

Formale Systeme

P. H. Schmitt

Winter 2007/2008
Version: 5. März 2008

Vorwort

Formale Methoden und die zu ihrer Umsetzung notwendigen formalen Systeme spielen in der Informatik von Anfang an eine wichtige Rolle. Umfang und Komplexität der einzelnen Computeranwendungen sind in den letzten Jahren enorm angewachsen, was die Kontrolle ihrer Korrektheit immer schwieriger macht. Neben den traditionellen Methoden zur Qualitätssicherung einschließlich extensiver Testläufe wird der Wunsch immer stärker, eine formale Verifikation der logischen Korrektheit durchführen zu können. Die Fortschritte in der Entwicklung formaler Beweismethoden und die Verfügbarkeit immer schnellerer Rechner hat diesen Wunsch der Verwirklichung ein Stück näher gebracht. Untrennbar verbunden mit dem operationalen Verhalten ist die Spezifikation und Analyse komplexer Systeme mittels deklarativer formaler Sprachen.

Die Fülle der in der Informatik benutzten formalen Systeme ist schier unübersehbar. Wir machen keinen Versuch einen Überblick oder auch nur den Ansatz zu einer umfassenden Definition zu geben. Statt dessen betrachten wir ausführlich einen Kernbereich für das Arbeiten mit formalen Systemen: Logik, mit Betonung ihrer Funktion in der Informatik.

Wir befassen uns in Kapitel 1 zunächst mit der *Aussagenlogik*, insbesondere, weil die Grundidee für Vollständigkeitsbeweise hier in einer besonders einfachen Form bereits realisiert ist. Einen wesentlich stärker algorithmisch geprägten Bereich betritt man im speziellen Fall der *Gleichungslogik*: nämlich das Gebiet der *Termerzeugungssysteme*, das die zentrale Methodik für symbolisches Rechnen und für abstrakte Datentypen liefert.

Das Ziel dieser Vorlesung ist dabei die exemplarische Vermittlung zentraler Ideen. Das geschieht dann gründlich, wobei nicht nur die gültigen Aussagen, wie Korrektheits- und Vollständigkeitssätze, genannt sondern auch vollständige Beweise gegeben werden. Im Vordergrund stehen dabei die Fragestellungen:

1. wie bekommen Zeichen eine Bedeutung ?

2. was ist ein Beweis ?

Die erste Frage findet ihre Antwort in der Beschreibung des Zusammenhangs zwischen Syntax und Semantik der Prädikatenlogik, die zweite in der Angabe formaler Beweiskalküle, die der Strenge der Begriffsbildungen der mathematischen Logik standhalten. Der Autor ist der Auffassung, daß ein Student der Informatik, wenigstens einmal während seines Studiums, auch die technischen Details in einem solchen Vorgehen kennenlernen sollte. Auch das geschieht wieder exemplarisch an einigen Stellen. Danach kehrt die Darstellung wieder auf eine etwas informellere Ebene zurück. Damit nicht durch zufällige Eigenheiten eines Beispiels ein falscher Eindruck entsteht, werden mehrere Logikkalküle vorgestellt.

Eine umfangreiche Sammlung von Übungsaufgaben, teilweise mit Lösungen, bietet dem Leser die Gelegenheit, neben der passiven Rezeption des Materials sich auch aktiv mit ihm auseinanderzusetzen.

Anwendungsmöglichkeiten werden an nicht trivialen Beispielen aufgezeigt. Realistische Anwendungsszenarien erwiesen sich dagegen als zu umfangreich. Notwendigerweise gibt es Themen, die in diesem Skript nicht behandelt werden konnten. So werden Logiken höherer Stufe und nichtklassische Logiken nur sehr knapp behandelt, und der Lambda-Kalkül kommt überhaupt nicht vor.

Danksagungen

Der vorliegende Text ist eine Fortentwicklung des gleichnamigen Vorlesungsskripts, das meinem Kollegen Wolfram Menzel und mir in den Jahren 1988 bis 2001 als gemeinsame Grundlage für unsere Vorlesungen diente. Ich bin Wolfram Menzel dankbar für die offene und fruchtbare Kooperation und insbesondere für die Vorleistungen, von denen er mir erlaubt hat zu profitieren.

Ein herzliches Dankeschön ergeht an alle Studentinnen und Studenten, es sind zu viele um sie hier namentlich aufzuführen, die durch ihre Kritik und Anregungen zur stetigen Verbesserung des Manuskripts beigetragen haben.

Referenzen

Verweise auf relevante oder benutzte Literatur werden an den entsprechenden Stellen im Skriptum gegeben werden. Wir listen hier jedoch einige Bücher auf, die in einem allgemeineren Sinne zur Vorbereitung der Vorlesung *Formale Systeme* herangezogen wurden.

- M. Fitting*: First Order Logic and Automated Theorem Proving, siehe [Fit90]
- U. Schöning*: Logik für Informatiker, siehe [Sch00]
- V. Sperschneider/G. Antoniou*: Logic: a Foundation for Computer Science, [SA91]
- P. H. Schmitt*: Theorie der logischen Programmierung, siehe [Sch92]
- Huth und Ryan*: Logic in Computer Science, siehe [HR00].
- Ehrig, Mahr et al*: Mathematische Grundlagen der Informatik, siehe [EMC⁺99]
- E. Börger*: Logik, Berechenbarkeit, Komplexität, siehe [B98]
- Ebbinghaus/Flum/Thomas*: Mathematische Logik, siehe [EFT92]
- R. Smullyan*: First Order Logic, siehe [Smu95]

Inhaltsverzeichnis

Vorwort	iv
1 Voraussetzungen	1
2 Aussagenlogik: Syntax und Semantik	6
2.1 Einleitendes Beispiel	7
2.1.1 Sudoku	7
2.1.2 Das Acht-Damen-Problem	10
2.2 Syntax der Aussagenlogik	12
2.2.1 Strukturelle Induktion	13
2.2.2 Einfache Fakten zur Syntax, Abkürzungen	14
2.2.3 Übungsaufgaben	14
2.3 Semantik der Aussagenlogik	16
2.3.1 Wahrheitstabellen, Boole'sche Funktionen, Basen	17
2.3.2 Allgemeingültigkeit und Erfüllbarkeit	18
2.3.3 Einige allgemeingültige Formeln	19
2.3.4 Semantische Folgerbarkeit	21
2.3.5 Bemerkung zur Notation	24
2.3.6 Übungsaufgaben	24
2.4 Normalformen	26
2.4.1 Disjunktive und konjunktive Normalform	26
2.4.2 Primimplikanten	27
2.4.3 Kurze Konjunktive Normalform	28

2.4.4	Shannons Normalform	33
2.4.5	Übungsaufgaben	44
2.5	Erfüllbarkeit in speziellen Formelklassen	48
2.5.1	Horn-Formeln	48
2.5.2	Äquivalenzformeln	51
2.5.3	Übungsaufgaben	54
3	Aussagenlogik: Beweistheorie	55
3.1	Einleitende Bemerkungen	55
3.1.1	Abstrakte Kalküle	56
3.1.2	Abstrakte Ableitbarkeit	57
3.1.3	Übungsaufgaben	59
3.2	Der aussagenlogische Hilbertkalkül	60
3.2.1	Der Kalkül	60
3.2.2	Metatheoreme	66
3.3	Aussagenlogische Resolution	68
3.3.1	Syntax und Semantik	68
3.3.2	Kalkül	69
3.3.3	Korrektheit und Vollständigkeit	70
3.3.4	Übungsaufgaben	74
3.4	Aussagenlogische Tableaux	77
3.4.1	Syntax und Semantik	77
3.4.2	Kalkül	78
3.4.3	Generierung von Tableauregeln	85
3.4.4	Übungsaufgaben	86
3.5	Sequenzkalküle	89
3.5.1	Syntax und Semantik	89
3.5.2	Kalkül	90
3.5.3	Übungsaufgaben	94
3.6	Sonstige Kalküle	95
3.7	Anwendungen der Aussagenlogik	100

3.7.1	Beschreibung von Schaltkreisen	100
3.7.2	Wissensrepräsentation	101
3.7.3	Übungsaufgaben	103
4	Prädikatenlogik erster Ordnung: Syntax und Semantik	104
4.1	Einführende Beispiele	105
4.1.1	Alltagslogik	105
4.1.2	Spezifikationen im Java Card API	106
4.2	Syntax der Prädikatenlogik	108
4.2.1	Terme und Formeln	108
4.2.2	Gebundene und freie Variable. Substitutionen	111
4.2.3	Unifikation	115
4.2.4	Unifikationsalgorithmus	118
4.2.5	Übungsaufgaben	121
4.3	Semantik der PL1	124
4.3.1	Interpretationen	124
4.3.2	Allgemeingültigkeit, logische Äquivalenz	133
4.3.3	Übungsaufgaben	138
4.4	Normalformen	144
4.4.1	Skolem-Normalform	145
4.4.2	Der Satz von Herbrand	150
4.4.3	Übungsaufgaben	153
5	Prädikatenlogik erster Ordnung: Beweistheorie	156
5.1	Tableaukalkül	157
5.1.1	Tableauregeln	157
5.1.2	Korrektheit	164
5.1.3	Hintikka-Mengen	168
5.1.4	Vollständigkeit	171
5.1.5	Übungsaufgaben	174
5.2	Sonstige Kalküle	177

5.2.1	Hilbertkalkül	177
5.2.2	Resolutionskalkül	178
5.2.3	Ein Sequenzenkalkül	187
5.2.4	Übungsaufgaben	193
5.3	Weitere Anmerkungen zur Prädikatenlogik erster Ordnung . .	194
5.3.1	Andere Notationsformen	194
5.3.2	Metaresultate	195
5.3.3	Sorten	196
5.3.4	Übungsaufgaben	199
5.4	Anwendung	200
5.4.1	Programmverifikation und -Synthese	200
5.4.2	Verifikation eines Schaltkreises	200
5.4.3	Anwendung in der Mathematik	201
5.4.4	Semantische Technologien	205
5.4.5	Übungsaufgaben	212
5.5	Reduktionssysteme	215
5.6	Termersetzungssysteme	221
6	Die Objektbeschreibungssprache OCL	223
6.1	Historie und Motivation	224
6.2	Einführung in OCL	225
6.3	Übersetzung in Prädikatenlogik	232
6.4	Übungsaufgaben	235
7	Modale Aussagenlogik	240
7.1	Einführung	241
7.2	Syntax und Semantik	246
7.3	Eigenschaften von Kripke-Rahmen	251
7.4	Entscheidbarkeit modaler Logiken	255
7.5	Übungsaufgaben	257

8	Temporale Logik	260
8.1	Lineare Temporale Logik (LTL)	262
8.1.1	Übungsaufgaben	266
9	Stärkere Logiken	270
9.1	Prädikatenlogik zweiter Ordnung	271
9.1.1	Übungsaufgaben	275
9.2	Dynamische Logik	275
10	Automaten	283
10.1	Endliche Automaten	284
10.1.1	Deterministische endliche Automaten	284
10.1.2	Nichtdeterministische endliche Automaten	286
10.1.3	Reguläre Ausdrücke	291
10.1.4	Übungsaufgaben	295
10.2	Omega Automaten	296
10.2.1	Büchi Automaten	297
10.2.2	Abschlußeigenschaften	300
10.2.3	Varianten von Büchi Automaten	305
10.2.4	Übungsaufgaben	307
11	Modellprüfung	312
11.1	Einführung	313
11.2	Büchi Automaten und LTL	317
11.2.1	Übungsaufgaben	323
11.3	Modellprüfung für LTL	325
11.3.1	Problemstellung	325
11.3.2	Methode	328
11.3.3	Übungsaufgaben	329
11.4	Aussagenlogische Modellprüfung (Zusatzstoff)	330
11.4.1	Übungsaufgaben	338
11.5	Modellprüfung mit SPIN	339

12 Lösungen	345
Literatur	345
Index	351

Kapitel 1

Voraussetzungen

Der Stoff des Vorstudiums im Bereich der Theoretischen Informatik sollte bekannt sein. Insbesondere wird unterstellt:

1. eine gewisse elementare Vertrautheit mit Aussagenlogik (Schaltalgebra)
2. gute Vertrautheit mit den Grundbegriffen der Theorie der Berechenbarkeit (aus „Informatik III“).
3. insbesondere setzen wir voraus, daß dem Leser die Unterscheidung von Objektsprache und Metasprache geläufig ist. Das sollte uns erlauben bei der Unterscheidung der beiden Sprachebenen nicht gar so engherzig zu sein.

Wir stellen einige der im folgenden häufig benutzten Definitionen noch einmal zusammen.

Definition 1.1 (Eigenschaften von Relationen)

Sei R eine zweistellige Relation auf einer Menge D . R ist:

reflexiv Für alle $d \in D$ gilt $R(d, d)$

irreflexiv Für alle $d \in D$ gilt nicht $R(d, d)$

transitiv Für $d_1, d_2, d_3 \in D$ mit $R(d_1, d_2)$ und $R(d_2, d_3)$, gilt auch $R(d_1, d_3)$

symmetrisch Für alle $a, b \in D$ mit $R(a, b)$ gilt auch $R(b, a)$.

antisymmetrisch Für alle $d_1, d_2 \in D$ $R(d_1, d_2)$ und $R(d_2, d_1)$ gilt $d_1 = d_2$.

asymmetrisch Für alle $d_1, d_2 \in D$ kann nicht $R(d_1, d_2)$ und $R(d_2, d_1)$ gleichzeitig gelten.

funktional Aus $R(a, b_1)$ und $R(a, b_2)$ folgt $b_1 = b_2$.

Insbesondere ist die *leere* Relation R_\emptyset unktional.

Ordnung R ist reflexiv, transitiv und antisymmetrisch.

Ordnungsrelationen werden häufig mit $a \preceq b$ notiert anstelle von $R(a, b)$.

strikte Ordnung R ist transitiv, asymmetrisch und irreflexiv.

Strikte Ordnungsrelationen werden häufig mit $a \prec b$ notiert anstelle von $R(a, b)$.

Quasiordnung R ist reflexiv und transitiv.

totale Ordnung R ist eine Ordnungsrelation und zusätzlich gilt für alle $d_1, d_2 \in D$ mit $d_1 \neq d_2$ entweder $R(d_1, d_2)$ oder $R(d_2, d_1)$.

Totale Ordnungen werden auch **lineare Ordnungen** genannt.

Äquivalenzrelation R ist reflexiv, transitiv und symmetrisch.

Wenn wir Ordnung oder strikte Ordnung sagen, meinen wir eine nicht notwendig lineare Ordnung. Wenn wir diese Tatsache betonen wollen, fügen wir das Attribut **partiell** hinzu, reden also von einer partiellen Ordnung oder einer strikten partiellen Ordnung.

Lemma 1.2

1. Sei (D, \preceq) eine Ordnung. Wir definieren für $a, b \in D$:

$$a \prec b \text{ gdw } a \preceq b \text{ und } a \neq b$$

Dann ist (D, \prec) eine strikte Ordnung.

2. Sei (D, \prec) eine strikte Ordnung. Wir definieren für $a, b \in D$:

$$a \preceq b \text{ gdw } a \prec b \text{ oder } a = b$$

Dann ist (D, \preceq) eine Ordnung.

Beweis: Einfach. ■

Beispiel 1.3

Die folgenden Strukturen sind Beispiel für strikte, totale Ordnungen:

1. $(\mathbb{N}, <)$, die natürlichen Zahlen mit der üblichen strikten Ordnung
Es gibt ein kleinstes Element, jedes Element hat einen nächsten Nachfolger und es gibt kein größtes Element.

2. $(\{0, \dots, n\}, <)$, die natürlichen Zahlen von 0 bis einschließlich n mit der üblichen strikten Ordnung
Es gibt ein kleinstes Element, es gibt ein größtes Element und jedes Element außer dem größten hat einen nächsten Nachfolger.
3. $(\mathbb{Z}, <)$, die ganzen Zahlen mit der üblichen strikten Ordnung
Es gibt kein kleinstes Element, es gibt kein größtes Element und jedes Element hat einen nächsten Nachfolger und Vorgänger.
4. $(\mathbb{Q}, <)$, die rationalen Zahlen mit der üblichen strikten Ordnung
Es gibt kein kleinstes Element, es gibt kein größtes Element und zwischen je zwei Elementen liegt ein drittes Element.
5. $(\mathbb{N} \times \mathbb{N}, <_{lex})$, wobei
 $(a_1, b_1) <_{lex} (a_2, b_2)$ gdw $a_1 < a_2$ oder $a_1 = a_2$ und $b_1 < b_2$.
Es gibt ein kleinstes Element, es gibt kein größtes Element und jedes Element hat einen nächsten Nachfolger und Vorgänger. Im Unterschied zu 3 gibt es jedoch ein Element x (in der Tat sehr viele), so daß unendliche viele Elemente kleiner als x sind.

Beispiel 1.4

Die folgende Struktur ist ein Beispiel für eine strikte, partielle Ordnungen, die nicht total ist.

$$(\mathbb{N} \times \mathbb{N}, <_{comp})$$

wobei $(a_1, b_1) <_{comp} (a_2, b_2)$ gdw $a_1 < a_2$ und $b_1 < b_2$.

Definition 1.5 (Transitive Hülle)

Sei R eine beliebige binäre Relation mit Definitionsbereich D .

Die **transitive Hülle** von R ist die kleinste Relation R^+ auf D mit den Eigenschaften:

1. $R \subseteq R^+$,
2. R^+ ist eine transitive Relation.

Entsprechend stehen R^* und R^s für die kleinste, R umfassende transitive, reflexive und für die transitive, reflexive und symmetrische Relation.

Lemma 1.6

Zu jeder Relation R gibt stets eine

1. transitive Hülle
2. transitive, reflexive Hülle
3. transitive, reflexive, symmetrische Hülle

Beweis Wir betrachten die erste Behauptung.

Sei R eine binäre Relation mit Definitionsbereich D .

Sei $\mathcal{R} = \{R' \mid R \subseteq R' \subseteq D \times D \text{ und } R' \text{ ist transitiv}\}$. Die Menge \mathcal{R} von Relationen ist nicht leer, da auf jeden Fall $D \times D$ in ihr liegt. Man sieht leicht, daß $R^+ = \bigcap \mathcal{R}$ gilt. Der Nachweis für die beiden restlichen Behauptungen erfolgt mit den offensichtlichen Variationen. ■

Lemma 1.7

Sei R eine Relation mit Definitionsbereich D .

Die Relation S sei definiert durch

$$aSb \Leftrightarrow \begin{array}{l} \text{es gibt } n \in \mathbb{N} \text{ und } d_1, \dots, d_n \in D \text{ mit} \\ d_1 = a \text{ und } d_n = b \text{ und} \\ d_i R d_{i+1} \text{ gilt für alle } 1 \leq i < n \end{array}$$

Dann ist S die transitive Hülle von R .

Beweis Einfach. ■

Lemma 1.8

Jede Ordnungsrelation (D, \leq) kann zu einer totalen Ordnung (D, \leq_t) erweitert werden.

Beweis Wenn (D, \leq) noch keine totale Ordnung ist, dann gibt es $a, b \in D$ mit $a \not\leq b$ und $b \not\leq a$. Wir definieren

$$x \leq_1 y \Leftrightarrow (x \leq y \text{ oder } (x \leq a \text{ und } b \leq y))$$

Man zeigt, daß (D, \leq_1) wieder eine Ordnungsrelation ist. Diese Operation führt man durch bis eine totale Ordnung erreicht ist. Ist D eine endliche Menge, dann ist das klar. Für unendliches D benötigt man transfinite Induktion. Man setzt $(D, \leq^0) = (D, \leq)$. Falls (D, \leq^α) noch nicht total ist setzt man $(D, \leq^{\alpha+1}) = (D, (\leq^\alpha)q_1)$ und für Limesordinalzahlen λ setzt man $(D, \leq^\lambda) = \bigcup_{\alpha < \lambda} (D, \leq^\alpha)$. ■

Es gibt eine Standardmethode um von einer Quasiordnung zu einer Ordnung zu kommen.

Lemma 1.9

Sei eine Quasiordnung (D, \preceq) gegeben.

1. Dann wird durch

$$a \sim b \text{ gdw } a \preceq b \text{ und } b \preceq a$$

eine Äquivalenzrelation auf D definiert.

2. Bezeichne $[a]_{\sim} = \{b \in D : a \sim b\}$ die Äquivalenzklasse eines Elements a . Setze $D_{\sim} = \{[a]_{\sim} : a \in D\}$ und $[a]_{\sim} \preceq [b]_{\sim}$ gdw $a \preceq b$, dann ist (D_{\sim}, \preceq) wohldefiniert und eine Ordnungsrelation.

Beweis: Einfach. ■

Definition 1.10 (Kongruenzrelation)

Sei R eine zweistellige Relation und zusätzlich eine Menge Σ von Funktionssymbolen gegeben. R heißt eine **Kongruenzrelation**, wenn

1. R eine Äquivalenzrelation ist und
2. für jedes Funktionszeichen $f \in \Sigma$ und jede Wahl von Elementen $a_1, \dots, a_n, b_1, \dots, b_n$ aus dem Definitionsbereich von R , $n =$ Stelligkeit von f , gilt aus $a_1 R b_1 \dots a_n R b_n$ folgt $f(a_1, \dots, a_n) R f(b_1, \dots, b_n)$.

Aus dieser Definition folgt insbesondere für 0-stellige Funktionssymbole c , d.h. für Konstantensymbole, stets $c R c$.

Definition 1.11 (Homomorphismus)

Es seien $\mathcal{A}_1 = (A_1, I_1)$ und $\mathcal{A}_2 = (A_2, I_2)$ Interpretationen über Σ . Ein *Homomorphismus* von \mathcal{A}_1 nach \mathcal{A}_2 ist eine Abbildung

$$\varphi : A_1 \rightarrow A_2,$$

so daß für alle $n \in \mathbb{N}$, $f \in F_{\Sigma}$ mit $\alpha_{\Sigma}(f) = n$, $p \in P_{\Sigma}$ mit $\alpha_{\Sigma}(p) = n$ und $d_1, \dots, d_n \in D$ gilt:

1. $\varphi(I_1(f)(d_1, \dots, d_n)) = (I_2(f))(\varphi(d_1), \dots, \varphi(d_n))$
im Falle $n = 0$ also: $\varphi(I_1(f)) = I_2(f)$
2. $(d_1, \dots, d_n) \in I_1(p) \Leftrightarrow (\varphi(d_1), \dots, \varphi(d_n)) \in I_2(p)$
im Falle $n = 0$: $I(p) = I'(p)$.

Ein *Isomorphismus* ist ein bijektiver Homomorphismus.

$\mathcal{A}_1 = (A_1, I_1)$ und $\mathcal{A}_2 = (A_2, I_2)$ heißen *isomorph*, geschrieben $\mathcal{A}_1 = (A_1, I_1) \cong \mathcal{A}_2$ wenn es einen Isomorphismus von $\mathcal{A}_1 = (A_1, I_1)$ auf \mathcal{A}_2 gibt.

Kapitel 2

Aussagenlogik: Syntax und Semantik

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Abbildung 2.1: Eine Sudoku Problem

Der folgende Text geht davon aus, daß der Leser schon eine gewisse Vertrautheit mit der Aussagenlogik mitbringt. Zwar werden die einschlägigen Definitionen noch einmal wiederholt, aber nur in knappster Form.

2.1 Einleitendes Beispiel

2.1.1 Sudoku

Sudoku Probleme, ein aus Japan stammende Art von Denksportaufgaben, sind in letzter Zeit sehr populär geworden. Abb. 2.1.1 zeigt ein typisches, einfaches Sudoku Problem. Die Aufgabenstellung besteht darin, das vorgegebene Diagramm so zu vervollständigen, daß

1. in jeder Zeile jede Ziffer zwischen 1 und 9 mindestens einmal vorkommt,
2. in jeder Spalte jede Ziffer zwischen 1 und 9 mindestens einmal vorkommt,
3. in jeder der neun Teilregionen jede Ziffer zwischen 1 und 9 mindestens einmal vorkommt.

Da in jeder Zelle höchstens eine Ziffer stehen kann, folgt, daß in jeder der oben beschriebenen Situationen jede Ziffer sogar genau einmal vorkommt. Eine Lösung des Sudokus aus Abb. 2.1.1 ist in Abb. 2.1.1 abgebildet. uns interessiert hier die Formulierung von Sudoku Problemen als aussagenlogische

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Abbildung 2.2: Die Lösung des Problems aus Abb. 2.1.1

Erfüllbarkeitsprobleme. Wir führen für jede Zellenposition (i, j) des Sudoku und jede Zahl k zwischen 1 und 9 eine Boolesche Variable

$$D_{i,j}^k$$

ein, mit der Vorstellung, daß $D_{i,j}^k$ den Wert *wahr* hat, wenn auf dem Feld (i, j) die Zahl k steht. Die linke untere Zelle soll dabei die Koordinaten $(1, 1)$ haben. So ist z.B. $D_{9,1}^9$ wahr, wenn in der rechten unteren Ecke die Zahl 9 steht. Die Sudoku Regeln lassen sich mit diesem Vokabular als aussagenlogische Formeln schreiben. Wir betrachten dazu einige Beispiele.

$$D_{1,9}^1 \vee D_{2,9}^1 \vee D_{3,9}^1 \vee D_{4,9}^1 \vee D_{5,9}^1 \vee D_{6,9}^1 \vee D_{7,9}^1 \vee D_{8,9}^1 \vee D_{9,9}^1$$

sagt, daß die Ziffer 1 mindestens einmal in der ersten Zeile vorkommen muß.

$$D_{1,1}^1 \vee D_{1,2}^1 \vee D_{1,3}^1 \vee D_{1,4}^1 \vee D_{1,5}^1 \vee D_{1,6}^1 \vee D_{1,7}^1 \vee D_{1,8}^1 \vee D_{1,9}^1$$

sagt, daß die Ziffer 1 mindestens einmal in der ersten Spalte vorkommen muß.

$$D_{1,1}^1 \vee D_{1,2}^1 \vee D_{1,3}^1 \vee D_{2,1}^1 \vee D_{2,2}^1 \vee D_{2,3}^1 \vee D_{3,1}^1 \vee D_{3,2}^1 \vee D_{3,3}^1$$

sagt, daß die Ziffer 1 mindestens einmal in der Region links unten vorkommen muß.

Die bisherigen Formeln genügen jedoch noch nicht, um das Problem korrekt zu beschreiben. Man muß noch ausdrücken, daß in jeder Zelle höchstens eine Ziffer stehen kann. die Konjunktion der folgenden Formel stellt das für die Zelle $(1, 1)$ sicher:

$$\begin{aligned}
& \neg(D_{1,1}^1 \wedge D_{1,1}^2), \neg(D_{1,1}^1 \wedge D_{1,1}^3), \neg(D_{1,1}^1 \wedge D_{1,1}^4), \neg(D_{1,1}^1 \wedge D_{1,1}^5), \neg(D_{1,1}^1 \wedge D_{1,1}^6), \\
& \neg(D_{1,1}^1 \wedge D_{1,1}^7), \neg(D_{1,1}^1 \wedge D_{1,1}^8), \neg(D_{1,1}^1 \wedge D_{1,1}^9), \neg(D_{1,1}^2 \wedge D_{1,1}^3), \neg(D_{1,1}^2 \wedge D_{1,1}^4), \\
& \neg(D_{1,1}^2 \wedge D_{1,1}^5), \neg(D_{1,1}^2 \wedge D_{1,1}^6), \neg(D_{1,1}^2 \wedge D_{1,1}^7), \neg(D_{1,1}^2 \wedge D_{1,1}^8), \neg(D_{1,1}^2 \wedge D_{1,1}^9), \\
& \neg(D_{1,1}^3 \wedge D_{1,1}^4), \neg(D_{1,1}^3 \wedge D_{1,1}^5), \neg(D_{1,1}^3 \wedge D_{1,1}^6), \neg(D_{1,1}^3 \wedge D_{1,1}^7), \neg(D_{1,1}^3 \wedge D_{1,1}^8), \\
& \neg(D_{1,1}^3 \wedge D_{1,1}^9), \neg(D_{1,1}^4 \wedge D_{1,1}^5), \neg(D_{1,1}^4 \wedge D_{1,1}^6), \neg(D_{1,1}^4 \wedge D_{1,1}^7), \neg(D_{1,1}^4 \wedge D_{1,1}^8), \\
& \neg(D_{1,1}^4 \wedge D_{1,1}^9), \neg(D_{1,1}^5 \wedge D_{1,1}^6), \neg(D_{1,1}^5 \wedge D_{1,1}^7), \neg(D_{1,1}^5 \wedge D_{1,1}^8), \\
& \neg(D_{1,1}^5 \wedge D_{1,1}^9), \neg(D_{1,1}^6 \wedge D_{1,1}^7), \neg(D_{1,1}^6 \wedge D_{1,1}^8), \neg(D_{1,1}^6 \wedge D_{1,1}^9), \neg(D_{1,1}^7 \wedge D_{1,1}^8), \\
& \neg(D_{1,1}^7 \wedge D_{1,1}^9) \neg(D_{1,1}^8 \wedge D_{1,1}^9)
\end{aligned}$$

Kompakter können wir diese Formelmenge beschreiben durch

$$\neg(D_{i,j}^s \wedge D_{i,j}^t)$$

für alle $1 \leq i, j, s, t \leq 9$ mit $s < t$.

Auf diese Weise ergeben sich $81 * 36 = 2916$ Formeln.

Auch die Sudoku Regeln selbst lassen sich in dieser kompakten Weise beschreiben:

Für Zeilen:

$$D_{1,j}^k \vee D_{2,j}^k \vee D_{3,j}^k \vee D_{4,j}^k \vee D_{5,j}^k \vee D_{6,j}^k \vee D_{7,j}^k \vee D_{8,j}^k \vee D_{9,j}^k$$

für alle $1 \leq k, j \leq 9$.

Für Spalten:

$$D_{i,1}^k \vee D_{i,2}^k \vee D_{i,3}^k \vee D_{i,4}^k \vee D_{i,5}^k \vee D_{i,6}^k \vee D_{i,7}^k \vee D_{i,8}^k \vee D_{i,9}^k$$

für alle $1 \leq k, i \leq 9$.

Für Regionen:

$$\begin{aligned}
& D_{1+3*i,1+3*j}^k \vee D_{2+3*i,1+3*j}^k \vee D_{3+3*i,1+3*j}^k \vee D_{1+3*i,2+3*j}^k \vee D_{2+3*i,2+3*j}^k \vee D_{3+3*i,2+3*j}^k \\
& \vee D_{1+3*i,3+3*j}^k \vee D_{2+3*i,3+3*j}^k \vee D_{3+3*i,3+3*j}^k
\end{aligned}$$

für $1 \leq k \leq 9$ und $0 \leq i, j \leq 2$.

Ingesamt braucht man $2916 + 3 * 81 = 3159$ Formeln.

2.1.2 Das Acht-Damen-Problem

Wir betrachten als einführendes Beispiel ein bekanntes kombinatorisches Problem. Bei dem sogenannten *8-Damen-Problem* geht es darum acht Damen so auf einem Schachbrett (mit den üblichen 64 Feldern) zu plazieren, daß sie sich gegenseitig nicht bedrohen.

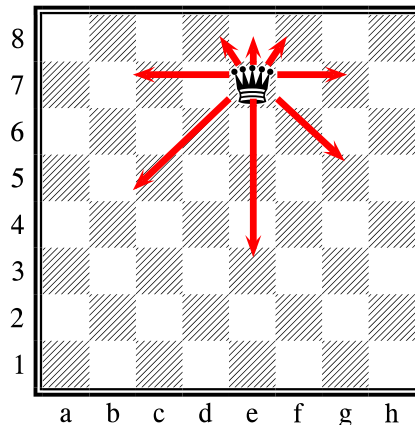


Abbildung 2.3: Die Dame im Schachspiel

Nach den Schachregeln bedroht die Dame alle Felder die sie horizontal, vertikal und diagonal in beide Richtungen erreichen kann, siehe Abb. 2.1.2. Man sieht, daß die Dame in der in der Abbildung gezeigten Position schon 24 Felder (einschließlich des von ihr besetzten Feldes) beherrscht. Eine der insgesamt 92 (12 wenn man symmetrische Lösungen nur einmal zählt) möglichen Lösungen des 8-Damen-Problems ist in Abb. 2.1.2 zu sehen.

Uns interessiert hier ein aussagenlogischer Lösungsansatz. Dazu führen wir für jedes Feld des Schachbretts eine Boolesche Variable $D_{i,j}$ ein, mit der Vorstellung, daß $D_{i,j}$ den Wert *wahr* hat, wenn auf dem Feld (i,j) eine Dame steht, sonst hat $D_{i,j}$ den Wert *falsch*. Wir benutzen hier zur Notation von Positionen auf dem Schachbrett nicht die in der Schachliteratur übliche Bezeichnung durch Buchstaben und Ziffern, sondern benutzen (diskrete) kartesische Koordinaten. Die Dame in Abb. 2.1.2 steht in dieser Notation auf dem Feld $(5,7)$. Als nächstes beschreiben wir die durch die Problemstellung geforderten Einschränkungen. Steht z.B. auf dem Feld $(1,1)$ eine Dame, dann kann auf den Feldern der ersten Spalte $(1,2), (1,3), (1,4), (1,5), (1,6), (1,7), (1,8)$, der ersten Reihe $(2,1), (3,1), (4,1), (5,1), (6,1), (7,1), (8,1)$ und der Diagonalen $(2,2), (3,3), (4,4), (5,5), (6,6), (7,7), (8,8)$ keine weitere Dame stehen. Aussagenlogisch formuliert heißt das, daß die drei folgenden Formeln den

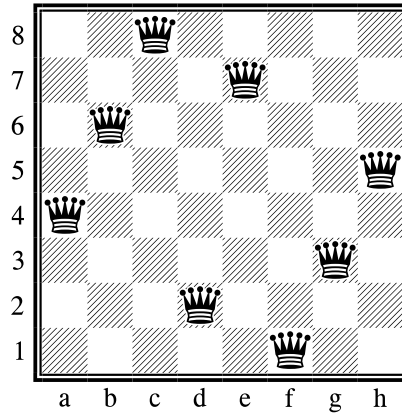


Abbildung 2.4: Eine Lösung des 8-Damenproblems

Wahrheitswert *wahr* erhalten sollen:

$$\begin{aligned}
 D_{1,1} &\rightarrow \neg D_{1,2} \wedge \neg D_{1,3} \wedge \neg D_{1,4} \wedge \neg D_{1,5} \wedge \neg D_{1,6} \wedge \neg D_{1,7} \wedge \neg D_{1,8} \\
 D_{1,1} &\rightarrow \neg D_{2,1} \wedge \neg D_{3,1} \wedge \neg D_{4,1} \wedge \neg D_{5,1} \wedge \neg D_{6,1} \wedge \neg D_{7,1} \wedge \neg D_{8,1} \\
 D_{1,1} &\rightarrow \neg D_{2,2} \wedge \neg D_{3,3} \wedge \neg D_{4,4} \wedge \neg D_{5,5} \wedge \neg D_{6,6} \wedge \neg D_{7,7} \wedge \neg D_{8,8}
 \end{aligned}$$

Für $D_{5,7}$ ergeben sich, jetzt gleich in aussagenlogischer Notation geschrieben, die folgenden Einschränkungen:

$$\begin{aligned}
 D_{5,7} &\rightarrow \neg D_{5,8} \wedge \neg D_{5,6} \wedge \neg D_{5,5} \wedge \neg D_{5,4} \wedge \neg D_{5,3} \wedge \neg D_{5,2} \wedge \neg D_{5,1} \\
 D_{5,7} &\rightarrow \neg D_{4,7} \wedge \neg D_{3,7} \wedge \neg D_{2,7} \wedge \neg D_{1,7} \wedge \neg D_{6,7} \wedge \neg D_{7,7} \wedge \neg D_{8,7} \\
 D_{5,7} &\rightarrow \neg D_{6,8} \wedge \neg D_{4,6} \wedge \neg D_{3,5} \wedge \neg D_{2,4} \wedge \neg D_{1,3} \\
 D_{5,7} &\rightarrow \neg D_{4,8} \wedge \neg D_{6,6} \wedge \neg D_{7,5} \wedge \neg D_{8,4}
 \end{aligned}$$

Für jedes Feld (i, j) sei $FE_{i,j}$ die Konjunktion der Formeln, welche, wie in den beiden betrachteten Beispielen $(i, j) = (1, 1)$ und $(i, j) = (5, 7)$, die Problemeinschränkungen für dieses Feld beschreiben. Wir müssen noch die Forderung ausdrücken, daß für genau acht Positionen (i, j) die Boolesche Variable $D_{i,j}$ wahr sein soll. Das erreichen wir, indem wir für jedes k , $1 \leq k \leq 8$ verlangen daß die Formeln R_k wahr sein soll:

$$D_{1,k} \vee D_{2,k} \vee D_{3,k} \vee D_{4,k} \vee D_{5,k} \vee D_{6,k} \vee D_{7,k} \vee D_{8,k}$$

Ein Lösung des 8-Damen-Problems besteht nun darin eine Belegung aller 64 Booleschen Variablen $D_{i,j}$ zu finden, so daß alle Formeln $FE_{i,j}$ und R_k wahr werden. In aussagenlogischer Terminologie nennt man diese Aufgabenstellung ein *Erfüllbarkeitsproblem*. Wir werden im folgenden Methoden zur Lösung solcher Probleme kennen lernen.

2.2 Syntax der Aussagenlogik

Die Zeichen, aus denen aussagenlogische Formeln aufgebaut werden, lassen sich in zwei Kategorien unterteilen: in logische Zeichen und in einen *Signatur* genannten Teil.

Definition 2.1 (Logische Zeichen)

1 Symbol für den Wahrheitswert „wahr“

0 Symbol für den Wahrheitswert „falsch“

\neg Negationssymbol („nicht“)

\wedge Konjunktionssymbol („und“)

\vee Disjunktionssymbol („oder“)

\rightarrow Implikationssymbol („wenn ... dann“, „impliziert“)

\leftrightarrow Symbol für beiderseitige Implikation („genau dann, wenn“)

(,) die beiden Klammern

Definition 2.2 (Signatur)

Eine (aussagenlogische) *Signatur* ist eine abzählbare Menge Σ von Symbolen, etwa $\Sigma = \{P_0, P_1, \dots\}$.

Genau genommen gibt es nicht *die* Aussagenlogik, sondern eine Familie von Aussagenlogiken. Erst durch die Festlegung einer Signatur entsteht eine *bestimmte* aussagenlogische Sprache.

Das im folgenden in Bezeichnungen verwendete Suffix „0“ zeigt an, daß wir uns in der Aussagenlogik befinden.

Definition 2.3 (Aussagenlogische Formeln, For_0)

Zur Signatur Σ ist For_0_Σ , die Menge der *Formeln über Σ* (oder der *Aussagen über Σ*) induktiv definiert durch

1. $\mathbf{1}, \mathbf{0} \in For_0_\Sigma, \Sigma \subseteq For_0_\Sigma$

2. Mit A, B sind auch

$$\neg A, (A \wedge B), (A \vee B), (A \rightarrow B), (A \leftrightarrow B)$$

Elemente von For_0_Σ

Wir nennen die Sonderzeichen mit Ausnahme der Klammern auch die *logischen Operatoren*, unter ihnen $\mathbf{1}$, $\mathbf{0}$ die *logischen Konstanten*. Die Elemente von Σ heißen auch *atomare Aussagen*, *Atome* oder *Aussagevariablen*.

Wenn klar ist, um welches Σ es sich handelt, schreiben wir oft einfach $For0$ statt $For0_\Sigma$.

2.2.1 Strukturelle Induktion

Ein wichtiges Prinzip um Sätze in der Aussagenlogik zu beweisen ist die

Lemma 2.4 (Strukturelle Induktion)

Gilt für eine Eigenschaft Eig

1. $\mathbf{1}$, $\mathbf{0}$ und jedes Atom $p \in \Sigma$ haben die Eigenschaft Eig
2. Für beliebige $A, B \in For0_\Sigma$:
 - Hat A die Eigenschaft Eig , dann auch $\neg A$.
 - Haben A, B die Eigenschaft Eig , dann auch $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, $(A \leftrightarrow B)$.

dann gilt Eig für alle $A \in For0_\Sigma$.

Man nennt die *strukturelle Induktion* auch Induktion nach dem Aufbau der Formeln.

Eine Variante des Beweisprinzips der strukturellen Induktion ist die *Definition einer Funktion* ϕ auf $For0_\Sigma$ durch strukturelle Induktion durchführen:

Lemma 2.5

Ist eine Funktion f

1. eindeutig definiert auf $\mathbf{1}$, $\mathbf{0}$ und den Atomen.
2. sind $f(\neg A)$, $f((A \wedge B))$, $f((A \vee B))$, $f((A \rightarrow B))$, $f((A \leftrightarrow B))$ eindeutig definiert unter der Annahme, es seien $f(A)$, $f(B)$ schon definiert

dann ist f auf der gesamten Menge $For0_\Sigma$ eindeutig definiert.

Siehe Übungsaufgabe 2.2.1

2.2.2 Einfache Fakten zur Syntax, Abkürzungen

Definition 2.6 (Teilformeln)

Eine *Teilformel* einer Formel A ist ein Teilwort von A , welches Formel ist. Es handelt sich um eine *echte* Teilformel, wenn sie von A verschieden ist.

Zwei Teilwörter u, v einer Formel A *liegen disjunkt*, wenn kein Auftreten eines Zeichens in A sowohl in u wie in v stattfindet. Ein *Präfix* von A ist ein Teilwort von A , mit dem A (von links her gelesen) beginnt.

Abkürzungen

1. Ganz außen stehende Klammern in einer Formel dürfen weggelassen werden.
2. Klammern dürfen weggelassen werden gemäß der Prioritätsregel: \wedge, \vee binden stärker als $\rightarrow, \leftrightarrow$. (Achtung: nach Def. 2.1 ist $\neg A \wedge B$ immer als $(\neg A \wedge B)$ und nicht als $\neg(A \wedge B)$ zu lesen, da zu \neg keine Klammern gehören.)

Weitere Regeln zur Klammereinsparung folgen später.

Man beachte, daß dies nur eine „Kurzschrift“ ist. Unsere „offizielle“ Sprache bleibt, wie oben definiert.

2.2.3 Übungsaufgaben

Übungsaufgabe 2.2.1

Man zeige durch strukturelle Induktion:

1. Ist $A \in For_0\Sigma$ und sind B, C Teilformeln von A , dann gilt
 - entweder C ist Teilformel von B
 - oder B ist echte Teilformel von C
 - oder B, C liegen disjunkt.
2. Ist B Teilformel von $A \in For_0\Sigma$ und zugleich Präfix von A , dann sind A, B identisch.

Übungsaufgabe 2.2.2

1. Σ sei endlich.
Geben Sie eine kontextfreie Grammatik für $For_0\Sigma$ an.

2. Lösen Sie dieselbe Aufgabe für unendliches $\Sigma = \{p_0, p_1, \dots\}$.

Hinweis: Schreibe P_i als $P \underbrace{|\dots|}_{i \text{ Striche}}$.

3. Beschränkt man sich auf die Sonderzeichen $\{(\,), \mathbf{0}, \rightarrow\}$ (siehe 2.3.1), so findet man auch eine SLL1-Grammatik (zur Wahl der richtigen Produktion der Grammatik wird jeweils nur das vorderste Zeichen gebraucht).

2.3 Semantik der Aussagenlogik

Definition 2.7 (Wahrheitswerte)

Für alles Folgende seien zwei feste, ansonsten beliebige Objekte \mathbf{W}, \mathbf{F} ausgezeichnet, die wir die beiden *Wahrheitswerte* nennen. (Vorausgesetzt wird nur, daß beide voneinander verschieden sind.)

Definition 2.8 (Semantik der Aussagenlogik)

Es sei Σ eine aussagenlogische Signatur. Eine *Interpretation über Σ* ist eine beliebige Abbildung $I : \Sigma \rightarrow \{\mathbf{W}, \mathbf{F}\}$. Zu jeder Interpretation I über Σ wird nun die zugehörige *Auswertung* der Formeln über Σ definiert als die Abbildung

$$val_I : For0_\Sigma \rightarrow \{\mathbf{W}, \mathbf{F}\}$$

mit:

$$\begin{aligned} val_I(\mathbf{1}) &= \mathbf{W} \\ val_I(\mathbf{0}) &= \mathbf{F} \\ val_I(P) &= I(P) \quad \text{für jedes } P \in \Sigma \end{aligned}$$

$$val_I(\neg A) = \begin{cases} \mathbf{F} & \text{falls } val_I(A) = \mathbf{W} \\ \mathbf{W} & \text{falls } val_I(A) = \mathbf{F} \end{cases}$$

val_I auf $A \wedge B, A \vee B, A \rightarrow B, A \leftrightarrow B$ gemäß Tabelle (2.1)

$val_I(A), val_I(B)$	$val_I(C)$ für $C =$			
	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$
\mathbf{W}, \mathbf{W}	\mathbf{W}	\mathbf{W}	\mathbf{W}	\mathbf{W}
\mathbf{W}, \mathbf{F}	\mathbf{F}	\mathbf{W}	\mathbf{F}	\mathbf{F}
\mathbf{F}, \mathbf{W}	\mathbf{F}	\mathbf{W}	\mathbf{W}	\mathbf{F}
\mathbf{F}, \mathbf{F}	\mathbf{F}	\mathbf{F}	\mathbf{W}	\mathbf{W}

Tabelle 2.1:

(Man beachte, daß dies eine Definition durch strukturelle Induktion ist.)

Definition 2.9

Ein *Modell* einer Formel $A \in For0_\Sigma$ ist eine Interpretation I über Σ mit $val_I(A) = \mathbf{W}$. Zu einer Formelmenge $M \subseteq For0_\Sigma$ ist ein *Modell von M* eine Interpretation I , welche Modell von jedem $A \in M$ ist.

Man beachte, daß die leere Menge jede Interpretation zum Modell hat.

Beispiel 2.10

Bei der Auswertung einer Formel werden der Übersichtlichkeit halber die Werte der Teilformeln mitnotiert.

$$\Sigma = \{P, Q, R\}$$

$$I : I(P) = \mathbf{W}, I(Q) = \mathbf{F}, I(R) = \mathbf{W}.$$

Wir berechnen $val_I((P \wedge \neg R) \rightarrow \neg(R \vee Q))$

P	Q	R	$\neg R$	$(P \wedge \neg R)$	$(R \vee Q)$	$\neg(R \vee Q)$	$(P \wedge \neg R) \rightarrow \neg(R \vee Q)$
\mathbf{W}	\mathbf{F}	\mathbf{W}	\mathbf{F}	\mathbf{F}	\mathbf{W}	\mathbf{F}	\mathbf{W}

Tabelle 2.2:

2.3.1 Wahrheitstabeln, Boole'sche Funktionen, Basen

Es sei $A \in For0_\Sigma$ gegeben. Welche Werte durchläuft $val_I(A)$ für die verschiedenen Interpretationen I ? Nach Definition ist klar, daß diese nur von den Werten $I(P)$ für die in A *tatsächlich auftretenden* Atome P abhängen. Man erhält also (für festes A) die Funktion, welche jedem I zuordnet $val_I(A)$, als eine *endliche Tabelle*. Diese heißt die *Wahrheitstafel* von A .

Beispiel 2.11

...

Eine *Boole'sche Funktion* ist eine Funktion von $\{\mathbf{W}, \mathbf{F}\}^n$ nach $\{\mathbf{W}, \mathbf{F}\}$, für ein $n \in \mathbb{N}$. (\mathbb{N} ist die Menge der natürlichen Zahlen einschließlich 0.) Ist n die Anzahl der in einer Formel A auftretenden Atome, und legt man für diese eine *bestimmte Reihenfolge* fest – identifiziert sie also mit Argumentstellen –, so liefert die Wahrheitstafel von A eine Boole'sche Funktion $\{\mathbf{W}, \mathbf{F}\}^n \rightarrow \{\mathbf{W}, \mathbf{F}\}$. Es ist bekannt (*leichte Übung*), daß sich umgekehrt auch *jede* Boole'sche Funktion als Wahrheitstafel einer Formel in dieser Weise erhalten läßt.

Basen

Es ist ebenfalls bekannt, daß unsere Sprache viel zu reichhaltig ist, wenn man lediglich beabsichtigt, jede Boole'sche Funktion notieren zu können. Statt der hier gewählten logischen Operatoren

$$\mathbf{1}, \mathbf{0}, \neg, \wedge, \vee, \rightarrow, \leftrightarrow$$

hätte man, um bloß diesen Zweck zu erreichen, auch etwa wählen können

$\mathbf{0}, \rightarrow$
 oder: \neg, \rightarrow
 oder: \neg, \wedge

und andere „Sätze von Operatoren“ mehr, jeweils zusammen mit der entsprechenden Einschränkung der Semantik. Wir nennen jede solche Menge – jede Menge von logischen Operatoren, so daß durch die mit ihnen gebildeten Formeln sich alle Boole'schen Funktionen notieren lassen – eine *Basis*. Bekanntlich gibt es auch einelementige Basen, nämlich:

$\{\mid\}$ mit: $A \mid B$ steht für $\neg(A \wedge B)$ (NAND)
 $\{\downarrow\}$ mit: $A \downarrow B$ steht für $\neg(A \vee B)$ (NOR).

2.3.2 Allgemeingültigkeit und Erfüllbarkeit

Definition 2.12

$A \in For0_\Sigma$ heißt *allgemeingültig* (über Σ) : $\Leftrightarrow val_I(A) = \mathbf{W}$ für jede Interpretation I über Σ .

A heißt *erfüllbar* (über Σ) : \Leftrightarrow es gibt eine Interpretation I über Σ mit $val_I(A) = \mathbf{W}$.

Es ist also A allgemeingültig genau dann, wenn jede Interpretation Modell von A ist, und A ist erfüllbar genau dann, wenn es ein Modell hat.

Korollar 2.13

Es gilt

A erfüllbar $\Leftrightarrow \neg A$ nicht allgemeingültig,
 A allgemeingültig $\Leftrightarrow \neg A$ nicht erfüllbar.

Man nennt die allgemeingültigen Formeln auch *Tautologien*. (Erst in der Prädikatenlogik werden beide Begriffe differieren.)

Bemerkung

Um zu testen, ob A allgemeingültig (bzw. erfüllbar) ist, genügt es, die Wahrheitstafel von A aufzustellen und abzulesen, ob jede Zeile – jede Belegung der Atome in A mit Werten $\in \{\mathbf{W}, \mathbf{F}\}$ – den Wert \mathbf{W} ergibt (bzw. dies für mindestens eine Zeile der Fall ist). Die Allgemeingültigkeit (Erfüllbarkeit) ist also entscheidbar, und zwar in einer Zeit, welche exponentiell ist, in der Anzahl der in der gegebenen Formel auftretenden Signatursymbole, oder auch in der Länge der Formel.

Zu gegebenem I ist die Berechnung von $val_I(A)$ linear (in der Anzahl der Signatursymbole in A , oder der Länge von A) (Übung). Also liegt das Problem SAT, die Erfüllbarkeit einer aussagenlogischen Formel zu entscheiden, in der Klasse NP der nicht deterministisch polynomialen Probleme. SAT ist sogar *NP-vollständig*: Gäbe es einen (deterministischen) polynomialen Entscheidungsalgorithmus für die Erfüllbarkeit, dann wäre $NP = P$, d. h. jedes nichtdeterministisch-polynomiale Entscheidungsproblem auch deterministisch-polynomial. Die Frage, ob dies so sei, ist das wohl bekannteste offene Problem der theoretischen Informatik.

Es gibt interessante Teilmengen von $For0_{\Sigma}$, für welche die Erfüllbarkeit polynomial entscheidbar ist. Wir werden hierauf noch kurz zurückkommen.

2.3.3 Einige allgemeingültige Formeln

Es folgt eine Liste von Schemata für allgemeingültige Formeln. (D. h. jede Ersetzung von A, B, C unten durch irgendwelche Formeln – jeweils dieselbe Formel für A bzw. B bzw. C – liefert eine allgemeingültige Formel. Entsprechend bei spezielleren Ersetzungen, wie unten angegeben.)

