Theoretische Grundlagen der Logikprogrammierung

Skript zur Vorlesung
von
Thomas Piecha

Sommersemester 2018

Universität Tübingen
Fachbereich Informatik

Inhaltsverzeichnis

1 Einleitung

1.1 Literatur

In dieser Vorlesung werden die theoretischen Grundlagen der Logikprogrammierung behandelt. Es werden unter anderem folgende Quellen verwendet (weitere finden sich im Literaturverzeichnis):

- K. R. Apt (1997), *From Logic Programming to Prolog*. London: Prentice Hall.
- J. W. Lloyd (1993), *Foundations of Logic Programming*, 2nd edition. Berlin: Springer-Verlag.
- S.-H. Nienhuys-Cheng & R. de Wolf (1997), *Foundations of Inductive Logic Programming*. Lecture Notes in Artificial Intelligence 1228, Berlin: Springer.

In Teilen orientiert sich die Vorlesung auch an einer von Robert Stärk an der ETH Zürich gehaltenen Vorlesung (Stärk, 2000), an einer auf diese zurückgreifende Vorlesung, die Reinhard Kahle im Wintersemester 2000/2001 an der Universität Tübingen gehalten hat (Kahle, 2001), meinen Mitschrieben aus ebendieser Vorlesung, sowie an Vorlesungen von Peter Schroeder-Heister. Es wird darauf verzichtet, jede Verwendung dieser Quellen immer kenntlich zu machen.

Die praktische Programmierung in Prolog steht hier im Hintergrund. Als Einführungen seien dennoch genannt:

- P. Blackburn, J. Bos & K. Striegnitz (2006), *Learn Prolog Now!*. London: College Publications. Frei zugänglich unter http://www.learnprolognow.org/.
- W. F. Clocksin & C. S. Mellish (2003), *Programming in Prolog*, 5th edition. Berlin: Springer-Verlag.

1.2 Programmierparadigma

Die *Logikprogrammierung* (und deren Implementierung durch Programmiersprachen wie *Prolog* oder *SWI-Prolog*) folgt dem Paradigma der *deklarativen* Programmierung. Hier steht die Frage

Was wird programmiert?

im Vordergrund: Spezifikationen fungieren als Programm. Mit der deklarativen Interpretation von Logikprogrammen ist deren prozedurale Interpretation verbunden: Wie geht eine Berechnung vonstatten? Zur deklarativen Programmierung zählen neben der Logikprogrammierung die funktionalen Programmiersprachen, wie z. B. *Scheme* und *Haskell*.
Im Gegensatz dazu steht bei der *imperativen* Programmierung die Frage im Vordergrund, *wie* etwas programmiert wird bzw. wie etwas berechnet werden soll. Beispiele für imperative Programmiersprachen sind *C* und *Pascal*. Die *objektorientierte* Programmierung bedient sich beider Paradigmen.

Das Paradigma der Logikprogrammierung soll an folgenden Beispielen (aus Apt, 1997) veranschaulicht werden.

Beispiel. Wir möchten eine Datenbank mit Flugverbindungen erstellen, die es uns erlaubt, Fragen der folgenden Art zu beantworten:

– Gibt es einen Direktflug von A nach B?

– Gibt es einen Flug von A nach B?

– Welche Ziele kann ich von A aus erreichen?

Die Direktflüge seien durch folgende Datenbank gegeben:

```
direct(amsterdam, seattle).
direct(amsterdam, paramaribo).
direct(seattle, anchorage).
direct(anchorage, fairbanks).
```

Um auch indirekte Flüge zu berücksichtigen, müssen wir noch angeben, was eine Flugverbindung im Allgemeinen ist:

```
connection(X,Y) :- direct(X,Y).
connection(X,Y) :- direct(X,Z), connection(Z,Y).
```

Die erste Klausel besagt, dass es eine Verbindung von X nach Y gibt, falls es einen Direktflug von X nach Y gibt.
Die zweite Klausel berücksichtigt indirekte Flüge; sie besagt, dass es eine Verbindung von X nach Y gibt, falls es einen Direktflug von X nach Z und eine Verbindung von Z nach Y gibt.

Bemerkung. Wir verwenden hier die Syntax von Prolog: Kleingeschriebene Ausdrücke stehen für Relationen, Funktionen oder Konstanten. Zum Beispiel ist direct eine Relation, und amsterdam ist eine Konstante. Funktionen kommen in diesem Beispiel noch nicht vor. Großgeschriebene Ausdrücke stehen stets für Variablen (im Beispiel X, Y und Z). Der Ausdruck ':-' steht für 'falls', das Komma für 'und'. Der Punkt schließt eine Klausel (oder auch eine Anfrage an ein Logikprogramm) ab. Eine Klausel mit leerer Bedingung wird auch *Faktum* genannt (Beispiele sind die obigen Klauseln für Direktflüge). Klauseln mit Bedingung(en) heißen *Regeln*. Falls es nicht um konkrete Prolog-Programme, sondern allgemein um Logikprogramme geht, werden wir die Sprache der Aussagen- bzw. Prädikatenlogik verwenden.

Die 6 Klauseln unserer Flugdatenbank sind ein Prolog-Programm. Mit ihm können wir z. B. die Frage beantworten, ob es einen Flug von Amsterdam nach Fairbanks gibt:

```
?- connection(amsterdam, fairbanks).
yes
```

Die erste Zeile stellt die Anfrage an das vorher geladene Programm dar. Die zweite Zeile ist die Ausgabe des Prolog-Interpreters (bei SWI-Prolog wäre die Ausgabe true).
Man kann auch fragen, wohin man von Seattle aus fliegen kann:

```
?- connection(seattle, X).
X = anchorage ;
X = fairbanks ;
no
```

Als erste Antwort erhält man die Variablenbindung X = anchorage. Durch Eingabe von ';' kann der Benutzer weitere Antworten abfragen. Falls es keine weiteren positiven Antworten gibt, erhält man die Ausgabe no (bzw. false bei SWI-Prolog). Stellt man die obige Frage für Fairbanks, so erhält man als Antwort direkt no:

```
?- connection(fairbanks, X).
no
```

Das Beispiel zeigt zwei Aspekte der Logikprogrammierung:

(i) *Ein* Programm kann Antworten für *verschiedene* Probleme (bzw. Fragen) liefern.

(ii) Ein Programm kann wie eine Datenbank verwendet werden. Eine relationale Datenbank lässt sich durch die alleinige Angabe von Fakten implementieren. Darüberhinaus können durch die Verwendung von Regeln jedoch auch sogenannte *deduktive Datenbanken* implementiert werden. Diese ermöglichen die Abfrage von Daten, die nicht unbedingt direkt abgelegt wurden, sondern unter Verwendung der Regeln berechnet werden.

Bezüglich der intendierten Interpretation einer Klausel wie

```
connection(X,Y) :- direct(X,Z), connection(Z,Y)
```

haben wir schon gesagt, dass ':-' für 'falls' und das Komma für 'und' stehen soll. Wie aber sind die Variablen X, Y und Z aufzufassen? Die Klauseln sollen stets als *allquantifiziert* gelesen werden; also obige Klausel umgangssprachlich als

> Für alle X, Y und Z gilt connection(X,Y), falls direct(X,Z) gilt und connection(Z,Y) gilt.

Formal:

$$\forall X \forall Y \forall Z((\mathsf{direct}(X,Z) \wedge \mathsf{connection}(Z,Y)) \rightarrow \mathsf{connection}(X,Y))$$

Da hier die Variable Z nur im Antezedens der Implikation vorkommt, können wir dies umformen zu

$$\forall X \forall Y(\exists Z(\mathsf{direct}(X,Z) \wedge \mathsf{connection}(Z,Y)) \rightarrow \mathsf{connection}(X,Y))$$

Beziehungsweise umgangssprachlich:

> Für alle X, Y gilt connection(X,Y), falls es ein Z gibt, so dass direct(X,Z) gilt und connection(Z,Y) gilt.

Beispiel. Wir möchten alle Elemente finden, die in zwei gegebenen Listen enthalten sind.

Eine Liste von Elementen $a_1, \ldots, a_n$ wird durch $[a_1, \ldots, a_n]$ oder alternativ durch $[a_1 \mid [a_2, \ldots, a_n]]$ dargestellt. Es ist a_1 der *Kopf* von $[a_1, \ldots, a_n]$, und $[a_2, \ldots, a_n]$ ist der *Rest* von $[a_1, \ldots, a_n]$. Es ist z. B. 1 der Kopf von $[1, 2, 3, 4, 5]$ und $[2, 3, 4, 5]$ der Rest von $[1, 2, 3, 4, 5]$; es ist $[1, 2, 3, 4, 5] = [1 \mid [2, 3, 4, 5]]$.

Das folgende Programm definiert, wann ein Element X in einer Liste enthalten ist. Es sind zwei Fälle zu behandeln: Falls X Kopf einer Liste ist, dann ist die Antwort positiv. Andernfalls müssen wir prüfen, ob X im Rest enthalten ist.

```
member(X, [X | List]).
```

```
member(X, [Y | List]) :- member(X, List).
```

Um nun unser Problem zu lösen, fügen wir einfach folgende Regel hinzu:

```
member_both(X, L1, L2) :- member(X, L1), member(X,L2).
```

Sie besagt, dass X sowohl in L1 als auch in L2 enthalten ist, falls X Element von L1 ist und X Element von L2 ist. Für die Listen [1,2,3] und [2,3,4,5] erhält man dann:

```
?- member_both(X, [1,2,3], [2,3,4,5]).
X = 2 ;
X = 3 ;
no
```

Für ein imperatives Programm bräuchte man hierfür mindestens zwei geschachtelte Schleifen (vgl. das Beispiel für ein Pascal-Programm in Apt, 1997, S. 5).

Im Gegensatz zum imperativen Programm können mit dem Prolog-Programm auch noch andere Anfragen bearbeitet werden, wie z. B. der Test, ob ein bestimmtes Element in beiden Listen enthalten ist:

```
?- member_both(2, [1,2,3], [2,3,4,5]).
yes
```

oder die Instantiierung eines Listenelements:

```
?- member_both(2, [1,2,3], [X,3,4,5]).
X = 2
```

Bei der Frage nach dem schon in beiden Listen enthaltenen Element 3 liefert SWI-Prolog erst die Belegung X = 3 und als weitere Antwort true, da 3 für alle Belegungen von X (also unabhängig davon) in beiden Listen enthalten ist; dies sind alle Antworten, und eine weitere Anfrage liefert false:

```
?- member_both(3, [1,2,3], [X,3,4,5]).
X = 3 ;
true ;
false.
```

Die Länge der Listen muss nicht vorher festgelegt sein; auch unendliche Listen sind zugelassen.

Bemerkung. Die Frage

```
?- member(X, [f(X), X]).
```

soll testen, ob X in der Liste [f(X), X] vorkommt. Prolog antwortet hierauf jedoch nicht wie erwartet mit yes, sondern mit der (nicht abbrechenden) Ausgabe

```
X = f(f(f(f( ...
```

Dieses Problem lässt sich durch einen *occur check* (auch: *occurs check*) vermeiden, bei dem überprüft wird, ob eine in einem Term vorkommende Variable durch den Term selbst ersetzt werden könnte. Aus Effizienzgründen verzichtet Prolog aber darauf. Auch in SWI-Prolog wird standardmäßig kein occur check durchgeführt; jedoch werden sogenannte *cyclic terms*, d. h. Terme, für die $X = f(X)$ gilt, unterstützt:

```
?- member(X, [f(X), X]).
X = f(X) ;
true ;
false.
```

Der Verzicht auf den occur check hat den Verlust von Korrektheit zur Folge: Obgleich aus dem Programm

```
gleich(X,X).
gleich(0,1) :- gleich(X,f(X)).
```

die Aussage gleich(0,1) nicht logisch folgt, liefert SWI-Prolog

```
?- gleich(0,1).
true
```

Wird der occur check mittels

```
?- set_prolog_flag(occurs_check, true).
true.
```

aktiviert, erhält man die korrekte Antwort

```
?- gleich(0,1).
false.
```

In der Theorie der Logikprogrammierung stellt der occur check eine wesentliche Komponente dar (vgl. Theorem 3.34).

Beispiel. Die natürlichen Zahlen können durch folgendes Logikprogramm definiert werden:

```
nat(zero).
nat(s(X)) :- nat(X).
```

Welche Terme gehören zu nat?

```
?- nat(X).
X = zero ;
X = s(zero) ;
X = s(s(zero)) ;
X = s(s(s(zero))) ;
  ⋮
```

Hier kommt das Funktionszeichen s vor. In Prolog können Funktionszeichen nur zur Bildung von Termen verwendet werden, und Funktionen werden nicht direkt berechnet. Funktionen können (hier) nicht durch Gleichungen definiert werden, sondern müssen durch ihren Graphen, d. h. als Relation, angegeben werden.
Die Addition $x + y = z$ muss somit durch eine dreistellige Relation add(X, Y, Z) definiert werden:

```
add(X, zero, X).
add(X, s(Y), s(Z)) :- add(X, Y, Z).
```

Im Fall der Addition $2 + 1$:

```
?- add(s(s(zero)), s(zero), Z).
Z = s(s(s(zero)))
```

Als Vorteil der Logikprogrammierung kann man ansehen, dass mit der *Spezifikation* eines Problems in Form von Regeln und Fakten schon ein Programm vorliegt. Der Benutzer erhält dann die richtigen Antworten auf Anfragen an das Programm.
Ein zentraler Gegenstand dieser Vorlesung ist die Beantwortung der Fragen, *wie* diese Antworten erzeugt werden, *warum* sie korrekt sind, und *ob* alle möglichen Antworten erzeugt werden können. Dazu behandeln wir die sogenannte SLD-Resolution, welche die theoretische Grundlage der Logikprogrammierung im engeren Sinne darstellt. Wir behandeln zunächst aussagenlogische Resolution (Abschnitt 2), und verallgemeinern diese dann zur quantorenlogischen Resolution (Abschnitt 3). Daran schließt sich die Darstellung der SLD-Resolution (Abschnitt 4) als eingeschränkter Form der quantorenlogischen Resolution an. Schließlich beweisen wir Korrektheit und Vollständigkeit der SLD-Resolution (Abschnitt 5).

2 Aussagenlogische Resolution

2.1 Syntax und Semantik der Aussagenlogik

Definition 2.1 Das *Alphabet der Sprache der Aussagenlogik* besteht aus folgenden Zeichen: *Alphabet*

(i) *Aussagesymbole*: $p_0, p_1, p_2, \ldots$, wobei AS die Menge der Aussagesymbole bezeichne.

(ii) *Konnektive*: $\neg$ (Negation), $\wedge$ (Konjunktion), $\vee$ (Disjunktion) und $\rightarrow$ (Implikation)

(iii) *Klammern*: (,)

Definition 2.2 Die (*aussagenlogischen*) *Formeln* über $\mathrm{AS} = \{p_0, p_1, p_2, \ldots\}$ sind wie folgt definiert: *Formeln*

(i) Jedes Aussagesymbol in AS ist eine Formel.

(ii) Wenn A eine Formel ist, dann auch $\neg A$.

(iii) Wenn A und B Formeln sind, dann auch $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$.

Die *Sprache der Aussagenlogik* ist die Menge aller (aussagenlogischen) Formeln. Diese Menge bezeichnen wir auch mit FORMEL. *Sprache der Aussagenlogik*

Die Aussagesymbole werden auch als *atomare Formeln*, kurz: *Atome*, bezeichnet. Nichtatomare Formeln heißen *komplex*. *atomar/komplex*

Beispiel. Der Ausdruck $(((p_0 \wedge \neg p_1) \vee p_2) \rightarrow p_3)$ ist eine aussagenlogische Formel.

Wir fassen die Klauseln (i)-(iii) als Regeln zur Erzeugung von Formeln auf, so dass Formeln genau die mit diesen Regeln erzeugbaren Ausdrücke sind. Wir verzichten daher auf die weitere Bedingung, dass nichts sonst eine Formel sei, oder dass die Menge der Formeln gemäß (i)-(iii) die kleinste derartige Menge sei. (Wir werden das so auch bei anderen Definitionen entsprechender Form handhaben.)

Bemerkungen. (i) Die Sprache der Aussagenlogik ist unsere *Objektsprache*. In ihr kommen ausschließlich Zeichen des Alphabets gemäß Definition 2.1 vor. Wir verwenden $A, B, C, \ldots$ als *metasprachliche Variablen* für in der Objektsprache ausgedrückte Formeln.

(ii) Wir werden nur metasprachliche Variablen (kurz: *Metavariablen*) für Objektzeichen verwenden, aber keine Objektzeichen. Wir reden nicht über konkrete Formeln, sondern nur über Formeln von bestimmter Form.

(iii) Insbesondere werden wir auch Metavariablen (statt Aussagesymbolen) für Atome verwenden. Falls durch den Kontext nicht klar, sagen wir z. B. „sei A atomar“, um auszudrücken, dass in der Formel A keine Konnektive vorkommen.

Bemerkung. Regeln zur *Klammerersparnis*: *Klammerersparnis*

(i) Außenklammern können weggelassen werden.

(ii) Für die *Bindungsstärke* der Konnektive gilt wie üblich, dass $\neg$ am stärksten bindet, und $\wedge, \vee$ stärker binden als $\rightarrow$. *Bindungsstärke*

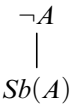
(iii) Wir verwenden Linksklammerung.

Beispiel. Die Formel mit Klammerersparnis $A \wedge B \wedge C \wedge D \to A \vee B \vee C \vee D$ steht für die Formel $((((A \wedge B) \wedge C) \wedge D) \to (((A \vee B) \vee C) \vee D))$.

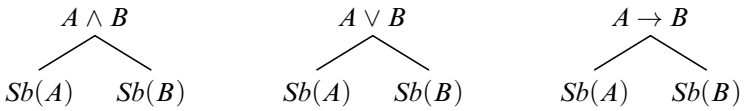
Definition 2.3 Die Erzeugung von Formeln kann als *Strukturbaum* dargestellt werden. Dieser ist rekursiv definiert wie folgt: *Strukturbaum*

(i) Der Strukturbaum $Sb(A)$ einer atomaren Formel A ist der Knoten A.

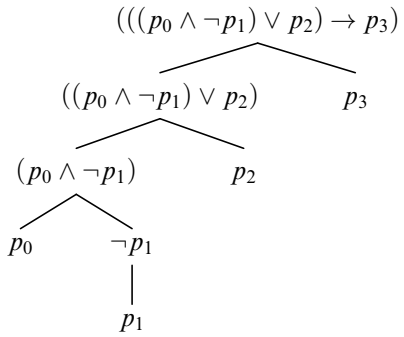
(ii) Der Strukturbaum $Sb(\neg A)$ einer negierten Formel $\neg A$ ist



(iii) Die Strukturbäume $Sb(A \wedge B)$, $Sb(A \vee B)$ und $Sb(A \to B)$ von Formeln $A \wedge B$, $A \vee B$ und $A \to B$ sind



Beispiel. Der Strukturbaum der Formel $(((p_0 \wedge \neg p_1) \vee p_2) \to p_3)$ ist:



Definition 2.4 (i) *Teilformeln* einer Formel A sind alle in $Sb(A)$ vorkommenden Formeln (einschließlich A). *Teilformel*

(ii) *Echte Teilformeln* einer Formel A sind alle Teilformeln von A mit Ausnahme von A selbst. *echte Teilformel*

(iii) *Unmittelbare Teilformeln* einer Formel A sind die in $Sb(A)$ unmittelbar auf A folgenden Knoten. *unmittelbare Teilformel*

Beispiel. Für $(((p_0 \wedge \neg p_1) \vee p_2) \to p_3)$:

– unmittelbare Teilformeln: $((p_0 \wedge \neg p_1) \vee p_2)$ und p_3;

– echte Teilformeln: $((p_0 \wedge \neg p_1) \vee p_2)$, $(p_0 \wedge \neg p_1)$, p_0, $\neg p_1$, p_1, p_2, p_3;

– Teilformeln: $(((p_0 \wedge \neg p_1) \vee p_2) \to p_3)$, $((p_0 \wedge \neg p_1) \vee p_2)$, $(p_0 \wedge \neg p_1)$, p_0, $\neg p_1$, p_1, p_2, p_3.

Soviel zur Syntax. Nun behandeln wir noch die Semantik.

In der Semantik werden zunächst den Aussagesymbolen Wahrheitswerte zugeordnet

(durch eine Bewertung $\mathcal{I}$), und die Bedeutung der logischen Konstanten $\neg$, $\wedge$, $\vee$, $\rightarrow$ wird durch Funktionen von Wahrheitswerten festgelegt (z. B. durch elementare Wahrheitstafeln).

Definition 2.5 Eine *Bewertung* $\mathcal{I}$ ist eine Funktion, die jedem Aussagesymbol aus AS einen der Wahrheitswerte w oder f zuordnet, d. h. $\mathcal{I} : \mathrm{AS} \to \{\mathrm{w}, \mathrm{f}\}$. Für $A \in \mathrm{AS}$ heißt A *wahr unter* $\mathcal{I}$, falls $\mathcal{I}(A) = \mathrm{w}$, und *falsch unter* $\mathcal{I}$, falls $\mathcal{I}(A) = \mathrm{f}$. *Bewertung*

Definition 2.6 Durch eine Funktion $[\![\ ]\!]^{\mathcal{I}} : \textsc{Formel} \to \{\mathrm{w}, \mathrm{f}\}$ wird jeder Formel über AS ein Wahrheitswert w oder f zugeordnet. Der *Wahrheitswert* $[\![A]\!]^{\mathcal{I}}$ einer Formel A unter einer Bewertung $\mathcal{I}$ ist wie folgt über dem Aufbau von Formeln definiert: *Wahrheitswert einer Formel*

$$\text{Für } A \in \mathrm{AS}\text{: } [\![A]\!]^{\mathcal{I}} := \begin{cases} \mathrm{w} & \text{falls } \mathcal{I}(A) = \mathrm{w}, \\ \mathrm{f} & \text{sonst.} \end{cases}$$

$$[\![\neg A]\!]^{\mathcal{I}} := \begin{cases} \mathrm{w} & \text{falls } [\![A]\!]^{\mathcal{I}} = \mathrm{f}, \\ \mathrm{f} & \text{sonst.} \end{cases}$$

$$[\![A \wedge B]\!]^{\mathcal{I}} := \begin{cases} \mathrm{w} & \text{falls } [\![A]\!]^{\mathcal{I}} = \mathrm{w} \text{ und } [\![B]\!]^{\mathcal{I}} = \mathrm{w}, \\ \mathrm{f} & \text{sonst.} \end{cases}$$

$$[\![A \vee B]\!]^{\mathcal{I}} := \begin{cases} \mathrm{w} & \text{falls } [\![A]\!]^{\mathcal{I}} = \mathrm{w} \text{ oder } [\![B]\!]^{\mathcal{I}} = \mathrm{w}, \\ \mathrm{f} & \text{sonst.} \end{cases}$$

$$[\![A \rightarrow B]\!]^{\mathcal{I}} := \begin{cases} \mathrm{w} & \text{falls } [\![A]\!]^{\mathcal{I}} = \mathrm{f} \text{ oder } [\![B]\!]^{\mathcal{I}} = \mathrm{w}, \\ \mathrm{f} & \text{sonst.} \end{cases}$$

Bemerkungen. (i) Statt $[\![A]\!]^{\mathcal{I}} = \mathrm{w}$ schreibt man auch $\mathcal{I} \vDash A$; statt $[\![A]\!]^{\mathcal{I}} = \mathrm{f}$ entsprechend $\mathcal{I} \nvDash A$.

(ii) Die Wahrheitswertabhängigkeit komplexer Formeln kann auch durch Wahrheitstafeln angegeben werden:

Negation

A	$\neg A$
w	f
f	w

Konjunktion

A	B	$A \wedge B$
w	w	w
w	f	f
f	w	f
f	f	f

Disjunktion

A	B	$A \vee B$
w	w	w
w	f	w
f	w	w
f	f	f

Implikation

A	B	$A \rightarrow B$
w	w	w
w	f	f
f	w	w
f	f	w

Definition 2.7 (i) A heißt *allgemeingültig* (oder *tautologisch*), falls A unter allen Bewertungen wahr ist. Schreibweise: $\vDash A$. *allgemeingültig*

(ii) A heißt *erfüllbar* (oder *konsistent*), falls A unter mindestens einer Bewertung wahr ist. *erfüllbar*

(iii) A heißt *unerfüllbar* (oder *inkonsistent* oder *kontradiktorisch*), falls A unter keiner Bewertung wahr ist. *unerfüllbar*

(iv) A heißt *kontingent*, falls A weder allgemeingültig noch unerfüllbar ist. *kontingent*

Definition 2.8 (i) Erfüllt eine Bewertung $\mathcal{I}$ eine Formel A, so heißt $\mathcal{I}$ *Modell* von A. *Modell*
(Man sagt dann auch, dass A die Bewertung $\mathcal{I}$ als Modell hat.)

(ii) Sei Γ eine Menge von Formeln und $\mathcal{I}$ eine Bewertung. Dann heißt $\mathcal{I}$ *Modell* von Γ, falls $\mathcal{I}$ ein Modell aller Formeln $A \in \Gamma$ ist.

Definition 2.9 (i) Sei Γ eine Menge von Formeln und A eine Formel. Dann ist die Relation der *(aussagen-)logischen Folgerung* wie folgt definiert: *logische Folgerung*

$$\Gamma \models A \ :\Longleftrightarrow \text{ Für jede Bewertung } \mathcal{I}\text{: Wenn } \mathcal{I} \models \Gamma, \text{ dann } \mathcal{I} \models A.$$

Man sagt auch "Γ impliziert logisch A" oder einfach "Γ impliziert A".

Die Formeln in Γ werden auch als *Prämissen* oder *Annahmen* bezeichnet, die Formel A als *Konklusion*.

Für endliche Prämissenmengen $\Gamma = \{A_1, \ldots, A_n\}$ und Konklusionen A notieren wir die Folgerungsrelation wie folgt: $A_1, \ldots, A_n \models A$.

(ii) Zwei Formeln A und B heißen *logisch äquivalent*, falls $A \models B$ und $B \models A$. *logisch äquivalent*

Notation: $A \dashv\models B$.

Beispiele. Überprüfung der logischen Folgerung unter Verwendung von Wahrheitstafeln:

(i) Gilt die Folgerungsbehauptung $A \vee B, \neg A \models B$?

	A	B	$A \vee B$	$\neg A$	$\models$	B
	w	w	w	f		w
	w	f	w	f		f
→	f	w	w	w	✓	w
	f	f	f	w		f

Es gibt nur eine Bewertung, unter der alle Prämissen wahr sind (dritte Zeile). Unter dieser Bewertung ist auch die Konklusion B wahr. Die Konklusion B ist also unter allen Bewertungen wahr, unter denen auch die Prämissen $A \vee B, \neg A$ wahr sind, es gilt also $A \vee B, \neg A \models B$.

(ii) Gilt die Folgerungsbehauptung $A \vee B, A \models B$?

	A	B	$A \vee B$	A	$\models$	B
→	w	w	w	w	✓	w
→	w	f	w	w	✗	f
	f	w	w	f		w
	f	f	f	f		f

Es gibt zwei Bewertungen, unter denen alle Prämissen wahr sind (erste und zweite Zeile). Allerdings ist die Konklusion B unter der zweiten Bewertung falsch. Es ist also $A \vee B, A \not\models B$, d. h., $A \vee B, A \models B$ gilt nicht.

2.2 Der Resolutionskalkül

Der Nachweis von logischen Folgerungen oder der Allgemeingültigkeit von Formeln kann unter Verwendung von Resolutionsverfahren erbracht werden. Wir führen zunächst den Resolutionskalkül ein, und beweisen dessen semantische Korrektheit. Anschließend

behandeln wir Resolutionswiderlegungen. Die Grundidee besteht darin, die Allgemeingültigkeit einer Formel A nachzuweisen, indem mittels Resolution gezeigt wird, dass $\neg A$ unerfüllbar ist. Für Resolutionswiderlegungen besteht der Resolutionskalkül aus nur einer Regel, und ist deshalb einfach zu implementieren. Das Verfahren setzt jedoch Klauselmengen voraus, was für Formeln bedeutet, dass diese erst in konjunktive Normalform überführt werden müssen.

Definition 2.10 (i) Eine *Klausel* ist eine Sequenz *Klausel*

$$A_1, \ldots, A_n \vdash B_1, \ldots, B_m$$

wobei $A_1, \ldots, A_n, B_1, \ldots, B_m$ atomare Formeln sind.

(ii) Links vom *Sequenzenzeichen* $\vdash$ steht das *Antezedens*, rechts davon das *Sukzedens*.

Definition 2.11 (i) Ein *Literal* ist ein Atom A (*positives Literal*) oder dessen Negation $\neg A$ (*negatives Literal*). *Literal*

(ii) Zwei Literale heißen *komplementär*, wenn eines die Negation des anderen ist. *komplementär*

Definition 2.12 Eine Formel ist in *konjunktiver Normalform* (kurz: KNF), falls sie die Form einer Konjunktion von Disjunktionen von Literalen hat. *konjunktive Normalform*

Bemerkungen. (i) Die Sequenz $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ steht für die Disjunktion $\neg A_1 \vee \ldots \vee \neg A_n \vee B_1 \vee \ldots \vee B_m$ von Literalen.

(ii) Antezedens und Sukzedens einer Klausel werden als Mengen aufgefasst; d. h., Multiplizität und Anordnung der Listenelemente spielen keine Rolle.

(iii) Eine Klausel wird auch als Menge von Literalen aufgefasst.

(iv) Metasprachliche Variablen für endliche Mengen von Atomen sind X, Y, Z, ggf. mit Indizes; für Klauseln: $S, S_1, S_2, \ldots$; und für Mengen von Klauseln: $\Gamma, \Delta, \ldots$.

(v) Da Sequenzen Formeln entsprechen, können wir die für Formeln eingeführte logische Folgerung auch für Klauseln notieren: $\Gamma \models S$.

(vi) $X, A \vdash Y, B$ steht für $X \cup \{A\} \vdash Y \cup \{B\}$, wobei A und B hier auch Element von X bzw. Y sein können.

(vii) Die *leere Klausel* enthält keine Atome. Notation: $\vdash$ bzw. $\square$.

Die leere Klausel ist unerfüllbar.

Definition 2.13 Für Atome A und möglicherweise leere Mengen von Atomen $X, Y, \ldots$ ist der *aussagenlogische Resolutionskalkül* gegeben durch das Axiom (bzw. Axiomenschema) *Resolutionskalkül*

$$(\mathrm{Ax})\ \frac{}{X, A \vdash A, Y}$$

und die *aussagenlogische Resolutionsregel*: *Resolutionsregel*

$$\frac{X_1 \vdash Y_1, A \qquad A, X_2 \vdash Y_2}{X_1, X_2 \vdash Y_1, Y_2}\ (\mathcal{R})$$

Die Konklusion von $(\mathcal{R})$ heißt *Resolvente* (bzgl. A) der Prämissen. *Resolvente*

Definition 2.14 *Ableitungen* im aussagenlogischen Resolutionskalkül sind induktiv definiert wie folgt (wobei wir $\dfrac{\mathscr{D}}{S}$ für die mit der Klausel S endende Ableitung $\mathscr{D}$ schreiben): *Ableitung*

(i) $X \vdash Y$ ist eine Ableitung (von $X \vdash Y$ aus $X \vdash Y$).

(ii) $(\mathrm{Ax})\ \dfrac{}{X, A \vdash A, Y}$ ist eine Ableitung.

(iii) Sind $\dfrac{\mathscr{D}_1}{X_1 \vdash Y_1, A}$ und $\dfrac{\mathscr{D}_2}{A, X_2 \vdash Y_2}$ Ableitungen, dann ist auch

$$\dfrac{\dfrac{\mathscr{D}_1}{X_1 \vdash Y_1, A} \qquad \dfrac{\mathscr{D}_2}{A, X_2 \vdash Y_2}}{X_1, X_2 \vdash Y_1, Y_2}\,(\mathcal{R})$$

eine Ableitung.

Definition 2.15 Die Menge der *Annahmen* (oder *Hypothesen*) einer Ableitung ist rekursiv definiert durch: *Annahmen*

(i) $Hyp(X \vdash Y) := \{X \vdash Y\}$.

(ii) $Hyp\Big((\mathrm{Ax})\ \dfrac{}{X, A \vdash A, Y}\Big) := \emptyset$.

(iii) $Hyp\left(\dfrac{\dfrac{\mathscr{D}_1}{X_1 \vdash Y_1, A} \qquad \dfrac{\mathscr{D}_2}{A, X_2 \vdash Y_2}}{X_1, X_2 \vdash Y_1, Y_2}\,(\mathcal{R})\right) := Hyp\Big(\dfrac{\mathscr{D}_1}{X_1 \vdash Y_1, A}\Big) \cup Hyp\Big(\dfrac{\mathscr{D}_2}{A, X_2 \vdash Y_2}\Big)$.

Bemerkung. Eine Ableitung ist also ein (nach oben verzweigender) binärer Baum, dessen Blätter Annahmen sind. Im Fall des Axioms (Ax) ist dessen leere Prämisse als leere Annahme ein Blatt.

Definition 2.16 Es gilt $\Gamma \vdash_{\mathrm{Res}} S$ genau dann, wenn es eine mit der Klausel S endende Ableitung $\dfrac{\mathscr{D}}{S}$ aus Annahmen $Hyp\Big(\dfrac{\mathscr{D}}{S}\Big) \subseteq \Gamma$ gibt. $\Gamma \vdash_{\mathrm{Res}} S$

$\Gamma \vdash_{\mathrm{Res}} S$ bedeutet also, dass die Klausel S im aussagenlogischen Resolutionskalkül aus der Menge von Klauseln Γ *ableitbar* ist. Es ist $\vdash_{\mathrm{Res}}$ die *Ableitbarkeitsrelation* (sie darf nicht mit dem Sequenzenzeichen $\vdash$ verwechselt werden). *ableitbar*

Bemerkungen. (i) Das Axiom erzeugt *tautologische Klauseln*. Eine Klausel *tautologische Klausel*

$$\neg A_1 \vee \ldots \vee \neg A_n \vee B_1 \vee \ldots \vee B_m$$

ist genau dann tautologisch, wenn sie zwei komplementäre Literale enthält (d. h., wenn $A_i = B_j$ für mindestens ein Paar i, j). Denn dann enthält die Klausel die Teilformel $\neg A \vee A$, was wiederum genau dann der Fall ist, wenn die Klausel ein Axiom ist.

(ii) Mithilfe des Axioms können Klauseln abgeschwächt (verdünnt) werden. Die Klausel $A, X_2 \vdash Y_2$ kann mit dem Axiom $X_1, A \vdash A, Y_1$ über

$$(\mathrm{Ax})\ \dfrac{\dfrac{}{X_1, A \vdash A, Y_1} \qquad A, X_2 \vdash Y_2}{X_1, A, X_2 \vdash Y_1, Y_2}\,(\mathcal{R})$$

zu $X_1, A, X_2 \vdash Y_1, Y_2$ verdünnt werden. Entsprechend können Klauseln der Form $X_2 \vdash A, Y_2$ zu $X_1, X_2 \vdash A, Y_1, Y_2$ verdünnt werden.

