymbolen ist, jeweils mit ihrer Stelligkeit. Ein Beispiel für eine Signatur wäre:
$S = (\{\text{wievieltesSemester}/_1\}, \{\text{istStudent}/_1, \text{wohntIn}/_2\})$.

Für die Menge der prädicatenlogischen Formeln zu einer Signatur S schreiben wir $FO(S)$. Wir können diese Formelmenge wie bei der Aussagenlogik inductiv definieren; diesmal sind die atomaren Formeln aber alle Prädicate und nicht bloß die nullstelligen, außerdem ist die Gleichheit eine atomare Formel, da auch sie nicht aus Teilformeln aufgebaut ist. Für die Menge der Variablen in diesem Context schreiben wir V, für die Menge der Variablen in einer Formel $V(A)$. $FV(A)$ ist die Menge der Variablen, die in der Formel frei vorkommen, während $GV(A)$ die Menge der durch Quantoren gebunden vorkommenden Variablen bezeichnet. Diese müssen nicht disjunct sein - ein Variablensymbol kann in einer Formel einmal frei und einmal gebunden vorkommen, z.B. x in der Formel $A \equiv x > 5 \wedge \exists x\, x = 5$.

Bis jetzt waren alle Definitionen rein syntactisch; zur Bestimmung der Erfülltheit müssen wir den Functions- und Prädicatssymbolen auch noch eine Semantik, d.h. eine Bedeutung, zuweisen und den Datenbereich festlegen, den die Variablen und Functionen annehmen können. Dies tun wir mit einer **Structur** zusätzlich zu unserer Signatur. Eine Structur $\mathcal{M}$ ist ein Paar aus einem nicht leeren Datenbereich D und einer **Interpretation** I. Die Interpretation I ist eine Function, die jedes Functions- und Prädicatssymbol wiederum auf eine Function abbildet, deren Stelligkeit der des Symbols entspricht. Für ein k-stelliges Functionssymbol f ist $I(f)$ eine Function, die von k Elementen des Datenbereichs auf ein einziges abbildet. Bei einem k-stelligen Prädicatsymbol p bildet $I(p)$ von k Datenbereichselementen auf einen Wahrheitswert ab. Wir schreiben auch $f^{\mathcal{M}}$ für $I(f)$ und $p^{\mathcal{M}}$ für $I(p)$. Da Gleichheit kein Prädicat im prädicatenlogischen Sinn ist, wird es auch nicht durch die Structur interpretiert.

Zuletzt gibt es wieder den Begriff der **Belegung**; eine Belegung σ aus der Menge aller möglichen Belegungen D^V ist eine Function, die den Variablen je ein Element des Datenbereichs zuweist, also von V auf D abbildet. Dabei ist zu beachten, dass die aussagenlogischen Variablen in der Prädicatenlogik keine Variablen, sondern Prädicate sind und somit durch die Interpretation und nicht die Belegung eine Bedeutung bekommen. Wir können Belegungen auch modificieren; eine **Modification** einer Belegung $\sigma \in D^V$

ist eine Belegung $\sigma\{x/d\}$, welche alles wie σ belegt, nur, dass es der Variable x den Wert $d \in D$ statt $\sigma(x)$ zuweist. Eine Formel ohne freie Variablen nennen wir **geschlossene Formel** (auf den Folien auch **abgeschlossene Formel**), für eine solche ist die Belegung irrelevant. Wir schreiben $FO_{\text{abg}}(S)$ für die Menge der abgeschlossenen Formeln der Signatur S.

Wir schreiben $\mathcal{M}[\![t]\!](\sigma)$ für die **Semantik** eines Terms t, d.h. den Wert, den t unter der Structur $\mathcal{M}$ mit der zugehörigen Signatur hat, wenn die Variablen durch σ belegt werden. Analog schreiben wir $\mathcal{M}[\![A]\!](\sigma)$ für die Semantik, d.h. den Wahrheitswert, einer Formel A in diesem Context. Das **Coincidenzlemma** besagt, dass eine Formel unter zwei Belegungen die gleiche Semantik hat, wenn die Belegungen alle freien Variablen in der Formel gleich belegen. Dies gilt insbesondere für alle Belegungen bei einer geschlossenen Formel. Siehe auf den Folien S. 118/119 für eine inductive Definition.