$\mathbf{1}$	Wahrheitswerte
$\neg \mathbf{0}$	Wahrheitswerte
$A \rightarrow A$	Selbstimplikation
$\neg A \vee A$	Tertium non datur
$A \rightarrow (B \rightarrow A)$	Abschwächung
$\mathbf{0} \rightarrow A$	Ex falso quodlibet

Alle folgenden Tautologien sind „Äquivalenzen“ (vgl. 2.16).

$$A \wedge A \leftrightarrow A$$

$$A \vee A \leftrightarrow A$$

Idempotenz von \wedge, \vee

$$A \wedge B \leftrightarrow B \wedge A$$

$$A \vee B \leftrightarrow B \vee A$$

Kommutativität von \wedge, \vee

$$(A \wedge B) \wedge C \leftrightarrow (A \wedge (B \wedge C))$$

$$((A \vee B) \vee C) \leftrightarrow (A \vee (B \vee C))$$

Assoziativität von \wedge, \vee

$$(A \leftrightarrow (B \leftrightarrow C)) \leftrightarrow$$

$$(A \leftrightarrow B) \leftrightarrow C$$

Assoziativität von \leftrightarrow

$$A \wedge (A \vee B) \leftrightarrow A$$

$$A \vee (A \wedge B) \leftrightarrow A$$

Absorption von \wedge, \vee

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

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

Distributivität von \wedge, \vee

$$\neg\neg A \leftrightarrow A$$

Doppelnegation

$$(A \rightarrow B) \leftrightarrow (\neg B \rightarrow \neg A)$$

Kontraposition

$$(\neg A \rightarrow A) \leftrightarrow A$$

Selbstbeweis

$$(A \rightarrow (B \rightarrow C)) \leftrightarrow$$

$$((A \rightarrow B) \rightarrow (A \rightarrow C))$$

Verteilen und Ausklammern von \rightarrow

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

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

De Morgan'sche Gesetze

$$\neg A \leftrightarrow (A \rightarrow \mathbf{0})$$

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

$$A \wedge B \leftrightarrow \neg(A \rightarrow \neg B)$$

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

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

$$(A \leftrightarrow B) \leftrightarrow (A \rightarrow B) \wedge (B \rightarrow A)$$

Weitere Möglichkeiten zum Umschreiben logischer Operatoren

$$\begin{aligned}
A \vee \mathbf{1} &\leftrightarrow \mathbf{1} \\
A \wedge \mathbf{1} &\leftrightarrow A \\
A \vee \mathbf{0} &\leftrightarrow A \\
A \wedge \mathbf{0} &\leftrightarrow \mathbf{0}
\end{aligned}$$

2.3.4 Semantische Folgerbarkeit

Definition 2.14

Es seien: Σ eine Signatur, $M \subseteq For0_\Sigma$, $A \in For0_\Sigma$.

Wir definieren

$$M \models A \text{ gdw Jedes Modell von } M \text{ ist auch Modell von } A$$

$M \models_\Sigma A$ wird gelesen als: aus M folgt A

Statt $\emptyset \models A$ schreiben wir kurz $\models A$, statt $\{B\} \models A$ schreiben wir $B \models A$.
Falls erforderlich schreiben wir genauer \models_Σ anstelle von \models .

Korollar 2.15

Es gilt

1. $\models A \Leftrightarrow A$ ist allgemeingültig.
2. $\models \neg A \Leftrightarrow A$ ist unerfüllbar.
3. $A \models B \Leftrightarrow \models A \rightarrow B$
4. $M \cup \{A\} \models B \Leftrightarrow M \models A \rightarrow B$

Definition 2.16

$A, B \in For0_\Sigma$ heißen *logisch äquivalent* $:\Leftrightarrow A \models_\Sigma B$ und $B \models_\Sigma A$. Wir benutzen gelegentlich die symbolische Schreibweise $A \equiv B$ zur Bezeichnung der logischen Äquivalenz von A und B .

Korollar 2.17

Zwei Formeln A, B sind logisch äquivalent genau dann, wenn $A \leftrightarrow B$ eine Tautologie ist.

Beweis A, B logisch äquivalent

$$\Leftrightarrow val_I(A) = val_I(B) \text{ für alle Interpretationen } I \text{ (über } \Sigma)$$

(d. h. A und B haben dieselben Modelle)

$$\Leftrightarrow \models_\Sigma A \leftrightarrow B$$

$$\Leftrightarrow A \leftrightarrow B \text{ ist allgemeingültig.}$$

Die Schemata allgemeingültiger Formeln der Form $\dots \leftrightarrow \dots$ in 2.3.2 liefern uns also viele Beispiele logischer Äquivalenz. ■

Treten in A und B dieselben Atome auf, so ist die logische Äquivalenz von A und B damit gleichbedeutend, daß A und B dieselbe Wahrheitstafel haben.

Lemma 2.18

Für $A, A', B, B' \in For0_\Sigma$ gelte, daß A logisch äquivalent zu A' und B logisch äquivalent zu B' ist. Dann sind auch logisch äquivalent

- $\neg A$ und $\neg A'$
- $A \rightarrow B$ und $A' \rightarrow B'$
- $A \wedge B$ und $A' \wedge B'$
- $A \vee B$ und $A' \vee B'$
- $A \leftrightarrow B$ und $A' \leftrightarrow B'$

Beweis: Direktes Nachrechnen. ■

Offensichtlich ist die logische Äquivalenz eine Äquivalenzrelation auf $For0_\Sigma$. Aus dem obigen Lemma folgt dann:

Korollar 2.19

Logische Äquivalenz ist bezüglich der aussagenlogischen Operatoren eine Kongruenzrelation auf $For0_\Sigma$. Insbesondere gilt für beliebige $A \in For0_\Sigma$

- A allgemeingültig $\Leftrightarrow A$ logisch äquivalent zu **1**
- A unerfüllbar $\Leftrightarrow A$ logisch äquivalent zu **0**.

Korollar 2.20

(Ersetzungstheorem) Es seien $A, A', B, B' \in For0_\Sigma$, B eine Teilformel von A und B' logisch äquivalent zu B . Ferner entstehe A' aus A , indem an irgendwelchen Stellen, wo B in A auftritt, B durch B' ersetzt wird. Dann ist A' logisch äquivalent zu A .

Beweis Übung (Strukturelle Induktion)

Das folgende Lemma gehörte schon immer zum festen Bestand der Aussagen über die Ableitbarkeit in der Aussagenlogik, war aber in Vergessenheit geraten, da man es nur von theoretischem Interesse hielt. Das änderte sich schlagartig, als das Interpolationslemma in der Arbeit [McM03] über Modellprüfungsverfahren (model checking) eine ausschlaggebende Rolle spielte. (Siehe auch <http://www.cs.utah.edu/tphols2004/mcmillan.abstract.html>)

Definition 2.21 (Interpolante)

Seien A, B aussagenlogische Formeln, so daß $A \rightarrow B$ eine Tautologie ist. Eine Formel C heißt eine *Interpolante* von $A \rightarrow B$, falls

1. $A \rightarrow C$ und $C \rightarrow B$ Tautologien sind und
2. in C nur solche aussagenlogischen Atome $P \in \Sigma$ vorkommen, die sowohl in A als auch in B vorkommen.
An eventuelle Vorkommen von $\mathbf{1}$ und $\mathbf{0}$ in C werden keinerlei Einschränkungen gemacht.

Lemma 2.22 (Craigsches Interpolationslemma)

Zu je zwei aussagenlogische Formeln A, B , so daß $A \rightarrow B$ eine Tautologie ist, existiert eine Interpolante.

Beweis Seien P_1, \dots, P_n alle in A vorkommenden aussagenlogischen Atome, die nicht in B vorkommen. Für Konstanten $c_i \in \{\mathbf{1}, \mathbf{0}\}$ bezeichnen wir mit $A[c_1, \dots, c_n]$ die Formeln, die aus A hervorgeht, indem P_i durch c_i ersetzt wird für alle $1 \leq i \leq n$. Wir setzen

$$C \equiv \bigvee_{(c_1, \dots, c_n) \in \{\mathbf{1}, \mathbf{0}\}^n} A[c_1, \dots, c_n]$$

Offensichtlich kommen in C nur noch aussagenlogische Atome vor, die A und B gemeinsam sind.

Sei jetzt I eine Interpretation mit $val_I(A) = \mathbf{W}$, dann gilt auch $val_I(A[c_1, \dots, c_n]) = \mathbf{W}$ für $c_i = I(P_i)$. Damit gilt auch $val_I(C) = \mathbf{W}$. Insgesamt haben wir also schon $\models A \rightarrow C$ gezeigt.

Sei jetzt I eine Interpretation mit $val_I(C) = \mathbf{W}$. Für (mindestens) eine Wahl von Konstanten $(c_1, \dots, c_n) \in \{\mathbf{1}, \mathbf{0}\}^n$ gilt also $val_I(A[c_1, \dots, c_n]) = \mathbf{W}$. Dieselbe Aussage können wir auch anders schreiben: Wir definieren die Belegung J durch

$$J(P) = \begin{cases} c_i & \text{falls } P = P_i \text{ für } 1 \leq i \leq n \\ I(P) & \text{sonst} \end{cases}$$

Offensichtlich liefern $val_J(A)$ und $val_I(A[c_1, \dots, c_n])$ dasselbe Ergebnis. Also haben wir $val_J(A) = \mathbf{W}$. Nach der Voraussetzung gilt also auch $val_J(B) = \mathbf{W}$. Das ist schön, aber eigentlich wollten wir $val_I(B)$ zeigen. Das ist aber auch klar, da I und J sich nur für die aussagenlogische Atome P_1, \dots, P_n unterscheiden, die nicht in B vorkommen. ■

2.3.5 Bemerkung zur Notation

Im Laufe des bisherigen Textes sind die Symbole \leftrightarrow , $=$, \equiv , \Leftrightarrow eingeführt oder benutzt worden, die in verschiedenen Zusammenhängen die Gleichheit oder Gleichwertigkeit bezeichnen. Es lohnt sich einen Moment über die Unterschiede nachzudenken.

Das einfachste Symbol ist \leftrightarrow . Es kommt nur in Formeln der Aussagenlogik vor. Wenn A und B Formeln sind, dann gilt das auch für $A \leftrightarrow B$. Die übrigen Symbole $=$, \equiv , \Leftrightarrow sind Abkürzungen. Die Benutzung des Gleichheitszeichens $=$ ist selbst dem gebildeten Laien geläufig, das Symbol \equiv wurde in Definition 2.16 als symbolische Abkürzung für die Äquivalenz von Formeln eingeführt. Anstelle des Satzes „Stehen A und B für die gleiche Formel, dann sind sie trivialerweise logisch äquivalent“ kann man dann auch kürzer schreiben „Für $A = B$ gilt trivialerweise $A \equiv B$. Das Zeichen \Leftrightarrow ist eine Abkürzung für die Redewendung *genau dann wenn*, die man auch durch *gdw* abkürzen könnte.

Einzig \leftrightarrow ist unter den betrachteten Symbolen ein Zeichen der *Objektebene*; die anderen gehören der *Metaebene* an.

2.3.6 Übungsaufgaben

Übungsaufgabe 2.3.1

Wir nennen zwei aussagenlogische Formeln A, B *wechselseitig inkonsistent* wenn $\neg(A \wedge B)$ eine Tautologie ist, i.e. $\models \neg(A \wedge B)$.

Zeigen Sie die folgende Behauptung.

Zu zwei wechselseitig inkonsistenten aussagenlogische Formeln A, B gibt es eine Formel C die nur Atome enthält, die in A und B vorkommen mit

$$\models A \rightarrow C$$

C und B sind wechselseitig inkonsistent

Das ist die Formulierung, in der das Interpolationslemma in der Arbeit von McMillan [McM03] benutzt wird. Auch die Terminologie der wechselseitigen Inkonsistenz ist daraus entnommen.

Übungsaufgabe 2.3.2

Seien C_1, C_2 beides Interpolanten für die Implikation

$$A \rightarrow B.$$

Dann sind auch $C_1 \wedge C_2$ und $C_1 \vee C_2$ Interpolanten für $A \rightarrow B$.

Übungsaufgabe 2.3.3

Sei $A \rightarrow B$ eine Tautologie.

Seien Q_1, \dots, Q_k alle Variablen in B , die nicht in A vorkommen. Sind c_i Konstanten aus $\{\mathbf{1}, \mathbf{0}\}$ dann bezeichne $B[c_1, \dots, c_k]$, wie im Beweis von Lemma 2.22, die Formel, die aus B entsteht, wenn man alle Vorkommen von Q_i durch c_i ersetzt. Außerdem sei:

$$D \equiv \bigwedge_{(c_1, \dots, c_n) \in \{\mathbf{1}, \mathbf{0}\}^n} B[c_1, \dots, c_n]$$

Zeigen Sie, daß D eine Interpolante von $A \rightarrow B$ ist.

Übungsaufgabe 2.3.4

Sei $A \rightarrow B$ eine Tautologie. Sei C die im Beweis von Lemma 2.22 konstruierte Interpolante von $A \rightarrow B$ und D die Formel aus der vorangegangenen Übungsaufgabe 2.3.3. Zeigen Sie, daß C die stärkste und D die schwächste Interpolante von $A \rightarrow B$ ist, d.h. zeigen Sie, daß für jede Interpolante U

$$C \rightarrow U \text{ und } U \rightarrow B$$

Tautologien sind.

2.4 Normalformen

Wegen der Assoziativgesetze für \wedge und \vee können wir für $A \wedge (B \wedge C)$ oder $(A \wedge B) \wedge C$ kurz $A \wedge B \wedge C$ schreiben, entsprechend $A_1 \wedge \dots \wedge A_n$, entsprechend $A_1 \vee \dots \vee A_n$. Wir sprechen von Konjunktionen bzw. Disjunktionen.

2.4.1 Disjunktive und konjunktive Normalform

Definition 2.23

1. Ein *Literal* ist ein Atom oder ein negiertes Atom.
2. Eine Formel ist in *disjunktiver Normalform* (DNF), wenn sie Disjunktion von Konjunktionen von Literalen ist.
3. Eine Formel ist in *konjunktiver Normalform* (KNF), wenn sie Konjunktion von Disjunktionen von Literalen ist.

Eine Disjunktion von Literalen wird häufig auch eine *Klausel* genannt. Eine Formel in konjunktiver Normalform ist also eine Konjunktion von Klauseln. Die konjunktive Normalform wird deswegen auch *Klauselnormalform* genannt.

Fakten

Es ist bekannt, daß man zu jeder aussagenlogischen Formel A eine logisch äquivalente in disjunktiver Normalform angeben kann und ebenso eine logisch äquivalente in konjunktiver Normalform. Die Algorithmen zur Herstellung dieser beiden Normalformen ergeben sich unmittelbar aus den Tautologien in 2.3.2.

Mittels der logischen Äquivalenzen

Umschreiben von \rightarrow , \leftrightarrow ,
Doppelnegation,
Distributivität,
De Morgan'sche Gesetze,
Assoziativität

läßt sich „von innen nach außen“ durch Ersetzen von Teilformeln durch logisch äquivalente die gewünschte Normalform herstellen. Durch Anwendung der Kommutativität, Idempotenz, Absorption sowie der Schemata im letzten Block in 2.3.2 läßt sich die Formel gegebenenfalls noch vereinfachen.

Ist die Wahrheitstafel einer Formel gegeben, so lassen sich disjunktive und konjunktive Normalform aus dieser „direkt ablesen“. Wir setzen hier als bekannt voraus, wie dies geschieht.

Zu einer Formel A gibt es (auch wenn man von trivialen Umordnungen absieht) i. a. viele verschiedene äquivalente in DNF bzw. KNF. Beide genannten Algorithmen zur Herstellung einer Normalform – der „syntaktische“ wie das „Ablesen aus der Wahrheitstafel“ – liefern nicht notwendig eine „günstige“ (etwa: kürzeste) Formel. Verfahren zur Findung einer kürzestmöglichen DNF (KNF) werden in der Schaltkreistheorie behandelt.

2.4.2 Primimplikanten

Definition 2.24 (Primimplikanten)

Sei C eine beliebige aussagenlogische Formel.

Ein Primimplikant P von C ist

1. eine Konjunktion von Literalen,
2. so daß $P \rightarrow C$ eine Tautologie ist, und
3. für jede Teilkonjunktion P' von P ist $P' \rightarrow C$ keine Tautologie.

Definition 2.25 (Essentielle Primimplikanten)

Sei C eine beliebige aussagenlogische Formel und P ein Primimplikant von C .

P heißt ein essentieller Primimplikant von C , falls für jede Teilmenge \mathcal{Q} von Primimplikanten von C

- für die $C \leftrightarrow \bigvee \mathcal{Q}$ gilt
- $P \in \mathcal{Q}$ gelten muß.

Lemma 2.26

Sei C eine beliebige aussagenlogische Formel, dann ist

$$C \leftrightarrow \bigvee \{P \mid P \text{ ist Primimplikant von } C\}$$

eine Tautologie.

Beweis

Wir haben zu zeigen, daß

$$\begin{aligned}
C &\leftarrow \bigvee \{P \mid P \text{ ist Primimplikant von } C\} \\
&\quad \text{und} \\
C &\rightarrow \bigvee \{P \mid P \text{ ist Primimplikant von } C\}
\end{aligned}$$

beides Tautologien sind. Für die erste Formel ist das eine unmittelbare Konsequenz aus der Definition. Zum Beweis der zweiten Implikation betrachten wir eine Interpretation I mit $val_I(C) = \mathbf{W}$ und wollen $val_I(RS) = \mathbf{W}$ für die rechte Seite RS der Implikation zeigen. Seien A_1, \dots, A_n alle in C vorkommenden Atome. Für jedes i mit $1 \leq i \leq n$ sei

$$L_i = \begin{cases} A_i & \text{falls } val_I(A_i) = \mathbf{W} \\ \neg A_i & \text{falls } val_I(A_i) = \mathbf{F} \end{cases}$$

Außerdem sei $P_I = \bigwedge_i L_i$. Für jede Interpretation J mit $val_J(P_I) = \mathbf{W}$ stimmt J mit I auf allen Atomen, die in C vorkommen, überein. Also gilt auch $val_J(C) = \mathbf{W}$. Damit ist $P_I \rightarrow C$ eine Tautologie. Sei P_I^0 die kürzeste Teilkonjunktion von P_I , so daß $P_I^0 \rightarrow C$ immer noch eine Tautologie ist. Nach Definition ist P_I^0 ein Primimplikant von C und da P_I^0 eine Teilkonjunktion von P_I ist, gilt auch $val_I(P_I^0) = \mathbf{W}$. Damit gilt aber auch für die rechte Seite RS der obigen Implikation $val_I(RS) = \mathbf{W}$. ■

2.4.3 Kurze Konjunktive Normalform

Die konjunktive Normalform spielt auch in dem weiter unten vorzustellenden Resolutionskalkül eine wichtige Rolle. Will man in diesem Kalkül zeigen, daß eine Formel A eine Tautologie ist, so bringt man die Formel $\neg A$ in konjunktive Normalform und zeigt ihre Unerfüllbarkeit.

Beispiel 2.27 (Konjunktive Normalform)

Um zu prüfen, ob

$$A = (\neg P_{1,1} \vee \neg P_{1,2}) \wedge \dots \wedge (\neg P_{n,1} \vee \neg P_{n,2})$$

eine Tautologie ist, wird die Unerfüllbarkeit von

$$\neg A = (P_{1,1} \wedge P_{1,2}) \vee \dots \vee (P_{n,1} \wedge P_{n,2})$$

geprüft. Die konjunktive Normalform von $\neg A$ ist:

$$\bigwedge \{P_{1,f(1)} \vee \dots \vee P_{n,f(n)} \mid f : \{1, \dots, n\} \rightarrow \{1, 2\}\}.$$

Für $n = 3$ ist das:

$$\begin{aligned}
& (P_{1,1} \vee P_{2,1} \vee P_{3,1}) \wedge (P_{1,1} \vee P_{2,1} \vee P_{3,2}) \wedge \\
& (P_{1,1} \vee P_{2,2} \vee P_{3,1}) \wedge (P_{1,1} \vee P_{2,2} \vee P_{3,2}) \wedge \\
& (P_{1,2} \vee P_{2,1} \vee P_{3,1}) \wedge (P_{1,2} \vee P_{2,1} \vee P_{3,2}) \wedge \\
& (P_{1,2} \vee P_{2,2} \vee P_{3,1}) \wedge (P_{1,2} \vee P_{2,2} \vee P_{3,2})
\end{aligned}$$

Das Beispiel 2.27 zeigt eine Schwierigkeit des vorgeschlagenen Tautologienachweises: in $\neg A$ treten $2 * n$ Literale auf, aber in der konjunktiven Normalform sind es $n * 2^n$. Wir wollen hier eine, zwar schon lange bekannte, aber wenig verbreitete Transformation vorstellen, die zu einer beliebigen aussagenlogischen Formel A mit n Literalen eine konjunktive Normalform mit $c * n$ Literalen herstellt für eine Konstante c . Wir erklären zunächst die Methode anhand des Beispiels 2.27. Der allgemeine Fall wird dann in Definition 2.29 behandelt.

Beispiel 2.28 (Kurze konjunktive Normalform)

1.Schritt

$$\begin{aligned}
Q_1 & \leftrightarrow \neg P_{1,1} \vee \neg P_{1,2} \\
& \vdots \\
Q_i & \leftrightarrow \neg P_{i,1} \vee \neg P_{i,2} \\
& \vdots \\
Q_n & \leftrightarrow \neg P_{n,1} \vee \neg P_{n,2} \\
Q_{n+1} & \leftrightarrow Q_1 \wedge Q_2 \\
Q_{n+2} & \leftrightarrow Q_{n+1} \wedge Q_3 \\
& \vdots \\
& \vdots \\
Q_{2n-1} & \leftrightarrow Q_{2n-2} \wedge Q_n \\
& \quad \neg Q_{2n-1}
\end{aligned}$$

2.Schritt:

$$\begin{aligned}
& \neg Q_1 \vee \neg P_{1,1} \vee \neg P_{1,2} \\
Q_1 & \vee \neg(\neg P_{1,1} \vee \neg P_{1,2}) \\
& \vdots \\
& \vdots \\
& \neg Q_i \vee \neg P_{i,1} \vee \neg P_{i,2} \\
Q_i & \vee \neg(\neg P_{i,1} \vee \neg P_{i,2}) \\
& \vdots
\end{aligned}$$

$$\begin{array}{c}
\cdot \\
\neg Q_n \vee \neg P_{n,1} \vee \neg P_{n,2} \\
Q_n \vee \neg(\neg P_{n,1} \vee \neg P_{n,2}) \\
\neg Q_{n+1} \vee (Q_1 \wedge Q_2) \\
Q_{n+1} \vee \neg(Q_1 \wedge Q_2) \\
\neg Q_{n+2} \vee (Q_{n+1} \wedge Q_3) \\
Q_{n+2} \vee \neg(Q_{n+1} \wedge Q_3) \\
\cdot \\
\cdot \\
\neg Q_{2n-1} \vee (Q_{2n-2} \wedge Q_n) \\
Q_{2n-1} \vee \neg(Q_{2n-2} \wedge Q_n) \\
\neg Q_{2n-1}
\end{array}$$

$(\neg A)_{kknf}$:

$$\begin{array}{c}
\neg Q_1 \vee \neg P_{1,1} \vee \neg P_{1,2} \\
(Q_1 \vee P_{1,1}) \wedge (Q_1 \vee P_{1,2}) \\
\cdot \\
\cdot \\
\neg Q_n \vee \neg P_{n,1} \vee \neg P_{n,2} \\
(Q_n \vee P_{n,1}) \wedge (Q_n \vee P_{n,2}) \\
(\neg Q_{n+1} \vee Q_1) \wedge (\neg Q_{n+1} \vee Q_2) \\
Q_{n+1} \vee \neg Q_1 \vee \neg Q_2 \\
\cdot \\
\cdot \\
(\neg Q_{2n-1} \vee Q_{2n-2}) \wedge (\neg Q_{2n-1} \vee Q_n) \\
Q_{2n-1} \vee \neg Q_{2n-2} \vee \neg Q_n \\
\neg Q_{2n-1}
\end{array}$$

Im vorangegangenen Beispiel 2.28 wurde die Idee der kurzen KNF deutlich. Hier folgt jetzt die Definition im allgemeinen Fall. Unsere Darstellung orientiert sich an [Min90]. Eine ausführliche Behandlung findet man auch in [Ede92], pp. 48 - 52.

Definition 2.29 (Kurze konjunktive Normalform)

Sei A eine beliebige aussagenlogische Formel. Wir geben ein Verfahren an, um A in eine KNF, bezeichnet mit A_{kknf} und *kurze konjunktive Normalform von A* genannt, zu transformieren.

Wir setzen voraus, daß in A nicht zwei Negationszeichen unmittelbar aufeinander folgen.

Ist A ein Literal, dann setzten wir $A_{kknf} = A$.

Wir gehen jetzt davon aus, daß A kein Literal ist.

Wir werden im Laufe der Konstruktion von A_{kknf} neue aussagenlogische Atome Q_i einführen, und zwar für jede Teilformel A_i von A von der Form $A_i = A_{i1} \circ A_{i2}$ ein neues. Man beachte, daß eine Teilformel A_i von dieser Form kein Atom sein kann und auch nicht mit einer Negation beginnen kann.

1.Schritt Wir bilden die Konjunktion K der folgenden Formeln.

Für jede Teilformel A_i der Form $A_{i1} \circ A_{i2}$ mit dem zugehörigen neuen aussagenlogischen Atom Q_i gehört

$$Q_i \leftrightarrow (B_{i1} \circ B_{i2})$$

zur Konjunktion K . Zur Erklärung der B_{ij} müssen wir vier Fälle unterscheiden:

$$\begin{aligned} B_{ij} &= A_{ij} && \text{falls } A_{ij} \text{ ein Literal oder ein negiertes Literal ist.} \\ B_{ij} &= Q_{ij} && A_{ij} \text{ ist von der Form } C_1 \circ C_2 \text{ und } Q_{ij} \\ &&& \text{das zugehörige neue Atom} \\ B_{ij} &= \neg Q_{ij} && A_{ij} \text{ ist von der Form } \neg(C_1 \circ C_2) \\ &&& \text{und } Q_{ij} \text{ das zu } (C_1 \circ C_2) \text{ gehörige neue Atom} \end{aligned}$$

Ist A selbst von der Form $C_1 \circ C_1$ und Q_0 das zugehörige neue Atom, dann ist auch das Literal

$$Q_0$$

ein konjunktiver Bestandteil von K . Anderenfalls, das heißt hier, wenn A von der Form $\neg(C_1 \circ C_1)$, dann ist

$$\neg Q_0$$

ein konjunktiver Bestandteil von K , wobei Q_0 das zu $(C_1 \circ C_1)$ gehörige neue Atom ist.

2.Schritt Die Äquivalenzen werden aufgelöst in zwei Implikationen und diese weiter durch \neg und \vee ausgedrückt. Es entsteht die Konjunktion der Formeln

$$\begin{aligned} \neg Q_i \vee (B_{i1} \circ B_{i2}) & \quad 1 \leq i \leq k \\ \neg(B_{i1} \circ B_{i2}) \vee Q_i & \quad 1 \leq i \leq k \end{aligned}$$

verlängert um

$$\neg Q_0 \text{ oder } Q_0$$

3.Schritt Die Teilformeln aus Schritt 2 werden KNF umgeformt. Die Konjunktion der resultierenden Formeln ist A_{kknf} .

Satz 2.30

Die Formel A ist erfüllbar gdw A_{kknf} erfüllbar ist.

Beweis: Ist A ein Literal, dann gilt $A_{kknf} = A$ und es ist nichts zu zeigen.

Anderenfalls sei $K = Q_0 \wedge \bigwedge_{1 \leq i \leq k} Q_i \leftrightarrow B_{i1} \circ B_{i2}$ bzw. $K = \neg Q_0 \wedge \bigwedge_{1 \leq i \leq k} Q_i \leftrightarrow B_{i1} \circ B_{i2}$ die Konjunktion aus Schritt 1. Es genügt zu zeigen, daß A und K erfüllbarkeitsäquivalent sind. Die restlichen Umformungen, von K zu A_{kknf} , sind elementare aussagenlogische Äquivalenzen.

1.Teil A sei erfüllbar, es gibt also eine Interpretation I mit $val_I(A) = \mathbf{W}$. Für die neuen aussagenlogischen Variablen definieren wir

$$I(Q_i) = val_I(A_i),$$

wobei Q_i die für die Teilformel A_i eingeführte neue AL Variable ist.

Wir betrachten zuerst die erste konjunktive Teilformel F von K . Falls A nicht mit einem Negationszeichen beginnt ist $F = Q_0$ und es folgt $val_I(F) = val_I(Q_0) = val_I(A)$. Im Falle $A = \neg A'$ ist $F = \neg Q_0$ und nach Definition ergibt sich $val_I(F) = val_I(\neg Q_0) = val_I(\neg A') = val_I(A)$.

Als nächstes ist für jedes $1 \leq i \leq k$ $val_I(Q_i) \leftrightarrow (B_{i1} \circ B_{i2}) = \mathbf{W}$ zu zeigen. Dazu genügt es, sich zu überlegen, daß $val_I(B_{i1} \circ B_{i2}) = val_I(A_i)$ gilt. Nach Konstruktion gilt $A_i = A_{i1} \circ A_{i2}$. Wir sind fertig, wenn wir $val_I(A_{i1}) = val_I(B_{i1})$ und $val_I(A_{i2}) = val_I(B_{i2})$ gezeigt haben. Das folgt aber unmittelbar aus den Definitionen der B_{ij} und $val_I(Q_j)$.

2.Teil K sei erfüllbar und I eine Interpretation mit $val_I(K) = \mathbf{W}$. Wir zeigen, daß dann für jedes neue Atom Q_i und die zugehörige Teilformel A_i von A gelten muß: $val_I(Q_i) = val_I(A_i)$. Das zeigen wir, durch Induktion über die Komplexität von A_i . Sei also $A_i = A_{i1} \circ A_{i2}$. Nach Induktionsvoraussetzung können wir $val_I(B_{i1}) = val_I(A_{i1})$ und $val_I(B_{i2}) = val_I(A_{i2})$ annehmen. Also auch $val_I(B_{i2} \circ B_{i2}) = val_I(A_i)$. Aus $val_I(K) = \mathbf{W}$ folgt insbesondere $val_I(Q_i) \leftrightarrow (B_{i1} \circ B_{i2}) = \mathbf{W}$ und somit insgesamt, wie gewünscht, $val_I(Q_i) = val_I(A_i)$. Beginnt A nicht mit einer Negation, dann ist Q_0 konjunktiver Teil von K , also auch $val_I(Q_0) = \mathbf{W}$. Nach dem gerade Bewiesenen folgt dann auch $val_I(A) = \mathbf{W}$. Im anderen Fall gilt $A = \neg A'$ und $val_I(\neg Q_0) = \mathbf{W}$, woraus ebenfalls $val_I(A) = \mathbf{W}$ folgt. ■

2.4.4 Shannons Normalform

In diesem Kapitel benutzen wir die Bezeichnungen 0 für den Wahrheitswert **F** und 1 für **W**, gemäß der zugrunde liegenden Literatur.

Neben der DNF und KNF gibt es noch eine Vielzahl von Normalformen. Wir wollen hier kurz auf eine von R. E. BRYANT (siehe [Bry86]) gefundene, graphbasierte Normalform eingehen, die auf einen Ansatz von C. E. SHANNON (siehe [Sha38]) zurückgeht. Außerdem wird diese Normalform behandelt in dem Buch [Chu56] von A. CHURCH in §24, Seite 129 ff. Wir beginnen mit einigen Vorbereitungen und kommen danach auf die eigentlichen Shannon-Graphen zu sprechen.

Definition 2.31 (Shannon-Formeln)

Wir betrachten aussagenlogische Formeln, die aufgebaut sind aus

- dem dreistelligen Operator sh
- den Konstanten 0 und 1
- Aussagevariablen P_1, \dots, P_n, \dots

Wir wollen solche Formeln sh -Formeln nennen und die Menge aller sh -Formeln mit $shFor0$ bezeichnen.

Die Auswertung einer Formel der Form $sh(A_1, A_2, A_3)$ (sh als Referenz auf Shannon) bezüglich einer Interpretation I wird gegeben durch

$$val_I(sh(A_1, A_2, A_3)) = \begin{cases} val_I(A_2) & \text{falls } val_I(A_1) = \mathbf{F} \\ val_I(A_3) & \text{falls } val_I(A_1) = \mathbf{W} \end{cases}$$

oder in Tabellenform:

P_1	W	W	W	W	F	F	F	F
P_2	W	W	F	F	W	W	F	F
P_3	W	F	W	F	W	F	W	F
$sh(P_1, P_2, P_3)$	W	F	W	F	W	W	F	F

Lemma 2.32

Es gelten die folgenden Äquivalenzen:

1. $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \wedge P_2) \vee (P_1 \wedge P_3)$
2. $sh(0, P_2, P_3) \leftrightarrow P_2, sh(1, P_2, P_3) \leftrightarrow P_3$

3. $sh(P, 0, 1) \leftrightarrow P$
4. $sh(P, 1, 0) \leftrightarrow \neg P$
5. $sh(P_1, P_2, P_2) \leftrightarrow P_2$
6. $sh(sh(P_1, P_2, P_3), P_4, P_5) \leftrightarrow sh(P_1, sh(P_2, P_4, P_5), sh(P_3, P_4, P_5))$
7. $A \leftrightarrow sh(P, A_{P=0}, A_{P=1})$
8. $\neg sh(P_1, P_2, P_3) \leftrightarrow sh(P_1, \neg P_2, \neg P_3)$

Hierbei ist $A_{P=0}$ die Formel, die aus A entsteht, indem jedes Vorkommen der Aussagenvariablen P durch die Konstante 0 ersetzt wird. Analog ist $A_{P=1}$ zu verstehen.

Beweis Offensichtliche Gleichungen bis auf Punkt 6. Siehe dazu Übungsaufgabe 2.4.14.

Definition 2.33 (Normierte sh -Formeln)

Wir fixieren eine Ordnung auf der Menge der Aussagevariablen, etwa die durch die Ordnung der Indizes gegebene. Eine sh -Formel A heißt eine *normierte sh -Formel* (wenn wir genauer sein wollen, sagen wir eine bzgl. der fixierten Ordnung der Aussagevariablen normierte sh -Formel), wenn sie der folgenden rekursiven Definition genügt.

1. Die Konstanten $0, 1$ sind normierte sh -Formeln.
2. Sei P_i eine Aussagevariable, seien A, B normierte sh -Formeln, so daß keine Aussagevariable P_j mit $i \geq j$ in A oder B auftritt. Dann ist $sh(P_i, A, B)$ eine normierte sh -Formel.

Die Menge der normierten sh -Formeln ist die kleinste Menge, die bezüglich 1. und 2. abgeschlossen ist.

Lemma 2.34

Zu jeder aussagenlogischen Formel A gibt es eine äquivalente normierte sh -Formel B .

Beweis: Induktion nach der Anzahl n der in A vorkommenden Aussagevariablen.

Für $n = 0$ kann A logisch äquivalent auf eine der Konstanten 0 oder 1 reduziert werden. Konstanten sind normierte sh -Formeln.

Im Induktionsschritt wählen wir die in A vorkommende Aussagevariable P_i mit dem kleinsten Index. Mit A_0 bezeichnen wir die Formel, die aus A entsteht, indem jedes Vorkommen von P_i durch 0 ersetzt wird. Entsprechend wird A_1 gebildet. Nach Induktionsvoraussetzung gibt es normierte sh -Formeln B_0, B_1 , die logisch äquivalent (siehe Def. 2.16) sind zu A_0, A_1 . Offensichtlich ist A äquivalent zu $sh(P_i, B_0, B_1)$, und $sh(P_i, B_0, B_1)$ ist eine normierte sh -Formel. ■

Definition 2.35

Ein sh -Graph ist ein gerichteter, binärer, zusammenhängender Graph.

- Jedem nichtterminalen Knoten v ist eine natürliche Zahl $index(v)$ zugeordnet.
- Von jedem nichtterminalen Knoten v gehen zwei Kanten aus. Eine davon ist mit 0, die andere mit 1 gekennzeichnet.
- Jedem terminalen Knoten v ist eine der Zahlen 0 oder 1 zugeordnet, bezeichnet mit $wert(v)$.
- Ist der nichtterminale Knoten w ein unmittelbarer Nachfolger von v , dann gilt $index(v) < index(w)$.
- Es gibt genau einen Wurzelknoten.

Wegen der vierten Forderung ist jeder sh -Graph zyklensfrei.

Bemerkung Es besteht eine offensichtliche Zuordnung zwischen normierten sh -Formeln und sh -Graphen.

Jedem sh -Graphen G kann man eine m -stellige Boole'sche Funktion f_G zuordnen, wobei m die Anzahl der in G vorkommenden verschiedenen Indizes i_1, \dots, i_m ist. Wir fassen f_G als eine Funktion mit den Eingabevariablen P_{i_1}, \dots, P_{i_m} auf und bestimmen den Funktionswert $f_G(P_{i_1}, \dots, P_{i_m})$, indem wir an der Wurzel von G beginnend einen Pfad durch G wählen. Am Knoten v folgen wir der Kante 0, wenn die Eingabevariable $P_{index(v)}$ den Wert 0 hat,

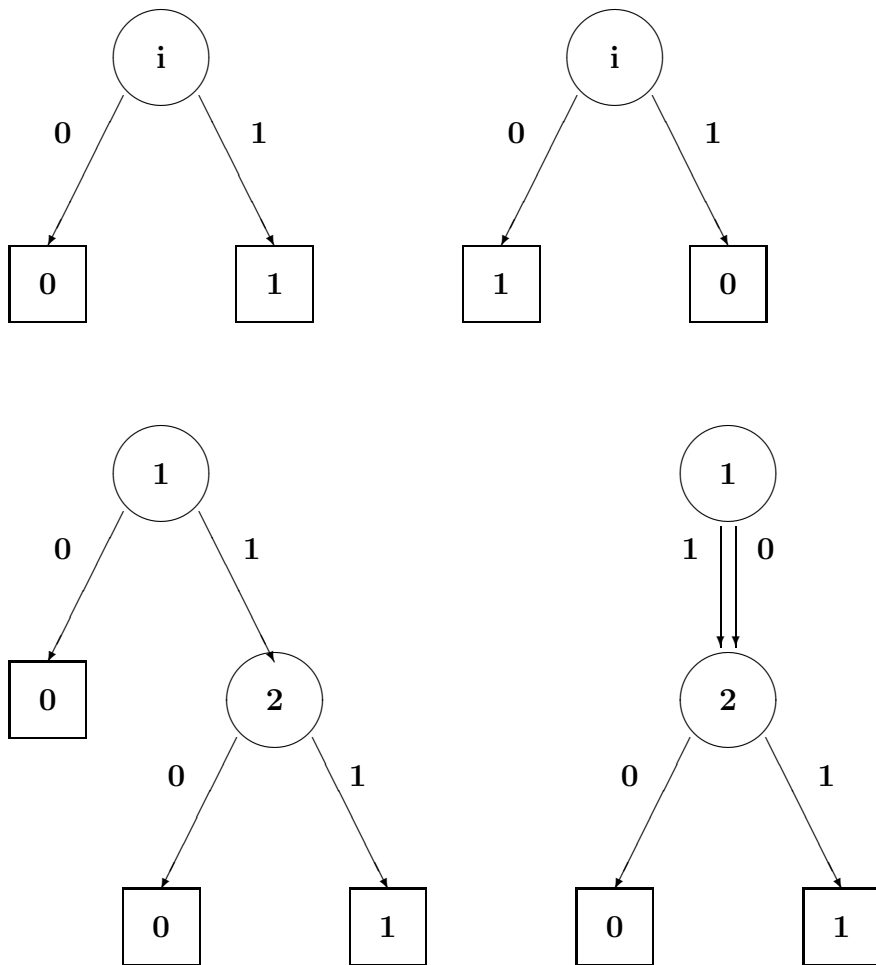


Abbildung 2.5: Beispiele von sh -Graphen

sonst der Kante 1. Der Wert des terminalen Knotens ist dann der gesuchte Funktionswert, siehe Abb. 2.4.4.

Das führt zu der folgenden induktiven Definition:

Definition 2.36 (f_G)

Sei G ein sh -Graph, i_1, \dots, i_m seien die in G auftretenden Indizes mit dem kleinsten beginnend der Größe nach geordnet.

Die von G induzierte Boole'sche Funktion $f_G : \{0, 1\}^m \rightarrow \{0, 1\}$ ist definiert durch:

1. $f_G(b_1, \dots, b_m) = 0$ falls G nur aus dem einzigen Knoten v besteht, mit $wert(v) = 0$,

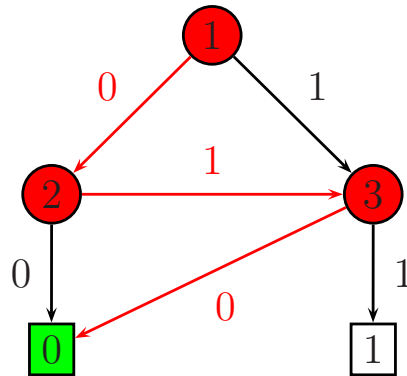


Abbildung 2.6: Berechnung der von G induzierten Funktion $f_G(0, 1, 0) = 0$

2. $f_G(b_1, \dots, b_m) = 1$ falls G nur aus dem einzigen Knoten v besteht, mit $wert(v) = 1$,
3. $f_G(b_1, \dots, b_m) = \begin{cases} f_{G_0}(b_2, \dots, b_m) & \text{falls } b_1 = 0 \\ f_{G_1}(b_2, \dots, b_m) & \text{falls } b_1 = 1 \end{cases}$

Hierbei ist G_i der Teilgraph von G , der mit dem Knoten beginnt, der von der Wurzel aus entlang der mit i markierten Kante erreicht wird.

Lemma 2.37

Zu jeder Boole'schen Funktion $f : \{0, 1\}^m \rightarrow \{0, 1\}$ und jeder aufsteigenden Folge $i_1 < \dots < i_m$ von Indizes gibt es einen *sh*-Graphen G mit

$$f_G = f.$$

Beweis: Der Beweis geschieht durch Induktion über m .

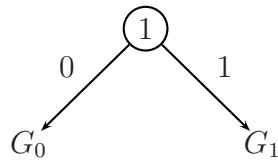
Ist $m = 0$ und f die konstante Funktion mit Wert 0, dann besteht G nur aus einem Knoten v , der dann notwendigerweise ein terminaler Knoten ist, mit $wert(v) = 0$.

Ist $m = 0$ und f die konstante Funktion mit Wert 1, dann besteht G nur aus einem Knoten v mit $wert(v) = 1$.

Sei jetzt $m > 0$. Wir definieren die $m - 1$ stelligen Funktionen f_0 und f_1 durch

$$f_i(b_2, \dots, b_m) = \begin{cases} f(0, b_2, \dots, b_m) & \text{falls } i = 0 \\ f(1, b_2, \dots, b_m) & \text{falls } i = 1 \end{cases}$$

Nach Induktionsvoraussetzung gibt es *sh*-Graphen G_i mit $f_{G_i} = f_i$ für $i = 0, 1$. Sei G der *sh*-Graph



dann gilt offensichtlich $f_G = f$.

■

Definition 2.38

Ein *sh*-Graph heißt *reduziert*, wenn

1. es keine zwei Knoten v und w ($v \neq w$) gibt, so daß der in v verwurzelte Teilgraph G_v mit dem in w verwurzelten Teilgraph G_w isomorph ist.
2. es keinen Knoten v gibt, so daß beide von v ausgehenden Kanten zum selben Nachfolgerknoten führen.

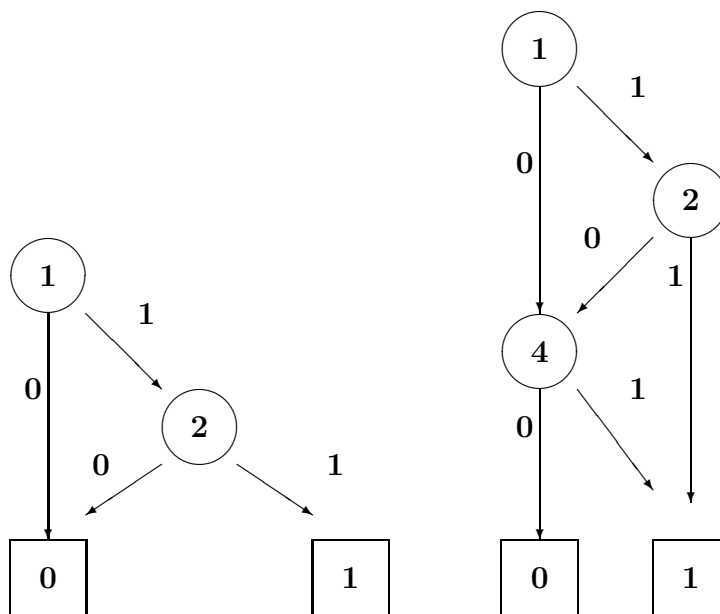


Abbildung 2.7: Beispiele reduzierter *sh*-Graphen

Ein reduzierter Shannongraph heißt auch BDD bzw. OBDD (*ordered binary decision diagram*).



Abbildung 2.8: Einfachstes Beispiel isomorpher Shannon-Graphen

Definition 2.39 (Isomorphie von Shannongraphen)

Seien zwei *sh*-Graphen H, G gegeben. Ihre Knotenmengen seien V_1, V_2 . H, G heißen zueinander *isomorph* ($H \cong G$) genau dann, wenn es eine bijektive Abbildung π von V_1 nach V_2 gibt mit:

1. $index(k) = index(\pi(k))$ für jeden Nichtterminalknoten $k \in V_1$
2. $wert(k) = wert(\pi(k))$ für jeden Terminalknoten $k \in V_1$
3. Für jeden Nichtterminalknoten $k \in V_1$, dessen 0-Kante/1-Kante zu dem Knoten k_0/k_1 führt, gilt: die 0-Kante von $\pi(k)$ führt zu $\pi(k_0)$, die 1-Kante zu $\pi(k_1)$.

Im simpelsten Fall besagt die Definition 2.39, daß es auf die Anordnung der Kanten des Graph G in der Zeichenebene nicht ankommt. Ein typisches Beispiel isomorpher Graphen zeigt Abb. 2.8.

Ein etwas komplexeres Beispiel isomorpher Teilgraphen zeigt Abbildung 2.9. Die beiden Teilgraphen unterhalb und einschließlich der beiden Knoten mit Index 1, nennen wir sie für den Augenblick $N11$ und $N12$, sind isomorph. Die Knoten an den 0-Kanten von $N11$ und $N12$ sind allerdings nicht identisch. Ebenso wenig die Knoten an den 1-Kanten. Verfolgt man den Isomorphismus weiter nach unten, stößt man allerdings wieder auf diese Situation. Der Isomorphismus bildet den am weitesten links stehenden Knoten mit dem Index 2, $N21$ ab auf den dritten Knoten in derselben Reihe mit dem Index 2, $N23$. Die 0-Nachfolger von $N21$ sind $N23$ sind identisch, nämlich gleich dem Blattknoten mit Wert 0. Das liefert die Beweisidee für Lemma 2.40. Die Abbildung 2.10 zeigt die schrittweise Reduktion des Graphen aus Abbildung 2.9.

Lemma 2.40

Sei G ein Shannongraph, so daß für jedes Paar von Knoten v, w gilt

- wenn die 1-Nachfolger von v und w gleich sind und
 die 0-Nachfolger von v und w gleich sind
dann $v = w$

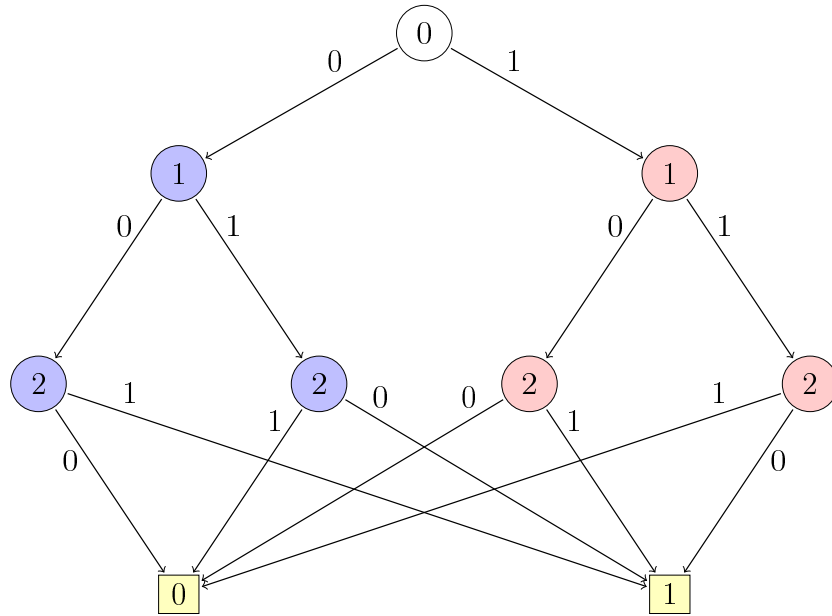


Abbildung 2.9: Beispiel isomorphe Shannon-Graphen

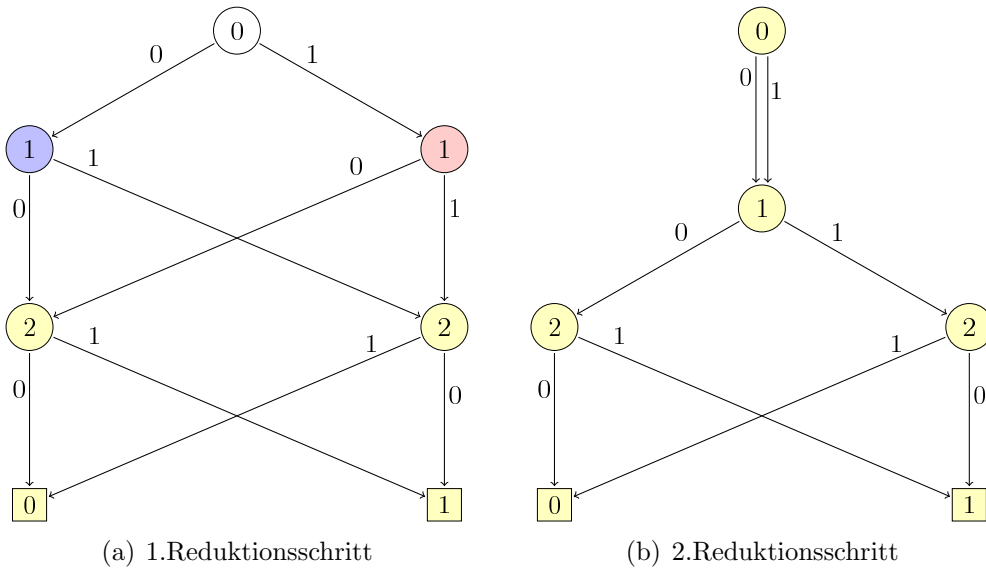


Abbildung 2.10: Reduktion des Shannongraphen aus Abb.2.9

Dann erfüllt G die Bedingung (1) aus Definition 2.38.

Beweis: Seien x, y zwei Knoten und π ein Isomorphismus von dem Teilgraphen G_x unterhalb von x auf den Teilgraphen G_y unterhalb von y . Gilt für alle Knoten v in G_x schon $\pi(v) = v$, dann auch $x = y$ und wir sind fertig. Anderenfalls wählen wir einen kleinsten Knoten v (d.h. mit möglichst kurzer Entfernung von einem Blattknoten) mit $\pi(v) \neq v$. Da für Blattknoten B nach Definition eines Isomorphismus $\pi(B) = B$ gelten muß, ist v ein Nichtterminalknoten. Sei v_1 der 1-Nachfolger und v_0 der 0-Nachfolger von v . Wegen der speziellen Wahl von v gilt einerseits $\pi(v) \neq v$, andererseits $\pi(v_0) = v_0$ und $\pi(v_1) = v_1$. Wegen der Isomorphieeigenschaft von π gilt ist $\pi(v_1)$ der 1-Nachfolger von $\pi(v)$ und $\pi(v_0)$ der 0-Nachfolger von $\pi(v)$. Wir haben somit zwei Knoten v und $\pi(v)$ gefunden, deren beide Nachfolger übereinstimmen. Nach Voraussetzung über G muß v und $\pi(v)$ gelten. Dieser Widerspruch zeigt daß für alle v gelten muß $\pi(v) = v$. ■

Lemma 2.41

Sind G, H reduzierte sh -Graphen zu $\Sigma = \{P_1, \dots, P_n\}$, dann gilt

$$f_G = f_H \Leftrightarrow G \cong H.$$

(Zu jeder Boole'schen Funktion f gibt es bis auf Isomorphismus genau einen reduzierten sh -Graphen H mit $f = f_H$).