Theorem 2.17 (Korrektheit) *Der aussagenlogische Resolutionskalkül ist semantisch korrekt, d. h., es gilt: Wenn* $\Gamma \vdash_{\text{Res}} S$*, dann* $\Gamma \models S$.

Beweis. Angenommen $\Gamma \vdash_{\text{Res}} S$. Dann gibt es eine Ableitung $\begin{smallmatrix}\mathscr{D}\\ S\end{smallmatrix}$ mit $Hyp\left(\begin{smallmatrix}\mathscr{D}\\ S\end{smallmatrix}\right) \subseteq \Gamma$.

Wir zeigen per Induktion über dem Aufbau von $\begin{smallmatrix}\mathscr{D}\\ S\end{smallmatrix}$, dass für jede Bewertung $\mathcal{I}$ gilt:

$$\text{Wenn } \mathcal{I} \text{ ein Modell von } Hyp\left(\begin{smallmatrix}\mathscr{D}\\ S\end{smallmatrix}\right) \text{ ist, dann ist } \mathcal{I} \text{ auch ein Modell von } S. \qquad (*)$$

Induktionsanfang:

Fall 1: $\begin{smallmatrix}\mathscr{D}\\ S\end{smallmatrix}$ habe die Form S.

Dann gilt $Hyp\left(\begin{smallmatrix}\mathscr{D}\\ S\end{smallmatrix}\right) = Hyp(S) = \{S\}$, so dass $(*)$ trivialerweise für jede Bewertung $\mathcal{I}$ gilt.

Fall 2: $\begin{smallmatrix}\mathscr{D}\\ S\end{smallmatrix}$ habe die Form $(\text{Ax})\ \dfrac{}{X, A \vdash A, Y}$, wobei $S = (X, A \vdash A, Y)$.

Da S die Tautologie $\neg A \vee A$ enthält, ist jede Bewertung $\mathcal{I}$ ein Modell von S. Somit gilt $(*)$.

Induktionsvoraussetzung: Für Ableitungen $\begin{smallmatrix}\mathscr{D}_1\\ X_1 \vdash Y_1, A\end{smallmatrix}$ und $\begin{smallmatrix}\mathscr{D}_2\\ A, X_2 \vdash Y_2\end{smallmatrix}$ gelte $(*)$.

Induktionsschritt: $\begin{smallmatrix}\mathscr{D}\\ S\end{smallmatrix}$ habe die Form

$$\frac{\begin{smallmatrix}\mathscr{D}_1\\ X_1 \vdash Y_1, A\end{smallmatrix} \qquad \begin{smallmatrix}\mathscr{D}_2\\ A, X_2 \vdash Y_2\end{smallmatrix}}{X_1, X_2 \vdash Y_1, Y_2}\,(\mathcal{R})$$

Wir betrachten eine beliebige Bewertung $\mathcal{I}$, die ein Modell von

$$Hyp\left(\begin{smallmatrix}\mathscr{D}\\ S\end{smallmatrix}\right) = Hyp\left(\begin{smallmatrix}\mathscr{D}_1\\ X_1 \vdash Y_1, A\end{smallmatrix}\right) \cup Hyp\left(\begin{smallmatrix}\mathscr{D}_2\\ A, X_2 \vdash Y_2\end{smallmatrix}\right)$$

ist. Nach Induktionsvoraussetzung ist $\mathcal{I}$ dann ein Modell sowohl von $X_1 \vdash Y_1, A$ als auch von $A, X_2 \vdash Y_2$, da

$$Hyp\left(\begin{smallmatrix}\mathscr{D}_1\\ X_1 \vdash Y_1, A\end{smallmatrix}\right) \subseteq Hyp\left(\begin{smallmatrix}\mathscr{D}\\ S\end{smallmatrix}\right) \quad \text{und} \quad Hyp\left(\begin{smallmatrix}\mathscr{D}_2\\ A, X_2 \vdash Y_2\end{smallmatrix}\right) \subseteq Hyp\left(\begin{smallmatrix}\mathscr{D}\\ S\end{smallmatrix}\right)$$

Wir müssen zeigen, dass $\mathcal{I}$ auch ein Modell von $S = (X_1, X_2 \vdash Y_1, Y_2)$ ist. Dazu können wir o.B.d.A. annehmen, dass $\mathcal{I}$ ein Modell aller in X_1 und X_2 vorkommenden Atome ist. Andernfalls erfüllt $\mathcal{I}$ die Klausel S unmittelbar.
Für die Bewertung $\mathcal{I}(A)$ sind zwei Fälle zu unterscheiden:

Fall 1: $\mathcal{I}(A) = \text{f}$. Da $\mathcal{I}$ nach Induktionsvoraussetzung ein Modell von $X_1 \vdash Y_1, A$ ist, muss $\mathcal{I}(B) = \text{w}$ für mindestens ein Atom B in Y_1 gelten.

Dann ist $\mathcal{I}$ auch ein Modell von S.

Fall 2: $\mathcal{I}(A) = \text{w}$. Da $\mathcal{I}$ nach Induktionsvoraussetzung ein Modell von $A, X_2 \vdash Y_2$ ist, muss $\mathcal{I}(B) = \text{w}$ für mindestens ein Atom B in Y_2 gelten.

Dann ist $\mathcal{I}$ ebenfalls ein Modell von S.

Die Bewertung $\mathcal{I}$ ist also ein Modell von S. Damit ist $(*)$ gezeigt. Mit $Hyp\binom{\mathscr{D}}{S} \subseteq \Gamma$ folgt $\Gamma \models S$. QED

Im Folgenden behandeln wir den Resolutionskalkül als *Widerlegungskalkül*. Auf das Axiom kann dann verzichtet werden, obgleich die jeweils zu widerlegenden Klauselmengen auch Klauseln von der Form des Axioms enthalten dürfen. Der Resolutionskalkül besteht dann allein aus der Resolutionsregel

$$\frac{X_1 \vdash Y_1, A \qquad A, X_2 \vdash Y_2}{X_1, X_2 \vdash Y_1, Y_2}\,(\mathcal{R})$$

wobei wir im Folgenden $A \notin Y_1$ und $A \notin X_2$ annehmen.

Definition 2.18 Eine *Resolutionswiderlegung* einer Menge Γ von Klauseln ist eine Ableitung der leeren Klausel $\vdash$ (bzw. $\square$) aus Klauseln in Γ. *Resolutionswiderlegung*

Da die Resolutionsregel semantisch korrekt ist, muss für jede Bewertung mindestens eine der Klauseln in Γ falsch sein, falls eine Resolutionswiderlegung von Γ vorliegt. Mit anderen Worten: Eine Resolutionswiderlegung von Γ bedeutet $\Gamma \vdash_{\text{Res}} \square$ und ist aufgrund semantischer Korrektheit der Resolutionsregel ein Beweis für $\Gamma \models \square$, d. h. für die Unerfüllbarkeit von Γ.
Betrachtet man eine Formel A, so muss diese zunächst in eine *Klauselmenge* $Kl(A)$ übersetzt werden, damit Resolution anwendbar wird.

Definition 2.19 Ein *Resolutionsbeweis* für eine Formel A ist eine Resolutionswiderlegung von $Kl(\neg A)$. *Resolutionsbeweis*

Aus einem Resolutionsbeweis für A folgt $\models A$. Denn es gilt allgemein

$$\begin{aligned}\Gamma \models A &\iff A \text{ ist wahr in allen Modellen von } \Gamma\\ &\iff \Gamma \cup \{\neg A\} \text{ hat kein Modell}\\ &\iff \Gamma \cup \{\neg A\} \text{ ist unerfüllbar}\end{aligned}$$

und für $\Gamma = \emptyset$ somit insbesondere

$$\models A \iff \neg A \text{ ist unerfüllbar.}$$

Aufgrund semantischer Korrektheit der Resolutionsregel folgt aus einer Ableitung der unerfüllbaren leeren Klausel $\square$ die Unerfüllbarkeit von $Kl(\neg A)$, und somit auch die Unerfüllbarkeit von $\neg A$.
Allerdings muss für einen Resolutionsbeweis für eine Formel A zunächst die konjunktive Normalform von $\neg A$ gebildet werden, um die erforderliche Klauselmenge $Kl(\neg A)$ angeben zu können. Zur Bildung der KNF wird kein semantisches Verfahren (z. B. Wahrheitstafeln) verwendet, da dieses die Formel entscheidet, und somit das Resolutionsverfahren überflüssig macht. Stattdessen verwenden wir das folgende

Verfahren zur Erzeugung einer KNF

(1) Eliminiere logische Zeichen, die von $\neg, \wedge, \vee$ verschieden sind; d. h., eliminiere Vorkommen von $\rightarrow$ durch folgende Umformung:

$$(A \rightarrow B) \rightsquigarrow (\neg A \vee B)$$

(2) Ziehe Negationen nach innen (De Morgan) und beseitige doppelte Negationen:

$$\neg(A \wedge B) \rightsquigarrow (\neg A \vee \neg B)$$
$$\neg(A \vee B) \rightsquigarrow (\neg A \wedge \neg B)$$
$$\neg\neg A \rightsquigarrow A$$

(3) Ziehe $\wedge$ nach außen mit Hilfe der Distributivität:

$$(A \wedge B) \vee C \rightsquigarrow (A \vee C) \wedge (B \vee C)$$

Assoziativität von $\wedge$ und $\vee$ verwenden wir implizit.

Jedem Konjunktionsglied der resultierenden KNF entspricht eine Klausel. Der KNF

$$(\neg A_{11} \vee \ldots \vee \neg A_{1k_1} \vee B_{11} \vee \ldots \vee B_{1l_1}) \wedge \ldots \wedge (\neg A_{j1} \vee \ldots \vee \neg A_{jk_j} \vee B_{j1} \vee \ldots \vee B_{jl_j})$$

einer Formel A entspricht somit eine *Klauselmenge* *Klauselmenge*

$$Kl(A) = \{A_{11}, \ldots, A_{1k_1} \vdash B_{11}, \ldots, B_{1l_1} \; ; \; \ldots \; ; \; A_{j1}, \ldots, A_{jk_j} \vdash B_{j1}, \ldots, B_{jl_j}\}$$

die per Resolution behandelbar ist. (Wir grenzen die Elemente von Klauselmengen durch ';' ab.)

Bemerkung. In Disjunktionen von Literalen $A_1 \vee \ldots \vee A_n$ können gleiche Literale mehrfach vorkommen. Sind A_i und A_j (für $1 \leq i < j \leq n$) zwei gleiche Literale, dann bezeichnet man die Disjunktion

$$A_1 \vee \ldots \vee A_i \vee A_{i+1} \vee \ldots \vee A_{j-1} \vee A_{j+1} \vee \ldots \vee A_n$$

in der das Vorkommen A_j beseitigt wurde, als *Faktor* der ursprünglichen Disjunktion. *Faktor*
Falls keine gleichen Literale mehrfach vorkommen, heißt die Disjunktion von Literalen *faktorfrei*. *faktorfrei*
Da Antezedens und Sukzedens einer Klausel als Mengen aufgefasst werden, entspricht z. B. der Formel $\neg A \vee \neg A \vee B \vee B$ (für Atome A, B) die Klausel $A \vdash B$, der die faktorfreie Formel $\neg A \vee B$ entspricht.

Theorem 2.20 (Import-Export)
Es gilt $\Gamma \models A \to B$ *genau dann, wenn* $\Gamma \cup \{A\} \models B$.

Beweis. Sei $\mathcal{I}$ eine beliebige Bewertung.

$$\begin{aligned}
\Gamma \cup \{A\} \models B &\iff \text{Wenn } \mathcal{I} \models \Gamma \cup \{A\}, \text{ dann } \mathcal{I} \models B \\
&\iff \mathcal{I} \not\models \Gamma \cup \{A\} \text{ oder } \mathcal{I} \models B \\
&\iff \mathcal{I} \not\models \Gamma \text{ oder } \mathcal{I} \not\models A \text{ oder } \mathcal{I} \models B \\
&\iff \mathcal{I} \not\models \Gamma \text{ oder (wenn } \mathcal{I} \models A, \text{ dann } \mathcal{I} \models B) \\
&\iff \mathcal{I} \not\models \Gamma \text{ oder } \mathcal{I} \models A \to B \\
&\iff \text{Wenn } \mathcal{I} \models \Gamma, \text{ dann } \mathcal{I} \models A \to B \\
&\iff \Gamma \models A \to B
\end{aligned}$$

QED

Korollar 2.21
Da $A_1 \to (A_2 \to (\cdots (A_n \to B)\cdots)) \dashv\vDash (A_1 \wedge A_2 \wedge \ldots \wedge A_n) \to B$, *gilt* $A_1, \ldots, A_n \vDash B$ *genau dann, wenn* $\vDash A_1 \wedge \ldots \wedge A_n \to B$.

Bemerkungen. (i) Statt $Kl(\neg(A \to B))$ kann auch $Kl(A) \cup Kl(\neg B)$ betrachtet werden.

(ii) Zur Behandlung von Folgerungsbehauptungen $A_1, \ldots, A_n \vDash B$ kann

$$Kl(\neg(A_1 \wedge \ldots \wedge A_n \to B)) \qquad \text{oder} \qquad \bigcup_{1 \le i \le n} Kl(A_i) \cup Kl(\neg B)$$

betrachtet werden.

Beispiele. Seien A, B, C Atome. (Wir verzichten auf die Angabe aller Zwischenschritte bei der Erzeugung der jeweiligen KNF.)

(i) $\vDash A \to A \vee B$

KNF von $\neg(A \to A \vee B)$:

$$\begin{aligned}\neg(A \to A \vee B) &\rightsquigarrow \neg(\neg A \vee A \vee B)\\ &\rightsquigarrow \neg\neg A \wedge \neg A \wedge \neg B\\ &\rightsquigarrow A \wedge \neg A \wedge \neg B\\ &\rightsquigarrow \{\vdash A\,;\ A \vdash\ ;\ B \vdash\} = Kl(\neg(A \to A \vee B))\end{aligned}$$

Resolutionswiderlegung: $\dfrac{\vdash A \qquad A \vdash}{\vdash}\,(\mathcal{R})$

Es wurde

$$Kl(\neg(A \to A \vee B)) \vdash_{\text{Res}} \square$$

gezeigt. Mit Korrektheit folgt

$$Kl(\neg(A \to A \vee B)) \vDash \square$$

Da $\square$ unerfüllbar ist, muss $Kl(\neg(A \to A \vee B))$ und damit $\neg(A \to A \vee B)$ unerfüllbar sein, d. h., $\vDash A \to A \vee B$.

(ii) $\vDash A \vee (B \vee C) \to (A \vee B) \vee C$

KNF von $\neg(A \vee (B \vee C) \to (A \vee B) \vee C)$:

$$\begin{aligned}\neg(A \vee (B \vee C) \to (A \vee B) \vee C) &\rightsquigarrow \neg(\neg(A \vee (B \vee C)) \vee ((A \vee B) \vee C))\\ &\rightsquigarrow (A \vee (B \vee C)) \wedge \neg((A \vee B) \vee C)\\ &\rightsquigarrow (A \vee B \vee C) \wedge \neg A \wedge \neg B \wedge \neg C\\ &\rightsquigarrow \{\vdash A, B, C\ ;\ A \vdash\ ;\ B \vdash\ ;\ C \vdash\}\end{aligned}$$

Resolutionswiderlegung:
$$\dfrac{\dfrac{\dfrac{\vdash A, B, C \qquad A \vdash}{\vdash B, C}\,(\mathcal{R}) \qquad B \vdash}{\vdash C}\,(\mathcal{R}) \qquad C \vdash}{\vdash}\,(\mathcal{R})$$

Aufgrund des Import-Export-Theorems können wir alternativ auch die Folgerungsbehauptung $A \vee (B \vee C) \vDash (A \vee B) \vee C$ betrachten:

KNF von $A \vee (B \vee C)$: $A \vee B \vee C \rightsquigarrow \{\vdash A, B, C\}$.

KNF von $\neg((A \vee B) \vee C)$: $\neg((A \vee B) \vee C) \rightsquigarrow \neg A \wedge \neg B \wedge \neg C \rightsquigarrow \{A \vdash\,; B \vdash\,; C \vdash\}$.

Man erhält also ebenfalls die Klauselmenge $\{\vdash A, B, C\ ;\ A \vdash\ ;\ A \vdash\ ;\ B \vdash\ ;\ C \vdash\}$ und die gezeigte Resolutionswiderlegung.

(iii) $\vDash (\neg A \to \neg B) \to (B \to A)$

KNF von $\neg((\neg A \to \neg B) \to (B \to A))$:

$$\begin{aligned}\neg((\neg A \to \neg B) \to (B \to A)) &\rightsquigarrow \neg((A \vee \neg B) \to (\neg B \vee A))\\ &\rightsquigarrow \neg(\neg(A \vee \neg B) \vee \neg B \vee A)\\ &\rightsquigarrow \neg((\neg A \wedge B) \vee \neg B \vee A)\\ &\rightsquigarrow \neg(\neg A \wedge B) \wedge B \wedge \neg A\\ &\rightsquigarrow (A \vee \neg B) \wedge B \wedge \neg A\\ &\rightsquigarrow \{B \vdash A\ ;\ \vdash B\ ;\ A \vdash\}\end{aligned}$$

Resolutionswiderlegung:
$$\dfrac{\dfrac{B \vdash A \qquad A \vdash}{\vdash B}\,(\mathcal{R}) \qquad B \vdash}{\vdash}\,(\mathcal{R})$$

Alternativ für $\neg A \to \neg B \vDash B \to A$:

KNF von $\neg A \to \neg B$: $A \vee \neg B \rightsquigarrow \{B \vdash A\}$,

KNF von $\neg(B \to A)$: $B \wedge \neg A \rightsquigarrow \{\vdash B\ ;\ A \vdash\}$,

Klauselmenge (wie oben): $\{B \vdash A\ ;\ \vdash B\ ;\ A \vdash\}$; mit der obigen Resolutionswiderlegung.

Bemerkungen. (i) Ein großer Teil des Aufwands geht hierbei in die Bestimmung der KNF.

(ii) Die KNF muss nicht logisch äquivalent zur Ausgangsformel sein. Für den Widerlegungskalkül genügt eine *erfüllbarkeitsäquivalente* KNF, d. h., es muss lediglich gelten: KNF erfüllbar genau dann, wenn Ausgangsformel erfüllbar.

(iii) Erfüllbarkeitsäquivalenz ist eine wesentlich schwächere Eigenschaft als logische Äquivalenz: Sei A eine kontingente Formel, dann gilt A erfüllbar gdw. $\neg A$ erfüllbar. Hingegen können A und $\neg A$ natürlich nicht logisch äquivalent sein.

(iv) Die Anzahl der Schritte zur Erzeugung einer *logisch äquivalenten* KNF oder einer disjunktiven Normalform (DNF) ist exponentiell in der Anzahl der Aussagevariablen.

(v) Für die Erzeugung einer *erfüllbarkeitsäquivalenten* KNF gibt es aber Verfahren mit polynomialer Laufzeit (siehe z. B. Leitsch, 1997, S. 19–21). Dies ist von Vorteil für das Resolutionsverfahren. (Für die Erzeugung einer *allgemeingültigkeitsäquivalenten* KNF gibt es kein Verfahren mit polynomialer Laufzeit.)

Für das Auffinden einer Resolutionswiderlegung geben wir nun ein Verfahren an (Theorem 2.23), das für jede gegebene endliche Klauselmenge Γ entscheidet, ob es eine Resolutionswiderlegung für Γ gibt oder nicht.

Definition 2.22 (i) Es ist $\mathcal{R}_A(S_1, S_2)$ die *Resolvente der Klauseln S_1 und S_2 bezüglich A*, falls eine solche existiert; andernfalls sei $\mathcal{R}_A(S_1, S_2)$ nicht definiert. $\mathcal{R}_A(S_1, S_2)$

(ii) Es ist $\mathcal{R}_A(\Gamma) := \{\mathcal{R}_A(S_1, S_2) \mid S_1, S_2 \in \Gamma\}$ die *Menge aller möglichen Resolutionsresultate bezüglich A*. $\mathcal{R}_A(\Gamma)$

(iii) Sei Γ_A die Menge aller Klauseln in Γ, in denen A vorkommt.

Dann ist $Res_A(\Gamma) := (\Gamma \setminus \Gamma_A) \cup \mathcal{R}_A(\Gamma_A)$. $Res_A(\Gamma)$

Theorem 2.23 *Gegeben sei das folgende Verfahren, das wir für Klauselmengen Γ angeben, in denen genau die Atome $A_1, \ldots, A_n$ vorkommen.*
Für i von 1 bis n:

(1) *Eliminiere tautologische Klauseln aus Γ. Die resultierende Klauselmenge sei Γ'.*

(2) *Bilde $Res_{A_i}(\Gamma')$.*

(3) *Setze $\Gamma := Res_{A_i}(\Gamma')$.*

Dann gilt: Das Verfahren liefert die Klauselmenge $\{\square\}$ genau dann, wenn es eine Resolutionswiderlegung für Γ gibt.

Beweis. Übungsaufgabe. (Gegebenenfalls unter Verwendung der im nächsten Abschnitt behandelten Vollständigkeitsresultate.) QED

Beispiel. Wir betrachten die Klauselmenge $\Gamma = \{A_2 \vdash A_1\ ;\ \vdash A_2\ ;\ A_1 \vdash\ ;\ A_1, A_2 \vdash A_2\}$.

(1) Eliminiere die tautologische Klausel $A_1, A_2 \vdash A_2$: $\Gamma' := \Gamma \setminus \{A_1, A_2 \vdash A_2\}$.

(2) $Res_{A_1}(\Gamma') = (\Gamma' \setminus \Gamma'_{A_1}) \cup \mathcal{R}_{A_1}(\Gamma'_{A_1}) = \{\vdash A_2\} \cup \{A_2 \vdash\}$

(3) $\Gamma := \{\vdash A_2\ ;\ A_2 \vdash\}$

(1) Γ enthält keine tautologischen Klauseln. $\Gamma' := \Gamma$.

(2) $Res_{A_2}(\Gamma') = (\Gamma' \setminus \Gamma'_{A_2}) \cup \mathcal{R}_{A_2}(\Gamma'_{A_2}) = \emptyset \cup \{\square\}$

(3) $\Gamma := \{\square\}$.

2.3 Vollständigkeit

Die Korrektheit des Resolutionskalküls (Theorem 2.17) garantiert, dass jede aus einer Klauselmenge Γ im Kalkül ableitbare Klausel S auch logisch aus Γ folgt: $\Gamma \vdash_{\text{Res}} S \implies \Gamma \models S$. Im Widerlegungskalkül konnten wir dann mit Korrektheit insbesondere von einer Resolutionswiderlegung von Γ auf die Unerfüllbarkeit von Γ schließen. Zudem gibt es nach Theorem 2.23 ein Verfahren, das die Existenz einer Resolutionswiderlegung für jede gegebene (aussagenlogische) Klauselmenge entscheidet.
Nun wollen wir zeigen, dass jede aus einer Klauselmenge Γ logisch folgende Klausel S im Resolutionskalkül aus Γ abgeleitet werden kann, d. h., wir wollen zeigen, dass der Resolutionskalkül auch semantisch vollständig ist: $\Gamma \models S \implies \Gamma \vdash_{\text{Res}} S$. Dazu beweisen wir zunächst die semantische Vollständigkeit des Widerlegungskalküls (Theorem 2.26), und zeigen dann, dass der Resolutionskalkül semantisch vollständig ist (Theorem 2.27).

Definition 2.24 (i) $Res(S_1, S_2, S_3)$ bedeute, dass S_3 eine Resolvente von S_1 und S_2 ist.

(ii) $Res(\Gamma) := \{S \mid Res(S_1, S_2, S) \text{ für } S_1, S_2 \in \Gamma\}$ ist die Menge aller Resolventen S von Klauseln $S_1, S_2 \in \Gamma$.

(iii) Die *Resolutionsstufe* $Rs^n(\Gamma)$ einer Klauselmenge Γ definieren wir durch: *Resolutionsstufe*

$$Rs^0(\Gamma) := \Gamma$$
$$Rs^{n+1}(\Gamma) := Rs^n(\Gamma) \cup Res(Rs^n(\Gamma))$$

(iv) Es ist

$$R(\Gamma) := \bigcup_{n \in \mathbb{N}} Rs^n(\Gamma)$$

der *Resolutionsabschluss* von Γ. *Resolutionsabschluss*

Beispiel. Sei $\Gamma = \{ \vdash A \; ; \; A \vdash B \; ; \; A, B \vdash \}$. Dann ist

$$
\begin{aligned}
Rs^0(\Gamma) &= \Gamma = \{ \vdash A \; ; \; A \vdash B \; ; \; A, B \vdash \} \\
Rs^1(\Gamma) &= Rs^0(\Gamma) \cup Res(Rs^0(\Gamma)) = \Gamma \cup Res(\Gamma) = \Gamma \cup Res(\{ \vdash A \; ; \; A \vdash B \; ; \; A, B \vdash \}) \\
&= \{ \vdash A \; ; \; A \vdash B \; ; \; A, B \vdash \} \cup \{ \vdash B \; ; \; B \vdash \; ; \; A \vdash \} \\
&= \{ \vdash A \; ; \; A \vdash B \; ; \; A, B \vdash \; ; \; \vdash B \; ; \; B \vdash \; ; \; A \vdash \} \\
Rs^2(\Gamma) &= Rs^1(\Gamma) \cup Res(Rs^1(\Gamma)) \\
&= Rs^1(\Gamma) \cup Res(\{ \vdash A \; ; \; A \vdash B \; ; \; A, B \vdash \; ; \; \vdash B \; ; \; B \vdash \; ; \; A \vdash \}) \\
&= Rs^1(\Gamma) \cup \{ \vdash B \; ; \; A \vdash \; ; \; B \vdash \; ; \; \square \} \\
&= \{ \vdash A \; ; \; A \vdash B \; ; \; A, B \vdash \; ; \; \vdash B \; ; \; B \vdash \; ; \; A \vdash \; ; \; \square \} \\
Rs^3(\Gamma) &= Rs^2(\Gamma) \quad \text{(da mit } \square \text{ keine neuen Resolventen erzeugt werden können).}
\end{aligned}
$$

Folglich ist der Resolutionsabschluss $R(\Gamma) = Rs^2(\Gamma)$.

Wir können uns im Folgenden auf die Betrachtung *endlicher* Klauselmengen beschränken, denn es gilt:

Theorem 2.25 (Endlichkeitssatz) *Wenn $\Gamma \models S$, dann gibt es eine endliche Menge $\Gamma' \subseteq \Gamma$ mit $\Gamma' \models S$. (Entsprechend gilt: Wenn Γ unerfüllbar ist, dann gibt es eine endliche unerfüllbare Menge $\Gamma' \subseteq \Gamma$.)*

Beweis. Den Endlichkeitssatz erhält man als Folgerung aus Korrektheit und Vollständigkeit aussagenlogischer Kalküle wie z. B. dem Kalkül des natürlichen Schließens, dessen Ableitungen stets endliche Mengen von Annahmen haben (siehe z. B. van Dalen, 2013). Für einen direkten Beweis siehe Goltz & Herre (1990). QED

Theorem 2.26 (Vollständigkeit Widerlegungskalkül) *Wenn die Klauselmenge Γ unerfüllbar ist, dann gibt es ein $n \in \mathbb{N}$ mit $\square \in Rs^n(\Gamma)$, und somit auch $\square \in R(\Gamma)$.*

Beweis. Die (endliche) Klauselmenge Γ sei unerfüllbar.

Da in Γ nur endlich viele Atome vorkommen und Multiplizität in Klauseln keine Rolle spielt, gibt es es ein k, so dass $Rs^k(\Gamma) = Rs^{k+1}(\Gamma)$. Das heißt, es gibt eine Resolutionsstufe k, ab der keine Resolventen mehr hinzukommen.

Wir nehmen an, dass $\Delta := Rs^k(\Gamma)$ die niedrigste derartige Resolutionsstufe ist.

Wir müssen zeigen: Δ unerfüllbar $\implies \square \in \Delta$. Dann gilt:

$$
\begin{aligned}
\Gamma \text{ unerfüllbar} &\implies \Delta \text{ unerfüllbar} && \text{(da } \Gamma \subseteq \Delta \text{, und Erweiterungen unerfüllbarer Mengen bleiben unerfüllbar)} \\
&\implies \square \in \Delta && \text{(dieser Schritt ist zu zeigen)} \\
&\implies \square \in Rs^k(\Gamma) && \text{(Definition von } \Delta\text{)} \\
&\implies \square \in R(\Gamma) && \text{(da } Rs^k(\Gamma) \subseteq R(\Gamma)\text{)}
\end{aligned}
$$

Wir sagen, dass eine Menge L von Literalen eine Klausel S der Form

$$A_1, \ldots, A_n \vdash B_1, \ldots, B_m$$

falsifiziert, falls

$$\{A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m\} \subseteq L.$$

Es gilt dann für jede Bewertung $\mathcal{I}$: $\mathcal{I} \models L \implies \mathcal{I} \not\models S$.

BEISPIEL. Die Menge von Literalen $L = \{A, \neg B\}$ falsifiziert die Klausel $A \vdash B$. Es ist $\mathcal{I} \vDash L$ nur für $\mathcal{I}(A) = \text{w}$ und $\mathcal{I}(B) = \text{f}$, und für diese Bewertung gilt $\mathcal{I} \nvDash \neg A \vee B$.

Die leere Klausel wird von jeder Menge von Literalen falsifiziert.

Seien $A_1, \ldots, A_j$ die in Δ vorkommenden, paarweise verschiedenen Atome. Wir definieren eine Folge $L_0, L_1, \ldots, L_j$ von Mengen von Literalen mit $0 \leq i \leq j$ wie folgt:

$$L_0 \coloneqq \emptyset$$

$$L_{i+1} \coloneqq \begin{cases} L_i \cup \{A_{i+1}\} & \text{falls } L_i \cup \{A_{i+1}\} \text{ keine Klausel aus } \Delta \text{ falsifiziert,} \\ L_i \cup \{\neg A_{i+1}\} & \text{sonst.} \end{cases}$$

BEISPIEL. Sei $\Delta = \{ \vdash A_1 \; ; \; A_1 \vdash A_2 \; ; \; A_2 \vdash ; \vdash A_2 \; ; \; A_1 \vdash \; ; \; \square\}$. Dann ist $L_0 = \emptyset$, $L_1 = L_0 \cup \{\neg A_1\}$, da $L_0 \cup \{A_1\}$ die Klausel $A_1 \vdash$ falsifiziert, und $L_2 = L_1 \cup \{\neg A_2\}$, da $L_1 \cup \{A_2\}$ eine Klausel aus Δ falsifiziert (dies ist auch schon für L_1 allein der Fall).

Da die leere Menge $\emptyset$ erfüllbar ist und die Atome $A_1, \ldots, A_j$ paarweise verschieden sind, ist jede der Mengen $L_0, L_1, \ldots, L_j$ erfüllbar.

Sei nun Δ unerfüllbar. Dann gilt insbesondere für L_j, dass kein Modell von L_j ein Modell aller Elemente von Δ sein kann. Wir zeigen, dass daraus $\square \in \Delta$ folgt, bzw. dass ein Widerspruch folgt, mit demselben Ergebnis.

Sei i das kleinste k (für $0 \leq k \leq j$), so dass die Menge L_k eine Klausel $S \in \Delta$ falsifiziert.

Fall $i = 0$: Es ist $L_i = \emptyset$. Da $\emptyset$ nur $\square$ falsifiziert, gilt $\square \in \Delta$.

Fall $i \neq 0$: Die Menge $L_{i-1} \cup \{A_i\}$ muss nach Voraussetzung eine Klausel $S' \in \Delta$ falsifizieren; folglich ist $L_i = L_{i-1} \cup \{\neg A_i\}$.

Wegen der Minimalität von i gilt, dass A_i im Antezedens von S' und im Sukzedens von S vorkommt; andernfalls würde schon die Menge L_{i-1} eine Klausel aus Δ falsifizieren.

Dann falsifiziert L_{i-1} aber die Resolvente von S' und S bezüglich A_i. Doch diese Resolvente gehört zu Δ, da Δ nach Definition unter Resolventenbildung abgeschlossen ist. Dann kann i aber nicht minimal sein, im Widerspruch zur Annahme.

Folglich gilt: Δ unerfüllbar $\implies \square \in \Delta$. QED

Bemerkung. Zur Verdeutlichung: Der Beweis von

$$\Delta \text{ unerfüllbar} \implies \square \in \Delta$$

hat (im natürlichen Schließen) die Struktur

$$\dfrac{i = 0 \vee i \neq 0 \qquad \dfrac{\dfrac{\Delta \text{ unerfüllbar}^{(2)} \quad i = 0^{(1)}}{\square \in \Delta} \qquad \dfrac{\Delta \text{ unerfüllbar}^{(2)} \quad i \neq 0^{(1)}}{\dfrac{\text{Widerspruch}}{\square \in \Delta}}}{\square \in \Delta}\,(1)}{\Delta \text{ unerfüllbar} \implies \square \in \Delta}\,(2)$$

Als Instanz des *tertium non datur* gilt $i = 0 \vee i \neq 0$. Die Fallunterscheidung führt (jeweils unter der zusätzlichen Annahme, dass Δ unerfüllbar ist) auf $\square \in \Delta$. Im zweiten Fall erhält man $\square \in \Delta$ per *ex falso quodlibet* aus dem Widerspruch. Da in beiden Fällen $\square \in \Delta$ gilt, schließen wir mit Disjunktionsbeseitigung (1) auf $\square \in \Delta$, was jetzt nicht mehr von den Annahmen $i = 0$ und $i \neq 0$ abhängt. Die Aussage $\square \in \Delta$ hängt noch von der Annahme „Δ unerfüllbar“ ab, die im letzten Schritt bei der Implikationseinführung (2) gelöscht wird.

Theorem 2.27 (Vollständigkeit Resolutionskalkül) *Sei $\square \notin \Gamma$. Dann ist der aussagenlogische Resolutionskalkül semantisch vollständig, d. h., es gilt: Wenn $\Gamma \vDash S$, dann $\Gamma \vdash_{\text{Res}} S$.*

Beweisskizze. Wir geben ein Verfahren an, um aus einer Resolutionswiderlegung eine Ableitung von S aus Γ im Resolutionskalkül zu erhalten, falls $\Gamma \vDash S$ gilt:

Die Klausel S habe die Form $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$. Sei

$$\Delta = \{ \vdash A_1 \,;\, \ldots \,;\, \vdash A_n \,;\, B_1 \vdash \,;\, \ldots \,;\, B_m \vdash \}.$$

Dann entspricht Δ die Menge $\{A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m\}$, welche S falsifiziert; d. h., es gilt für jede Bewertung $\mathcal{I}$: $\mathcal{I} \vDash \Delta \implies \mathcal{I} \nvDash S$. Mit $\Gamma \vDash S$ folgt, dass $\Gamma \cup \Delta$ unerfüllbar ist.