Wir sagen, dass eine Formel A unter einer Structur $\mathcal{M}$ und einer Belegung σ gilt, wenn $\mathcal{M}[\![A]\!](\sigma) = 1$ gilt. Dann schreiben wir $\mathcal{M}, \sigma \models A$. Für eine geschlossene Formel A schreiben wir auch $\mathcal{M} \models A$ und nennen $\mathcal{M}$ ein Modell für A. Die Bezeichnungen Tautologie, erfüllbar, allgemeingültig u.s.w. verwenden wir dann wie bei der Aussagenlogik - eine Formel A ist erfüllbar, falls eine Structur und Belegung existiert, die sie erfüllt; und eine Structur ist allgemeingültig, also eine Tautologie, falls jede Structur mit jeder Belegung die Formel erfüllt, dann schreiben wir auch $\models A$. Formeln sind logisch äquivalent, wenn sie unter allen Structuren und Belegungen gleich bewertet werden. Logische Äquivalenz ist noch immer eine Congruenz.

Es gelten die logischen Äquivalencen $\neg\forall xA \models\!\!\!\dashv \exists x\neg A$, $\forall xA \wedge \forall xB \models\!\!\!\dashv \forall x(A \wedge B)$, $\forall x\forall yA \models\!\!\!\dashv \forall y\forall xA$ und analog $\neg\exists xA \models\!\!\!\dashv \forall x\neg A$, $\exists xA \vee \exists xB \models\!\!\!\dashv \exists x(A \vee B)$, $\exists x\exists yA \models\!\!\!\dashv \exists y\exists xA$. Der Compactheitssatz gilt auch für die Prädicatenlogik erster Stufe. Die Aussagen über Formelmengen gelten auch weiterhin: $\Sigma \models A \iff \{\Sigma, \neg A\}$ unerfüllbar, $\emptyset \models A \iff \models A$, Σ unerfüllbar $\iff \Sigma \models A$ für alle $A \in FO(S)$, $\Gamma \models A, \Gamma \subseteq \Sigma \Rightarrow \Sigma \models A$, $A \models\!\!\!\dashv B \Rightarrow \Sigma \models A \iff \Sigma \models B$. $\Gamma \models\!\!\!\dashv \Sigma$ bedeutet, dass $\text{Folg}(\Gamma) = \text{Folg}(\Sigma)$ und dass Γ genau dann erfüllbar ist, wenn Σ erfüllbar ist. Es ist $A \models\!\!\!\dashv B$ genau dann, wenn $A \models B$ und $B \models A$ gelten, d.h. genau dann wenn $\models A \leftrightarrow B$ gilt, d.h. genau dann wenn für alle Structuren $\mathcal{M}$ und Belegungen σ gilt, dass $\mathcal{M}[\![A]\!](\sigma) = \mathcal{M}[\![B]\!](\sigma)$. Deductionstheorem, Modus Ponens und Contraposition gelten weiterhin semantisch. Das **Generalisierungstheorem** besagt, dass wir vor eine Formel einen Allquantor setzen können, der eine in der Formel nicht frei vorkommende Variable bindet, und die so entstandene Formel aus den gleichen Mengen herleitbar wie die alte Formel ist, also logisch äquivalent zur alten ist. Setzen wir hingegen einen Allquantor (oder mehrere) mit einer frei vorkommenden Variable davor, dann nennen wir dies eine **Generalisierung** der alten Formel - diese ist aber nicht mehr unbedingt logisch äquivalent.

$\neq$ ist bei uns als Prädicatssymbol zu verstehen, wenn es in der Aufgabe nicht anders definiert ist. Das heißt, dass $x \neq y$ nicht unbedingt das gleiche ist wie $\neg(x = y)$, sondern dass dies von der Interpretation abhängt.

Der **universelle Abschluss** einer Formel ist die Formel mit neuen, vorangestellten

Allquantoren, wobei für jede freie Variable ein diese bindender Allquantor eingefügt wird. Der **existencielle Abschluss** ist analog definiert mit die freien Variablen bindenden Existenzquantoren.

Eine **Substitution** θ ist eine Abbildung von Variablen auf Terme. Wie bei Modificationen schreiben wir auch $\{x/t\}$. Wir können eine Substitution θ auf einen Term t oder eine Formel A anwenden und ersetzen damit die Variablen darin; wir schreiben $t\theta$ bzw. $A\theta$ für den neuen Term bzw. die neue Formel.