Beweis

Die Idee des Existenzbeweises eines Äquivalenten reduzierten sh -Graphen ist einfach: Man geht von einem entsprechenden sh -Graphen aus und beseitigt nach und nach alle Situationen, die der Reduziertheit entgegenstehen. Bei jedem solchen Schritt wird die Anzahl der Knoten um mindestens eins kleiner, so daß die Terminierung gesichert ist.

Beweis der Eindeutigkeit: Seien H und G zwei reduzierte sh -Graphen mit $f_H = f_G = f$. Wir müssen $H \cong G$ zeigen. Das geschieht durch Induktion über $k =$ die Länge des längsten Pfades in G und H .

$k = 0$:

G und H bestehen nur aus einem Blatt, das zugleich Wurzel ist; d.h., G und H haben die Form $\boxed{0}$ oder $\boxed{1}$. f_H und f_G sind 0-stellige Funktionen, d.h. Konstanten und aus $f_H = f_G$ folgt, daß $H = G = \boxed{0}$ oder $H = G = \boxed{1}$ gelten muß,

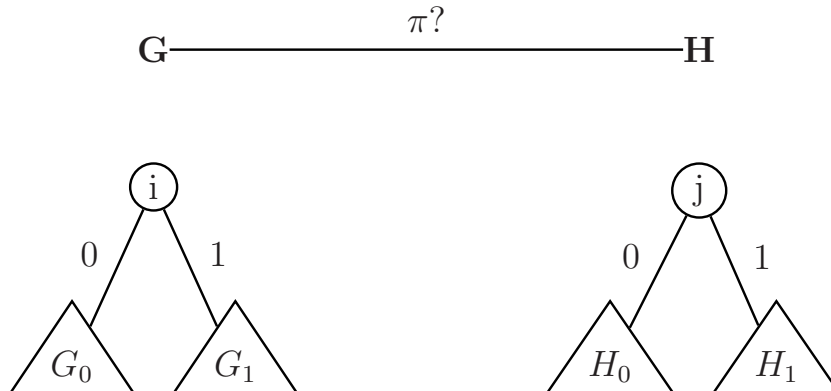


Abbildung 2.11: Konstruktion eines Isomorphismus zwischen G und h

$k > 0$:

Die Äquivalenz gelte schon für alle Paare G', H' mit maximaler Pfadlänge $< k$:

Es sei

$$\begin{aligned} v_0 & \text{ Wurzel von } G, & \text{index}(v_0) &= i \\ w_0 & \text{ Wurzel von } H, & \text{index}(w_0) &= j \end{aligned}$$

Siehe Abbildung 2.11

Wir zeigen i ist der kleinste Index, so daß f_G von P_i abhängt.

Andernfalls wäre $f_{P_i=0} = f_{P_i=1}$. Sind G_0, G_1 die Teilgraphen von G , die über die 0-Kante bzw. die 1-Kante direkt unter v_0 hängen, so gilt $f_{G_0} = f_{P_i=0}, f_{G_1} = f_{P_i=1}$; da die Länge des längsten Pfades in G_0 und $G_1 < k$ ist, folgt nach Induktionsannahme $G_0 \cong G_1$, entgegen der Reduziertheit von G . Da für alle weiteren Knoten v in G gilt $i < \text{index}(v)$ muß i der kleinste Index sein, so daß f_G von P_i abhängt. Genauso zeigt man, daß $\text{index}(w_0) = j$ der kleinste Index ist, so daß f_H von P_j abhängt. Aus der Voraussetzung $f_G = f_H$ folgt somit $i = j$.

Man betrachte nun G_0, G_1, H_0, H_1 , die Teilgraphen aus G bzw. H , die an den Knoten v_0 bzw. w_0 an den 0-Kanten bzw. 1-Kanten hängen. Aus $f_G = f_H = f$ folgt $f_{G_0} = f_{P_i=0} = f_{H_0}$ und $f_{G_1} = f_{P_i=1} = f_{H_1}$. Nach Induktionsannahme gilt also $G_0 \cong H_0$ und $G_1 \cong H_1$.

Seien π_0, π_1 Isomorphismen $\pi_0 : G_0 \rightarrow H_0, \pi_1 : G_1 \rightarrow H_1$. Es gilt:

(*) Wenn v Knoten in G_0 und G_1 ist, dann $\pi_0(v) = \pi_1(v)$.

Angenommen, das wäre nicht wahr. Dann wählen wir ein v in $G_0 \cap G_1$, so daß $\pi_0(v) \neq \pi_1(v)$ gilt. Die in $\pi_0(v)$ und $\pi_1(v)$ verwurzelten Teilgraphen von H definieren dieselbe Funktion, wären also nach Induktionsannahme isomorph. Das widerspricht der Reduziertheit von H .

Definiere $\pi : G \rightarrow H$ durch

$$\pi(v) = \begin{cases} w_0 & \text{falls } v = v_0 \\ \pi_0(v) & \text{falls } v \text{ in } G_0 \\ \pi_1(v) & \text{falls } v \text{ in } G_1 \end{cases}$$

π ist injektiv: Begründung wie für (*).

π ist surjektiv: klar.

π erhält Kanten und Marken: klar.

Also ist π ein Isomorphismus und damit $G \cong H$.

■

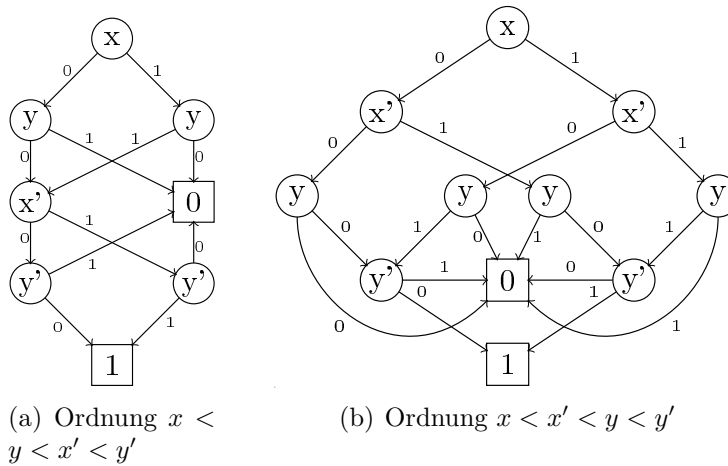


Abbildung 2.12: Shannon Graphen für $(x \leftrightarrow y) \wedge (x' \leftrightarrow y')$

Die Größe eines Shannongraphen für eine vorgegebene Boole'sche Funktion hängt manchmal entscheidend von der Variablenordnung ab, wie man in Abbildung 2.12 sehen kann. Heuristiken für eine geschickte Wahl der Variablenordnung sind daher in vielen Anwendungen OBDDs von großer Wichtigkeit. Allerdings gibt es auch hart Nüsse, die bezüglich jeder Variablenordnung große Shannongraphen liefern.

Beispiel 2.42

X sei die Menge der $2k$ Variablen $\{x_0, \dots, x_{k-1}, y_0, \dots, y_{k-1}\}$. $x = x_0 \dots x_{k-1}$ und $y = y_0 \dots y_{k-1}$ bezeichnen k -stellige Binrzahlen. Für $0 \leq i < 2k$ bezeichne $Mult_i$ die boolesche Funktion, die das i -te Bit des Produktes von x mit y beschreibt.

Lemma 2.43

Für jede Ordnung $<$ der Variablen in X gibt es einen Index $0 \leq i < 2k$, so dass der BDD $B_{Mult_i, <}$ mindestens $2^{k/8}$ Knoten besitzt.

2.4.5 Übungsaufgaben

Übungsaufgabe 2.4.1

Berechnen Sie die konjunktive und disjunktive Normalform der Formel $A \leftrightarrow (B \leftrightarrow C)$.

Übungsaufgabe 2.4.2

Geben Sie eine Familie F_n von Formeln mit n Atomen an, so daß die konjunktive Normalform exponentiell zu n wächst.

Übungsaufgabe 2.4.3

Betrachten Sie die DNF

$$A = (\neg P \wedge Q \wedge \neg R) \vee (P \wedge Q \wedge \neg R) \vee (P \wedge Q \wedge R) \vee (\neg P \wedge \neg Q \wedge R) \vee (P \wedge \neg Q \wedge R)$$

Zeigen Sie, daß die Normalformen

$$\begin{aligned} A' &= (\neg Q \wedge R) \vee (Q \wedge \neg R) \vee (P \wedge R) \\ A'' &= (\neg Q \wedge R) \vee (Q \wedge \neg R) \vee (P \wedge Q) \end{aligned}$$

äquivalent zu A sind.

Übungsaufgabe 2.4.4

Der Begriff *Primimplikant* wurde in Definition 2.24 eingeführt.

Zeigen Sie: alle Disjunktionsglieder von

$$A_1 = (\neg Q \wedge R) \vee (Q \wedge \neg R) \vee (P \wedge R) \vee (P \wedge Q)$$

sind Primimplikanten der Formel A aus Aufgabe 2.4.3.

Übungsaufgabe 2.4.5

Zeigen Sie: Ist A eine DNF für B und ist A von minimaler (Zeichen-)Länge, dann sind alle Disjunktionsglieder von A Primimplikanten von B .

Übungsaufgabe 2.4.6

In Lemma 2.22 wurde das Craigsche Interpolationslemma formuliert und bewiesen.

Sei jetzt eine aussagenlogische Formeln A in disjunktiver Normalform $A = \bigvee_i \bigwedge_j A_{ij}$ und eine Formel B in konjunktive Normalform $B = \bigwedge_r \bigvee_s B_{rs}$ gegeben, so daß

$$A \rightarrow B$$

eine Tautologie ist. Finden Sie ein Verfahren um eine Formel C mit den Eigenschaften:

1. $A \rightarrow C$ und $C \rightarrow B$ sind Tautologien
2. In C kommen nur Atome vor, die sowohl in A als auch in B vorkommen.

direkt aus den Normalformen von A und B abzulesen.

W. Craig hat die in Lemma 2.22 als erster entdeckt und bewiesen, sogar für den prädikatenlogischen Fall. Der Beweis der hier genannten aussagenlogischen Variante ist wesentlich einfacher.

Übungsaufgabe 2.4.7

Für jedes n sei

$$C_n := \neg \left(\overbrace{P_1 \leftrightarrow (P_2 \leftrightarrow (P_3 \leftrightarrow \dots (P_n \leftrightarrow \neg P_1) \dots))}^{a_1} \right)_{a_2}$$

1. Ist C_n für jedes n eine Tautologie ?
2. Bringen Sie $\neg C_n$ in KNF.
3. Geben Sie $(C_n)_{kknf}$ an.

Übungsaufgabe 2.4.8

Kann es eine kurze DNF geben ?

Übungsaufgabe 2.4.9

Sei n die Anzahl der Literalvorkommen in C .

Geben Sie in Abhängigkeit von n eine obere Schranke für die Anzahl der Literalvorkommen in C_{kknf} an.

Übungsaufgabe 2.4.10

Finden Sie äquivalente normierte *sh*-Formeln zu:

1. P_i
2. $P_1 \wedge P_2$
3. $\neg P_i$
4. $P_1 \vee P_2$

Übungsaufgabe 2.4.11

1. Finde den reduzierten *sh*-Graphen für die von der Formel $(P_1 \wedge P_2) \vee P_4$ erzeugte Funktion.

2. Finde den reduzierten *sh*-Graphen für die folgende Funktion

$$f_0p(P_1, P_2, P_3, P_4) = \begin{cases} 1 & \text{falls die Summe } P_1 + \dots + P_4 \text{ ungerade ist} \\ 0 & \text{sonst} \end{cases}$$

Sei \circ irgendeine zweistellige Boole'sche Operation. Eine wichtige Aufgabe bei der Manipulation von *sh*-Graphen besteht darin, aus Graphen G_1 und G_2 für die Funktionen f_1 und f_2 einen *sh*-Graphen für die Funktion $f_1 \circ f_2$ zu konstruieren. Die folgende Aufgabe stellt die Basis für diese Konstruktion dar, effiziente Algorithmen findet man z. B. wieder in [Bry86].

Übungsaufgabe 2.4.12

Sei \circ ein beliebiger binärer aussagenlogischer Operator. Zeigen Sie

1. $sh(P_i, A, B) \circ sh(P_i, C, D) \leftrightarrow sh(P_i, A \circ C, B \circ D)$
2. $sh(P_i, A, B) \circ sh(P_j, C, D) \leftrightarrow sh(P_i, A \circ sh(P_j, C, D), B \circ sh(P_j, C, D))$

Dabei sind $sh(P_i, A, B)$, $sh(P_j, C, D)$, $sh(P_i, C, D)$ normierte *sh*-Formeln und $i < j$.

Übungsaufgabe 2.4.13

Sei A eine normierte *sh*-Formel, B entstehe aus A durch Vertauschung der Konstanten 0 und 1. Zeigen Sie $\neg A \leftrightarrow B$.

Übungsaufgabe 2.4.14

Zeigen Sie:

$$sh(sh(P_1, P_2, P_3), P_4, P_5) \equiv sh(P_1, sh(P_2, P_4, P_5), sh(P_3, P_4, P_5))$$

Übungsaufgabe 2.4.15

Berechnen Sie eine kurze konjunktive Normalform für die Formel

$$A = ((A_1 \wedge A_2) \vee (A_1 \wedge A_3)) \vee \neg(A_2 \wedge A_3)$$

Übungsaufgabe 2.4.16

Sei S eine Menge von Klauseln. Für ein Literal L sei \bar{L} das zu L komplementäre Literal, d.h.

$$\bar{L} = \begin{cases} \neg A & \text{falls } L = A \\ A & \text{falls } L = \neg A \end{cases}$$

Ein Literal L heißt *isoliert* in S (im Englischen *pure literal in S*), wenn \bar{L} in keiner Klausel in S vorkommt. Eine Klausel C in S heißt *isoliert*, wenn sie ein isoliertes Literal enthält.

Ist S eine unerfüllbare Klauselmenge und C eine isolierte Klausel in S , dann ist auch $S \setminus \{C\}$ unerfüllbar.

Übungsaufgabe 2.4.17

Definition 2.44

Eine unerfüllbare Klauselmengemenge S heißt eine *minimal unerfüllbare* Klauselmengemenge, wenn jede echte Teilmengemenge $S_0 \subset S$ erfüllbar ist.

Sei S eine minimal unerfüllbare Klauselmengemenge dann gilt

1. S enthält keine tautologische Klausel, d.h. keine Klausel C , die für eine aussagenlogische Variable P sowohl P selbst als auch $\neg P$ enthält.
2. für jede Interpretation I gibt es Klauseln $C, D \in S$ und ein Literal L , so daß $I(C) = W$, $I(D) = F$, L kommt in C und \bar{L} kommt in D vor.

2.5 Erfüllbarkeit in speziellen Formelklassen

Wir erinnern uns an das in 2.3.2 zu SAT Gesagte, das Problem der Entscheidung der Erfüllbarkeit einer aussagenlogischen Formel. Wie verhält sich SAT, wenn für die vorgelegte Formel eine bestimmte Normalform vorausgesetzt wird?

Faktum

Für Formeln in DNF ist die Erfüllbarkeit in linearer Zeit, d.h. in $O(n)$, entscheidbar, wobei n die Länge der Eingabeformel ist.

Beweis: Übung.

Satz 2.45

(wird hier nicht bewiesen) Für Formeln in KNF ist das Erfüllbarkeitsproblem bereits NP-vollständig. Das gilt sogar für Formeln in sogenannter 3-KNF, in der jede der konjunktiv verknüpften Disjunktionen höchstens 3 Literale enthält.

Definition 2.46

Eine *Krom-Formel* ist eine aussagenlogische Formel in KNF, in der jede Disjunktion höchstens zwei Literale enthält.

Satz 2.47

Für Krom-Formeln ist die Erfüllbarkeit in polynomialer Zeit entscheidbar.

Beweis

(wird später bewiesen)

2.5.1 Horn-Formeln

Definition 2.48

Eine *Horn-Formel* ist eine aussagenlogische Formel in KNF, in der jede Disjunktion höchstens ein positives Literale enthält. Eine solche Disjunktion heißt eine *Horn-Klausel*.

Beispiel 2.49

$\neg P \wedge (Q \vee \neg R \vee \neg S) \wedge (\neg Q \vee \neg S) \wedge R \wedge S \wedge (\neg Q \vee P)$ ist eine Horn-Formel.

Eine Horn-Klauseln schreibt man häufig logisch äquivalent als Implikation:

$\neg B_1 \vee \dots \vee \neg B_m \vee A$	$B_1 \wedge \dots \wedge B_m \rightarrow A$
$\neg B_1 \vee \dots \vee \neg B_m$	$B_1 \wedge \dots \wedge B_m \rightarrow \mathbf{0}$
A	A

Dabei heißt $B_1 \wedge \dots \wedge B_m$ der *Rumpf* und A der *Kopf* der Horn-Klausel $B_1 \wedge \dots \wedge B_m \rightarrow A$

Unser Beispiel lautet in dieser Schreibweise:

$$(P \rightarrow 0) \wedge (R \wedge S \rightarrow Q) \wedge (Q \wedge S \rightarrow 0) \wedge R \wedge S \wedge (Q \rightarrow P)$$

Satz 2.50

Für Horn-Formeln ist die Erfüllbarkeit in quadratischer Zeit entscheidbar.

Beweis Wir geben einen quadratischen Algorithmus an, siehe Abbildung 2.13.

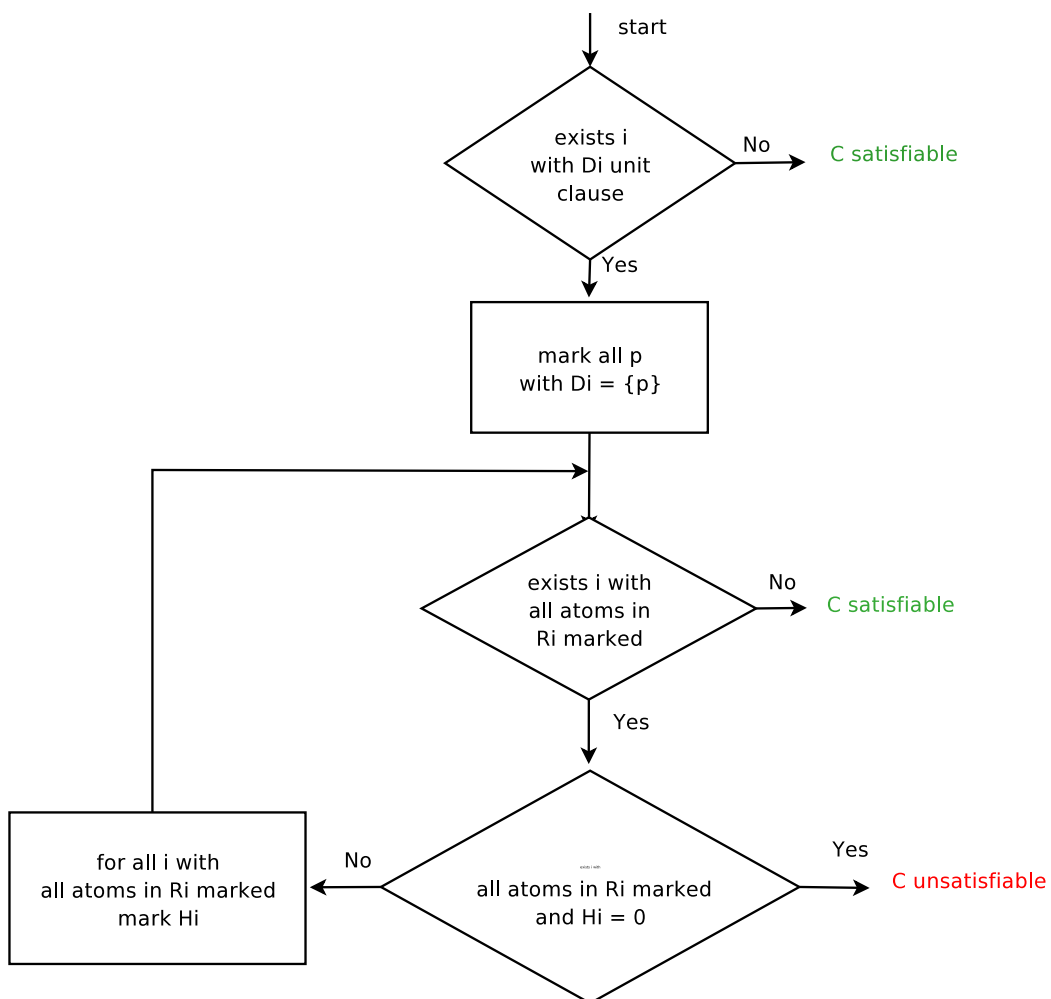


Abbildung 2.13: Erfüllbarkeitsalgorithmus für Hornklauseln

Die Idee ist, sukzessive diejenigen Atome zu *markieren*, welche bei einer (vielleicht existierenden) erfüllenden Belegung den Wert W erhalten müßten. Man

erhält dann tatsächlich eine erfüllende Belegung oder stößt beim Versuch, eine solche zu finden, auf einen Widerspruch.

Sei $C = D_1 \wedge \dots \wedge D_m$ eine Hornformel. Daß wir ein Atom in C *markieren*, bedeutet, daß wir es an allen Stellen seines Auftretens in C markieren.

Schritt 0:

Markiere alle Fakten (Horn-Klauseln mit leerem Rumpf). Wenn keine vorhanden sind: gib aus „erfüllbar“ und halte an.

Schritt 1:

Suche nach einem $D_i = R_i \rightarrow K_i$ in C , so daß alle Atome im Rumpf markiert sind aber K_i noch nicht. Falls keines existiert, gebe aus „erfüllbar“ und halte an.

Andernfalls sei $R_j \rightarrow K_j$ das erste solche D_i .

- falls $K_j \neq \mathbf{0}$: markiere K_j überall in C und wiederhole Schritt 1.
- falls $K_j = \mathbf{0}$: gebe aus „unerfüllbar“ und halte an.

Gemessen an der Länge der Eingabeformel C haben beide Schritte lineare Laufzeit. Da höchstens so viele Schritte zu machen sind, wie es Atome in C gibt, hat der Algorithmus einen quadratischen Zeitaufwand.

Korrektheitsbeweis

Die Terminierung ist, wie erwähnt, stets gewährleistet. Zu zeigen ist noch, daß

a) wenn der Algorithmus mit „erfüllbar“ endet, er auch eine erfüllende Belegung gefunden hat und

b) wenn C erfüllbar ist, der Algorithmus mit „erfüllbar“ abbricht.

a) Angenommen der Algorithmus endet mit der Ausgabe „erfüllbar“. Wir definieren I durch:

$$I(P) = W \quad \Leftrightarrow \quad P \text{ wurde im Lauf des Algorithmus markiert.}$$

Wir zeigen:

$val_I(D_i) = W$ für alle $i = 1, \dots, m$, woraus $val_I(C) = val_I(D_1 \wedge \dots \wedge D_m) = W$ folgt.

Falls D_i ein Atom ist, so wurde D_i in Schritt 0 markiert und $val_I(D_i) = W$ gilt. Falls $D_i = R_i \rightarrow \mathbf{0}$, dann wurden nicht alle Atome in R_i markiert, weil sonst die Ausgabe „unerfüllbar“ erfolgt wäre. Somit gilt $val_I(A) = F$ für

ein in R_i vorkommendes Atom und damit $val_I(R_i \rightarrow \mathbf{0}) = W$. Bleibt $D_i = R_i \rightarrow K_i$. Falls ein Atom in R_i nicht markiert wurde, so gilt $val_I(R_i) = \mathbf{F}$ und damit $val_I(R_i \rightarrow K_i) = \mathbf{W}$. Falls alle Atome in R_i markiert wurden, so wurde auch K_i markiert, und es gilt ebenfalls $val_I(R_i \rightarrow K_i) = \mathbf{W}$.

b) Angenommen C ist erfüllbar und I eine Interpretation mit $val_I(C) = \mathbf{W}$. Wir zeigen, daß für jedes Atom A , das im Laufe des Algorithmus markiert wird, $val_I(A) = \mathbf{W}$ gilt. Das ist offensichtlich wahr für die Markierungen im Schritt 0. Angenommen das Atom A wird im $(t + 1)$ -ten Durchlauf durch Schritt 1 markiert. Dann gibt es ein $D_i = R_i \rightarrow A$ in C , so daß im t -ten Durchlauf alle Atome von R_i markiert waren. Nach Induktionsvoraussetzung wissen wir, daß $val_I(R_i) = \mathbf{W}$ gilt. Da außerdem nach Annahme $val_I(D_i) = \mathbf{W}$ sein soll, gilt auch $val_I(A) = \mathbf{W}$.

Daraus folgt, daß der Algorithmus nicht mit „unerfüllbar“ abbrechen kann, da nicht alle Atome in einer Horn-Klausel der Form $R_i \rightarrow \mathbf{0}$ markiert werden können, denn dann wäre I keine C erfüllende Belegung. ■

2.5.2 Äquivalenzformeln

Wir betrachten das Fragment $\ddot{A}qFor$ aussagenlogischer Formeln, das einzig aus

$$\leftrightarrow, \mathbf{1}, \mathbf{0}$$

und aussagenlogischen Variablen aufgebaute Formeln enthält.

Satz 2.51

Eine Formel A aus $\ddot{A}qFor$ ist eine Tautologie gdw.

jede Aussagenvariable hat eine gerade Anzahl von Vorkommen in A und die Konstante $\mathbf{0}$ hat eine gerade Anzahl von Vorkommen in A .

Nach diesem Satz wäre z. B.

$$\left((P \leftrightarrow \mathbf{1}) \leftrightarrow (\mathbf{0} \leftrightarrow Q) \right) \leftrightarrow \left((P \leftrightarrow Q) \leftrightarrow \mathbf{0} \right)$$

eine Tautologie,

$$\left((P \leftrightarrow \mathbf{0}) \leftrightarrow P \right) \leftrightarrow \left(Q \leftrightarrow (\mathbf{0} \leftrightarrow P) \leftrightarrow P \right)$$

aber nicht.

Man beachte, daß über die Häufigkeit des Vorkommens der Konstanten $\mathbf{1}$ nichts verlangt wird.

Beweis: Wir beginnen mit einem schnellen Beweis. Sei A eine Äquivalenzformel, in der jede aussagenlogische Variable und die Konstante $\mathbf{0}$ in gerader Anzahl vorkommen. Wir zeigen, daß A eine Tautologie ist.

1. Unter Ausnutzung der Assoziativität

$$(A \leftrightarrow (B \leftrightarrow C)) \leftrightarrow ((A \leftrightarrow B) \leftrightarrow C)$$

des Operators \leftrightarrow können wir in A alle Klammern weglassen.

2. durch wiederholte Anwendung des Kommutativität den \leftrightarrow bringen wir all gleichen Variablen und Konstanten unmittelbar nebeneinander.
3. indem wir $(X \leftrightarrow X) \leftrightarrow \mathbf{1}$ mehrfach ausnutzen eliminieren wir paarweise Atome und Vorkommen von $\mathbf{0}$.
4. nach Voraussetzung über A bleibt eine Äquivalenzformel übrig, in der nur noch die Konstante $\mathbf{1}$ vorkommt. Diese ist äquivalent zu $\mathbf{1}$.

In dem ersten Beispiel, sieht das so aus:

1. $((P \leftrightarrow \mathbf{1}) \leftrightarrow (\mathbf{0} \leftrightarrow Q)) \leftrightarrow ((P \leftrightarrow Q) \leftrightarrow \mathbf{0})$
2. $P \leftrightarrow \mathbf{1} \leftrightarrow \mathbf{0} \leftrightarrow Q \leftrightarrow P \leftrightarrow Q \leftrightarrow \mathbf{0}$
3. $\mathbf{1} \leftrightarrow P \leftrightarrow P \leftrightarrow Q \leftrightarrow Q \leftrightarrow \mathbf{0} \leftrightarrow \mathbf{0}$
4. $\mathbf{1}$

Das ist sehr überzeugend und für das mathematische Beweisen als ein sozialer Prozeß ausreichend. Ein strenger Beweis ist das nicht. Das Hauptproblem ist, daß in unserer Notation eine klammerfreie Schreibweise nicht vorgesehen ist. Wollte man den obigen Beweis einem maschinellen *proof checker* zur Kontrolle geben, würde der schon mit einem Syntaxfehler beim Parsen abbrechen.

Hier folgt eine mathematisch rigoroser Beweis.

Wir betrachten zunächst den Spezialfall, daß in A keine Aussagevariablen vorkommen. A ist also nur aus \leftrightarrow , $\mathbf{1}$ und $\mathbf{0}$ aufgebaut.

Wir zeigen:

1. $val(A) = \mathbf{W}$, wenn die Konstante $\mathbf{0}$ eine gerade Anzahl von Vorkommen in A hat

2. $val(A) = \mathbf{F}$ sonst

Das läßt sich durch Induktion über die Anzahl n der in A vorkommenden \leftrightarrow -Zeichen beweisen.

Im Fall $n = 0$ besteht A nur aus einer Konstanten, die im Teil 1 die Konstante $\mathbf{1}$ und im Teil 2 die Konstante $\mathbf{0}$ sein muß.

Im Fall $n > 0$ enthält A eine Teilformel der Form $c_1 \leftrightarrow c_2$, wobei c_1, c_2 Konstanten sind. Es sind drei Fälle möglich:

1. **Fall:** c_1 und c_2 sind beide $\mathbf{1}$
2. **Fall:** c_1 und c_2 sind beide $\mathbf{0}$
3. **Fall:** sonst.

A_0 sei die Formel, die aus A hervorgeht, indem wir $c_1 \leftrightarrow c_2$ ersetzen durch $\mathbf{1}$ in den beiden ersten Fällen und durch $\mathbf{0}$ im dritten Fall.

Offensichtlich gilt $val(A) = val(A_0)$, und die Anzahl der Vorkommen von $\mathbf{0}$ ist entweder unverändert geblieben oder hat sich um 2 verringert. Die gewünschte Behauptung für A folgt somit aus der Induktionshypothese für A_0 .

Wir kommen jetzt zum Beweis der allgemeinen Behauptung.

Zunächst nehmen wir an, daß jede Aussagenvariable und die Konstante $\mathbf{0}$ geradzahlig in A vorkommt. Formel A_I entstehe, indem man jede Variable P in A ersetzt durch $\mathbf{1}$ falls $I(P) = \mathbf{W}$ bzw. durch $\mathbf{0}$ falls $I(P) = \mathbf{F}$. A_I ist offensichtlich eine Formel vom Typ 1. Nach dem Resultat des Spezialfalls gilt also $val_I(A) = val(A_I) = \mathbf{W}$.

Kommt die Konstante \mathbf{F} ungeradzahlig vor, so belege I alle Aussagevariablen mit \mathbf{W} . Nach Teil 2 des Spezialfalls erhalten wir $val_I(A) = \mathbf{F}$ und A kann keine Tautologie sein.

Kommt die Konstante $\mathbf{0}$ zwar geradzahlig vor, aber eine Aussagevariable P ungeradzahlig, so sei I eine Belegung, die P mit \mathbf{F} und alle anderen Aussagevariablen durch \mathbf{W} belegt. Wieder gilt $val_I(A) = \mathbf{W}$ und A kann keine Tautologie sein.

■

2.5.3 Übungsaufgaben

Übungsaufgabe 2.5.1

Formulieren Sie ein Kriterium für die Erfüllbarkeit einer Formel in $\mathcal{A}qFor$.

Definition 2.52

Seien I, J zwei aussagenlogische Interpretationen. Wir sagen I ist kleiner als J , $I \leq J$, wenn für jede aussagenlogische Variable P mit $I(P) = W$ auch $J(P) = W$ gilt.

Übungsaufgabe 2.5.2

Zeigen Sie, daß es für jede erfüllbare Horn-Formel eine kleinste erfüllende Interpretation gibt.

Definition 2.53

Eine *definite Horn-Formel* ist eine aussagenlogische Formel in KNF, in der jede Disjunktion genau ein positives Literal enthält. Eine solche Disjunktion heißt eine *definite Horn-Klausel*.

$P_1 \wedge \dots \wedge P_k \rightarrow \mathbf{0}$ ist eine Horn-Formel, aber keine definite Horn-Formel.

Übungsaufgabe 2.5.3

Zeigen Sie, daß jede definite Horn-Formel erfüllbar ist.

Kapitel 3

Aussagenlogik: Beweistheorie

3.1 Einleitende Bemerkungen

Unter der *Beweistheorie* einer Logik versteht man die Bemühungen und Resultate zum Anliegen, die in der Logik festgelegte Folgerungsbeziehung \models *rein syntaktisch*, nämlich als *Ableitbarkeit in einem bestimmten Regelsystem* nachzumodellieren. Es ist ja \models unter Bezug auf Interpretationen der gegebenen Sprache definiert, und das in der „Formalität“ der Logik zu Ausdruck kommende Ziel besteht darin, auch unter Absehung von jedem solchen Bezug „schließen“ zu können. Falls und soweit das gelingt, ist „Schließen“, „Folgern“, „Beweisen“ im Prinzip automatisierbar. Zur Informatik, dem Bereich des algorithmischen Denkens, ergibt sich ein reiches Feld an Gemeinsamkeiten und Beziehungen, das die Entwicklung in den letzten zwanzig Jahren besonders nachhaltig geprägt hat.

Ein Regelsystem wie oben angesprochen heißt traditionellerweise ein *Kalkül*. Wir stellen vier solcher Kalküle vor

- einen vom „Hilbert-Typ“: weil sich theoretische Resultate hier besonders leicht erhalten lassen, und weil solche Kalküle die Wissenschaftsgeschichte nachhaltig geprägt haben;
- *Resolution* als den Kern des automatischen Beweisens;
- das *Tableauverfahren*, das in einer besonders einfachen und direkten Weise das semantikbezogene Folgern nachbildet;
- schließlich, als ein Vorläufer der Tableaumethode, einen *Sequenzkalkül*, als Beispiel eines reichhaltigen Kalküls, in dem sich beweistheoretische Untersuchungen gut unterbringen lassen.

Alle vier Kalküle werden wir später zu prädikatenlogischen ausbauen. Erst dann sind sie eigentlich interessant, doch wird das Kennzeichnende der Ansätze in unseren aussagenlogischen Vorversionen in besonders einfacher Weise deutlich.

3.1.1 Abstrakte Kalküle

Definition 3.1

Gegeben sei ein *syntaktischer Formalismus* oder eine *Sprache* (Δ, L) , d.h.: Es ist Δ ein (endliches oder unendliches, effektiv gegebenes) Alphabet und L eine entscheidbare Menge von endlichen Objekten über Δ , wie z.B.: Wörtern über dem Alphabet Δ , Listen aus Elementen aus Δ , endlichen Bäumen mit Knotenmarkierungen aus Δ^* etc.. ("Entscheidbar" bedeutet dann: entscheidbar in der Menge aller Objekte der gegebenen Art.) Hier wird Δ zunächst ein $For0_\Sigma$ sein, später wird es sich auch um andere, dem jeweiligen Konzept der "Beweisführung" entsprechende Datenobjekte handeln.

Für ein $n \in \mathbb{N}$ ist eine *n-stellige Regel* in (Δ, L) eine entscheidbare, $n + 1$ -stellige Relation über L . Ist $R \subseteq L^{n+1}$ eine solche Regel und ist $(u_1, \dots, u_n, u_{n+1}) \in R$, so heißen

- $(u_1, \dots, u_n, u_{n+1})$ eine *Instanz* von R
- u_1, \dots, u_n die *Prämissen* dieser Instanz
- u_{n+1} die *Conclusio* dieser Instanz.

Die Instanzen von nullstelligen Regeln heißen auch *Axiome*.

Ein *Kalkül* über der Sprache (Δ, L) ist eine endliche Menge von Regeln in dieser Sprache.

In unserem Fall werden die Regeln oft sehr *einfach* entscheidbare Relationen sein: Sie lassen sich angeben durch *Schemata*, so daß die sämtlichen Instanzen einer Regel entstehen, indem man in einem festen Kontext Variable von Objekten in L durch beliebige solche Objekte ersetzt (ggf. unter mit angegebenen Einschränkungen). In suggestiver Weise schreibt man ein solches Schema dann unter Verwendung eines horizontalen Strichs, welcher die Prämissen einer Instanz von ihrer Conclusio trennt.

Beispiel: Das Schema

$$\frac{A \vee B, B \rightarrow C, C \rightarrow A}{A}$$

steht für die dreistellige Regel

$$\{(A \vee B, B \rightarrow C, C \rightarrow A, A \mid A, B, C \in \text{For}0_\Sigma)\}.$$

3.1.2 Abstrakte Ableitbarkeit

Definition 3.2

In einer Sprache (Δ, L) sei Kal ein Kalkül, und es sei M eine Teilmenge von L . Eine *Ableitung* aus M in Kal ist eine Folge

$$(u_1, \dots, u_m)$$

von Wörtern in L , so daß für jedes $i \in \{1, \dots, m\}$ gilt:

- u_i ist Axiom; oder
- $u_i \in M$; oder
- es gibt eine Regel $R \in Kal$ einer Stelligkeit $n \geq 1$ sowie Indizes $j_1, \dots, j_n \in \{1, \dots, i-1\}$, so daß $(u_{j_1}, \dots, u_{j_n}, u_i) \in R$.

Im letzten Fall entsteht u_i durch Anwendung von R auf gewisse vorangegangene u_j .

Man beachte, daß für $i = 1$ gilt $\{1, \dots, i-1\} = \emptyset$, d. h. u_1 muß Axiom oder ein Element von M sein.

$u \in L$ heißt *ableitbar* aus M in Kal , kurz

$$M \vdash_{Kal} u$$

genau dann, wenn es eine Ableitung (u_1, \dots, u_m) in Kal gibt mit $u_m = u$.

Für $\emptyset \vdash_{Kal} u$ schreiben wir $\vdash_{Kal} u$, für $\{v\} \vdash_{Kal} u$ schreiben wir $v \vdash_{Kal} u$.

Lemma 3.3

(Kompositionen von Ableitungen) Es sei (u_1, \dots, u_n, v) eine Instanz einer Regel $R \in Kal$, und für die Mengen $M_1, \dots, M_n \subseteq L$ gelte

$$M_i \vdash_{Kal} u_i \text{ für } i = 1, \dots, n. \text{ Dann gilt } \bigcup_{i=1}^n M_i \vdash_{Kal} v$$

Beweis

Klar nach Definition von \vdash_{Kal} . ■

Lemma 3.4

(Kompaktheit der Ableitbarkeit) Es sei Kal ein Kalkül über (Δ, L) . Dann gilt für beliebige $M \subseteq L$, $u \in L$:

$$M \vdash_{Kal} u \iff \text{Es gibt eine endliche Teilmenge } E \text{ von } M \text{ mit } E \vdash_{Kal} u.$$

Beweis

\Rightarrow :

Sei (u_1, \dots, u_m) eine Ableitung von u aus M in Kal . Setzt man

$$E := M \cap \{u_1, \dots, u_m\},$$

so ist (u_1, \dots, u_m) auch Ableitung von u aus E . Also $E \vdash_{Kal} u$.

\Leftarrow ist trivial. ■

Bemerkung

Die Sprache für unsere Kalküle kann aus den Formeln über einem Alphabet Σ bestehen, aber auch aus Modifikationen solcher Formeln oder aus komplizierteren, aus Formeln aufgebauten Datenobjekten. In solchen Fällen werden wir die Definition von \vdash_{Kal} , der Ableitbarkeit in Kal , ggf. so anpassen, daß man doch wieder (direkt) für Formelmengen M und Formeln A schreiben kann

$$M \vdash_{Kal} A.$$

Nimmt man das einmal vorweg, so ist unser Ziel bei zu entwerfenden Kalkülen, daß sie korrekt und vollständig seien im Sinne der folgenden Definition.

Definition 3.5

Kal ist *korrekt*, wenn für beliebige $M \subseteq For0_\Sigma$ gilt

$$M \vdash_{Kal} A \implies M \models A.$$

Definition 3.6

Kal ist *vollständig*, wenn entsprechend stets gilt

$$M \models A \implies M \vdash_{Kal} A.$$

Ein korrekter und vollständiger Kalkül modelliert also genau \models .

Neben der Möglichkeit die Beweise in einem Kalkül als Folgen von Formeln zu definieren, wie wir das in Definition 3.2 getan haben, ist es häufig hilfreich Beweisbäume zu betrachten.

Definition 3.7

Ein *Beweisbaum*, B , in einem Kalkül über einer Menge M von Voraussetzungen ist ein Baum, dessen Knoten mit Formeln des Kalküls, d.h. Elementen aus L , markiert sind, so daß gilt:

- alle Blätter von B sind mit Axiomen oder Elementen aus M markiert,
- ist K ein innerer Knoten von B mit den unmittelbaren Nachfolgerknoten K_1, \dots, K_m , sei u die Markierung von K und u_i die Markierung von K_i , dann gibt es eine Regel R des Kalküls, so daß (u_1, \dots, u_m, u) eine Instanz von R ist.

Satz 3.8

Sei Kal ein beliebiger Kalkül, $M \subset L$.

Dann gilt $M \vdash_{Kal} A$ genau dann, wenn es einen Beweisbaum in Kal über M gibt mit der Wurzelmarkierung A .

Auch der Begriff der Ableitung – oder: des „Beweises“ – wird bei manchen Kalkülen noch etwas modifiziert werden müssen. Wir tun das ohne eine allgemeinere Definition, jeweils am Beispiel.

3.1.3 Übungsaufgaben

Übungsaufgabe 3.1.1

Sei Kal ein beliebiger Kalkül. Zeigen Sie, daß unabhängig von der speziellen Wahl der Regeln für Kal die Ableitungsrelation \vdash_{Kal} monoton und transitiv ist.

Monotonie:

Aus $M \vdash_{Kal} A$ und $M \subseteq M'$ folgt $M' \vdash_{Kal} A$.

Transitivität:

Aus $M \vdash_{Kal} A$ und $N \vdash_{Kal} B$ für alle $B \in M$ folgt $N \vdash_{Kal} A$.

3.2 Der aussagenlogische Hilbertkalkül

DAVID HILBERT ist einer der wesentlichen Begründer der *axiomatische Logik*. Seine Arbeiten markieren den Übergang der Logik von einer philosophischen Disziplin zur *Mathematischen Logik*.

Kalküle vom Hilbert-Typ sind weniger auf Effizienz (Kürze der Beweise) und Komfort (leichtes Finden von Beweisen, Flexibilität im Formulieren von Beweisideen) hin ausgelegt, als auf Sparsamkeit.

Die nachfolgende Darstellung folgt im wesentlichen [Men87].

3.2.1 Der Kalkül

Der folgende Kalkül arbeitet „offiziell“ nicht auf der vollen Sprache $For0_\Sigma$, sondern auf ihrer Einschränkung auf die Basis $\{\neg, \rightarrow\}$. Das nämlich sind die einzigen in den Axiomen auftretenden logischen Operatoren, und auch durch weitere Regelanwendung können keine neuen hinzukommen. Wir gestatten uns trotzdem, gleich in unserer vollen Sprache $For0_\Sigma$ zu arbeiten, indem wir $\mathbf{1}$, $\mathbf{0}$, \wedge , \vee , \leftrightarrow einfach als *Abkürzungen* auffassen. Es stehe

$\mathbf{1}$	für:	$P_0 \rightarrow P_0$	(P_0 ist das erste Element von Σ)
$\mathbf{0}$	für:	$\neg \mathbf{1}$	
$A \wedge B$	für:	$\neg(A \rightarrow \neg B)$	
$A \vee B$	für:	$\neg A \rightarrow B$	
$A \leftrightarrow B$	für:	$(A \rightarrow B) \wedge (B \rightarrow A)$	

Mit der Semantik dieser Operatoren steht das ja im Einklang.

Definition 3.9

Es sei Σ eine aussagenlogische Signatur. $\mathbf{H}0_\Sigma$, oder (im folgenden stets) kurz $\mathbf{H}0$, ist der folgende Kalkül in der aussagenlogischen Sprache über Σ .

$$\text{Ax1: } \frac{}{A \rightarrow (B \rightarrow A)} \quad (\text{Abschwächung})$$

$$\text{Ax2: } \frac{}{(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))} \quad (\text{Verteilung von } \rightarrow)$$

$$\text{Ax3: } \frac{}{(\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A)} \quad (\text{Kontraposition})$$

$$\text{Mp: } \frac{A, A \rightarrow B}{B} \quad (\text{Modus ponens})$$

Die (im Prinzip natürlich jeweils notwendige) Indizierung durch Σ haben wir weggelassen.

Lemma 3.10

Jedes Axiom von $\mathbf{H0}$ (d. h. $A \in \text{Ax1} \cup \text{Ax2} \cup \text{Ax3}$) ist allgemeingültig.

Beweis

Direktes Nachrechnen

■

Beispiel 3.11

Für jede Formel A gilt: $\vdash_{\mathbf{H0}} A \rightarrow A$.

Beweis

Wir geben eine Ableitung aus \emptyset an.

1. $(A \rightarrow ((\underbrace{A \rightarrow A}_B) \rightarrow \underbrace{A}_C)) \rightarrow ((A \rightarrow (\underbrace{A \rightarrow A}_B)) \rightarrow (A \rightarrow \underbrace{A}_C))$
(Ax2)
2. $A \rightarrow ((A \rightarrow A) \rightarrow A)$ Ax1
3. $(A \rightarrow (A \rightarrow A)) \rightarrow (A \rightarrow A)$ Mp auf (2),(1)
4. $A \rightarrow (A \rightarrow A)$ Ax1
5. $A \rightarrow A$ Mp auf (3),(4)

$\mathbf{H0}$ eignet sich nicht besonders gut, um Ableitungen zielgerichtet zu finden. Einiges hierzu läßt sich gleichwohl sagen.

Zunächst einmal haben wir das folgende „Kompositionslemma“

Lemma 3.12

(Komposition mittels Modus ponens)

$$M_1 \vdash_{\mathbf{H0}} A, \quad M_2 \vdash_{\mathbf{H0}} A \rightarrow B \quad \Rightarrow \quad M_1 \cup M_2 \vdash_{\mathbf{H0}} B.$$

Das wichtigste Instrument jedoch für das Finden einer Ableitung bildet der folgende Satz, der eine bestimmte Austauschbarkeit zwischen dem metasprachlichen Operator $\vdash_{\mathbf{H0}}$ und dem objektsprachlichen Operator \rightarrow zum Inhalt hat.

Satz 3.13

(Deduktionstheorem der Aussagenlogik) Für beliebige Formelmengen M und Formeln A, B gilt:

$$M \vdash_{\mathbf{H0}} A \rightarrow B \quad \Leftrightarrow \quad M \cup \{A\} \vdash_{\mathbf{H0}} B$$

Beweis

Wir lassen den Index **H0** weg.

\Rightarrow .

Es gelte $M \vdash A \rightarrow B$. Dann
 $M \cup \{A\} \vdash A \rightarrow B$. (erst recht)
 $M \cup \{A\} \vdash A$ (trivialerweise)
 $M \cup \{A\} \vdash B$ (Mp)

\Leftarrow .

Es gelte: $M \cup \{A\} \vdash B$. Sei (A_1, \dots, A_m) Ableitung von B aus $M \cup \{A\}$. Wir wollen zeigen: $M \vdash A \rightarrow B$, d. h. $M \vdash A \rightarrow A_m$. Wir zeigen sogar für alle $i \in \{1, \dots, m\}$, daß $M \vdash A \rightarrow A_i$, und zwar durch Induktion über i . Sei i mit $1 \leq i \leq m$ gegeben. Nehmen wir also an, daß für alle $j < i$ schon gezeigt ist: $M \vdash A \rightarrow A_j$.

Fall 1:

$A_i \in M \cup \text{Ax1} \cup \text{Ax2} \cup \text{Ax3}$. Dann gilt:
 $M \vdash A_i$ (trivial)
 $M \vdash A_i \rightarrow (A \rightarrow A_i)$ (Ax1)
 $M \vdash A \rightarrow A_i$ (Mp)

Fall 2:

$A_i = A$. In Beispiel 3.11 haben wir schon gezeigt: $M \vdash A \rightarrow A_i$.

Fall 3:

Es gibt $j < i$ und $k < i$ mit $A_k = A_j \rightarrow A_i$. Nach obiger Annahme (Induktionsvoraussetzung) wissen wir:

$M \vdash A \rightarrow A_j$
 $M \vdash A \rightarrow (A_j \rightarrow A_i)$

Man hat ferner

$M \vdash (A \rightarrow (A_j \rightarrow A_i)) \rightarrow ((A \rightarrow A_j) \rightarrow (A \rightarrow A_i))$ (Ax2)

also

$M \vdash A \rightarrow A_i$ (2mal Mp). ■

Wir geben jetzt Beispiele ableitbarer Formeln an (aus [Men87]). Die hingeschriebenen Beweise sind nicht unmittelbar Ableitungen, sondern Folgen von Ableitbarkeitsaussagen, die mit Hilfe von

- Axiomen

- dem Kompositionslemma (Mp)
- dem Deduktionstheorem

gewonnen sind und aus denen sich Ableitungen dann unmittelbar herstellen lassen.

Beispiel 3.14

Für beliebige Formeln A, B, C sind in $\mathbf{H0}$ aus \emptyset ableitbar

- | | | |
|------|---|-------------------------------|
| (1) | $(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$ | (Transitivität) |
| (2) | $A \rightarrow ((A \rightarrow B) \rightarrow B)$ | (Modus ponens) |
| (3) | $\neg\neg A \rightarrow A$ | (Doppelte Negation) |
| (4) | $A \rightarrow \neg\neg A$ | |
| (5) | $(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$ | (Kontraposition) |
| (6) | $A \rightarrow (\neg B \rightarrow \neg(A \rightarrow B))$ | (Semantik von \rightarrow) |
| (7) | $\neg A \rightarrow (A \rightarrow B)$ | |
| (8) | $A \rightarrow (B \rightarrow (A \rightarrow B))$ | |
| (9) | $\neg(A \rightarrow B) \rightarrow A$ | |
| (10) | $\neg(A \rightarrow B) \rightarrow \neg B$ | |
| (11) | $(A \rightarrow \neg A) \rightarrow \neg A$ | (Selbstwiderlegung) |
| (12) | $(\neg A \rightarrow A) \rightarrow A$ | |
| (13) | $(A \rightarrow B) \rightarrow ((\neg A \rightarrow B) \rightarrow B)$ | (Fallunterscheidung) |
| (14) | $\neg(A \rightarrow A) \rightarrow B$ | (Ex falso quodlibet) |

Beweis

Wir sollten versuchen, so oft wie möglich das Deduktionstheorem (DT) zu verwenden. Statt $\vdash_{\mathbf{H0}}$ schreiben wir kurz \vdash .

- | | | |
|-----|--|-----------|
| (1) | $\{A \rightarrow B, B \rightarrow C, A\} \vdash A$ | (trivial) |
| | $\{A \rightarrow B, B \rightarrow C, A\} \vdash A \rightarrow B$ | (trivial) |
| | $\{A \rightarrow B, B \rightarrow C, A\} \vdash B$ | (Mp) |
| | $\{A \rightarrow B, B \rightarrow C, A\} \vdash B \rightarrow C$ | (trivial) |
| | $\{A \rightarrow B, B \rightarrow C, A\} \vdash C$ | (Mp) |
| | $\{A \rightarrow B, B \rightarrow C\} \vdash A \rightarrow C$ | (DT) |
| | $\{A \rightarrow B\} \vdash (B \rightarrow C) \rightarrow (A \rightarrow C)$ | (DT) |
| | $\vdash (A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$ | (DT) |

(Versuchen Sie (eine Zeit lang) direkt, also ohne Verwendung des Deduktionstheorems, eine Ableitung (A_1, \dots, A_n) von $(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$ aus \emptyset hinzuschreiben.)