Aufgrund der Vollständigkeit des Widerlegungskalküls (Theorem 2.26) gilt dann

$$\square \in R(\Gamma \cup \Delta)$$

d. h., es gibt eine Ableitung der leeren Klausel aus $\Gamma \cup \Delta$.

In dieser Ableitung eliminiert man nun alle Resolutionen mit Klauseln aus Δ.

Anstelle der Ableitung der leeren Klausel erhält man so die Ableitung einer Teilklausel von S oder S selbst.

Im ersten Fall erhält man durch Verwendung des Axioms (Verdünnung) schließlich S. (Im zweiten Fall hat man schon die gesuchte Klausel S.)

Beispiel. Sei $\Gamma = \{ \vdash A\}$ und S die Klausel $\vdash A, B$. Dann ist $\Delta = \{A \vdash \,;\, B \vdash\}$ eine Menge, die S falsifiziert. Eine Ableitung der leeren Klausel ist dann

$$\frac{\vdash A \qquad A \vdash}{\vdash} \,(\mathcal{R})$$

Nach Elimination der Resolution mit der Klausel $A \vdash$ aus Δ bleibt $\vdash A$. Daraus erhält man durch Verdünnung mit dem Axiom eine Ableitung von S aus Γ:

$$\frac{\vdash A \qquad \dfrac{}{A \vdash A, B}\,(\text{Ax})}{\vdash A, B}\,(\mathcal{R})$$

Im Fall

$$\frac{\dfrac{\vdash A, B \qquad B \vdash}{\vdash A}\,(\mathcal{R}) \qquad A \vdash}{\vdash}\,(\mathcal{R})$$

erhält man nach Elimination der beiden Resolutionsschritte mit den Klauseln $B \vdash$ und $A \vdash$ aus Δ die Klausel S selbst. In jedem Fall erhält man eine Ableitung von S aus Γ.

Falls in der Ausgangsableitung Klauseln aus Δ mit Klauseln aus Δ resolviert wurden, d. h., wenn S tautologisch ist (es ist z. B. $\Delta = \{ \vdash A\,;\ A \vdash \}$ eine Menge, die $S = (A \vdash A)$ falsifiziert), erhält man ebenfalls durch Benutzung des Axioms die gesuchte Ableitung.

(Um aus dieser Beweisskizze einen Beweis zu erhalten, muss man zeigen, dass das angegebene Verfahren für eine gegebene Resolutionswiderlegung von $\Gamma \cup \Delta$ stets die gesuchte Ableitung von S aus Γ im Resolutionskalkül liefert.) QED

Zusammen mit der semantischen Korrektheit des aussagenlogischen Resolutionskalküls (Theorem 2.17) gilt somit für $\square \notin \Gamma$: $\Gamma \models S$ *genau dann, wenn* $\Gamma \vdash_{\text{Res}} S$.

Die Einschränkung $\square \notin \Gamma$ in Theorem 2.27 ist nötig, da im Resolutionskalkül die leere Klausel $\square$ nicht verdünnt werden kann. Ist $\square \in \Gamma$, dann gilt $\Gamma \models S$ für jede Klausel S, da Γ unerfüllbar ist. Falls $\Gamma \setminus \{\square\}$ erfüllbar ist, gilt dann jedoch $\Gamma \nvdash_{\text{Res}} S$ für alle nicht-tautologischen Klauseln $S \notin \Gamma$. (Alle tautologischen Klauseln und alle Klauseln in Γ sind trivialerweise ableitbar.) Der aussagenlogische Resolutionskalkül kann dann nicht semantisch vollständig sein.
Alternativ kann man eine der beiden folgenden Optionen wählen:

(i) Im Fall $\square \in \Gamma$ ersetzen wir die leere Klausel $\square$ durch zwei Klauseln $\vdash A$ und $A \vdash$. Die neue Klauselmenge sei Γ'.

Da Γ' ebenso wie Γ unerfüllbar ist, gilt auch $\Gamma' \models S$ für alle S. Für den Kalkül gilt $\Gamma' \vdash_{\text{Res}} S$ für alle S, da nun jede beliebige Klausel $S = (X \vdash Y)$ aus Γ' abgeleitet werden kann:

$$\dfrac{\dfrac{\vdash A \qquad \dfrac{}{X, A \vdash A, Y}\,(\text{Ax})}{X \vdash A, Y}\,(\mathcal{R}) \qquad A \vdash}{X \vdash Y}\,(\mathcal{R})$$

(Anstatt $\square$ in Γ durch die Klauseln $\vdash A$ und $A \vdash$ zu ersetzen, kann man Γ auch einfach um diese beiden Klauseln zu einer Menge Γ_e erweitern. Denn $\Gamma_e \setminus \{\square\}$ ist unerfüllbar unabhängig davon, ob $\Gamma \setminus \{\square\}$ erfüllbar ist oder nicht. Man könnte deshalb auch die Einschränkung $\square \notin \Gamma$ in Theorem 2.27 abschwächen, indem man $\square \notin \Gamma$ nur dann fordert, wenn $\Gamma \setminus \{\square\}$ erfüllbar ist. Ist nämlich $\Gamma \setminus \{\square\}$ schon unerfüllbar, so lässt sich jede beliebige Klausel daraus ableiten.)

(ii) Wir lassen $\square \in \Gamma$ zu, erweitern aber den Resolutionskalkül um eine Verdünnungsregel

$$\dfrac{X_1 \vdash Y_1}{X_1, X_2 \vdash Y_1, Y_2}\,(\mathcal{V})$$

wobei X_1 und X_2 insbesondere leer sein dürfen. Dann kann aus der leeren Klausel ebenfalls jede beliebige Klausel abgeleitet werden.

Für jede der beiden Optionen gilt, dass der (ggf. um die Verdünnungsregel $(\mathcal{V})$ erweiterte) aussagenlogische Resolutionskalkül semantisch korrekt und vollständig ist.

Bei der Behandlung des Resolutionskalküls als Widerlegungskalkül (also ohne Axiom und Verdünnungsregel) interessiert man sich ausschließlich für das Auffinden von Ableitungen der leeren Klausel aus gegebenen Klauselmengen. Die leere Klausel ist dann stets Endpunkt solcher Ableitungen.

3 Quantorenlogische Resolution

3.1 Syntax der Quantorenlogik

Definition 3.1 Das *Alphabet* einer formalen Sprache $\mathscr{L}$ bestehe aus folgenden Zeichen: *Alphabet*

(i) Logische Zeichen:
- *Konnektive*: $\neg, \wedge, \vee, \rightarrow$
- *Quantoren*: $\forall$ (Allquantor) und $\exists$ (Existenzquantor)

(ii) Nicht-logische Zeichen (ggf. mit Indizes):
- *Individuenvariablen* (kurz: *Variablen*): $x, y, z, \ldots$
  (Können auch zu den logischen Zeichen gezählt werden.)
- *Individuenkonstanten* (kurz: *Konstanten*): $a, b, c, \ldots$
- *Funktionszeichen*: $f, g, h, \ldots$
  (Gegebenenfalls mit Stelligkeit. Statt Konstanten könnten wir auch 0-stellige Funktionszeichen verwenden.)
- *Relationszeichen* (bzw. *Prädikatsymbole*): $P, Q, R, \ldots$
  (Gegebenenfalls mit fester Stelligkeit. 0-stellige Relationszeichen entsprechen Aussagesymbolen.)

(iii) *Hilfszeichen*: Klammern (,) und das Komma ,

Bemerkung. Wir verwenden $x, y, z, \ldots$ auch als metasprachliche Zeichen für Variablen, R als metasprachliche Variable für Relationszeichen, und $t, s, t_1, s_1, \ldots$ als metasprachliche Variablen für Terme (im Sinne der folgenden Definition).

Definition 3.2 *Terme* sind wie folgt induktiv definiert: *Terme*

(i) Jede Variable ist ein Term.

(ii) Jede Individuenkonstante ist ein Term.

(iii) Ist f ein n-stelliges Funktionszeichen und sind $t_1, \ldots, t_n$ Terme, dann ist auch $f(t_1, \ldots, t_n)$ ein Term.

Ein Term, der keine Variable enthält, heißt *Grundterm*. *Grundterm*

Beispiele. (i) Es sind a und $g(b, f(c))$ Grundterme; $h(x)$ ist *kein* Grundterm.

(ii) *Keine* Terme sind $x(y, z)$ und $(x \vee f(x))$.

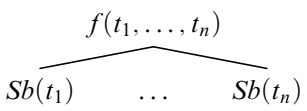
(iii) $P(a, b)$ ist *kein* Term, da das Prädikatsymbol P nicht zur Termbildung verwendet werden kann.

Definition 3.3 Die Erzeugung von Termen t kann als *Strukturbaum* dargestellt werden, der wie folgt definiert ist:

(i) Der Strukturbaum $Sb(x)$ einer Variable x ist der Knoten x.

(ii) Der Strukturbaum $Sb(a)$ einer Konstante a ist der Knoten a.

(iii) Der Strukturbaum $Sb(f(t_1, \ldots, t_n))$ eines Terms $f(t_1, \ldots, t_n)$ ist



Definition 3.4 (i) *Teilterme* eines Terms t sind alle in $Sb(t)$ vorkommenden Terme (einschließlich t). *Teilterm*

(ii) *Echte Teilterme* eines Terms t sind alle Teilterme von t mit Ausnahme von t selbst. *echter Teilterm*

(iii) *Unmittelbare Teilterme* eines Terms t sind die in $Sb(t)$ unmittelbar auf t folgenden Knoten. *unmittelbarer Teilterm*

Definition 3.5 Die *(quantorenlogischen) Formeln* sind wie folgt induktiv definiert: *Formeln*

(i) Ist R ein n-stelliges Relationszeichen und sind $t_1, \ldots, t_n$ Terme, dann ist $R(t_1, \ldots, t_n)$ eine Formel, die wir *atomare Formel* bzw. *Atom* nennen.

(Abkürzend schreiben wir auch: $Rt_1, \ldots, t_n$.)

(ii) Sind A und B Formeln, dann auch $\neg A$, $(A \wedge B)$, $(A \vee B)$ und $(A \to B)$.

(iii) Ist A eine Formel und ist x eine Variable, dann sind $\forall xA$ und $\exists xA$ Formeln.

Bemerkungen. (i) Die Quantoren binden stärker als die Konnektive; Klammerersparnis entsprechend. *Bindungsstärke*

(ii) Das *Vorkommen* eines Zeichens in einem Term oder in einer Formel sei nur anhand zweier Beispiele erläutert: Im Term $g(x, f(x, x))$ kommt x dreimal vor; in der Formel $\forall xP(x) \wedge Q(x, h(x))$ kommt x viermal vor. *Vorkommen*

Definition 3.6 Wir definieren *freie* bzw. *gebundene* Variablenvorkommen: *frei/gebunden*

(i) Jedes Variablenvorkommen in einer atomaren Formel ist frei.

(ii) Die freien bzw. gebundenen Variablenvorkommen in A und B sind auch freie bzw. gebundene Variablenvorkommen in $\neg A$, $(A \wedge B)$, $(A \vee B)$, $(A \to B)$.

(iii) Jedes freie Vorkommen von x in A ist ein gebundenes Vorkommen von x in $\forall xA$ und $\exists xA$. Alle anderen freien bzw. gebundenen Variablenvorkommen in A sind auch freie bzw. gebundene Variablenvorkommen in $\forall xA$ und $\exists xA$.

Definition 3.7 (i) Die freien Vorkommen von x in A liegen im *Wirkungsbereich* des $\forall$- bzw. $\exists$-Quantors in $\forall xA$ bzw. $\exists xA$. *Wirkungsbereich*

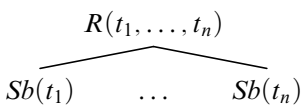
(ii) In den Ausdrücken $\forall x$ und $\exists x$ kommt x weder frei noch gebunden vor.

(iii) Die Menge der in einer Formel A frei vorkommenden Variablen bezeichnen wir mit $\mathrm{FV}(A)$. Für Terme t bezeichnet $\mathrm{FV}(t)$ die Menge der in t vorkommenden Variablen. (Wir werden FV auch bei anderen, noch zu definierenden Ausdrücken entsprechend verwenden.) FV

(iv) Eine Formel ohne freie Variablenvorkommen heißt *geschlossen*, andernfalls *offen*. Eine geschlossene Formel wird auch als *Aussage* bezeichnet. *geschlossen/offen* *Aussage*

Definition 3.8 Für quantorenlogische Formeln definieren wir *Strukturbäume* wie folgt (wobei wir auf Definition 2.3 für Strukturbäume aussagenlogischer Formeln zurückgreifen): *Strukturbaum*

(i) Der Strukturbaum $Sb(R(t_1, \ldots, t_n))$ einer atomaren Formel $R(t_1, \ldots, t_n)$ ist



(ii) Gemäß Definition 2.3(ii).

(iii) Gemäß Definition 2.3(iii).

(iv) Die Strukturbäume $Sb(\forall xA)$ und $Sb(\exists xA)$ von Formeln $\forall xA$ und $\exists xA$ sind

$$\begin{array}{ccc} \forall xA & & \exists xA \\ | & \text{und} & | \\ Sb(A) & & Sb(A) \end{array}$$

Beispiel. Wir betrachten die Formel

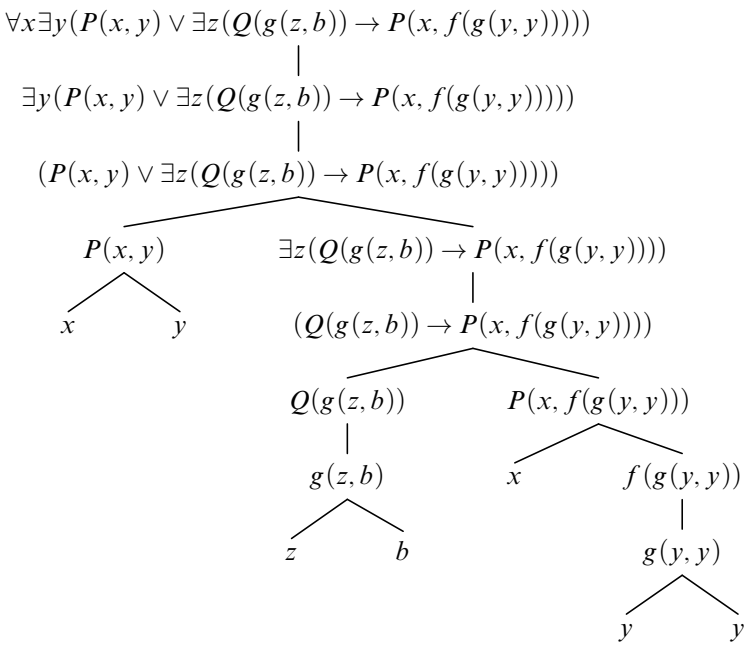
$$\forall x\,\exists y(P(x,y) \vee \exists z\,(Q(g(z,b)) \to P(x,f(g(y,y)))))$$

Wirkungsbereich von $\exists z$: $(Q(g(z,b)) \to P(x,f(g(y,y))))$

Wirkungsbereich von $\exists y$: $(P(x,y) \vee \exists z\,(Q(g(z,b)) \to P(x,f(g(y,y)))))$

Wirkungsbereich von $\forall x$: $\forall x\,\exists y(P(x,y) \vee \exists z\,(Q(g(z,b)) \to P(x,f(g(y,y)))))$

Sie hat folgenden Strukturbaum:



Für die in den Teilformeln vorkommenden Variablen gilt:

- In $P(x,y)$ sind x,y frei. In $Q(g(z,b))$ ist z frei. In $P(x,f(g(y,y)))$ sind x,y frei.
- In $(Q(g(z,b)) \to P(x,f(g(y,y))))$ sind x,y,z frei.
- In $\exists z(Q(g(z,b)) \to P(x,f(g(y,y))))$ sind x,y frei; z ist gebunden.
  (In $\exists z$ ist z weder frei noch gebunden.)
- In $(P(x,y) \vee \exists z(Q(g(z,b)) \to P(x,f(g(y,y)))))$ sind x,y frei; z ist gebunden.
- In $\exists y(P(x,y) \vee \exists z(Q(g(z,b)) \to P(x,f(g(y,y)))))$ ist x frei; y,z sind gebunden.
  (In $\exists y$ ist y weder frei noch gebunden.)
- In $\forall x\exists y(P(x,y) \vee \exists z(Q(g(z,b)) \to P(x,f(g(y,y)))))$ sind x,y,z gebunden.
  (In $\forall x$ ist x weder frei noch gebunden.)

3.2 Substitution

Definition 3.9 (i) Eine *Substitution* ist eine Funktion, die Variablen auf Terme abbildet und für nur endlich viele Variablen nicht identisch ist. *Substitution*

(ii) Wir notieren Substitutionen als endliche Mengen (die wir mit eckigen Klammern [,] schreiben) der Form

$$[x_1/t_1, \ldots, x_n/t_n] \qquad (\text{für } 0 \leq i \leq n),$$

für paarweise verschiedene Variablen x_i und Terme t_i, so dass $t_i \neq x_i$.

In der Notation $[x_1/t_1, \ldots, x_n/t_n]$ werden also nur die (immer endlich vielen) nicht-identischen Zuordnungen von Termen zu Variablen einer Substitution angegeben.

(iii) Es heißt x_i/t_i *Bindung* für x_i. *Bindung*

Wir sagen auch "t_i wird für x_i substituiert" oder "x_i wird durch t_i ersetzt".

(iv) Substitutionen bezeichnen wir mit $\sigma, \rho, \tau, \vartheta, \ldots$

(v) Eine Substitution $\sigma = [x_1/t_1, \ldots, x_n/t_n]$ heißt *Grundsubstitution*, falls jeder Term t_i ein Grundterm ist. *Grundsubstitution*

(vi) Ist $\sigma = \emptyset$, so heißt σ *leere Substitution* und wird mit ε bezeichnet. *leere Substitution*

Beispiele. (i) $[x_1/f(x), y/g(a,z), z/x]$ ist eine Substitution.

(ii) $[x/h(a,b)]$ ist eine Grundsubstitution.

Definition 3.10 Sei $\sigma = [x_1/t_1, \ldots, x_n/t_n]$ eine Substitution und E ein Ausdruck, d. h. eine Formel oder ein Term.

(i) Die simultane Ersetzung jedes freien Vorkommens von x_i in E durch t_i für $0 \leq i \leq n$ heißt *Anwendung* von σ auf E. *Anwendung*

Notation: $E\sigma$. Die Substitution wirkt also nach links.

Ist E eine Formel, wird zusätzlich gefordert, dass t_i in E *frei einsetzbar* ist für x_i; d. h., x_i darf nur dann durch t_i ersetzt werden, wenn in t_i vorkommende Variablen in E nicht durch einen Quantor gebunden werden. *frei einsetzbar*

(ii) Der resultierende Ausdruck $E\sigma$ heißt *(Substitutions-) Instanz* von E für σ. *Instanz*

(iii) Ist $X = \{E_1, \ldots, E_n\}$ eine endliche Menge von Ausdrücken, so steht $X\sigma$ für die Menge $\{E_1\sigma, \ldots, E_n\sigma\}$.

Bemerkung. Da Substitutionen simultan ausgeführt werden, ist

$$f(x,y)[x/y, y/b] = f(y,b)$$

und *nicht* $f(b,b)$, was man durch die Hintereinanderausführung

$$(f(x,y)[x/y])[y/b] = f(y,y)[y/b] = f(b,b)$$

erhalten würde.

Definition 3.11 Die *Komposition* $\sigma\tau$ zweier Substitutionen *Komposition*

$$\sigma = [x_1/s_1, \ldots, x_n/s_n] \quad \text{und} \quad \tau = [y_1/t_1, \ldots, y_m/t_m]$$

mit $0 \leq i \leq n$ und $0 \leq j \leq m$ ist wie folgt gegeben: Wir bilden die Folge von Bindungen

$$x_1/(s_1\tau), \ldots, x_n/(s_n\tau), y_1/t_1, \ldots, y_m/t_m.$$

Aus dieser Folge entfernen wir

– alle Bindungen $x_i/(s_i\tau)$, für die $x_i = (s_i\tau)$,

– und alle Bindungen y_j/t_j, für die $y_j \in \{x_1, \ldots, x_n\}$.

Die aus den Bindungen in der resultierenden Folge bestehende Substitution ist die *Komposition* $\sigma\tau$ von σ und τ.

Beispiel. Sei $\sigma = [x/f(y), y/z]$ und $\tau = [x/a, y/b, z/y]$. Dann ist die Folge von Bindungen

$$\underbrace{x/(f(y)[x/a, y/b, z/y])}_{x/f(b)}, \underbrace{y/(z[x/a, y/b, z/y])}_{y/y}, x/a, y/b, z/y$$

aus der die Bindungen y/y, x/a und y/b zu entfernen sind. Man erhält die Komposition $\sigma\tau = [x/f(b), z/y]$.

Lemma 3.12 *Seien ρ, σ, ϑ Substitutionen und ε die leere Substitution. Dann gilt:*

(i) $\rho\varepsilon = \varepsilon\rho = \rho$.

(ii) $(\rho\sigma)\vartheta = \rho(\sigma\vartheta)$.

(iii) $(E\rho)\sigma = E(\rho\sigma)$, *für beliebige Formeln und Terme E.*

Beweis. Übungsaufgabe. QED

Lemma 3.13 *Die Komposition von Substitutionen ist nicht kommutativ.*

Beweis. Übungsaufgabe. QED

3.3 Semantische Begriffe der Quantorenlogik

Die Semantik der Quantorenlogik wird in Abschnitt 5.1 ausführlich behandelt. Da wir im Folgenden schon semantische Begriffe wie z. B. *Erfüllbarkeit* und *logische Äquivalenz* quantorenlogischer Formeln verwenden möchten, sollen diese hier kurz erläutert werden.

Um einer quantorenlogischen Formel in einer Sprache $\mathscr{L}$ einen Wahrheitswert zuzuordnen, wird zunächst ein nicht-leerer *Gegenstandsbereich* M gewählt. Durch eine *Interpretation* $\mathcal{I}$ werden dann folgende Zuordnungen vorgenommen: *Gegenstandsbereich* *Interpretation*

Zeichen in $\mathscr{L}$	*Interpretation $\mathcal{I}$*
Individuenkonstante	Individuum in M
n-stelliges Funktionszeichen	n-stellige Funktion über M^n
n-stelliges Relationszeichen	n-stellige Relation über M^n

Das Paar $\langle M, \mathcal{I} \rangle$ bezeichnet man als *Struktur* (für die Sprache $\mathscr{L}$). Für eine gegebene Struktur $\langle M, \mathcal{I} \rangle$ für $\mathscr{L}$ bezeichnen Grundterme aus $\mathscr{L}$ Gegenstände in M. Geschlossene Formeln stehen für Aussagen über Gegenstände in M. Aussagen können wahr oder falsch sein; Terme nicht. *Struktur*

Beispiel. Für $M = \mathbb{N}$ kann die Individuenkonstante a z. B. durch $0 \in \mathbb{N}$ interpretiert werden, und das 1-stellige Funktionszeichen f als Nachfolgerfunktion über $\mathbb{N}$. Der geschlossene Term $f(a)$ bedeutet dann die Zahl 1 (da 1 der Nachfolger von 0 ist). Wird das 2-stellige Relationszeichen R als $\leq$ über M^2 interpretiert, so ist die geschlossene Formel $R(a, f(a))$ in der gegebenen Struktur wahr, da $0 \leq 1$.

Um auch offenen Termen und Formeln eine Bedeutung zuordnen zu können, müssen auch Variablen interpretiert werden. Dies geschieht durch eine *Variablenbelegung* v, die jeder Variable einen Gegenstand aus dem Gegenstandsbereich zuordnet. Bei gegebener Struktur und Variablenbelegung kann dann jeder Formel ein Wahrheitswert zugeordnet werden. *Variablenbelegung*

Beispiel. Für die Struktur aus dem vorherigen Beispiel und eine Variablenbelegung, die x den Gegenstand 2 zuordnet, hat die offene Formel $R(f(a), x)$ den Wahrheitswert w (da $1 \leq 2$).

Für die im Folgenden verwendeten semantischen Begriffe gilt (vgl. Definition 5.4 und 5.5):

(i) Eine Formel $\forall x A$ ist in einer Struktur mit Gegenstandsbereich M unter einer Variablenbelegung v genau dann *wahr*, wenn die Formel A in dieser Struktur mit $v(x) = m$ für alle $m \in M$ wahr ist, und eine Formel $\exists x A$ ist genau dann *wahr*, wenn A mit $v(x) = m$ für (mindestens) ein $m \in M$ wahr ist. *wahr in Struktur unter v*

(ii) Eine Formel A ist *erfüllbar*, falls es eine Struktur mit einer Variablenbelegung gibt, so dass A wahr ist. *erfüllbar*

(iii) Eine Formel A ist *allgemeingültig*, falls A in jeder Struktur unter allen jeweiligen Variablenbelegungen wahr ist. *allgemeingültig*

(iv) Die *(quantoren-)logische Folgerung* $\Gamma \vDash A$ ist für geschlossene Formeln in einer Sprache $\mathscr{L}$ analog zur aussagenlogischen Folgerung definiert: *logische Folgerung*

$$\Gamma \vDash A \ :\Longleftrightarrow\ \text{Für jede Struktur } \mathfrak{M} \text{ für } \mathscr{L}\text{: Wenn } \mathfrak{M} \vDash \Gamma, \text{ dann } \mathfrak{M} \vDash A.$$

Für offene Formeln sind für jede Struktur noch alle jeweiligen Variablenbelegungen zu berücksichtigen; siehe Definition 5.6.

(v) Zwei Formeln A und B sind *logisch äquivalent* (Notation: $A \dashv\vDash B$) genau dann, wenn A und B in jeder Struktur unter allen jeweiligen Variablenbelegungen denselben Wahrheitswert haben. *logisch äquivalent*

3.4 Pränexe Normalform

Definition 3.14 (i) Eine Formel A einer Sprache $\mathscr{L}$ ist in *pränexer Normalform* (kurz: PNF), wenn sie die Form $Q_1 x_1 \ldots Q_n x_n B$ hat, wobei B quantorenfrei, $n \geq 0$, Q_i entweder $\forall$ oder $\exists$ ist, und alle x_i paarweise verschieden sind. *pränexe Normalform*

(ii) Es heißt $Q_1 x_1 \ldots Q_n x_n$ *Präfix* der Formel A, und B heißt *Kern* oder *Matrix von A*.

Im Fall $n = 0$ ist A quantorenfrei und identisch mit B.

Theorem 3.15 *Sei A eine Formel. Dann gibt es eine Formel B in pränexer Normalform, so dass A und B logisch äquivalent sind.*

Beweis. Wir geben ein Verfahren an, um die Formel A in eine logisch äquivalente Formel B in pränexer Normalform zu überführen. Die den Umformungen zugrunde liegenden logischen Äquivalenzen werden hier nicht bewiesen. Die einzelnen Umformungsschritte werden am Beispiel der Formel

$$\forall z(\forall x(P(x) \to P(f(x))) \vee \neg\forall x(Q(x) \vee R(x,a)))$$

dargestellt.

(1) Streiche alle leeren Quantoren in A, d. h.

(a) $\forall x A_1 \rightsquigarrow A_1$, falls x nicht frei in A_1; (b) $\exists x A_1 \rightsquigarrow A_1$, falls x nicht frei in A_1.

Es gilt $A \dashv\models A_1$.

BEISPIEL. $\forall z(\forall x(P(x) \to P(f(x))) \vee \neg\forall x(Q(x) \vee R(x,a))) \rightsquigarrow$
$\forall x(P(x) \to P(f(x))) \vee \neg\forall x(Q(x) \vee R(x,a))$

(2) Benenne gebundene Variablen in A_1 so um, dass alle Quantoren verschiedene Variablen haben, keine Variable sowohl frei als auch gebunden vorkommt, und freie Variablen nicht gebunden werden.

Die resultierende Formel sei A_2. Es gilt $A_1 \dashv\models A_2$.

BEISPIEL. $\forall x(P(x) \to P(f(x))) \vee \neg\forall x(Q(x) \vee R(x,a)) \rightsquigarrow$
$\forall x(P(x) \to P(f(x))) \vee \neg\forall y(Q(y) \vee R(y,a))$

(3) Ziehe Negationen nach innen, so dass diese nur noch vor Atomen vorkommen, und beseitige doppelte Negationen:

(a) $\neg\forall x C \rightsquigarrow \exists x \neg C$
(b) $\neg\exists x C \rightsquigarrow \forall x \neg C$
(c) $\neg(C \wedge D) \rightsquigarrow (\neg C \vee \neg D)$
(d) $\neg(C \vee D) \rightsquigarrow (\neg C \wedge \neg D)$
(e) $\neg(C \to D) \rightsquigarrow (C \wedge \neg D)$
(f) $\neg\neg C \rightsquigarrow C$

Die resultierende Formel sei A_3. Es gilt $A_2 \dashv\models A_3$.

BEISPIEL. $\forall x(P(x) \to P(f(x))) \vee \neg\forall y(Q(y) \vee R(y,a)) \rightsquigarrow$
$\forall x(P(x) \to P(f(x))) \vee \exists y \neg(Q(y) \vee R(y,a)) \rightsquigarrow$
$\forall x(P(x) \to P(f(x))) \vee \exists y(\neg Q(y) \wedge \neg R(y,a))$

(4) Ziehe Quantoren nach außen:

(a) $\forall x C \wedge D \rightsquigarrow \forall x(C \wedge D)$
(b) $C \wedge \forall x D \rightsquigarrow \forall x(C \wedge D)$
(c) $\exists x C \wedge D \rightsquigarrow \exists x(C \wedge D)$
(d) $C \wedge \exists x D \rightsquigarrow \exists x(C \wedge D)$
(e) $\forall x C \vee D \rightsquigarrow \forall x(C \vee D)$
(f) $C \vee \forall x D \rightsquigarrow \forall x(C \vee D)$
(g) $\exists x C \vee D \rightsquigarrow \exists x(C \vee D)$
(h) $C \vee \exists x D \rightsquigarrow \exists x(C \vee D)$
(i) $\forall x C \to D \rightsquigarrow \exists x(C \to D)$
(j) $C \to \forall x D \rightsquigarrow \forall x(C \to D)$
(k) $\exists x C \to D \rightsquigarrow \forall x(C \to D)$
(l) $C \to \exists x D \rightsquigarrow \exists x(C \to D)$

Durch (2) ist sichergestellt, dass hierdurch keine freien Variablen gebunden werden.

Die resultierende Formel ist die gesuchte pränexe Normalform B der Ausgangsformel A. Es gilt $A_3 \dashv\vDash B$ und, da $A \dashv\vDash A_1 \dashv\vDash A_2 \dashv\vDash A_3 \dashv\vDash B$, auch $A \dashv\vDash B$. Die pränexe Normalform ist also logisch äquivalent zur Ausgangsformel.

BEISPIEL. $\forall x(P(x) \to P(f(x))) \vee \exists y(\neg Q(y) \wedge \neg R(y,a)) \rightsquigarrow$
$\forall x((P(x) \to P(f(x))) \vee \exists y(\neg Q(y) \wedge \neg R(y,a))) \rightsquigarrow$
$\forall x \exists y((P(x) \to P(f(x))) \vee (\neg Q(y) \wedge \neg R(y,a)))$

oder auch $\forall x(P(x) \to P(f(x))) \vee \exists y(\neg Q(y) \wedge \neg R(y,a)) \rightsquigarrow$
$\exists y(\forall x(P(x) \to P(f(x))) \vee (\neg Q(y) \wedge \neg R(y,a))) \rightsquigarrow$
$\exists y \forall x((P(x) \to P(f(x))) \vee (\neg Q(y) \wedge \neg R(y,a)))$

Für das Verfahren ist nur die Reihenfolge der Schritte (1)-(4) zu beachten. Die Reihenfolge der Teilschritte (1a)/(1b), (3a)-(3f) und (4a)-(4l) ist hingegen beliebig.

Beispiele. (i) $\forall z(\exists x P(x) \to \neg\exists x Q(x,y)) \overset{(1a)}{\rightsquigarrow} \exists x P(x) \to \neg\exists x Q(x,y)$
$\overset{(2)}{\rightsquigarrow} \exists x P(x) \to \neg\exists z Q(z,y)$
$\overset{(3b)}{\rightsquigarrow} \exists x P(x) \to \forall z \neg Q(z,y)$
$\overset{(4j)}{\rightsquigarrow} \forall z(\exists x P(x) \to \neg Q(z,y))$
$\overset{(4k)}{\rightsquigarrow} \forall z \forall x(P(x) \to \neg Q(z,y))$

(ii) Man kann auch zuerst (4k) und dann (4j) anwenden:

$\exists x P(x) \to \forall z \neg Q(z,y) \overset{(4k)}{\rightsquigarrow} \forall x(P(x) \to \forall z \neg Q(z,y)) \overset{(4j)}{\rightsquigarrow} \forall x \forall z(P(x) \to \neg Q(z,y))$

Die pränexe Normalform einer Formel ist nicht eindeutig bestimmt, da

(i) unterschiedliche gebundene Umbenennungen möglich sind,

(ii) die Reihenfolge, in der Quantoren nach außen gezogen werden können, nicht festgelegt ist,

(iii) und der Kern prinzipiell durch jede zu diesem logisch äquivalente quantorenfreie Formel ersetzt werden darf.

3.5 Skolemisierung

Um zu einer gegebenen Formel in pränexer Normalform in einer Sprache $\mathscr{L}$ eine Klauselmenge bilden zu können, müssen zunächst Quantoren in geeigneter Weise beseitigt werden. Dies geschieht mittels *Skolemisierung* (nach Thoralf Skolem, 1887–1963), wodurch man eine quantorenfreie Formel in einer erweiterten Sprache $\mathscr{L}_S \supseteq \mathscr{L}$ erhält. Diese Formel ist im Allgemeinen nicht mehr logisch äquivalent zur Ausgangsformel, aber in jedem Fall noch erfüllbarkeitsäquivalent zu derselben.
Zur Erläuterung der Idee bei Skolemisierung betrachten wir die (in der Standardinterpretation wahre) Aussage

Für alle $n \in \mathbb{N}$ gibt es ein $m \in \mathbb{N}$, so dass $n < m$.