Es gibt auch eine **Prädicatenlogik** zweiter Stufe; dort können wir zudem noch Prädicate durch Quantoren binden. Mit dieser beschäftigen wir uns allerdings nicht.

Normalformen

Für die Prädicatenlogik erster Stufe gibt es einige neue Normalformen. Wir nennen die logisch äquivalente Form einer Formel, bei der alle Quantoren durch obige Umformungen nach außen gezogen wurden, **Pränexnormalform** bzw. **Pränexform**.

Wir bezeichnen eine Formel als **bereinigt**, falls jede Variable nur frei vorkommt oder nur gebunden vorkommt und gebunden vorkommende Variablen nur im Geltungsbereich des selben Quantors vorkommen. Zum Bereinigen einer Formel führen wir wiederholt gebundene Umbenennung durch, d.h. wir ersetzen gebundene Vorkommen von Variablen per Substitution durch Vorkommen einer neuen Variable, wenn die alte Variable bereits frei vorkommt oder in einem anderen Geltungsbereich vorkommt. Die bereinigte Formel ist logisch äquivalent zur noch nicht bereinigten. Eine bereinigte Pränexform kürzen wir mit BPF ab.

Die **Skolemform** (bzw. **Skolemnormalform**) einer Formel ist die Form einer Formel, die eine Pränexnormalform nur mit Allquantoren ist. Zur Umwandlung einer Formel in Pränexnormalform in eine in Skolemform substituieren wir der Reihe nach jede existenzquantificierte Variable mit einer frischen **Skolemfunction** mit der Anzahl der Allquantoren vor dem Existenzquantor als Stelligkeit. Frische Skolemfunction bedeutet hier, dass wir unsere Signatur um ein neues Functionssymbol erweitern. Die so neu eingeführten Functionssymbole repräsentieren den Variablenwert, für den die Formel erfüllt wird, sofern es einen solchen gibt. Gibt es keinen, so ist egal, wofür sie stehen. Die Einführung von Skolemfunctionen nennen wir **Skolemisierung**. Wir schreiben Sko für die Menge der Skolemfunctionen; die neue Signatur $S \uplus \mathrm{Sko}$ der Formel hat nun als Functionssymbole auch die Skolemfunctionen ($\uplus$ steht für die Vereinigung disjuncter Mengen, gemeint ist hier die Vereinigung von Sko mit den Functionssymbolen von S).

Die Skolemform ist nicht mehr notwendigerweise logisch äquivalent, sondern nur **erfüllbarkeitsäquivalent**, d.h. sie ist genau dann erfüllbar, wenn die ursprüngliche Formel erfüllbar ist, wird aber nicht unbedingt von den gleichen Structuren und Belegungen erfüllt.

Im alten Logik-Script von Madlener steht auf Seite 60 ein Verfahren, um eine beliebige Formel in Skolemform (dort heißt sie **Clauselnormalform**) zu bringen und dabei zu minimieren. Ein Beispiel dafür für die Formel $\neg(\forall z_1[q(z_1)] \vee \neg\forall x[(q(x) \vee r(x)) \wedge \exists z_2[\neg p(z_2) \wedge (p(z_2) \vee \neg r(x))]])$ aus Präsenzblatt 7:

1. Da keine freien Variablen in der Formel vorkommen, ist der existenzielle Abschluss der Formel noch immer das gleiche wie die Negation, also $\neg(\forall z_1[q(z_1)] \vee \neg\forall x[(q(x) \vee r(x)) \wedge \exists z_2[\neg p(z_2) \wedge (p(z_2) \vee \neg r(x))]])$.

2. Überflüssige Quantoren haben wir auch keine.

3. Keine Variable wird mal von einem und mal von einem anderen Quantor gebunden, d.h. die "Variablennamen" haben in der ganzen Formel immer die gleiche Bedeutung. Also müssen wir hier noch keine Variablen umbenennen und die Formel ist immernoch die gleiche wie vorher.

4. $\rightarrow$ und $\leftrightarrow$ kommen auch nicht vor; die Formel bleibt gleich.