- (2) $\{A, A \rightarrow B\} \vdash A$ (trivial)
 $\{A, A \rightarrow B\} \vdash A \rightarrow B$ (trivial)
 $\{A, A \rightarrow B\} \vdash B$ (Mp)
 $\vdash A \rightarrow ((A \rightarrow B) \rightarrow B)$ (2mal DT)
- (3) $\vdash \neg\neg A \rightarrow (\neg\neg\neg\neg A \rightarrow \neg\neg A)$ (Ax1)
 $\{\neg\neg A\} \vdash \neg\neg\neg\neg A \rightarrow \neg\neg A$ (DT)
 $\{\neg\neg A\} \vdash (\neg\neg\neg\neg A \rightarrow \neg\neg A) \rightarrow (\neg A \rightarrow \neg\neg\neg A)$ (Ax3)
 $\{\neg\neg A\} \vdash \neg A \rightarrow \neg\neg\neg A$ (Mp)
 $\{\neg\neg A\} \vdash (\neg A \rightarrow \neg\neg\neg A) \rightarrow (\neg\neg A \rightarrow A)$ (Ax3)
 $\{\neg\neg A\} \vdash \neg\neg A \rightarrow A$ (Mp)
 $\{\neg\neg A\} \vdash A$ (DT)
 $\vdash \neg\neg A \rightarrow A$ (DT)
- (4) $\vdash \neg\neg\neg A \rightarrow \neg A$ (mit (3))
 $\vdash (\neg\neg\neg A \rightarrow \neg A) \rightarrow (A \rightarrow \neg\neg A)$ (Ax3)
 $\vdash A \rightarrow \neg\neg A$ (Mp)
- (5) $\{A \rightarrow B\} \vdash (\neg\neg A \rightarrow \neg\neg B) \rightarrow (\neg B \rightarrow \neg A)$ (Ax3)
 $\{A \rightarrow B\} \vdash \neg\neg A \rightarrow A$ (mit (3))
 $\{A \rightarrow B\} \vdash A \rightarrow B$ (trivial)
 $\{A \rightarrow B\} \vdash \neg\neg A \rightarrow B$ (mit (1), 2mal Mp)
 $\{A \rightarrow B\} \vdash B \rightarrow \neg\neg B$ (mit (4))
 $\{A \rightarrow B\} \vdash \neg\neg A \rightarrow \neg\neg B$ (mit (1), 2mal Mp)
 $\{A \rightarrow B\} \vdash \neg B \rightarrow \neg a$ (Mp)
 $\vdash (A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$ (DT)
- (6) $\vdash A \rightarrow ((A \rightarrow B) \rightarrow B)$ (mit (2))
 $\{A\} \vdash (A \rightarrow B) \rightarrow B$ (DT)
 $\{A\} \vdash ((A \rightarrow B) \rightarrow B) \rightarrow (\neg B \rightarrow \neg(A \rightarrow B))$ (siehe (5))
 $\{A\} \vdash \neg B \rightarrow \neg(A \rightarrow B)$ (Mp)
 $\vdash A \rightarrow (\neg B \rightarrow \neg(A \rightarrow B))$ (DT)
- (7) $\vdash \neg A \rightarrow (\neg B \rightarrow \neg A)$ (Ax1)
 $\{\neg A\} \vdash \neg B \rightarrow \neg A$ (DT)
 $\{\neg A\} \vdash (\neg B \rightarrow \neg A) \rightarrow (A \rightarrow B)$ (Ax3)
 $\{\neg A\} \vdash (A \rightarrow B)$ (Mp)
 $\vdash \neg A \rightarrow (A \rightarrow B)$ (DT)

- (8) $\{A\} \vdash B \rightarrow (A \rightarrow B)$ (Ax1)
 $\vdash A \rightarrow (B \rightarrow (A \rightarrow B))$ (DT)
- (9) $\vdash \neg A \rightarrow (A \rightarrow B)$ (mit (7))
 $\vdash \neg(A \rightarrow B) \rightarrow \neg\neg A$ (mit (5), Mp)
 $\vdash \neg\neg A \rightarrow A$ (mit (3))
 $\vdash \neg(A \rightarrow B) \rightarrow A$ (mit (1), 2mal Mp)
- (10) $\vdash B \rightarrow (A \rightarrow B)$ (Ax1)
 $\vdash \neg(A \rightarrow B) \rightarrow \neg B$ (mit (5), Mp)
- (11) $\vdash A \rightarrow (\neg\neg A \rightarrow \neg(A \rightarrow \neg A))$ (mit (6))
 $\{A\} \vdash \neg\neg A \rightarrow \neg(A \rightarrow \neg A)$ (mit DT)
 $\{A\} \vdash \neg\neg A$ (mit (4), Mp)
 $\{A\} \vdash \neg(A \rightarrow \neg A)$ (Mp)
 $\vdash A \rightarrow \neg(A \rightarrow \neg A)$ (DT)
 $\vdash (A \rightarrow \neg(A \rightarrow \neg A)) \rightarrow (\neg\neg(A \rightarrow \neg A) \rightarrow \neg A)$ (mit (5))
 $\vdash \neg\neg(A \rightarrow \neg A) \rightarrow \neg A$ (Mp)
 $\vdash (A \rightarrow \neg A) \rightarrow \neg\neg(A \rightarrow \neg A)$ (mit (4))
 $\vdash (A \rightarrow \neg A) \rightarrow \neg A$ (mit (1), 2mal Mp)
- (12) $\vdash (\neg A \rightarrow \neg\neg A) \rightarrow \neg\neg A$ (mit (11))
 Verwende nun (3) und (4).
- (13) $\{(A \rightarrow B), (\neg A \rightarrow B)\} \vdash (A \rightarrow B)$ (trivial)
 $\{(A \rightarrow B), (\neg A \rightarrow B)\} \vdash (\neg B \rightarrow \neg A)$ (mit (5), Mp)
 $\{(A \rightarrow B), (\neg A \rightarrow B)\} \vdash (\neg A \rightarrow B)$ (trivial)
 $\{(A \rightarrow B), (\neg A \rightarrow B)\} \vdash (\neg B \rightarrow B)$ (mit (1), 2mal Mp)
 $\vdash (\neg B \rightarrow B) \rightarrow B$ (mit (12))
 $\{(A \rightarrow B), (\neg A \rightarrow B)\} \vdash B$ (Mp)
 $\vdash (A \rightarrow B) \rightarrow ((\neg A \rightarrow B) \rightarrow B)$ (2mal DT)
- (14) $\{\neg B\} \vdash A \rightarrow A$ (mit dem Beispiel in 3.2)
 $\vdash (A \rightarrow A) \rightarrow \neg\neg(A \rightarrow A)$ (mit (4))
 $\{\neg B\} \vdash \neg\neg(A \rightarrow A)$ (Mp)
 $\vdash \neg B \rightarrow \neg\neg(A \rightarrow A)$ (DT)
 $\vdash \neg(A \rightarrow A) \rightarrow B$ (Ax3, Mp)

Für die folgenden Formelschemata versuche der Leser selbst, die Ableitbar-

keit zu beweisen. Wie schon gesagt, sind $\mathbf{1}$, $\mathbf{0}$, \vee , \wedge , \leftrightarrow als Abkürzungen aufzufassen.

$$(15) \quad A \rightarrow \mathbf{1}$$

$$(20) \quad A \rightarrow (A \vee B)$$

$$(16) \quad \mathbf{0} \rightarrow A$$

$$(21) \quad A \wedge B \leftrightarrow B \wedge A$$

$$(17) \quad A \rightarrow (B \rightarrow A \wedge B)$$

$$(22) \quad A \wedge A \leftrightarrow A$$

$$(18) \quad A \vee B \leftrightarrow B \vee A$$

$$(23) \quad A \wedge A \rightarrow A$$

$$(19) \quad A \vee A \leftrightarrow A$$

$$(24) \quad \neg A \leftrightarrow (A \rightarrow \mathbf{0})$$

■

Übungsaufgabe 3.2.1

Beweisen Sie im Hilbert-Kalkül: $A, \neg A \vdash B$.

3.2.2 Metatheoreme

Satz 3.15

Sei $M \subseteq \text{For}_{0\Sigma}$ und $A \in \text{For}_{0\Sigma}$.

1. Aus $M \vdash_{\mathbf{H0}} A$ folgt $M \models A$
Korrektheit von **H0**.
2. Insbesondere gilt für $M = \emptyset$
Gilt $M \vdash_{\mathbf{H0}} A$ dann ist A eine Tautologie.
3. Aus $M \models A$ folgt $M \vdash_{\mathbf{H0}} A$.
Vollständigkeit von **H0**.
4. Insbesondere gilt für $M = \emptyset$
Ist A eine Tautologie dann gilt $M \vdash_{\mathbf{H0}} A$.
5. Gilt $M \models A$, dann gibt es eine endliche Teilmenge $E \subseteq M$ mit $E \models A$.

Beweis:

zu Teil 1 Es gelte $M \vdash_{\mathbf{H0}} A$, d. h. es gibt eine Ableitung (A_1, \dots, A_m) in $\mathbf{H0}$ mit $A_m = A$. Zu zeigen ist $M \models A$, d. h. $val_I(A) = \mathbf{W}$ für jede Interpretation I von Σ , welche Modell von M ist. Sei I eine solche Interpretation. Wir zeigen (schärfer)

$$val_I(A_i) = \mathbf{W} \text{ für } i = 1, \dots, m$$

und zwar durch vollständige Induktion nach i . Nehmen wir also an, daß für alle $j < i$ schon gezeigt sei $val_I(A_j) = \mathbf{W}$.

- Fall 1: $A_i \in M$. Dann gilt $val_I(A_i) = \mathbf{W}$ nach Voraussetzung.
- Fall 2: A_i ist Axiom. Dann ist A_i allgemeingültig (Lemma 3.10).
- Fall 3: Es gibt $j, k < i$ mit $A_k = A_j \rightarrow A_i$. Nach Induktionsvoraussetzung ist dann $val_I(A_j) = \mathbf{W}$ und $val_I(A_j \rightarrow A_i) = \mathbf{W}$, und aus der Semantik von \rightarrow ergibt sich $val_I(A_i) = \mathbf{W}$.

zu Teil 2 Spezialfall von Teil 1.

zu Teil 3 Siehe z.B. [Men87, Ric78]

zu Teil 4 Spezialfall von Teil 3.

zu Teil 5 Gilt $M \models A$, dann gilt auch $M \vdash_{\mathbf{H0}} A$ nach Teil 3. Nach Definition ist eine Ableitung eine endliche Folge von Formeln. In dieser endlichen Folge kann nur eine endliche Teilmenge E von Elementen aus M vorkommen. Offensichtlich gilt immer noch $E \vdash_{\mathbf{H0}} A$ und mit Teil 1 folgt $E \models A$.

■

Übungsaufgabe 3.2.2

(Endlichkeitssatz) Für beliebige $M \subseteq For0_\Sigma$ gilt:

M hat ein Modell \Leftrightarrow Jede endliche Teilmenge von M hat ein Modell.

3.3 Aussagenlogische Resolution

Merkmale des Resolutionskalküls

- Widerlegungskalkül
- Voraussetzung: Alle Formeln sind in konjunktiver Normalform.
- Es gibt eine einzige Regel (Resolution), die eine Modifikation des Modus ponens ist
- Es sind keine logischen Axiome mehr notwendig; insbesondere entfällt die Suche nach („langen“) passenden Instantiierungen der Axiomenschemata Ax1, Ax2, Ax3.

3.3.1 Syntax und Semantik

Wir verwenden die *Sonderzeichen*

- \neg
- \square (für die leere Klausel, s. u.)
- ferner Zeichen für die Bildung von Mengen: geschweifte Klammern und Komma.

Gegeben sei eine aussagenlogische *Signatur* Σ (Atome), *Literale* sind die Atome und die negierten Atome. Eine *Klausel* ist eine endliche Menge von Literalen. \square ist die *leere Klausel*. Die grundlegende Datenstruktur, auf der wir operieren, sind die *Mengen von Klauseln*.

Eine Klausel lesen wir als die Disjunktion der ihr angehörenden Literale, eine Menge von Klauseln als die Konjunktion der den Klauseln entsprechenden Disjunktionen. Bis auf die Reihenfolge der Disjunktions- bzw. Konjunktionsglieder sind also in umkehrbar eindeutiger Weise die Formeln in konjunktiver Normalform wiedergegeben. Wir schreiben C, C_i, \dots für Klauseln (also $C = \square$ oder $C = \{L_1, \dots, L_k\}$ für Literale L_1, \dots, L_k) und M, M_j, \dots für Mengen von Klauseln. Ist $I : \Sigma \rightarrow \{W, F\}$ eine Interpretation der Atome, so hat man als zugehörige Auswertung

$$val_I : \text{Mengen von Klauseln} \rightarrow \{W, F\}$$

die folgende:

$$val_I(\{\{L\}\}) = \begin{cases} I(L) & \text{falls } L \in \Sigma \\ nicht(I(P)) & \text{falls } L = \neg P, P \in \Sigma \end{cases}$$

für Literale L,

$$val_I(\{C\}) = \begin{cases} W & \text{falls ein } L \in C \text{ existiert mit } val_I(\{\{L\}\}) = W \\ F & \text{sonst} \end{cases}$$

für Klauseln C,

$$val_I(M) = \begin{cases} W & \text{falls für alle } C \in M \text{ gilt: } val_I(\{C\}) = W \\ F & \text{sonst} \end{cases}$$

Es folgt also

$$\begin{aligned} val_I(\{\square\}) &= F, \\ val_I(\emptyset) &= W \end{aligned}$$

für die leere Klauselmenge \emptyset . Wir schreiben zur Abkürzung

$$\begin{aligned} val_I(L) &\text{ statt } val_I(\{\{L\}\}), \\ val_I(C) &\text{ statt } val_I(\{C\}). \end{aligned}$$

3.3.2 Kalkül

Definition 3.16 (Resolutionsregel)

Die *aussagenlogische Resolutionregel* ist die Regel

$$\frac{C_1 \cup \{P\}, C_2 \cup \{\neg P\}}{C_1 \cup C_2} \quad \text{mit: } P \in \Sigma, C_1, C_2 \text{ Klauseln über } \Sigma.$$

$C_1 \cup C_2$ heißt *Resolvente* von $C_1 \cup \{P\}$, $C_2 \cup \{\neg P\}$.

Die Resolutionsregel ist eine Modifikation des modus ponens. Es ist nämlich auch die Erweiterung von MP,

$$(MP') \quad \frac{A \vee C, A \rightarrow B}{C \vee B}, \quad \text{d.h.} \quad \frac{A \vee C, \neg A \vee B}{C \vee B}$$

eine gültige Regel. Setzt man $A := P$, $C_1 := C$ und $C_2 := B$, so erhält man gerade die Resolutionsregel.

Definition 3.17 (Resolutionskalkül)

Sei M eine Klauselmenge.

1. Mit $Res(M)$ bezeichnen wir die Menge aller Resolventen von Klauseln aus M .
2. $R_0^0(M) = M$
3. $R_0^{n+1}(M) = Res(R^n) \cup R_0^n$
4. $M \vdash_{\mathbf{R0}} A$ gilt genau dann, wenn es ein n gibt mit $A \in R_0^n(M)$

Der Index 0 soll wieder daran erinnern, daß wir es hier mit der aussagenlogischen Variante des Resolutionskalküls zu tun haben. Die Erweiterung auf die Prädikatenlogik erfolgt später, siehe Def. 5.25.

Da $\mathbf{R0}$ als Widerlegungskalkül gedacht ist, interessieren wir uns für Situationen $M \vdash_{\mathbf{R0}} \square$, wo also aus der Klauselmenge M die leere Klausel durch Resolution ableitbar ist. Wenn aus dem Kontext ersichtlich schreiben wir \vdash einfach anstelle von $\vdash_{\mathbf{R0}}$.

Beispiel 3.18

Gegeben sei die Klauselmenge

$$M = \{\{P_1, P_2\}, \{P_1, \neg P_2\}, \{\neg P_1, P_2\}, \{\neg P_1, \neg P_2\}\}$$

$$\frac{\{P_1, P_2\}, \{P_1, \neg P_2\}}{\{P_1\}}$$

$$\frac{\{\neg P_1, P_2\}, \{\neg P_1, \neg P_2\}}{\{\neg P_1\}}$$

$$\frac{\{P_1\}, \{\neg P_1\}}{\square}$$

Insgesamt:

$$M \vdash_{Res} \square$$

3.3.3 Korrektheit und Vollständigkeit

Vorbemerkung

Im Gegensatz zu $\mathbf{H0}$ enthält $\mathbf{R0}$ keine logischen Axiome. Im Sinne des direkten Beweisens kann $\mathbf{R0}$ also nicht vollständig sein. Z. B. gilt für die Formel $P \rightarrow P$ – oder gleichbedeutend für die Klausel $\{\neg P, P\}$ – sicherlich

$$\emptyset \not\vdash_{\mathbf{R0}} \{\neg P, P\}.$$

R0 soll ja auch als Widerlegungskalkül arbeiten, und zwar auf Klauselmengen, d. h. Formeln in KNF. Es wird also (wenn wir zunächst wieder in unserer vertrauten Sprache mit Formeln aus $For0_\Sigma$ formulieren) eine Folgerbarkeitsaussage

$$M \models A$$

($M \subseteq For0_\Sigma, A \in For0_\Sigma$) zuerst einmal umformuliert in die gleichbedeutende

$$M \cup \{\neg A\} \text{ ist unerfüllbar}$$

oder

$$M \cup \{\neg A\} \models \underline{false}.$$

Bringt man die Formeln $M \cup \{\neg A\}$ in KNF und schreibt sie dann als Klauselmenge, etwa \overline{M} , schreibt man ferner wieder \square statt \underline{false} , so erhält man als wiederum gleichbedeutende Aussage

$$\overline{M} \text{ ist unerfüllbar}$$

oder

$$\overline{M} \models \square.$$

R0 wäre als Widerlegungskalkül korrekt und vollständig, wenn dies, für beliebige \overline{M} , äquivalent wäre zur Ableitbarkeit von \square aus \overline{M} in **R0**.

In unserem Beispiel $P \rightarrow P$ haben wir die $\{\neg[P \rightarrow P]\}$ entsprechende Klauselmenge zu bilden, also $\{\{\neg P\}, \{P\}\}$. Sie liefert in einem einzigen Resolutionschritt \square .

Satz 3.19 (Korrektheit und Vollständigkeit des aussagenlogischen Resolutionskalküls)

Es sei M eine Menge von Klauseln über einer aussagenlogischen Signatur Σ .

Dann gilt

$$M \text{ unerfüllbar} \quad \Leftrightarrow \quad M \vdash_{\mathbf{R0}} \square.$$

Beweis

Korrektheit: Übung

Vollständigkeit:

Wir zeigen die gewünschte Behauptung durch Kontraposition, d.h. wir gehen von der Voraussetzung aus, daß \square aus M mit Hilfe des Kalküls **R0** nicht herleitbar ist und beweisen die Erfüllbarkeit von M .

Wir fixieren eine feste Reihenfolge der Atome, z. B. die durch den Index gegebene, P_0, \dots, P_n, \dots .

Mit M_0 bezeichnen wir die Menge aller Klauseln C , für die $M \vdash_{R0} C$ gilt. Es gilt also $M \subseteq M_0$ und $\square \notin M_0$. Wir werden im folgenden eine Interpretation I angeben, so daß $val_I(M_0) = W$ ist, d.h. für alle $C \in M_0$ ist $val_I(C) = W$. Insbesondere ist dann M als erfüllbar nachgewiesen.

Wir definieren I induktiv. Angenommen $I(P_0), \dots, I(P_{n-1})$ sind schon festgelegt, dann setzen wir

$$I(P_n) = W,$$

genau dann, wenn gilt:

M_0 enthält keine Klausel $C = C_1 \cup \{\neg P_n\}$, so daß in C_1 nur Atome P_i mit $i < n$ auftreten, und $val_I(C_1) = F$.

Für den Induktionsanfang besagt das:

$$I(P_0) = W \Leftrightarrow \{\neg P_0\} \notin M_0.$$

Es bleibt nachzuweisen, daß für alle $C \in M_0$ gilt $val_I(C) = W$. Für jede Klausel C sei $index(C) = i + 1$, wobei i die größte natürliche Zahl, so daß P_i , negiert oder unnegiert, in C vorkommt. Beachten Sie, daß $\square \notin M_0$. Die Behauptung wird durch Induktion über $index(C)$ bewiesen.

Der Induktionsanfang ist trivialerweise erfüllt, da für keine Klausel C gilt, daß $index(C) = 0$.

Sei also für alle $C \in M_0$ mit $index(C) < n$ schon $val_I(C) = W$ nachgewiesen und $D \in M_0$ mit $index(D) = n$ vorgelegt.

1. Fall $D = D_1 \cup \{\neg P_n\}$, $index(D_1) < n$

Falls schon $val_I(D_1) = W$ gilt, dann gilt natürlich umso mehr $val_I(D) = W$. Falls $val_I(D_1) = F$, dann sind wir genau in der Situation, die $I(P_n) = F$ in der Definition von I erzwungen hat. Woraus auch $val_I(D) = W$ folgt.

2. Fall $D = D_1 \cup \{P_n\}$, $index(D_1) < n$

Falls schon $val_I(D_1) = W$ gilt, dann sind wir natürlich fertig. Also betrachten wir im folgenden nur noch die Möglichkeit $val_I(D_1) = F$. Falls $I(P_n) = W$ gilt, dann haben wir sofort $val_I(D) = W$, wie gewünscht. Wenn $I(P_n) = F$ gilt, dann muß es dafür nach Definition von I einen Grund geben, d.h. eine Klausel $C = C_1 \cup \{\neg P_n\} \in M_0$, so daß $index(C_1) < n$ und $val_I(C_1) = F$. Da M_0 abgeschlossen ist unter Ableitbarkeit in $R0$ muß auch die Resolvente $R = C_1 \cup D_1$ als das Ergebnis der Resolution von C mit D in M_0 liegen. Da offensichtlich $index(R) < n$ gilt, muß nach Induktionsvoraussetzung $val_I(R) = W$ sein. Das steht aber im Widerspruch zu $val_I(D_1) = F$ und $val_I(C_1) = F$. Der zuletzt betrachtete Fall kann also nicht auftreten und

$val_I(D) = W$ ist in allen Fällen nachgewiesen.

3.Fall $D = D_1 \cup \{P_n, \neg P_n\}$, $index(D_1) < n$

In diesem Fall ist D eine Tautologie und $val_I(D) = W$ ist in jedem Fall wahr. ■

Beispiel 3.20

Als ein erstes Beispiel für das Arbeiten mit dem Resolutionskalkül stellen wir uns die Aufgabe zu zeigen, daß

$$(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$$

eine Tautologie ist.

Dazu werden wir zeigen, daß die Negation dieser Formel

$$\neg((A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C)))$$

nicht erfüllbar ist.

Als erstes muß diese Formel in Klauselnormalform gebracht werden. Das Ergebnis ist

$$M = \{\{\neg A, B\}, \{\neg B, C\}, \{A\}, \{\neg C\}\}$$

Aus M läßt sich wie folgt die leere Klausel ableiten

- (1) \square $\{\neg A, B\}$
- (2) \square $\{\neg B, C\}$
- (3) \square $\{A\}$
- (4) \square $\{\neg C\}$
- (5) [1, 3] $\{B\}$
- (6) [2, 5] $\{C\}$
- (7) [4, 6] \square

Um Resolutionsbeweise leichter lesbar zu machen numerieren wir alle darin vorkommenden Klauseln durch und geben in eckigen Klammern jeweils die Nummern der Elternklauseln an.

Notwendigkeit der Mengenschreibweise

Die Menge von Formeln

$$E = \{P_1 \vee \neg P_2, \neg P_1 \vee P_2, \neg P_1 \vee \neg P_2, P_1 \vee P_2\}$$

ist sicherlich nicht erfüllbar.

Resolutionsmöglichkeiten:

$$\frac{\neg P_1 \vee \neg P_2, P_1 \vee \neg P_2}{\neg P_2 \vee \neg P_2} \quad \frac{\neg P_1 \vee \neg P_2, \neg P_1 \vee P_2}{\neg P_1 \vee \neg P_1}$$

$$\frac{\neg P_1 \vee \neg P_1, P_1 \vee \neg P_2}{\neg P_1 \vee \neg P_2} \quad \frac{\neg P_2 \vee \neg P_2, \neg P_1 \vee P_2}{\neg P_1 \vee \neg P_2}$$

Auf diese Weise ist \square nicht herleitbar.

3.3.4 Übungsaufgaben

Übungsaufgabe 3.3.1

Beweisen Sie mit Resolution:

$$\begin{array}{l}
 (p1h1 \rightarrow \neg p2h1) \\
 \wedge (p1h1 \rightarrow \neg p3h1) \\
 \wedge (p2h1 \rightarrow \neg p3h1) \\
 \wedge (p1h2 \rightarrow \neg p2h2) \\
 \wedge (p1h2 \rightarrow \neg p3h2) \\
 \wedge (p2h2 \rightarrow \neg p3h2)
 \end{array}
 \quad \vdash \quad
 \begin{array}{l}
 (\neg p1h1 \wedge \neg p1h2) \\
 \vee (\neg p2h1 \wedge \neg p2h2) \\
 \vee (\neg p3h1 \wedge \neg p3h2)
 \end{array}$$

Diese Aufgabe ist die aussagenlogische Formulierung eines kombinatorischen Prinzips, des sog. Dirichletschen Schubfachprinzip. Es thematisiert die offensichtliche Tatsache, daß man $n + 1$ verschiedene Dinge nur dann in n Schubladen unterbringen kann, wenn in mindestens einer Schublade mindestens zwei Dinge untergebracht sind. Im Englischen redet man von $n + 1$ Tauben und n Taubenschlägen und nennt das ganze *pigeon hole principle*. Dieses Prinzip ist für den Fall $n = 2$ so in aussagenlogische Form umgesetzt, daß ein Atom $p1h1$ wahr ist, wenn die 1. Taube im 1. Taubenschlag untergebracht ist, und analog für die anderen Atome $p1h_j$. Die Voraussetzung besagt dann, daß keine zwei Tauben im selben Taubenschlag untergebracht sind und die zu beweisende Disjunktion besagt, daß dann eine Taube nicht untergebracht ist.

Übungsaufgabe 3.3.2

Man könnte versucht sein, zur Verkürzung von Beweisen im Resolutionskalkül zwei Resolutionsanwendungen in einer neuen Regel zusammenzufassen:

$$\frac{C_1 \cup \{P, Q\}, C_2 \cup \{\neg P, \neg Q\}}{C_1 \cup C_2}$$

Zeigen Sie, daß diese Regel nicht korrekt ist.

Übungsaufgabe 3.3.3

Beweisen Sie den Satz 2.47 (Seite 48):

Für Krom-Formeln ist die Erfüllbarkeit in polynomialer Zeit entscheidbar.

Übungsaufgabe 3.3.4

Definition 3.21 (Geordnete Resolution)

Sei $< \subseteq At \times At$ eine beliebige Ordnungsrelation auf der Menge At , siehe Definition 1.1 auf Seite 1

Die Ordnungsrelation wird fortgesetzt auf die Menge aller Literale, indem die Negationszeichen schlichtweg ignoriert werden, d.h. für zwei Literale $L_1 \in \{A_1, \neg A_1\}$, $L_2 \in \{A_2, \neg A_2\}$ gilt

$$L_1 < L_2 \quad \text{gdw} \quad A_1 < A_2$$

.

Ein Literal L ist $<$ -maximal in einer Klausel C gdw. es kein $L' \in C$ gibt mit $L < L'$.

Eine Anwendung der Resolutionsregel

$$\frac{C_1 \cup \{P\}, C_2 \cup \{\neg P\}}{C_1 \cup C_2}$$

heißt ein geordneter Resolutionsschritt bezüglich $<$, wenn P maximal in $C_1 \cup \{P\}$ und $\neg P$ maximal in $C_2 \cup \{\neg P\}$ ist.

Beweisen oder widerlegen Sie die Vollständigkeit einer Variante des Resolutionskalküls, bei der nur geordnete Resolutionsschritte erlaubt sind.

Hinweis: Beachten Sie den Beweis des Vollständigkeitssatzes 3.19 von Seite 71.

Übungsaufgabe 3.3.5

In dieser Aufgabe wollen wir eine Variante des Vollständigkeitsbeweises des Resolutionskalküls, siehe Satz 3.19, untersuchen.

Wir ändern die induktive Definition der Belegung I folgendermaßen ab:

$$I(P_n) = W,$$

genau dann, wenn gilt:

M_0 enthält keine Klausel $C = C_1 \cup \{\neg P_n\}$, so daß in C_1 nur negative Literale $\neg P_i$ mit $i < n$ auftreten, und $val_I(C_1) = F$.

Zeigen Sie, daß auch für diese Belegung I für alle Klauseln $C \in M_0$ gilt $val_I(C) = W$.

Übungsaufgabe 3.3.6

Wir nennen einen Resolutionsschritt

$$\frac{C_1 \cup \{P\}, C_2 \cup \{\neg P\}}{C_1 \cup C_2}$$

einen *negativen Resolutionsschritt* wenn die Klausel $C_2 \cup \{\neg P\}$ nur negative Literale enthält.

Beweisen oder widerlegen Sie die Vollständigkeit einer Variante des Resolutionskalküls, bei der nur negative Resolutionsschritte erlaubt sind.

3.4 Aussagenlogische Tableaux

Vorbemerkung

Auch der im folgenden vorgestellte Tableauekalkül ist ein Widerlegungskalkül, und er ist korrekt und vollständig:

$$M \models A \quad \Leftrightarrow \quad M \cup \{\neg A\} \vdash_{\mathbf{T0}} \mathbf{0}.$$

Er leistet aber noch mehr: wenn $M \not\models A$ und M endlich, dann wird eine Interpretation I , so daß val_I auf M wahr und auf A falsch ist, tatsächlich geliefert.

Literatur: [Smu68], [Fit90].

3.4.1 Syntax und Semantik

Wir betrachten in diesem Abschnitt aussagenlogische Formeln aus $A \in For0_\Sigma$, wie sie in Definition 2.3 eingeführt wurden. Die Operatoren sind also \neg , \wedge , \vee , \rightarrow und $\mathbf{1}$ und $\mathbf{0}$ treten als Konstanten auf.

Definition 3.22 (Vorzeichenformeln)

Eine *Vorzeichenformel* ist eine Zeichenkette der Gestalt $0A$ oder $1A$ mit $A \in For0_\Sigma$. Dabei sind 0 , 1 neue Sonderzeichen, die Vorzeichen, im Alphabet der Objektsprache.

Vorzeichen treten nur einmal am Beginn einer Formel auf, deswegen ist auch eine Verwechslung mit den Booleschen Konstanten ausgeschlossen.

Mit der Kuriosität, daß $01, 11, 00, 10$ legitime Vorzeichenformeln sind wollen wir leben,.

Bemerkung

Die Spracherweiterung ist nicht wirklich notwendig, sondern dient nur der Durchsichtigkeit und Anschaulichkeit des Tableauverfahrens und bietet außerdem einen Ansatzpunkt für Verallgemeinerungen auf Tableauverfahren für nichtklassische Logiken. Das Vorzeichen und die aussagenlogische Verknüpfung führen zu einer Unterscheidung von zehn Typen von Formeln. Um spätere Beweise durch Fallunterscheidung zu vereinfachen, übernehmen wir die von R.Smullyan eingeführte **uniforme Notation**.

Definition 3.23 (Typen von Vorzeichenformeln)

Typ ϵ (elementarer Typ): $0P, 1P$ für $P \in \Sigma$
 Typ α (konjunktiver Typ): $0\neg B, 1\neg B, 1(B \wedge C), 0(B \vee C), 0(B \rightarrow C)$
 Typ β (disjunktiver Typ): $0(B \wedge C), 1(B \vee C), 1(B \rightarrow C)$

Definition 3.24 (Abkömmlinge)

Zu jeder nicht elementaren Vorzeichenformel V definieren wir zwei *Abkömmlinge* V_1, V_2 (die auch identisch sein können) gemäß den folgenden Tabellen

Vom Typ α	V_1	V_2
$0\neg B$	$1B$	$1B$
$1\neg B$	$0B$	$0B$
$1(B \wedge C)$	$1B$	$1C$
$0(B \vee C)$	$0B$	$0C$
$0(B \rightarrow C)$	$1B$	$0C$

Vom Typ β	V_1	V_2
$0(B \wedge C)$	$0B$	$0C$
$1(B \vee C)$	$1B$	$1C$
$1(B \rightarrow C)$	$0B$	$1C$

Elementare Vorzeichenformeln haben keine Abkömmlinge.

Ist I eine Interpretation der Atome und val_I die zugehörige Auswertung der Formeln ohne Vorzeichen in $For0_\Sigma$. Wir setzen val_I fort auf die Menge aller Vorzeichenformeln durch

$$val_I(0A) = val_I(\neg A),$$

und

$$val_I(1A) = val_I(A).$$

Lemma 3.25

Für eine Vorzeichenformel V gilt

$$\begin{aligned} \text{vom Typ } \alpha: \quad & val_I(V) = W \Leftrightarrow val_I(V_1) = W \quad \text{und} \quad val_I(V_2) = W \\ \text{vom Typ } \beta: \quad & val_I(V) = W \Leftrightarrow val_I(V_1) = W \quad \text{oder} \quad val_I(V_2) = W \end{aligned}$$

Hieran orientiert sich nun das nachfolgende Tableauverfahren.

3.4.2 Kalkül

Definition 3.26 (Pfade)

In der nachfolgenden Definition betrachten wir Bäume, deren Knoten durch Vorzeichenformeln beschriftet sind. Ein **Pfad** in einem Baum ist eine maximale, linear geordnete Menge von Knoten. Wollen wir die Maximalität nicht verlangen, so sprechen wir von einem **Teilpfad**.

Bevor wir die formale Definition des Tableauealküls geben, betrachten wir Abb. 3.2 als Einstiegsbeispiel. Es soll bewiesen werden, daß $(P \vee (Q \wedge R)) \rightarrow ((P \vee Q) \wedge (P \vee R))$ eine Tautologie ist. Der Tableauealkül ist ein Widerspruchskalkül. Wir nehmen also an, daß $(P \vee (Q \wedge R)) \rightarrow ((P \vee Q) \wedge (P \vee R))$ keine Tautologie ist, also bei geeigneter Interpretation falsch werden kann. Diese Annahme deuten wir durch das Voranstellen der 0 an. Die betrachtete Implikation kann offensichtlich nur falsch sein, wenn die Prämisse, $P \vee (Q \wedge R)$, wahr und die Konklusion, $(P \vee Q) \wedge (P \vee R)$, falsch ist. Das notieren wir, indem wir die beiden Formeln unter die Anfangsannahme schreiben, mit dem Vorzeichen 1, bzw. 0, versehen. Die Formel $P \vee (Q \wedge R)$ ist wahr, wenn P oder $Q \wedge R$ wahr ist. Diese Alternative deuten wir im Tableau durch eine Verzweigung an, der linke Pfad führt zur Formel $1P$, der rechte zu $1(Q \wedge R)$. Wir führen zuerst die Argumentationskette auf dem linken Pfad weiter. Unter den Folgerungen, zu denen uns die Anfangsannahme bisher geführt hat, ist auch die Aussage, daß $(P \vee Q) \wedge (P \vee R)$ falsch ist. Eine Konjunktion ist falsch, wenn eines ihrer Konjunktionglieder falsch ist. Das führt also zu einer weiteren Verzweigung des linken Pfades mit den in der Abbildung angegebenen Markierungen. Die bisherigen Überlegungen haben auf dem linkesten Pfad zur Annahme geführt, daß die Formel $P \vee Q$ falsch ist, das bedeutet, daß P und Q falsch sein müssen. Alle Formeln auf diesem Ast sind jetzt abgearbeitet worden und auf Atome mit Vorzeichen reduziert worden. An dieser Stelle erinnern wir uns, daß unser Ansatz ein Widerspruchsbeweis ist. Wir erwarten, im positiven Fall, daß unsere Anfangsannahme zu Widersprüchen führt. Für den Pfad, den wir gerade betrachten, ist das schon der Fall. Er enthält die Formeln $1P$ und $0P$, die besagen, daß P wahr ist und P auch falsch ist. In entsprechender Weise müssen noch die übrigen offenen Alternativen in dem Tableau betrachtet werden. In dem Beispiel in Abbildung 3.2 erweisen sich alle Pfade als geschlossen. Damit ist die Anfangsannahme zu einem Widerspruch geführt worden.

Es ist wichtig zu bemerken, daß bei der Verlängerung eines Pfades nicht notwendigerweise nur die letzte Formel auf dem Pfad betrachtet wird.

Definition 3.27 (Tableaux)

Es sei M eine Menge von Formeln und A eine Formel. Ein (aussagenlogisches) **Tableau** für A über M ist ein Baum mit Beschriftung der Knoten durch markierte Formeln, gemäß der folgenden rekursiven Definition:

1. Ein einziger mit $0A$ markierter Knoten ist ein Tableau für A über M .
2. Es sei T ein Tableau für A über M und π ein Pfad in T , so daß auf π ein Knoten liegt, der mit einem V vom Typ α beschriftet ist. Dann ist auch T_1 ein Tableau für A über M , welches dadurch aus T entsteht,

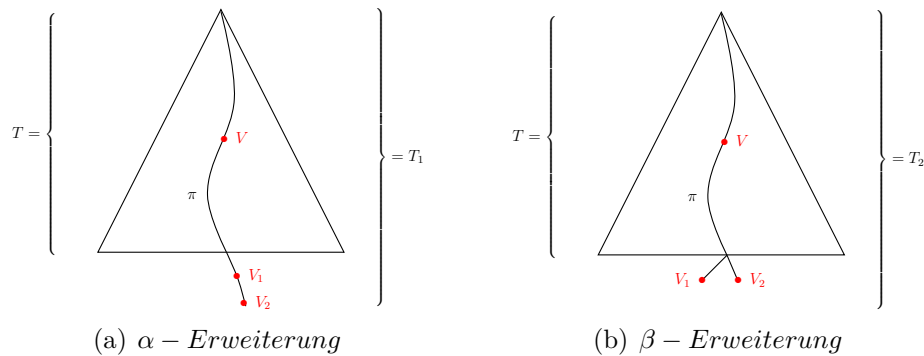


Abbildung 3.1: Tableau-Erweiterungen

daß an π zwei neue Knoten angefügt werden, welche mit V_1 und mit V_2 beschriftet sind, siehe Abbildung 3.1 (linke Seite). Wir sagen in diesem Fall, daß T_1 aus T durch eine Anwendung der α -**Regel** aus T hervorgeht.

3. Es sei T ein Tableau für A über M und π ein Pfad in T , so daß auf π ein Knoten liegt, der mit einem V vom Typ β beschriftet ist. Dann ist auch T_2 ein Tableau für A über M , welches dadurch entsteht, daß an π zwei verzweigende Kanten angefügt werden, welche zu neuen Blättern mit Beschriftungen V_1 bzw. V_2 führen, siehe Abbildung 3.1 (rechte Seite). Wir sagen, T_2 geht aus T durch eine Anwendung der β -**Regel** aus T hervor.
4. Es sei T ein Tableau für A über M und π ein Pfad in T , dann ist auch T_3 ein Tableau für A über M , welches durch Verlängerung von π um einen mit $1B$ beschrifteten Knoten entsteht, für eine Formel B aus M . Wir sagen, T_3 geht aus T durch eine Anwendung der **Voraussetzungsregel** aus T hervor.

Der Verzweigungsgrad eines Tableaubaaumes (die maximale Anzahl der Nachfolger eines Knotens) hängt von der benutzten Regeln ab. Die Regeln, die wir bisher kennen gelernt haben, führen zu Tableaubäumen mit maximalem Verzweigungsgrad 2. Wir werden in Unterabschnitt 3.4.3 ein allgemeineres Regelformat betrachten.

Definition 3.28 (Geschlossenes Tableau)

Ein Tableau T heißt **geschlossen**, wenn alle seine Pfade geschlossen sind. Ein Pfad π heißt **geschlossen**, wenn zwei Knoten auf π liegen, welche respektive

beschriftet sind mit $0A$ und $1A$, für eine beliebige Formel A . Auch ein Pfad, auf dem die Formel 01 oder 10 vorkommt zählt als geschlossen. Gelegentlich deuten wir die Abgeschlossenheit eines Pfades π an durch Anfügen einer Kante an π , welche zu einem mit $*$ beschrifteten Blatt führt.

Wir verwenden ggf. die folgende abkürzende Notation ("uniforme Notation"). Schreibt man kurz α für eine α -Formel, so heißen ihre Abkömmlinge α_1, α_2 . Entsprechend für β . Hiermit definieren wir

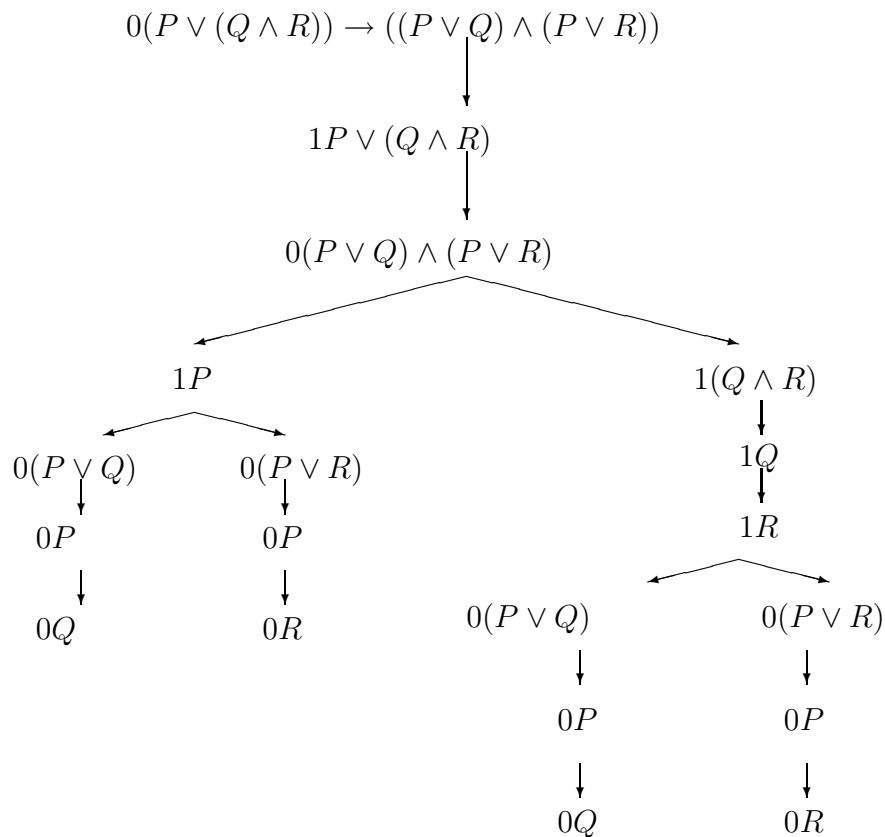


Abbildung 3.2: Beispiel eines Tableau-Beweises

Wir benutzen die folgende Kurzschreibweise für Tableauregeln

$$\frac{\alpha}{\alpha_1} \quad \frac{\beta}{\beta_1 \mid \beta_2} \\
 \alpha_2$$

Definition 3.29 (Ableitbarkeit)

Es sei $A \in For_{0\Sigma}$ und $M \subseteq For_{0\Sigma}$. A ist aus M ableitbar in $\mathbf{T0}$,

$$M \vdash_{\mathbf{T0}} A,$$

g.d.w. es ein geschlossenes Tableau für $0A$ über M gibt.

Man beachte, daß zur Definition der Ableitbarkeit einer Formel aus einer Formelmenge hinzugenommen sind:

- die *Initialisierung*: Erzeuge einen mit $0A$ beschrifteten Knoten;
- die V_M -*Regel* (Voraussetzungsregel bzg. M):
- Vergrößere ein Tableau durch Anhängen einer Kante von einem Blatt zu einem neuen, mit $1B$ für ein $B \in M$ beschrifteten Knoten.

Bemerkung 3.30

(Beziehung zum Begriff des abstrakten Kalküls)

Muss noch ergänzt werden

Definition 3.31 (Semantik von Tableaux)

Für eine Interpretation I der Atome haben wir bereits oben (3.24, 3.25) die Auswertung der Vorzeichenformeln definiert. Ist T ein Tableau und π ein Pfad in T , so setzen wir

$$\begin{aligned} val_I(\pi) = W & \quad :\Leftrightarrow \quad val_I(V) = W \text{ für alle } V \text{ auf } \pi \\ val_I(T) = W & \quad :\Leftrightarrow \quad \text{Es gibt einen Pfad } \pi \text{ in } T \text{ mit } val_I(\pi) = W \end{aligned}$$

Eine Interpretation I mit $val_I(T) = W$ heißt **Modell** von T .

Satz 3.32 (Korrektheit und Vollständigkeit)

Es seien $A \in For0_\Sigma$ und $M \subseteq For0_\Sigma$. Dann gilt

$$M \vdash_{T_0} A \quad \Leftrightarrow \quad M \models A.$$

Beweis der Korrektheit:

Aus der Semantik der Tableau folgt, daß ein geschlossenes Tableau kein Modell haben kann (Definitionen 3.28, 3.31). Wenn nun $M \not\models A$, d.h. $M \cup \{\neg A\}$ hat ein Modell, dann zeigen wir:

- Jedes Tableau für A über M hat ein Modell. Somit kann kein solches Tableau geschlossen sein.
- folgt aus dem

Lemma 3.33 (Korrektheitslemma)

Ist I Modell von $M \cup \{\neg A\}$, dann ist I Modell eines jeden Tableau für A über M .

Beweis:

Initialisierung: I ist Modell von $\neg A$, also von $0A$.

Im Induktionsschritt unterscheiden wir nach dem Typ der auf ein Tableau T angewandten Regel.

α -Fall:

Nach Induktionsannahme gilt $val_I(T) = W$, es gibt also einen Pfad π in T mit $val_I(\pi) = W$. Zur Anwendung der α -Regel wird ein Pfad π_1 in T und eine α -Formel V auf π_1 gewählt, π_1 wird verlängert um V_1, V_2 . Wenn $\pi_1 \neq \pi$, gilt für das neu entstehende Tableau T' trivialerweise $val_I(T') = W$. Wenn $\pi_1 = \pi$, haben wir aus $val_I(\pi) = W$, daß $val_I(V) = W$, also $val_I(V_1) = W$ und $val_I(V_2) = W$ nach Lemma 3.25. Somit $val_I(\pi_1) = W$ für den neu entstehenden Pfad π' , d.h. $val_I(T') = W$

β -Fall:

Entsprechend mit „oder“ statt „und“. (Es entstehen zwei Pfade, von denen mindestens einer I zum Modell haben muß.

V_M -Fall:

Nach Voraussetzung ist I Modell von M , also von jedem $1B$, $B \in M$.

■

Beweis der Vollständigkeit

Es gelte $M \models A$. Wegen der Kompaktheit der Aussagenlogik (Satz 3.15.5) gibt es dann ein endliches $E \subseteq M$ mit $E \models A$. Wir wollen zeigen, daß dann ein geschlossenes Tableau für A über E existiert. Dieses ist erst recht ein geschlossenes Tableau für A über M .

Wir zeigen etwas mehr. Definieren wir zunächst *erschöpfte* Tableaux als solche, die sich nur „auf trivial wiederholende Weise“ noch vergrößern lassen:

Definition 3.34 (Erschöpftes Tableau)

Das Tableau T für A über der endlichen Formelmengemenge E heißt **erschöpft**, wenn auf jedem offenen Pfad π von T jede α - und β -Formel benutzt wurde (d.h. die entsprechende Regel auf sie angewandt wurde), und jedes $1B$, $B \in E$, auf π vorkommt.

Offensichtlich gibt es mindestens ein erschöpftes Tableau für A über E , man muß nur in systematischer Weise auf jedem Pfad jede mögliche Regelanwendung vornehmen. Etwas genauer müssen wir uns dieses Verfahren schon ansehen, denn durch die Anwendung einer Regel erscheinen neue Formeln in dem Tableau, auf die eventuell wieder Regeln angewendet werden können. Eine Besonderheit des Tableauverfahrens liegt nun darin, daß alle während der Beweissuche auftretenden Formeln Teilformeln von A oder E

sind. Während der Konstruktion eines Tableau werden also keine neuen Formeln zusammengesetzt, synthetisiert, sondern es werden nur die vorhandenen Formeln analysiert. Das hat der hier präsentierten Methode den Namen *analytische Tableaux* eingetragen. Kommen also auf einem Pfad mehr Formeln vor als Teilformeln in der Ausgangsmenge, dann kann das nur daran liegen, daß eine Formel mehrfach auftritt. Ausgehend von A und E gibt es nicht nur ein erschöpftes Tableau, sondern abhängig von der Reihenfolge der Regelanwendungen mehrere. Durch sorgfältige Buchführung könnte man sich ein Verfahren ausdenken, das alle Variationsmöglichkeiten durchprobiert. Wie die nachfolgenden Ergebnisse zeigen, ist das jedoch nicht nötig. Wir zeigen (Lemma 3.35), daß für jedes von ihnen, T , gilt: Hat T einen offenen Pfad, so läßt sich aus ihm ein Modell von $E \cup \{\neg A\}$ ablesen. Insbesondere hat man dann:

$$\begin{aligned} E \not\vdash_{\mathbf{T0}} A &\Leftrightarrow \text{kein Tableau für } A \text{ über } E \text{ ist geschlossen} \\ &\Rightarrow \text{kein erschöpftes Tableau für } A \text{ über } E \text{ ist geschlossen} \\ &\Rightarrow E \cup \{\neg A\} \text{ hat ein Modell, d.h. } E \not\models A. \end{aligned}$$

Aus der Korrektheit von $\mathbf{T0}$ folgt auch noch: Entweder alle erschöpften Tableaux für A über E sind geschlossen, oder keins.

Lemma 3.35 (Vollständigkeitslemma)

Wenn es ein erschöpftes, nicht geschlossenes Tableau für A über E gibt, dann hat $E \cup \{\neg A\}$ ein Modell.

Beweis:

Sei π ein offener Pfad im erschöpften Tableau T für A über E . Wir schreiben $V \in \pi$, wenn V auf π liegt. Da T erschöpft ist, hat man

- Für jede α -Formel $V \in \pi$: $V_1 \in \pi$ und $V_2 \in \pi$
- für jede β -Formel $V \in \pi$: $V_1 \in \pi$ oder $V_2 \in \pi$
- Für jedes $B \in E$: $1B \in \pi$.

Insbesondere liegt für jedes in A oder einem $B \in E$ vorkommenden Atom P genau entweder $0P$ oder $1P$ auf π . Wir definieren

$$I(P) := \begin{cases} W & \text{falls } 1P \in \pi \\ F & \text{falls } 0P \in \pi \\ \text{beliebig} & \text{sonst} \end{cases}$$

Durch Induktion zeigt man leicht, daß $I(V) = W$ für jedes V auf π . Es folgt, daß I Modell von $E \cup \{\neg A\}$ ist. ■

3.4.3 Generierung von Tableauregeln

Woher kommen die Tableauregeln? Für die bisher betrachteten einfachen logischen Verknüpfungen waren die zugehörigen Regeln zu naheliegend, als daß man dieser Frage große Aufmerksamkeit geschenkt hätte. Die beiden Regeln für die Konjunktion

$$\frac{1(A \wedge B)}{\begin{array}{l} 1A \\ 1B \end{array}} \quad \frac{0(A \wedge B)}{0A \mid 0B}$$

bieten sich direkt an. Wie müsste aber eine Tableauregel für die Formel $1(A \leftrightarrow B)$ aussehen? oder für den Shannon-Operator? (siehe Definition 2.31). Bevor wir diese Frage beantworten, wollen wir das Format für Tableauregeln verallgemeinern. Statt einer Verzweigung in zwei Fortsetzungen wollen wir beliebige endliche Verzweigungen zulassen. Auch soll pro Fortsetzung erlaubt sein endliche viele Formeln anzugeben. In den bisherigen Beispielen wurden höchstens zwei Formeln pro Fortsetzung benötigt. Eine allgemeine Tableauregel hat also die Form

$$\frac{\phi}{\begin{array}{c|ccc} \psi_{1,1} & \dots & \psi_{n,1} \\ \vdots & \dots & \vdots \\ \vdots & \dots & \vdots \\ \psi_{1,k_1} & \dots & \psi_{n,k_n} \end{array}}$$

Um die Teilformeleigenschaft des Tableaurekalküls zu gewährleisten, wird gefordert, daß alle Vorzeichenformeln $\psi_{i,j}$ Teilformeln der Vorzeichenformel ϕ sind.