Für die Funktion $f(n) = n + 1$ gilt $n < f(n)$ für alle n. Die durch den Existenzquantor gebundene Variable m kann also durch die Funktion $f(n)$ ersetzt werden, wodurch der Existenzquantor überflüssig wird. Fasst man freie Variablen zudem als universell auf, so kann auch der Allquantor weggelassen werden. Lässt man auch von der Standardinterpretation abweichende Interpretationen zu, so ist die resultierende Aussage zwar nicht mehr logisch äquivalent zur Ausgangsaussage, doch die resultierende Aussage ist erfüllbar genau dann, wenn die Ausgangsaussage erfüllbar ist.
Obige Aussage hat die Form

$$\forall x \exists y P(x, y)$$

wobei wir hier annehmen wollen, dass dies eine Formel in einer Sprache $\mathscr{L}$ ist, die keine Funktionszeichen enthält. Nun ersetzt man in der Formel $\forall x \exists y P(x, y)$ die durch den Existenzquantor gebundene Variable y durch einen Funktionsterm $f(x)$. Dies stellt eine Spracherweiterung um das einstellige Funktionszeichen f dar. Lässt man jetzt noch den Allquantor weg, resultiert als Skolem-Normalform die Formel $P(x, f(x))$. Diese Formel ist erfüllbar genau dann, wenn die Ausgangsformel erfüllbar ist.

Definition 3.16 Der *All-Abschluss* $\forall A$ einer Formel A ist die Formel *All-Abschluss*

$$\forall x_1 \ldots \forall x_n A[y_1/x_1, \ldots, y_n/x_n]$$

wobei $y_1, \ldots, y_n$ *alle* freien Variablen in A und $x_1, \ldots, x_n$ Variablen sind, die in A nicht vorkommen. (Da die paarweise verschiedenen Variablen x_i ansonsten frei wählbar sind, ist dies eine nicht-deterministische Definition.)

Definition 3.17 Sei $A \in \mathscr{L}$ eine Formel der Form $\mathsf{Q}_1 x_1 \ldots \mathsf{Q}_n x_n B$ in pränexer Normalform mit Kern B. Dann ist eine *Skolem-Normalform* A_S von A in einer geeigneten Erweiterungssprache $\mathscr{L}_\mathrm{S}$ von $\mathscr{L}$ durch das folgende Verfahren definiert: *Skolem-Normalform*

(1) Setze $A_\mathrm{S} = \forall A$.

(2) Falls das Präfix von A_S nur Allquantoren enthält, streiche diese und halte. Ist das Präfix leer, so halte ebenfalls. A_S ist eine Skolem-Normalform von A.

(3) Sei $\mathsf{Q}_i x_i$ der erste Existenzquantor (von links) in A_S. Seien $x_{i_1}, \ldots, x_{i_j}$ die Variablen links von x_i; dies sind jene Variablen aus $\{x_1, \ldots, x_{i-1}\}$, die noch nicht entfernt wurden.

(4) Falls in A_S links von x_i keine Allquantoren stehen, dann erweitere $\mathscr{L}$ durch Hinzunahme einer neuen Konstante a_i (bzw. gleichbedeutend damit durch ein neues 0-stelliges Funktionszeichen) und ersetze jedes Vorkommen von x_i im Kern von A_S durch a_i.

(5) Andernfalls erweitere $\mathscr{L}$ durch Hinzunahme eines neuen j-stelligen Funktionszeichens f_i. Ersetze jedes Vorkommen von x_i im Kern von A_S durch den Term $f_i(x_{i_1}, \ldots, x_{i_j})$.

(6) Entferne $\exists x_i$ aus dem Präfix von A_S. Gehe zu Schritt (2).

Beispiele. Wir verwenden im Folgenden indizierte Variablen. Die Wahl der Individuenkonstanten und Funktionszeichen ist dann durch das Verfahren festgelegt. Bei Formeln mit nicht-indizierten Variablen ist darauf zu achten, dass in den Schritten (4) und (5) jeweils neue Konstanten, bzw. neue Funktionszeichen eingeführt werden (vgl. Beispiel (v)).

(i) $\exists x_1 P(x_1) \rightsquigarrow P(a_1)$

(ii) $\exists x_1 \forall x_2 P(x_1, x_2) \rightsquigarrow \forall x_2 P(a_1, x_2) \rightsquigarrow P(a_1, x_2)$

(iii) $\forall x_1 \exists x_2 P(x_1, x_2) \rightsquigarrow \forall x_1 P(x_1, f_2(x_1)) \rightsquigarrow P(x_1, f_2(x_1))$

(iv) $\begin{aligned} \exists x_1 \forall x_2 \exists x_3 P(x_1, x_2, x_3) &\rightsquigarrow \forall x_2 \exists x_3 P(a_1, x_2, x_3) \\ &\rightsquigarrow \forall x_2 P(a_1, x_2, f_3(x_2)) \\ &\rightsquigarrow P(a_1, x_2, f_3(x_2)) \end{aligned}$

(v) $\begin{aligned} \forall x_1 \exists x_2 \forall x_3 \exists x_4 P(x_1, x_2, x_3, x_4) &\rightsquigarrow \forall x_1 \forall x_3 \exists x_4 P(x_1, f_2(x_1), x_3, x_4) \\ &\rightsquigarrow \forall x_1 \forall x_3 P(x_1, f_2(x_1), x_3, f_3(x_1, x_3)) \\ &\rightsquigarrow P(x_1, f_2(x_1), x_3, f_4(x_1, x_3)) \end{aligned}$

Für nicht indizierte Variablen:

$\begin{aligned} \forall x \exists y \forall z \exists u P(x, y, z, u) &\rightsquigarrow \forall x \forall z \exists u P(x, f(x), z, u) \\ &\rightsquigarrow \forall x \forall z P(x, f(x), z, g(x, z)) \\ &\rightsquigarrow P(x, f(x), z, g(x, z)) \end{aligned}$

Alternativ kann gemäß folgender Definition skolemisiert werden, wodurch Allquantoren nicht erst zum Schluss entfernt werden.

Definition 3.18 Sei $A \in \mathscr{L}$ eine Formel der Form $\mathsf{Q}_1 x_1 \ldots \mathsf{Q}_n x_n B$ (für $n \geq 0$) in pränexer Normalform mit Kern B. Das folgende Verfahren liefert eine *Skolem-Normalform* A_S von A in einer geeigneten Sprache $\mathscr{L}_\mathrm{S} \supseteq \mathscr{L}$. *Skolem-Normalform*

Für i von 1 bis n:

(1) Falls $n = 0$, setze $A_\mathrm{S} := A$.

(2) Falls Q_i ein Allquantor ist, setze $A := \mathsf{Q}_{i+1} x_{i+1} \ldots \mathsf{Q}_n x_n B$.

(3) Falls Q_i ein Existenzquantor ist, und $y_1, \ldots, y_m$ die in A frei vorkommenden Variablen sind, setze

$$A := \mathsf{Q}_{i+1} x_{i+1} \ldots \mathsf{Q}_n x_n B[x_i / f(y_1, \ldots, y_m)]$$

wobei f ein neues (noch nicht in A vorkommendes) m-stelliges Funktionszeichen ist, bzw. eine neue Konstante, falls $m = 0$.

Bemerkungen. (i) Das Verfahren ist nicht-deterministisch, solange man nicht die Reihenfolge von einzuführenden Zeichen festlegt.

(ii) Im Unterschied zu dem in Definition 3.17 angegebenen Verfahren, bei dem im ersten Schritt zunächst der All-Abschluss der zu skolemisierenden Formel (in pränexer Normalform) gebildet werden muss, können hier auch offene Formeln (in pränexer Normalform) ohne Weiteres behandelt werden. Denn in Schritt (3) werden alle in A frei vorkommenden Variablen entsprechend berücksichtigt.

Beispiele. (i) $\exists x P(x) \rightsquigarrow P(a)$

(ii) $\exists x \forall y P(x, y) \rightsquigarrow \forall y P(a, y) \rightsquigarrow P(a, y)$

(iii) $\forall x \exists y P(x, y) \rightsquigarrow \exists y P(x, y) \rightsquigarrow P(x, f(x))$

(iv) $\exists x \forall y \exists z P(x, y, z) \rightsquigarrow \forall y \exists z P(a, y, z) \rightsquigarrow \exists z P(a, y, z) \rightsquigarrow P(a, y, f(y))$

(v) $\forall x \exists y \forall z \exists u P(x, y, z, u) \rightsquigarrow \exists y \forall z \exists u P(x, y, z, u)$
$\rightsquigarrow \forall z \exists u P(x, f(x), z, u)$
$\rightsquigarrow \exists u P(x, f(x), z, u)$
$\rightsquigarrow P(x, f(x), z, g(x, z))$

Die Skolem-Normalform A_S einer Formel A ist im Allgemeinen *nicht* logisch äquivalent zu A (es gilt zwar stets $\forall A_S \vDash A$, aber für gewisse Formeln A ist $A \nvDash A_S$). Betrachte z. B. die Formel $\exists x P(x)$ mit Skolem-Normalform $P(a)$. Die Struktur $\langle \{0, 1\}, \mathcal{I} \rangle$ mit $\mathcal{I}(P) = \{0\}$ und $\mathcal{I}(a) = 1$ ist zwar ein Modell von $\exists x P(x)$, aber sie ist kein Modell von $P(a)$.
Da nicht jede Formel eine logisch äquivalente Skolem-Normalform hat, spricht man manchmal auch von Skolem-*Standard*form.

Es gilt aber Erfüllbarkeitsäquivalenz:

Theorem 3.19 *Es ist $\forall A$ erfüllbar unter einer Interpretation genau dann, wenn gilt: $\forall A_S$ ist erfüllbar in einer geeigneten erweiterten Interpretation, in der die durch Skolemisierung eingeführten Konstanten und Funktionszeichen interpretiert sind.*

Beweis. Siehe Nienhuys-Cheng & de Wolf, 1997 (Kapitel 3) oder Doets, 1994 (Kapitel 3), wo von geschlossenen Formeln A ausgegangen wird. QED

Korollar 3.20 (i) *Um zu zeigen, dass $\forall A$ unerfüllbar ist, d. h., um $\forall A$ zu widerlegen, reicht es aus, $\forall A_S$ zu widerlegen.*

(ii) *Sei $C_1 \wedge \ldots \wedge C_m$ eine KNF von A_S, d. h. die konjunktive Normalform der Skolem-Normalform (kurz:* konjunktive Skolem-Normalform*) der Ausgangsformel A. Um zu zeigen, dass $\forall A$ unerfüllbar ist, reicht es aus, $\forall C_1 \wedge \ldots \wedge \forall C_m$ zu widerlegen.*

Die konjunktive Skolem-Normalform $C_1 \wedge \ldots \wedge C_m$ von A kann man (wie in der Aussagenlogik) als *Klauselmenge* schreiben, die wir mit A_{SK} (*konjunktive Skolem-Normalform in Klauselform*) bezeichnen. *Klauselmenge*
Klauseln sind wie bisher als Sequenzen $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ definiert, wobei die atomaren Formeln $A_1, \ldots, A_n, B_1, \ldots, B_m$ nun quantorenlogische Atome sind. *Klausel*
Definiert man ein *Widerlegungsverfahren* für Klauseln so, dass man dabei die freien Variablen in Klauseln universell versteht (siehe Abschnitt 3.6), dann gilt: Um zu zeigen, dass $\forall A$ unerfüllbar ist, genügt es, A_{SK} nach diesem Verfahren zu widerlegen; zum Nachweis der Allgemeingültigkeit einer geschlossenen Formel A genügt es, $(\neg A)_{SK}$ nach diesem Verfahren zu widerlegen. *Widerlegungsverfahren*
Für eine beliebige quantorenlogische Formel A sind also die folgenden drei Schritte auszuführen:

(1) Bilde eine Skolem-Normalform zu $\neg A$.

(2) Bilde daraus eine KNF.

(3) Wende das Widerlegungsverfahren an.

oder

(1) Bilde eine pränexe Normalform zu $\neg A$ mit Kern in KNF.

(2) Skolemisiere.

(3) Wende das Widerlegungsverfahren an.

Im Fall von Folgerungsbehauptungen $A_1, \ldots, A_n \models A$ geht man wie folgt vor:

(1) Bilde $(A_1)_{\mathrm{SK}}, \ldots, (A_n)_{\mathrm{SK}}$ und $(\neg A)_{\mathrm{SK}}$.

Hierbei sind die durch die jeweilige Skolemisierung eingeführten Terme so zu wählen, dass die Mengen dieser Terme für $(A_1)_{\mathrm{SK}}, \ldots, (A_n)_{\mathrm{SK}}$ und $(\neg A)_{\mathrm{SK}}$ paarweise disjunkt sind.

(2) Wende das Widerlegungsverfahren an.

Beispiele. (Die Skolem-Normalform wird jeweils gemäß Definition 3.18 gebildet.)

(i) $\models \exists x \forall y P(x,y) \to \forall y \exists x P(x,y)$

$$
\begin{array}{ll}
\neg(\exists x \forall y P(x,y) \to \forall y \exists x P(x,y)) & \text{(Negation der Ausgangsformel)} \\
\rightsquigarrow \neg(\exists x \forall y P(x,y) \to \forall z \exists u P(u,z)) & \text{(gebundene Umbenennung)} \\
\rightsquigarrow \exists x \forall y P(x,y) \wedge \exists z \forall u \neg P(u,z) & \text{(Negation nach innen gezogen)} \\
\rightsquigarrow \exists x \forall y \exists z \forall u (P(x,y) \wedge \neg P(u,z)) & \text{(PNF mit Kern in KNF)} \\
\rightsquigarrow \forall y \exists z \forall u (P(a,y) \wedge \neg P(u,z)) & \text{(Skolemisiere \ldots)} \\
\rightsquigarrow \exists z \forall u (P(a,y) \wedge \neg P(u,z)) & \\
\rightsquigarrow \forall u (P(a,y) \wedge \neg P(u,f(y))) & \\
\rightsquigarrow P(a,y) \wedge \neg P(u,f(y)) & \text{(Skolem-Normalform in KNF)} \\
\rightsquigarrow \{ \vdash P(a,y) \,;\, P(u,f(y)) \vdash \} & \text{(Klauselmenge)}
\end{array}
$$

(ii) $\exists x \forall y P(x,y) \models \forall y \exists x P(x,y)$

$$\exists x \forall y P(x,y) \rightsquigarrow \forall y P(a,y) \rightsquigarrow P(a,y)$$

$$\neg \forall y \exists x P(x,y) \rightsquigarrow \exists y \forall x \neg P(x,y) \rightsquigarrow \forall x \neg P(x,b) \rightsquigarrow \neg P(x,b)$$

Man erhält die Klauselmenge $\{ \vdash P(a,y) \,;\, P(x,b) \vdash \}$.

Das Import-Export-Theorem 2.20 gilt auch für die Quantorenlogik; somit gilt die Behauptung der Allgemeingültigkeit in (i) genau dann, wenn die Folgerungsbehauptung in (ii) gilt. Dennoch unterscheiden sich die resultierenden Klauselmengen: Bei (ii) steht die Konstante b anstelle des Funktionsterms $f(y)$. Das ist jedoch unproblematisch, da die freien Variablen in jeder Klausel universell verstanden werden.
Obgleich beide Klauselmengen unerfüllbar sind, ist eine Resolutionswiderlegung mit der aussagenlogischen Resolutionsregel nicht möglich, da $P(a,y)$ und $P(u,f(y))$, bzw. $P(a,y)$ und $P(x,b)$, jeweils zwei verschiedene Formeln sind. (Dass dies keine aussagenlogischen, sondern quantorenlogische Formeln sind, ist hier nebensächlich.)

3.6 Unifikation

Um auch für unerfüllbare Klauselmengen wie jene im letzten Beispiel eine Resolutionswiderlegung erhalten zu können, müssen Atome A und B, für die $\{A, \neg B\}$ unerfüllbar ist, in geeigneter Weise durch *Unifikation* vereinheitlicht werden. Dies erreicht man durch Angabe einer Instanz A' von A und einer Instanz B' von B, so dass A' und B' syntaktisch gleich sind.

Definition 3.21 (i) Ist der Ausdruck (d. h. die Formel oder der Term) E' eine Instanz des Ausdrucks E (d. h. $E' = E\sigma$, für eine Substitution σ), dann heißt E *allgemeiner* als E'. *allgemeiner*

(ii) Sind σ, τ zwei Substitutionen, dann heißt σ *allgemeiner* als τ, wenn es eine Substitution ϑ gibt, so dass $\sigma\vartheta = \tau$.

(iii) Es ist $\mathrm{dom}(\vartheta) \coloneqq \{x \mid x\vartheta \neq x\}$ und $\mathrm{ran}(\vartheta) \coloneqq \{y_i \mid y_i \text{ kommt in } x_i\vartheta \text{ vor}\}$, wobei $x_i \in \mathrm{dom}(\vartheta)$.

(iv) Eine Substitution ϑ heißt *Variablensubstitution*, falls alle $x_i\vartheta$ für $x_i \in \mathrm{dom}(\vartheta)$ Variablen sind. *Variablensubstitution*

(v) Eine Variablensubstitution ϑ ist eine *Umbenennung*, falls ϑ die Variablen eineindeutig aufeinander abbildet. (Eine Umbenennung ist also eine Permutation von Variablen.) *Umbenennung*

(vi) Ist ϑ eine Umbenennung, so heißt $E\vartheta$ *Variante* von E. *Variante*

Beispiele. (i) Es ist x allgemeiner als $g(a, h(c))$, denn für $[x/g(a, h(c))]$ ist $g(a, h(c))$ eine Instanz von x.

(ii) Es ist $f(x, y)$ allgemeiner als $f(x, x)$, denn $f(x, y)[y/x] = f(x, x)$.

(iii) $[x/y]$ ist allgemeiner als $[x/a, y/a]$, da $[x/y][y/a] = [x/a, y/a]$.

(iv) Es ist σ allgemeiner als σ für jede Substitution σ, da $\sigma\varepsilon = \sigma$ für die leere Substitution ε.

(v) $[x/y]$ ist *nicht* allgemeiner als $[x/a]$.

Angenommen, es gäbe eine Substitution ϑ, so dass die Bindung x/a in $[x/y]\vartheta$ enthalten ist. Dann muss y/a in ϑ enthalten sein, und damit $y \in \mathrm{dom}([x/y]\vartheta)$. Folglich kann es keine Substitution ϑ mit $[x/y]\vartheta = [x/a]$ geben.

(vi) $[x/z, y/z]$ ist eine Variablensubstitution.

(vii) $[x/z, z/y, y/x]$ ist eine Umbenennung.

Die leere Substitution ε ist ebenfalls eine Umbenennung.

(viii) Es ist $f(x, y)$ eine Variante von $f(y, x)$, denn $f(y, x)[y/x, x/y] = f(x, y)$.

(ix) Es ist $f(x, z)$ eine Variante von $f(x, y)$, denn $f(x, y)[y/z, z/y] = f(x, z)$.

Die Bindung z/y ist nötig, damit die angegebene Substitution eine Umbenennung gemäß unserer Definition ist.

(x) Es ist $f(x, x)$ *keine* Variante von $f(x, y)$.

Wäre es eine Variante, dann müsste $f(x, y)\vartheta = f(x, x)$ für eine Umbenennung ϑ mit Bindung y/x gelten. Da ϑ eine Umbenennung ist, muss ϑ aber außerdem eine Bindung x/y mit $x \neq y$ enthalten. Doch dann wäre $f(x, y)\vartheta = f(y, x) \neq f(x, x)$.

Definition 3.22 (i) Eine Substitution σ heißt *Unifikator* von zwei Atomen A und B, wenn gilt: $A\sigma = B\sigma$ (d. h., wenn die beiden Ausdrücke $A\sigma$ und $B\sigma$ syntaktisch gleich sind). *Unifikator*

(ii) Gibt es einen Unifikator σ von A und B, dann heißen die Atome A und B *unifizierbar*. *unifizierbar*

Ist $\Gamma = \{A, B\}$ für zwei Atome A und B mit Unifikator σ, so sagen wir auch, dass σ die Menge Γ unifiziert.

Beispiel. Die beiden Atome $P(a, x)$ und $P(y, b)$ sind unifizierbar: die Substitution $\sigma = [x/b, y/a]$ ist ein Unifikator, da $P(a, x)\sigma = P(y, b)\sigma$.

Definition 3.23 Die *quantorenlogische Resolutionsregel* ist

quantorenlogische Resolutionsregel

$$\frac{X_1 \vdash Y_1, A \qquad B, X_2 \vdash Y_2}{(X_1, X_2 \vdash Y_1, Y_2)\sigma}\,(\mathcal{R})$$

wobei σ ein Unifikator von A und B ist, und die Mengen der freien Variablen der beiden Prämissen disjunkt sein müssen.

Beispiel. Die beiden Klauseln $Q(x) \vdash P(a,x)$ und $P(y,b) \vdash R(y)$ haben keine gemeinsamen Variablen, und die Substitution $[x/b, y/a]$ unifiziert $P(a,x)$ und $P(y,b)$. Somit ist

$$[x/b, y/a]\,\frac{Q(x) \vdash P(a,x) \qquad P(y,b) \vdash R(y)}{Q(b) \vdash R(a)}\,(\mathcal{R})$$

eine korrekte Anwendung der quantorenlogische Resolutionsregel. (Den verwendeten Unifikator haben wir neben dem Regelstrich notiert.)

Die freien Variablen in den Prämissen werden universell verstanden, und dürfen somit beliebig substituiert werden. Ist σ eine Substitution, die A und B unifiziert, so führt die Anwendung des Unifikators σ auf die Formeln in der quantorenlogischen Resolutionsregel der Form nach auf die aussagenlogische Resolutionsregel:

$$\frac{\begin{array}{cc} X_1 \vdash Y_1, A & B, X_2 \vdash Y_2 \\ \big\downarrow \sigma & \big\downarrow \sigma \end{array}}{\begin{array}{cc} X_1\sigma \vdash Y_1\sigma, A\sigma(=B\sigma) & B\sigma(=A\sigma), X_2\sigma \vdash Y_2\sigma \end{array}}$$
$$\overline{\qquad X_1\sigma, X_2\sigma \vdash Y_1\sigma, Y_2\sigma \qquad}$$

(Weil die Formeln hier Formeln in der Sprache der Quantorenlogik sind, erhält man nicht die eigentliche aussagenlogische Resolutionsregel, da diese nur für aussagenlogische Formeln angegeben wurde. Das ist hier aber nicht wesentlich.)

Beispiel. Für $\sigma = [x/b, y/a]$ als Unifikator von $P(a,x)$ und $P(y,b)$ erhält man:

$$\frac{\begin{array}{cc} Q(x) \vdash P(a,x) & P(y,b) \vdash R(y) \\ \big\downarrow \sigma & \big\downarrow \sigma \end{array}}{\begin{array}{cc} Q(b) \vdash P(a,b) & P(a,b) \vdash R(a) \end{array}}$$
$$\overline{\qquad Q(b) \vdash R(a) \qquad}\,(\mathcal{R})$$

Da freie Variablen universell verstanden werden, bedeutet die Forderung disjunkter Mengen freier Variablen der beiden Prämissen bei $(\mathcal{R})$ keine Einschränkung der Allgemeinheit. Die Forderung kann immer durch geeignete Ersetzungen der freien Variablen erfüllt werden, durch die freie Variablen in verschiedenen Klauseln *separiert* werden. (Vergleiche auch Korollar 3.20 (ii).)

Separierung freier Variablen

Betrachtet man z. B. die beiden Klauseln

$$\vdash P(a), Q(x) \quad \text{und} \quad P(a) \vdash R(x)$$

dann erhält man per (inkorrekter) Resolution

$$\frac{\vdash P(a), Q(x) \qquad P(a) \vdash R(x)}{\vdash Q(x), R(x)} \lightning$$

die Klausel $\vdash Q(x), R(x)$ als Resolvente; der Resolutionsschritt ist inkorrekt, da x in beiden Prämissen als freie Variable vorkommt. Separiert man die freien Variablen stattdessen zuerst, z. B. durch die Umbenennung

$$P(a) \vdash R(x)[x/y] = P(a) \vdash R(y)$$

so erhält man als Resolvente die Klausel $\vdash Q(x), R(y)$. Die beiden Resolventen sind nicht logisch äquivalent, da

$$Q(x) \vee R(y) \vDash Q(x) \vee R(x) \quad \text{aber} \quad Q(x) \vee R(x) \nvDash Q(x) \vee R(y).$$

Das Problem besteht darin, dass die Variable x in beiden Ausgangsklauseln vorkommt, obgleich die beiden Klauseln voneinander unabhängig sind. Es handelt sich deshalb eigentlich nicht um zwei Vorkommen derselben Variable x, sondern um zwei verschiedene Variablen mit demselben Namen. Diese Unterscheidung zweier Variablen geht aber im obigen (inkorrekten) Resolutionsschritt verloren.
Um stets die allgemeinsten Resolventen zu erhalten, werden Variablen in den Prämissen immer erst durch Umbenennung separiert.

Beispiele. (i) Jetzt kann das Beispiel (i) bzw. (ii) von S. 38 fortgesetzt werden:

$$\{\vdash P(a, y)\ ;\ P(u, f(y)) \vdash\} \overset{\text{Separierung der Variablen}}{\rightsquigarrow} \{\vdash P(a, y)\ ;\ P(u, f(v)) \vdash\}$$

Resolutionswiderlegung:

$$[y/f(v), u/a] \frac{\vdash P(a, y) \qquad P(u, f(v)) \vdash}{\vdash} (\mathcal{R})$$

Beziehungsweise für $\{\vdash P(a, y)\ ;\ P(x, b)) \vdash\}$:

$$[x/a, y/b] \frac{\vdash P(a, y) \qquad P(x, b) \vdash}{\vdash} (\mathcal{R})$$

(ii) $\exists x(P(x) \wedge \neg Q(x)), \forall x(P(x) \to R(x)) \vDash \exists x(R(x) \wedge \neg Q(x))$

$\exists x(P(x) \wedge \neg Q(x))$	$\forall x(P(x) \to R(x))$	$\neg\exists x(R(x) \wedge \neg Q(x))$
$\rightsquigarrow \{Q(a) \vdash\ ;\ \vdash P(a)\}$	$\rightsquigarrow \forall x(\neg P(x) \vee R(x))$	$\rightsquigarrow \forall x(\neg R(x) \vee Q(x))$
	$\rightsquigarrow \{P(x) \vdash R(x)\}$	$\rightsquigarrow \{R(y) \vdash Q(y)\}$

Man kann separat skolemisieren, da die Variablen separiert werden können.

Als Klauselmenge erhält man $\{Q(a) \vdash\ ;\ \vdash P(a)\ ;\ P(x) \vdash R(x)\ ;\ R(y) \vdash Q(y)\}$.

Resolutionswiderlegung:

$$[y/a] \frac{\vdash P(a) \qquad [x/y] \dfrac{P(x) \vdash R(x) \qquad R(y) \vdash Q(y)}{P(y) \vdash Q(y)} (\mathcal{R})}{\varepsilon \dfrac{\vdash Q(a) \qquad Q(a) \vdash}{\vdash} (\mathcal{R})}$$

Die quantorenlogische Resolutionsregel $(\mathcal{R})$ allein reicht nicht aus, um z. B. aus den beiden Klauseln

$$\vdash P(x), P(y) \quad \text{und} \quad P(z), P(u) \vdash$$

die leere Klausel ableiten zu können, obgleich die entsprechende Menge

$$\{P(x) \vee P(y), \neg P(z) \vee \neg P(u)\}$$

unerfüllbar ist. Denn jeder Resolutionsschritt erzeugt hier wieder eine Klausel mit zwei Atomen, z. B.

$$[y/z]\,\frac{\vdash P(x), P(y) \qquad P(z), P(u) \vdash}{P(u) \vdash P(x)}\,(\mathcal{R})$$

Da in jedem Resolutionsschritt die Mengen der freien Variablen der beiden Prämissen disjunkt sein müssen, könnte man nur mit geeigneten Varianten fortfahren; z. B. so:

$$[y/u]\,\frac{\vdash P(v), P(y) \qquad [y/z]\,\dfrac{\vdash P(x), P(y) \qquad P(z), P(u) \vdash}{P(u) \vdash P(x)}\,(\mathcal{R})}{\vdash P(v), P(u)}\,(\mathcal{R})$$

wobei $\vdash P(v), P(u)$ eine Variante der ersten Klausel $\vdash P(x), P(y)$ ist. Dies führt jedoch ebenfalls nur zu einer Klausel mit zwei Atomen, die in diesem Fall die Form der ersten Ausgangsklausel hat. Entsprechend erhält man auch in allen anderen Fällen stets Klauseln mit zwei Atomen.
Um ein korrektes Resolutionsverfahren zu erhalten, müssen deshalb zusätzlich sogenannte *Faktoren* von Klauseln gebildet werden können.

Definition 3.24 Ein *Faktor* S' einer Klausel S ist das Ergebnis der Anwendung einer Substitution auf S, durch die mehrere Atome in S unifiziert werden. *Faktor*

Die Ableitung von Faktoren ermöglichen wir durch Hinzunahme folgender Regel.

Definition 3.25 Die *Faktorisierungsregel* $(\mathcal{F})$ ist als Regelpaar für faktorisierbare Atome A, B im Antezedens bzw. Sukzedens gegeben durch: *Faktorisierungsregel*

$$\frac{X, A, B \vdash Y}{(X, A \vdash Y)\sigma}\,(\mathcal{F}) \qquad\qquad \frac{X \vdash A, B, Y}{(X \vdash A, Y)\sigma}\,(\mathcal{F})$$

wobei σ ein Unifikator von A und B ist.

Die Faktorisierungsregel ist eine Substitutionsregel, bei deren Anwendung A und B durch Unifikation identifiziert werden. Dies ist dadurch gerechtfertigt, dass Variablen in Klauseln universell aufgefasst werden.

Beispiel. Mithilfe der Faktorisierung ist es möglich, eine Resolutionswiderlegung von

$$\{\vdash P(x), P(y)\,;\; P(z), P(u) \vdash\}$$

zu erzeugen:

$$[x/z]\,\frac{[y/x]\,\dfrac{\vdash P(x), P(y)}{\vdash P(x)}\,(\mathcal{F}) \qquad [u/z]\,\dfrac{P(z), P(u) \vdash}{P(z) \vdash}\,(\mathcal{F})}{\vdash}\,(\mathcal{R})$$

Es ist auch möglich, $(\mathcal{R})$ und $(\mathcal{F})$ in einer einzigen Regel zusammenzufassen, indem man erlaubt, in einem Schritt mehrere Paare von Formeln zu unifizieren und zu resolvieren.

Definition 3.26 Die *verallgemeinerten Resolutionsregel* $(\mathcal{R}+\mathcal{F})$ lautet: *verallgemeinerte Resolutionsregel*

$$\frac{X_1 \vdash Y_1, A, B \qquad C, D, X_2 \vdash Y_2}{(X_1, X_2 \vdash Y_1, Y_2)\sigma}\,(\mathcal{R}+\mathcal{F})$$

falls σ die Menge $\{A, B, C, D\}$ unifiziert.

Beispiel. Für die Klauselmenge aus dem vorherigen Beispiel genügt eine Anwendung der verallgemeinerten Resolutionsregel, um die leere Klausel abzuleiten:

$$[y/x, u/z][z/x]\,\frac{\vdash P(x), P(y) \qquad P(z), P(u) \vdash}{\vdash}\,(\mathcal{R}+\mathcal{F})$$

Beispiel. Angenommen, es gelten die folgenden Aussagen:

(i) Kein Barbier rasiert jemanden, der sich selbst rasiert.

(ii) Jeder Barbier rasiert alle, die sich nicht selbst rasieren.

Um (i) und (ii) durch quantorenlogische Formeln auszudrücken, verwenden wir das einstellige Relationszeichen B und das zweistellige Relationszeichen R mit folgender Interpretation $\mathcal{I}$:

$$\mathcal{I}(B) = \{x \mid x \text{ ist ein Barbier}\}$$
$$\mathcal{I}(R) = \{\langle x, y\rangle \mid x \text{ rasiert } y\}$$

Die Variablen stehen für beliebige Gegenstände eines gegebenen Gegenstandsbereichs, wobei wir hier den Gegenstandsbereich auf Menschen (inkl. Barbiere) einschränken. Die Aussagen (i) und (ii) lassen sich dann durch die folgenden quantorenlogischen Formeln ausdrücken:

(i*) $\neg\exists x(B(x) \wedge \exists y(R(y,y) \wedge R(x,y)))$

(ii*) $\forall x(B(x) \rightarrow \forall y(\neg R(y,y) \rightarrow R(x,y)))$

Wir möchten wissen, ob es unter den Prämissen (i) und (ii) überhaupt Barbiere geben kann. Wir möchten also herausfinden, ob die Aussage

(iii) Es gibt keine Barbiere

aus (i) und (ii) logisch folgt. Dazu drücken wir auch (iii) noch durch eine quantorenlogische Formel aus:

(iii*) $\neg\exists x B(x)$

Nun wollen wir die Behauptung

$$\neg\exists x(B(x)\wedge\exists y(R(x,x)\wedge R(x,y))), \forall x(B(x)\rightarrow\forall y(\neg R(y,y)\rightarrow R(x,y))) \models \neg\exists x B(x)$$

zeigen, d. h. die Unerfüllbarkeit von

$$\{\neg\exists x(B(x)\wedge\exists y(R(y,y)\wedge R(x,y))), \forall x(B(x)\rightarrow\forall y(\neg R(y,y)\rightarrow R(x,y))), \neg\neg\exists x B(x)\}.$$

Dazu bilden wir zunächst die entsprechenden Klauselmengen:

(i*) $\neg\exists x(B(x) \wedge \exists y(R(y,y) \wedge R(x,y))) \rightsquigarrow \forall x\neg(B(x) \wedge \exists y(R(y,y) \wedge R(x,y)))$
$\rightsquigarrow \forall x(\neg B(x) \vee \neg\exists y(R(y,y) \wedge R(x,y)))$
$\rightsquigarrow \forall x(\neg B(x) \vee \forall y\neg(R(y,y) \wedge R(x,y)))$
$\rightsquigarrow \forall x(\neg B(x) \vee \forall y(\neg R(y,y) \vee \neg R(x,y)))$
$\rightsquigarrow \forall x\forall y(\neg B(x) \vee \neg R(y,y) \vee \neg R(x,y))$
$\rightsquigarrow \{B(x), R(y,y), R(x,y) \vdash\}$

(ii*) $\forall x(B(x) \to \forall y(\neg R(y,y) \to R(x,y))) \rightsquigarrow \forall x\forall y(B(x) \to (\neg R(y,y) \to R(x,y)))$
$\rightsquigarrow B(x) \to (\neg R(y,y) \to R(x,y))$
$\rightsquigarrow \neg B(x) \vee (\neg R(y,y) \to R(x,y))$
$\rightsquigarrow \neg B(x) \vee R(y,y) \vee R(x,y)$
$\rightsquigarrow \{B(x) \vdash R(y,y), R(x,y)\}$

(iii*) $\neg\neg\exists x B(x) \rightsquigarrow \exists x B(x) \rightsquigarrow B(a) \rightsquigarrow \{\vdash B(a)\}$.