5. $\neg$ muss nach innen gezogen werden (wie bei der Negationsnormalform): Aus der Formel $\neg(\forall z_1[q(z_1)] \vee \neg\forall x[(q(x) \vee r(x)) \wedge \exists z_2[\neg p(z_2) \wedge (p(z_2) \vee \neg r(x))]])$ wird damit zuerst $\neg\forall z_1[q(z_1)] \wedge \forall x[(q(x) \vee r(x)) \wedge \exists z_2[\neg p(z_2) \wedge (p(z_2) \vee \neg r(x))]])$ und dann $\exists z_1 \neg q(z_1) \wedge \forall x[(q(x) \vee r(x)) \wedge \exists z_2[\neg p(z_2) \wedge (p(z_2) \vee \neg r(x))]])$.

6. Quantoren müssen so weit wie möglich nach innen gezogen werden; das sind sie aber bereits.

7. Wir skolemisieren. Dabei ersetzen wir die Variable z_1, die bei ihrer existenziellen Quantificierung $\exists z_1$ in keinem Geltungsbereich eines Allquantors liegt, durch eine Skolem-Constante f_{z_1} und die Variable z_2, bei der $\exists z_2$ im Geltungsbereich von $\forall x$ liegt, durch die einstellige Skolemfunction f_{z_2}. Wir erhalten $\neg q(f_{z_1}) \wedge \forall x[(q(x) \vee r(x)) \wedge [\neg p(f_{z_2}(x)) \wedge (p(f_{z_2}(x)) \vee \neg r(x))]])$.

8. Zum Schluss ziehen wir alle verbleibenden Allquantoren (also hier nur $\forall x$) nach außen. Damit erhalten wir $\forall x\{\neg q(f_{z_1}) \wedge (q(x) \vee r(x)) \wedge \neg p(f_{z_2}(x)) \wedge (p(f_{z_2}(x)) \vee \neg r(x))\}$.

9. Jetzt müßten wir die Formel in conjunctive Normalform bringen; das ist sie aber schon.

Für die Menge der prädicatenlogischen Formeln einer Signatur S ohne $=$-Symbol schreiben wir $FO^{\neq}(S)$. Jede Formel in $FO(S)$ läßt sich in eine erfüllbarkeitsäquivalente Formel in $FO^{\neq}(S')$ umwandeln, indem wir ein neues zweistelliges Prädicat p in die Signatur einführen (wir erhalten eine erweiterte Signatur S') und alle Vorkommen von $t_1 = t_2$ in der Formel durch $p(t_1, t_2)$ ersetzen. Dann conjugieren wir die ganze Formel mit den Eigenschaften, die p erfüllen muss, um eine Äquivalenzrelation darzustellen: $\forall x p(x, x)$ (Reflexivität), $\forall x \forall y \forall z((p(x, y) \wedge p(y, z) \rightarrow p(x, z))$ (Transitivität) und $\forall x \forall y(p(x, y) \leftrightarrow p(y, x))$ (Symmetrie). Außerdem hat Gleichheit noch die Eigenschaft, dass für jede n-stellige Function gilt, dass $\forall x_1 \ldots \forall x_n \forall y_1 \ldots \forall y_n(\bigwedge_{i=1}^{n} x_i = y_i) \rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$. Dies entspricht nicht notwendigerweise genau der Äquivalenz, denn zwei Elemente des

Datenbereichs können sich in allen für die Erfüllbarkeit relevanten Eigenschaften genau gleich verhalten, aber dennoch unterschiedlich sein. Zum Beispiel wäre folgendes möglich: name(x) = Chris $\wedge\neg(x = y)$. Dies könnte man so interpretieren, dass Person x Chris heißen muss, aber nicht Person y sein darf. Angenommen wir wählen solch ein x mit unserer Belegung. Ersetzen wir nun aber die Formel durch name(x) = Chris $\wedge\neg p(x, y) \wedge \forall x p(x, x) \wedge \forall x \forall y \forall z((p(x, y) \wedge p(y, z) \rightarrow p(x, z)) \wedge \forall x \forall y(p(x, y) \leftrightarrow p(y, x)) \wedge \forall x \forall y(p(x, y) \rightarrow p(name(x), name(y)))$, dann könnten wir p interpretieren als „hat den gleichen Namen wie“-Relation und die Formel wäre nicht erfüllt.