Eine genaue Inspektion des Vollständigkeits- und Korrektheitsbeweises für den Tableaurekalkül zeigt, daß allein die Aussagen von Lemma 3.25 notwendig waren. Die Verallgemeinerung der Aussage dieses Lemmas führt uns zu der Definition

Definition 3.36

Eine allgemeine Tableauregel

$$\frac{\phi}{\begin{array}{c|ccc} \psi_{1,1} & \dots & \psi_{n,1} \\ \vdots & \dots & \vdots \\ \vdots & \dots & \vdots \\ \psi_{1,k_1} & \dots & \psi_{n,k_n} \end{array}}$$

heißt vollständig und korrekt, wenn für jede Interpretation I gilt

$$\begin{aligned} \text{val}_I(\phi) = W \quad \text{gdw} \quad & \text{es gibt ein } i, 1 \leq i \leq n, \text{ so daß} \\ & \text{für alle } j, 1 \leq j \leq k_i \text{ gilt} \\ & \text{val}_I(\psi_{i,j}) = W \end{aligned}$$

Satz 3.37

Ein Tableaurekalkül der nur korrekte und vollständige Regeln benutzt ist selbst korrekt und vollständig.

Beweis siehe Literatur. ■

Beispiel 3.38

Wir schauen uns noch einmal die Wahrheitstabelle für den logischen Äquivalenzoperator an.

\leftrightarrow	1	0
1	1	0
0	0	1

Man prüft leicht nach, daß die Tableauregel

$$\frac{1(A \leftrightarrow B)}{1A \quad | \quad 0A}$$

$$1B \quad | \quad 0B$$

die Forderungen der Definition 3.36 erfüllt.

3.4.4 Übungsaufgaben

Übungsaufgabe 3.4.1

In Anlehnung an [DGHP99, pp.70] definieren wir

Definition 3.39

Ein Tableau T heißt *regulär*, wenn keine Vorzeichenformel auf demselben Pfad zweimal vorkommt.

Das Tableau aus Abb.3.2 ist ein Beispiel eines regulären Tableaux.

1. Geben Sie Formeln A, B an, so daß das es ein geschlossenes Tableau für A über B gibt, aber keine reguläres geschlossenes.
2. Wie könnte man das Tableauverfahren aus Definition 3.27 abändern, so daß nur reguläre Tableaux entstehen und Korrektheit und Vollständigkeit erhalten bleiben?

Übungsaufgabe 3.4.2

Geben Sie vollständige und korrekte Tableauregeln an für

1. die Äquivalenz \leftrightarrow ,
2. den dreistelligen Shannon-Operator sh .

Übungsaufgabe 3.4.3

Beweisen Sie im Tableau-Kalkül:

$$\begin{array}{l}
 (p1h1 \rightarrow \neg p2h1) \\
 \wedge (p1h1 \rightarrow \neg p3h1) \\
 \wedge (p2h1 \rightarrow \neg p3h1) \\
 \wedge (p1h2 \rightarrow \neg p2h2) \\
 \wedge (p1h2 \rightarrow \neg p3h2) \\
 \wedge (p2h2 \rightarrow \neg p3h2)
 \end{array}
 \quad \vdash \quad
 \begin{array}{l}
 (\neg p1h1 \wedge \neg p1h2) \\
 \vee (\neg p2h1 \wedge \neg p2h2) \\
 \vee (\neg p3h1 \wedge \neg p3h2)
 \end{array}$$

Übungsaufgabe 3.4.4

Beweisen Sie im Tableau-Kalkül:

$$((A \wedge B) \rightarrow C) \rightarrow (A \rightarrow (B \rightarrow C))$$

Übungsaufgabe 3.4.5

Beweisen Sie den β -Fall von Lemma 3.33.

Übungsaufgabe 3.4.6

Anstelle der β -Regel werde die folgende β^+ -Regel benutzt:

$$\frac{\beta}{\beta_1 \mid \begin{array}{l} \neg\beta_1 \\ \beta_2 \end{array}}$$

Zeigen Sie, daß der so entstehende Tableaunkalkül auch korrekt und vollständig ist.

Übungsaufgabe 3.4.7

Im Kapitel *Tableau Methods for Classical Propositional Calculus* in [DGHP99] von Marcello D'Agostino wird ein **KE** genannter Tableaunkalkül vorgestellt. Die α -Regeln sind dieselben wie in Abschnitt 3.4.2 beschrieben. Die uniforme β -Regel sieht dagegen so aus:

$$\frac{\beta_1^c}{\beta} \quad \frac{\beta_2^c}{\beta}$$

wobei V^c die zu V komplementäre Vorzeichenformel ist, d.h. $V^c = 0F$ für $V = 1F$ und $V^c = 1F$ für $V = 0F$. In unserem Kalkül aus Abschnitt 3.4.2 besaßen die Regeln nur eine Prämisse, mit Ausnahme der Pfadabschlussregel. Im **KE** Kalkül besitzen alle β -Regeln zwei Voraussetzungen. Die generische β -Regel umfasst die folgenden Instanzen.

$$\frac{0F_1}{1(F_1 \vee F_2)} \quad \frac{0F_2}{1(F_1 \vee F_2)} \quad \frac{1F_1}{1(F_1 \rightarrow F_2)} \quad \frac{0F_2}{1(F_1 \rightarrow F_2)}$$

$$\frac{1F_1}{0(F_1 \wedge F_2)} \quad \frac{1F_2}{0(F_1 \wedge F_2)}$$

Die bisher präsentierten Regeln sind nicht vollständig, sie würden auch nur die Konstruktion von Tableaux mit einem einzigen Pfad erlauben. Hinzu kommt noch die eingeschränkte Schnittregel

$$\frac{}{1F \mid 0F}$$

Die Einschränkung besteht darin, daß die Schnittregel zu Verzweigung eines Tableaufades π nur angewendet werden darf wenn es eine β -Formel β auf π gibt, so daß $\beta_1 = 1F$ oder $\beta_1 = 0F$. Die Pfadabschlussregel bleibt wie bisher.

Beweisen Sie

1. Korrektheit und
2. Vollständigkeit des **KE** Kalküls

3.5 Sequenzenkalküle

Sequenzenkalküle wurden um 1935 von G. GENTZEN eingeführt. Deswegen werden sie auch *Gentzenkalküle* genannt. Sie spielen eine zentrale Rolle in der mathematischen Logik, insbesondere in der Beweistheorie und haben in jüngster Zeit vor allem im automatischen Beweisen an Aktualität gewonnen. Für eine weitergehende Darstellung konsultiere man das Buch [Gal86].

3.5.1 Syntax und Semantik

Definition 3.40 (Sequenz)

Eine *Sequenz* wird notiert als eine Folge zweier endlicher Mengen aussagenlogischer Formeln getrennt durch das Symbol \rightarrow :

$$\Gamma \rightarrow \Delta.$$

Γ wird Antezedent und Δ Sukzedent genannt. Sowohl links wie rechts vom Sequenzenpfeil \rightarrow kann auch die leere Folge stehen. Wir schreiben dann

$$\rightarrow \Delta \text{ bzw. } \Gamma \rightarrow \text{ bzw. } \rightarrow .$$

Das Wort „Sequenz“ für eine Zeichenfolge der Form $\Gamma \rightarrow \Delta$ hat sich fest eingebürgert, obwohl man einwenden könnte, daß „Sequenz“ zu sehr den Eindruck einer Folge vermittelt, was ja gerade nicht beabsichtigt ist. Die englischen Übersetzung von „Sequenz“ heißt *sequent* und nicht etwa *sequence*.

Wir benutzen die folgenden abkürzenden Schreibweisen zur Einsparung der Vereinigungsoperation zwischen Mengen:

$$\begin{array}{lll} A_1, \dots, A_n & \text{für} & \{A_1, \dots, A_n\} \\ \Gamma, B & \text{für} & \Gamma \cup \{B\} \\ \Gamma, \Delta & \text{für} & \Gamma \cup \Delta \end{array}$$

Ist Γ eine endliche Folge von Formeln, so schreiben wir $\bigwedge \Gamma$ für die Konjunktion der Formeln in Γ und $\bigvee \Gamma$ für die Disjunktion. Für $\Gamma = \emptyset$ setzen wir dabei $\bigwedge \emptyset = \mathbf{1}$ und $\bigvee \emptyset = \mathbf{0}$.

Definition 3.41 (Auswertung von Sequenzen)

Sei I eine Interpretation der Atome. Wir setzen die Auswertung val_I von Formeln einer Auswertung der Sequenzen fort durch.

$$val_I(\Gamma \rightarrow \Delta) = val_I(\bigwedge \Gamma \rightarrow \bigvee \Delta)$$

Die Begriffe *Tautologie*, *Erfüllbarkeit* werden jetzt in naheliegender Weise auch für Sequenzen anstelle von einzelnen Formeln benutzt.

3.5.2 Kalkül

Definition 3.42 (Axiome und Regeln des Sequenzenkalküls S0)

axiom

$$\frac{}{\Gamma, F \rightarrow F, \Delta}$$

not-left

$$\frac{\Gamma, \rightarrow F, \Delta}{\Gamma, \neg F \rightarrow \Delta}$$

not-right

$$\frac{\Gamma, F \rightarrow \Delta}{\Gamma \rightarrow \neg F, \Delta}$$

impl-left

$$\frac{\Gamma \rightarrow F, \Delta \quad \Gamma, G \rightarrow \Delta}{\Gamma, F \rightarrow G \rightarrow \Delta}$$

impl-right

$$\frac{\Gamma, F \rightarrow G, \Delta}{\Gamma \rightarrow F \rightarrow G, \Delta}$$

and-left

$$\frac{\Gamma, F, G \rightarrow \Delta}{\Gamma, F \wedge G \rightarrow \Delta}$$

and-right

$$\frac{\Gamma \rightarrow F, \Delta \quad \Gamma \rightarrow G, \Delta}{\Gamma \rightarrow F \wedge G, \Delta}$$

or-left

$$\frac{\Gamma, F \rightarrow \Delta \quad \Gamma, G \rightarrow \Delta}{\Gamma, F \vee G \rightarrow \Delta}$$

or-right

$$\frac{\Gamma \rightarrow F, G, \Delta}{\Gamma \rightarrow F \vee G, \Delta}$$

Definition 3.43 (Beweisbaum)

Ein Ableitungsbaum ist ein Baum, dessen Knoten mit Sequenzen markiert sind und für jeden Knoten n die folgende Einschränkung erfüllt:

1. ist n_1 der einzige Nachfolgerknoten von n und sind $\Gamma \rightarrow \Delta$ und $\Gamma_1 \rightarrow \Delta_1$ die Markierungen von n und n_1 , dann gibt es eine Sequenzenregel

$$\frac{\Gamma_1 \rightarrow \Delta_1}{\Gamma \rightarrow \Delta}$$

2. besitzt n die beiden Nachfolgerknoten n_1 und n_2 und sind $\Gamma \rightarrow \Delta$, $\Gamma_1 \rightarrow \Delta_1$ und $\Gamma_2 \rightarrow \Delta_2$ die Sequenzen an den Knoten n , n_1 und n_2 dann gibt es eine Sequenzenregel

$$\frac{\Gamma_1 \rightarrow \Delta_1 \quad \Gamma_2 \rightarrow \Delta_2}{\Gamma \rightarrow \Delta}$$

Wir nennen einen Beweisbaum *geschlossen* oder *vollständig* wenn er zusätzlich noch die folgende Bedingung erfüllt:

3. jeder Knoten n , der keinen Nachfolgerknoten hat, muß mit einem Axiom markiert sein.

Ein Beispiel eines Beweisbaumes ist in Abbildung 3.3 zu sehen. Die Defini-

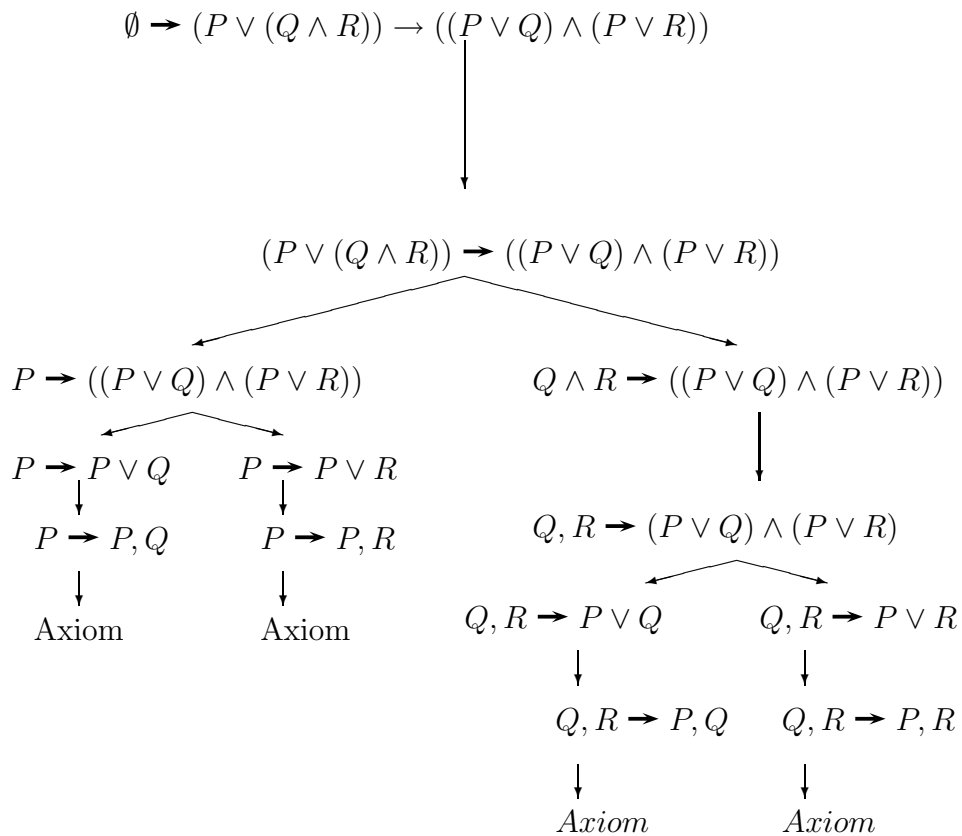


Abbildung 3.3: Beispiel eines Beweisbaums im Sequenzenkalkül

tion 3.43 eines Beweisbaumes legt nicht fest in welcher Reihenfolge man ihn konstruiert. Es ist jedoch in dem meisten Fällen zweckmäßig mit dem Wurzelknoten zu beginnen. Bei diesem Vorgehen werden die Regeln aus Definition 3.42 *von unten nach oben* benutzt.

Definition 3.44 (Ableitbarkeit in S0)

Für $A \in For0_\Sigma$, $M \subseteq For0_\Sigma$ sagen wir, daß A aus M ableitbar in **S0** ist, in Zeichen:

$$M \vdash_S A,$$

genau dann, wenn ein geschlossener Beweisbaum mit Wüzelmarkierung $M \rightarrow A$ in **S0** existiert.

Bemerkung 3.45

Für den seltenen, aber nicht auszuschließenden Fall, daß man mit einer unendlichen Menge M von Voraussetzungen arbeiten möchte ist das bisherige Vorgehen nicht geeignet. Man müsste dazu erklären, was die Konjunktion einer unendlichen Formelmenge sein soll. In diesem Fall führt man eine neue Beweisregel ein:

use premiss

$$\frac{B, \Gamma \rightarrow \Delta}{\Gamma \rightarrow \Delta}$$

für $B \in M$

Satz 3.46 (Korrektheit und Vollständigkeit von S0)

Für jede endliche Menge aussagenlogischer Formeln M und jede aussagenlogische Formel A gilt:

$$M \models A \quad \Leftrightarrow \quad M \vdash_{\mathbf{S0}} A.$$

Beweis:

Korrektheit Wir gehen aus von einem geschlossenen Beweisbaums \mathcal{T} mit Wurzelmarkierung $M \rightarrow A$ und müssen zeigen, daß die Sequenz $M \rightarrow A$ allgemeingültig ist. Dazu genügt es zu zeigen

1. jede atomare Sequenz $\Gamma, A \rightarrow \Delta, A$ ist allgemeingültig und
2. für jede Beweisregel

$$\frac{\Gamma_1 \rightarrow \Delta_1 \quad \Gamma_2 \rightarrow \Delta_2}{\Gamma \rightarrow \Delta}$$

gilt wenn $\Gamma_1 \rightarrow \Delta_1$ und $\Gamma_2 \rightarrow \Delta_2$ allgemeingültig sind, dann ist auch $\Gamma \rightarrow \Delta$ allgemeingültig.

Teil 1 ist unmittelbar einsichtig.

Zu Teil 2 führen wir zwei Beispiele vor und vertrauen darauf, daß der Leser danach weiss, wie er die restlichen Fälle selbst nachrechnen kann.

or-left

$$\frac{\Gamma, H \rightarrow \Delta \quad \Gamma, G \rightarrow \Delta}{\Gamma, H \vee G \rightarrow \Delta}$$

Wir haben als Voraussetzung, daß $\bigwedge \Gamma \wedge H \rightarrow \bigvee \Delta$ und $\bigwedge \Gamma \wedge G \rightarrow \bigvee \Delta$ allgemeingültig sind und sollen zeigen, daß das auch für $\bigwedge \Gamma \wedge (H \vee G) \rightarrow \bigvee \Delta$ zutrifft. Für jede Belegung I müssen wir also $val_I(\bigwedge \Gamma \wedge (H \vee G) \rightarrow \bigvee \Delta) = \mathbf{W}$ zeigen. Gilt $val_I(\bigwedge \Gamma \wedge (H \vee G)) = \mathbf{F}$, dann sind wir trivialerweise fertig. Gilt $val_I(\bigwedge \Gamma \wedge (H \vee G)) = \mathbf{W}$, dann gilt $val_I(\bigwedge \Gamma \wedge H) = \mathbf{W}$, oder $val_I(\bigwedge \Gamma \wedge G) = \mathbf{W}$. Indem wir von der ersten bzw. der zweiten Voraussetzung Gebrauch machen folgt $val_I(\bigvee \Delta) = \mathbf{W}$ und wir sind fertig.

impl-right

$$\frac{\Gamma, H \rightarrow G, \Delta}{\Gamma \rightarrow H \rightarrow G, \Delta}$$

Wir sollten an dieser Stelle deutlich machen, daß in der Beweisverpflichtung 2 nur die allgemeinere Regelform hingeschrieben wurde. Natürlich muß der Nachweis auch für Regeln mit nur einer Prämisse geführt werden.

Als Voraussetzung haben wir die Allgemeingültigkeit von $\bigwedge \Gamma \wedge H \rightarrow G \vee \bigvee \Delta$ und sollen für jede Interpretation I zeigen $val_I(\bigwedge \Gamma \rightarrow (H \rightarrow G \vee \bigvee \Delta)) = \mathbf{W}$. Nehmen wir also $val_I(\bigwedge \Gamma) = \mathbf{W}$ an (der Fall $val_I(\bigwedge \Gamma) = \mathbf{F}$ führt wieder trivialerweise zum Ziel) und versuchen $val_I(H \rightarrow G \vee \bigvee \Delta) = \mathbf{W}$ zu zeigen. Wir brauchen nur den Fall $val_I(\bigvee \Delta) = \mathbf{F}$ weiter zu verfolgen, im Fall $val_I(\bigvee \Delta) = \mathbf{W}$ sind wir wieder sofort fertig. Wir kommen zur letzten Fallunterscheidung. Gilt $val_I(H) = \mathbf{F}$, dann folgt trivialerweise $val_I(H \rightarrow G) = \mathbf{W}$ und wir haben es wieder geschafft. Es bleibt der Fall $val_I(H) = \mathbf{W}$. Dann ergibt sich aber zusammengenommen $val_I(\bigwedge \Gamma \wedge H) = \mathbf{W}$. Jetzt folgt mit der Voraussetzung $val_I(G \vee \bigvee \Delta) = \mathbf{W}$. Da wir aber im Laufe unserer Fallunterscheidungen uns im Fall $val_I(\bigvee \Delta) = \mathbf{F}$ befinden, muß $val_I(G) = \mathbf{W}$ gelten. Insgesamt haben wir damit $val_I(H \rightarrow G) = \mathbf{W}$ erhalten, wie gewünscht.

Vollständigkeit Angenommen es gibt keinen geschlossenen Beweisbaum mit Wurzelmarkierung $\Gamma \rightarrow \Delta$. Wir wollen zeigen, daß $\Gamma \rightarrow \Delta$ nicht allgemeingültig ist. Sei \mathcal{T} ein Beweisbaum mit Wurzelmarkierung $\Gamma \rightarrow \Delta$ auf den keine weitere Regelanwendung mehr möglich ist. Nach Annahme kann \mathcal{T}

nicht geschlossen sein. Es gibt also einen Knoten n ohne Nachfolgerknoten der mit $A_1, \dots, A_n \rightarrow B_1, \dots, B_k$ markiert ist mit $\{A_1, \dots, A_n\} \cap \{B_1, \dots, B_k\} = \emptyset$. Die Sequenz $A_1, \dots, A_n \rightarrow B_1, \dots, B_k$ ist nicht allgemeingültig, denn für die Interpretation I mit $I(A_i) = \mathbf{W}$ für alle $1 \leq i \leq n$ und $I(B_j) = \mathbf{F}$ für alle $1 \leq j \leq k$ gilt $val_I(A_1, \dots, A_n \rightarrow B_1, \dots, B_k) = \mathbf{F}$. Im Korrektheitsteil dieses Beweises haben wir gezeigt, daß für jede Regel

$$\frac{\Gamma_1 \rightarrow \Delta_1 \quad \Gamma_2 \rightarrow \Delta_2}{\Gamma \rightarrow \Delta}$$

gilt: wenn die beiden Prämissen (oder die eine, wenn es nur eine gibt) allgemeingültig sind dann ist auch die Konklusion allgemeingültig. Umformuliert heißt das, Wenn $\Gamma \rightarrow \Delta$ nicht allgemeingültig ist, dann ist $\Gamma_1 \rightarrow \Delta_1$ oder $\Gamma_2 \rightarrow \Delta_2$ nicht allgemeingültig. In dem Beweisbaum \mathcal{T} haben wir schon einen Blattknoten gefunden, der mit einer nicht allgemeingültigen Sequenz markiert ist. Wiederholte Anwendung der genannten Eigenschaft der Beweisregeln führt zur Aussage, daß die Markierung des Wurzelknotens nicht allgemeingültig ist. ■

3.5.3 Übungsaufgaben

Übungsaufgabe 3.5.1

Beweisen Sie in S_0 :

$$((A \wedge B) \rightarrow C) \rightarrow (A \rightarrow (B \rightarrow C))$$

Übungsaufgabe 3.5.2

Zeigen Sie die folgenden Äquivalenzen

A ist eine allgemeingültige Formel	gdw	$\rightarrow A$ ist eine allgemeingültige Sequenz
$\neg A$ ist eine allgemeingültige Formel	gdw	$A \rightarrow$ ist eine allgemeingültige Sequenz
A ist eine erfüllbare Formel	gdw	$\rightarrow A$ ist erfüllbare Sequenz
$\neg A$ ist eine erfüllbare Formel	gdw	$A \rightarrow$ ist erfüllbare Sequenz

3.6 Sonstige Kalküle

Die **1-Resolution (unit resolution)** benutzt dieselbe Notation wie im Resolutionskalkül.

Die 1-Resolutionsregel ist ein Spezialfall der allgemeinen Resolutionsregel:

$$\frac{\{P\}, C_2 \cup \{\neg P\}}{C_2} \quad \frac{\{\neg P\}, C_2 \cup \{P\}}{C_2}$$

Der 1-Resolutionskalkül ist nicht vollständig. Die Klauselmenge

$$E = \{\{P_1, P_2\}, \{P_1, \neg P_2\}, \{\neg P_1, P_2\}, \{\neg P_1, \neg P_2\}\}$$

ist nicht erfüllbar, aber mit 1-Resolution ist aus E nichts ableitbar, also auch nicht \emptyset .

Das im folgenden beschriebene **Davis-Putnam-Loveland Verfahren** ist das zur Zeit schnellste Verfahren um die Erfüllbarkeit aussagenlogischer Formeln festzustellen. Bei einem Wettbewerb, der 1991/92 an der Universität Paderborn durchgeführt wurde und an dem 36 Teilnehmer aus verschiedenen Ländern teilnahmen, wurden die ersten sechs Plätze von Programmen belegt, die nach dem Davis-Putnam-Verfahren oder Varianten davon arbeiteten, siehe [BB92].

S eine Menge von Klauseln.

1. Programm widerlege(S):
2. falls $S = \emptyset$, Ende (S ist erfüllbar).
3. falls S keine Einerklausel enthält, wähle eine Variable P ;
widerlege(S_P) ; widerlege($S_{\neg P}$).
4. sonst wähle eine Einerklausel $K \in S$
5. $S = \text{reduziere}(K, S)$
 - Lasse alle Klauseln weg, die K als Teilklausel enthalten,
 - Lasse in allen übrigen Klauseln das zu K komplementäre Literal weg.
6. falls $\square \in S$, Ende (S widersprüchlich),
sonst widerlege(S).

S_P , bzw. $S_{\neg P}$ entsteht, indem P bzw. $\neg P$ zu S hinzugefügt wird.

Beispiel 3.47 (Klauselmengenaus [Hoo88])

$$\begin{aligned}
 &P_1 \vee P_2 \vee P_3 \\
 &\neg P_1 \vee P_2 \vee \neg P_4 \\
 &\neg P_1 \vee P_3 \\
 &\neg P_1 \vee \neg P_3 \vee P_4 \\
 &P_1 \vee \neg P_3 \\
 &\neg P_2
 \end{aligned}$$

Wir zeigen anhand von Beispiel 3.47 die Arbeitsweise des Verfahrens von Davis-Putnam-Loveland.

Beim ersten Aufruf von $\text{widerlege}(S)$ wird das Unterprogramm $\text{reduziere}(\neg P_2, S)$ aufgerufen und liefert S_1 :

$$\begin{aligned}
 &P_1 \vee P_3 \\
 &\neg P_1 \vee \neg P_4 \\
 &\neg P_1 \vee P_3 \\
 &\neg P_1 \vee \neg P_3 \vee P_4 \\
 &P_1 \vee \neg P_3
 \end{aligned}$$

Da S_1 keine Einerklausel enthält wird eine Variable, in diesem Fall sei das P_1 gewählt und $\text{widerlege}(S_{1,0})$ und $\text{widerlege}(S_{1,1})$ aufgerufen mit:

$ \begin{aligned} S_{1,0} : \\ &P_1 \vee P_3 \\ &\neg P_1 \vee \neg P_4 \\ &\neg P_1 \vee P_3 \\ &\neg P_1 \vee \neg P_3 \vee P_4 \\ &P_1 \vee \neg P_3 \\ &P_1 \end{aligned} $	$ \begin{aligned} S_{1,1} : \\ &P_1 \vee P_3 \\ &\neg P_1 \vee \neg P_4 \\ &\neg P_1 \vee P_3 \\ &\neg P_1 \vee \neg P_3 \vee P_4 \\ &P_1 \vee \neg P_3 \\ &\neg P_1 \end{aligned} $
---	--

Betrachten wir zuerst die Abarbeitung von $\text{widerlege}(S_{1,0})$. Es wird $\text{reduziere}(P_1, S_{1,0})$ aufgerufen resultierend in $S_{2,0}$:

$$\begin{aligned}
 &\neg P_4 \\
 &P_3 \\
 &\neg P_3 \vee P_4
 \end{aligned}$$

Als nächstes wird $\text{reduziere}(P_3, S_{2,0})$ aufgerufen und liefert:

$$\begin{aligned}
 &\neg P_4 \\
 &P_4
 \end{aligned}$$

woraus im nächsten Schritt die Unerfüllbarkeit von $S_{1,0}$ folgt.

Wenden wir uns jetzt der Abarbeitung von $\text{widerlege}(S_{1,1})$ zu. Hier liefert $\text{reduziere}(\neg P_1, S_{1,1})$:

$$\begin{aligned}
 &P_3 \\
 &\neg P_3
 \end{aligned}$$

woraus schon im nächsten Schritt die Unerfüllbarkeit von $S_{1,1}$, und damit

insgesamt die Unerfüllbarkeit von S , folgt.

Numerische Verfahren

Gegeben: eine KNF $A = D_1 \wedge \dots \wedge D_k$

U_i entstehe aus D_i , indem P_j ersetzt wird durch X_j , $\neg P_j$ durch $(1 - X_j)$ und \vee durch $+$.

$U(A)$ ist die Menge der Ungleichungen

$$U_i \geq 1 \text{ für alle } i$$

und

$$0 \leq X_j \leq 1 \text{ für alle } j$$

Satz 3.48

A ist erfüllbar

gdw

$U(A)$ in den ganzen Zahlen lösbar ist.

Beispiel 3.49

Für

$$E = (P_1 \vee P_2) \wedge (P_1 \vee \neg P_2) \wedge (\neg P_1 \vee P_2) \wedge (\neg P_1 \vee \neg P_2)$$

ergibt sich $U(E)$:

$$\begin{array}{ll} X_1 + X_2 \geq 1 & X_1 + (1 - X_2) \geq 1 \\ (1 - X_1) + X_2 \geq 1 & (1 - X_1) + (1 - X_2) \geq 1 \\ 0 \leq X_1 \leq 1 & 0 \leq X_2 \leq 1 \end{array}$$

Vereinfacht:

$$\begin{array}{ll} X_1 + X_2 \geq 1 & X_1 - X_2 \geq 0 \\ X_2 - X_1 \geq 0 & X_1 + X_2 \leq 1 \\ 0 \leq X_1 \leq 1 & 0 \leq X_2 \leq 1 \end{array}$$

Vereinfacht

$$\begin{array}{ll} X_1 = X_2 & X_1 + X_2 = 1 \\ 0 \leq X_1 \leq 1 & 0 \leq X_2 \leq 1 \end{array}$$

Unlösbar mit ganzen Zahlen.

Die Gleichungen

$$\begin{array}{ll} X_1 = X_2 & X_1 + X_2 = 1 \\ 0 \leq X_1 \leq 1 & 0 \leq X_2 \leq 1 \end{array}$$

sind allerdings lösbar für rationale Zahlen.

Satz 3.50

[BJL88] $U(S)$ besitzt keine rationale Lösung
 gdw
 aus S ist mit 1-Resolution \square herleitbar.

Beweis: Sei S eine Menge von Klauseln, aus der mit 1-Resolution die leere Klausel \square nicht herleitbar ist. Dann gibt es eine konsistente Menge ML von Literalen und eine nicht leere Menge von Klauseln S_0 ohne Einerklauseln, so daß S erfüllbar ist genau dann, wenn $ML \cup S_0$ erfüllbar ist. Ordnet man den positiven Literalen in ML den Wert 1, den negativen Literalen in ML den Wert 0 und allen anderen Atomen den Wert $\frac{1}{2}$ zu, so sind alle Gleichungen in $U(ML \cup S_0)$ erfüllt und damit auch alle Gleichungen in dem ursprünglichen $U(S)$.

Kann man aus S mit 1-Resolution die leere Klausel herleiten und ist ML die Menge der Literale, die dabei als Einerklauseln auftreten, dann ist eine Belegung b der Variablen mit rationalen Zahlen eine Lösung für $U(S)$ genau dann, wenn b eine Lösung für $U(ML)$ ist. $U(ML)$ enthält aber nur Gleichungen der Form $x = 1$ oder $x = 0$, so daß jede Lösung eine ganzzahlige Lösung ist. Da aus ML außerdem \square ableitbar ist, existiert auch keine ganzzahlige Lösung.

Als ein weiteres Beispiel betrachten wir die Übersetzung in Gleichungen über den ganzen Zahlen der Klauselmenge aus 3.47

Beispiel 3.51

Gleichungssystem aus [Hoo88]

$$\begin{array}{rcll}
 X_1 + X_2 + X_3 & & & \geq 1 \\
 -X_1 + X_2 & & - X_4 & \geq -1 \\
 -X_1 + & & X_3 & \geq 0 \\
 -X_1 & & - X_3 + X_4 & \geq -1 \\
 X_1 & & - X_3 & \geq 0 \\
 & - X_2 & & \geq 0 \\
 0 \leq X_1, X_2, X_3, X_4 \leq 1
 \end{array}$$

Aus der letzten Ungleichung folgt sofort $X_2 = 0$. Aus der dritten und vorletzten Ungleichung folgt $X_1 = X_3$. Eingesetzt in das Gleichungssystem ergibt sich:

$$\begin{array}{rcll}
 2X_1 & & & \geq 1 \\
 -X_1 & - & X_4 & \geq -1 \\
 -2X_1 + X_4 & & & \geq -1 \\
 0 \leq X_1, X_4 \leq 1
 \end{array}$$

Aus der ersten Ungleichung folgt $X_1 = 1$ Eingesetzt in die beiden folgenden Ungleichung ergibt sich

$$\begin{array}{l} - X_4 \geq 0 \\ + X_4 \geq 1 \\ \quad 0 \leq X_4 \leq 1 \end{array}$$

Dieses System ist sicherlich unerfüllbar.

3.7 Anwendungen der Aussagenlogik

3.7.1 Beschreibung von Schaltkreisen

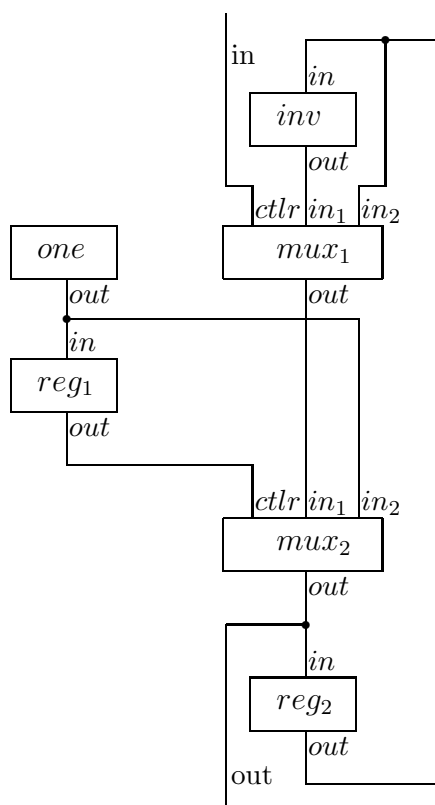


Abbildung 3.4: Schaltkreis für einen parity checker

Der in Abbildung 3.4 gezeigte Schaltkreis stammt aus der Arbeit [Gor88]. Er arbeitet, indem in reg_2 der Wert „1“ gespeichert wird, falls in der Folge der über in bisher empfangenen Eingabefolge eine gerade Anzahl von Einsen enthalten war, anderenfalls wird in reg_2 „0“ gespeichert. one ist ein Modul, das konstant den Wert „1“ liefert. reg_1 wird nur benötigt, um auch bei der Eingabefolge der Länge 0 den korrekten Wert „1“ in reg_2 zu erzeugen. Die beiden Register reg_0 und reg_1 haben den Initialwert 0.

Wir geben eine Beschreibung des Schaltkreises aus Abbildung 3.4 durch aussagenlogische Formeln. Genauer gesagt beschreiben die aussagenlogischen Formeln nur die Zusammenhänge zwischen den verschiedenen Ein- und Ausgängen der Schaltelemente zu einem festen Zeitpunkt. Das Verhalten der Schaltung über mehrere Zeittakte hinweg oder gar über beliebig viele Zeittakte hinweg

übersteigt bei weitem die Ausdruckstärke der Aussagenlogik.

$$\begin{aligned}
out &\leftrightarrow mux_2.out \\
mux_2.out &\leftrightarrow (reg_1.out \wedge mux_1.out) \vee (\neg reg_1.out \wedge one.out) \\
one.out &\leftrightarrow 1 \\
mux_1.out &\leftrightarrow (in \wedge inv.out) \vee (\neg in \wedge reg_2.out) \\
inv.out &\leftrightarrow \neg reg_2.out
\end{aligned}$$

Haben zu einem Zeitpunkt t beide Register reg_1 und reg_2 den Wert „1“, dann vereinfacht sich diese Beschreibung zu:

$$\begin{aligned}
out &\leftrightarrow mux_2.out \\
mux_2.out &\leftrightarrow mux_1.out \\
mux_1.out &\leftrightarrow (in \wedge inv.out) \vee \neg in \\
inv.out &\leftrightarrow 0
\end{aligned}$$

und weiter:

$$\begin{aligned}
out &\leftrightarrow mux_2.out \\
mux_2.out &\leftrightarrow mux_1.out \\
mux_1.out &\leftrightarrow \neg in
\end{aligned}$$

i.e.

$$out \leftrightarrow \neg in$$

Ist $in = 1$, dann wechselt zum Zeitpunkt $t + 1$ der Wert von reg_2 zu „0“. Ist $in = 0$, dann bleibt der Wert von reg_2 zum Zeitpunkt $t + 1$ unverändert.

Hat reg_1 zum Zeitpunkt t den Wert „1“ und reg_2 den Wert „0“ dann vereinfacht sich die Schaltungsbeschreibung zu:

$$\begin{aligned}
out &\leftrightarrow mux_2.out \\
mux_2.out &\leftrightarrow mux_1.out \\
mux_1.out &\leftrightarrow in \wedge inv.out \\
inv.out &\leftrightarrow 1
\end{aligned}$$

und weiter zu:

$$out \leftrightarrow in$$

3.7.2 Wissensrepräsentation

Die Ausdruckstärke der Aussagenlogik ist für viele Aufgaben der Repräsentation von Wissen zu gering. Manchmal lohnt sich jedoch die Mühe über eine aussagenlogische Formulierung nachzudenken. Wir demonstrieren das hier anhand einer Logelei von LEWIS CARROL, dem Autor von *Alice in Wonderland*. Das Beispiel ist dem Buch [Car58] entnommen.

The Lion and the Unicorn

When Alice entered the forest of forgetfulness, she did not forget everything, only certain things. She often forgot her name, and the most likely thing for her to forget was the day of the week. Now, the lion and the unicorn were frequent visitors to this forest. These two are strange creatures. The lion lies on Mondays, Tuesdays and Wednesdays and tells the truth on the other days of the week. The unicorn, on the other hand, lies on Thursdays, Fridays and Saturdays, but tells the truth on all the other days of the week.

One day Alice met the lion and the unicorn resting under a tree. They made the following statements:

lion : Yesterday was one of my lying days.
unicorn: Yesterday was one of my lying days.

From these statements, Alice, who was a bright girl, was able to deduce the day of the week. What was it?

Wir benutzen die folgenden Aussagenvariablen:

$Mo, Di, Mi, Do, Fr, Sa, So$
 $GMo, GDi, GMi, GDo, GFr, GSa, GSo$
 LW
 EW

mit der intendierten Bedeutung

X ist wahr, gdw heute Xtag ist

GX ist wahr, gdw gestern Xtag war

LW ist wahr, gdw der Löwe die Wahrheit sagt.

EW ist wahr, gdw das Einhorn die Wahrheit sagt.

Die aussagenlogische Formulierung des Rätsels vom Löwen und dem Einhorn läßt sich damit aufschreiben:

$$\begin{aligned} Mo &\leftrightarrow \neg(Di \vee Mi \vee Do \vee Fr \vee Sa \vee So) \\ Di &\leftrightarrow \neg(Mo \vee Mi \vee Do \vee Fr \vee Sa \vee So) \\ Mi &\leftrightarrow \neg(Mo \vee Di \vee Do \vee Fr \vee Sa \vee So) \\ Do &\leftrightarrow \neg(Mo \vee Di \vee Mi \vee Fr \vee Sa \vee So) \\ Fr &\leftrightarrow \neg(Mo \vee Di \vee Mi \vee Do \vee Sa \vee So) \\ Sa &\leftrightarrow \neg(Mo \vee Di \vee Mi \vee Do \vee Fr \vee So) \\ So &\leftrightarrow \neg(Mo \vee Di \vee Mi \vee Do \vee Fr \vee Sa) \end{aligned}$$

$$\begin{aligned}
GMo &\leftrightarrow Di \\
GDi &\leftrightarrow Mi \\
GMi &\leftrightarrow Do \\
GDo &\leftrightarrow Fr \\
GFr &\leftrightarrow Sa \\
GSa &\leftrightarrow So \\
GSo &\leftrightarrow Mo \\
\neg LW &\leftrightarrow Mo \vee Di \vee Mi \\
\neg EW &\leftrightarrow Do \vee Fr \vee Sa \\
LW &\leftrightarrow GMo \vee GDi \vee GMi \\
EW &\leftrightarrow GDo \vee GFr \vee GSa
\end{aligned}$$

Die Lösung ist „Donnerstag“.

3.7.3 Übungsaufgaben

Übungsaufgabe 3.7.1

Gegeben sei eine Landkarte mit L Ländern. Der Einfachheit halber seien die Länder mit den Zahlen von 0 bis $L - 1$ bezeichnet. Die binäre Relation $Na(i, j)$ trifft zu auf zwei Länder i, j wenn sie benachbart sind. Die Landkarte soll mit den Farben *rot*, *blau* und *grün* so eingefärbt werden, daß keine zwei benachbarten Länder dieselbe Farbe erhalten.

Finde Sie eine Menge aussagenlogischer Formeln FF , so daß FF erfüllbar ist, genau dann wenn eine Färbung der geforderten Art möglich ist. Aus eine erfüllenden Belegung für FF , soll außerdem eine korrekte Färbung der Landkarte ablesbar sein.

Kapitel 4

Prädikatenlogik erster Ordnung: Syntax und Semantik

4.1 Einführende Beispiele

4.1.1 Alltagslogik

Auch nur ein klein wenig interessantere Schlüsse der „Alltagslogik“ lassen sich nicht mehr in der Aussagenlogik wiedergeben. Im klassischen Beispiel

Alle Menschen sind sterblich.
Sokrates ist ein Mensch.
Also ist Sokrates sterblich.

ist der Schluß nicht mehr auf Grund von Bestandteilen der Sätze durchgeführt, die selbst wieder Sätze wären. Wir müssen eine feinere Aufteilung vornehmen.

In der Sprache der Prädikatenlogik (erster Ordnung) könnten wir die obigen Sätze schreiben:

$\forall x (\text{Mensch}(x) \rightarrow \text{sterblich}(x))$
 $\text{Mensch}(\text{Sokrates})$
 $\text{sterblich}(\text{Sokrates})$

„ \forall “ lesen wir „für alle“, und es ist mit Hilfe der *Variablen* x das umgangssprachliche

Alle Menschen sind sterblich

ausgedrückt in der Weise

Für alle x : Wenn x Mensch ist, dann ist x sterblich.

Genauer zur Syntax sagen wir sofort. Haben wir nun ein logisches Axiom

$\forall x (\text{Mensch}(x) \rightarrow \text{sterblich}(x)) \rightarrow (\text{Mensch}(\text{Sokrates}) \rightarrow \text{sterblich}(\text{Sokrates}))$

zur Verfügung, so läßt sich aus

$\{\forall x (\text{Mensch}(x) \rightarrow \text{sterblich}(x)), \text{Mensch}(\text{Sokrates})\}$

mittels Modus Ponens ableiten

$\text{sterblich}(\text{Sokrates})$.

4.1.2 Spezifikationen im Java Card API

Die Java Card Platform Specification v2.2.1 (siehe <http://java.sun.com/products/javacard/specs.html>) enthält unter vielen anderen die Klasse `Util`.

```
public class Util
    extends Object
```

In der Klasse kommt, wieder neben einigen anderen, die Methode `arrayCompare` vor, die zunächst übersichtsmäßig in einer Tabelle erklärt wird.

Method Summary	
static byte	<code>arrayCompare</code> (byte[] src, short srcOff, byte[] dest, short destOff, short length) Compares an array from the specified source array, beginning at the specified position, with the specified position of the destination array from left to right.

Darauf folgt eine detaillierte Erklärung.

Method Detail `arrayCompare`

```
public static final byte arrayCompare(byte [] src ,
                                       short srcOff ,
                                       byte [] dest ,
                                       short destOff ,
                                       short length) throws
    ArrayIndexOutOfBoundsException ,
    NullPointerException
```

Compares an array from the specified source array, beginning at the specified position, with the specified position of the destination array from left to right. Returns the ternary result of the comparison : less than(-1), equal(0) or greater than(1).

Note:

- If `srcOff` or `destOff` or `length` parameter is negative an `ArrayIndexOutOfBoundsException` exception is thrown.
- If `srcOff+length` is greater than `src.length`, the length of the `src` array a `ArrayIndexOutOfBoundsException` exception is thrown.
- If `destOff+length` is greater than `dest.length`, the length of the `dest` array an `ArrayIndexOutOfBoundsException` exception is thrown.

- If `src` or `dest` parameter is null a `NullPointerException` exception is thrown.

Parameters:

`src` - source byte array
`srcOff` - offset within source byte array to start compare
`dest` - destination byte array
`destOff` - offset within destination byte array to start compare
`length` - byte length to be compared

Returns: the result of the comparison as follows:

- 0 if identical
- -1 if the first miscomparing byte in source array is less than that in destination array
- 1 if the first miscomparing byte in source array is greater than that in destination array

Throws:

`ArrayIndexOutOfBoundsException` - if comparing all bytes would cause access of data outside array bounds
`NullPointerException` - if either `src` or `dest` is null

Wir wollen versuchen diese natürlich-sprachlichen Aussagen in einer formalen Sprache wiederzugeben. Wir formulieren dazu eine Nachbedingung und verlangen, daß jeder Aufruf der Methode `arrayCompare` terminiert und nach Beendigung die Formel

$$(\phi_0 \rightarrow \phi_1) \wedge \phi_2$$

wahr ist, wobei

Abkürzungen

<i>s</i>	für	<i>src</i>	source
<i>d</i>	für	<i>dest</i>	destination
<i>sO</i>	für	<i>srcOff</i>	source offset
<i>dO</i>	für	<i>destOff</i>	destination offset
<i>l</i>	für	<i>length</i>	Länge
<i>E</i>	für	<i>java :: lang :: Exception</i>	
<i>NPE</i>	für	<i>java :: lang :: NullPointerException</i>	
<i>OBE</i>	für	<i>java :: lang :: ArrayIndexOutOfBoundsException</i>	

$$\begin{aligned}
\phi_0 &\equiv s \neq \text{null} && \wedge \\
& sO \geq 0 && \wedge \\
& sO + l \leq \text{size}(s) && \wedge \\
& d \neq \text{null} && \wedge \\
& dO \geq 0 && \wedge \\
& dO + l \leq \text{size}(d) && \wedge \\
& l \geq 0 \\
\phi_1 &\equiv \neg \text{excThrown}(E) && \wedge \\
& (\text{result} = -1 \vee \text{result} = 0 \vee \text{result} = 1) && \wedge \\
& (\text{subSeq}(s, sO, sO + l) = \text{subSeq}(d, dO + 1, dO + l) \rightarrow \text{result} = 0) && \wedge \\
& (\exists i : \text{Int}(i \leq l \wedge (\text{at}(s, sO + i) < \text{at}(d, dO + i) && \wedge \\
& \forall j : \text{Int}(1 \leq j < i \rightarrow \text{at}(s, sO + j) = \text{at}(d, dO + j)))) \rightarrow \text{result} = -1) && \wedge \\
& (\exists i : \text{Int}(i \leq l \wedge (\text{at}(s, sO + i) > \text{at}(d, dO + i) && \wedge \\
& \forall j : \text{Int}(1 \leq j < i \rightarrow \text{at}(s, sO + j) = \text{at}(d, dO + j)))) \rightarrow \text{result} = 1) \\
\phi_2 &\equiv \neg \text{excThrown}(E) && \vee \\
& \text{excThrown}(NPE) \wedge (s = \text{null} \vee d = \text{null}) && \vee \\
& \text{excThrown}(OBE) \wedge \\
& (sO < 0 \vee dO < 0 \vee l < 0 \vee sO + l > \text{size}(s) \vee dO + l > \text{size}(d))
\end{aligned}$$

4.2 Syntax der Prädikatenlogik

Die jetzt aufzubauende Prädikatenlogik erster Ordnung nennen wir kurz auch: PL1.

4.2.1 Terme und Formeln

Definition 4.1 (Sonderzeichen)

Die folgenden Zeichen sind in jeder Sprache der PL1 vorhanden. Sie heißen *Logische Zeichen* (manchmal auch Sonderzeichen).

wie Aussagenlogik	neu
(, Komma
)	\doteq objektsprachliches Gleichheitssymbol
1	\forall Allquantor
0	\exists Existenzquantor
\neg	v_i Individuenvariablen, $i \in \mathbb{N}$
\wedge	
\vee	
\rightarrow	
\leftrightarrow	

$Var := \{v_i \mid i \in \mathbb{N}\}$ ist die Menge der *Individuenvariablen* oder kurz *Variablen*. Var ist disjunkt zur Menge der übrigen Sondersymbole.

Einige Zeichen, wie z.B. das Komma, kommen sowohl in der Objektsprache als auch in der Metasprache vor. Aber das wird kaum Anlaß zu Verwechslungen geben. Bei dem Zeichen für die Gleichheit machen wir hingegen explizit einen Unterschied. Wir benutzen ein eigenes Zeichen \doteq für die Gleichheit in der Objektsprache und $=$ in der Metasprache.

Definition 4.2 (Signatur)

Eine *Signatur* der PL1 ist ein Tripel $\Sigma = (F_\Sigma, P_\Sigma, \alpha_\Sigma)$ mit:

- F_Σ, P_Σ sind endliche oder abzählbar unendliche, effektiv gegebene Mengen
- F_Σ, P_Σ und die Menge der Sondersymbole sind paarweise disjunkt
- $\alpha_\Sigma : F_\Sigma \cup P_\Sigma \rightarrow \mathbb{N}$.

Die $f \in F_\Sigma$ heißen *Funktionssymbole*, die $p \in P_\Sigma$ *Prädikatsymbole*. α_Σ ordnet jedem Funktions- oder Prädikatsymbol seine *Stelligkeit* zu: f ist *n-stelliges Funktionssymbol*, wenn $\alpha_\Sigma(f) = n$; entsprechend für $p \in P_\Sigma$.

Ein nullstelliges Funktionssymbol heißt auch *Konstantensymbol* oder kurz *Konstante*, ein nullstelliges Prädikatsymbol ist ein *aussagenlogisches Atom*.

Wir setzen voraus, daß die Mengen der Signatursymbole und der Sondersymbole disjunkt sind.

Definition 4.3 (Terme)

$Term_\Sigma$, die Menge der *Terme über Σ* , ist induktiv definiert durch

1. $Var \subseteq Term_\Sigma$
2. Mit $f \in F_\Sigma, \alpha_\Sigma(f) = n, t_1, \dots, t_n \in Term_\Sigma$ ist auch $f(t_1, \dots, t_n) \in Term_\Sigma$

Man beachte, daß jede Konstante ein Term ist. Ein Term heißt *Grundterm*, wenn er keine Variablen enthält.

Definition 4.4 (Atomare Formeln)

At_Σ , die Menge der *atomaren Formeln* – oder *Atome* – über Σ , ist definiert als

$$At_\Sigma := \{s \doteq t \mid s, t \in Term_\Sigma\} \cup \{p(t_1, \dots, t_n) \mid p \in P_\Sigma, \alpha_\Sigma(p) = n, t_1, \dots, t_n \in Term_\Sigma\}$$

Definition 4.5 (Formeln)

For_Σ , die Menge der *Formeln über Σ* , ist induktiv definiert durch

1. $\{\mathbf{1}, \mathbf{0}\} \cup At_\Sigma \subseteq For_\Sigma$
2. Mit $x \in Var$ und $A, B \in For_\Sigma$ sind ebenfalls in For_Σ :

$$\neg A, (A \wedge B), (A \vee B), (A \rightarrow B), (A \leftrightarrow B), \forall x A, \exists x A$$

Bemerkungen

Wir schreiben in der Regel

x, y, z, \dots	für Variable
a, b, c, \dots	für Konstanten
f, g, h, \dots	für Funktionssymbole allgemein
P, Q, \dots	für aussagenlogische Atome
p, q, r, \dots	für Prädikatensymbole allgemein
s, t, u, \dots	für Terme
A, B, C, \dots	für Formeln

ggf. mit Indizes oder ähnlichen Unterscheidungszeichen.

Den Index Σ lassen wir oft weg. Zur kürzeren Notation von Formeln werden dieselben Klammereinsparungsregeln verwendet wie in der Aussagenlogik.