Durch die Skolemisierung wurde die Konstante a eingeführt, die für einen konkreten Gegenstand aus unserem Gegenstandsbereich steht; a ist hier also ein Name für einen Menschen.

Nach Separierung der Variablen erhält man die Klauselmenge

$$\{B(x), R(y,y), R(x,y) \vdash\ ;\ B(u) \vdash R(v,v), R(u,v)\ ;\ \vdash B(a)\},$$

für die es eine Resolutionswiderlegung gibt:

$$\dfrac{\dfrac{[u/a]\dfrac{\vdash B(a) \quad B(u) \vdash R(v,v), R(u,v)}{[v/a]\dfrac{\vdash R(v,v), R(a,v)}{\varepsilon\dfrac{\vdash R(a,a)}{\vdash}(\mathcal{F})}(\mathcal{R})} \quad [x/a]\dfrac{\vdash B(a) \quad B(x), R(y,y), R(x,y) \vdash}{[y/a]\dfrac{R(y,y), R(a,y) \vdash}{R(a,a) \vdash}(\mathcal{F})}(\mathcal{R})}{\vdash}$$

Daraus folgt die Behauptung aufgrund Korrektheit von $(\mathcal{R})$ und $(\mathcal{F})$.

Um Vollständigkeit zu erreichen, ist es wichtig, dass man *allgemeinste Unifikatoren* verwendet.

Definition 3.27 Es ist σ ein *allgemeinster Unifikator* (*most general unifier*, kurz: mgu) von A und B, wenn für alle Unifikatoren τ von A und B gilt: $\tau = \sigma\rho$ für eine Substitution ρ. *allgemeinster Unifikator*

Beispiele. (i) Für $P(f(x, g(a,y)))$ und $P(f(b,z))$ ist $[x/b, z/g(a,y)]$ ein allgemeinster Unifikator.

Für die angegebenen Atome ist $[x/b, y/c, z/g(a,c)]$ zwar ein Unifikator, aber *kein* allgemeinster Unifikator, da $[x/b, y/c, z/g(a,c)] = [x/b, z/g(a,y)][y/c]$.

(ii) Für die Menge $\{P(x), P(y)\}$ ist sowohl $[x/y]$ als auch $[y/x]$ ein mgu.

Allgemeinste Unifikatoren sind also nicht eindeutig bestimmt.

(iii) Es ist $\sigma = [x/z, y/z]$ *kein* mgu für $\{P(x), P(y)\}$, da es keine Substitution ρ gibt, so dass $[x/y] = \sigma\rho$.

Insbesondere ist auch $[z/y]$ keine derartige Substitution ρ, da $\sigma[z/y] = [x/y, z/y] \neq [x/y]$.

Theorem 3.28 *Mehrere allgemeinste Unifikatoren von A und B unterscheiden sich nur durch Umbenennung.*

Beweis. Siehe Lassez, Maher & Marriot, 1987, S. 605. QED

Definition 3.29 Sei Γ eine endliche Menge von Termen oder Atomen. Dann ist die *Unterschiedsmenge* U von Γ wie folgt definiert: *Unterschiedsmenge*

Finde die linkeste Stelle, an der nicht alle Ausdrücke in Γ dasselbe Symbol haben, und wähle von jedem Ausdruck in Γ jenen Teilausdruck, der an dieser Stelle beginnt.

Die Menge aller dieser Teilausdrücke ist die Unterschiedsmenge.

Beispiele. (i) Sei $\Gamma = \{P(a, \underline{f(x,y)}, g(z)), P(a, \underline{z}, h(u)), P(a, \underline{x}, y)\}$.

(Die relevanten Teilausdrücke sind unterstrichen.)

Dann ist die Unterschiedsmenge $U = \{f(x,y), z, x\}$.

(ii) Die Unterschiedsmenge für $\Gamma = \{\underline{Q(x)}, \underline{R(x,y)}\}$ ist $U = \{Q(x), R(x,y)\}$.

Wir können uns auf die Behandlung zweielementiger Mengen $\Gamma = \{A, B\}$ für Atome A, B beschränken. Für diese erzeugt der folgende *Unifikationsalgorithmus* einen mgu, sofern A und B unifizierbar sind. *Unifikations-algorithmus*

Unifikationsalgorithmus

Eingabe: Eine Menge $\Gamma = \{A, B\}$ zweier Atome A und B.

Ausgabe: Ein mgu für Γ, falls A und B unifizierbar sind.

(1) Setze $n = 0$ und $\sigma_0 = \varepsilon$, wobei ε die leere Substitution ist.

(2) Wenn $\Gamma\sigma_n$ einelementig ist, dann halte; σ_n ist ein mgu für Γ.

Andernfalls bilde die Unterschiedsmenge U_n von $\Gamma\sigma_n$.

(3) Wenn es eine Variable x und einen Term t in U_n gibt derart, dass die Variable x nicht in t vorkommt, dann setze $\sigma_{n+1} = \sigma_n[x/t]$ (d. h., bilde die Komposition $\sigma_n[x/t]$ mit der bisher erhaltenen Substitution σ_n), erhöhe n um 1, und gehe zu (2).

Andernfalls halte mit der Ausgabe, dass Γ nicht unifizierbar ist.

Bei Anwendung des Unifikationsalgorithmus im Rahmen der Resolution gilt für die Eingabe $\Gamma = \{A, B\}$, dass $\mathrm{FV}(A) \cap \mathrm{FV}(B) = \emptyset$; denn im Resolutionsschritt wird für die Mengen der freien Variablen der beiden Prämissen Disjunktheit gefordert.
Im Fall von Eingabemengen $\Gamma = \{A, B\}$ mit $\mathrm{FV}(A) \cap \mathrm{FV}(B) \neq \emptyset$ können die in den Atomen A und B vorkommenden Variablen erst separiert werden. Diese Variablenseparierung ist (wie bei Klauseln) dadurch gerechtfertigt, dass A und B stets als voneinander unabhängig aufgefasst werden können.

Bemerkung. Die Überprüfung in Schritt (3), ob x in t vorkommt, bezeichnet man als *occur check*. Ohne diesen würde der Algorithmus nicht in jedem Fall terminieren. Zum Beispiel würde der Algorithmus *ohne* occur check für die nicht unifizierbare Menge $\Gamma = \{P(\underline{x}, x), P(\underline{y}, f(x))\}$ Folgendes liefern: *occur check*

(1) $\sigma_0 = \varepsilon$

(2) $U_0 = \{x, y\}$

(3) $\sigma_1 = [x/y]$, $\Gamma\sigma_1 = \{P(y, \underline{y}), P(y, \underline{f(y)})\}$

(2) $U_1 = \{y, f(y)\}$

(3) Die Variable y kommt im Term $f(y)$ vor! Ohne occur check erhält man:
$\sigma_2 = [x/y][y/f(y)] = [x/f(y), y/f(y)]$,
$\Gamma\sigma_2 = \{P(f(y), f(\underline{y})), P(f(y), f(\underline{f(y)}))\}$

(2) $U_2 = \{y, f(y)\}$

(3) Die Variable y kommt im Term $f(y)$ vor! Ohne occur check erhält man:
$\sigma_3 = [x/f(y), y/f(y)][y/f(y)] = [x/f(f(y)), y/f(f(y))]$,
$\Gamma\sigma_3 = \{P(f(f(y)), f(f(\underline{y}))), P(f(f(y)), f(f(\underline{f(y)})))\}$
$\vdots$

Der Algorithmus würde also nicht halten, sondern in Schritt (3) stets $\sigma_{n+1} = \sigma_n[y/f(y)]$ setzen.

Beispiele. (i) $\Gamma = \{P(\underline{f(x)}, g(y)), P(\underline{z}, z)\}$

(1) $\sigma_0 = \varepsilon$

(2) $U_0 = \{f(x), z\}$

(3) $\sigma_1 = [z/f(x)]$, $\Gamma\sigma_1 = \{P(f(x), \underline{g(y)}), P(f(x), \underline{f(x)})\}$

(2) $U_1 = \{g(y), f(x)\}$

(3) Die Unterschiedsmenge U_1 enthält keine Variable als Element, also sind $P(f(x), g(y))$ und $P(z, z)$ nicht unifizierbar.

(ii) $\Gamma = \{P(\underline{f(x)}, y), P(\underline{z}, a)\}$

(1) $\sigma_0 = \varepsilon$

(2) $U_0 = \{f(x), z\}$

(3) $\sigma_1 = [z/f(x)]$, $\Gamma\sigma_1 = \{P(f(x), \underline{y}), P(f(x), \underline{a})\}$

(2) $U_1 = \{y, a\}$

(3) $\sigma_2 = [z/f(x)][y/a]$, $\Gamma\sigma_2 = \{P(f(x), a), P(f(x), a)\} = \{P(f(x), a)\}$

(2) $\Gamma\sigma_2$ einelementig, also Γ unifizierbar mit mgu $\sigma_2 = [z/f(x)][y/a]$.

(iii) $\Gamma = \{P(\underline{x}, x), P(\underline{y}, f(y))\}$

(1) $\sigma_0 = \varepsilon$

(2) $U_0 = \{x, y\}$

(3) $\sigma_1 = [x/y]$, $\Gamma\sigma_1 = \{P(y, \underline{y}), P(y, \underline{f(y)})\}$

(2) $U_1 = \{y, f(y)\}$

(3) Es kommt y in $f(y)$ vor (occur check), also sind $P(x, x)$ und $P(y, f(y))$ nicht unifizierbar.

(iv) $\Gamma = \{P(\underline{a}, x, h(g(z))), P(\underline{u}, h(y), h(y))\}$

(1) $\sigma_0 = \varepsilon$

(2) $U_0 = \{a, u\}$

(3) $\sigma_1 = [u/a]$, $\Gamma\sigma_1 = \{P(a, \underline{x}, h(g(z))), P(a, \underline{h(y)}, h(y))\}$

(2) $U_1 = \{x, h(y)\}$

(3) $\sigma_2 = [u/a][x/h(y)]$, $\Gamma\sigma_2 = \{P(a, h(y), h(\underline{g(z)})), P(a, h(y), h(\underline{y}))\}$

(2) $U_2 = \{g(z), y\}$

(3) $\sigma_3 = [u/a][x/h(y)][y/g(z)]$, $\Gamma\sigma_3 = \{P(a, h(g(z)), h(g(z)))\}$

(2) $\Gamma\sigma_3$ einelementig, also Γ unifizierbar mit mgu $\sigma_3 = [u/a][x/h(y)][y/g(z)] = [u/a, x/h(g(z)), y/g(z)]$.

Bemerkung. In Beispiel (iv) sieht man den Unterschied zwischen simultaner Substitution

$$[x_1/t_1, \ldots, x_n/t_n]$$

und der Hintereinanderausführung von Substitutionen

$$[x_1/t_1] \ldots [x_n/t_n].$$

Im Beispiel ist

$$\sigma_3 = [u/a][x/h(y)][y/g(z)] \neq [u/a, x/h(y), y/g(z)],$$

da

$$\Gamma[u/a, x/h(y), y/g(z)] = \{P(a, h(y), h(g(z))), P(a, h(g(z)), h(g(z)))\} \neq \Gamma\sigma_3.$$

Die simultane Substitution $[u/a, x/h(y), y/g(z)]$ angewendet auf Γ ergäbe also ein anderes Ergebnis als die Hintereinanderausführung $[u/a][x/h(y)][y/g(z)]$ angewendet auf Γ.

Definition 3.31 Eine Substitution ϑ heißt *idempotent*, falls $\vartheta\vartheta = \vartheta$. *idempotent*

Lemma 3.32 *Sei ϑ ein Unifikator von A und B. Dann sind die beiden folgenden Aussagen äquivalent:*

(i) *Es ist ϑ ein idempotenter allgemeinster Unifikator von A und B.*

(ii) *Für jeden Unifikator σ von A und B gilt $\vartheta\sigma = \sigma$.*

Beweis. (i) $\Longrightarrow$ (ii). Sei ϑ ein idempotenter allgemeinster Unifikator von A und B, und σ ein Unifikator von A und B. Dann gibt es eine Substitution ρ, so dass $\sigma = \vartheta\rho$. Damit gilt $\vartheta\sigma = \vartheta(\vartheta\rho) = (\vartheta\vartheta)\rho = \vartheta\rho = \sigma$.

(ii) $\Longrightarrow$ (i). Angenommen, es gilt $\vartheta\sigma = \sigma$ für jeden Unifikator σ von A und B. Dann gilt insbesondere für den allgemeinsten Unifikator ϑ, dass $\vartheta\vartheta = \vartheta$, d. h., ϑ ist idempotent. QED

Nicht jeder allgemeinste Unifikator ist auch idempotent. Für die identischen Atome $P(a)$ und $P(a)$ ist $[x/y, y/x]$ trivialerweise ein mgu, aber $[x/y, y/x][x/y, y/x] = \varepsilon \neq [x/y, y/x]$.

Lemma 3.33 *Eine Substitution ϑ ist genau dann idempotent, wenn* $\mathrm{dom}(\vartheta) \cap \mathrm{ran}(\vartheta) = \emptyset$.

Beweis. Übungsaufgabe. QED

Bemerkung. Durch den Test, ob $\mathrm{dom}(\vartheta) \cap \mathrm{ran}(\vartheta) = \emptyset$ stellt man leicht fest, dass beispielsweise $[x/u, z/f(u, v), y/v]$ idempotent ist, $[x/u, z/f(u, v), v/y]$ aber nicht.

Theorem 3.34 (Korrektheit des Unifikationsalgorithmus)
Seien A und B zwei Atome. Dann gilt:

(i) *Der Unifikationsalgorithmus terminiert immer.*

(ii) *Wenn A und B unifizierbar sind, dann berechnet der Unifikationsalgorithmus einen allgemeinsten Unifikator von A und B.*

(iii) *Wenn A und B nicht unifizierbar sind, dann terminiert der Unifikationsalgorithmus mit der Ausgabe, dass A und B nicht unifizierbar sind.*

Beweis. (i) Die Unterschiedsmenge U_n enthält nur endlich viele Variablen, und in Schritt (3) wird eine Variable eliminiert. Schritt (3) kann somit nur endlich oft durchlaufen werden.

(ii) Wir zeigen zunächst:

($*$) Wenn ϑ ein Unifikator von A und B ist, dann hält der Algorithmus nicht in Schritt (3), und in Schritt (2) gilt jeweils $\sigma_n\vartheta = \vartheta$.

Für $n = 0$ ist $\sigma_0 = \varepsilon$. Somit gilt $\sigma_0\vartheta = \vartheta$.

Angenommen, wir sind in Schritt (2) mit $\sigma_n\vartheta = \vartheta$.

Im Fall $A\sigma_n = B\sigma_n$ hält der Algorithmus in Schritt (2).

Im Fall $A\sigma_n \neq B\sigma_n$ wird die Unterschiedsmenge U_n gebildet, und der Algorithmus geht zu Schritt (3).

Sei $U_n = \{t, t'\}$. Es ist $t\sigma_n\vartheta = t\vartheta = t'\vartheta = t'\sigma_n\vartheta$, da ϑ nach Voraussetzung in ($*$) ein Unifikator von A und B ist. Folglich gilt $t\vartheta = t'\vartheta$.

1. Fall: t ist eine Variable x.
Da $x\vartheta = t'\vartheta$ und $x \neq t'$, kann x nicht in t' vorkommen.
Dann ist $\sigma_{n+1} = \sigma_n[x/t']$.
Wegen $x\vartheta = t'\vartheta$ ist $[x/t']\vartheta = \vartheta$.
Daraus folgt $\sigma_{n+1}\vartheta = (\sigma_n[x/t'])\vartheta = \sigma_n([x/t']\vartheta) = \sigma_n\vartheta = \vartheta$.

2. Fall: t' ist eine Variable. Analog zum 1. Fall.

3. Fall: Weder t noch t' ist eine Variable.
Da $t\vartheta = t'\vartheta$, müssen t und t' mit demselben Zeichen beginnen. Nach Voraussetzung gilt aber $U_n = \{t, t'\}$, d. h., t und t' sind genau die Teilterme, für die $A\sigma_n \neq B\sigma_n$. Folglich kann dieser Fall gar nicht eintreten, und der Algorithmus hält nicht in Schritt (3).

Damit ist ($*$) gezeigt.

Seien nun A und B unifizierbar. Aus ($*$) folgt mit (i), dass der Algorithmus nach n Schritten in Schritt (2) hält. Das Ergebnis σ_n ist ein Unifikator von A und B.

Sei ϑ ein weiterer Unifikator von A und B. Dann folgt mit ($*$), dass $\sigma_n\vartheta = \vartheta$. Da ϑ beliebig, folgt mit Lemma 3.32, dass σ_n ein (sogar idempotenter) allgemeinster Unifikator ist.

(iii) Wenn A und B nicht unifizierbar sind, dann kann der Unifikationsalgorithmus nicht in Schritt (2) halten (sonst wären A und B unifizierbar). Da der Algorithmus wie in (i) gezeigt aber in jedem Fall hält, muss er in diesem Fall in Schritt (3) halten mit der Ausgabe, dass A und B nicht unifizierbar sind. QED

Definition 3.35 Ein Unifikator ϑ zweier Atome A und B heißt *relevant*, falls *relevant*

$$\mathrm{FV}(\vartheta) \subseteq \mathrm{FV}(A) \cup \mathrm{FV}(B)$$

(d. h., falls durch ϑ keine neuen, also weder in A noch in B vorkommenden Variablen eingeführt werden).

Der Unifikationsalgorithmus liefert offensichtlich einen relevanten Unifikator von A und B, sofern A und B unifizierbar sind. Denn in Schritt (3) kommen nur solche Substitutionen hinzu, die Variablen aus A oder B enthalten.

Lemma 3.36 *Sei ϑ ein idempotenter allgemeinster Unifikator von A und B. Dann ist ϑ relevant.*

Beweis. Siehe Apt, 1997, S. 38 (Theorem 2.22). QED

Die Umkehrung gilt nicht, d. h., nicht jeder relevante allgemeinste Unifikator ist auch idempotent. Die Substitution $\vartheta = [x/f(z, y), y/z, z/y]$ ist ein relevanter allgemeinster Unifikator für $\{x, f(y, z)\}$, der jedoch nicht idempotent ist, da $\vartheta\vartheta = [x/f(y, z)] \neq \vartheta$.

Aufgrund des Unifikationsalgorithmus gilt somit Folgendes:

(i) Unifizierbarkeit zweier Atome ist entscheidbar.

(ii) Haben zwei Atome einen Unifikator, dann haben sie auch einen allgemeinsten Unifikator.

(iii) Allgemeinste Unifikatoren können berechnet werden.

(iv) Sind zwei Atome unifizierbar, dann haben sie sogar einen idempotenten allgemeinsten Unifikator.

(v) Der idempotente allgemeinste Unifikator ist außerdem relevant.

Im worst case ist die Laufzeit des Unifikationsalgorithmus exponentiell in der Länge der Eingabe. Sei

$$\Gamma = \{P(\underline{x_1}, \ldots, x_n), P(\underline{f(x_0, x_0)}, \ldots, f(x_{n-1}, x_{n-1}))\}.$$

Dann ist $\sigma_1 = [x_1/f(x_0, x_0)]$ und

$$\begin{aligned}\Gamma\sigma_1 = \{&P(f(x_0, x_0), \underline{x_2}, \ldots, x_n),\\ &P(f(x_0, x_0), \underline{f(f(x_0, x_0), f(x_0, x_0))}, f(x_2, x_2), \ldots, f(x_{n-1}, x_{n-1}))\}.\end{aligned}$$

Dann ist $\sigma_2 = [x_1/f(x_0, x_0), x_2/f(f(x_0, x_0), f(x_0, x_0))]$, und so weiter bis σ_n. Das zweite Atom in $\Gamma\sigma_n$ hat dann für $1 \leq k \leq n$ im k-ten Argument $2^k - 1$ Vorkommen des Funktionszeichens f. Insbesondere hat also das letzte Argument $2^n - 1$ Vorkommen von f. Der occur check in Schritt (3) des Algorithmus hat dann allein für die letzte Substitution σ_n schon eine exponentielle Laufzeit. Auch die Ausgabe von σ_n erfordert exponentielle Laufzeit. Folglich kann es keinen Unifikationsalgorithmus geben, der den mgu *explizit* angibt ohne im worst case exponentielle Laufzeit zu haben.
Es gibt jedoch effizientere Verfahren, wie z. B. den linearzeitlichen Unifikationsalgorithmus von Martelli & Montanari (siehe Apt, 1997, S. 32).

Einen anderen Weg gehen Prolog und SWI-Prolog, die standardmäßig auf den aufwändigen occur check verzichten. Aus theoretischer Sicht ist dies fatal, da dadurch die Korrektheit des Unifikationsalgorithmus, und als Folge davon auch die semantische Korrektheit der Logikprogrammierung, verlorengehen (der occur check kann jedoch sowohl lokal als auch global aktiviert werden; vergleiche die Bemerkung auf S. 8f.).

4 SLD-Resolution

Die Logikprogrammierung im engeren Sinne beruht auf einer eingeschränkten Form des Resolutionsverfahrens, der sogenannten *SLD-Resolution*. Bei dieser werden nicht mehr beliebige Klauseln zugelassen, sondern nur noch sogenannte *Hornklauseln* (benannt nach Alfred Horn, 1918–2001), die neben beliebig vielen negativen Literalen höchstens *ein* positives Literal enthalten dürfen. Zudem dürfen im Resolutionsschritt als Prämissen nur eine Hornklausel mit keinem positiven Literal und eine Hornklausel mit genau einem positiven Literal vorkommen. Letztere muss als Variante aus einer gegebenen Klauselmenge, dem *Logikprogramm*, gewählt werden. Diese Beschränkungen führen zu einfacheren Resolutionsableitungen. Das auf Hornklauseln eingeschränkte Fragment der Quantorenlogik ist jedoch wie diese unentscheidbar. Bei SLD-Resolution rücken zudem die in einer Ableitung berechneten Unifikatoren in den Vordergrund; diese stellen Antworten auf Anfragen dar, die an das jeweilige Logikprogramm gerichtet werden. Logikprogrammierung im Sinne von SLD-Resolution ist Turing-vollständig (vgl. Lloyd, 1993, Theorem 9.6).

4.1 Logikprogramme und SLD-Resolution

Definition 4.1 (i) Eine *Hornklausel* ist eine Klausel mit höchstens einem positiven Literal, d. h., eine Klausel der Form $X \vdash$ oder $X \vdash A$ (wobei A ein positives Literal ist, und die Menge X von Atomen auch leer sein darf). *Hornklausel*

Eine Klausel mit genau einem positiven Literal heißt *definite Hornklausel*. *definite Hornklausel*

Man schreibt mit der *reversen Implikation* '$\leftarrow$' auch $\leftarrow X$ bzw. $A \leftarrow X$. *reverse Implikation*

Motivation für die Schreibweise: $X \vdash A$ kann man als „A, falls X" lesen, $X \vdash$ als „X ist falsch" (bzw. als „Falsum, falls X").

(ii) Eine Klausel der Form $\leftarrow X$ heißt auch *Zielklausel* (kurz: *Ziel*, oder auch *Anfrage*). *Zielklausel*

Das leere Ziel wird mit $\leftarrow$ bezeichnet.

(iii) Eine Klausel der Form $A \leftarrow X$ heißt auch *Regel*; A heißt *Kopf* und X heißt *Rumpf* von $A \leftarrow X$. Eine Klausel der Form $A \leftarrow$ heißt auch *Faktum*. *Regel*

(iv) *Programmklauseln* sind Regeln oder Fakten (d. h. definite Hornklauseln). *Programmklausel*

(v) Ein *Logikprogramm* (kurz: *Programm*) Π ist eine endliche Menge von Programmklauseln. *Logikprogramm*

Definition 4.2 Ein (unrestringierter) *SLD-Resolutionsschritt* hat die Form *SLD-Resolutionsschritt*

$$\frac{\leftarrow X, A \qquad B \leftarrow Y}{(\leftarrow X, Y)\sigma}\,\sigma$$

wobei σ ein Unifikator von A und B ist, und die Mengen der freien Variablen der beiden Prämissen disjunkt sein müssen.

Bemerkungen. (i) In dieser Definition wird nur verlangt, dass σ ein Unifikator von A und B ist. Ein solcher SLD-Resolutionsschritt ist in diesem Sinne unrestringiert. Um Vollständigkeit zu erreichen, müssen jedoch in jedem SLD-Resolutionsschritt allgemeinste Unifikatoren gewählt werden.

Wir gehen im Folgenden davon aus, dass wir den Unifikationsalgorithmus zum Auffinden von σ verwenden, und somit immer allgemeinste Unifikatoren erhalten.

(ii) Mit Blick auf die noch zu besprechende *Auswahlfunktion* (siehe Abschnitt 4.2) werden wir die Klauselrümpfe im Folgenden nicht als Mengen, sondern als Listen auffassen.

Definition 4.3 (i) Eine *SLD-Ableitung* eines Ziels G aus einem Programm Π besteht aus *SLD-Ableitung*

- einer Folge $G_0, G_1, \ldots, G_n$ von Zielen,
- einer Folge $S_1, \ldots, S_n$ von Varianten von Klauseln aus Π,
- einer Folge $\vartheta_1, \ldots, \vartheta_n$ von Substitutionen,

so dass $G_0 = G$ und für alle $i < n$ Folgendes gilt (wobei die Listen von Atomen X_i, Y_i und Z_{i+1} auch leer sein können):

- G_i ist von der Form $\leftarrow X_i, A_i, Y_i$; dies schließt den Fall $\leftarrow$ ein.
- S_{i+1} ist von der Form $B_{i+1} \leftarrow Z_{i+1}$.
- S_{i+1} hat keine Variablen gemeinsam mit G_i oder $G_0\vartheta_1 \ldots \vartheta_i$.
- ϑ_{i+1} ist ein mgu von A_i und B_{i+1}.
- $G_{i+1} = (\leftarrow X_i, Z_{i+1}, Y_i)\vartheta_{i+1}$.

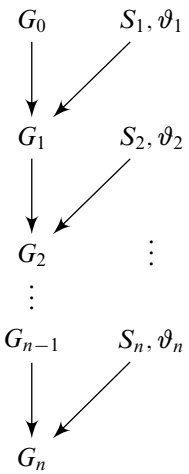
(ii) Die Zahl $n+1$ heißt *Länge* der SLD-Ableitung. *Länge*

(iii) Der Übergang von einem Ziel G_i zu einem neuen Ziel G_{i+1} ist ein *SLD-Resolutionsschritt*. *SLD-Resolutionsschritt*

(iv) Die Klausel S_i heißt i-te *Inputklausel*. *Inputklausel*

(v) Das Atom A_i heißt *ausgewähltes Atom*. *ausgewähltes Atom*

Eine SLD-Ableitung der Länge $n+1$ hat folgende Form:



SLD-Ableitungen können aber auch unendlich sein.

Bemerkung. In einem Resolutionsschritt von G_i nach G_{i+1} geschieht Folgendes:

(1) Im Ziel G_i wird ein Atom A_i ausgewählt.

(2) Aus dem Programm Π wird eine Klausel gewählt.

(3) Durch Separierung der Variablen (falls nötig) erhält man aus der gewählten Programmklausel die Variante S_{i+1}; dies ist die neue Inputklausel $B_{i+1} \leftarrow Z_{i+1}$.

(4) Falls es einen allgemeinsten Unifikator ϑ_{i+1} für den Kopf B_{i+1} der Klausel S_{i+1} und das ausgewählte Atom A_i gibt, dann wird A_i durch den Rumpf Z_{i+1} von S_{i+1} ersetzt, und ϑ_{i+1} wird auf die neue Zielklausel G_{i+1} angewendet.

Definition 4.4 Sei ein Logikprogramm Π und eine Zielklausel $\leftarrow X$ gegeben. Eine *SLD-Widerlegung von* $\leftarrow X$ *relativ zu* Π ist eine SLD-Ableitung, die mit $\leftarrow X$ als linker oberster Annahme beginnt, ansonsten nur (Varianten von) Klauseln aus Π als Annahmen benutzt, und mit der leeren Klausel $\leftarrow$ endet. *SLD-Widerlegung*

Definition 4.5 (i) Eine SLD-Ableitung heißt *erfolgreich*, falls es sich um eine SLD-Widerlegung handelt, d. h., wenn G_n die leere Klausel $\leftarrow$ ist. *erfolgreich*

(ii) Die auf $\mathrm{FV}(G_0)$ eingeschränkte Komposition $\vartheta_1 \ldots \vartheta_n$ der allgemeinsten Unifikatoren in einer erfolgreichen SLD-Ableitung heißt *berechnete Antwortsubstitution* für das Ziel G_0, und $G_0\vartheta_1 \ldots \vartheta_n$ heißt *berechnete Instanz* von G_0. *berechnete Antwortsubstitution*

Ist V eine Menge von Variablen, so bezeichnet $\vartheta \mid V$ die *Einschränkung* von ϑ auf die Variablen in V. *Einschränkung auf Variablen*

(iii) Eine SLD-Ableitung heißt *gescheitert*, falls G_n nicht die leere Klausel ist, und keine (Variante einer) Klausel aus Π für das in G_n ausgewählte Atom anwendbar ist. *gescheitert*

Definition 4.6 Ein *SLD-Beweis* für X aus Π ist eine SLD-Widerlegung von $\leftarrow X$ relativ zu Π. *SLD-Beweis*

Bemerkungen. (i) Notiert man die SLD-Resolutionsschritte gemäß Definition 4.2, so sieht ein *SLD-Beweis* für X_1 aus Π wie folgt aus:

$$\dfrac{\dfrac{\leftarrow X_1 \qquad A_1 \leftarrow Y_1}{\leftarrow X_2}\,\sigma_1 \qquad\qquad A_2 \leftarrow Y_2}{\leftarrow X_3}\,\sigma_2$$

$$\vdots$$

$$\leftarrow$$

Die Substitutionen $\sigma_1, \sigma_2, \ldots$ sind hierbei die im jeweiligen Schritt berechneten Unifikatoren. (Bei Verwendung des Unifikationsalgorithmus sind dies jeweils *allgemeinste* Unifikatoren.)

(ii) Die Abkürzung SLD steht für Folgendes:

S: Es wird eine *selection function* für die Auswahl des zu unifizierenden Atoms in der Zielklausel verwendet.

L: Die Form der Ableitung ist *linear*.
Dies wird besonders deutlich, wenn man die SLD-Ableitung wie folgt notiert:

$$G_0 \xrightarrow{S_1, \vartheta_1} G_1 \xrightarrow{S_2, \vartheta_2} G_2 \quad \cdots \quad G_{n-1} \xrightarrow{S_n, \vartheta_n} G_n \quad \cdots$$

D: Logikprogramme bestehen aus *definiten* Hornklauseln.

Wir lassen in unseren formalen Sprachen $\mathscr{L}$ nun auch 'sprechende' Relationssymbole wie *add*, Funktionszeichen wie s (für *successor*) und Konstanten wie 0 zu, um deren intendierte Interpretation anzudeuten. Wir erweitern hierdurch lediglich das Alphabet; die neuen Zeichen werden aber mit keinerlei bestimmter Bedeutung versehen. Die Bedeutung dieser Zeichen ist operativ bezüglich SLD-Resolution gegeben.

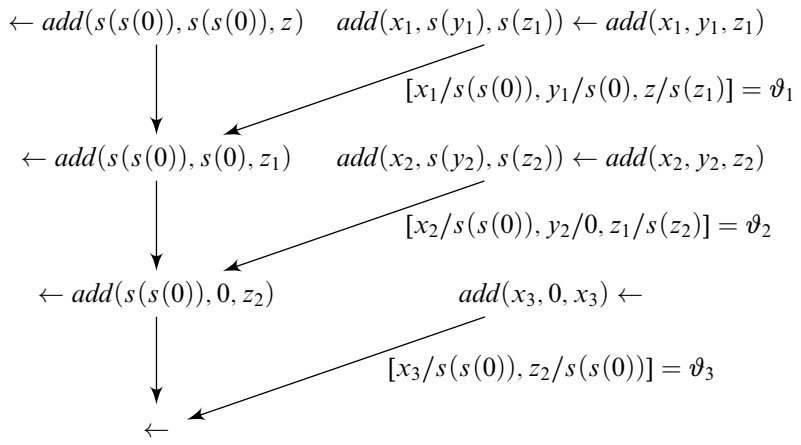
Beispiel. Wir betrachten das Logikprogramm Π_{add} für die Addition, welches aus zwei Programmklauseln, nämlich einem Faktum S_1 und einer Regel S_2, besteht (vgl. S. 10):

$$\begin{aligned} &S_1. && add(x,0,x) \leftarrow \\ &S_2. && add(x,s(y),s(z)) \leftarrow add(x,y,z) \end{aligned}$$

Eine SLD-Widerlegung der Zielklausel

$$\leftarrow add(s(s(0)),s(s(0)),z)$$

relativ zum Programm Π_{add} ist dann beispielsweise:

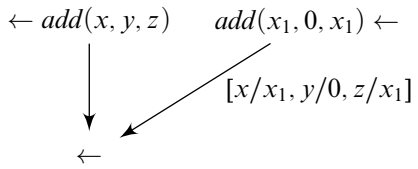


Die Komposition $\vartheta_1\vartheta_2\vartheta_3$ der allgemeinsten Unifikatoren schränken wir nun auf die im Ziel $\leftarrow add(s(s(0)),s(s(0)),z)$ vorkommenden Variablen (d. h. hier auf z) ein, um die berechnete Antwortsubstitution zu erhalten:

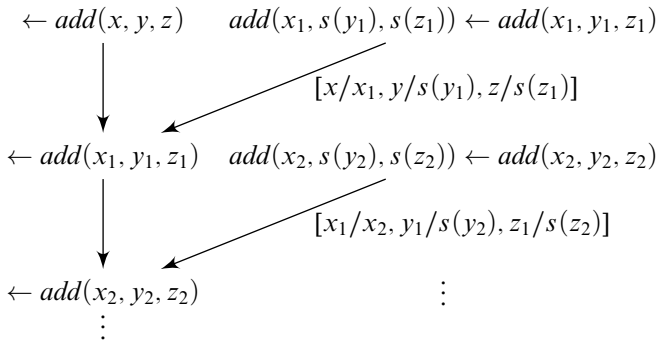
$$\begin{aligned} &\vartheta_1\vartheta_2\vartheta_3 \mid \mathrm{FV}(\leftarrow add(s(s(0)),s(s(0)),z)) \\ &= \vartheta_1\vartheta_2\vartheta_3 \mid \{z\} \\ &= [x_1/s(s(0)), y_1/s(0), z/s(z_1)][x_2/s(s(0)), y_2/0, z_1/s(z_2)]\vartheta_3 \mid \{z\} \\ &= [x_1/s(s(0)), x_2/s(s(0)), y_1/s(0), y_2/0, z/s(s(z_2))][x_3/s(s(0)), z_2/s(s(0))] \mid \{z\} \\ &= [x_1/s(s(0)), x_2/s(s(0)), x_3/s(s(0)), y_1/s(0), y_2/0, z/s(s(s(s(0))))] \mid \{z\} \\ &= [z/s(s(s(s(0))))] \end{aligned}$$

Die berechnete Antwortsubstitution ist also $[z/s(s(s(s(0))))]$; die berechnete Instanz der Zielklausel ist $\leftarrow add(s(s(0)),s(s(0)),s(s(s(s(0)))))$. (Für die intendierte Interpretation ist also $2+2=4$.)