Herbrandisierung

Die Problemstellung, ob eine gegebene Formel der Prädicatenlogik erster Stufe allgemeingültig ist, ist nicht entscheidbar, sondern nur semi-entscheidbar; man kann die duale Problemstellung, ob eine Formel unerfüllbar ist, semi-entscheiden, und damit auch semi-entscheiden, ob die Negation der gegebenen Formel unerfüllbar ist, d.h. ob die Formel allgemeingültig ist. Der **Satz von Church** sagt aus, dass das Allgemeingültigkeitsproblem für Formeln erster Stufe nicht nur semi-entscheidbar ist, sondern sogar **hart** für die Classe der semi-entscheidbaren Probleme ist, d.h. von jedem semi-entscheidbaren Problem läßt sich die Semi-Entscheidbarkeit beweisen, indem man es auf das Allgemeingültigkeitsproblem zurückführt („**reduciert**“). Wir sagen, dass ein Problem, was hart für eine Complexitätsclasse ist und zudem darin enthalten ist, **vollständig** in der Classe enthalten ist.

Um eine Formel in $FO^{\neq}(S)$ auf Erfüllbarkeit zu prüfen, müssen wir nicht alle möglichen Datenbereiche prüfen. Es genügt, sie in Skolemform umzuwandeln und dann für alle möglicherweise unterschiedlichen Terme unterschiedliche Werte zuzulassen. Dazu wählen wir als Datenbereich $D_{\mathcal{H}}$ alle möglichen variablenfreien Terme; diese können dementsprechend nur aus Functionssymbolen bestehen. Als Interpretation $I_{\mathcal{H}}(f)$ eines jeden Functionssymbols f bilden wir dann dieses Symbol auf den zugehörigen Term ab: $I_{\mathcal{H}}(f)(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$. Die Prädicatssymbole können weiterhin beliebig interpretiert werden. Eine solche Structur $\mathcal{H} = (D_{\mathcal{H}}, I_{\mathcal{H}})$ nennen wir **Herbrand-Structur**. Nach dem **Satz von Herbrand** ist eine Formel genau dann erfüllbar, wenn sie ein **Herbrand-Modell** hat, d.h. wenn sie eine Herbrand-Structur hat, die für sie ein Modell ist. Dadurch gilt auch der **Satz von Löwenheim-Skolem**: Eine erfüllbare Formel hat ein Modell mit abzählbarem Datenbereich (nämlich das Herbrand-Modell - da dieses recursiv definiert ist, ist es auch abzählbar).

Für eine Formel $A \in FO^{\neq}(S)$ in Skolemform nennen wir die Menge aller Formeln $E(A)$, bei denen alle Variablen durch Terme in $D_{\mathcal{H}}$ substituiert und die sie bindenden Allquantoren entfernt wurden, ihre **Herbrand-Expansion**. $E(A)$ ist abzählbar. Eine Formel in $E(A)$ kann nun nach einem **Gödel-Herbrand-Skolem-Satz** mit den Methoden aus der Aussagenlogik auf Erfüllbarkeit hin getestet werden, da in ihr keine prädicatenlogischen Variablen (weil sie in der Herbrand-Expansion liegt) und Functionen (durch die Skolemisierung) mehr vorkommen und die endlich vielen Prädicate in der Formel als

aussagenlogische Variablen aufgefaßt werden können - die erfüllende Interpretation der Prädicate entspricht also einer erfüllenden aussagenlogischen Belegung.

Solche variablenfreien Terme und Formeln nennen wir **Grundterme** bzw. **Grundformeln**.

Ein Herbrand-Modell muss allerdings für alle Variablenbelegungen gültig sein, d.h. A hat genau dann ein Herbrand-Modell, wenn die ganze Formelmenge $E(A)$ aussagenlogisch erfüllbar ist. Um die Formel A auf Unerfüllbarkeit hin zu überprüfen, müssen wir also der Reihe nach Formeln aus der Herbrand-Expansion auf Erfüllbarkeit hin überprüfen; sobald wir eine unerfüllbare Formel finden, ist nach dem Compactheitssatz der Aussagenlogik auch $E(A)$ unerfüllbar, in diesem Fall hat dann A kein Herbrand-Modell und damit ist A unerfüllbar. Damit ist die Unerfüllbarkeit einer prädicatenlogischen Formel semi-entscheidbar.