Die Anwendung von Funktionssymbolen geben wir manchmal auch in Infixschreibweise wieder: $(s + t)$ statt $+(s, t)$.

Wir denken uns die Mengen F_Σ, P_Σ stets *effektiv* gegeben (vgl.: Grundlagen der Berechenbarkeit) und nehmen entsprechend an, daß α_Σ *berechenbar* sei.

Strukturelle Induktion

Die Definition oder der Beweis einer Eigenschaft von Termen oder Formeln durch *strukturelle Induktion* geschieht völlig entsprechend zur Situation in der Aussagenlogik. Man formuliere das zur Übung aus. Wie wir sehen werden, sind Induktionsbeweise einer anderen Art als durch strukturelle Induktion manchmal besser geeignet.

Definition 4.6

Ein *Teilterm* oder *Unterterm* eines Terms t ist ein Teilwort von t , das selbst Term ist; entsprechend sind die Teilterme (Unterterme) einer Formel definiert. Eine *Teilformel* einer Formel A ist ein Teilwort von A , das selbst Formel ist.

4.2.2 Gebundene und freie Variable. Substitutionen

Definition 4.7

1. Wenn wir im folgenden vom Auftreten einer Variablen, z.B. x in einer Formel reden, so schließen wir das Vorkommen von x direkt nach einem Quantor, also $\forall x$ oder $\exists x$, aus. Dieses Vorkommen von x wird als Bestandteil des Quantors angesehen.
2. Mit $Var(A)$, $Var(t)$ bezeichnen wir alle in der Formel A , bzw. in dem Term t vorkommenden Variablen.
3. Hat eine Formel A die Gestalt $\forall xB$ oder $\exists xB$ so heißt B der *Wirkungsbereich* des Quantors $\forall x$ bzw. $\exists x$ von A .
4. Ein Auftreten einer Variablen x in einer Formel A heißt *gebunden*, wenn es innerhalb des Wirkungsbereichs eines Quantors $\forall x$ oder $\exists x$ einer Teilformel von A stattfindet.
5. Mit $Bd(A)$ bezeichnen wir die Menge der Variablen, die in A mindestens einmal gebunden auftreten.
6. Ein Auftreten einer Variablen x in einer Formel A heißt *frei*, wenn es nicht gebunden.
7. Mit $Frei(A)$ bezeichnen wir die Menge der Variablen, die in A mindestens einmal frei auftreten.

Man könnte hierbei das *Auftreten* eines Zeichens ζ in A definieren als ein Paar (ζ, i) mit: $1 \leq i \leq |A|$, und an der Position i in A steht ζ . Für unsere Zwecke genügt jedoch ein intuitives Verständnis dieses Konzepts.

Siehe auch Aufgabe 4.2.5.

Man beachte, daß eine Variable in einer Formel auch frei *und* gebunden auftreten kann, d.h. es kann $Frei(A) \cap Bd(A) \neq \{\}$ gelten. Ferner, daß im Wirkungsbereich eines Quantors, etwa $\forall x$, derselbe Quantor wieder auftreten kann.

Beispiel 4.8

In der Formel

$$\forall x(p_0(x, y) \rightarrow \forall z(\exists y p_1(y, z) \vee \forall x p_2(f(x), x)))$$

treten x und z nur gebunden auf, y tritt frei und gebunden auf.

Für Terme gibt es keine Bindungen von Variablen, so daß wir jedes Auftreten einer Variablen in einem Term als dort freies Auftreten ansehen wollen.

$t \in Term_\Sigma$ heißt *Grundterm*, wenn $Var(t) = \{\}$.

Definition 4.9

A heißt *geschlossen*, wenn $Frei(A) = \{\}$. Ist allgemein $Frei(A) = \{x_1, \dots, x_n\}$, so heißt

$\forall x_1 \dots \forall x_n A$ *Allabschluß*

$\exists x_1 \dots \exists x_n A$ *Existenzabschluß*

von A . Abkürzend schreiben wir $Cl_\forall A$ bzw. $Cl_\exists A$.

Ist A geschlossen, dann gilt also $Cl_\forall A = Cl_\exists A = A$.

Definition 4.10

(Substitutionen) Eine *Substitution* (über Σ) ist eine Abbildung

$$\sigma : Var \rightarrow Term_\Sigma$$

mit $\sigma(x) = x$ für fast alle $x \in Var$.

Sind x_1, \dots, x_m so, daß gilt $\{x \mid \sigma(x) \neq x\} \subseteq \{x_1, \dots, x_m\}$, und ist $\sigma(x_i) = s_i$ für $i = 1, \dots, m$, so geben wir σ auch an in der Schreibweise

$$\{x_1/s_1, \dots, x_m/s_m\}.$$

σ heißt *Grundsubstitution*, wenn für alle $x \in Var$ gilt: $\sigma(x) = x$ oder $\sigma(x)$ ist Grundterm.

Mit *id* bezeichnen wir die identische Substitution auf Var , d.h. $id(x) = x$ für alle $x \in Var$.

Definition 4.11

(Anwendung von Substitutionen) Wir setzen eine Substitution σ fort zu Abbildungen

$$Term_\Sigma \rightarrow Term_\Sigma$$

$$For_\Sigma \rightarrow For_\Sigma$$

die beide wieder mit σ bezeichnet werden, mittels:

- $\sigma(f(t_1, \dots, t_n)) = f(\sigma(t_1), \dots, \sigma(t_n))$
- $\sigma(p(t_1, \dots, t_n)) = p(\sigma(t_1), \dots, \sigma(t_n))$

- $\sigma(t \doteq s) = \sigma(t) \doteq \sigma(s)$
- $\sigma(\neg A) = \neg\sigma(A)$
- $\sigma(A \circ B) = \sigma(A) \circ \sigma(B)$ für jeden zweistelligen aussagenlogischen Operator \circ .
- $\sigma(QxA) = Qx\sigma_x(A)$, wobei $\sigma_x(x) = x$ und $\sigma_x(y) = \sigma(y)$ für $y \neq x$, für $Q \in \{\forall, \exists\}$

$\sigma(A)$ entsteht aus A , indem simultan für jedes $x \in Var$ an jeder Stelle, wo x frei in A auftritt, x ersetzt wird durch $\sigma(x)$.

Beispiel 4.12

1. Für $\sigma = \{x/f(x, y), y/g(x)\}$ gilt $\sigma(f(x, y)) = f(f(x, y), g(x))$.
2. Für $\mu = \{x/c, y/d\}$ gilt $\mu(\exists yp(x, y)) = \exists yp(c, y)$.
3. Für $\sigma_1 = \{x/f(x, x)\}$ gilt $\sigma_1(\forall yp(x, y)) = \forall yp(f(x, x), y)$.
4. Für $\mu_1 = \{x/y\}$ gilt $\mu_1(\forall yp(x, y)) = \forall yp(y, y)$.

Das Beispiel 4 unterscheidet sich von den vorangegangenen dadurch, daß an einer Position der Formel vor der Substitution eine ungebundene Variable steht, nämlich x , und nach der Substitution eine gebundene Variable, nämlich y . Man hat schon jetzt das Gefühl, daß eine solche Substitution die Bedeutung einer Formel in unerlaubter Weise verändert, was wir in Lemma 4.34 konkretisieren können. Wir wollen jedoch schon hier für die in Beispiel 4 aufgetretene Situation eine Bezeichnung einführen.

Definition 4.13

(kollisionsfreie Substitutionen)

Eine Substitution σ heißt *kollisionsfrei* für eine Formel A , wenn für jede Variable z und jede Stelle freien Auftretens von z in A gilt: Diese Stelle liegt nicht im Wirkungsbereich eines Quantors $\forall x$ oder $\exists x$, wo x eine Variable in $\sigma(z)$ ist.

Nach dieser Definition ist $\{x/y\}$ keine kollisionsfreie Substitution für $\forall yp(x, y)$.

Definition 4.14

(Komposition von Substitutionen)

Sind σ, τ Substitutionen, dann definieren wir die Komposition von τ mit σ durch

$$(\tau \circ \sigma)(x) = \tau(\sigma(x)).$$

Man beachte, daß auf der rechten Seite τ als die zugehörige, gleichnamige Abbildung $Term_\Sigma \rightarrow Term_\Sigma$ verstanden werden muß.

Lemma 4.15

1. Gilt für $t \in Term_\Sigma$ und Substitutionen σ, τ , daß $\sigma(t) = \tau(t)$, dann $\sigma(s) = \tau(s)$ für jeden Teilterm s von t .
2. Wenn $\sigma(t) = t$, dann $\sigma(s) = s$ für jeden Teilterm s von t .

Beweis

1. Strukturelle Induktion nach t .

- Ist $t \in Var$, dann ist t selbst sein einziger Teilterm.
- Sei $t = f(t_1, \dots, t_n)$. Dann gilt

$$\sigma(f(t_1, \dots, t_n)) = f(\sigma(t_1), \dots, \sigma(t_n))$$

und

$$\tau(f(t_1, \dots, t_n)) = f(\tau(t_1), \dots, \tau(t_n)).$$

Es folgt also $\sigma(t_i) = \tau(t_i)$ für $i = 1, \dots, n$. Der Term s ist mit t identisch oder Teilterm eines t_i (Übung in 4.2.1), nach Induktionsvoraussetzung folgt also die Behauptung.

2. Spezialfall von 1.

■

Für spätere Zwecke besprechen wir noch eine spezielle Sorte von Substitutionen.

Definition 4.16

Eine *Variablenumbenennung* ist eine Substitution σ mit

1. $\sigma(x) \in Var$ für alle $x \in Var$
2. σ ist injektiv

Korollar 4.17

Gilt für Substitutionen σ, τ , daß $\tau \circ \sigma = id$, dann ist σ eine Variablenumbenennung.

Beweis

Es ist $\tau(\sigma(x)) = x$ für jedes $x \in Var$, woraus folgt: $\sigma(x) \in Var$. Ferner haben wir: Wenn $\sigma(x) = \sigma(y)$, dann $x = \tau(\sigma(x)) = \tau(\sigma(y)) = y$. ■

4.2.3 Unifikation

Für eine Menge $T \subseteq Term_\Sigma$ schreiben wir kurz

$$\sigma(T)$$

für

$$\{\sigma(t) \mid t \in T\}.$$

Entsprechend für $M \subseteq For_\Sigma$.

Definition 4.18

Es sei $T \subseteq Term_\Sigma$, $T \neq \{\}$, und σ eine Substitution über Σ . σ *unifiziert* T , oder: ist *Unifikator von* T , genau dann, wenn $\#\sigma(T) = 1$. T heißt *unifizierbar*, wenn T einen Unifikator besitzt. Insbesondere sagen wir für zwei Terme s, t daß s *unifizierbar* sei *mit* t , wenn $\{s, t\}$ unifizierbar ist, d.h. in diesem Fall, wenn $\sigma(t) = \sigma(s)$.

Die Begriffe werden auf Formelmengen übertragen durch: σ *unifiziert* M genau dann, wenn $\#\sigma(M) = 1$. (Hierbei liefert $\#$ zu einer Menge ihre Mächtigkeit; insbesondere zu einer endlichen Menge die Anzahl ihrer Elemente.)

Beispiel 4.19

$\{f(g(a, x), g(y, b)), f(z, g(v, w)), f(g(x, a), g(v, b))\}$ wird unifiziert durch $\{x/a, y/v, z/g(a, a), w/b\}$.

Wir beschränken uns im folgenden auf die Betrachtung der Unifikation von Termmengen und überlassen die (einfache) Übertragung auf Formelmengen dem Leser.

Übungsaufgabe 4.2.1

1. Jeder Term ist mit sich selbst unifizierbar mittels *id*.

2. Zwei Terme der Gestalt

$$f(s_1, \dots, s_n), f(t_1, \dots, t_n)$$

(mit demselben Kopf) sind genau dann unifizierbar, wenn es eine Substitution σ gibt mit $\sigma(s_i) = \sigma(t_i)$ für $i = 1, \dots, n$.

3. Ist $x \in Var$ und t ein Term, der x nicht enthält, dann sind x und t unifizierbar.
4. Ist $x \in Var$ und t ein Term $\neq x$, der x enthält, dann sind x und t nicht unifizierbar

Der Leser formuliere entsprechende Aussagen für Formeln, soweit sie Sinn machen.

Man sollte erkennen, daß sich in der obigen Auflistung einfachster Fälle ein Algorithmus verbirgt, der die Unifizierbarkeit einer endlichen Menge von Termen testet und im positiven Fall einen Unifikator liefert. Bevor wir diesen Algorithmus hinschreiben, gehen wir noch auf die Frage der Eindeutigkeit eines Unifikators ein.

Beispiel 4.20

$$\{f(x, g(y)), f(g(a), g(z))\}$$

wird unifiziert durch

$$\sigma = \{x/g(a), z/y\},$$

aber auch durch

$$\tau = \{x/g(a), y/a, z/a\}.$$

σ ist *allgemeiner* als τ – oder τ *spezieller* als σ – insofern, als sich aus dem Resultat der Anwendung von σ auf eine beliebige Termmenge T immer noch, durch die nachgeschaltete Substitution $\{y/a\}$, das Resultat der Anwendung von τ auf T gewinnen läßt:

$$\tau(T) = \{y/a\}(\sigma(T)) \text{ für alle } T \subseteq Term_{\Sigma}$$

d. h.

$$\tau = \{y/a\} \circ \sigma.$$

Uns interessieren insbesondere Unifikatoren einer Termmenge, die in diesem Sinne so allgemein wie möglich sind.

Definition 4.21

Es sei $T \subseteq \text{Term}_\Sigma$. Ein *allgemeinster Unifikator* oder mgu (*most general unifier*) von T ist eine Substitution μ mit

1. μ unifiziert T
2. Zu jedem Unifikator σ von T gibt es eine Substitution σ' mit $\sigma = \sigma' \circ \mu$.

Lemma 4.22

Es sei T eine nichtleere Menge von Termen. Dann ist jeder allgemeinste Unifikator von T bis auf Variablenumbenennung eindeutig bestimmt, d. h.: Sind μ, μ' allgemeinste Unifikatoren von T mit $\mu(T) = \{t\}$ und $\mu'(T) = \{t'\}$, dann gibt es eine Variablenumbenennung π mit $t' = \pi(t)$.

Beweis

Nach der Definition allgemeinster Unifikatoren gibt es Substitutionen σ, σ' mit

- $\mu' = \sigma\mu$
- $\mu = \sigma'\mu'$

Daraus folgt $\mu = \sigma'\sigma\mu$ und insbesondere für jeden Term $t_0 \in T$ $\mu(t_0) = \sigma'\sigma\mu(t_0)$. Also auch $t = \sigma'\sigma(t)$. Das kann nur sein, wenn für jede Variable $x \in \text{Var}(t)$ gilt $\sigma'\sigma(x) = x$. Daraus folgt insbesondere, daß für jedes $x \in \text{Var}(t)$ $\sigma(x)$ wieder eine Variable sein muß und für $x, y \in \text{Var}(t)$ mit $x \neq y$ auch $\sigma(x) \neq \sigma(y)$ gilt. Wir definieren

$$\pi(x) = \begin{cases} \sigma(x) & \text{falls } x \in \text{Var}(t) \\ \varphi(x) & \text{falls } x \notin \text{Var}(t). \end{cases}$$

wobei φ eine beliebige bijektive Funktion von $\text{Var} \setminus \text{Var}(t)$ auf $\text{Var} \setminus \{\sigma(x) \mid x \in \text{Var}(t)\}$ ist mit: $\varphi(y) = y$ für fast alle $y \in \text{Var}$ (wegen der Endlichkeit von $\text{Var}(t)$). Offensichtlich ist π eine Variablenumbenennung und nach Wahl von σ gilt für jeden Term $t_0 \in T$ $\mu'(t_0) = \sigma\mu(t_0)$, also

$$t' = \mu'(t_0) = \sigma\mu(t_0) = \sigma(t).$$

■

4.2.4 Unifikationsalgorithmus

Wir stellen nun den von ROBINSON stammenden Algorithmus vor, der zu einer endlichen Menge T von Termen entscheidet, ob T unifizierbar ist, und im positiven Fall einen allgemeinsten Unifikator liefert. Wir benötigen die

Definition 4.23

Zu $t \in Term_{\Sigma}$ und $i \in \mathbb{N}$ sei

$t^{(i)}$:= der an Position i in t (beim Lesen von links nach rechts) beginnende
Teilterm von t , wenn dort eine Variable oder ein Funktionssymbol
steht;
undefiniert sonst.

Es sei nun $T \subseteq Term_{\Sigma}$ endlich und $\neq \emptyset$. Die *Differenz* von T ist die folgende Menge $D(T) \subseteq Term_{\Sigma}$

1. $D(T) := T$ falls $\#T = 1$
2. Falls $\#T \geq 2$, sei i das kleinste j mit: Mindestens zwei Terme $\in T$ unterscheiden sich an Position j . Setze $D(T) := \{t^{(i)} \mid t \in T\}$.

(Beachte: Nach Übung 4.2.2, 6 (Seite 121) ist $t^{(i)}$ wirklich ein Teilterm von t .)

Man bestätige zur Übung, daß gilt:

Es sei $\#T \geq 2$ und T unifizierbar. Dann

- 1) $\#D(T) \geq 2$
- 2) Jeder Unifikator von T unifiziert auch $D(T)$
- 3) $D(T)$ enthält eine Variable x und einen Term t mit $x \notin Var(t)$.

Satz 4.24

Der Algorithmus von ROBINSON (Bild 4.1, Seite 123) terminiert für jedes endliche, nichtleere $T \subseteq Term_{\Sigma}$. Wenn T unifizierbar ist, liefert er einen allgemeinsten Unifikator von T . Wenn T nicht unifizierbar ist, liefert er die Ausgabe „ T nicht unifizierbar“.

Beweis

Wir zeigen

1. Der Algorithmus terminiert.

2. Wenn er eine Substitution μ ausgibt, dann ist μ Unifikator von T .
3. Ist σ ein beliebiger Unifikator von T , dann gilt: Es gibt μ, σ' so daß
 - der Algorithmus mit Ausgabe μ terminiert,
 - $\sigma = \sigma' \circ \mu$
 - *Somit:* μ ist mgu von T .
4. Wenn der Algorithmus ausgibt „ T nicht unifizierbar“, dann ist T nicht unifizierbar.

Unter einem *Schleifendurchlauf* in dem obigem Algorithmus verstehen wir einen *vollständigen* Durchlauf der Schleife, d. h. der Befehlsfolge 2–4–5. Ist S eine Menge von Termen und x eine Variable, so sagen wir kurz, daß x in S *auftritt*, wenn x in einem Term in S auftritt.

Wir setzen

- $S_0 := T, \mu_0 := id$
- $S_{k+1} :=$ Wert von S nach dem $(k + 1)$ -ten Schleifendurchlauf
- $\mu_{k+1} :=$ entsprechend mit μ

sofern die Schleife tatsächlich so oft durchlaufen wird. Ferner seien dann x_k, t_k die im $(k + 1)$ -ten Schleifendurchlauf ausgewählten x, t .

Ad 1:

k sei so, daß die Schleife mindestens $(k + 1)$ -mal durchlaufen wird. Dann gilt

$$S_{k+1} = \{x_k/t_k\}(S_k).$$

Dabei tritt x_k in S_k , aber nicht in t_k auf, jede Variable in S_{k+1} aber auch in S_k . Also: Beim Übergang von S_k zu S_{k+1} vermindern sich die Variablen in S_{k+1} genau um x_k . Es folgt, daß die Schleife nur endlich oft durchlaufen wird. Das Programm terminiert.

Im Folgenden sei m die Anzahl der Schleifendurchläufe. (Es wird also 5 genau m -mal, 2 genau $(m + 1)$ -mal ausgeführt.)

Ad 2:

Durch Induktion über k ergibt sich unmittelbar, daß

$$S_k = \mu_k(T)$$

für alle $k \leq m$. Hält das Programm mit Ausgabe einer Substitution μ an, dann hat μ den Wert μ_m , und es ist

$$\#\mu_m(T) = \#S_m = 1.$$

μ_m ist Unifikator von T .

Ad 3:

Es sei σ ein Unifikator von T . Wir zeigen zunächst

(*) Für alle $k \leq m$: es gibt σ_k mit $\sigma = \sigma_k \circ \mu_k$.

$k = 0$:

Setze $\sigma_0 := \sigma$.

$k + 1$:

Nach Induktionsannahme existiert σ_k mit $\sigma = \sigma_k \circ \mu_k$. Wir haben

$$\#\sigma_k(S_k) = \#\sigma_k(\mu_k(T)) = \#\sigma(T) = 1$$

da σ Unifikator von T ist. Wegen $k + 1 \leq m$ wurde im $(k + 1)$ -ten Durchlauf noch Test 2 mit „Nein“ und Test 4 mit „Ja“ verlassen: es ist $\#S_k \geq 2$, und in $D(S_k)$ gibt es x_k, t_k mit $x_k \notin \text{Var}(t_k)$. Da σ_k Unifikator von S_k ist, muß gelten $\sigma_k(x_k) = \sigma_k(t_k)$. Wir setzen

$$\sigma_{k+1}(x) = \begin{cases} \sigma_k(x) & \text{falls } x \neq x_k \\ x_k & \text{falls } x = x_k. \end{cases}$$

Wir haben für alle x :

Falls $x \neq x_k$:

$$\sigma_{k+1}(\{x_k/t_k\}(x)) = \sigma_{k+1}(x) = \sigma_k(x),$$

falls $x = x_k$:

$$\begin{aligned} \sigma_{k+1}(\{x_k/t_k\}(x)) &= \sigma_{k+1}(\{x_k/t_k\}(x_k)) \\ &= \sigma_{k+1}(t_k) = \sigma_k(t_k) \quad (\text{da } x_k \notin \text{Var}(t_k)) \\ &= \sigma_k(x_k) = \sigma_k(x). \end{aligned}$$

Somit

$$\sigma_{k+1} \circ \{x_k/t_k\} = \sigma_k.$$

Es folgt

$$\begin{aligned} \sigma_{k+1} \circ \mu_{k+1} &= \sigma_{k+1} \circ \{x_k/t_k\} \circ \mu_k \\ &= \sigma_k \circ \mu_k = \sigma \end{aligned}$$

d. h. (*).

Insbesondere gilt

$$\sigma = \sigma_m \circ \mu_m.$$

Angenommen nun, bei der $(m + 1)$ -ten Durchführung des Tests 2 würde der „Nein“-Ausgang gewählt, d. h. es wäre $\#S_m \geq 2$.

σ_m unifiziert S_m (da σ T unifiziert). Also muß $D(S_m)$ eine Variable x und einen Term t enthalten mit $x \notin \text{Var}(t)$ (siehe die obige Übung zum Operator D). Test 4 würde mit „Ja“ verlassen, im Widerspruch zur Definition von m . Hiermit ist 3 bewiesen.

Ad 4: Folgt unmittelbar aus 3. ■

4.2.5 Übungsaufgaben

Übungsaufgabe 4.2.2

Beweisen Sie die folgenden Aussagen

1. $\text{Term}_\Sigma \cap \text{For}_\Sigma = \{\}$
2. Kein Teilwort eines Terms s ist eine Formel.
3. Ist s ein Term und sind t, u Teilterme von s , dann gilt genau eine der Aussagen
 - u ist Teilterm von t
 - t ist echter Teilterm von u
 - t, u liegen disjunkt

Entsprechendes gilt für Teilterme einer Formel und Teilformeln einer Formel.

4. Ist der Term t ein Präfix des Terms s , dann $s = t$. Ist die Formel B ein Präfix der Formel A , dann $A = B$. Keine Formel kann Präfix eines Terms sein.
5. Es sei t ein Term und i eine Position in t (d. h. $1 \leq i \leq |t|$; t wird von links nach rechts gelesen). Genau dann beginnt bei i ein Teilterm von t , wenn dort eine Variable oder ein Funktionssymbol steht.
6. Es seien $s, t \in \text{Term}_\Sigma$, $s \neq t$. Dann gibt es eine erste Position i , $1 \leq i \leq \min(|s|, |t|)$, an der s und t sich unterscheiden. Für dieses i gilt: Sowohl in s wie in t steht dort eine Variable oder ein Funktionssymbol. (Nach 5 also: dort beginnt ein Teilterm.)

Es sei nochmals betont, daß diese Aussagen sich auf unsere „offizielle“ Sprache (ohne Klammereinsparungen) beziehen.

Übungsaufgabe 4.2.3

Definieren Sie den Begriff eines „Teilterms zu einem Term s “ durch strukturelle Induktion. Entsprechend zu einer Formel A : „Teilformel von A “.

Übungsaufgabe 4.2.4

Im Falle einer endlichen Signatur Σ sind $Term_\Sigma$, For_Σ kontextfreie Sprachen, man gebe erzeugende Grammatiken an. Auch bei einer unendlichen Signatur lassen sich in manchen Fällen kontextfreie Grammatiken für $Term_\Sigma$ bzw. For_Σ angeben, wenn man die Funktions- und Prädikatsymbole geeignet codiert: etwa, wenn es zu jeder Stelligkeit n unendlich viele n -stellige Funktions- und Prädikatsymbole gibt; oder wenn dies, mit festem m , für alle $n \leq m$ der Fall ist und insgesamt nur für endlich viele Stelligkeiten solche Symbole vorhanden sind. Weshalb hat man im allgemeinen Fall Schwierigkeiten?

Übungsaufgabe 4.2.5

Man definiere $Frei(A)$ und $Bd(A)$ direkt durch strukturelle Induktion.

Übungsaufgabe 4.2.6

Zu jeder Substitution σ und jeder Formel A gibt es eine Formel A' , die aus A durch Umbenennung gebundener Variablen entsteht, so daß σ für A' kollisionsfrei ist.

Übungsaufgabe 4.2.7

Geben Sie (falls existent) für die folgenden Paare von Termen einen *allgemeinsten Unifikator* σ an. Falls kein mgu existiert, begründen Sie dies.

1. $\{p(f(x_1, x_1), x_1, f(x_2, x_2), x_2), p(y_1, f(y_2, y_2), y_2, f(y_3, y_3)))\}$
2. $\{p(f(x, y), g(x, y)), p(f(h(z), h(z)), g(h(z), z))\}$
3. $\{p(f(y), w, g(z)), p(u, u, v)\}$
4. $\{p(f(y), w, g(z)), p(v, u, v)\}$

Gegeben sei $T \subseteq Term_{\Sigma}$, T endlich und $\neq \emptyset$.

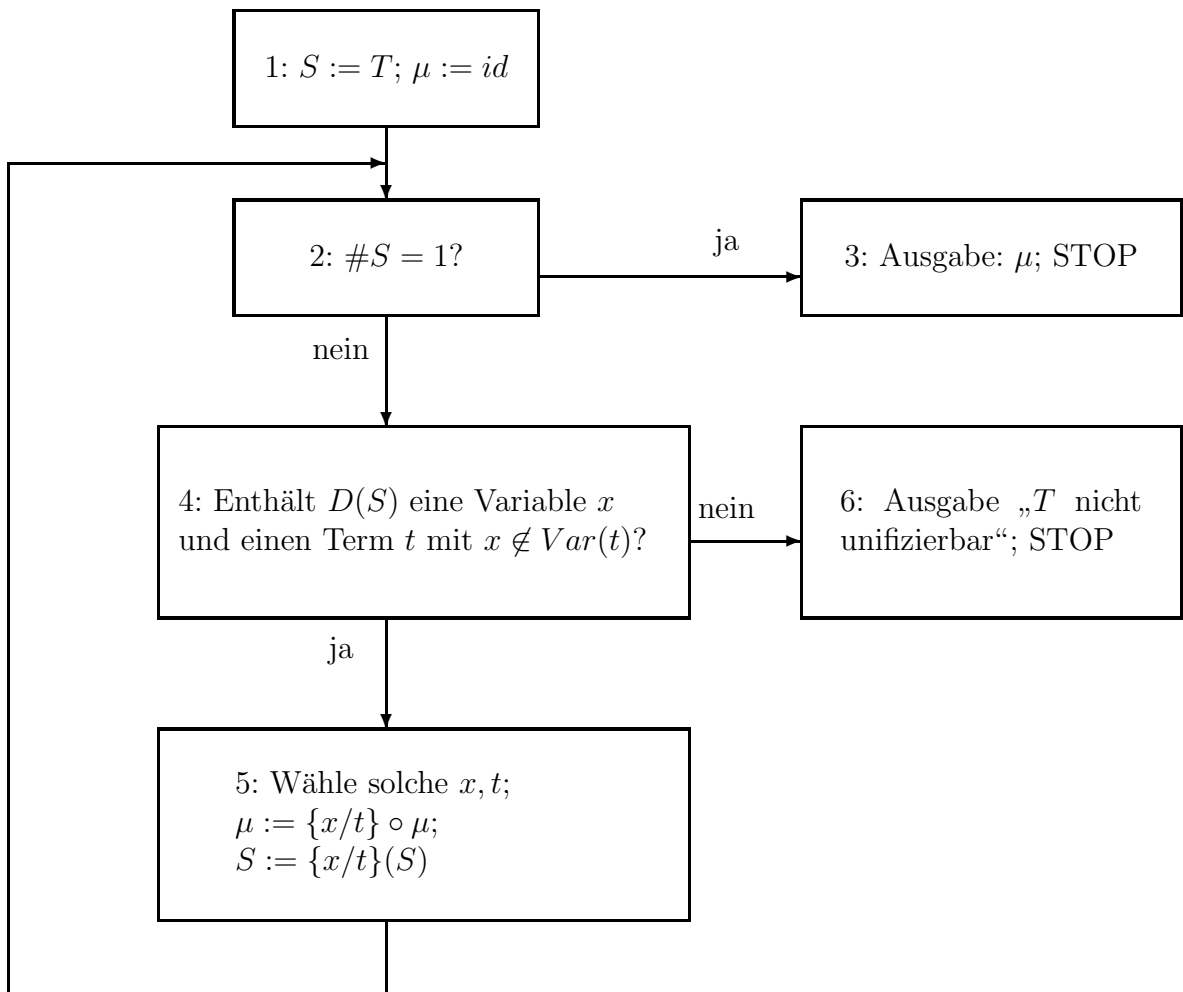


Abbildung 4.1: Algorithmus von Robinson

4.3 Semantik der PL1

Betrachten wir die Formel

$$q(x) \rightarrow \exists y(in(y, x) \wedge kl(y)),$$

so können wir bisher nur sagen, daß sie eine korrekt gebildet Formel über der Signatur $\Sigma = \{k(), q(), d(), kl(), gr(), in(,)\}$ ist. Es macht keinen Sinn zu fragen, ob diese Formel, wahr oder falsch ist. Dazu müßten wir zuerst wissen, welche Bedeutung die auftretenden Relationszeichen q , kl und in haben, wir müßten wissen über welchem Bereich von Elementen wir nach einem Beispiel für die durch $\exists y$ signalisierte Existenzbehauptung suchen sollen. Diese Informationen werden durch die im nächsten Abschnitt definierten *Interpretationen* festgelegt. Die Angabe einer Interpretation genügt aber noch immer nicht um der Formel einen Wahrheitswert zuordnen zu können, solange wir nicht wissen, auf welches Element die freie Variable x referenziert. Diese Information wird durch die sogenannten *Variablenbelegungen* festgelegt. In Abhängigkeit von einer Interpretation \mathcal{D} und einer Variablenbelegung β liegt jetzt die Wahrheit oder Falschheit der obigen Formel fest.

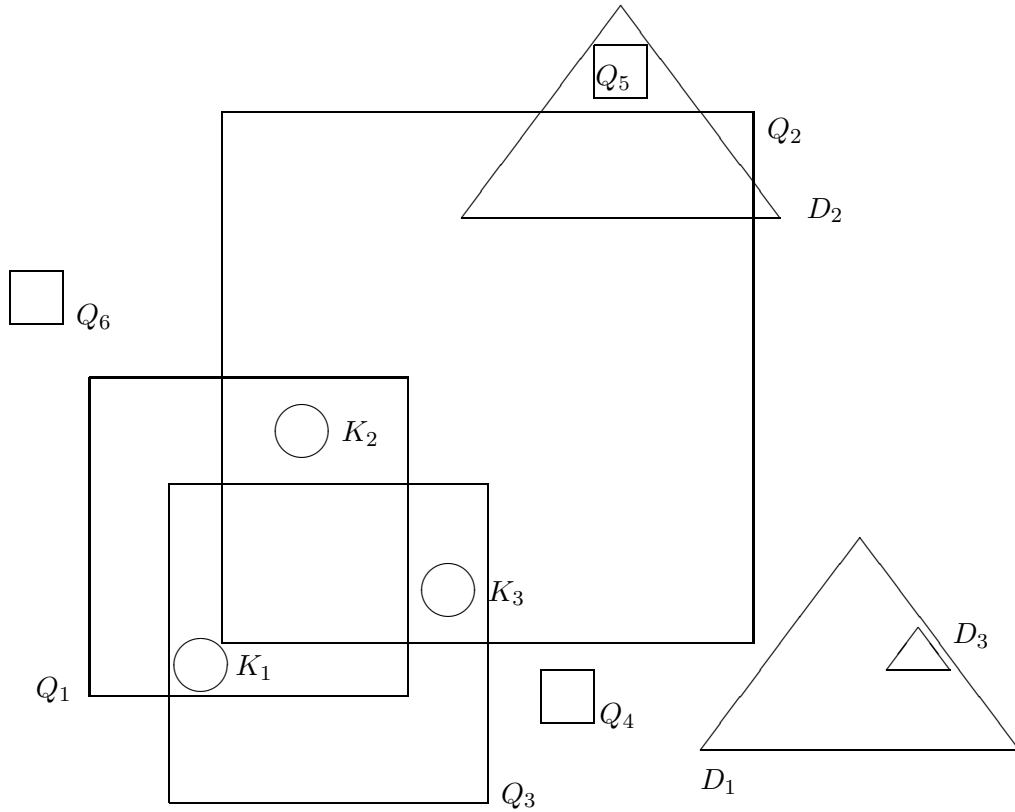
4.3.1 Interpretationen

Definition 4.25 (Interpretation)

Es sei Σ eine Signatur der PL1. Eine *Interpretation* \mathcal{D} von Σ ist ein Paar (D, I) mit

1. D ist eine beliebige, nichtleere Menge
2. I ist eine Abbildung der Signatursymbole, die
 - jeder Konstanten c ein Element $I(c) \in D$
 - für $n \geq 1$: jedem n -stelligen Funktionssymbol f eine Funktion $I(f) : D^n \rightarrow D$
 - jedem 0-stelligen Prädikatsymbol P einen Wahrheitswert $I(P) \in \{W, F\}$
 - für $n \geq 1$: jedem n -stelligen Prädikatsymbol p eine n -stellige Relation $I(p) \subseteq D^n$ zuordnet.

Wir werden manchmal eine Interpretation $\mathcal{D} = (D, I)$ auch eine *prädikatenlogische Struktur*, oder einfach eine *Struktur* nennen. Der Funktion I haben wir hier



$$\begin{aligned}
P_\Sigma &= \{k(\), q(\), d(\), kl(\), gr(\), in(\ , \))\} \\
D_{Bsp} &= \{Q_i : 1 \leq i \leq 6\} \cup \{K_1, K_2, K_3, D_1, D_2, D_3\} \\
I_{Bsp}(q) &= \{Q_i : 1 \leq i \leq 6\} \\
I_{Bsp}(k) &= \{K_1, K_2, K_3\} \\
I_{Bsp}(d) &= \{D_1, D_2, D_3\} \\
I_{Bsp}(in) &= \{(K_1, Q_1), (K_1, Q_3), (K_2, Q_1), (K_2, Q_2), (K_3, Q_2), (K_3, Q_3), (D_3, D_1), (Q_5, D_2)\}
\end{aligned}$$

Tabelle 4.1: Beispiel einer Interpretation

keinen eigenen Namen gegeben. Falls notwendig nennen wir I die *Interpretationsfunktion*.

Abbildung 4.1 zeigt ein Beispiel einer Interpretation Σ mit den Prädikatszeichen $P_\Sigma = \{k(\), q(\), d(\), kl(\), gr(\), in(\ , \))\}$. Die Interpretation des einstelligen Prädikatszeichens k ist dabei die Menge aller Kreise, die von q die Menge aller Quadrate, die von d die Menge aller Dreiecke. Das Universum besteht in diesem Beispiel aus der Menge aller aufgezeichneten geometrischen Objekte. Die Interpretation von kl ist die Menge aller kleinen Objekte, die von gr entsprechend die Menge aller großen Objekte. Da man sich darüber streiten kann, was groß und was klein ist, geben wir explizit an:

$$I_{Bsp}(kl) = \{K_1, K_2, K_3, Q_4, Q_5, Q_6, D_3\}$$

$$I_{Bsp}(gr) = \text{Komplement von } I_{Bsp}(kl) \text{ in } D.$$

Die Interpretation der binären Relation *in* wird in der Abbildung indirekt durch die geometrische Enthaltenseinsrelation dargestellt.

Definition 4.26 (Variablenbelegung)

Es sei (D, I) eine Interpretation von Σ . Eine *Variablenbelegung* (oder kurz *Belegung* über D) ist eine Funktion

$$\beta : Var \rightarrow D.$$

Zu β , $x \in Var$ und $d \in D$ definieren wir die *Modifikation* von β an der Stelle x zu d :

$$\beta_x^d(y) = \begin{cases} d & \text{falls } y = x \\ \beta(y) & \text{falls } y \neq x \end{cases}$$

Definition 4.27 (Auswertung von Termen und Formeln)

Es sei (D, I) eine Interpretation von Σ und β eine Variablenbelegung über D . Wir definieren eine Funktion $val_{D,I,\beta}$, welche jedem Term ein Element in D und jeder Formel einen Wahrheitswert zuordnet. Also:

$$val_{D,I,\beta} : Term_\Sigma \cup For_\Sigma \rightarrow D \cup \{W, F\}$$

mit

$$val_{D,I,\beta}(t) \in D \text{ für } t \in Term_\Sigma$$

$$val_{D,I,\beta}(A) \in \{W, F\} \text{ für } A \in For_\Sigma$$

1. $val_{D,I,\beta}$ auf $Term_\Sigma$:

$$val_{D,I,\beta}(x) = \beta(x) \text{ für } x \in Var$$

$$val_{D,I,\beta}(f(t_1, \dots, t_n)) = (I(f))(val_{D,I,\beta}(t_1), \dots, val_{D,I,\beta}(t_n))$$

2. $val_{D,I,\beta}$ auf For_Σ :

$$(a) \quad val_{D,I,\beta}(\mathbf{1}) = W$$

$$val_{D,I,\beta}(\mathbf{0}) = F$$

$$val_{D,I,\beta}(s \doteq t) := \begin{cases} W & \text{falls } val_{D,I,\beta}(s) = val_{D,I,\beta}(t) \\ F & \text{sonst} \end{cases}$$

$$val_{D,I,\beta}(P) := I(P) \text{ für 0-stellige Prädikate } P$$

$$val_{D,I,\beta}(p(t_1, \dots, t_n)) := \begin{cases} W & \text{falls } (val_{D,I,\beta}(t_1), \dots, val_{D,I,\beta}(t_n)) \in I(p) \\ F & \text{sonst} \end{cases}$$

- (b) Ist $val_{D,I,\beta}$ für Formeln A, B bereits definiert, so wird $val_{D,I,\beta}$ auf $\neg A, A \wedge B, A \vee B, A \rightarrow B, A \leftrightarrow B$ festgelegt wie in der Aussagenlogik. Ferner:

$$val_{D,I,\beta}(\forall x A) := \begin{cases} W & \text{falls für alle } d \in D : val_{D,I,\beta_x^d}(A) = W \\ F & \text{sonst} \end{cases}$$

$$val_{D,I,\beta}(\exists x A) := \begin{cases} W & \text{falls ein } d \in D \text{ existiert mit } val_{D,I,\beta_x^d}(A) = W \\ F & \text{sonst} \end{cases}$$

Insbesondere ist also $val_{D,I,\beta}(c) = I(c)$ für jede Konstante c .

Ferner erkennt man:

Ist t ein Grundterm, dann hängt $val_{D,I,\beta}(t)$ von β gar nicht ab. Wir schreiben dann $I(t)$ statt $val_{D,I,\beta}(t)$.

Beispiel 4.28

Wir wollen die Formel

$$q(x) \rightarrow \exists y(in(y, x) \wedge kl(y)),$$

in der Interpretation \mathcal{D}_{Bsp} aus Abbildung 4.1 mit der Variablenbelegung $\beta(x) = Q1$ auswerten.

Wir beginnen mit der links des Implikationspfeils \rightarrow stehenden Teilformel und bemerken $val_{\mathcal{D}_{Bsp},\beta}(q(x)) = Q1 \in I(q)$. Nach der Definition von val folgt also $val_{\mathcal{D}_{Bsp},\beta}(q(x)) = W$.

Wir behaupten, daß auch für die rechts von \rightarrow stehende Teilformel gilt:

$$val_{\mathcal{D}_{Bsp},\beta}(\exists y(in(y, x) \wedge kl(y))) = W$$

Dazu müssen wir ein geeignetes Element aus D für die Belegung der existentiell quantifizierten Variablen y finden. Im vorliegenden Fall eignet sich K_1 .

Wir behaupten nämlich

$$val_{\mathcal{D}_{Bsp},\beta_{K_1}^{K_1}}(in(y, x) \wedge kl(y)) = W$$

Durch weitere Anwendung der Definition von val kann diese Behauptung reduziert werden zu

$$(K_1, Q1) \in I_{Bsp}(in) \text{ und } K_1 \in I_{Bsp}(kl),$$

was offensichtlich nach Definition des Beispiels zutrifft. Insgesamt haben wir damit

$$val_{\mathcal{D}_{Bsp},\beta}(q(x) \rightarrow \exists y(in(y, x) \wedge kl(y))) = W$$

gezeigt.

Korollar 4.29

Es gelten die aus der Aussagenlogik bekannten Abhängigkeiten zwischen den Bedeutungen der Operatoren $\mathbf{1}, \mathbf{0}, \neg, \wedge, \vee, \rightarrow, \leftrightarrow$, so daß hier wieder die bekannten Teilmengen dieser Menge ausreichen. Ferner gilt für alle x, A :

$$\begin{aligned} val_{D,I,\beta}(\exists x A) &= val_{D,I,\beta}(\neg \forall x \neg A) \\ val_{D,I,\beta}(\forall x A) &= val_{D,I,\beta}(\neg \exists x \neg A) \end{aligned}$$

d. h. \exists ist durch \forall und \neg , \forall ist durch \exists und \neg ausdrückbar. Als *Basis* der Sprachen der PL1 können wir also etwa nehmen

$$\mathbf{0}, \rightarrow, \forall$$

oder

$$\neg, \rightarrow, \forall$$

oder

$$\neg, \wedge, \forall$$

und viele andere mehr. Die übrigen Operatoren lassen sich dann als Abkürzungen auffassen. Insbesondere genügt es bei struktureller Induktion über For_Σ , die Formeln über irgendeiner solchen Basis zu betrachten.

Zur **notationellen Vereinfachung** werden wir manchmal eine Interpretation (D, I) durch einen Buchstaben \mathcal{D} bezeichnen und schreiben:

$$\begin{array}{ll} val_{\mathcal{D},\beta} & \text{für } val_{D,I,\beta}. \\ \mathcal{D} \models A[a_1, \dots, a_n] & \text{für } val_{\mathcal{D},\beta}(A) = W, \text{ wobei } \beta(x_i) = a_i \text{ und} \\ & x_1, \dots, x_n \text{ die in } A \text{ frei vorkommenden Variablen sind, oder eine Obermenge davon} \\ t^{\mathcal{D}}[a_1, \dots, a_n] & \text{für } val_{\mathcal{D},\beta}(t) \text{ entsprechend} \end{array}$$

Wir gewinnen in dieser Notation an Übersichtlichkeit durch die Einsparung von β , müssen aber voraussetzen, daß die Zuordnung, welches der in eckigen Klammern angeführten Elemente als Belegung für welche Variable dient, aus dem Kontext ersichtlich ist. Wir belegen in der Regel die Variablen in ihrer alpha-numerischen Reihenfolge.

Das folgende Lemma faßt einige offensichtliche Konsequenzen aus der Definition der Auswertungsfunktion val zusammen. Zum Beispiel hängt der Wert von $val_{\mathcal{D},\beta}(A)$ nicht ab von den Funktionswerten $\beta(x)$ für Variable x , die in A nicht frei vorkommen. Diese Tatsachen werden im folgenden stillschweigend benutzt. Ein ähnliches Lemma ließe sich formulieren über die Unabhängigkeit von $val_{D,I,\beta}(A)$ von den Werten $I(P)$ für Prädikate P , die in A nicht vorkommen. Wir verzichten darauf und vertrauen der natürlichen Intelligenz des Lesers solche und ähnliche Offensichtlichkeiten im folgenden zu erkennen.

Lemma 4.30 (Koinzidenzlemma)

\mathcal{D} sei Interpretation über Σ .

1. Gilt für den Term t und die Variablenbelegungen β, γ , daß

$$\beta(x) = \gamma(x) \text{ für alle } x \in Var(t),$$

dann

$$val_{\mathcal{D},\beta}(t) = val_{\mathcal{D},\gamma}(t).$$

2. Gilt für die Formel A und die Variablenbelegungen β, γ , daß

$$\beta(x) = \gamma(x) \text{ für alle } x \in Frei(A),$$

dann

$$val_{\mathcal{D},\beta}(A) = val_{\mathcal{D},\gamma}(A).$$

3. Ist $A \in For_{\Sigma}$ geschlossen, dann gilt

$$val_{\mathcal{D},\beta}(A) = val_{\mathcal{D},\gamma}(A)$$

für alle Belegungen β, γ , d. h. der Wert von A hängt nicht von der Belegung ab.

Beweis: Durch strukturelle Induktion unter Ausnutzung der Definition von *val*. ■

Für die beiden folgenden Beispiel legen wir das arithmetische Signatur fest Σ_{arith} bestehend aus Zeichen für die Addition (+), Multiplikation (*), Ordnungsrelation (\leq).

Beispiel 4.31

Die Struktur \mathcal{Z} stehe für die gewohnte Interpretation der Signatur Σ_{arith} mit Universum \mathbb{Z} und der Additionsfunktion ($+_{\mathcal{Z}}$), Multiplikationsfunktion ($*_{\mathcal{Z}}$) und der Ordnungsrelation ($\leq_{\mathcal{Z}}$), zusammenfassend: $\mathcal{Z} = (\mathbb{Z}, +_{\mathcal{Z}}, *_{\mathcal{Z}}, \leq_{\mathcal{Z}})$.

Beispiel 4.32

Die Struktur \mathcal{Z}_{Jint} steht für den Java Datentyp `int`. Das Universum \mathbb{Z}_{Jint} besteht aus den ganzen zwischen *minInt* und *maxInt*, d.h. \mathbb{Z}_{Jint} ist das Intervall $[minInt, maxInt] = [-2147483648, 2147483647]$. Die Ordnungsrelation \leq_{Jint} ist die Einschränkung von $\leq_{\mathcal{Z}}$ auf das Intervall \mathbb{Z}_{Jint} . Liegt das Ergebnis einer Addition oder Multiplikation in \mathcal{Z}_{Jint} im Intervall $[minInt, maxInt]$, dann stimmt es mit dem entsprechenden Ergebnis in \mathcal{Z} überein. Anderenfalls wird modulo 4294967296 gerechnet. Diese Zahl ist die Länge des Intervalls $[minInt, maxInt]$ und damit gleich $-2 * minInt$. Präzise gilt

$$\begin{aligned} x +_{Jint} y &= x +_{\mathcal{Z}} y && \text{falls } minInt \leq x +_{\mathcal{Z}} y \leq maxInt \\ &mod_{Jint}(x +_{\mathcal{Z}} y) && \text{sonst} \\ x *_{Jint} y &= x *_{\mathcal{Z}} y && \text{falls } minInt \leq x *_{\mathcal{Z}} y \leq maxInt \\ &mod_{Jint}(x *_{\mathcal{Z}} y) && \text{sonst} \end{aligned}$$

wobei

$$mod_{Jint}(z) = (z - minInt) mod(-2 * minInt) + minInt$$

Anschaulich kann man sich vorstellen, daß in \mathcal{Z}_{Jint} nach Erreichen von $maxInt$ mit $minInt$ weitergezählt wird, die ganzen Zahlen in Java sind also in gewisser Weise zyklisch. Daß diese anschauliche Beschreibung mit der mathematischen übereinstimmt bestätigt die Rechnung:

$$\begin{aligned}
maxInt +_{Jint} 1 &= mod_{Jint}(maxInt + 1) \\
&= (maxInt + 1 - minInt) mod(-2 * minInt) + minInt \\
&= (-2 * minInt) mod(-2 * minInt) + minInt \\
&= 0 + minInt \\
&= minInt.
\end{aligned}$$

Hier sind einige Beispiel, welche die beiden arithmetischen Strukturen miteinander vergleichen:

Formel ϕ	$\mathcal{Z} \models \phi$	$\mathcal{Z}_{jint} \models \phi$
$\forall x \exists y (x < y)$	ja	nein
$\forall x, y ((x + 1) * y = x * y + y)$	ja	ja
$\exists x (0 < x \wedge x + 1 < 0)$	nein	ja

Die folgenden beiden Lemmata sind technischer Natur, spielen aber eine Schlüsselrolle bei Beweisen durch strukturelle Induktion. Darüberhinaus sagen sie Grundsätzliches über die Substitution aus. Es geht um folgendes: Der Wert, der bei der Auswertung eines Terms oder einer Formel einer freien Variablen x zukommt, kann auf zwei Arten beeinflußt werden. Man kann durch eine Substitution σ an die Stelle der freien Variablen x einen Term $\sigma(x)$ setzen, oder man kann die Variablenbelegung β an der Stelle x modifizieren. Die Substitutionslemmata beschreiben den Zusammenhang zwischen diesen beiden Möglichkeiten.

Lemma 4.33 (Substitutionslemma für Terme)

Σ sei eine Signatur, \mathcal{D} eine Interpretation für Σ , β eine Belegung, σ eine Substitution und $t \in Term_{\Sigma}$. Dann gilt

$$val_{\mathcal{D},\beta}(\sigma(t)) = val_{\mathcal{D},\beta'}(t).$$

wobei $\beta'(x) = val_{\mathcal{D},\beta}(\sigma(x))$ für alle $x \in Var$.

Beweis

Strukturelle Induktion nach t .

$$t = x \in Var:$$

$$\begin{aligned}
val_{\mathcal{D},\beta}(\sigma(x)) &= \beta'(x) && \text{nach Def. von } \beta' \\
&= val_{\mathcal{D},\beta'}(x) && \text{nach Def. von } val \text{ für Variable}
\end{aligned}$$

$$\begin{aligned}
t &= f(t_1, \dots, t_n): \\
val_{\mathcal{D}, \beta}(\sigma(f(t_1, \dots, t_n))) & \\
&= val_{\mathcal{D}, \beta}(f(\sigma(t_1), \dots, \sigma(t_n))) \\
&= I(f)(val_{\mathcal{D}, \beta}(\sigma(t_1)), \dots, val_{\mathcal{D}, \beta}(\sigma(t_n))) \\
&= I(f)(val_{\mathcal{D}, \beta'}(t_1), \dots, (val_{\mathcal{D}, \beta'}(t_n))) \text{ (nach Induktionsannahme)} \\
&= val_{\mathcal{D}, \beta'}(f(t_1, \dots, t_n)).
\end{aligned}$$

■

Lemma 4.34 (Substitutionslemma für Formeln)

Σ sei eine Signatur, \mathcal{D} eine Interpretation für Σ , β eine Belegung, $A \in For_{\Sigma}$ und σ eine für A kollisionsfreie Substitution. Dann gilt:

$$val_{\mathcal{D}, \beta}(\sigma(A)) = val_{\mathcal{D}, \beta'}(A),$$

wobei $\beta'(x) = val_{\mathcal{D}, \beta}(\sigma(x))$ für alle $x \in Var$.