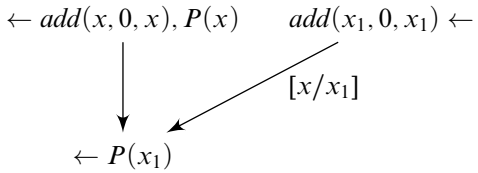
Für die Anfrage $\leftarrow add(x,y,z)$ gibt es relativ zu Π_{add} eine erfolgreiche Ableitung



aber auch unendliche SLD-Ableitungen wie z. B.



Für die Zielklausel $\leftarrow add(x, y, z), P(x)$ gibt es relativ zu Π_{add} z. B. die folgende gescheiterte Ableitung:



SLD-Ableitungen können also erfolgreich, unendlich oder gescheitert sein.

Im Unterschied zur Resolution für beliebige Formeln (die erst in Skolem-Normalform gebracht werden müssen), behandelt SLD-Resolution Formeln, die schon in bestimmter Form sind. Nämlich Programmklauseln, welche die Gesetzmäßigkeiten eines Gegenstandsbereichs beschreiben, und Zielklauseln, die etwas beschreiben, was man über den Gegenstandsbereich wissen will. Die entsprechende Logik ist unentscheidbar, aber im Rahmen der Logikprogrammierung maschinell behandelbar.
SLD-Ableitungen können auch als 'Rückwärts-Schließen' bzw. *zielgerichtetes Räsonieren* interpretiert werden:

- Gegeben als Ziel: X, A (notiert als $\leftarrow X, A$).
- Wir wissen: $B\sigma$ falls $Y\sigma$ für alle σ (notiert als $B \leftarrow Y$).
- Ferner wissen wir: $A\sigma = B\sigma$.
- Als neues Ziel erhält man: $X\sigma$, $Y\sigma$ (notiert als $(\leftarrow X, Y)\sigma$).

Wenn wir also X, A für die Substitution σ zeigen wollen, dann genügt es, $X\sigma$, $Y\sigma$ zu zeigen. Dies schließt die Abfrage von Daten ein. Das Ziel enthält freie Variablen für die zu suchenden Daten; das Programm enthält die Daten in Form von Fakten. Einen prinzipiellen Unterschied zwischen Daten und Programmen gibt es nicht.

Beispiel. Gegeben das Faktum $P(a,b) \leftarrow$, für welche x, y gilt $P(x,y)$?

SLD-Resolution: $\dfrac{\leftarrow P(x,y) \qquad P(a,b) \leftarrow}{\leftarrow}\,[x/a, y/b]$

Antwort: Für $x = a$ und $y = b$.

In einer SLD-Ableitung spielen verschiedene Arten von Nichtdeterminismus eine Rolle, da in jedem Resolutionsschritt vier Entscheidungen zu treffen sind:

(A) Die Auswahl eines Atoms in der aktuellen Zielklausel.

(B) Die Wahl einer Programmklausel für das ausgewählte Atom.

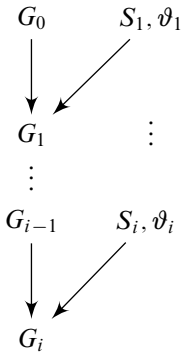
(C) Die Umbenennung der in der gewählten Programmklausel vorkommenden Variablen; also die Auswahl einer Variante.

(D) Die Wahl des allgemeinsten Unifikators.

Der Nichtdeterminismus durch (C) und (D) ist unproblematisch. Die Wahl des allgemeinsten Unifikators (D) ist durch die Verwendung des Unifikationsalgorithmus festgelegt. Und da aus dem vom Unifikationsalgorithmus gelieferten allgemeinsten Unifikator alle anderen allgemeinsten Unifikatoren durch Umbenennung erzeugt werden können (vgl. Theorem 3.28), spielt die konkrete Variante eines allgemeinsten Unifikators für die für ein Ziel G berechnete Antwortsubstitution keine Rolle. Dasselbe gilt auch für die Umbenennung der in der gewählten Programmklausel vorkommenden Variablen (C), da verschiedene Umbenennungen lediglich zu Varianten der berechneten Antwortsubstitutionen führen. Durch bloße Umbenennung können also keine Antworten verlorengehen.
Hingegen ist es für die SLD-Resolution entscheidend, wie der durch (A) und (B) gegebene Nichtdeterminismus gehandhabt wird. Die Auswahl eines Atoms in der aktuellen Zielklausel (A) wird durch *Auswahlfunktionen* festgelegt, die wir als nächstes behandeln. Danach berücksichtigen wir auch die Wahl einer Programmklausel für das ausgewählte Atom (B) unter Verwendung von *SLD-Bäumen*.

4.2 Auswahlfunktionen

Definition 4.7 Eine *Auswahlfunktion* $\mathcal{R}$ ordnet einer gegebenen SLD-Ableitung *Auswahlfunktion*

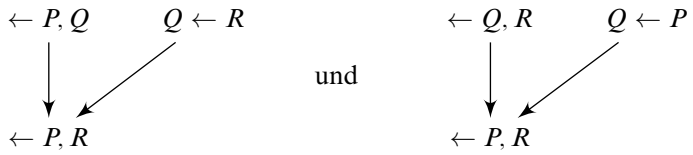


(kurz: $\langle G_0, G_1, \ldots, G_i\rangle, \langle S_1, \ldots, S_i\rangle, \langle \vartheta_1, \ldots, \vartheta_i\rangle$) für $i \geq 0$ ein Atom A_j in der Zielklausel $G_i = (\leftarrow A_1, \ldots, A_k)$ für $1 \leq j \leq k$ zu, sofern G_i nicht die leere Klausel ist. Es ist A_j das *ausgewählte Atom*. *ausgewähltes Atom*

Die Auswahlfunktion $\mathcal{R}$ ist also eine 3-stellige Funktion in die natürlichen Zahlen, die der folgenden Bedingung genügt: Falls $G_i = (\leftarrow A_1, \ldots, A_k)$, für $k \geq 1$, und $j = \mathcal{R}(\langle G_0, G_1, \ldots, G_i\rangle, \langle S_1, \ldots, S_i\rangle, \langle \vartheta_1, \ldots, \vartheta_i\rangle)$, dann muss $1 \leq j \leq k$ gelten.
Wir sagen dann, dass $\mathcal{R}$ das Atom A_j in G_i auswählt.

Definition 4.8 Ist $\mathcal{R}$ die für eine SLD-Ableitung verwendete Auswahlfunktion, so sprechen wir von einer SLD-Ableitung *gemäß* $\mathcal{R}$.

Die Auswahl eines Atoms kann von der gesamten bisherigen SLD-Ableitung abhängen. Dies ermöglicht z. B. die Implementierung einer *first in, first out*-Auswahl, bei der in der aktuellen Zielklausel stets ein Atom ausgewählt wird, das mindestens so oft wie andere Atome in vorherigen Zielklauseln vorkam. Im Fall der beiden SLD-Ableitungen



die beide mit derselben Zielklausel enden, bedeutet dies, dass in der linken Zielklausel das Atom P ausgewählt wird, während in der rechten das Atom R ausgewählt wird. Würde man nur Auswahlfunktionen zulassen, die statt SLD-Ableitungen nur Zielklauseln auf ein Atom (das ausgewählte Atom) abbilden, so würde in beiden Fällen dasselbe Atom ausgewählt werden (sofern nicht gerade eine Zufallsfunktion für die Auswahl verwendet wird).
Prolog verwendet hingegen eine einfache Auswahlfunktion, bei der stets das linkeste Atom in einer Zielklausel ausgewählt wird.
Später werden wir das Resultat erhalten, dass die erfolgreichen SLD-Ableitungen unabhängig von der Auswahlfunktion sind (siehe Abschnitt 5.3). Dies kann auch direkt gezeigt werden, indem man nachweist, dass die Auswahl von Atomen A_i und A_j in zwei aufeinanderfolgenden Zielklauseln stets in umgekehrter Reihenfolge erfolgen kann, also erst A_j und dann A_i ausgewählt wird (bei entsprechender Vertauschung der Inputklauseln); dieses Resultat wird auch als *Switching Lemma* bezeichnet.

4.3 SLD-Bäume

Obgleich die *Existenz* einer erfolgreichen SLD-Ableitung unabhängig von der gegebenen Auswahlfunktion ist (d. h., der durch (A) gegebene Nichtdeterminismus in diesem Sinne unproblematisch ist), ist die Wahl der Programmklausel (Nichtdeterminismus durch (B)) entscheidend für das *Auffinden* einer solchen SLD-Ableitung. Durch (B) wird ein *Suchraum* für SLD-Ableitungen gemäß $\mathcal{R}$, der sogenannte *SLD-Baum*, aufgespannt.

Definition 4.9 Ein *SLD-Baum* für ein Ziel G_0 relativ zu einem Programm Π gemäß einer Auswahlfunktion $\mathcal{R}$ ist ein wie folgt gegebener (nach unten verzweigender) Baum: *SLD-Baum*

(i) Jeder Knoten des Baums is ein (möglicherweise leeres) Ziel.

(ii) Der Wurzelknoten ist G_0.

(iii) Jeder Knoten G_i mit ausgewähltem Atom A_i hat genau einen Nachfolger für jede auf A_i anwendbare Klausel S aus Π.

Der Nachfolger ist eine Resolvente von G_i und S (bzw. von G_i und der gewählten Variante von S) bezüglich A_i.

(iv) Knoten, die ein leeres Ziel sind, haben keine Nachfolger.

Zweige in SLD-Bäumen sind SLD-Ableitungen für G_0 relativ zu Π. Wir notieren diese in der Form

$$G_0 \xrightarrow{S_1, \vartheta_1} G_1 \xrightarrow{S_2, \vartheta_2} G_2 \quad \cdots \quad G_{n-1} \xrightarrow{S_n, \vartheta_n} G_n \quad \cdots$$

von oben (d. h., vom Wurzelknoten G_0 aus) nach unten, wie in den folgenden Beispielen gezeigt.

Definition 4.10 (i) Ein SLD-Baum heißt *erfolgreich*, falls er die leere Klausel enthält. *erfolgreich*
(Zweige, die mit der leeren Klausel enden, markieren wir mit 'success'.)

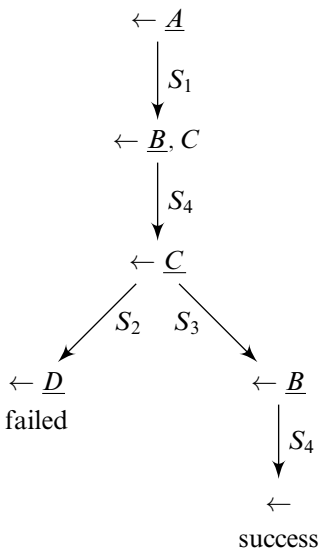
(ii) Ein SLD-Baum heißt *endlich gescheitert*, falls er endlich und nicht erfolgreich ist. *endlich gescheitert*
Dies ist der Fall, wenn alle Zweige gescheiterte SLD-Ableitungen sind.
(Zweige gescheiterter SLD-Ableitungen markieren wir mit 'failed'.)

Beispiel. Wir betrachten zunächst das Logikprogramm

S_1. $A \leftarrow B, C$
S_2. $C \leftarrow D$
S_3. $C \leftarrow B$
S_4. $B \leftarrow$

(in dem nur 0-stellige Relationszeichen, bzw. Aussagesymbole vorkommen).

Sei $\mathcal{R}$ jene Auswahlfunktion, die stets das linkeste Atom (im Folgenden unterstrichen) im Rumpf einer Zielklausel auswählt. Für die Anfrage $\leftarrow A$ relativ zu unserem Programm ist der folgende Baum ein SLD-Baum gemäß $\mathcal{R}$:



Der linke Zweig ist eine gescheiterte SLD-Ableitung, da unser Logikprogramm keine auf die Zielklausel $\leftarrow \underline{D}$ anwendbare Klausel enthält. Der rechte Zweig endet mit der leeren Klausel, ist also eine erfolgreiche SLD-Ableitung, und der SLD-Baum ist somit erfolgreich.

Eine *Tiefensuche* könnte hier zuerst in den linken, gescheiterten Zweig geraten. Durch *backtracking*, d. h., durch das Zurückgehen zu einer früheren Verzweigung und Wahl einer anderen Alternative, kann aber der erfolgreiche Zweig dennoch aufgefunden werden. *backtracking*

Beispiel. Nun betrachten wir das Logikprogramm

S_1. $P(x,z) \leftarrow Q(x,y), P(y,z)$

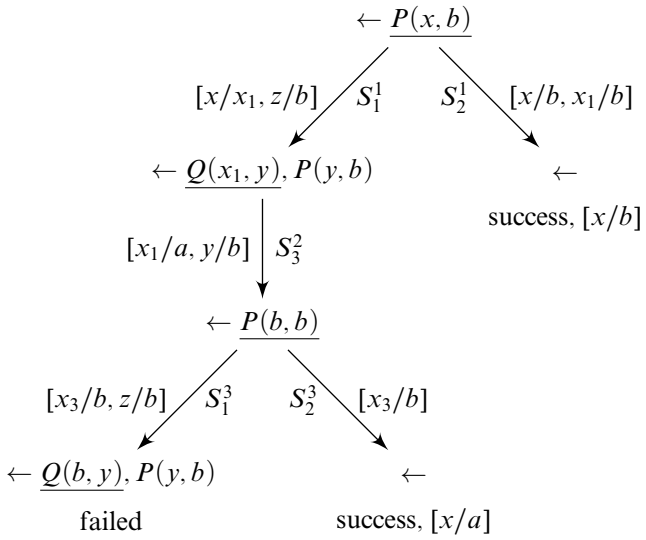
S_2. $P(x,x) \leftarrow$

S_3. $Q(a,b) \leftarrow$

und die Zielklausel

G. $\leftarrow P(x,b)$

Wird durch die Auswahlfunktion stets das *linkeste Atom* gewählt, so erhält man einen SLD-Baum der folgenden Form, wobei die im jeweils i-ten Resolutionsschritt verwendeten Inputklauseln jene Varianten S_1^i, S_2^i und S_3^i der entsprechenden Programmklauseln S_1, S_2 und S_3 sind, bei denen die zur Variablenseparierung umzubenennenden Variablen den Index i erhalten (auf S. 60 sind die jeweiligen Varianten explizit angegeben):



Der SLD-Baum ist endlich und erfolgreich. Die berechneten Antwortsubstitutionen sind

$$\begin{aligned}[x/x_1, z/b][x_1/a, y/b][x_3/b] \mid \{x\} &= [x/a, x_1/a, y/b, z/b][x_3/b] \mid \{x\} \\ &= [x/a, x_1/a, y/b, z/b, x_3/b] \mid \{x\} \\ &= [x/a] \quad \text{(mittlerer Zweig)}\end{aligned}$$

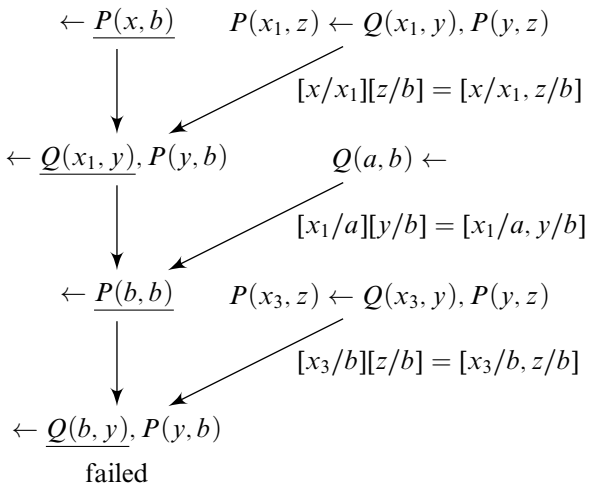
und

$$[x/b, x_1/b] \mid \{x\} = [x/b] \quad \text{(rechter Zweig)}.$$

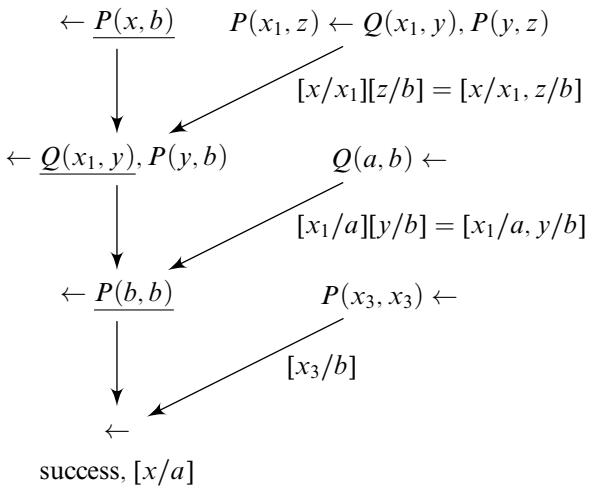
Andere SLD-Bäume unterscheiden sich von dem gezeigten SLD-Baum nur bezüglich Umbenennungen, sind aber der Form nach gleich.

Zur Verdeutlichung geben wir die drei Zweige des SLD-Baums (d. h. die drei SLD-Ableitungen) noch einmal einzeln an, wobei wir auch die bei der Berechnung der allgemeinsten Unifikatoren erzeugten Substitutionen kenntlich machen.

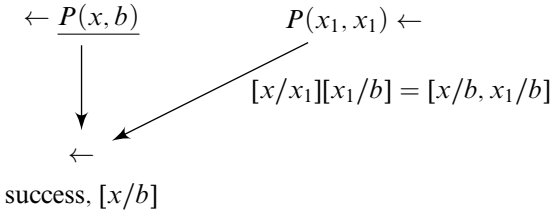
Linker Zweig:



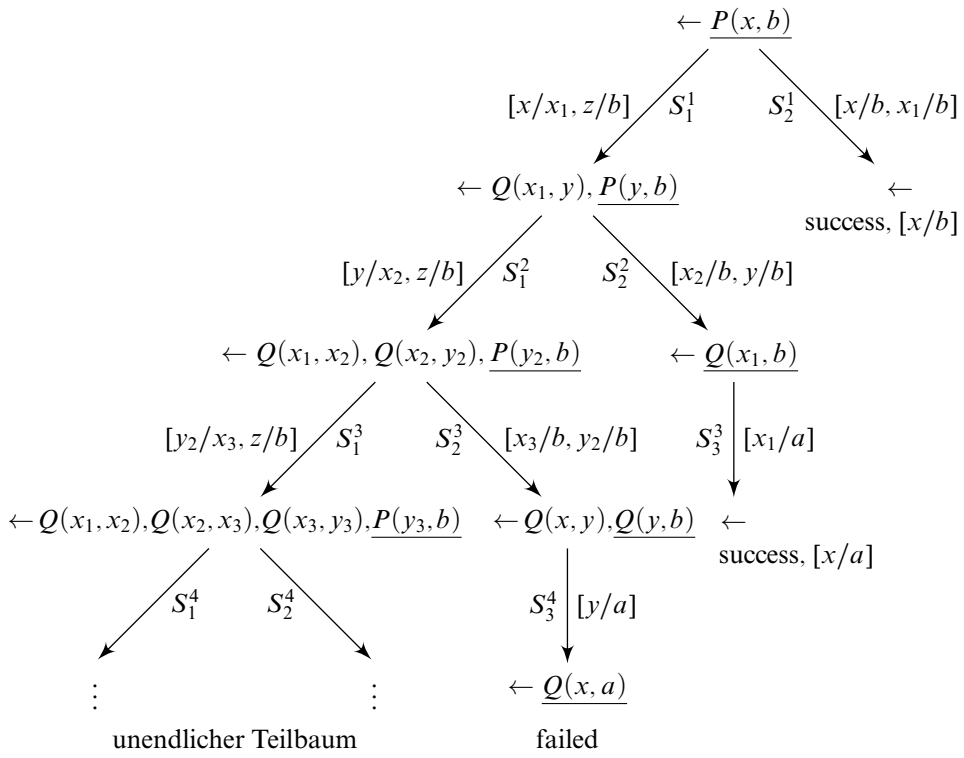
Mittlerer Zweig:



Rechter Zweig:



Wird durch die Auswahlfunktion stets das *rechteste Atom* gewählt, so erhält man einen SLD-Baum der folgenden Form, wobei für die im i-ten Resolutionsschritt verwendeten Varianten S_1^i, S_2^i und S_3^i der entsprechenden Programmklauseln S_1, S_2 und S_3 das oben Gesagte gilt.



Dieser SLD-Baum ist ebenfalls erfolgreich mit denselben berechneten Antwortsubstitutionen $[x/a]$ und $[x/b]$ wie beim ersten SLD-Baum. Im Unterschied zum ersten ist dieser zweite SLD-Baum jedoch unendlich; im linken Zweig können stets neue Zielklauseln erzeugt werden.
Startet eine Tiefensuche hier im linken Zweig, so können die erfolgreichen SLD-Ableitungen rechts nicht mehr aufgefunden werden, da der linke Zweig unendlich ist, und *backtracking* natürlich nur bei bei endlichen Zweigen möglich ist. Bei einer *Breitensuche* werden auch die erfolgreichen SLD-Ableitungen gefunden.

5 Korrektheit und Vollständigkeit der SLD-Resolution

5.1 Semantik der Quantorenlogik

Wir nehmen zu unserem Alphabet noch das logische Zeichen $\top$ (*Verum*) hinzu. *Verum*

Definition 5.1 Eine *Struktur* für eine Sprache $\mathscr{L}$ ist ein geordnetes Paar $\mathfrak{M} = \langle M, \mathcal{I} \rangle$, wobei M eine nicht-leere Menge ist und $\mathcal{I}$ eine Funktion, die jedem Symbol aus $\mathscr{L}$ eine Interpretation zuordnet wie folgt: *Struktur*

(i) $\mathcal{I}(c) \in M$ für Konstanten $c \in \mathscr{L}$ (und ebenso für 0-stellige Funktionszeichen),

(ii) $\mathcal{I}(f) : M^n \to M$, falls $f \in \mathscr{L}$ ein n-stelliges Funktionszeichen ist ($n \geq 1$),

(iii) $\mathcal{I}(R) \subseteq M^n$, falls $R \in \mathscr{L}$ ein n-stelliges Relationszeichen ist ($n \geq 1$),

(iv) $\mathcal{I}(R) \in \{\mathrm{w}, \mathrm{f}\}$, falls $R \in \mathscr{L}$ ein 0-stelliges Relationszeichen, d. h., ein Aussagesymbol ist, und $\mathrm{w} = \emptyset$, $\mathrm{f} = M$.

Die Menge M heißt *Universum* (oder auch *Gegenstandsbereich*, *Individuenbereich*). *Universum*

Man schreibt auch $R^{\mathfrak{M}}$ für $\mathcal{I}(R)$, $f^{\mathfrak{M}}$ für $\mathcal{I}(f)$, $c^{\mathfrak{M}}$ für $\mathcal{I}(c)$, und $\mathfrak{M}$ damit dann als $\langle M, R^{\mathfrak{M}}, \ldots, f^{\mathfrak{M}}, \ldots, c^{\mathfrak{M}} \rangle$.

Beispiel. $\mathscr{L}$ enthalte lediglich die Konstante a und das 1-stellige Funktionszeichen s. Die natürlichen Zahlen können dann durch eine Struktur $\mathfrak{N} = \langle M, \mathcal{I} \rangle$ wie folgt dargestellt werden:

(i) Das Universum M sei $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.

(ii) $\mathcal{I}(a) = 0$, d. h., $\mathcal{I}$ ordnet der Individuenkonstante a das Element $0 \in \mathbb{N}$ zu.

(iii) $\mathcal{I}(s) : M^1 \to M, m \mapsto m+1$, wobei '+' hier die normale Addition ist; d. h., $\mathcal{I}$ ordnet dem 1-stelligen Funktionszeichen s die 1-stellige Funktion $f(m) = m + 1$ zu. (Hier bezeichnet 'f' also eine Funktion auf $\mathbb{N}$; in $\mathscr{L}$ kommt 'f' als Funktionszeichen gar nicht vor.)

In dieser Struktur bezeichnet a die Zahl 0, $s(a)$ die Zahl 1 (da $\mathcal{I}(s)(\mathcal{I}(a)) = f(0) = 1$), $s(s(a))$ die Zahl 2 (da $\mathcal{I}(s)(\mathcal{I}(s)(\mathcal{I}(a))) = f(f(0)) = f(1) = 2$), usw.

Durch eine Struktur $\langle M, \mathcal{I} \rangle$ für eine Sprache $\mathscr{L}$ werden den *Konstanten* in $\mathscr{L}$ also *Gegenstände* in M zugeordnet, und n-stelligen Funktions*zeichen* in $\mathscr{L}$ werden n-stellige *Funktionen* $M^n \to M$ zugeordnet.
Den Grundtermen können somit Gegenstände in M zugeordnet werden. Um beliebige Terme behandeln zu können (also auch solche, die keine Grundterme sind, sondern Variablen enthalten), benötigen wir zusätzlich Variablenbelegungen.

Definition 5.2 (i) Sei $\mathfrak{M} = \langle M, \mathcal{I} \rangle$ eine Struktur. Eine *Variablenbelegung* in $\mathfrak{M}$ ist eine Funktion v, die jeder Variablen x einen Wert $v(x) \in M$ zuordnet. (Also $v : \textsc{Var} \to M$, wobei $\textsc{Var}$ die Menge aller Variablen sei.) *Variablenbelegung*

(ii) Unterscheidet sich eine Variablenbelegung v' höchstens durch die Zuordnung $v'(x) = m \in M$ von einer Variablenbelegung v, so schreiben wir $v[x \mapsto m]$ für v'. Es gilt

$$v[x \mapsto m](y) = \begin{cases} m & \text{falls } x = y, \\ v(y) & \text{falls } x \neq y. \end{cases}$$

Die Variablenbelegung $v[x \mapsto m]$ heißt x-*Variante* von v. x-*Variante*

Beispiel. Wir erweitern die Sprache $\mathscr{L}$ aus dem vorherigen Beispiel um die Variablen x, y, z, z_1, und behalten die dort angegebene Struktur $\mathfrak{N} = \langle \mathbb{N}, \mathcal{I} \rangle$ bei. Dann ist durch

$$v(x) = 4, \quad v(y) = 7, \quad v(z) = 0, \quad v(z_1) = 7$$

eine Variablenbelegung v in $\mathfrak{N}$ gegeben. Die durch die Zuordnungen

$$v'(x) = 4, \quad v'(y) = 11, \quad v'(z) = 0, \quad v'(z_1) = 7$$

gegebene Variablenbelegung v' ist eine y-Variante von v, da $v' = v[y \mapsto 11]$.

Durch Interpretation $\mathcal{I}$ und Variablenbelegung v können nun Termen Gegenstände des Universums zugeordnet werden. Ordnet z. B. die Interpretation $\mathcal{I}$ dem Funktionszeichen s wieder die Nachfolgerfunktion auf den natürlichen Zahlen zu, und gilt für die Variablenbelegung z. B. wieder $v(y) = 7$, so ist dem Term $s(y)$ die Zahl 8 zugeordnet. Durch Kombination von Interpretation und Variablenbelegung erhält man so eine *Termbelegung*.

Definition 5.3 Sei $\mathfrak{M} = \langle M, \mathcal{I} \rangle$ eine Struktur für $\mathscr{L}$, v eine Variablenbelegung in $\mathfrak{M}$ und TERM die Menge aller Terme. Dann ist die *Termbelegung* als Funktion

$$[\![\ ]\!]_v^{\mathfrak{M}} : \text{TERM} \to M$$

wie folgt induktiv definiert:

(i) $[\![x]\!]_v^{\mathfrak{M}} := v(x)$,

(ii) $[\![c]\!]_v^{\mathfrak{M}} := \mathcal{I}(c)$,

(iii) $[\![f(t_1, \ldots, t_n)]\!]_v^{\mathfrak{M}} := \mathcal{I}(f)([\![t_1]\!]_v^{\mathfrak{M}}, \ldots, [\![t_n]\!]_v^{\mathfrak{M}})$.
(Bezeichnet z. B. f' die dem Funktionszeichen f durch $\mathcal{I}(f)$ zugeordnete Funktion, so kann man (iii) schreiben als $[\![f(t_1, \ldots, t_n)]\!]_v^{\mathfrak{M}} := f'([\![t_1]\!]_v^{\mathfrak{M}}, \ldots, [\![t_n]\!]_v^{\mathfrak{M}})$.)

Beispiel. Für unsere Struktur $\mathfrak{N} = \langle \mathbb{N}, \mathcal{I} \rangle$ und die Variablenbelegung v' erhalten wir beispielsweise folgende Termbelegungen (wobei dem 1-stelligen Funktionszeichen s durch $\mathcal{I}$ wieder die Nachfolgerfunktion f auf $\mathbb{N}$ zugeordnet ist):

(i) $[\![a]\!]_{v'}^{\mathfrak{N}} = [\![z]\!]_{v'}^{\mathfrak{N}} = 0$,

(ii) $[\![s(a)]\!]_{v'}^{\mathfrak{N}} = \mathcal{I}(s)([\![a]\!]_{v'}^{\mathfrak{N}}) = \mathcal{I}(s)(0) = f(0) = 1$,

(iii) $[\![s(s(z_1))]\!]_{v'}^{\mathfrak{N}} = \mathcal{I}(s)(\mathcal{I}(s)([\![z_1]\!]_{v'}^{\mathfrak{N}})) = f(f([\![z_1]\!]_{v'}^{\mathfrak{N}})) = f(f(7)) = 9$,

(iv) $[\![s(s(s(x)))]\!]_{v'}^{\mathfrak{N}} = 7$.

Definition 5.4 Nun erweitern wir die Termbelegung $[\![\ ]\!]_v^{\mathfrak{M}}$ zu einer Funktion

$$[\![\ ]\!]_v^{\mathfrak{M}} : \text{FORMEL} \to \{\text{w}, \text{f}\}$$

die jeder Formel in $\mathscr{L}$ einen Wahrheitswert w oder f zuordnet. Der *Wahrheitswert* $[\![A]\!]_v^{\mathfrak{M}}$ einer Formel A in der Struktur $\mathfrak{M}$ unter der Variablenbelegung v in M ist wie folgt über dem Aufbau von Formeln definiert:

Wahrheitswert einer Formel

$$[\![\top]\!]_v^{\mathfrak{M}} := \text{w}.$$

$$[\![R(t_1,\ldots,t_n)]\!]^{\mathfrak{M}}_v := \begin{cases} \text{w} & \text{falls } \langle [\![t_1]\!]^{\mathfrak{M}}_v, \ldots, [\![t_n]\!]^{\mathfrak{M}}_v \rangle \in \mathcal{I}(R), \\ \text{f} & \text{sonst.} \end{cases}$$

$$[\![\neg A]\!]^{\mathfrak{M}}_v := \begin{cases} \text{w} & \text{falls } [\![A]\!]^{\mathfrak{M}}_v = \text{f}, \\ \text{f} & \text{sonst.} \end{cases}$$

$$[\![A \wedge B]\!]^{\mathfrak{M}}_v := \begin{cases} \text{w} & \text{falls } [\![A]\!]^{\mathfrak{M}}_v = \text{w und } [\![B]\!]^{\mathfrak{M}}_v = \text{w}, \\ \text{f} & \text{sonst.} \end{cases}$$

$$[\![A \vee B]\!]^{\mathfrak{M}}_v := \begin{cases} \text{w} & \text{falls } [\![A]\!]^{\mathfrak{M}}_v = \text{w oder } [\![B]\!]^{\mathfrak{M}}_v = \text{w}, \\ \text{f} & \text{sonst.} \end{cases}$$

$$[\![A \to B]\!]^{\mathfrak{M}}_v := \begin{cases} \text{w} & \text{falls } [\![A]\!]^{\mathfrak{M}}_v = \text{f oder } [\![B]\!]^{\mathfrak{M}}_v = \text{w}, \\ \text{f} & \text{sonst.} \end{cases}$$

$$[\![\forall x A]\!]^{\mathfrak{M}}_v := \begin{cases} \text{w} & \text{falls für alle } m \in M \text{ gilt: } [\![A]\!]^{\mathfrak{M}}_{v[x\mapsto m]} = \text{w}, \\ \text{f} & \text{sonst.} \end{cases}$$

$$[\![\exists x A]\!]^{\mathfrak{M}}_v := \begin{cases} \text{w} & \text{falls es ein } m \in M \text{ gibt, so dass } [\![A]\!]^{\mathfrak{M}}_{v[x\mapsto m]} = \text{w}, \\ \text{f} & \text{sonst.} \end{cases}$$

Im Fall 0-stelliger Relationszeichen R wird gemäß Definition 5.1 durch eine Bewertung $\mathcal{I}(R) \in \{\text{w}, \text{f}\}$ einfach ein Wahrheitswert zugeordnet. Die 0-stelligen Relationszeichen entsprechen somit den atomaren aussagenlogischen Formeln (d. h. den Aussagesymbolen).

Beispiele. Wir gehen wieder von unserer Struktur $\mathfrak{N} = \langle \mathbb{N}, \mathcal{I} \rangle$ mit der Variablenbelegung v' aus, erweitern aber $\mathscr{L}$ um das 3-stellige Relationszeichen *add*, das wie folgt interpretiert sei:

$$\mathcal{I}(add) = \{\langle k, l, m \rangle \mid k + l = m, \text{ für } k, l, m \in \mathbb{N}\}.$$

Dann gilt:

(i) $[\![add(s(a), a, s(a))]\!]^{\mathfrak{N}}_{v'} = \text{w}$, da

$$\begin{aligned} [\![add(s(a), a, s(a))]\!]^{\mathfrak{N}}_{v'} &= \langle [\![s(a)]\!]^{\mathfrak{N}}_{v'}, [\![a]\!]^{\mathfrak{N}}_{v'}, [\![s(a)]\!]^{\mathfrak{N}}_{v'} \rangle \\ &= \langle \mathcal{I}(s)([\![a]\!]^{\mathfrak{N}}_{v'}), 0, \mathcal{I}(s)([\![a]\!]^{\mathfrak{N}}_{v'}) \rangle \\ &= \langle f(0), 0, f(0) \rangle \\ &= \langle 1, 0, 1 \rangle \in \mathcal{I}(add). \end{aligned}$$

(ii) $[\![add(s(x), z, y)]\!]^{\mathfrak{N}}_{v'} = \text{f}$, da

$$\begin{aligned} [\![add(s(x), z, y)]\!]^{\mathfrak{N}}_{v'} &= \langle [\![s(x)]\!]^{\mathfrak{N}}_{v'}, [\![z]\!]^{\mathfrak{N}}_{v'}, [\![y]\!]^{\mathfrak{N}}_{v'} \rangle \\ &= \langle \mathcal{I}(s)([\![x]\!]^{\mathfrak{N}}_{v'}), 0, 11 \rangle \\ &= \langle f(4), 0, 11 \rangle \\ &= \langle 5, 0, 11 \rangle \notin \mathcal{I}(add). \end{aligned}$$

(iii) $[\![add(s(a), a, s(a)) \to add(s(x), z, y)]\!]^{\mathfrak{N}}_{v'} = \text{f}$, da

$$[\![add(s(a), a, s(a))]\!]^{\mathfrak{N}}_{v'} = \text{w} \quad \text{und} \quad [\![add(s(x), z, y)]\!]^{\mathfrak{N}}_{v'} = \text{f}.$$

(iv) $[\![\forall x\, add(x, s(a), s(x))]\!]^{\mathfrak{N}}_{v'} = \mathrm{w}$, da

$$\text{für alle } n \in \mathbb{N} \text{ gilt: } [\![add(x, s(a), s(x))]\!]^{\mathfrak{N}}_{v'[x \mapsto n]} = \mathrm{w}.$$

(Es ist $\langle x, 1, x+1\rangle \in \mathcal{I}(add(x, y, z))$ für alle x-Varianten $v'[x \mapsto n]$ der Variablenbelegung v'.)