Den Algorithmus, der eine Aufzählung $\{A_1, A_2, A_3, \dots\}$ der Elemente der Herbrandexpansion $E(A)$ als Eingabe nimmt und der Reihe nach jedes einer anfangs leeren Menge von Formeln hinzufügt, diese auf aussagenlogische Erfüllbarkeit hin prüft und bei deren Unerfüllbarkeit terminiert, nennen wir **Algorithmus von Gilmore**. Er semi-entscheidet die Unerfüllbarkeit von A.

Das System $\mathcal{F}$

Es gibt ein deductives System $\mathcal{F}$ für die Prädicatenlogik, welches correct und vollständig ist. Es hat als Schlussregel den Modus Ponens und als Axiome alle aussagenlogischen Tautologien (benötigt davon aber nur die eines nur Modus Ponens benutzenden correcten, vollständigen deductiven Systems der Aussagenlogik wie $\mathcal{F}_0$) und außerdem $\forall x\, A \to A\{x/t\}$ (für einen beliebigen Term t), $\forall x\, (A \to B) \to (\forall x A \to \forall x B)$, $A \to \forall x A$ (für $x \notin FV(A)$), $x = x$, $x = y \to (A \to A')$ (für ein A', dass aus A durch Ersetzen von einigen freien Vorkommen von x durch y entstanden ist). Es gelten dafür insbesondere das Deductions- und das Generalisierungstheorem ebenso wie die Contraposition (da es ja correct und vollständig ist) und die Aussagen über Consistenz, die auch in der Aussagenlogik galten. Den Satz über die Correctheit und Vollständigkeit, dass Σ genau dann erfüllbar ist, wenn es nach $\mathcal{F}$ consistent ist, und $\Sigma \underset{\mathcal{F}}{\vdash} A \iff \Sigma \models A$ gilt, nennen wir **Satz von Gödel**.

Theorien

Eine **Theorie** erster Stufe ist eine Menge von geschlossenen Formeln $\Gamma \subseteq FO(S)$, die abgeschlossen gegenüber logischer Folgerung ist, d.h. in der auch alle Folgerungen jeder enthaltenen Formel enthalten sind. Wir schreiben T_S für die Theorie, die genau alle Tautologien der Signatur S beinhaltet. Wir schreiben T_Σ für die Theorie, die genau Σ und alle logischen Folgerungen daraus beinhaltet (vorausgesetzt alle Formeln in Σ

sind geschlossen). Wir schreiben $T_{\mathcal{M}}$ für die Theorie, die alle geschlossenen Formeln beinhaltet, für die $\mathcal{M}$ ein Modell ist.

Eine Theorie T wird als **vollständige Theorie** bezeichnet, wenn für jede abgeschlossene Formel $A \in FO_{\text{abg}}(S)$ der Signatur gilt, dass entweder A oder $\neg A$ in T enthalten ist. Für eine Structur $\mathcal{M}$ ist die zugehörige Theorie $T_{\mathcal{M}}$ also immer vollständig und consistent; außerdem gibt es für jede vollständige, consistente Theorie eine solche Structur $\mathcal{M}$. Eine Theorie T ist genau dann erfüllbar, wenn sie consistent ist, also für keine Formel auch deren Negation enthält. Eine Theorie T nennen wir **aufzählbar axiomatisierbar**, wenn es eine aufzählbare Menge Σ abgeschlossener Formeln gibt, so dass $T = T_\Sigma$. T ist **endlich axiomatisierbar**, falls es ein endliches solches Σ gibt. Aufzählbar axiomatisierbare Theorien sind wegen der recursiven Definition der Herleitung auch selbst aufzählbar. Eine vollständige, aufzählbar axiomatisierbare Theorie ist entscheidbar.

Es gibt einige bekannte Axiomatisierungen, aus denen die Theorien zum Beispiel der **Presburger-Arithmetik** und der dafür üblichen Structur mit der üblichen Bedeutung (wir nennen eine solche Structur **Standardmodell** der Theorie) zu nur der Addition der natürlichen Zahlen und den Constanten 0 und bei uns auch 1 sowie der Gleichheit als Prädicat hergeleitet werden können. Die Presburger-Arithmetik ist durch die Axiomatisierung sogar entscheidbar.