Beweis: Geschieht durch strukturelle Induktion nach A . Wir führen hier exemplarisch den Induktionsschritt von A nach $\exists xA$ vor. Wir schreiben val_{β} anstelle von $val_{\mathcal{D}, \beta}$. Außerdem sei σ_x definiert durch $\sigma_x(x) = x$, $\sigma_x(y) = \sigma(y)$ für $y \neq x$.

$$\begin{aligned}
val_{\beta}(\sigma(\exists xA)) &= W \\
\text{gdw } val_{\beta}(\exists x\sigma_x(A)) &= W && \text{Anwendung von } \sigma \\
\text{gdw } val_{\beta_x^d}(\sigma_x(A)) &= W \text{ für ein } d \in D && \text{Def. von } val \\
\text{gdw } val_{(\beta_x^d)''} (A) &= W && \text{Ind.Voraussetz.} \\
&&& \text{wo } (\beta_x^d)''(y) = val_{\beta_x^d}(\sigma_x(y)) \text{ für all } y. \\
\text{gdw } val_{(\beta')_x^d} (A) &= W && \text{Lücke} \\
\text{gdw } val_{\beta'}(\exists xA) &= W && \text{Def. von } val
\end{aligned}$$

Man beachte, daß die Induktionsvoraussetzung für die Formel A mit der Variablenbelegung β_x^d und der Substitution σ_x benutzt wird. Der Beweis wird vollständig geführt sein, wenn wir die Lücke

$$(\beta_x^d)'' = (\beta')_x^d$$

schließen können. Wir müssen für jede Variable $y \in Frei(A)$ zeigen $(\beta_x^d)''(y) = (\beta')_x^d(y)$.

$y = x$:

$$\begin{aligned}
(\beta_x^d)''(x) &= val_{\beta_x^d}(\sigma_x(x)) && \text{Def. von } (\beta_x^d)'' \\
&= val_{\beta_x^d}(x) && \text{Def. von } \sigma_x \\
&= \beta_x^d(x) && \text{Def. von } val \text{ für Variable} \\
&= d && \text{Def. der modifizierten Belegung} \\
&= (\beta')_x^d(x) && \text{Def. der modifizierten Belegung}
\end{aligned}$$

$$\begin{aligned}
& y \neq x, y \text{ frei in } A: \\
(\beta_x^d)''(y) &= \text{val}_{\beta_x^d}(\sigma_x(y)) && \text{Def. von } (\beta_x^d)'' \\
&= \text{val}_{\beta_x^d}(\sigma(y)) && \text{Def. von } \sigma_x \\
&= \text{val}_{\beta}(\sigma(y)) && \text{da } x \text{ nicht in } \sigma(y) \text{ vorkommt} \\
& && \text{Kollisionsfreiheit von } \sigma \\
&= \beta'(y) && \text{Def. von } \beta' \\
&= (\beta')_x^d(y) && \text{Def. der modifizierten Belegung}
\end{aligned}$$

■

Wir beenden das Thema „Substitutionslemma“ mit zwei Anwendungsbeispielen.

Beispiel 4.35 (Hoare-Kalkül)

Die Zuweisungsregel im Hoare-Kalkül ([dB80], Seite 38) lautet:

$$\{\{x/s\}A\} x := s \{A\}$$

wobei die Substitution kollisionsfrei sein muß und ausdrücken soll, daß ausgehend von einem Zustand, in dem die Formel $\{x/s\}A$ wahr ist, nach Ausführung der Programmstücks $x := s$ ein Zustand erreicht wird, in dem die Formel A gilt. Die Struktur, in welcher die Wahrheit der Formeln auszuwerten ist, wird in der Regel nicht explizit angegeben. Sie ergibt sich aus dem Kontext und legt die übliche Bedeutung der verwendeten Datentypen fest, z.B. der natürlichen Zahlen, der Listen von Zahlen oder der Bäumen samt der üblichen Funktionen und Relationen auf diesen Datentypen. Wir nennen diese Hintergrund-Interpretation \mathcal{H} . Ein Zustand ist eindeutig bestimmt durch die augenblickliche Belegung der Programmvariablen und entspricht somit unserem Begriff einer Variablenbelegung. Wir betrachten eine Variablenbelegung (einen Zustand) β , in dem $\text{val}_{\mathcal{H},\beta}(\{x/s\}A) = W$ gilt. Die Voraussetzungen für die Anwendung der Zuweisungsregel ist also erfüllt. Nach Ausführung der Zuweisung $x := s$ wird ein Zustand β' erreicht mit

$$\beta'(y) := \begin{cases} \text{val}_{\mathcal{H},\beta}(s) & \text{falls } x = y \\ \beta(y) & \text{sonst} \end{cases}$$

Behauptet wird, daß in dem neuen Zustand, β' , die Formel A gilt, oder in formaler Notation aufgeschrieben, daß $\text{val}_{\mathcal{H},\beta'}(A) = W$. Ein genauer Vergleich zeigt, daß das gerade die Aussage des Substitutionslemmas für Formeln ist für die Substitution $\sigma = \{x/s\}$.

Das zweite Anwendungsbeispiel tritt im Beweis des folgenden Lemmas auf:

Lemma 4.36

Sei Σ eine Signatur, \mathcal{D} eine Interpretation für Σ , β eine Belegung und σ eine für A kollisionsfreie Substitution mit $\sigma(y) = y$ für alle Variablen $y \neq x$, dann gilt:

- $val_{\mathcal{D},\beta}(\forall x A \rightarrow \sigma(A)) = W$
- $val_{\mathcal{D},\beta}(\sigma(A) \rightarrow \exists x A) = W$.

Beweis

Wir nehmen an, daß $val_{\mathcal{D},\beta}(\forall x A) = W$ gilt, d.h.

$$val_{\mathcal{D},\beta_x^d}(A) = W \text{ für alle } d \in D.$$

Zu zeigen ist

$$val_{\mathcal{D},\beta}(\sigma(A)) = W$$

Nach dem Substitutionslemma ist das gleichbedeutend mit

$$val_{\mathcal{D},\beta'}(A) = W$$

wobei

$$\beta'(y) = val_{\mathcal{D},\beta}(\sigma(y)) = \begin{cases} \beta(y) & \text{falls } x \neq y \\ val_{\mathcal{D},\beta}(\sigma(x)) & \text{falls } y = x \end{cases}$$

Also $\beta' = \beta_x^d$ für $d = val_{\mathcal{D},\beta}(\sigma(x))$.

Die zweite Aussage läßt sich analog beweisen. ■

4.3.2 Allgemeingültigkeit, logische Äquivalenz

Wir wollen uns bei der Behandlung der logischen Folgerbarkeit beschränken auf Formeln, die keine freien Variablen enthalten. Das ist mit Abstand der typischste Anwendungsfall. Der Fall mit freien Variablen wird in den Übungsaufgaben 4.3.6 bis 4.3.9 ausführlich behandelt.

Wenn wir im folgenden keine genauen Abgaben machen ist jede Formel als eine Formel ohne freie Variablen zu verstehen.

Definition 4.37 (Modell)

Eine Interpretation \mathcal{D} über Σ heißt **Modell** einer Formel A ohne freie Variablen über Σ , wenn gilt $val_{\mathcal{D}}(A) = W$.

\mathcal{D} heißt **Modell** einer Formelmenge M ohne freie Variable, wenn für jede Formel $B \in M$ gilt $val_{\mathcal{D}}(B) = W$.

Definition 4.38 (Folgerbarkeit)

Es sei M eine Menge von Formeln aus For_Σ und A eine einzelne Formel aus For_Σ , wobei weder in M noch in A freie Variablen vorkommen.

$$M \models_\Sigma A \quad :\Leftrightarrow \quad \text{Jedes Modell von } M \text{ ist auch Modell von } A.$$

Lies: **Aus M folgt A** (über Σ).

Wir werden meistens kurz \models statt \models_Σ schreiben. Ferner stehe wieder $\models A$ für $\emptyset \models A$ und $B \models A$ für $\{B\} \models A$.

Definition 4.39

$A \in For_\Sigma$ ohne freie Variablen heißt

- **allgemeingültig** gdw $\models A$
- **erfüllbar** gdw $\neg A$ ist nicht allgemeingültig.

Lemma 4.40

Sei A eine variablenfreie Formel.

1. Die folgenden Aussagen sind äquivalent:
 - (a) A allgemeingültig
 - (b) Jede Interpretation \mathcal{D} ist Modell von A .
 - (c) $val_{\mathcal{D},\beta}(A) = W$ für alle Interpretationen \mathcal{D} .
2. Die folgenden Aussagen sind äquivalent:
 - (a) A erfüllbar
 - (b) Es gibt eine Interpretation \mathcal{D} mit $val_{\mathcal{D}}(A) = W$
3. Falls M keine freien Variablen enthält gilt:
 $M \cup \{\neg A\}$ ist nicht erfüllbar gdw $M \models A$.

Beweis: Übung

Wir werden im folgenden viele Beweise allgemeingültiger Formeln kennenlernen. Ein besonders einfacher Fall liegt vor, wenn die Allgemeingültigkeit „schon allein aufgrund der aussagenlogischen Struktur besteht“:

Definition 4.41

(Tautologie) $A \in For_\Sigma$ heißt *Tautologie*, wenn es eine endliche aussagenlogische Signatur $\Sigma' = \{P_0, \dots, P_{n-1}\}$, ein $A' \in For_{\Sigma'}$ und Formeln $A_0, \dots, A_{n-1} \in For_\Sigma$ gibt, so daß

- A' ist (aussagenlogisch) allgemeingültig über Σ'
- A entsteht aus A' , indem man dort P_i durch A_i ersetzt (für $i = 0, \dots, n-1$).

Beispiel 4.42

(wir lassen keine Klammern weg):

$$((((p(x) \wedge \neg q(y, c, x)) \rightarrow p(c)) \wedge (\neg(p(x) \wedge \neg q(y, c, x)) \rightarrow p(c))) \rightarrow p(c))$$

ist eine Tautologie, denn

$$((P_0 \rightarrow P_1) \wedge (\neg P_0 \rightarrow P_1)) \rightarrow P_1$$

ist aussagenlogisch allgemeingültig, und hieraus entsteht durch Ersetzen von P_0 durch $(p(x) \wedge \neg q(y, c, x))$
 P_1 durch $p(c)$
 die Ausgangsformel.

Lemma 4.43

Jede Tautologie ist allgemeingültig.

Beweis

Zur Tautologie A seien A_0, \dots, A_{n-1} Teilformeln, wie in der Definition festgelegt. Es sei D, I, β gegeben. Gemäß der Definition der A_i ist dann $val_{D,I,\beta}(A)$ eindeutig festgelegt durch die Werte $val_{D,I,\beta}(A_i)$, $i = 0, \dots, n-1$, und ist für jede Verteilung dieser Werte W .

■

Besonders wichtig jedoch sind – wie in der Aussagenlogik – allgemeingültige Formeln der Gestalt $A \leftrightarrow B$.

Definition 4.44

$A, B \in For_\Sigma$ heißen *logisch äquivalent* gdw $\models A \leftrightarrow B$ (d. h. $A \leftrightarrow B$ ist allgemeingültig)

Wie es wünschenswert ist und wie wir gleich festhalten, ist logische Äquivalenz eine Kongruenz auf For_Σ .

Korollar 4.45

Logische Äquivalenz ist eine Äquivalenzrelation auf For_Σ . Sie ist darüberhinaus eine Kongruenz, d. h.: gilt

$$\models A \leftrightarrow A', \models B \leftrightarrow B'$$

dann auch

$$\models \neg A \leftrightarrow \neg A'$$

$$\models (A \text{ op } B) \leftrightarrow (A' \text{ op } B') \text{ für } op \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$$

$$\models QxA \leftrightarrow QxA' \text{ für } x \in Var, Q \in \{\forall, \exists\}$$

Beweis

Unmittelbar aus der Definition. ■

Korollar 4.46 (Ersetzungstheorem)

Es seien $A, A', B, B' \in For_\Sigma$ mit

- B ist Teilformel von A
- B ist logisch äquivalent zu B'
- A' entsteht aus A , indem man dort an irgendwelchen Stellen (nicht notwendig überall) B durch B' ersetzt.

Dann ist A logisch äquivalent zu A' .

Beweis

Klar nach dem letzten Korollar. ■

Satz 4.47

Für $x, y \in Var, A, B \in For_\Sigma$ und $Q \in \{\forall, \exists\}$ sind logisch äquivalente Formelpaare:

1. A und B , wenn $A \leftrightarrow B$ Tautologie ist,
2. $\neg\forall xA$ und $\exists x\neg A$, $\neg\exists xA$ und $\forall x\neg A$
3. $\forall x\forall yA$ und $\forall y\forall xA$, $\exists x\exists yA$ und $\exists y\exists xA$

4. $\forall x(A \wedge B)$ und $\forall xA \wedge \forall xB$
5. $\exists x(A \vee B)$ und $\exists xA \vee \exists xB$
6. QxA und $Qy\{x/y\}(A)$, falls $\{x/y\}$ kollisionsfrei ist für A und $y \notin \text{Frei}(A)$
7. $A \wedge QxB$ und $Qx(A \wedge B)$, falls $x \notin \text{Frei}(A)$
8. $QxA \wedge B$ und $Qx(A \wedge B)$, falls $x \notin \text{Frei}(B)$
9. $A \vee QxB$ und $Qx(A \vee B)$, falls $x \notin \text{Frei}(A)$
10. $QxA \vee B$ und $Qx(A \vee B)$, falls $x \notin \text{Frei}(B)$
11. $A \rightarrow \forall xB$ und $\forall x(A \rightarrow B)$, falls $x \notin \text{Frei}(A)$
12. $\forall xA \rightarrow B$ und $\exists x(A \rightarrow B)$, falls $x \notin \text{Frei}(B)$
13. $A \rightarrow \exists xB$ und $\exists x(A \rightarrow B)$, falls $x \notin \text{Frei}(A)$
14. $\exists xA \rightarrow B$ und $\forall x(A \rightarrow B)$, falls $x \notin \text{Frei}(B)$

Man bemerke, daß diese Liste teilweise redundant ist (5 ergibt sich aus 4 etc.). Wir haben sie absichtlich in dieser Reichhaltigkeit aufgeschrieben. Die Äquivalenz 6 nennt man *gebundene Umbenennung*.

Beweis

1 wissen wir schon, 2 bis 5 rechnet man unmittelbar nach. Man mache 6 als Übung (wobei man das Substitutionslemma verwende); ohnehin wird die Allgemeingültigkeit der gebundenen Umbenennung aus späteren Resultaten folgen. Stellvertretend für 7 bis 14 zeigen wir 11. (Die anderen Aussagen dieser Gruppe folgen hieraus oder werden entsprechend bewiesen.)

Es ist zu zeigen, daß für alle D, I, β gilt:

$$\text{val}_{D,I,\beta}(A \rightarrow \forall xB) = \text{val}_{D,I,\beta}(\forall x(A \rightarrow B))$$

wenn $x \notin \text{Frei}(A)$. Falls $\text{val}_{D,I,\beta}(A \rightarrow \forall xB) = W$, hat man $\text{val}_{D,I,\beta}(\forall x(A \rightarrow B)) = W$ unmittelbar aus der Definition von $\text{val} \dots$ (Übung).

Sei umgekehrt $\text{val}_{D,I,\beta}(\forall x(A \rightarrow B)) = W$, d. h.

1. Für alle $d \in D$: $(\text{val}_{D,I,\beta_x^d}(A) = W \Rightarrow \text{val}_{D,I,\beta_x^d}(B) = W)$.
Angenommen nun, es wäre $\text{val}_{D,I,\beta}(A \rightarrow \forall xB) = F$. Dann gilt also
2. $\text{val}_{D,I,\beta}(A) = W$
3. $\text{val}_{D,I,\beta}(\forall xB) = F$, d. h.: es gibt ein $e \in D$ mit $\text{val}_{D,I,\beta_x^e}(B) = F$.
Da $x \notin \text{Frei}(A)$, liefern 2 und das Koinzidenzlemma für dieses e :

4. $val_{D,I,\beta_x^e}(A) = W$.
 Aus 4 und 1 hat man
5. $val_{D,I,\beta_x^e}(B) = W$
 im Widerspruch zu 3.

■

4.3.3 Übungsaufgaben

Übungsaufgabe 4.3.1

Ist $x \notin Frei(A)$, dann

$$val_{D,I,\beta}(A) = val_{D,I,\beta^d}(A) \text{ für alle } d \in D.$$

Insbesondere: Ist A geschlossen, dann

$$val_{D,I,\beta}(A) = val_{D,I,\beta^d}(A) \text{ für alle } x \in Var \text{ und } d \in D.$$

Übungsaufgabe 4.3.2

Geben Sie eine geschlossene prädikatenlogische Formel A an, für die gilt:

Jedes Modell (D, I) von A hat einen unendlich großen Grundbereich D .

Übungsaufgabe 4.3.3

Der Beweis von Teil 2 aus Lemma 4.30 wird durch strukturelle Induktion über A geführt. Beweisen Sie den Induktionsschritt von B auf $\forall xB$.

Übungsaufgabe 4.3.4

Für beliebige Mengen P, Q, R, S gilt:

$$\begin{array}{l} S \cap Q = \emptyset \\ P \subseteq Q \cup R \\ P = \emptyset \Rightarrow Q \neq \emptyset \\ Q \cup R \subseteq S \end{array} \quad \Rightarrow \quad P \cap R \neq \emptyset$$

Geben Sie eine geschlossene prädikatenlogische Formel A über der Signatur

$$(\emptyset, \{p, q, r, s\}, \alpha(p) = \alpha(q) = \alpha(r) = \alpha(s) = 1)$$

an, die diesen Sachverhalt modelliert, d.h. jede Interpretation, die die Extensionen von p, q, r, s als Mengen interpretiert, ist ein Modell von A .

Übungsaufgabe 4.3.5 (First-Order n -Damen Problem)

Auf einem $n \times n$ Felder großen Schachbrett sind n Damen so zu verteilen, daß keine eine andere schlagen kann.

Finden Sie geschlossene prädikatenlogische Formeln q_n so, daß für alle $n \geq 1$ gilt:

$$\begin{array}{c} q_n \text{ ist erfüllbar} \\ \text{gdw} \\ \text{das } n\text{-Damen Problem hat wenigstens eine Lösung.} \end{array}$$

Versuchen Sie eine Lösung zu finden, bei der die Länge von q_n linear mit n wächst.

Übungsaufgabe 4.3.6

Ist M eine Menge von Formeln, in denen auch freie Variablen vorkommen können und A eine Formel, die ebenfalls freie Variablen enthalten kann. Es gibt zwei Möglichkeiten die Definition der logischen Folgerbarkeit, Def.4.38, auf diesen Fall auszudehnen.

Möglichkeit 1 Man erweitert den Modellbegriff auf diese Situation

Definition 4.48 (Modell)

Eine Interpretation \mathcal{D} über Σ heißt **Modell** einer Formel A über Σ , wenn für jedes β gilt $val_{\mathcal{D},\beta}(A) = W$.

\mathcal{D} heißt **Modell** einer Formelmenge M , wenn für jedes β und jede Formel $B \in M$ gilt $val_{\mathcal{D},\beta}(B) = W$.

Möglichkeit 2 Wir bilden den *universellen Abschluß* $Cl_{\forall}(A)$ einer Formel (A) mit freien Variablen. Sind x_1, \dots, x_n alle freien Variablen in A , dann ist $Cl_{\forall}(A) = \forall x_1 \dots \forall x_n A$. Mit $Cl_{\forall}(M)$ bezeichnen wir die Menge $\{Cl_{\forall}(B) \mid B \in M\}$. Jetzt kann die Definition $M \models A$ zurückführen auf den variablenfreien Fall durch $Cl_{\forall}(M) \models Cl_{\forall}(A)$.

Zeigen Sie, daß die beiden Möglichkeiten zur selben Folgerungsrelation führen.

Übungsaufgabe 4.3.7

Die folgende Aufgabe ist nur sinnvoll, wenn in A freie Variablen vorkommen. Es soll die Definition 4.48 benutzt werden.

Wenn A ein Modell besitzt, dann ist A erfüllbar. Zeigen Sie, daß die Umkehrung nicht richtig sein muß.

Übungsaufgabe 4.3.8

Wie die vorangegangene Aufgabe 4.3.7 ist auch diese nur sinnvoll, wenn in A freie Variablen vorkommen.

Wenn zwei Formeln logisch äquivalent sind, dann haben sie dieselben Modelle. Geben Sie zwei Formeln an, die dieselben Modelle haben, aber nicht logisch äquivalent sind.

Übungsaufgabe 4.3.9

In der einschlägigen Literatur findet sich gelegentlich eine zweite Definition der Folgerbarkeitsbeziehung von Formeln mit freien Variablen, die von der Definition 4.3.6. Da meist explizit oder stillschweigend angenommen wird, daß in Aussagen über Ableitbarkeit keine freien Variablen vorkommen, findet man in der Regel in diesen Texten keinen Hinweis auf die zwei Varianten der Folgerbarkeitsbeziehung. Hier jetzt die zweite Definition der Folgerbarkeitsbeziehung:

Definition 4.49 (Lokale Folgerbarkeit)

Es sei $M \subseteq For_\Sigma$, $A \in For_\Sigma$.

$M \models_\Sigma^\circ A$ \Leftrightarrow Für jede Interpretation von \mathcal{D} und jede Belegung β gilt
wenn $val_{\mathcal{D},\beta}(M) = W$,
dann gilt auch $val_{\mathcal{D},\beta}(A) = W$.

Wenn eine verbale Unterscheidung nötig ist, nennen wir die in Definition 4.38 definierte Relation \models die **globale** und \models° die **lokale** Folgerungsbeziehung.

Zeigen Sie:

1. \models° schärfer als \models : $M \models^\circ A \Rightarrow M \models A$, aber i. a. nicht umgekehrt.
2. $A \models^\circ B \Leftrightarrow \models^\circ A \rightarrow B \Leftrightarrow \models A \rightarrow B$
3. $\models A \rightarrow B \Rightarrow A \models B$ aber i. a. nicht umgekehrt.
4. Ist M variablenfrei, dann gilt für alle A :

$$M \models^\circ A \Leftrightarrow M \models A$$

Übungsaufgabe 4.3.10

Diese Aufgabe soll zeigen, wie sich die beiden Folgerungsrelationen auf den variablenfreien Fall zurückführen lassen.

1. $A \models B$ gdw $Cl_{\forall} A \models^\circ Cl_{\forall} B$
2. $A \models^\circ B$ gdw $\bar{A} \models \bar{B}$
Seien x_1, \dots, x_k alle freien Variablen in A und B und c_1, \dots, c_k neue paarweise verschiedene Konstanten. \bar{A} entsteht aus A , indem jede Variable x_i durch c_i ersetzt wird. Entsprechend entsteht \bar{B} aus B .

Übungsaufgabe 4.3.11

Beweisen Sie die folgende Behauptung:

A und B sind logisch äquivalent

gdw Für alle \mathcal{D} und β gilt $val_{\mathcal{D},\beta}(A) = val_{\mathcal{D},\beta}(B)$.

gdw $A \models^{\circ} B$ und $B \models^{\circ} A$

Übungsaufgabe 4.3.12

1. Finden Sie eine Formel ϕ_3 der Prädikatenlogik erster Stufe mit leeren Vokabular, so daß $\mathcal{M} \models \phi_3$ genau dann gilt, wenn \mathcal{M} genau drei Elemente hat.
Die Formel ϕ_3 enthält also als einziges Relationszeichen das Symbol \doteq für die Gleichheit.
2. Geben Sie ϕ_n für beliebige $n \geq 1$ an.

Übungsaufgabe 4.3.13

Definition 4.50

Sei A eine Formel der PL1 ohne freie Variablen. Das **Spektrum** von A , $spec(A)$ ist die folgende Menge natürlicher Zahlen:

$$spec(A) = \{n \in \mathbb{N} : \text{es gibt ein Modell } \mathcal{D} \text{ von } A, \\ \text{dessen Universum genau } n \text{ Elemente hat}\}$$

Es kann A auch noch unendliche Modelle besitzen, aber diese spielen für die Definition des Spektrums keine Rolle.

Nach Aufgabe 4.3.12 ist klar, daß es für jede endliche Teilmenge $E \subset \mathbb{N}$ eine erster Stufe Formel Sp_E gibt mit $spec(Sp_E) = E$. Finden Sie

1. eine Formel A mit $spec(A) =$ die Menge aller geraden Zahlen.
2. eine Formel B mit $spec(B) =$ die Menge aller Quadratzahlen.
3. eine Formel C mit $spec(C) =$ die Menge aller Zahlen, die keine Primzahlen sind.

In allen drei Fällen steht Ihnen die Wahl einer passenden Signatur völlig frei. Kommentar: Das Spektrumsproblem, also die Frage welche Teilmengen von \mathbb{N} können als Spektren von Formeln erster Stufe auftreten, hat in der mathematischen Logik und theoretischen Informatik viel Interesse gefunden. Eine komplexitätstheoretische Charakterisierung wurde bereits 1974 von Jones und Selman gefunden [JS74]: Die Klasse *SPEC* der Spektren stimmt überein mit der Klasse der Teilmengen von \mathbb{N} die von einer nichtdeterministischen Turingmaschine in 2^{ct} Schritten erkannt werden können, wobei c

eine Konstante und t die Länge der Eingabe ist. Dabei wird eine natürliche Zahl n in Binärdarstellung eingegeben, also $t \approx \log_2(n)$.

Die Frage ob *SPEC* unter mengentheoretischen Komplementen abgeschlossen ist, Vermutung von Asser, ist bis heute ungelöst.

Übungsaufgabe 4.3.14

Finden Sie Formeln A und B , die freie Variablen enthalten können, so daß $A \models B$ gilt aber nicht $\models A \rightarrow B$.

Übungsaufgabe 4.3.15

Sei $\phi(x, y)$ die Abkürzung für die Formel

$$0 \leq y \wedge 0 \leq x \wedge x * x \leq y \wedge y < (x + 1) * (x + 1).$$

Für ganze Zahlen a, b gilt $\mathcal{Z} \models \phi[a, b]$ genau dann wenn a positiv ist und b die positive ganzzahlige Quadratwurzel von a ist. So ist z.B. 2 die positive ganzzahlige Quadratwurzel von 5.

Geben Sie ein Beispiel, daß diese Aussage für $\mathcal{Z}_{jint} \models \phi[a, b]$ nicht mehr richtig ist.

Zur Definition von \mathcal{Z} , \mathcal{Z}_{jint} siehe Beispiele 4.31 und 4.32 auf Seite 129.

Die Notation $\mathcal{Z} \models \phi[a, b]$ ist eine abkürzende Schreibweise für $\mathcal{Z}, \beta \models \phi$ mit $\beta(x) = a$ und $\beta(y) = b$, siehe Seite 128.

Die Formel ϕ war, in anderer Notation, in [LBR03] als Spezifikation für die ganzzahlige Quadratwurzel benutzt worden. Auf den Fehler in der Spezifikation wurde in [Cha03] aufmerksam gemacht.

Übungsaufgabe 4.3.16

In vielen Programmiersprachen gibt es die Möglichkeit Datenwerte abhängig von einer Bedingung zu referenzieren, z.B. $v = \text{if } x = 0 \text{ then } 5 \text{ else } w$. Konstrukte der Art $\text{if } \phi \text{ then } t_1 \text{ else } t_2$ nennt man *bedingte Terme* (*conditional terms*). In der Definition 4.3 von Seite 109 kommen bedingte Terme nicht vor. Ihre Einführung würde sogar zu einem völlig neuen Phänomän führen: bisher konnten Terme in Formeln vorkommen, aber umgekehrt keine Formeln in Termen. Das passiert aber in $\text{if } \phi \text{ then } t_1 \text{ else } t_2$. In dieser Aufgabe wollen wir die die formale Einführung bedingter Terme untersuchen. Für die Zwecke dieser Übungsaufgabe erweitern wir die Definition 4.3 um die folgende Zeile

3. Sind t_1, t_2 Terme und ist ϕ eine Formel, so ist $\text{if } \phi \text{ then } t_1 \text{ else } t_2$ ein Term.

Die zugehörige Semantikdefinition 4.27 wird ebenfalls ergänzt. Sei dazu (D, I) eine Interpretation des Vokabulars Σ und β eine Variablenbelegung über D .

$$3. \text{val}_{D,I,\beta}(\text{if } \phi \text{ then } t_1 \text{ else } t_2) = \begin{cases} \text{val}_{D,I,\beta}(t_1) & \text{falls } \text{val}_{D,I,\beta}(\phi) = \mathbf{1} \\ \text{val}_{D,I,\beta}(t_2) & \text{falls } \text{val}_{D,I,\beta}(\phi) = \mathbf{0} \end{cases}$$

Zeigen Sie:

1. Finden Sie eine zu

$$\forall n \forall d (jdiv(n, d) = \text{if } (n \geq 0) \text{ then } div(n, d) \text{ else } -div(-n, d))$$

äquivalente Formel ohne `if then else`.

2. Sei s beliebiger Term in der erweiterten Syntax und occ ein Vorkommen eines bedingten Terms `if ϕ then t_1 else t_2` in s . Mit s_1 , bzw. s_2 bezeichnen wir die Terme, die aus s entstehen, wenn man occ ersetzt durch t_1 , bzw. t_2 . Dann gilt für jede Interpretation (D, I) und Variablenbelegung β

$$\text{val}(s) = \begin{cases} \text{val}(s_1) & \text{falls } \text{val}(\phi) = W \\ \text{val}(s_2) & \text{falls } \text{val}(\phi) = F \end{cases}$$

wobei wir der Kürze halber nur val geschrieben haben anstelle von $\text{val}_{D,I,\beta}$.

3. Sei F eine quantorenfreie Formel und occ ein Vorkommen eines bedingten Terms `if ϕ then t_1 else t_2` in F . Sei F_1 die Formel, die entsteht, wenn man occ in F ersetzt durch t_1 and F_2 die analog entstehende Formel, wenn occ durch t_2 ersetzt wird. Dann gilt

$$F \leftrightarrow (\phi \wedge F_1) \vee (\neg\phi \wedge F_2)$$

4. Zu jeder Formel F in der erweiterten Syntax gibt es eine logisch äquivalente Formel G , die keine bedingten Terme enthält.

Übungsaufgabe 4.3.17

In einem Tutorium zur Vorlesung *Formale System* wurde die folgende Vermutung diskutiert:

Sei F eine Formel, in der genau die Variablen x, y vorkommen. Außerdem wird angenommen, daß in F nur einstellige Prädikatszeichen und keine Funktionszeichen vorkommen. Dann ist

$$\forall x \exists y F \rightarrow \exists y \forall x F$$

allgemeingültig.

Stimmt diese Vermutung?

4.4 Normalformen

Wie in der Aussagenlogik wollen wir die aufgeführten Fälle logischer Äquivalenz benutzen, um für prädikatenlogische Formeln Normalformen herzustellen. Wir beginnen mit der einfachsten.

Definition 4.51 (Negationsnormalform)

Eine Formel $A \in For_\Sigma$ ist in **Negationsnormalform**, wenn jedes Negationszeichen in A vor einer atomaren Teilformel steht.

(Insbesondere kommt keine Teilformel der Form $\neg\neg B$ in A vor.)

Lemma 4.52

Zu jeder Formel A gibt es eine logisch äquivalente Formel B in Negationsnormalform.

Definition 4.53

Eine Formel $A \in For_\Sigma$ heißt *bereinigt*, wenn

- $Frei(A) \cap Bd(A) = \emptyset$
- die hinter Quantoren stehenden Variablen paarweise verschieden sind.

Satz 4.54

Zu jeder Formel A gibt es eine äquivalente bereinigte. Sie läßt sich aus A algorithmisch herstellen.

Beweis

Man wende die gebundene Umbenennung an. ■

Definition 4.55 (Pränexe Normalform)

$A \in For_\Sigma$ hat *Pränex-Normalform*, wenn A die Gestalt hat

$$Q_1x_1 \dots Q_nx_nB$$

mit $Q_i \in \{\forall, \exists\}$, $x_i \in Var$ ($i = 1, \dots, n$) und: B ist eine quantorenfreie Formel. Man nennt B auch die **Matrix** von A .

Satz 4.56

Zu jeder Formel A gibt es eine äquivalente in Pränex-Normalform. Sie läßt sich aus A algorithmisch ableiten.

Beweis

Nachdem man A bereinigt hat, wende man von innen nach außen die Äquivalenzen 7 bis 14 in Satz 4.47 (Seite 136) auf Teilformeln an, um sukzessive Quantoren „nach links zu schieben“.

■

Bemerkungen

Beim geschilderten Verfahren bleibt die Eigenschaft „bereinigt“ erhalten. Man beachte, daß die Pränex-Normalform einer Formel A i. a. nicht eindeutig bestimmt ist. Abhängig von der Reihenfolge der angewandten Äquivalenzen kann man z. B. aus

sowohl $\forall x p(x) \rightarrow \forall y q(y)$
als auch $\exists x \forall y (p(x) \rightarrow q(y))$
erhalten. $\forall y \exists x (p(x) \rightarrow q(y))$

Beispiel 4.57 (Pränex Normalform)

Aus $\forall y (\forall x \forall z p(x, y) \rightarrow \exists x r(x, y))$ erhält man sukzessive:

$$\forall y (\forall x \forall z p(x, z) \rightarrow \exists u r(u, y))$$

$$\forall y (\exists x (\forall z p(x, z) \rightarrow \exists u r(u, y)))$$

$$\forall y \exists x \exists z (p(x, z) \rightarrow \exists u r(u, y))$$

$$\forall y \exists x \exists z \exists u (p(x, z) \rightarrow r(u, y))$$

Eine Formel in Pränex-Normalform läßt sich noch weiter normieren. Für quantorenfreie Formeln können wir in unmittelbarer Übertragung der Definition aus der Aussagenlogik sagen, wann eine solche in *disjunktiver* bzw. *konjunktiver* Normalform ist. Mit Hilfe der Tautologien läßt sich eine Formel in Pränex-Normalform dann in eine äquivalente überführen, deren Matrix in DNF oder KNF ist.

4.4.1 Skolem-Normalform

Wir gehen zum Abschluß auf eine Normalform ein, die noch wesentlich restriktiver ist als etwa die Pränex-Normalform mit Matrix in KNF. Allerdings

kann man jetzt nicht mehr zu beliebig vorgelegter Formel A eine Formel A' in dieser neuen Normalform erhalten, welche logisch äquivalent zu A wäre. Nur eine viel schwächere Äquivalenz zwischen A und A' läßt sich erreichen: A' besitzt genau dann ein Modell, wenn A eines hat. Für das wichtigste Anwendungsgebiet dieser Normierung reicht das aber auch aus: Die Überführbarkeit einer Formel in „Skolem-Normalform“ ist grundlegend für die Theorie des automatischen Beweisens.

Als Vorbereitung auf das nächste Lemma betrachten wir anhand dreier Formeln die alternativen Möglichkeiten einen Sachverhalt unter Benutzung von Existenzquantoren auszudrücken (Beispiel 4.58) oder durch Einführung entsprechender Funktionszeichen (Beispiel 4.59). Das Vokabular Σ sei durch die Relations- und Funktionszeichen $\{<, +, 0, 1\}$ gegeben, die in der Interpretation \mathcal{N} über dem Universum der natürlichen Zahlen in der üblichen Weise interpretiert werden. Mit $\Sigma_1, \Sigma_2, \Sigma_3$ bezeichnen wir die jeweilige Erweiterung von Σ um ein einstelliges Funktionszeichen do , um ein einstelliges Funktionszeichen gr bzw. um ein zweistelliges Funktionszeichen $diff$.

Beispiel 4.58 (Darstellung mit Existenzquantor)

1. $\forall x \exists y (y \doteq x + x)$
2. $\forall x \exists y (x < y)$
3. $\forall x \forall y \exists z (x < y \rightarrow x + z \doteq y)$

Beispiel 4.59 (Darstellung mit Funktionszeichen)

1. $\forall x (do(x) \doteq x + x)$
2. $\forall x (x < gr(x))$
3. $\forall x \forall y (x < y \rightarrow x + diff(x, y) \doteq y)$

Alle drei Formeln aus Beispiel 4.58 sind offensichtlich in der Interpretation \mathcal{N} wahr. Damit die Formeln aus Beispiel 4.59 in den entsprechenden Erweiterungen auf die Signaturen Σ_i , bezeichnet mit $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3$ wahr werden, müssen die neuen Funktionszeichen richtig interpretiert werden:

Beispiel 4.60

1. Dieser Fall ist einfach, da wir für die Definition der Interpretation des Funktionszeichens do in \mathcal{N}_1 keine Wahl haben:
Setze $do^{\mathcal{N}_1}(d) = d + d$

2. Hier gibt es sehr viele Möglichkeiten für die Wahl einer Interpretation von gr . Die einzige Forderung, die erfüllt sein muß ist, daß der Funktionswert grösser als das Argument ist. Also etwa:

$$\text{Setze } gr^{\mathcal{N}_2}(d) = d + 1$$

3. Hier setzen wir etwa:

$$\text{diff}^{\mathcal{N}_3}(d_1, d_2) = \begin{cases} d_2 - d_1 & \text{falls } d_1 < d_2 \\ 0 & \text{sonst} \end{cases}$$

Der Wert im Fall $d_2 \leq d_1$ ist hier willkürlich gewählt, aber das stört nicht die Aussage des Lemmas, denn wenn die Prämisse der Implikation nicht erfüllt ist, dann ist die ganze Formel wahr, unabhängig davon, was rechts des Implikationspfeiles steht.

Lemma 4.61 (Beseitigen von Existenzquantoren)

Gegeben sei eine geschlossene Formel über Σ der Gestalt

$$\forall x_1 \dots \forall x_n \exists y B,$$

in der x_1, \dots, x_n, y paarweise verschieden sind und $x_1, \dots, x_n \notin Bd(B)$ ($n = 0$ ist erlaubt, die Formel ist dann $\exists y B$). Wir erweitern Σ um ein neues, n -stelliges Funktionssymbol g zu einer Signatur Σ_g (d. h. $F_{\Sigma_g} = F_{\Sigma} \cup \{g\}$).

Dann gilt

$$\begin{aligned} \forall x_1 \dots \forall x_n \exists y B \text{ hat ein Modell über } \Sigma \\ \Leftrightarrow \forall x_1 \dots \forall x_n \{y/g(x_1, \dots, x_n)\}(B) \text{ hat ein Modell über } \Sigma_g. \end{aligned}$$

Beweis

\Rightarrow :

(D, I) sei Modell von $\forall x_1 \dots \forall x_n \exists y B$. Das bedeutet: Zu jedem $(d_1, \dots, d_n) \in D^n$ gibt es ein $e \in D$, so daß für jede Variablenbelegung β gilt

$$\text{val}_{D, I, \beta_{x_1, \dots, x_n, y}^{d_1, \dots, d_n, e}}(B) = W$$

(Dabei steht $\beta_{x_1, \dots, x_n, y}^{d_1, \dots, d_n, e}$ kurz für $((\dots (\beta_{x_1}^{d_1}) \dots)_{x_n}^{d_n})_y^e$. Im Falle $n = 0$ lautet die obige Aussage einfach: Es gibt e mit $\text{val}_{D, I, \beta_y^e}(B) = W$ für alle β .)

Es sei $\bar{g} : D^n \rightarrow D$ eine Funktion, die zu jedem (d_1, \dots, d_n) ein solches e festlegt. Also:

$$\text{val}_{D, I, \beta_{x_1, \dots, x_n, y}^{d_1, \dots, d_n, \bar{g}(d_1, \dots, d_n)}}(B) = W$$

für alle (d_1, \dots, d_n) und β .

Zur erweiterten Signatur Σ_g definieren wir die Erweiterung I_g von I durch

$$I_g(g) = \bar{g}.$$

Nach dem Substitutionslemma gilt dann für (D, I_g) , jede Variablenbelegung β und alle $(d_1, \dots, d_n) \in D^n$:

$$\begin{aligned} & \text{val}_{D, I_g, \beta_{x_1, \dots, x_n}^{d_1, \dots, d_n}}(\{y/g(x_1, \dots, x_n)\}(B)) \\ &= \text{val}_{D, I_g, \beta_{x_1, \dots, x_n, y}^{d_1, \dots, d_n, \bar{g}(d_1, \dots, d_n)}}(B) \\ &= \text{val}_{D, I, \beta_{x_1, \dots, x_n, y}^{d_1, \dots, d_n, \bar{g}(d_1, \dots, d_n)}}(B) \\ & \text{da } I_g \text{ und } I \text{ auf den Signatursymbolen in } B \text{ übereinstimmen} \\ &= W \end{aligned}$$

Es folgt, daß (D, I_g) Modell von $\forall x_1 \dots \forall x_n \{y/g(x_1, \dots, x_n)\}(B)$ ist.

\Leftarrow :

Die Formel

$$\forall x_1 \dots \forall x_n \{y/g(x_1, \dots, x_n)\}(B) \rightarrow \forall x_1 \dots \forall x_n \exists y B$$

ist allgemeingültig. (Korollar zum Lemma 4.36 auf Seite 133). Jedes Modell (D, I_g) von $\forall x_1 \dots \forall x_n \{y/g(x_1, \dots, x_n)\}(B)$ (über Σ_g) ist also auch Modell von $\forall x_1 \dots \forall x_n \exists y B$. Da die letztere Formel das neue Symbol g nicht enthält, hat sie auch (D, I) mit

$$I := \text{Einschränkungen von } I_g \text{ auf } F_\Sigma \cup P_\Sigma$$

zum Modell. (D, I) ist Interpretation über Σ . ■

Definition 4.62 (Skolem-Normalform)

Eine Formel ist in *Skolem-Normalform*, wenn sie

- geschlossen ist
- die Gestalt hat $\forall x_1 \dots \forall x_n B$ (nur Allquantoren im Präfix), mit quantorenfreiem B
- die Matrix B in KNF ist.

Satz 4.63

Zu jedem $A \in For_\Sigma$ gibt es eine endliche Erweiterung Σ_{sk} von Σ und eine Formel $A_{sk} \in For_{\Sigma_{sk}}$ mit

- A_{sk} ist in Skolem-Normalform
- A_{sk} hat ein Modell genau dann, wenn A ein Modell hat.

A_{sk} läßt sich aus A algorithmisch erhalten.

Beweis

Nachdem wir A bereinigt und in Pränex-Normalform gebracht haben, bilden wir den Allabschluß und wenden dann – im Präfix von links nach rechts fortschreitend – Lemma 4.61 an, um der Reihe nach alle Existenzquantoren zu beseitigen. Die Matrix der erhaltenen Formel wird durch die bekannten Tautologien in KNF gebracht. ■

Bemerkung

Die bei Anwendung von Lemma 4.61 neu eingeführten Funktionssymbole nennt man *Skolemfunktionen*, im nullstelligen Fall *Skolemkonstanten*. Das Verfahren selber heißt *Skolemisierung*. Man beachte, daß die neue Signatur Σ_{sk} von der vorgelegten Formel A abhängt.

Beispiel 4.64

Gegeben:

$$\forall x(\exists y(p(y)) \wedge \exists z(q(x, z)))$$

Pränex Normalform:

$$\forall x \exists y \exists z (p(y) \wedge q(x, z))$$

Skolem Normalform:

$$\forall x (p(f_1(x)) \wedge q(x, f_2(x)))$$

Beispiel 4.65

Gegeben:

$$\exists x(p(w, x) \vee \forall y(q(w, x, y) \wedge \exists z r(y, z)))$$

All-Abschluß:

$$\forall w \exists x (p(w, x) \vee \forall y (q(w, x, y) \wedge \exists z r(y, z)))$$

Pränex Normalform:

$$\forall w \exists x \forall y \exists z (p(w, x) \vee (q(w, x, y) \wedge r(y, z)))$$

Skolemisierung:

$$\forall w \forall y (p(w, f_1(w)) \vee (q(w, f_1(w), y) \wedge r(y, f_2(w, y))))$$

Matrix in KNF, Skolem Normalform:

$$\forall w \forall y ((p(w, f_1(w)) \vee q(w, f_1(w), y)) \wedge (p(w, f_1(w)) \vee r(y, f_2(w, y))))$$

4.4.2 Der Satz von Herbrand

Formeln in Skolem-Normalform sind geschlossene, *universell quantifizierte* Formeln: Nur Allquantoren kommen vor, und sie stehen sämtlich im Präfix zu Beginn der Formel. Bei einer solchen Formel läßt sich wesentlich Genaueres über ihre Modelle aussagen als im allgemeinen Fall. Wir gehen hierauf noch kurz ein.

Notation

Zur Signatur Σ sei $Term_{\Sigma}^0$ die Menge aller Grundterme über Σ .

Definition 4.66 (Herbrand-Algebra)

Die Signatur Σ enthalte mindestens eine Konstante. Eine Interpretation (D, I) von Σ heißt *Herbrand-Interpretation* oder *Herbrand-Algebra* genau dann, wenn

1. $D = Term_{\Sigma}^0$
2. $I(f)(t_1, \dots, t_n) = f(t_1, \dots, t_n)$ für alle n -stelligen Funktionssymbole f , und beliebige Grundterme t_1, \dots, t_n . (Im Falle $n = 0$ heißt das: $I(c) = c$.)

Zu einer Formelmengemenge $M \subseteq For_{\Sigma}$ ist ein *Herbrand-Modell* von M eine Herbrand-Algebra über Σ , welche Modell von M ist.

In einer Herbrand-Algebra wird also jeder Grundterm als er selbst interpretiert,

$$I(t) = t \text{ für Grundterme,}$$

insbesondere gilt das für Konstanten. Spielraum für *verschiedene* Herbrand-Algebren gibt es nur bei der Interpretation der Prädikatsymbole. Wir notieren noch das

Korollar 4.67

Ist $(Term_{\Sigma}^0, I)$ Herbrand-Algebra über Σ , $t \in Term_{\Sigma}$ mit $Var(t) = \{x_1, \dots, x_n\}$, und β eine Belegung der Variablen mit Grundtermen, dann ist

$$val_{D, I, \beta}(t) = \{x_1/\beta(x_1), \dots, x_n/\beta(x_n)\}(t).$$

Beweis

Übung

(Hieraus folgt nochmals, daß in einer Herbrand-Algebra gilt $I(t) = t$ für jeden Grundterm t .)

Man beachte nochmals, daß die Voraussetzung, daß Σ mindestens eine Konstante enthalte, bedeutungslos ist: Ist das nicht der Fall, so nehme man eine Konstante hinzu.

Definition 4.68

Sei $A := \forall x_1 \dots \forall x_n B$ mit quantoremfreiem B eine geschlossenen Formel. Eine **Grundinstanz** von A ist eine Formel

$$\{x_1/t_1, \dots, x_n/t_n\}(B)$$

mit: $t_1, \dots, t_n \in Term_{\Sigma}^0$.

Ist M eine Menge geschlossener, universell quantifizierter Formeln, so sei

$$\text{Grundinstanzen}(M)$$

die Menge aller Grundinstanzen von Formeln in M .

Satz 4.69 (Satz von HERBRAND)

Σ enthalte mindestens eine Konstante, und es sei M eine Menge geschlossener, universell quantifizierter Formeln. Ferner enthalte keine Formel in M das Gleichheitssymbol \doteq . Dann sind äquivalente Aussagen

1. M hat ein Modell
2. M hat ein Herbrand-Modell
3. Grundinstanzen(M) hat ein Modell
4. Grundinstanzen(M) hat ein Herbrand-Modell.

Beweis

Die Implikationen $4 \Rightarrow 3$ und $2 \Rightarrow 1$ sind trivial; ebenso wegen der Allgemeingültigkeit von

$$\forall x_1 \dots \forall x_n B \rightarrow \{x_1/t_1, \dots, x_n/t_n\}(B)$$

die Implikationen $1 \Rightarrow 3$ und $2 \Rightarrow 4$. Wir brauchen nur noch zu zeigen, daß $3 \Rightarrow 2$.

Es sei (D, I) ein Modell von Grundinstanzen(M). Wir definieren eine Herbrand-Interpretation $(Term_{\Sigma}^0, J)$. Auf den Funktionssymbolen ist J schon vorgeschrieben. Wir setzen.

$$\begin{aligned} J(P) &:= I(P) \quad \text{für aussagenlogische Atome } P \\ J(p) &:= \{(t_1, \dots, t_n) \mid t_1, \dots, t_n \in Term_{\Sigma}^0, \text{val}_{D,I}(p(t_1, \dots, t_n)) = W\} \\ &\quad \text{für Prädikatsymbole } p \text{ einer Stelligkeit } n \geq 1. \end{aligned}$$

Das heißt also, es gilt

$$val_{Term_{\Sigma}^0, J}(A) = val_{D, I}(A)$$

für jedes geschlossene Atom A , das keine Gleichung ist. Mittels trivialer Induktion haben wir dies dann auch für alle geschlossenen, quantorenfreien Formeln, die \doteq nicht enthalten. Da (D, I) Modell von Grundinstanzen(M) ist, folgt

$$val_{Term_{\Sigma}^0, J}(\{x_1/t_1, \dots, x_n/t_n\}(B)) = W$$

für alle Formeln $\forall x_1 \dots \forall x_n B \in M$ und alle $(t_1, \dots, t_n) \in (Term_{\Sigma}^0)^n$. Mit Hilfe des Substitutionslemmas schließen wir hieraus

$$val_{Term_{\Sigma}^0, J}(\forall x_1 \dots \forall x_n(B)) = W$$

für jedes $\forall x_1 \dots \forall x_n B \in M$, d. h. $(Term_{\Sigma}^0, J)$ ist Modell von M . ■

Satz 4.70 (Satz von HERBRAND (2. Form))

Sei ϕ eine quantorenfreie Formel ohne Gleichheit mit einer freien Variablen x . Dann gilt

$\exists x \phi$ ist allgemeingültig

gdw

es gibt eine natürliche Zahl n und Grundterme t_1, \dots, t_n , sodaß $\phi(t_1) \vee \dots \vee \phi(t_n)$ allgemeingültig ist.

Beweis der 2. Form des Satzes von HERBRAND

$\exists x \phi$ ist allgemeingültig

$\Leftrightarrow \neg \exists x \phi$ besitzt kein Modell

$\Leftrightarrow \forall x \neg \phi$ besitzt kein Modell

$\Leftrightarrow \{\neg \phi(t) \mid t \text{ Grundterm}\}$ besitzt kein Modell

\Leftrightarrow es gibt ein n und t_1, \dots, t_n so daß $\{\neg \phi(t_1), \dots, \neg \phi(t_n)\}$ kein Modell besitzt (Anwendung des Endlichkeitssatzes)

$\Leftrightarrow \neg \phi(t_1) \wedge \dots \wedge \neg \phi(t_n)$ besitzt kein Modell

$\Leftrightarrow \phi(t_1) \vee \dots \vee \phi(t_n)$ ist allgemeingültig ■

4.4.3 Übungsaufgaben

Übungsaufgabe 4.4.1

Weshalb haben wir die Voraussetzung machen müssen, daß \doteq nicht in den Formeln in M auftritt? Was wäre eine naheliegende Verallgemeinerung des Begriffs der Herbrand-Algebra, bei der der obige Satz auch bei Formeln mit \doteq richtig bleibt?

Mengen von variablenfreien Formeln ohne \doteq – wie oben Grundinstanzen(M) – haben besonders schöne Eigenschaften, auf die wir zum Schluß noch hinweisen wollen. Wir wollen etwa annehmen, daß die Formeln in konjunkti-
ver oder disjunkti-
ver Normalform vorliegen. Das nachfolgende Lemma zeigt dann, weshalb solche Formeln günstig sind: insbesondere für automatisches Beweisen mit der Resolutionsregel.

Definition 4.71

Ein *Literal* ist ein Atom oder negiertes Atom. Es heißt *positives Literal* im ersten und *negatives* im zweiten Fall. Ein *Grundliteral* ist ein Literal ohne Variable. Ein *komplementäres Paar* von Literalen ist ein Paar $\{L, \neg L\}$ (L Atom).

Lemma 4.72

Für eine Konjunktion $L_1 \wedge \dots \wedge L_k$, $k \geq 1$, von Grundliteralen gilt:

1. $L_1 \wedge \dots \wedge L_k$ ist nicht allgemeingültig
2. $L_1 \wedge \dots \wedge L_k$ hat ein Modell $\Leftrightarrow \{L_1, \dots, L_k\}$ enthält kein komplementäres Paar

Für eine Disjunktion $L_1 \vee \dots \vee L_k$, $k \geq 1$ von Grundliteralen gilt:

3. $L_1 \vee \dots \vee L_k$ hat ein Modell
4. $L_1 \vee \dots \vee L_k$ ist allgemeingültig $\Leftrightarrow \{L_1, \dots, L_k\}$ enthält ein komplementäres Paar

Beweis

1. ist trivial.

Zu 2.:

Falls die Konjunktion $L_1 \wedge \dots \wedge L_k$ ein komplementäres Paar enthält, kann sie sicherlich kein Modell haben.