(v) $[\![\exists x\, add(x, x, s(a))]\!]^{\mathfrak{N}}_{v'} = \mathrm{f}$, da

$$\text{es kein } n \in \mathbb{N} \text{ gibt, so dass } [\![add(x, x, s(a))]\!]^{\mathfrak{N}}_{v'[x \mapsto n]} = \mathrm{w}.$$

(Es ist $\langle x, x, 1\rangle \notin \mathcal{I}(add(x, y, z))$ für alle x-Varianten $v'[x \mapsto n]$ von v'.)

Definition 5.5 Sei $\mathfrak{M}$ eine Struktur für $\mathscr{L}$, A eine Formel in $\mathscr{L}$, und $\Gamma \subseteq \mathscr{L}$ eine Formelmenge.

(i) A ist *gültig in einer Struktur $\mathfrak{M}$ unter einer Variablenbelegung v* (formal: $\mathfrak{M} \vDash_v A$), falls A in $\mathfrak{M}$ unter v den Wahrheitswert w hat (d. h., falls $[\![A]\!]^{\mathfrak{M}}_v = \mathrm{w}$). Es ist also $\mathfrak{M} \vDash_v A$, falls $[\![A]\!]^{\mathfrak{M}}_v = \mathrm{w}$. *gültig in $\mathfrak{M}$ unter v*

(Ist $[\![A]\!]^{\mathfrak{M}}_v = \mathrm{f}$, so schreiben wir $\mathfrak{M} \nvDash_v A$ für „*A gilt nicht in $\mathfrak{M}$ unter v*“. Es ist also $\mathfrak{M} \nvDash_v A$, falls $[\![A]\!]^{\mathfrak{M}}_v = \mathrm{f}$.)

(ii) A ist in einer Struktur $\mathfrak{M}$ *gültig*, falls $[\![A]\!]^{\mathfrak{M}}_v = \mathrm{w}$ für alle Variablenbelegungen v. *gültig in Struktur*

In diesem Fall ist $\mathfrak{M}$ ein *Modell* von A (formal: $\mathfrak{M} \vDash A$). *Modell*

(Falls $\mathfrak{M}$ kein Modell von A ist, schreiben wir auch $\mathfrak{M} \nvDash A$.)

(iii) Entsprechend ist $\mathfrak{M}$ ein *Modell* von Γ (formal: $\mathfrak{M} \vDash \Gamma$), falls für alle $A \in \Gamma$ die Struktur $\mathfrak{M}$ ein Modell von A ist.

(Falls $\mathfrak{M}$ kein Modell von Γ ist, schreiben wir auch $\mathfrak{M} \nvDash \Gamma$.)

(iv) A heißt *allgemeingültig* (formal: $\vDash A$), falls A in jeder Struktur gültig ist. *allgemeingültig*

(v) A heißt *erfüllbar* (oder *konsistent*), falls es eine Struktur $\mathfrak{M}$ und eine Variablenbelegung v gibt, so dass A in $\mathfrak{M}$ unter v gültig ist. *erfüllbar*

Ist A eine geschlossene Formel, dann heißt A *erfüllbar*, falls es eine Struktur gibt, in der A gültig ist, d. h., falls A ein Modell hat. *erfüllbar*

Andernfalls heißt A *unerfüllbar* (oder *inkonsistent* oder *kontradiktorisch*). *unerfüllbar*

(vi) Entsprechend heißt Γ *erfüllbar*, falls es eine Struktur $\mathfrak{M}$ und eine Variablenbelegung v gibt, so dass alle $A \in \Gamma$ in $\mathfrak{M}$ unter v gültig sind.

Andernfalls heißt Γ *unerfüllbar*.

(vii) A heißt *kontingent*, falls A erfüllbar aber nicht allgemeingültig ist. *kontingent*

Definition 5.6 Sei $\Gamma \subseteq \mathscr{L}$ eine Formelmenge und $A \in \mathscr{L}$ eine Formel. Wir schreiben $\mathfrak{M} \vDash_v \Gamma$, falls $\mathfrak{M} \vDash_v A$ für alle $A \in \Gamma$.

(i) Aus Γ *folgt (quantoren-)logisch* A (formal: $\Gamma \vDash A$), falls für alle Strukturen $\mathfrak{M}$ von $\mathscr{L}$ und alle Variablenbelegungen v in $\mathfrak{M}$ gilt: Wenn $\mathfrak{M} \vDash_v \Gamma$, dann $\mathfrak{M} \vDash_v A$. *logische Folgerung*

(ii) Gilt $A \vDash B$ und $B \vDash A$, d. h., sind die Formeln A und B *(quantoren-)logisch äquivalent*, so schreiben wir $A \dashv\vDash B$. *logisch äquivalent*

Entsprechend für Mengen von Formeln: $\Gamma \dashv\vDash \Delta$, falls $\Gamma \vDash \Delta$ und $\Delta \vDash \Gamma$.

Im Unterschied zur Aussagenlogik gilt:

Theorem 5.7 (Unentscheidbarkeit der Quantorenlogik)
Es gibt kein Verfahren, welches für beliebige Formelmengen Γ *und beliebige Formeln* A *entscheidet, ob* $\Gamma \models A$.

Bemerkungen. (i) Wir verwenden das Zeichen '$\models$' sowohl für die Modellbeziehung ($\mathfrak{M} \models A$, bzw. $\mathfrak{M} \models \Gamma$) als auch für die logische Folgerung ($\Gamma \models A$). Die jeweilige Bedeutung ist aber durch den Bezug auf eine Struktur $\mathfrak{M}$ bzw. auf eine Formelmenge Γ eindeutig.

(ii) Für Formeln mit freien Variablen ist die logische Folgerung hier so definiert, dass $\Gamma \models A$ für jede Variablenbelegung gilt. Für geschlossene Formeln kann $\Gamma \models A$ definiert werden durch die Bedingung: Wenn $\mathfrak{M} \models \Gamma$, dann $\mathfrak{M} \models A$ für alle Strukturen $\mathfrak{M}$ von $\mathscr{L}$.

(iii) Um diese Vereinfachung zu erreichen, kann man offene Formeln als allquantifiziert auffassen, d. h., Formeln A mit freien Variablen als $\forall A$ behandeln. Im Kontext von Modellbeziehung und logischer Folgerung kämen dann lediglich geschlossene Formeln vor. Für Klauseln werden wir dies jedoch durch eine gesonderte Notation kenntlich machen (siehe Definition 5.14).

Lemma 5.8 (Substitutionslemma)
Sei $\mathfrak{M}$ *eine Struktur für* $\mathscr{L}$, $v' = v[x \mapsto [\![t]\!]^{\mathfrak{M}}_v]$, x *frei in* A, *und* t *in* A *frei einsetzbar für* x *(d. h.,* t *enthalte keine Variablen, die in* $A[x/t]$ *gebunden werden würden). Dann gilt:*

(i) $[\![s[x/t]]\!]^{\mathfrak{M}}_v = [\![s]\!]^{\mathfrak{M}}_{v'}$.

(ii) $\mathfrak{M} \models_v A[x/t]$ *genau dann, wenn* $\mathfrak{M} \models_{v'} A$.

Beweis. Per Induktion über dem Aufbau von Termen, bzw. Formeln. QED

Das Substitutionslemma besagt Folgendes:

(i) Für eine Struktur $\mathfrak{M}$ und eine Variablenbelegung v ist die Termbelegung eines Terms s, in dem erst x durch t ersetzt wird, gleich der Termbelegung von s für die x-Variante v' von v, bei der x durch die Termbelegung von t für v ersetzt wird.

(ii) Es macht für den Wahrheitswert einer Formel A keinen Unterschied, ob man zuerst einen Term t für eine freie Variable x in A substituiert und dann den Wahrheitswert von $A[x/t]$ bestimmt, oder ob man zuerst die Termbelegung für t bestimmt, dann die Variablenbelegung zu einer x-Variante abändert, welche x die Termbelegung von t zuordnet, um schließlich den Wahrheitswert von A unter dieser x-Variante zu bestimmen.

5.2 Herbrandmodelle

Für die Behandlung von Klauseln sind die nach dem französischen Logiker Jacques Herbrand (1908–1931) benannten *Herbrandmodelle* von besonderer Bedeutung.

Definition 5.9 Das *Herbranduniversum* $U_{\mathscr{L}}$ ist die Menge aller Grundterme von $\mathscr{L}$. Falls $\mathscr{L}$ keine Konstanten enthält, fügen wir eine beliebige Konstante zu $\mathscr{L}$ hinzu, um Grundterme bilden zu können. *Herbranduniversum*

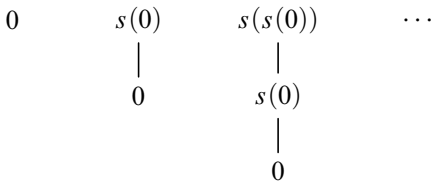
Beispiel. Für die Formelmenge

$$\{P(a), Q(a, h(b)), \forall x(P(x) \to Q(x, x))\}$$

sei die Sprache $\mathscr{L}$ durch die in dieser Menge vorkommenden Zeichen gegeben. Dann ist das Herbranduniversum $U_{\mathscr{L}} = \{a, b, h(a), h(b), h(h(a)), h(h(b)), \ldots\}$.

Das Herbranduniversum ist der Datenbereich eines Logikprogramms. Da Grundterme endlichen Bäumen entsprechen (vgl. die Strukturbäume für Terme, Definition 3.3), haben Logikprogramme nur einen (universellen) Datentyp, nämlich den Typ der endlichen Bäume. Ein Logikprogramm definiert somit Relationen zwischen endlichen Bäumen.

Beispiel. Die Sprache $\mathscr{L}$ enthalte eine Konstante 0 und ein 1-stelliges Funktionszeichen s. Dann können die natürlichen Zahlen $0, 1, 2, \ldots$ als Grundterme $0, s(0), s(s(0)), \ldots$ dargestellt werden, die endlichen Bäumen entsprechen wie folgt:



Definition 5.10 Eine *Herbrandstruktur* ist eine Struktur $\langle U_{\mathscr{L}}, \mathcal{I}\rangle$, wobei *Herbrandstruktur*

(i) $\mathcal{I}(f)(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$, für n-stellige Funktionszeichen f und Grundterme $t_1, \ldots, t_n \in U_{\mathscr{L}}$;

(ii) $\mathcal{I}(c) = c$, für Konstanten (bzw. 0-stellige Funktionszeichen) $c \in U_{\mathscr{L}}$.

Sei $\mathfrak{M}$ eine Herbrandstruktur. Das folgende Lemma zeigt, dass man $[\![t]\!]_v^{\mathfrak{M}}$ (d. h. die Bedeutung des Terms t in $\mathfrak{M}$ unter der Variablenbelegung v) erhält, indem man die in t vorkommenden Variablen x durch die Grundterme $v(x)$ ersetzt.

Lemma 5.11 *Sei $\mathfrak{M}$ eine Herbrandstruktur, v eine Variablenbelegung in $\mathfrak{M}$, und σ die Grundsubstitution $[x_1/v(x_1), \ldots, x_n/v(x_n)]$. Für Terme t mit $\mathrm{FV}(t) \subseteq \{x_1, \ldots, x_n\}$ gilt dann $[\![t]\!]_v^{\mathfrak{M}} = t\sigma$.*

Beweis. Per Induktion über dem Aufbau von t.

Fall 1: Sei t eine Konstante c. Es ist $[\![c]\!]_v^{\mathfrak{M}} = c = c\sigma$.

Fall 2: Sei t eine Variable x_i, für $1 \le i \le n$. Es ist $[\![x_i]\!]_v^{\mathfrak{M}} = v(x_i) = x_i\sigma$.

Fall 3: Sei t ein Funktionsterm $f(t_1, \ldots, t_n)$.

Nach Induktionsannahme gilt $[\![t_i]\!]_v^{\mathfrak{M}} = t_i\sigma$, für $1 \le i \le n$.

Damit gilt $[\![f(t_1, \ldots, t_n)]\!]_v^{\mathfrak{M}} = \mathcal{I}(f)([\![t_1]\!]_v^{\mathfrak{M}}, \ldots, [\![t_n]\!]_v^{\mathfrak{M}}) = f(t_1\sigma, \ldots, t_n\sigma) = t\sigma$. QED

Herbrandstrukturen sind in der Theorie der Logikprogrammierung von großer Bedeutung, da deren Herbranduniversen durch die jeweils betrachteten Logikprogramme stets konkret gegeben sind.

Definition 5.12 Die *Herbrandbasis* $B_{\mathscr{L}}$ ist die Menge aller Grundatome von $\mathscr{L}$. *Herbrandbasis*

Beispiel. Wir betrachten wieder die Formelmenge

$$\{P(a), Q(a, h(b)), \forall x(P(x) \to Q(x, x))\}$$

mit

$$U_{\mathscr{L}} = \{a, b, h(a), h(b), h(h(a)), h(h(b)), \ldots\}.$$

Dann ist die Herbrandbasis

$$B_{\mathscr{L}} = \{P(a), P(b), Q(a, b), P(h(a)), P(h(b)), Q(a, h(a)), Q(a, h(b)), \ldots\}.$$

Bemerkungen. (i) Durch eine Herbrandstruktur $\mathfrak{M} = \langle U_{\mathscr{L}}, \mathcal{I}\rangle$ ist für jedes Grundatom A eine Interpretation $\mathcal{I}(A)$ gegeben, durch die dessen Wahrheitswert festgelegt ist. Somit unterteilt eine Herbrandstruktur $\mathfrak{M}$ die Herbrandbasis $B_{\mathscr{L}}$ in die beiden disjunkten Teilmengen

$$\{R(t_1, \ldots, t_n) \in B_{\mathscr{L}} \mid [\![R(t_1, \ldots, t_n)]\!]^{\mathfrak{M}} = \mathrm{w}\}$$

und

$$\{R(t_1, \ldots, t_n) \in B_{\mathscr{L}} \mid [\![R(t_1, \ldots, t_n)]\!]^{\mathfrak{M}} = \mathrm{f}\}.$$

(ii) Damit können Herbrandstrukturen durch Teilmengen der Herbrandbasis $B_{\mathscr{L}}$ angegeben werden. Man ordnet einer Teilmenge $\mathfrak{J} \subseteq B_{\mathscr{L}}$ die Herbrandstruktur $\langle U_{\mathscr{L}}, \mathcal{I}\rangle$ zu, so dass

$$\langle t_1, \ldots, t_n\rangle \in \mathcal{I}(R) \text{ genau dann, wenn } R(t_1, \ldots, t_n) \in \mathfrak{J}$$

für n-stellige Relationszeichen R in $\mathscr{L}$ und Grundterme $t_1, \ldots, t_n \in U_{\mathscr{L}}$.

(iii) Einer Herbrandstruktur $\langle U_{\mathscr{L}}, \mathcal{I}\rangle$ kann umgekehrt die Menge von Grundatomen

$$\{R(t_1, \ldots, t_n) \in B_{\mathscr{L}} \mid \langle t_1, \ldots, t_n\rangle \in \mathcal{I}(R)\}$$

(für jedes beliebige n-stellige Relationszeichen R in $\mathscr{L}$) zugeordnet werden.

(iv) Für eine gegebene Sprache $\mathscr{L}$ unterscheiden sich Herbrandstrukturen $\langle U_{\mathscr{L}}, \mathcal{I}\rangle$ ausschließlich durch die Interpretation der Relationszeichen, da durch $\mathscr{L}$ sowohl das Herbranduniversum $U_{\mathscr{L}}$ als auch die Interpretation der Funktionszeichen schon festgelegt sind.

Beispiel. Wir gehen wieder von der Formelmenge

$$\{P(a), Q(a, h(b)), \forall x(P(x) \to Q(x, x))\}$$

mit der Herbrandbasis

$$B_{\mathscr{L}} = \{P(a), P(b), Q(a, b), P(h(a)), P(h(b)), Q(a, h(a)), Q(a, h(b)), \ldots\}.$$

aus. Eine Herbrandstruktur $\mathfrak{J} = \langle U_{\mathscr{L}}, \mathcal{I}\rangle$ ist durch das Herbranduniversum

$$U_{\mathscr{L}} = \{a, b, h(a), h(b), h(h(a)), h(h(b)), \ldots\}.$$

und die folgende Interpretation $\mathcal{I}$ gegeben:

(i) $\mathcal{I}(a) = a \in U_{\mathscr{L}}$ und $\mathcal{I}(b) = b \in U_{\mathscr{L}}$;

(ii) $\mathcal{I}(h)(t) = h(t)$ für $t \in U_{\mathscr{L}}$;

(iii) $\mathcal{I}(P) = \{a, h(h(a))\}$ und $\mathcal{I}(Q) = \{\langle t, t\rangle \mid t \in U_{\mathscr{L}}\}$.

Damit ist $\mathfrak{J}$ als (unendliche) Teilmenge von $B_{\mathscr{L}}$ gegeben, d. h.

$$\mathfrak{J} = \{P(a), P(h(h(a))), Q(a,a), Q(b,b), Q(h(a),h(a)), Q(h(b),h(b)), \ldots\} \subset B_{\mathscr{L}}.$$

Definition 5.13 Eine Herbrandstruktur $\mathfrak{M}$ heißt *Herbrandmodell* von A, falls $\mathfrak{M} \models A$, d. h., falls $\mathfrak{M}$ ein Modell von A ist. *Herbrandmodell*

Beispiel. Sei wieder $\Gamma = \{P(a), Q(a,h(b)), \forall x(P(x) \to Q(x,x))\}$ mit

$$B_{\mathscr{L}} = \{P(a), P(b), Q(a,b), P(h(a)), P(h(b)), Q(a,h(a)), Q(a,h(b)), \ldots\}.$$

Beispiele für Herbrandstrukturen (angegeben als Teilmengen von $B_{\mathscr{L}}$) sind:

$$\begin{aligned}
\mathfrak{J}_1 &= \{P(a), P(b), Q(a,b), Q(b,b)\},\\
\mathfrak{J}_2 &= \{P(a), Q(a,a), Q(a,h(b))\},\\
\mathfrak{J}_3 &= \{P(h(h(a))), P(b), Q(a,a), Q(a,h(b))\},\\
\mathfrak{J}_4 &= \{P(a), P(b), Q(a,a), Q(b,b), Q(a,h(b))\}.
\end{aligned}$$

Die Herbrandstruktur $\mathfrak{J}_2$ ist ein Herbrandmodell von Γ:

- Es ist $\mathfrak{J}_2 \models P(a)$ und $\mathfrak{J}_2 \models Q(a,h(b))$, da $P(a), Q(a,h(b)) \in \mathfrak{J}_2$.
- $\mathfrak{J}_2 \models \forall x(P(x) \to Q(x,x))$ gilt, falls $[\![P(x) \to Q(x,x)]\!]^{\mathfrak{J}_2}_{v[x\mapsto t]} = \mathrm{w}$ für alle $t \in U_{\mathscr{L}}$, also falls $[\![P(x)]\!]^{\mathfrak{J}_2}_{v} = \mathrm{f}$ oder $[\![Q(x,x)]\!]^{\mathfrak{J}_2}_{v} = \mathrm{w}$ für alle $t \in U_{\mathscr{L}}$.

  Es ist $[\![P(x)]\!]^{\mathfrak{J}_2}_{v} = \mathrm{w}$ nur für $v(x) = a$, da in $\mathfrak{J}_2$ außer $P(a)$ keine weiteren Grundatome vorkommen, die mit dem Relationszeichen P beginnen.

  In diesem Fall ist auch $[\![Q(x,x)]\!]^{\mathfrak{J}_2}_{v} = \mathrm{w}$, da $Q(a,a) \in \mathfrak{J}_2$.

  Also $[\![P(x) \to Q(x,x)]\!]^{\mathfrak{J}_2}_{v[x\mapsto a]} = \mathrm{w}$.

  Für alle anderen Variablenbelegungen $v(x) = t \in U_{\mathscr{L}}$ ist $[\![P(x)]\!]^{\mathfrak{J}_2}_{v} = \mathrm{f}$ (denn für alle $t \neq a$ ist $P(x)[x/t] \notin \mathfrak{J}_2$), und somit $[\![P(x) \to Q(x,x)]\!]^{\mathfrak{J}_2}_{v} = \mathrm{w}$ für diese Belegungen v.

  Also gilt $[\![P(x) \to Q(x,x)]\!]^{\mathfrak{J}_2}_{v[x\mapsto t]} = \mathrm{w}$ für alle $t \in U_{\mathscr{L}}$.

  Damit ist $\mathfrak{J}_2 \models \forall x(P(x) \to Q(x,x))$.
- Die Herbrandstruktur $\mathfrak{J}_2$ ist also ein Herbrandmodell aller Elemente von Γ; folglich gilt $\mathfrak{J}_2 \models \Gamma$.

$\mathfrak{J}_4$ ist ebenfalls ein Herbrandmodell von Γ.

Die Herbrandstrukturen $\mathfrak{J}_1$ und $\mathfrak{J}_3$ sind keine Herbrandmodelle von Γ, da $\mathfrak{J}_1 \not\models Q(a,h(b))$ (wegen $Q(a,h(b)) \notin \mathfrak{J}_1$) und $\mathfrak{J}_3 \not\models P(a)$ (wegen $P(a) \notin \mathfrak{J}_3$), und folglich $\mathfrak{J}_1 \not\models \Gamma$ und $\mathfrak{J}_3 \not\models \Gamma$.

Definition 5.14 (i) Für Klauseln $S = (A \leftarrow X)$ sei $\overline{S}$ die Formel $\forall(\mathbin{\text{⩓}} X \to A)$, wobei $\mathbin{\text{⩓}} X$ für die Konjunktion aller Atome in X steht.

(ii) Für Fakten $S = (A \leftarrow)$ ist die Formel $\overline{S} = \forall(\mathbin{\text{⩓}}\emptyset \to A) = \forall(\top \to A) = \forall A$.

(iii) Für Zielklauseln $G = (\leftarrow X)$ ist die Formel $\overline{G} = \forall\neg(\mathbin{\text{⩓}} X)$.

Wir schreiben auch $\mathbin{\text{⩓}} G$ für die Konjunktion aller im Rumpf X einer Zielklausel $G = (\leftarrow X)$ vorkommenden Atome.

Der leeren Klausel $\leftarrow$ entspricht die Formel $\neg\top$. In diesem Fall ist $\mathbin{\text{⩓}} G$ $(= \mathbin{\text{⩓}} X)$ die leere Konjunktion, und es gilt $\mathbin{\text{⩓}} G = \top$.

(iv) Für ein Logikprogramm Π ist mit $\overline{\Pi} := \{\overline{S} \mid S \in \Pi\}$ eine Formelmenge gegeben, die ausschließlich geschlossene Formeln enthält.

Beispiel. Für $S = (P(x,y) \leftarrow Q(x,z), P(z,y))$ ist

$$\overline{S} = (\forall x \forall y \forall z (Q(x,z) \wedge P(z,y) \rightarrow P(x,y))).$$

Aufgrund der logischen Äquivalenz

$$\forall x \forall y \forall z (Q(x,z) \wedge P(z,y) \rightarrow P(x,y)) \dashv\vDash \forall x \forall y (\exists z (Q(x,z) \wedge P(z,y)) \rightarrow P(x,y))$$

sind Variablen, die ausschließlich im Rumpf einer Klausel vorkommen, als existenzquantifiziert aufzufassen (vgl. S. 7).

Lemma 5.15 *Sei $\mathfrak{J} \subseteq B_{\mathscr{L}}$ eine Herbrandstruktur und S die Klausel $A \leftarrow B_1, \ldots, B_n$. Dann sind folgende Aussagen äquivalent:*

(i) *$\mathfrak{J}$ ist ein Herbrandmodell von $\overline{S}$, d. h. $\mathfrak{J} \vDash \forall(B_1 \wedge \ldots \wedge B_n \rightarrow A)$.*

(ii) *Für alle Substitutionen σ mit $\mathrm{FV}(S\sigma) = \emptyset$ gilt: Wenn $B_i\sigma \in \mathfrak{J}$ für $1 \leq i \leq n$, dann $A\sigma \in \mathfrak{J}$.*

Beweis. Folgt aus dem Substitutionslemma (Lemma 5.8) mit Lemma 5.11. QED

Theorem 5.16 *Sei Γ eine Klauselmenge. Γ hat ein Modell $\mathfrak{M}$ genau dann, wenn Γ ein Herbrandmodell $\mathfrak{J}$ hat.*

Beweis. Angenommen, Γ hat ein Modell $\mathfrak{M}$. Wir definieren die Herbrandstruktur $\mathfrak{J}$ wie folgt:

$$\mathfrak{J} = \{R(t_1, \ldots, t_n) \in B_{\mathscr{L}} \mid R(t_1, \ldots, t_n) \text{ gilt in } \mathfrak{M}\}.$$

Man zeigt leicht (Übungsaufgabe), dass $\mathfrak{J}$ ein Herbrandmodell von Γ ist.
Die Umkehrung gilt trivialerweise, da jedes Herbrandmodell ein Modell ist. QED

Theorem 5.17 *Sei Γ eine Klauselmenge. Dann sind die beiden folgenden Aussagen äquivalent:*

(i) *Γ ist unerfüllbar.*

(ii) *Γ hat kein Herbrandmodell.*

Beweis. Wir zeigen beide Richtungen per Kontraposition:
$\neg$(ii) $\Longrightarrow$ $\neg$(i). Wenn Γ ein Herbrandmodell hat, dann ist Γ erfüllbar, da ein Herbrandmodell ein Modell ist.
$\neg$(i) $\Longrightarrow$ $\neg$(ii). Wenn Γ erfüllbar ist (d. h. ein Modell hat), dann hat Γ aufgrund Theorem 5.16 ein Herbrandmodell. QED

Die Theoreme 5.16 und 5.17 zeigen, dass zum Nachweis der Unerfüllbarkeit einer Klauselmenge lediglich Herbrandstrukturen betrachtet werden müssen.
Die beiden Theoreme gelten jedoch nur für Klauselmengen. Die Unerfüllbarkeit beliebiger Formelmengen kann im Allgemeinen nicht gezeigt werden, wenn lediglich Herbrandstrukturen betrachtet werden.

Beispiel. Wir betrachten die Formelmenge $\Gamma = \{R(a), \exists x \neg R(x)\}$; die Formel $\exists x \neg R(x)$ ist keine Klausel. Sei $\mathfrak{M} = \langle \{0,1\}, \mathcal{I} \rangle$ eine Struktur mit $\mathcal{I}(a) = 0$ und $\mathcal{I}(R) = \{0\}$. Es ist $[\![R(a)]\!]^{\mathfrak{M}} = \mathrm{w}$, und $[\![\exists x \neg R(x)]\!]^{\mathfrak{M}}_{v} = \mathrm{w}$, da $[\![R(x)]\!]^{\mathfrak{M}}_{v[x \mapsto 1]} = \mathrm{f}$. Somit ist $\mathfrak{M}$ ein Modell von Γ. Die einzigen Herbrandstrukturen für Γ sind $\emptyset$ und $\{R(a)\}$; keine ist ein Herbrandmodell von Γ.

Theorem 5.18 (Skolem–Herbrand–Gödel)
Eine Klauselmenge Γ ist unerfüllbar genau dann, wenn es eine endliche unerfüllbare Menge Γ' von Grundinstanzen von Klauseln aus Γ gibt.

Beweis. Für die Richtung von rechts nach links genügt die Feststellung, dass Gültigkeit unter Substitution abgeschlossen ist. Somit gilt $\Gamma \models \Gamma'$. Ist nun Γ' unerfüllbar, muss auch Γ unerfüllbar sein.

Für die Richtung von links nach rechts nehmen wir an, dass die Klauselmenge Γ unerfüllbar ist.

Nach Theorem 5.16 ist dies genau dann der Fall, wenn Γ kein Herbrandmodell hat.

Da Herbrandstrukturen stets durch Teilmengen der Herbrandbasis, d. h. als Mengen von Grundatomen, angegeben werden können, ist eine Herbrandstruktur $\mathfrak{J}$ genau dann ein Modell einer Klausel S, wenn $\mathfrak{J}$ ein Modell der Menge aller Grundinstanzen von S ist.

Für Klauselmengen Γ ist eine solche Herbrandstruktur somit genau dann ein Modell, wenn $\mathfrak{J}$ ein Modell einer (möglicherweise unendlichen) Menge Γ^* aller Grundinstanzen von Klauseln $S \in \Gamma$ ist.

Somit hat Γ genau dann kein Herbrandmodell, wenn Γ^* kein Modell hat. Letzteres ist mit Theorem 5.16 genau dann der Fall, wenn Γ^* unerfüllbar ist.

Mit dem Endlichkeitssatz (vgl. Theorem 2.25, das auch für die Quantorenlogik gilt) folgt, dass es eine endliche unerfüllbare Menge Γ' von Γ^* gibt. QED

Bemerkung. Neben dem für den semantischen Begriff der Unerfüllbarkeit formulierten Theorem 5.18 von Skolem, Herbrand und Gödel gilt auch das für den syntaktischen Begriff der Beweisbarkeit formulierte *Theorem von Herbrand* (siehe z. B. Troelstra & Schwichtenberg, 2000): Eine Formel A in pränexer Normalform mit Kern $B(x_1, \ldots, x_n)$ ist beweisbar (in einem Beweissystem wie z. B. dem Sequenzenkalkül LK) genau dann, wenn es eine Disjunktion D von Substitutionsinstanzen von $B(x_1, \ldots, x_n)$ gibt, die rein aussagenlogisch beweisbar ist, und A aus D allein durch quantorenlogische (und gegebenenfalls strukturelle) Regeln beweisbar ist.

Das folgende Lemma zeigt, dass für die aus einem Logikprogramm logisch folgenden *Grund*atome lediglich die *Herbrand*modelle des Logikprogramms betrachtet werden müssen.

Lemma 5.19 *Sei Π ein Logikprogramm und A ein Grundatom. Dann sind die beiden folgenden Aussagen äquivalent:*

(i) $\overline{\Pi} \models A$.

(ii) *A gilt in allen Herbrandmodellen von Π.*

Beweis. (i) $\Longrightarrow$ (ii). Ist $\overline{\Pi} \models A$, dann gilt A in allen Modellen $\mathfrak{M}$ von Π, und somit insbesondere auch in allen Herbrandmodellen von Π.

(ii) $\Longrightarrow$ (i). Wir nehmen an, dass $\mathfrak{M} \models \overline{\Pi}$ und $\mathfrak{M} \not\models A$, d. h., $\mathfrak{M}$ sei ein Modell von $\overline{\Pi}$, in dem A nicht gilt. Wir zeigen, dass es dann ein Herbrandmodell $\mathfrak{J}$ von $\overline{\Pi}$ gibt, so dass $A \notin \mathfrak{J}$.

Dazu brauchen wir die folgende Eigenschaft, die aus dem Substitutionslemma (Lemma 5.8) folgt:

Sei B ein Atom und σ eine Substitution, so dass $B\sigma$ eine geschlossene Formel ist, d. h. $\mathrm{FV}(B\sigma) = \emptyset$. Sei $v(x) := [\![x\sigma]\!]^{\mathfrak{M}}$ eine Variablenbelegung für alle Variablen x in B. Dann gilt

$$\mathfrak{M} \models_v B \text{ genau dann, wenn } \mathfrak{M} \models B\sigma. \qquad (*)$$

Sei $\mathfrak{J} := \{B \in B_{\mathscr{L}} \mid \mathfrak{M} \models B\}$. Wir zeigen, dass $\mathfrak{J}$ ein Modell von $\overline{\Pi}$ ist.

Sei $S \in \Pi$ die Klausel $B_0 \leftarrow B_1, \ldots, B_n$ und σ eine Substitution, so dass $\mathrm{FV}(S\sigma) = \emptyset$.

Wir nehmen an, dass $B_i\sigma \in \mathfrak{J}$ für $1 \leq i \leq n$. Wir müssen zeigen, dass dann $B_0\sigma \in \mathfrak{J}$.

Sei $v(x) := [\![x\sigma]\!]^{\mathfrak{J}}$ eine Variablenbelegung für alle Variablen x in S. Aus $(*)$ folgt, dass $\mathfrak{M} \models_v B_i$ für $1 \leq i \leq n$. Wegen $\mathfrak{M} \models \overline{\Pi}$ gilt $\mathfrak{M} \models \overline{S}$. Damit folgt $\mathfrak{M} \models_v B_0$.

Mit $(*)$ gilt dann des Weiteren $\mathfrak{M} \models B_0\sigma$, und nach Definition von $\mathfrak{J}$ ist $B_0\sigma \in \mathfrak{J}$.

Da S und σ beliebig gewählt werden können (wobei σ nur Bindungen x/t für $t \in U_{\mathscr{L}}$ enthalten kann), folgt $\mathfrak{J} \models \overline{\Pi}$.

Aus der Annahme $\mathfrak{M} \not\models A$ folgt $A \notin \mathfrak{J}$. QED

Theorem 5.20 *Sei Γ eine nicht-leere Klauselmenge, und sei S eine Grundinstanz einer Klausel. Dann gilt $\Gamma \models S$ genau dann, wenn es eine endliche Menge Γ' von Grundinstanzen von Klauseln in Γ gibt, so dass $\Gamma' \models S$.*

Beweis. Übungsaufgabe, unter Verwendung von Theorem 5.18. QED

Beispiel. Für die Klauselmenge $\Gamma = \{P(a)\ ;\ P(x) \vdash P(s(x))\}$ und die Grundinstanz $S = (\vdash P(s(s(a))))$ einer Klausel gilt $\Gamma \models S$.