Die **First-Order-Arithmetik** mit dem Standardmodell, das zusätzlich noch die Multiplication beinhaltet, ist eine Theorie, die vollständig, aber nicht aufzählbar axiomatisierbar und somit nicht entscheidbar ist. Die **Peano-Axiome** (genauer gesagt die für die Prädicatenlogik erster Stufe; dies gilt nicht für die Peano-Axiome zur Prädicatenlogik zweiter Stufe) sind eine Axiomatisierung für die First-Order-Arithmetik, die zwar aufzählbar, aber nicht vollständig ist, d.h. es gibt geschlossene Formeln, für die die First-Order-Arithmetik ein Modell ist, die aber nicht aus den Peano-Axiomen folgen, da es auch Nichtstandardmodelle zu den Peano-Axiomen mit mehr als nur den natürlichen Zahlen gibt, für die andere Formeln gelten.

Es gibt auch für andere Theorien wie Arrays Axiomatisierungen wie die **McCarthy-Axiome**, deren Theorie ebenfalls nicht vollständig ist.

Tableaus

Wir versuchen, Algorithmen zur systematischen Semi-Entscheidung der Allgemeingültigkeit prädicatenlogischer Formeln zu finden. Im Gegensatz zum Algorithmus von Gilmore sollen diese selbstständig der Reihe nach die Elemente der Herbrand-Expansion durchprobieren.

Hier betrachten wir nur geschlossene Formeln in $FO^{\neq}(S)$. Wir prüfen die Unerfüllbarkeit genau wie bei der Aussagenlogik mit Tableaus, wobei die atomaren Formeln nun Prädicate sind. Wir haben zusätzlich noch **γ- und δ-Formeln**, die das Einführen von neuen Formeln in den Ast erlauben. γ-Formeln sind $\forall x A$ und $\neg\exists x A$; mit ihr können wir $A\{x/t\}$

bzw. $\neg A\{x/t\}$ für einen beliebigen Grundterm t am selben Ast einführen - wir können also die gleiche γ-Formel mehrfach zum Hinzufügen von neuen Formeln benutzen und müssen dies teils auch, um Unerfüllbarkeit zu zeigen. δ-Formeln sind $\exists x A$ und $\neg\forall x A$, womit wir eine einzige Formel $A\{x/c\}$ bzw. $\neg A\{x/c\}$ mit einer frischen Skolemconstante c dem Ast hinzufügen können (es handelt sich immer um eine Skolemconstante und keine mehrstellige Function, da wir im Tableau den Existenzquantor bzw. negierten Allquantor nur dann ersetzen, wenn die Variable nicht im Geltungsbereich eines weiteren Quantors liegt). Die frische Skolemconstante kann dann auch in den Folgerungen aus γ-Formeln benutzt werden. Es ist oft sinnvoll, erst mit existierenden Constanten Widersprüche zu suchen, bevor wir neue einführen. Ein Beispiel von den Folien S. 177/178, was schief gehen kann, wenn man neue Constanten ausprobiert, bevor man alle alten probiert hat:

Wir suchen Modelle für $\{\exists x\neg p(x,x), \forall x\exists y p(x,y)\}$:

$\exists x\neg p(x,x)$
|
$\forall x\exists y p(x,y)$
|
$\neg p(a,a)$
|
$\exists y p(a,y)$
|
$p(a,b)$
|
$\exists y p(b,y)$
|
$p(b,c)$
|
$\ldots$

Besser:

$$
\begin{array}{c}
\exists x \neg p(x,x) \\
| \\
\forall x \exists y p(x,y) \\
| \\
\neg p(a,a) \\
| \\
\exists y p(a,y) \\
| \\
p(a,b) \\
| \\
\exists y p(b,y) \\
| \\
p(b,a)
\end{array}
$$

Das Tableau-Verfahren ist correct und vollständig, muss aber nicht terminieren.

Resolution

Die Herbrand-Expansion läßt sich wie in der Aussagenlogik resolvieren, da die Formelmenge als eine große Conjunction darstellbar ist und dann in CNF ist (z.B. entspricht $E(A) = \{p(a) \wedge \neg p(f(a)), p(f(a)) \wedge \neg p(f(f(a))), \dots\}$ die CNF $p(a) \wedge \neg p(f(a)) \wedge p(f(a)) \wedge \neg p(f(f(a))) \wedge \dots$, welche sich durch Resolution schnell als unerfüllbar herausstellt). Offensichtlich könnten wir auch direct die Substitution für die einzelnen Clauseln statt für die ganzen Formeln in $E(A)$ durchführen, wodurch weniger in der Resolution nicht weiter verwendete Clauseln entständen. Wir substituieren dann nur die Clauseln, die wir im nächsten Resolutionsschritt brauchen. Die Substitutionen nennen wir **Grundsubstitutionen**, wenn wir damit Variablen durch Terme ersetzen, und die Resolventen daraus **Grundinstancen**. Statt Grundsubstitutionen durchzuführen, reicht es, mit eventuell neuen Variablen zu substituieren.

Eine Substitution, die verschiedene Literale einer Menge auf das gleiche Literal abbildet, nennen wir **Unificator** (engl. „unifier“). Existiert ein Unificator für eine Literalmenge, so nennen wir die Menge unificierbar. Ein allgemeinster **Unificator** („most general unifier“/ **MGU**) ist ein Unificator ohne unnötige Substitutionen, d.h. einer, bei dem sich jeder andere Unificator als Abbildungsproduct aus dem allgemeinsten Unificator und weiteren Substitutionen darstellen läßt.

Zum Finden des allgemeinsten Unificators zweier Literale schauen wir der Reihe nach alle Symbole in den Literalen durch, bis wir einen Unterschied in den Literalen finden. Ist in einem der Literale dann eine Variable an dieser Stelle, so ersetzen wir die Variable durch den Term, der an dieser Stelle im anderen Literal steht, sofern in dem Term nicht

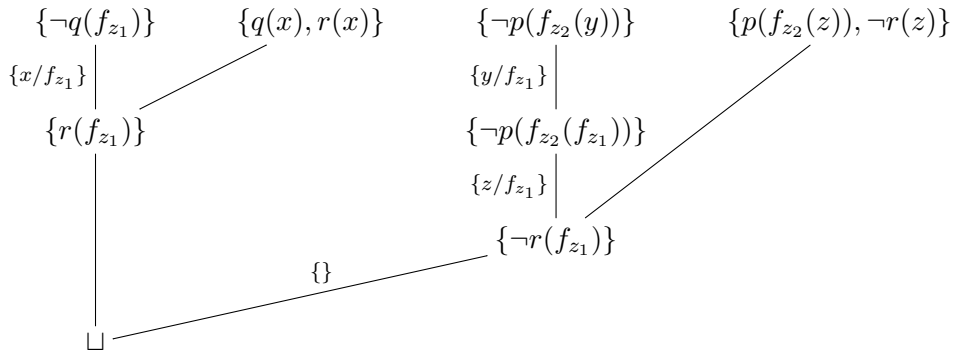
auch die Variable steht (gibt es keine Variable an der Stelle oder kommt die Variable auch im anderen Term vor, so gibt es keinen Unificator für die Literale). Dann suchen wir weiter. Erreichen wir das Ende der beiden Literale, so haben wir den allgemeinsten Unificator gefunden. Den Unificator wenden wir zusammen mit einem Resolutionsschritt an.

Als Beispiel die eigentliche Resolution aus dem Präsenzblatt; dazu schreiben wir die Clauseln als Mengen:

$\{\neg q(f_{z_1})\}$ $\{q(x), r(x)\}$ $\{\neg p(f_{z_2}(x))\}$ $\{p(f_{z_2}(x)), \neg r(x)\}$

Wir führen die Resolution durch:

Bei der Resolution sind alle Variablen, die in den Clauseln vorkommen, allquantificiert (da wir vorher den universellen Abschluss bilden). Deshalb dürfen und müssen wir am Anfang der eigentlichen Resolution den Variablen, die in mehreren Clauseln vorkommen, verschiedene Namen geben. Ebenso geben wir den Variablen in Resolventen bei der Modificationen nur vorher noch nicht verwendete Namen. Diese Disjunctheit der Variablen von denen in alten Clauseln ist nicht immer nötig, z.b. dann nicht, wenn die jeweilige alte Clausel mit den selben Namen nicht mehr für die Resolution verwendet wird. Es ist aber nie falsch, doch sicherheitshalber neue Namen zu wählen.



Die prädicatenlogische Resolution ist wieder correct und widerlegungsvollständig, das Verfahren terminiert allerdings nicht, wenn wir immer die falschen Clauseln resolvieren.