- Der Klausel S entspricht als Formel das Grundatom $P(s(s(a)))$. Aufgrund von Lemma 5.19 gilt $P(s(s(a)))$ in allen Herbrandmodellen von Γ.
- Aufgrund von Theorem 5.20 muss es eine endliche Menge Γ' von Grundinstanzen von Klauseln aus Γ geben, so dass $\Gamma' \models S$.

  Eine solche Menge ist $\Gamma' = \{\vdash P(a)\ ;\ P(a) \vdash P(s(a))\ ;\ P(s(a)) \vdash P(s(s(a)))\}$.

Beispiel. Wir betrachten das folgende Logikprogramm (vgl. das Beispiel auf S. 59f.):

S_1. $P(x,z) \leftarrow Q(x,y), P(y,z)$

S_2. $P(x,x) \leftarrow$

S_3. $Q(a,b) \leftarrow$

Das Herbranduniversum ist $U_{\mathscr{L}} = \{a,b\}$.

Die Herbrandbasis ist

$$B_{\mathscr{L}} = \{Q(a,a), Q(a,b), Q(b,a), Q(b,b), P(a,a), P(a,b), P(b,a), P(b,b)\}.$$

Die Herbrandstruktur $\mathfrak{J} = \langle U_{\mathscr{L}}, \mathcal{I}\rangle = \langle \{a,b\}, \mathcal{I}\rangle$ mit

$$\mathcal{I}(Q) = \{\langle a,b\rangle\} \quad \text{und} \quad \mathcal{I}(P) = \{\langle a,a\rangle, \langle a,b\rangle, \langle b,b\rangle\}$$

ist ein Herbrandmodell. Als Teilmenge der Herbrandbasis $B_{\mathscr{L}}$ geschrieben, ist

$$\mathfrak{J} = \{Q(a,b), P(a,a), P(a,b), P(b,b)\}.$$

Ein weiteres Herbrandmodell (für das gleiche Herbranduniversum und die gleiche Herbrandbasis) ist

$$\mathfrak{F} = \{Q(a,b), Q(b,a), P(a,a), P(a,b), P(b,a), P(b,b)\}.$$

Es ist zwar $Q(b,a)$ nicht aus dem Programm ableitbar, aber für die Modelleigenschaft genügt es, dass alle Programmklauseln in der Herbrandstruktur gültig sind. Das ist für $\mathfrak{F}$ der Fall:

- S_3 ist gültig in $\mathfrak{F}$, da $Q(a,b) \in \mathfrak{F}$,
- S_2 ist gültig in $\mathfrak{F}$, da $\{P(a,a), P(b,b)\} \subset \mathfrak{F}$,
- S_1 ist gültig in $\mathfrak{F}$, da $\{P(a,a), P(a,b), P(b,a), P(b,b)\} \subset \mathfrak{F}$.

Es gilt also für alle Variablenbelegungen v, dass $[\![S_1]\!]_v^{\mathfrak{F}} = \mathrm{w}$, $[\![S_2]\!]_v^{\mathfrak{F}} = \mathrm{w}$ und $[\![S_3]\!]_v^{\mathfrak{F}} = \mathrm{w}$.

Hingegen ist die Menge $\{Q(a,b), Q(b,a), P(a,a), P(a,b), P(b,b)\}$ *kein* Herbrandmodell, da mit $Q(b,a)$ auch $P(b,a)$ enthalten sein müsste, damit $[\![S_1]\!]_v^{\mathfrak{F}} = \mathrm{w}$ für alle Variablenbelegungen v.

Beispiel. Wir betrachten das folgende Logikprogramm (vgl. das Beispiel auf S. 54f.):

$S_1.$ $\quad add(x,0,x) \leftarrow$

$S_2.$ $\quad add(x,s(y),s(z)) \leftarrow add(x,y,z)$

Das Herbranduniversum ist die Menge aller Terme der Form $s^n(0)$, für $n \geq 0$, wobei $s^n(0)$ für die n-fache Anwendung von s auf 0 steht.

Die Herbrandbasis ist

$$B_{\mathscr{L}} = \{add(s^i(0), s^j(0), s^k(0)) \mid i,j,k \geq 0\}.$$

Ein Herbrandmodell ist (als Teilmenge der Herbrandbasis $B_{\mathscr{L}}$) durch die Menge

$$\{add(s^i(0), s^j(0), s^{i+j}(0)) \mid i,j \geq 0\}$$

gegeben.

Bemerkungen. (i) Ist Π ein Logikprogramm mit Herbrandbasis $B_{\mathscr{L}}$, dann ist $B_{\mathscr{L}}$ ein Herbrandmodell für jede Klausel $S \in \Pi$. Jedes Logikprogramm ist also erfüllbar.

(ii) Die leere Menge ist ein Herbrandmodell für jede Zielklausel, mit Ausnahme der leeren Klausel.

Theorem 5.21 *Sei Π ein Logikprogramm und seien $\mathfrak{M}_i$ $(i \geq 0)$ Herbrandmodelle von Π. Dann ist die Schnittmenge $\mathfrak{M} = \bigcap_i \mathfrak{M}_i$ ebenfalls ein Herbrandmodell von Π.*

Beweis. Angenommen jede Struktur $\mathfrak{M}_i$ ist ein Herbrandmodell von Π, aber die Schnittmenge $\mathfrak{M} = \bigcap_i \mathfrak{M}_i$ ist keines. (Als Schnittmenge von Herbrandmodellen ist $\mathfrak{M}$ aber eine Herbrandstruktur.)

Dann gibt es eine Grundinstanz $S\vartheta$ einer Klausel $S \in \Pi$ für die $\mathfrak{M} \not\models \overline{S\vartheta}$.

Sei $S\vartheta = (A \leftarrow B_1, \ldots, B_n)$ für $n \geq 0$. Dann ist $B_j \in \mathfrak{M}$ für alle $1 \leq j \leq n$ (da $\mathfrak{M}$ eine Herbrandstruktur ist), aber $A \notin \mathfrak{M}$.

Da $\mathfrak{M} = \bigcap_i \mathfrak{M}_i$, ist $B_j \in \mathfrak{M}_i$ für alle $1 \leq j \leq n$ und $i \geq 1$.

Es ist $\mathfrak{M}_i \models \overline{S}\vartheta$ für alle $\mathfrak{M}_i$, da $\Pi \models \overline{S}\vartheta$ und nach Annahme $\mathfrak{M}_i \models \overline{\Pi}$ für alle $\mathfrak{M}_i$. Dann muss jedoch auch $A \in \mathfrak{M}_i$ für alle $i \geq 1$ gelten.

Daraus folgt $A \in \mathfrak{M}$, im Widerspruch zur Annahme. QED

Korollar 5.22 *Die Schnittmenge* aller *Herbrandmodelle eines Logikprogramms* Π *ist ebenfalls ein Herbrandmodell von* Π.

Die Beschränkung auf Logikprogramme, d. h. auf Mengen definiter Hornklauseln, ist hier wesentlich. Für Mengen Γ beliebiger Klauseln gilt im Allgemeinen nicht, dass die Schnittmenge von Herbrandmodellen von Γ ein Herbrandmodell von Γ ist. Zum Beispiel ist sowohl $\mathfrak{M}_1 = \{P(a)\}$ als auch $\mathfrak{M}_2 = \{Q(a)\}$ ein Herbrandmodell der Klauselmenge $\Gamma = \{\vdash P(a), Q(a)\}$, aber $\mathfrak{M}_1 \cap \mathfrak{M}_2 = \emptyset$ ist kein Herbrandmodell von Γ.

Definition 5.23 Sei Π ein Logikprogramm. Die Schnittmenge aller Herbrandmodelle von Π heißt *kleinstes Herbrandmodell* von Π und wird mit $\mathfrak{M}_\Pi$ bezeichnet. (Das kleinste Herbrandmodell wird auch *Standardmodell* genannt.) *kleinstes Herbrandmodell*

Theorem 5.24 *Sei* Π *ein Logikprogramm. Dann ist das kleinste Herbrandmodell* $\mathfrak{M}_\Pi = \{A \in B_\Pi \mid \overline{\Pi} \models A\}$*, wobei* B_Π *die Herbrandbasis von* Π *ist.*

Beweis. Sei $A \in B_\Pi$. Dann gilt

$$\begin{aligned}
\overline{\Pi} \models A &\iff \overline{\Pi} \cup \{\neg A\} \text{ ist unerfüllbar} \quad \text{(siehe Bemerkung nach Definition 2.17)} \\
&\iff \overline{\Pi} \cup \{\neg A\} \text{ hat kein Herbrandmodell} \quad \text{(Theorem 5.17)} \\
&\iff \text{für alle Herbrandmodelle } \mathfrak{M} \text{ von } \Pi \text{ gilt: } \mathfrak{M} \models A \\
&\iff A \in \mathfrak{M}_\Pi.
\end{aligned}$$

QED

5.3 Korrektheit und Vollständigkeit

Nun beweisen wird, dass SLD-Resolution korrekt und vollständig ist. Wir folgen dabei im Wesentlichen der Beweisführung in Stärk (1990, 2000). Andere Beweise finden sich z. B. in Doets (1994) und Apt (1997), wo auch einige wenige subtile Fehler des in Lloyd (1993) gegebenen Vollständigkeitsbeweises behoben sind.
Wir zeigen zunächst Korrektheit, d. h., dass alles, was per SLD-Resolution aus einem Logikprogramm berechnet werden kann, logisch aus dem Programm folgt.

Theorem 5.25 (Korrektheit der SLD-Resolution)
Wenn ϑ *eine berechnete Antwortsubstitution für das Ziel* G *relativ zu einem Logikprogramm* Π *ist, dann gilt* $\overline{\Pi} \models \forall G\vartheta$.

Beweis. Zunächst beweisen wir die folgende
Behauptung. *Sei* $\langle G_0, G_1, \ldots, G_n\rangle, \langle S_1, \ldots, S_n\rangle, \langle \vartheta_1, \ldots, \vartheta_n\rangle$ *eine SLD-Ableitung von* G_0 *aus* Π *mit allgemeinsten Unifikatoren* $\vartheta_1, \ldots, \vartheta_n$*. Dann gilt:*

$$\overline{\Pi} \models \forall G_n \rightarrow \forall G_0\vartheta_1 \ldots \vartheta_n.$$

BEWEIS. Wir zeigen, dass $\overline{\Pi} \models \bigwedge G_{i+1} \to \bigwedge G_i \vartheta_{i+1}$ für alle $i < n$.
Sei $\mathfrak{M}$ eine Struktur und v eine Variablenbelegung, so dass

$$\mathfrak{M} \models \overline{\Pi} \qquad \text{und} \qquad \mathfrak{M} \models_v \bigwedge G_{i+1}.$$

Für die SLD-Ableitung $\langle G_0, G_1, \ldots, G_n\rangle, \langle S_1, \ldots, S_n\rangle, \langle \vartheta_1, \ldots, \vartheta_n\rangle$ von G_0 aus Π mit allgemeinsten Unifikatoren $\vartheta_1, \ldots, \vartheta_n$ gelte:

- G_i ist von der Form $\leftarrow X_i, A_i, Y_i$.
- S_{i+1} ist von der Form $B_{i+1} \leftarrow Z_{i+1}$, und hat keine Variablen gemeinsam mit G_i oder $G_0\vartheta_1 \ldots \vartheta_i$.
- G_{i+1} ist von der Form $(\leftarrow X_i, Z_{i+1}, Y_i)\vartheta_{i+1}$.
- ϑ_{i+1} ist ein mgu von A_i und B_{i+1}.

Dann folgt aus der Annahme $\mathfrak{M} \models_v \bigwedge G_{i+1}$ für die Komponenten des Ziels G_{i+1}:

$$\mathfrak{M} \models_v \bigwedge X_i\vartheta_{i+1}, \quad \mathfrak{M} \models_v \bigwedge Z_{i+1}\vartheta_{i+1} \quad \text{und} \quad \mathfrak{M} \models_v \bigwedge Y_i\vartheta_{i+1}.$$

Da $\overline{\Pi} \models \bigwedge Z_{i+1}\vartheta_{i+1} \to B_{i+1}\vartheta_{i+1}$, gilt aufgrund der Annahme $\mathfrak{M} \models \overline{\Pi}$ und mit $\mathfrak{M} \models_v \bigwedge Z_{i+1}\vartheta_{i+1}$:

$$\mathfrak{M} \models_v B_{i+1}\vartheta_{i+1}.$$

Mit $A_i\vartheta_{i+1} = B_{i+1}\vartheta_{i+1}$ folgt

$$\mathfrak{M} \models_v A_i\vartheta_{i+1}.$$

Somit gilt $\mathfrak{M} \models_v \bigwedge G_i\vartheta_{i+1}$; also mit $\mathfrak{M} \models \overline{\Pi}$ und $\mathfrak{M} \models_v \bigwedge G_{i+1}$ auch

$$\overline{\Pi} \models \bigwedge G_{i+1} \to \bigwedge G_i\vartheta_{i+1}$$

für alle $i < n$. Daraus folgt die Behauptung. QED

Sei nun ϑ eine berechnete Antwortsubstitution für das Ziel G relativ zu einem Logikprogramm Π; d. h., wir nehmen an, dass es eine erfolgreiche SLD-Ableitung $\langle G_0, G_1, \ldots, G_n\rangle$, $\langle S_1, \ldots, S_n\rangle, \langle \vartheta_1, \ldots, \vartheta_n\rangle$ von $G = G_0$ relativ zu Π gibt mit allgemeinsten Unifikatoren $\vartheta_1, \ldots, \vartheta_n$, so dass $G\vartheta = G_0\vartheta_1 \ldots \vartheta_n$.
Aufgrund der eben bewiesenen Behauptung gilt

$$\overline{\Pi} \models \bigwedge G_n \to \bigwedge G_0\vartheta_1 \ldots \vartheta_n.$$

Da die SLD-Ableitung erfolgreich ist, muss G_n die leere Klausel sein; es ist dann (vgl. Definition 5.14 (iii))

$$\bigwedge G_n = \top.$$

Damit gilt

$$\begin{aligned} \overline{\Pi} \models \top \to \bigwedge G_0\vartheta_1 \ldots \vartheta_n &\iff \overline{\Pi} \models \bigwedge G_0\vartheta_1 \ldots \vartheta_n && \text{da } \top \to A \dashv\models A; \\ &\iff \overline{\Pi} \models \bigwedge G\vartheta && \text{da } G\vartheta = G_0\vartheta_1 \ldots \vartheta_n. \quad \text{QED} \end{aligned}$$

Nun beweisen wir Vollständigkeit, d. h., dass alles, was aus einem Logikprogramm logisch folgt, per SLD-Resolution berechnet werden kann. Dazu benötigen wir den Begriff des Implikationsbaums.

Definition 5.26 Ein *Implikationsbaum* für ein Atom A bezüglich eines Logikprogramms Π ist ein endlicher (nach unten verzweigender) Baum $\mathcal{T}$ von Atomen, der induktiv wie folgt definiert ist: *Implikationsbaum*

(i) Die Wurzel von $\mathcal{T}$ ist das Atom A.

(ii) Ist B ein Knoten in $\mathcal{T}$, und ist $B \leftarrow C_1, \ldots, C_n$, für $n \geq 0$, die Instanz einer Klausel in Π, so sind die Nachfolgeknoten von B die Atome $C_1, \ldots, C_n$.

Im Fall $n = 0$ handelt es sich um ein Faktum $B \leftarrow$, und der Knoten B ist ein Blatt.

Die Anzahl der Knoten in $\mathcal{T}$ bezeichnen wir als *Rang* von $\mathcal{T}$. *Rang*

Da es uns im Wesentlichen darauf ankommen wird, dass ein Atom einen *Implikationsbaum von bestimmtem Rang* hat, können wir auch die folgende Definition zugrunde legen.

Definition 5.27 (i) Ist $B \leftarrow$ ein Faktum aus Π und σ eine Substitution, dann hat $B\sigma$ einen *Implikationsbaum vom Rang* 1. *Implikationsbaum mit Rang*

(ii) Ist $B \leftarrow C_1, \ldots, C_k$ eine Programmklausel aus Π, und ist σ eine Substitution, so dass $C_i\sigma$ für $1 \leq i \leq k$ einen Implikationsbaum vom Rang n_i hat, dann hat $B\sigma$ einen *Implikationsbaum vom Rang* $n_1 + \ldots + n_k + 1$.

Beispiel. Sei Π das folgende Logikprogramm:

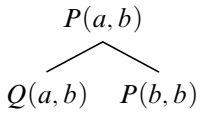
S_1. $P(x, z) \leftarrow Q(x, y), P(y, z)$

S_2. $P(x, x) \leftarrow$

S_3. $Q(a, b) \leftarrow$

(i) Dann ist z. B. $P(d, d)$ ein Implikationsbaum für $P(x, x)[x/d]$ bezüglich Π vom Rang 1.

(ii) Ein Implikationsbaum für $P(x, z)[x/a, y/b, z/b]$ bezüglich Π vom Rang 3 ist:



Das Atom $Q(a, b)$ hat aufgrund von S_3 für die leere Substitution einen Implikationsbaum vom Rang 1. Das Atom $P(b, b)$ hat aufgrund von S_2 für die Substitution $[x/b]$ ebenfalls einen Implikationsbaum vom Rang 1. Aufgrund von S_1 hat $P(a, b)$ also einen Implikationsbaum vom Rang 3 (Substitution: $[x/a, y/b, z/b]$).

Die in einem Implikationsbaum vorkommenden Atome können Variablen enthalten. Wir gehen wie bisher davon aus, dass stets neue Variablen hinzugenommen werden können, wir also nicht auf die Menge der im jeweils betrachteten Logikprogramm vorkommenden Variablen beschränkt sind.

Definition 5.28 Es sei $\mathscr{Z}_\Pi^n$ (für $n \in \mathbb{N}$) die Menge aller Ziele G der Form $\leftarrow A_1, \ldots, A_k$ mit der folgenden Eigenschaft: Es gibt $n_1, \ldots, n_k \in \mathbb{N}$, so dass *Menge $\mathscr{Z}_\Pi^n$ von Zielen*

(i) A_i (für $1 \leq i \leq k$) einen Implikationsbaum bezüglich Π vom Rang n_i hat;

(ii) $n_1 + \ldots + n_k \leq n$.

Das heißt, ein Ziel G ist Element von $\mathscr{Z}_\Pi^n$ genau dann, wenn alle Atome in G Implikationsbäume bezüglich Π haben, so dass die Summe der Ränge aller dieser Implikationsbäume $\leq n$ ist.

Folgerung 5.29 *Wenn ein Ziel $\leftarrow X, A, Y$ Element von $\mathscr{Z}_\Pi^{n+1}$ ist, dann gibt es eine Klausel $B \leftarrow Z$ in Π und eine Substitution σ, so dass $A = B\sigma$, und das Ziel $\leftarrow X, Z\sigma, Y$ Element von $\mathscr{Z}_\Pi^n$ ist.*

Lemma 5.30 *Wenn $\overline{\Pi} \vDash A$ für ein Atom A, dann hat A einen Implikationsbaum bezüglich Π.*

Beweis. Wir gehen wie folgt vor: (1) Wir definieren eine Termstruktur $\mathfrak{M} = \langle M, \mathcal{I} \rangle$, und zeigen, (2) dass $\mathfrak{M}$ ein Modell von $\overline{\Pi}$ ist. Eine *Termstruktur* unterscheidet sich von einer Herbrandstruktur dadurch, dass M nicht auf das Herbranduniversum $U_{\mathscr{L}}$ beschränkt ist, sondern alle Terme einschließlich Variablen beinhaltet. (3) Dann zeigen wir, dass $A\sigma$ genau dann einen Implikationsbaum bezüglich Π hat, wenn A in $\mathfrak{M}$ unter einer bestimmten Variablenbelegung gültig ist. (4) Mit der Annahme $\overline{\Pi} \vDash A$ folgt dann, dass A einen Implikationsbaum bezüglich Π hat.

(1) Die Termstruktur $\mathfrak{M} = \langle M, \mathcal{I} \rangle$ definieren wir wie folgt:

(i) Es sei M die Menge aller Terme (einschließlich Variablen).

(ii) Für n-stellige Funktionszeichen f und Terme $t_1, \ldots, t_n \in M$ sei

$$\mathcal{I}(f)(t_1, \ldots, t_n) \coloneqq f(t_1, \ldots, t_n).$$

(iii) Für n-stellige Relationszeichen R und Terme $t_1, \ldots, t_n \in M$ sei

$$\langle t_1, \ldots, t_n \rangle \in \mathcal{I}(R) \;:\Longleftrightarrow\; R(t_1, \ldots, t_n) \text{ hat einen Implikationsbaum bezüglich } \Pi.$$

(2) Sei v eine Variablenbelegung in $\mathfrak{M}$ und σ die Substitution $[x_1/v(x_1), \ldots, x_n/v(x_n)]$. Als Verallgemeinerung von Lemma 5.11 (Beweis analog) gilt: Ist t ein Term mit $\mathrm{FV}(t) \subseteq \{x_1, \ldots, x_n\}$, dann gilt $[\![t]\!]_v^{\mathfrak{M}} = t\sigma$.
Für Atome A mit $\mathrm{FV}(A) \subseteq \{x_1, \ldots, x_n\}$ gilt damit:

$$\mathfrak{M} \vDash_v A \iff A\sigma \text{ hat einen Implikationsbaum bezüglich } \Pi. \qquad (*)$$

Sei nun $S = (B \leftarrow C_1, \ldots, C_k)$ eine Klausel in Π mit $\mathrm{FV}(S) \subseteq \{x_1, \ldots, x_n\}$. Wir zeigen $\mathfrak{M} \vDash_v (C_1 \wedge \ldots \wedge C_k) \to B$.
Angenommen, $\mathfrak{M} \vDash_v C_1 \wedge \ldots \wedge C_k$. Dann folgt mit $(*)$, dass jedes $C_i\sigma$ (für $1 \leq i \leq k$) einen Implikationsbaum hat. Damit hat auch $B\sigma$ einen Implikationsbaum, und es gilt $\mathfrak{M} \vDash_v B$.
Da die Variablenbelegung v beliebig war, gilt $\mathfrak{M} \vDash \overline{S}$; und da $S \in \Pi$ beliebig war, gilt $\mathfrak{M} \vDash \overline{\Pi}$.

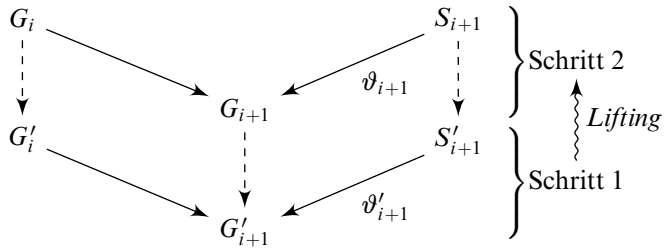
(3) Sei nun $\overline{\Pi} \vDash A$. Dann gilt insbesondere für die identische Variablenbelegung v_ε mit $v_\varepsilon(x) = x$ für alle Variablen x: $\mathfrak{M} \vDash_{v_\varepsilon} A$. Die entsprechende Substitution σ ist dann die leere Substitution ε, also $A\sigma = A\varepsilon = A$.

(4) Aus $(*)$ folgt, dass A einen Implikationsbaum bezüglich Π hat. QED

Es gilt auch die umgekehrte Richtung: *Falls das Atom A einen Implikationsbaum bezüglich Π hat, dann gilt $\overline{\Pi} \vDash A$.* Wir benötigen jedoch nur die eben bewiesene Richtung.

Das folgende *Liftinglemma* garantiert, dass stets allgemeinste Ziele abgeleitet werden können, sofern überhaupt ein Ziel abgeleitet werden kann. Es sagt im Wesentlichen aus, dass wir die Ableitung eines Ziels G'_{i+1} aus einem Ziel G'_i mit Inputklausel S'_{i+1} und mgu ϑ'_{i+1} zu einer Ableitung liften können, bei der eine Verallgemeinerung G_{i+1} des Ziels G'_{i+1} aus Verallgemeinerungen G_i und S_{i+1} von G'_i bzw. S'_{i+1} mit einem mgu ϑ_{i+1} abgeleitet werden kann. Betrachtet man zwei SLD-Resolutionsschritte

Liftinglemma



so bedeutet dies, dass wir Schritt 1 zu Schritt 2 liften können, dessen Klauseln allgemeiner (gestrichelte Pfeile) als die von Schritt 1 sind.

Lemma 5.31 (Lifting)

(i) *Sei G das Ziel $\leftarrow A, X$, und sei S die Klausel $B \leftarrow Z$.*

(ii) *Seien σ und τ zwei Substitutionen, so dass $A\sigma = B\tau$.*

(iii) *Sei $B' \leftarrow Z'$ eine Variante von S, die mit G keine Variablen gemeinsam hat.*

Dann sind A und B' unifizierbar mit allgemeinstem Unifikator ϑ, und es gibt eine Substitution σ', so dass $(\leftarrow X, Z')\vartheta\sigma' = (\leftarrow X\sigma, Z\tau)$.

(Mit anderen Worten: Wenn $G = (\leftarrow A, X)$, $S = (B \leftarrow Z)$ und $A\sigma = B\tau$, dann ist $\leftarrow X\sigma, Z\tau$ eine Instanz jeder Resolvente von G mit einer Variante von S, welche keine Variablen mit G gemeinsam hat.)

Beweis. Da $B' \leftarrow Z'$ eine Variante von S ist, gibt es eine Substitution η, so dass $(B' \leftarrow Z')\eta = (B \leftarrow Z)$.

Sei die Substitution $\rho = \sigma \mid \mathrm{FV}(G) \cup (\eta\tau) \mid \mathrm{FV}(B' \leftarrow Z')$. Dann ist

$$A\rho = A\sigma = B\tau = (B'\eta)\tau = B'\rho.$$

Die Substitution ρ ist also ein Unifikator von A und B'.

Sei nun ϑ ein allgemeinster Unifikator von A und B'. Dann gibt es eine Substitution σ', so dass $\vartheta\sigma' = \rho$. Somit ist

$$X\vartheta\sigma' = X\rho = X\sigma \qquad \text{und} \qquad Z'\vartheta\sigma' = Z'\rho = Z'\eta\tau = Z\tau.$$

Also ist $(\leftarrow X, Z')\vartheta\sigma' = (\leftarrow X\sigma, Z\tau)$. QED

Lemma 5.32 (Konstruktion einer erfolgreichen SLD-Ableitung)
Sei $\mathcal{R}$ eine Auswahlfunktion. Wenn $G\sigma \in \mathscr{Z}^n_\Pi$, dann gibt es eine erfolgreiche SLD-Ableitung der Länge n für G aus Π gemäß $\mathcal{R}$ mit einer berechneten Antwortsubstitution ϑ, so dass $G\vartheta$ allgemeiner ist als $G\sigma$, es also eine Substitution τ gibt mit $G\vartheta\tau = G\sigma$.

Beweis. Sei $G_0\sigma \in \mathscr{Z}^n_\Pi$. Per Induktion nach $i \leq n$ konstruieren wir jetzt eine SLD-Ableitung $\langle G_0, G_1, \ldots, G_i\rangle, \langle S_1, \ldots, S_i\rangle, \langle \vartheta_1, \ldots, \vartheta_i\rangle$ gemäß $\mathcal{R}$ und eine Folge von Substitutionen $\sigma_0, \sigma_1, \ldots, \sigma_i$, so dass gilt:

(i) $G_0\vartheta_1 \ldots \vartheta_i\sigma_i = G_0\sigma$;

(ii) $G_i\sigma_i \in \mathscr{Z}_\Pi^{n-i}$.

Für $i = 0$ setzen wir $\sigma_0 := \sigma$.

Sei nun $0 < i < n$. Wir verwenden als Induktionsannahme, dass eine SLD-Ableitung und eine Folge von Substitutionen mit den Eigenschaften (i) und (ii) schon bis i konstruiert wurden.

Ist G_i die leere Klausel, so ist die SLD-Ableitung erfolgreich, und wir sind fertig.

Andernfalls hat G_i die Form $\leftarrow X, A, Y$. Wir nehmen an, dass $\mathcal{R}$ (basierend auf der bereits konstruierten SLD-Ableitung) im Ziel G_i das Atom A auswählt.

Da $G_i\sigma_i \in \mathscr{Z}_\Pi^{n-i}$, gibt es nach Folgerung 5.29 eine Klausel $B \leftarrow Z$ in Π und eine Substitution τ, so dass $A\sigma_i = B\tau$ und $(\leftarrow X\sigma_i, Z\tau, Y\sigma_i) \in \mathscr{Z}_\Pi^{n-(i+1)}$.

Sei $B' \leftarrow Z'$ eine Variante von $B \leftarrow Z$, die mit G_i oder $G_0\vartheta_1 \ldots \vartheta_i$ keine Variablen gemeinsam hat.

Aufgrund des Liftinglemmas (Lemma 5.31) sind A und B' unifizierbar.

Sei ϑ_{i+1} ein mgu von A und B', und $G_{i+1} := (\leftarrow X, Z', Y)\vartheta_{i+1}$. Dann erhalten wir mit dem Liftinglemma eine Substitution σ_{i+1}, so dass gilt:

(i) $G_0\vartheta_1 \ldots \vartheta_i\vartheta_{i+1}\sigma_{i+1} = G_0\vartheta_1 \ldots \vartheta_i\sigma_i = G_0\sigma$;

(ii) $G_{i+1}\sigma_{i+1} = (\leftarrow X, Z', Y)\vartheta_{i+1}\sigma_{i+1} = (\leftarrow X\sigma_i, Z\tau, Y\sigma_i) \in \mathscr{Z}_\Pi^{n-(i+1)}$.

Damit haben wir die Ableitung um einen SLD-Resolutionsschritt verlängert.

Für $i = n$ gilt dann schließlich $G_n\sigma_n \in \mathscr{Z}_\Pi^0$. Dann kann G_n aber nur die leere Klausel sein, da jede andere Zielklausel Rang > 0 hat. Wir haben somit eine *erfolgreiche* SLD-Ableitung konstruiert. QED

Theorem 5.33 (Vollständigkeit der SLD-Resolution)
Sei $\mathcal{R}$ eine Auswahlfunktion. Wenn $\overline{\Pi} \models \mathbb{A}G\sigma$, dann gibt es eine erfolgreiche SLD-Ableitung von G aus Π gemäß $\mathcal{R}$ mit einer berechneten Antwortsubstitution ϑ, so dass $G\vartheta$ allgemeiner ist als $G\sigma$, es also eine Substitution τ gibt mit $G\vartheta\tau = G\sigma$.

Beweis. Angenommen $\overline{\Pi} \models \mathbb{A}G\sigma$.

Mit Lemma 5.30 folgt, dass $A\sigma$ für jedes Atom A in G einen Implikationsbaum bezüglich Π hat.

Daher gibt es ein $n \in \mathbb{N}$, so dass $G\sigma \in \mathscr{Z}_\Pi^n$.

Damit ist die Annahme $G\sigma \in \mathscr{Z}_\Pi^n$ in Lemma 5.32 erfüllt, und wir erhalten eine erfolgreiche SLD-Ableitung mit den gewünschten Eigenschaften. QED

Korollar 5.34 *Das Standardmodell eines Logikprogramms Π mit Herbrandbasis $B_\mathscr{L}$ kann damit durch die Menge*

$$\{A \in B_\mathscr{L} \mid \textit{es gibt eine erfolgreiche SLD-Ableitung für } A \textit{ aus } \Pi\}$$

angegeben werden.

Da die Auswahlfunktion für die Auswahl des Atoms in der jeweils aktuellen Zielklausel im Vollständigkeitssatz beliebig ist, bleibt Vollständigkeit für jede bestimmte verwendete Auswahlfunktion bestehen.

Hingegen kann die Auswahlstrategie für die Wahl einer Programmklausel für das ausgewählte Atom zum Verlust der Vollständigkeit in dem Sinne führen, dass es zwar eine erfolgreiche SLD-Ableitung gibt, diese aber nicht gefunden wird.
Zum Beispiel wählt Prolog die Programmklauseln in derselben Reihenfolge wie sie im Programm stehen. Für das aus den beiden Klauseln

S_1. $P(a) \leftarrow P(a)$
S_2. $P(a) \leftarrow$

bestehende Programm Π gilt dann: $P(a)$ folgt logisch aus $\overline{\Pi}$, d. h. $\overline{\Pi} \models P(a)$, aber Prolog findet keine erfolgreiche SLD-Ableitung für das Ziel $\leftarrow P(a)$. Prolog wendet nur die Klausel S_1, aber nie die Klausel S_2 an. Doch nur mit S_2 kann die leere Klausel abgeleitet werden.
Dies ist kein Widerspruch zum Vollständigkeitssatz, da dieser ja nur eine Aussage über die *Existenz* einer erfolgreichen SLD-Ableitung macht, jedoch nichts darüber sagt, ob diese auch in jedem Fall *gefunden* wird.

Literatur

K. R. Apt (1997), *From Logic Programming to Prolog*. London: Prentice Hall.

P. Blackburn, J. Bos & K. Striegnitz (2006), *Learn Prolog Now!*. London: College Publications, sowie online frei zugänglich unter http://www.learnprolognow.org/.

W. F. Clocksin & C. S. Mellish (2003), *Programming in Prolog*, 5th edition. Berlin: Springer-Verlag.

K. Doets (1994), *From Logic to Logic Programming*. Cambridge, Massachusetts: The MIT Press.

H. J. Goltz & H. Herre (1990), *Grundlagen der logischen Programmierung*. Weinheim: Wiley-VCH.

R. Kahle (2001), *Einführung in die Logikprogrammierung*, Vorlesungsskript, Universität Tübingen.

J-L. Lassez, M. J. Maher & K. Marriot (1987), Unification Revisited. In: J. Minker (Hrsg.), *Foundations of Deductive Databases and Logic Programming*, Los Altos: Morgan Kaufmann. S. 587–625.

A. Leitsch (1997), *The Resolution Calculus*. Berlin: Springer-Verlag.

J. W. Lloyd (1993), *Foundations of Logic Programming*, 2nd edition. Berlin: Springer-Verlag.

S.-H. Nienhuys-Cheng & R. de Wolf (1997), *Foundations of Inductive Logic Programming*. Lecture Notes in Artificial Intelligence 1228, Berlin: Springer.

U. Schöning (2000), *Logik für Informatiker*, 5. Auflage. Heidelberg: Spektrum Akademischer Verlag.

R. F. Stärk (1990), A direct proof for the completeness of SLD-resolution. In: E. Börger, H. Kleine Büning & M. M. Richter (Hrsg.), *CSL '89, 3rd Workshop on Computer Science Logic, Kaiserslautern, Germany, October 2–6, 1989*. Lecture Notes in Computer Science 440, Berlin: Springer. S. 382–383.

R. F. Stärk (2000), *Logikprogrammierung*, Vorlesungsskript, ETH Zürich.

A. S. Troelstra & H. Schwichtenberg (2000), *Basic Proof Theory*, 2nd edition. Cambridge University Press.

D. van Dalen (2013), *Logic and Structure*, 5th edition. Springer.

Sachverzeichnis