$L_1 \wedge \dots \wedge L_k$ enthalte kein komplementäres Paar. Über der gegebenen Signatur Σ betrachten wir die Herbrand-Algebra $(Term_\Sigma^0, I)$, in der festgelegt wird

$$I(p) := \{(t_1, \dots, t_n) \mid t_1, \dots, t_n \in Term_\Sigma^0, p(t_1, \dots, t_n) \in \{L_1, \dots, L_k\}\}$$

für Prädikatsymbole p einer Stelligkeit $n \geq 1$;

$$I(P) = W : \Leftrightarrow P \in \{L_1, \dots, L_k\}$$

für aussagenlogische Atome P .

Wir zeigen, daß $(Term_\Sigma^0, I)$ Modell jedes L_i ist ($i = 1, \dots, k$), mithin Modell von $L_1 \wedge \dots \wedge L_k$.

Wenn L_i Atom ist, etwa $L_i = p(t_1, \dots, t_n)$, dann $(t_1, \dots, t_n) \in I(p)$ nach Definition von I , d. h. $val_{Term_\Sigma^0, I}(p(t_1, \dots, t_n)) = val_{Term_\Sigma^0, I}(L_i) = W$. Ist L_i negatives Literal, etwa $L_i = \neg p(t_1, \dots, t_n)$, dann gilt $p(t_1, \dots, t_n) \notin \{L_1, \dots, L_k\}$, da es keine komplementäre Paare gibt. Also ist $val_{Term_\Sigma^0, I}(p(t_1, \dots, t_n)) = F$ und damit $val_{Term_\Sigma^0, I}(L_i) = val_{Term_\Sigma^0, I}(\neg p(t_1, \dots, t_n)) = W$. Entsprechend ist die Situation für aussagenlogische Atome P .

Zu 3.:

Insbesondere hat jedes einzelne Grundliteral stets ein Modell, also auch jede Disjunktion $L_1 \vee \dots \vee L_k$.

Zu 4.:

Schließlich gilt

$$L_1 \vee \dots \vee L_k \text{ ist allgemeingültig} \Leftrightarrow \neg(L_1 \vee \dots \vee L_k) \text{ hat kein Modell}$$

da die L_i geschlossen sind und 4. ergibt sich aus 2..

■

Übungsaufgabe 4.4.2

1. Wo haben wir im Beweis von Lemma 4.61 verwendet, daß die Ausgangsformel geschlossen ist?
2. Wo haben wir die Voraussetzung verwendet, daß $x_1, \dots, x_n \notin Bd(B)$?

Übungsaufgabe 4.4.3

Natürlich braucht A_{sk} durch A bei weitem nicht eindeutig bestimmt zu sein. Man mache sich das an Beispielen klar. Ferner finde man Beispiele dafür, daß A_{sk} und A nicht logisch äquivalent zueinander sind.

Übungsaufgabe 4.4.4

Die Elimination von Existenzquantoren kann auch ohne vorherigen Übergang zur pränexen Normalform erhalten werden. Dazu definieren wir rekursiv für Formeln A in Negationsnormalform:

- $sk(A) = A$, falls A ein Atom oder das Negat eines Atoms ist.
- $sk(A \wedge B) = sk(A) \wedge sk(B)$
- $sk(A \vee B) = sk(A) \vee sk(B)$
- $sk(\forall x A) = \forall x sk(A)$
- $sk(\exists x A) = sk(x/f(y_1, \dots, y_n))$, wobei y_1, \dots, y_n alle freien Variablen in $\exists x A$ sind.

1. Zeigen Sie, daß A und $sk(A)$ erfüllbarkeitsäquivalent sind.
2. Berechnen Sie $sk(A)$ für die Formeln aus Beispiel 4.64 und 4.65

Übungsaufgabe 4.4.5

Berechnen Sie zuerst die Pränex Normalform und dann die Skolem Normalform für folgende Formeln:

1. $(\forall x p(x) \rightarrow \forall x q(x)) \rightarrow \forall x(p(x) \rightarrow q(x))$
2. $\exists x(\forall y p(x, y) \vee \exists z(p(x, z) \wedge \forall x p(z, x)))$

Übungsaufgabe 4.4.6

In der zweiten Version des Satzes von Herbrand, Satz 4.70, war die Allgemeingültigkeit einer existentiellen Formel charakterisiert worden.

Gilt auch die folgende Aussage über die Erfüllbarkeit einer existentiellen Formel. Die Voraussetzungen seien wie in Satz 4.70:

$\exists x \phi$ ist erfüllbar

gdw

es gibt eine natürliche Zahl n und Grundterme t_1, \dots, t_n , sodaß $\phi(t_1) \vee \dots \vee \phi(t_n)$ erfüllbar ist.

Kapitel 5

Prädikatenlogik erster Ordnung: Beweistheorie

5.1 Tableauregeln

Eine ausführliche Darstellung des Tableauregeln findet man in den Büchern [Smu68] und [Fit90]. Abgesehen von der Orientierung zum automatischen Beweisen, enthält das Buch von Fitting gegenüber dem von Smullyan eine neue Behandlung des Allquantors, der es ermöglicht auch im Tableauregeln die Technik der Unifikation zu nutzen. Eine umfassende Darstellung der Tableau Methode wird in dem Handbuch [DGHP99] gegeben. Wir folgen der Darstellung in [Fit90] mit den Verbesserungen aus [HS94].

Wir arbeiten über der Basis $\{\neg, \wedge, \vee, \rightarrow, \forall, \exists\}$. Eine Transformation der am Beweis beteiligten Formeln in eine Normalform ist nicht erforderlich.

5.1.1 Tableauregeln

Prädikatenlogische Vorzeichenformeln sind prädikatenlogischen Formeln, vor denen 0 oder 1 als Vorzeichen stehen, wie in 3.4. Vorzeichenformeln werden wieder in Typen eingeteilt, und entsprechend dieser Einteilung werden Abkömmlinge definiert:

Uniforme Notation

Typ ϵ : $1A, 0A$ für Atome A

Typ α :

V	V_1	V_2
$1\neg A$	$0A$	$-$
$0\neg A$	$1A$	$-$
$1A \wedge B$	$1A$	$1B$
$0A \vee B$	$0A$	$0B$
$0A \rightarrow B$	$1A$	$0B$

Typ β :

V	V_1	V_2
$0A \wedge B$	$0A$	$0B$
$1A \vee B$	$1A$	$1B$
$1A \rightarrow B$	$0A$	$1B$

Typ γ :

V	V_1
$1\forall xA(x)$	$1A(x)$
$0\exists xA(x)$	$0A(x)$

Typ δ :

V	V_1
$1\exists xA(x)$	$1A(x)$
$0\forall xA(x)$	$0A(x)$

Durch die uniforme Notation wird in den nachfolgenden Beweisen eine Fallunterscheidung in elf Fälle auf vier Fälle reduziert. Das ist einzig und allein der Grund für ihre Einführung. Wesentlich dafür sind die folgenden Gemeinsamkeiten von Formeln desselben Typs:

Lemma 5.1

Sei (D, I) eine beliebige Interpretation, β eine beliebige Variablenbelegung für (D, I) . Wir schreiben val_β statt $val_{D,I,\beta}$. Dann gilt

1. für jede α -Formel V :

$$val_\beta(V) = W \Leftrightarrow val_\beta(V_1) = W \text{ und } val_\beta(V_2) = W$$

2. für jede β -Formel V :

$$val_\beta(V) = W \Leftrightarrow val_\beta(V_1) = W \text{ oder } val_\beta(V_2) = W$$

3. für jede γ -Formel V :

$$val_\beta(V) = W \Leftrightarrow \text{Für alle } d \in D : val_{\beta_x^d}(V_1(x)) = W$$

4. für jede δ -Formel V :

$$val_\beta(V) = W \Leftrightarrow \text{Es gibt } d \in D \text{ mit: } val_{\beta_x^d}(V_1(x)) = W.$$

(Hierbei ist $V_1(x)$ der Rumpf und x die quantifizierte Variable von V .)

Definition 5.2 (Tableaux)

Sei A eine Formel und M eine Menge von Formeln, alle ohne freie Variablen. Ein **Tableau für A über M** ist wie folgt rekursiv definiert.

1. Ein einziger mit $0A$ markierter Knoten ist ein Tableau für A über M .
2. α -Regel (wie im AL-Fall).
3. β -Regel (wie im AL-Fall).
4. Es sei T ein Tableau für A über M und π ein Pfad in T , so daß auf π ein Knoten liegt, der mit einem V vom Typ γ beschriftet ist, $V = 1\forall x B_1(x)$ oder $V = 0\exists x B_1(x)$. Dann ist auch T' ein Tableau für A über M , welches dadurch aus T entsteht, daß an π ein neuer Knoten angefügt wird, welcher mit $V_1(y)$ (d.h. $1B_1(y)$ bzw. $0B_1(y)$) beschriftet ist für eine Variable y , die noch nicht auf π frei vorkommt. Wir sagen in diesem Fall, daß T' aus T durch eine Anwendung der γ -**Regel** hervorgeht.

5. Es sei T ein Tableau für A über M und π ein Pfad in T , so daß auf π ein Knoten liegt, der mit einem V vom Typ δ beschriftet ist, $V = 0\forall xB_1(x)$ oder $V = 1\exists xB_1(x)$. Dann ist auch T' ein Tableau für A über M , welches dadurch aus T entsteht, daß an π ein neuer Knoten angefügt wird, welcher mit $V_1(f(x_1, \dots, x_n))$ beschriftet ist, wobei x_1, \dots, x_n alle freien Variablen in V sind und f ein neues Funktionszeichen ist. Wir sagen, T' geht aus T durch eine Anwendung der δ -**Regel** hervor.
6. Es sei T ein Tableau für A über M und π ein Pfad in T , dann ist auch T' ein Tableau für A über M , welches durch Verlängerung von π um einen mit $t \doteq t$ beschrifteten Knoten entsteht.
7. Es sei T ein Tableau für A über M und π ein Pfad in T , auf welchem die Formeln $1t \doteq s$ und V vorkommen, wobei t' ein in V vorkommender Teilterm ist. Außerdem existiere eine Substitution σ mit $\sigma(t') = \sigma(t)$. Dann ist auch T' ein Tableau für A über M , welches durch Verlängerung von π um einen mit V' beschrifteten Knoten und Anwendung von σ auf das ganze Tableau entsteht, wobei V' aus $\sigma(V)$ entsteht, indem ein Vorkommen von $\sigma(t)$ durch $\sigma(s)$ ersetzt wird. Wir sagen, T' geht aus T durch eine Anwendung der G_l -**Regel** aus T hervor.
8. Analog für die G_r -Regel.
9. Es sei T ein Tableau für A über M und π ein Pfad in T , dann ist auch T' ein Tableau für A über M , welches durch Verlängerung von π um einen mit $1B$ beschrifteten Knoten entsteht, für eine Formel B aus M . Wir sagen, T' geht aus T durch eine Anwendung der **V-Regel** aus T hervor.

Diese Definition ist eine Fortsetzung der entsprechenden aussagenlogischen Definition 3.27 auf Seite 79.

Zusammenfassung der Tableauregeln in Kurzform

α -Regel	$\frac{V}{\frac{V_1}{V_2}}$	für α -Formeln V
β -Regel	$\frac{V}{V_1 V_2}$	für β -Formeln V
γ -Regel	$\frac{V}{V_1(y)}$	für γ -Formeln V und jede Variable y , die auf dem Pfad noch nicht vorkommt
δ -Regel	$\frac{V}{V_1(f(x_1, \dots, x_n))}$	für δ -Formeln V , wobei x_1, \dots, x_n alle freien Variablen in V sind und f ein neues n -stelliges Funktionssymbol
G_{Id} -Regel	$\frac{}{1t \doteq t}$	
G_r -Regel	$\frac{1t \doteq s \quad V(s')}{\sigma V(t)}$	
G_l -Regel	$\frac{1t \doteq s \quad V(t')}{\sigma V(s)}$	wobei $\sigma(t') = \sigma(t)$
G_r -Regel	$\frac{1t \doteq s \quad V(s')}{\sigma V(t)}$	wobei $\sigma(s') = \sigma(s)$
Anfangsregel	$\frac{}{0A}$	für die zu beweisende Formel A A ohne freie Variable
V -Regel	$\frac{}{1B}$	für jedes $B \in M$ B ohne freie Variable

Die Substitution σ in der G_r - und G_l -Regel wird jeweils auf das ganze Tableau angewendet.

Definition 5.3

Es sei π ein Pfad in T und σ eine Substitution. σ *schließt* π , wenn es

- Formeln B, C gibt, so daß $\sigma(B) = \sigma(C)$, σ kollisionsfrei für B und C ist und $1B, 0C$ auf π liegen oder
- Terme s, t gibt, so daß $\sigma(s) = \sigma(t)$ und $0s \doteq t$ auf π liegt oder

- eine der Formeln **01** oder **10** liegt auf π .

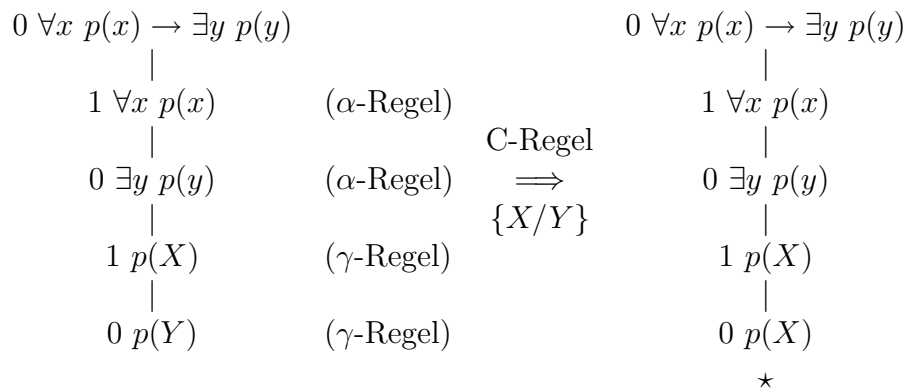
Definition 5.4 (Abschlußregel)

Die Schließregel oder C-Regel ist:

C-Regel Aus einem Tableau T erzeuge ein Tableau T_1 dadurch, daß für einen Pfad π und eine Substitution σ , die π schließt, σ auf ganz T angewandt wird. Nach Anwendung der C-Regel ist der zugehörige Pfad π geschlossen. Ein Tableau T heißt *geschlossen*, wenn alle seine Pfade geschlossen sind.

Beispiel 5.5

Ein Tableau, das mit der C-Regel geschlossen wird:



Aus der zu beweisenden Aussage ist durch einmalige Anwendung der α -Regel und zweimalige Anwendung der γ -Regel das linke Tableau entstanden. Aus diesem ist dann das rechts stehende durch die Anwendung der C-Regel entstanden. Zur Verdeutlichung der freien Variablen verwendet man üblicherweise Großbuchstaben für die Variablen, die bei Anwendung der γ -Regel substituiert werden.

Definition 5.6 (Der Kalkül T)

Sei A eine Formel, M eine Formelmengung, alle ohne freie Variable.

Wir sagen, A ist im Tableauekalkül aus M ableitbar, in Symbolen

$$M \vdash_{\mathbf{T}} A$$

wenn es ein geschlossenes Tableau für A über M gibt.

Der Kalkül \mathbf{T} arbeitet auf Tableaux als Objekten für die Regeln. Die Regeln von \mathbf{T} stellen jeweils aus einem Tableau T ein Tableau T_1 her gemäß einer der Regeln (α , β , γ , δ , G_l , G_r und C) anzuwenden auf eine oder zwei Vorzeichenformeln auf einem Pfad π in T .

Man beachte, daß zum Zwecke des Herstellens eines geschlossenen Tableaus für A über M zu den Regeln von \mathbf{T} die *Anfangsregel* und die *V-Regel* hinzuzunehmen sind.

In dieser Terminologie ist eine *Ableitung* in \mathbf{T} eine Folge von Tableaux, beginnend mit einem Ein-Knoten-Tableau der Art $0A$ und jeweils fortgesetzt durch Anwenden einer der angegebenen Regeln.

Bemerkung 5.7

Um $M \vdash_{\mathbf{T}} A$ zu etablieren, versuchen wir, ein geschlossenes Tableau für A über M zu finden. Ausgehend vom Ein-Knoten-Tableau $0A$ werden durch Anwendung der Regeln von \mathbf{T} und der V-Regel für M sukzessive Tableaux gebildet. Ist man bei einem solchen T angelangt, so sind Fortsetzungsmöglichkeiten

1. das Schließen eines Pfades in T durch die C-Regel, also der Übergang zu $\sigma(T)$ (in offensichtlicher Notation) und
2. das Verlängern von T mittels einer der weiteren Regeln von T oder der V-Regel.

In der Regel wird man zunächst versuchen, Fortsetzung 1 zu wählen. Ist dies nicht möglich, setzt man also mit 2 fort, so ist hierzu folgendes zu bedenken. Es ist nicht verboten, eine Regel auf denselben Knoten – also die dort stehende Vorzeichenformel – zweimal anzuwenden. Man erkennt jedoch, daß dies für alle Regeln *außer der γ -Regel* unnötig ist, der bearbeitete Knoten könnte im Prinzip gelöscht werden (das folgt aus Lemma 5.1). Die γ -Regel spielt eine Sonderrolle: Um (etwa) in $1\forall x B(x)$ für das Schließen immer neue Substitutionen $1B(t_1), 1B(t_2), \dots$ zu ermöglichen, müssen Formeln $1B(X_1), 1B(X_2), \dots$ bereitgehalten werden, d.h., $1\forall x B(x)$ wird immer wieder verwendet. Um keine Möglichkeit auszulassen, sollte also eine Strategie zur Wahl der Formel, mit deren Bearbeitung fortgesetzt wird, *fair* sein: Solange ein Pfad nicht geschlossen ist, wird versucht, jede $\alpha, \beta, \delta, G_l, G_r$ -Formel auf ihm genau einmal zu benutzen, jede γ -Formel "immer wieder" (unbeschränkt oft), mit immer neuen eingesetzten Variablen.

Beispiele für ein geschlossenes und ein offenes Tableau findet man in der Abb. 5.1 bzw. in Abb. 5.2. Um die Tableaux leichter nachvollziehbar zu machen, sind die Knoten mit $n[m]$ markiert. Die erste Zahl gibt eine fortlaufende Nummerierung aller Knoten. Die Zahl m in eckigen Klammern gibt an, welcher Knoten benutzt wurde, um den Knoten n zu erzeugen.

Abb. 5.3 zeigt ein Beispiel für die Anwendung der Gleichungsregel G_l . Um die Formel 6 zu erhalten betrachtet man die Gleichung $1t \doteq s = 1f(f(y)) \doteq g(y)$ in Zeile 5 und die Formel in Zeile 3. Die Rolle des Teilterms t' spielt $f(x)$.

1[]	$0\exists y\forall xp(x, y) \rightarrow \forall x\exists yp(x, y)$
2[1]	$1\exists y\forall xp(x, y)$
3[1]	$0\forall x\exists yp(x, y)$
4[2]	$1\forall xp(x, a)$
5[3]	$0\exists yp(b, y)$
6[4]	$1p(X, a)$
7[5]	$0p(b, Y)$

geschlossen mit $\sigma(X) = b$ und $\sigma(Y) = a$

Tabelle 5.1: Ein geschlossenes Tableau

1[]	$0\forall x\exists yp(x, y) \rightarrow \exists y\forall xp(x, y)$
2[1]	$0\exists y\forall xp(x, y)$
3[1]	$1\forall x\exists yp(x, y)$
4[2]	$0\forall xp(x, Y)$
5[3]	$1\exists yp(X, y)$
6[4]	$0p(f(Y), Y)$
7[5]	$1p(X, g(X))$

$p(f(Y), Y)$ und $p(X, g(X))$ sind nicht unifizierbar
es müßte $\sigma(X) = g(Y)$ und $\sigma(Y) = \sigma(f(g(Y)))$ gelten

Tabelle 5.2: Ein offenes Tableau

1[]	$0\forall x(f(g(x)) \doteq g(f(x)))$
2[V]	$1\forall x(f(f(x)) \doteq g(x))$
3[2]	$1f(f(X)) \doteq g(X)$
4[1]	$0f(g(c)) \doteq g(f(c))$
5[2]	$1f(f(Y)) \doteq g(Y)$
6[5, 3]	$1f(g(Y)) \doteq g(f(Y))$
	$\sigma(X) = f(Y)^1$

geschlossen mit 6, 4 und $\sigma(Y) = c$

Tabelle 5.3: Beispiel für die Anwendung der Gleichungsregel.

Durch die Substitution $\sigma(x) = f(y)$, wird $\sigma(t) = \sigma(t')$. Ersetzt man in Zeile 3 $t' = f(x)$ durch die rechte Seite der Gleichung in Zeile 5, so entsteht Zeile 6. Bildlich kann diese Operation dargestellt werden durch die farblich unterlegte Gegenüberstellung des Regelschemas mit der konkreten Anwendung

$$\frac{1t \doteq s \qquad F(t')}{1\sigma(t') = \sigma(t) \doteq \sigma(s) \quad \sigma(F(t')) \quad (\text{Zwischenschritt})}$$

$$\frac{1f(f(Y)) \doteq g(Y) \quad 1f(f(X)) \doteq g(X)}{1f(f(Y)) \doteq g(Y) \quad 1f(f(f(Y))) \doteq g(f(Y))}$$

$$\frac{\qquad}{1f(g(Y)) \doteq g(f(Y))}$$

Das geschlossene Tableau beweist die logische Folgerungsbeziehung:

$$\forall x(f(f(x)) \doteq g(x)) \models \forall x(f(g(x)) \doteq g(f(x)))$$

Abb. 5.4 zeigt eine zweite Möglichkeit, dasselbe Tableau zu schließen, jetzt unter Anwendung der Gleichungsregel G_r . Die folgende Gegenüberstellung hilft die Regelanwendung, die zu Zeile 5 führt, nachzuvollziehen.

$$\frac{1t \doteq s \qquad F(t')}{1\sigma(t) \doteq \sigma(s) = \sigma(t') \quad \sigma(F(t')) \quad (\text{Zwischenschritt})}$$

$$\frac{1f(f(X)) \doteq g(X) \quad 0f(g(c)) \doteq g(f(c))}{1f(f(c)) \doteq g(c) \quad 0f(g(c)) \doteq g(f(c))}$$

$$\frac{\qquad}{1f(f(f(c))) \doteq g(f(c))}$$

$$1[] \quad 0\forall x(f(g(x)) \doteq g(f(x)))$$

$$2[V] \quad 1\forall x(f(f(x)) \doteq g(x))$$

$$3[2] \quad 1f(f(X)) \doteq g(X)$$

$$4[1] \quad 0f(g(c)) \doteq g(f(c))$$

$$5[4, 3] \quad 0f(f(f(c))) \doteq g(f(c))$$

$$\qquad \sigma(X) = c^1$$

$$6[2] \quad 1f(f(Y)) \doteq g(Y)$$

$$\qquad \text{geschlossen mit 6, 5 und } \sigma(Y) = f(c)^1$$

Tabelle 5.4: 2. Beispiel für die Anwendung der Gleichungsregel.

5.1.2 Korrektheit

Wir führen den Begriff des Modells auch für Tableaux ein.

¹Kurzform, neues Tableau durch Substitution

Definition 5.8

Es seien $A \in For_\Sigma$, $M \subseteq For_\Sigma$, T ein Tableau für A über M und (D, I) eine Interpretation über $\bar{\Sigma}$, wobei $\bar{\Sigma} = \Sigma \cup \{f \mid f \text{ neues Funktionssymbol in } T\}$. (D, I) heißt **Modell von T über M** gdw. gilt

- (D, I) ist Modell von M
- zu jeder Variablenbelegung β gibt es einen Pfad π in T mit $val_{D,I,\beta}(V) = W$ für alle V auf π .

Vorbemerkung zu den nachfolgenden Beweisen.

Wir wollen die folgende (übliche) **Schreibweise** verwenden. Mit $\mathcal{D} = (D, I)$, β eine zugehörige Variablenbelegung, B eine Formel, stehe

$\mathcal{D} \models B$ für: \mathcal{D} ist Modell von B

$\mathcal{D} \models B[\beta]$ für: $val_{\mathcal{D},\beta}(B) = W$.

Entsprechend schreiben wir $\mathcal{D} \models V$, $\mathcal{D} \models V[\beta]$ für eine Vorzeichenformel V , ferner $\mathcal{D} \models N$, $\mathcal{D} \models N[\beta]$, $\mathcal{D} \models \pi$, $\mathcal{D} \models \pi[\beta]$ für eine Menge N von Formeln bzw. von Vorzeichenformeln, bzw. einen Pfad π (als Menge von Vorzeichenformeln).

Lemma 5.9

M sei eine Formelmenge ohne freie Variablen.

Das Tableau T' über M gehe aus T über M durch Anwendung einer Tableauregel hervor.

Hat T ein Modell über M , dann auch T' .

Beweis:

Der Beweis gliedert sich in acht Fälle, je nachdem, welche Tableauregel von T nach T' führt.

1. *β -Fall:* Auf einem Pfad π von T liegt die β -Formel V , und T' entsteht, indem der Pfad π erweitert wird einmal zum Pfad π_1 durch Anhängen der Formel V_1 und zum anderen zum Pfad π_2 durch Anhängen der Formel V_2 . Nach Voraussetzung hat T ein Modell \mathcal{D} , d. h. für jede Variablenbelegung β existiert ein Pfad π_0 von T mit $\mathcal{D} \models M$ und $\mathcal{D} \models \pi_0[\beta]$. Wir wollen zeigen, daß \mathcal{D} auch Modell von T' ist. Betrachten wir dazu eine beliebige Variablenbelegung β . Ist der Pfad π_0 von T mit $\mathcal{D} \models \pi_0[\beta]$, der nach Voraussetzung existiert, verschieden von π , dann ist π_0 auch ein Pfad in T' , und wir sind fertig. Ist $\pi_0 = \pi$, so liegt V auf π_0 und somit $\mathcal{D} \models V[\beta]$. Aus der Eigenschaft von β -Formeln folgt $\mathcal{D} \models V_1[\beta]$ oder $\mathcal{D} \models V_2[\beta]$ und somit $\mathcal{D} \models \pi_1[\beta]$ oder $\mathcal{D} \models \pi_2[\beta]$, und ohnehin ist $\mathcal{D} \models M$.

2. α -Fall: Analog. Bleibt dem Leser überlassen.
3. γ -Fall: Auf dem Pfad π von T kommt die γ -Formel V vor, und T' entsteht, indem π durch Hinzufügen der Formel $V_1(y)$, für eine auf π neue freie Variable y , zu dem Pfad π_1 verlängert wird. \mathcal{D} sei ein Modell von T über M . Wir zeigen, daß \mathcal{D} auch Modell von T' ist. Ist β eine Belegung, dann $\mathcal{D} \models \pi_0$ für ein π_0 in T . Wenn $\pi_0 \neq \pi$, ist π_0 auf Pfad T' , fertig. Wenn $\pi_0 = \pi$, hat man mit $\mathcal{D} \models V[\beta]$ auch $\mathcal{D} \models V_1(y)[\beta]$, also $\mathcal{D} \models \pi_1$.
4. δ -Fall: Auf dem Pfad π von T kommt die δ -Formel V vor mit den freien Variablen x_1, \dots, x_n , und T' entsteht, indem π durch Anhängen der Formel $V_1(f(x_1, \dots, x_n))$ für ein neues Funktionszeichen f zu π_1 verlängert wird. Nach Voraussetzung sei \mathcal{D} Modell von T über M . Wir konstruieren eine Interpretation \mathcal{D}' , die sich von \mathcal{D} nur darin unterscheidet, daß dem Funktionszeichen f eine Interpretation $f^{\mathcal{D}'}$ zugeordnet wird. Für $d_1, \dots, d_n \in D$ und einer Variablenbelegung β mit $\beta(x_i) = d_i$ für $i = 1, \dots, n$ gilt entweder

$$\mathcal{D} \models V[\beta],$$

in diesem Fall gibt es ein $d \in D$ mit

$$\mathcal{D} \models V_1[\beta_x^d],$$

oder $\mathcal{D} \models V[\beta]$ gilt nicht. Im letzten Fall wählen wir einen beliebigen Wert $d \in D$. Wir definieren dann $f^{\mathcal{D}'}(d_1, \dots, d_n) = d$ für die verschiedenen d_1, \dots, d_n mit dem entsprechenden d .

Damit ist die Definition der Interpretation \mathcal{D}' abgeschlossen. Wir wollen zeigen, daß \mathcal{D}' Modell von T' ist. Es sei β eine beliebige Belegung bzgl. \mathcal{D}' , β ist auch Belegung bzgl. \mathcal{D} , da sich der Grundbereich nicht geändert hat. Es gibt π_0 in T mit $\mathcal{D} \models \pi_0[\beta]$. Ist $\pi_0 \neq \pi$, so ist π_0 auch Pfad in T' . Ist $\pi_0 = \pi$, hat man $\mathcal{D} \models V[\beta]$, und nach Konstruktion von \mathcal{D}' auch $\mathcal{D}' \models V_1[\beta]$. Da in den restlichen Formeln des Pfades π_1 und in M das Zeichen f nicht vorkommt, erhalten wir insgesamt $\mathcal{D}' \models \pi_1[\beta]$ und $\mathcal{D}' \models M$.

5. Offensichtlich.
6. G_I -Regel: Auf dem Pfad π von T kommen die Formeln $1t \doteq s$ und $V(t')$, und für eine Substitution σ gelte $\sigma(t') = \sigma(t)$. T' entstehe aus T , indem π um die Formel $V(s)$ verlängert wird und auf das ganze Tableau σ angewendet wird. Nach Voraussetzung hat T ein Modell \mathcal{D} . Wir zeigen, daß \mathcal{D} auch Modell von T' ist.

Es sei β eine Belegung. Ist β' definiert durch

$$\beta'(y) = \text{val}_\beta(\sigma(y)),$$

dann gilt nach dem Substitutionstheorem

$$\mathcal{D} \models \sigma(C)[\beta] \quad \Leftrightarrow \quad \mathcal{D} \models C[\beta']$$

für jede Formel C . Nach Voraussetzung gibt es zu β' einen Pfad π_0 in T mit $\mathcal{D} \models \pi_0[\beta']$. Für die Formeln auf π_0 folgt, daß $\mathcal{D} \models \pi_0'[\beta]$ für den aus π_0 bei σ entstehenden Pfad π_0' . Wenn $\pi_0 \neq \pi$, hat man mit π_0' den gesuchten Pfad gefunden. Wenn $\pi_0 = \pi$, ist π_0' nicht maximal in T' , sondern dort noch um $\sigma(V(s))$ zu verlängern. Man hat $\mathcal{D} \models 1\sigma(t) = \sigma(s)[\beta]$, $\mathcal{D} \models \sigma(V(t'))[\beta]$, sowie die Termgleichheit $\sigma(t') = \sigma(t)$. Bei Ersetzung von t' in V durch s erhält man $V(s)$ mit $\mathcal{D} \models \sigma(V(s))[\beta]$.

7. G_r -Regel: vollkommen analog zu 6
8. V -Regel: offensichtlich, da ein Modell von T über M stets Modell von M , also von M ist.

■

Lemma 5.10

Ist \mathcal{D} Modell von T über M und entsteht T' aus T durch Schließen eines Pfades, dann ist \mathcal{D} auch Modell von T' .

Beweis:

Gemäß Voraussetzung gibt es zu jeder Belegung β einen Pfad π in T mit $\mathcal{D} \models \pi[\beta]$. T' entstehe durch Anwenden der Substitution σ und Schließen eines Pfades gemäß einer der beiden Möglichkeiten in 5.3. Sei β eine Belegung. Wir definieren β' als die Modifikation von β gemäß σ , wie im letzten Teil des Beweises zu 5.9.

Nach dem Substitutionslemma gilt

$$\mathcal{D} \models C[\beta'] \text{ gdw. } \mathcal{D} \models \sigma(C)[\beta] \text{ für alle } C,$$

so daß aus $\mathcal{D} \models \pi[\beta']$ für den zu β' gehörigen Pfad π folgt: $\mathcal{D} \models \sigma(\pi)[\beta]$ (wo $\sigma(\pi) = \{\sigma(V) \mid V \text{ auf } \pi\}$).

■

Satz 5.11 (Korrektheitssatz des Tableauealküls)

Sei $A \in \text{For}_\Sigma$, $M \subseteq \text{For}_\Sigma$ alle ohne freie Variable

Wenn es ein geschlossenes Tableau für A über M gibt, dann ist $M \models A$.

Beweis:

Wir bemerken zunächst, daß ein geschlossenes Tableau für A über M offensichtlich kein Modell haben M kann.

Wir definieren für die Zwecke dieses Beweises das Anfangstableau T_0 für A über M als das aus einem Pfad bestehende Tableau das genau die Formeln

- $0A$
- $1B$ für $B \in M$

enthält. Der bisherige beschriebene Kalkül schreibt zwar vor, daß $0A$ in jedem Tableau einer Ableitung auf jedem Pfad vorkommen muß, aber darüber, wann eine Voraussetzung B aus M in ein Tableau eingebaut wird, gibt es keine Festlegung. In den kleinen Beispielen, die wir bisher betrachtet haben, haben wir es so eingerichtet, daß auch die Formeln aus M von Anfang an dabei sind. Offenbar können wir diese Normierung vornehmen ohne Einschränkung der Allgemeinheit.

Sei jetzt

$$T_0, \dots, T_k, T_{k+1}, \dots, T_n$$

eine Ableitung für A über M . T_n ist nicht erfüllbar, da es ein abgeschlossenes Tableau ist. Dann sind aber auch alle vorangegangenen Tableaux unerfüllbar. Das können wir so einsehen. Angenommen T_{k+1} ist unerfüllbar. Wäre T_k erfüllbar, so müsste nach Lemma 5.9 auch T_{k+1} erfüllbar sein. Nach unserer Annahme kann das nicht sein, als ist zwangsläufig T_k unerfüllbar. Insbesondere ist T_0 nicht erfüllbar, was nach Definition von T_0 gleichbedeutend ist mit $M \models A$.

■

5.1.3 Hintikka-Mengen

Dieses Unterkapitel bereitet den Beweis des Vollständigkeitssatzes vor, der im folgenden Unterkapitel geführt wird. Ein wesentlicher Schritt in diesem Beweis ist die Konstruktion eines Modells für Formelmengen mit speziellen Eigenschaften, eben den Hintikka-Mengen. Die Konstruktion ist unabhängig vom Rest des Beweises und kann deshalb hier schon erledigt werden.

Definition 5.12
(Hintikka-Menge)

Eine Menge H von geschlossenen Vorzeichenformeln über einer Signatur Σ heißt eine **Hintikka-Menge**, wenn die folgenden Bedingungen erfüllt sind:

- (H 1) Gilt für eine α -Formel V , $V \in H$,
dann auch $V_1 \in H$ und $V_2 \in H$.
- (H 2) Gilt $V \in H$ für eine β -Formel V ,
dann auch $V_1 \in H$ oder $V_2 \in H$.
- (H 3) Gilt $V \in H$ für eine δ -Formel V ,
dann gibt es einen variablenfreien Term t mit $V_1(t) \in H$.
- (H 4) Gilt $V \in H$ für eine γ -Formel V ,
dann gilt $V_1(t) \in H$ für jeden variablenfreien Term t .
- (H 5) Liegt $1s \doteq t$ in H und $V \in H$, so daß s ein Teilterm von V ist, dann gilt auch $V' \in H$ wobei V' aus V entsteht, indem ein Vorkommen von s in V durch t ersetzt wird.
Dasselbe gilt analog für Terme t in V .
- (H 6) Für keine Formel A kommt sowohl $1A$, als auch $0A$ in H vor.
- (H 7) Für jeden variablenfreien Term t kommt $1t \doteq t$ in H vor.

Definition 5.13

Für eine Menge H variablenfreier Formeln definieren wir:

$$t \sim^H s \quad \text{gdw.} \quad 1t \doteq s \in H$$

Lemma 5.14

Ist H eine Hintikka-Menge, dann ist \sim^H eine Kongruenzrelation, d.h. \sim^H ist eine Äquivalenzrelation und erfüllt zusätzlich:

1. aus $t_1 \sim^H s_1, \dots, t_n \sim^H s_n$ folgt $f(t_1, \dots, t_n) \sim^H f(s_1, \dots, s_n)$
2. aus $t_1 \sim^H s_1, \dots, t_n \sim^H s_n$ und $1p(t_1, \dots, t_n) \in H$ folgt $1p(s_1, \dots, s_n) \in H$,
aus $t \sim^H s$ folgt $1t \doteq s \in H$

für jedes Funktionszeichen f und jedes Prädikatssymbol p .

Beweis:

Die Reflexivität von \sim^H wird durch (H 7) gewährleistet. Gilt $t \sim^H s$ und $s \sim^H r$, dann gilt nach Definition $1t \doteq s \in H$ und $1s \doteq r \in H$. Nach (H 5) liegt dann auch $1t \doteq r$ in H , d.h. es gilt $t \sim^H r$. Damit ist auch die Transitivität von \sim^H gezeigt. Bleibt die Symmetrieeigenschaft. Dazu gelte $s \sim^H r$, d.h. $1s \doteq r \in H$. Nach (H 7) gilt auch $1s \doteq s \in H$, woraus mit (H 5) auch $1r \doteq s \in H$ folgt.

Als nächstes nehmen wir uns den Nachweis der Kongruenzeigenschaft (1) vor. Gelte also $t_1 \sim^H s_1, \dots, t_n \sim^H s_n$. Nach Definition von \sim^H gilt also auch $1t_i \doteq s_i \in H$ für alle $1 \leq i \leq n$. Wegen (H 7) liegt auch $1f(t_1, \dots, t_n) \doteq f(t_1, \dots, t_n)$ in H . Durch mehrfache Anwendung von (H 5) erhält man nacheinander

$$\begin{aligned} 1f(t_1, \dots, t_n) &\doteq f(s_1, t_2, \dots, t_n) \in H \\ 1f(t_1, \dots, t_n) &\doteq f(s_1, s_2, t_3, \dots, t_n) \in H \\ &\text{bis} \\ 1f(t_1, \dots, t_n) &\doteq f(s_1, \dots, s_n) \in H \end{aligned}$$

Also gilt $f(t_1, \dots, t_n) \sim^H f(s_1, \dots, s_n)$, wie behauptet. Der Nachweis der Eigenschaft (2) bleibe dem Leser als Übungsaufgabe überlassen.

Lemma 5.15

(Modell-Lemma) Jede Hintikka-Menge H besitzt ein Modell.

Beweis:

Wir setzen

$$D = \{[t] : t \text{ ein Grundterm}\}$$

wobei $[t]$ die Äquivalenzklasse von t bzgl. \sim^H bezeichnet. Falls die ursprüngliche Signatur kein Konstantensymbol enthält fügen wir an dieser Stelle ein beliebiges neues Konstantensymbol hinzu. Wir müssen sicherstellen, daß es mindestens einen Grundterm gibt, das Universum D also nicht leer ist.

I wird definiert durch

$$\begin{aligned} I(f)([t_1], \dots, [t_n]) &= [f(t_1, \dots, t_n)] \\ ([t_1], \dots, [t_n]) \in I(p) &\Leftrightarrow 1p(t_1, \dots, t_n) \in H \end{aligned}$$

Die Kongruenzeigenschaften von \sim stellen sicher, daß diese Definitionen unabhängig von der Wahl der Repräsentanten der Äquivalenzklassen sind.

Mit dieser Definition gilt für jeden Grundterm:

$$I(t) = [t]$$

Wir beweisen diese Behauptung durch Induktion über den Termaufbau. Für $t = c$, ein Konstantensymbol, gilt nach Definition

$$I(c) = [c].$$

Sei jetzt $t = f(t_1, \dots, t_n)$:

$$\begin{aligned}
I(t) &= I(f)(t_1^{\mathcal{D}}, \dots, t_n^{\mathcal{D}}) && \text{(Def. von } I(t)) \\
&= I(f)([t_1], \dots, [t_n]) && \text{(Ind.Vor.)} \\
&= [f(t_1, \dots, t_n)] && \text{(Def. von } I(f))
\end{aligned}$$

Es bleibt, um den Beweis des Modell-Lemmas zu vervollständigen, noch nachzuweisen, daß für jede Formel $V \in H$ gilt

$$(D, I) \models V.$$

Dieser Nachweis wird wieder durch Induktion über den Aufbau von V geführt. (Man beachte, daß H nur geschlossene Formeln enthält.)

1. Fall: $V = 1p(t_1, \dots, t_n)$

Falls $V \in H$, dann gilt $\mathcal{D} \models V$ nach Definition von \mathcal{D} .

2. Fall: $V = 1t \doteq s$.

Aus $V \in H$ folgt $t \sim^H s$, und damit $[t] = [s]$

3. Fall: $V = 0p(t_1, \dots, t_n)$.

Wenn $V \in H$, dann gilt wegen (H 6) $1p(t_1, \dots, t_n) \notin H$. Nach Definition von (D, I) also $(D, I) \not\models p(t_1, \dots, t_n)$, d. h. $(D, I) \models \neg p(t_1, \dots, t_n)$

4. Fall: $V = 0t \doteq s$.

Sei $V \in H$.

Angenommen, es würde $t \sim^H s$ gelten, dann wegen der Kongruenzeigenschaft auch $0t \doteq t \in H$, was im Widerspruch zu (P7) steht. Also gilt $t \not\sim^H s$, d. h. $[t] \neq [s]$.

Die weiteren Induktionsschritte sind jetzt einfache Konsequenzen aus (H 1) bis (H 4). ■

5.1.4 Vollständigkeit

Satz 5.16 (Vollständigkeit von T)

Sei A eine Formel und M eine Menge von Formeln, alle ohne freie Variable. Gilt $M \models A$, dann gibt es ein geschlossenes Tableau für A über M .

Die Beweisidee ist folgende: Wir geben *Tableaukonstruktionsvorschriften* an, nach denen ein geschlossenes Tableau für A über M konstruiert werden soll. Wichtig dabei ist, daß die Formeln für die Regelanwendungen *fair* ausgewählt

werden d.h. nicht irgendwelche Möglichkeiten unberücksichtigt bleiben (vgl. oben 5.7). Wird auf diese Weise kein geschlossenes Tableau gefunden, so zeigen wir, daß dann $M \cup \{\neg A\}$ ein Modell haben muß, was im Widerspruch zu $M \models A$ steht.

Definition 5.17 (Tableaukonstruktion)

Es werden sukzessiv nicht nur Tableaux konstruiert, sondern parallel zu jedem Tableau eine Substitution für die freien Variablen in dem jeweiligen Tableau. Starttableau und -Substitution ist das Paar $(\{0A\}, id)$. (T_{n+1}, σ_{n+1}) entsteht aus (T_n, σ_n) , indem das Tableau T_n durch eine Tableauregel *fair* erweitert wird. Geschieht diese Erweiterung nicht durch Anwendung der γ -Regel, so ist $\sigma_{n+1} = \sigma_n$. Ansonsten entsteht σ_{n+1} aus σ_n , indem es um eine Substitution für die neue Variable aus der γ -Regel erweitert wird. Wodurch die Variable substituiert wird, ergibt sich aus einer (beliebigen aber festen) Aufzählung aller Grundterm $\{t_n : n \geq 1\}$. Bei der n -ten Anwendung der γ -Regel auf einem Pfad auf eine Vorzeichenformel V wird die neue Variable durch t_n substituiert. Wann Tableauerweiterungen *fair* sind, wird in der folgenden Definition festgelegt.

Definition 5.18 (Fairnessforderungen)

Die Konstruktion der (T_n, σ_n) ist fair, wenn:

1. nur Pfade π aus T_i verlängert werden, für die $\sigma_i(\pi)$ nicht geschlossen ist (geschlossene Pfade werden nicht erweitert),
2. für jede α -, β - oder δ -Formel V , die in einem Tableau T_j vorkommt, ein Index i mit $j < i$ existiert, so daß für jeden Pfad π in T_i , der V enthält, gilt: $\sigma_i(\pi)$ ist geschlossen oder V wurde zur Pfadverlängerung genutzt,
3. für jede γ -Formel V , die in einem Tableau T_j vorkommt, und jedem $m > 0$ ein Index i mit $j < i$ existiert, so daß für jeden Pfad π in T_i , der V enthält, gilt: $\sigma_i(\pi)$ ist geschlossen oder V wurde zur Pfadverlängerung m -mal genutzt (hier kommen die unterschiedlichen Substitutionen zum tragen, die neue Variable in T_i wird in σ_i durch t_m substituiert),
4. für alle $B \in M$ ein Index i existiert, so daß für jeden Pfad π in T_i gilt: $\sigma_i(\pi)$ ist geschlossen oder $1B$ liegt auf π .
5. für jede Formel $V(t)$ und jede Formel $1t \doteq s$, die in einem Tableau T_j vorkommen, ein Index i mit $j < i$ existiert, so daß für jeden Pfad π in

T_i , der $V(t)$ und $1t \doteq s$ enthält, gilt: $\sigma_i(\pi)$ ist geschlossen oder $V(t)$ und $1t \doteq s$ wurden (mit G_l -Regel) zur Pfadverlängerung genutzt,

6. für jede Formel $V(s)$ und jede Formel $1t \doteq s$, die in einem Tableau T_j vorkommen, ein Index i mit $j < i$ existiert, so daß für jeden Pfad π in T_i , der $V(s)$ und $1t \doteq s$ enthält, gilt: $\sigma_i(\pi)$ ist geschlossen oder $V(s)$ und $1t \doteq s$ wurden (mit G_r -Regel) zur Pfadverlängerung genutzt,

Man beachte, daß Fairness eine sehr einschneidende Forderung ist, die im wesentlichen eine Art Breitensuche erzwingt.

Beweis von Satz 5.16

Die Aussage des Satzes kann erweitert werden zu:

Sei A eine Formel und M eine Menge von Formeln.

Gilt $M \models A$ und konstruiert man nach den Tableaunkonstruktionsvorschriften 5.17 Tableaux für $cl_{\forall}A$ über M , dann wird nach endlich vielen Regelanwendungen ein geschlossenes Tableau $\sigma_i(T_i)$ gefunden.

Wir wollen einen Widerspruchsbeweis führen, d.h. wir nehmen an, daß kein geschlossenes Tableau gefunden wird.

Man beachte, daß bei der Konstruktion der T_i die Substitutionen $\sigma(i)$ nicht tatsächlich ausgeführt werden. Sie werden nur "notiert", um auszuweisen, daß ein Pfad in einem T_i "jetzt geschlossen werden könnte". Dieser Pfad wird dann nicht mehr fortgesetzt. Es entsteht also jeweils T_{i+1} aus T_i durch Anhängen von Knoten an einem Pfad π , so daß $\sigma_i(\pi)$ nicht geschlossen ist. Durch Vereinigung der Tableaux T_i , $i = 0, 1, 2, \dots$, entsteht ein Tableau T , und aus der Voraussetzung "kein $\sigma_i(T_i)$ ist geschlossen" folgt, daß T unendlich ist. Wir verwenden nun das aus der Graphentheorie bekannte

Königs Lemma: In jedem unendlichen, endlich verzweigenden Baum existiert ein unendlicher Pfad.

Es sei π ein unendlicher Pfad in T . Wir zeigen: *Es gibt eine Hintikka-Menge H , die alle Vorzeichenformeln auf $\sigma(\pi)$ enthält.*

Man zeigt zuerst, daß $\sigma(\pi)$, aufgefaßt als Formelmenge, die Eigenschaften (H 1) – (H 6) erfüllt. (H 1) bis (H 5) und (H 7) folgen unmittelbar aus den Fairnessforderungen, während (H 6) aus der Tatsache folgt, daß der Pfad nicht geschlossen wird.

Betrachten wir (H 4) etwas genauer. Zunächst wissen wir, daß nach der Fairnessvorschrift jede γ -Formeln $V \in \sigma(\pi)$ unendlich oft zur Verlängerung des Pfades benutzt wurde und damit für jeden variablenfreien Term t auch $V_1(t)$ auf dem Pfad liegt (V_1 sei der Rumpf von V ohne Quantor).

Nach Lemma 5.15 hat H ein Modell. Da $H \models A$ und alle $\models B$, $B \in M$ enthält, ist dieses auch Modell von $M \cup \{\neg A\}$. Also $M \not\models A$ entgegen Annahme. ■

5.1.5 Übungsaufgaben

Übungsaufgabe 5.1.1

Zeigen Sie, daß die Formel

$$\forall x \forall y (\phi(x, y) \vee \psi(x, y)) \rightarrow \forall x \exists y \phi(x, y) \vee \exists x \forall y \psi(x, y)$$

eine Tautologie ist.

Diese simple Tautologie ist der Ausgangspunkt einer Methode, genannt Funktionseinführung (function introduction), zur Beschleunigung von Resolutionsbeweisen. Siehe etwa [BL92].

Übungsaufgabe 5.1.2

Geben Sie ein geschlossenes Tableau für die folgende Formelmengenan (Hinweis: die γ -Formel muß dreimal instantiiert werden):

$$\Phi := \{\exists x, y \forall z (z \doteq x \vee z \doteq y), p(a) \wedge p(b), \neg a \doteq b, \neg \forall x p(x)\}$$

Übungsaufgabe 5.1.3

Zeigen Sie die logische Folgerungsbeziehung:

$$\{\forall x \forall y (f(x) \doteq g(y) \rightarrow p(x, y)), f(a) \doteq c, g(b) \doteq c\} \vdash p(a, b)$$

Übungsaufgabe 5.1.4

Die Formulierungen in den Sätzen 5.11 und 5.16 sind so allgemein gefasst, daß sie auch die Fälle abdecken, in denen die Prämissenmenge M oder das zu beweisende Theorem A freie Variable enthalten. In der Regel wird das nicht der Fall sein. In dieser Aufgabe wollen wir genau diesen seltenen Fall betrachten. Was geschieht, wenn man die Bildung des universellen Abschlusses wegläßt? Geben Sie Beispiele, daß keine der beiden Implikationen in der folgenden Äquivalenzaussage richtig ist:

$M \models A$ gilt genau dann, wenn es ein geschlossenes Tableau für A über M gibt.

Übungsaufgabe 5.1.5

Zeigen Sie, daß man die G_r Regel mit Hilfe der G_l und G_{Id} Regel simulieren kann.

Übungsaufgabe 5.1.6

Sie H_0 eine Formelmenge, die alle Eigenschaften einer Hintikka Menge erfüllt außer (H 7). Die dafür aber die folgende Eigenschaft hat

$$(0t \doteq t) \notin H_0 \text{ für jeden variablenfreien Term } t$$

1. Sei $H_1 = H_0 \cup \{1t \doteq t \mid t \text{ ein variablenfreier Term}\}$ Zeigen Sie zunächst, daß H_1 nicht notwendigerweise eine Hintikka Menge sein muß.

Übungsaufgabe 5.1.7

In der Definition 4.25 einer prädikatenlogischen Interpretation war verlangt worden, daß nur nichtleere Mengen als Trägermengen D auftreten können. Für die Zwecke dieser Übungsaufgabe wollen wir diese Einschränkung fallen lassen und auch $D = \emptyset$ zulassen. Wir nennen dies die *Nullsemantik*

1. Geben Sie ein Beispiel für eine Formel an, die in der üblichen Semantik allgemeingültig ist, aber in der Nullsemantik nicht.
2. Wenn man den in diesem Kapitel beschriebenen Tableaurekalkül für die Nullsemantik benutzt, welche Eigenschaft geht dabei verloren: die Vollständigkeit oder die Korrektheit?
3. Wie muß das Tableaurekalkül abgeändert werden, damit es vollständig und korrekt bezüglich der Nullsemantik ist?

Übungsaufgabe 5.1.8

Man beweise im Tableaurekalkül:

$$\forall t(a(\text{next}(t)) \leftrightarrow b(t) \wedge \neg \text{reset}(t)) \vdash \forall t(\text{reset}(t) \rightarrow \neg a(\text{next}(t)))$$

Übungsaufgabe 5.1.9

Zeigen Sie, daß eine Robbins Algebra \mathcal{R} , in der die Formel

$$\forall x(\neg\neg x = x)$$

gilt, schon eine Boolesche Algebra ist.

Übungsaufgabe 5.1.10

Zeigen Sie, daß das folgende Tableau geschlossen werden kann:

$$\begin{array}{l} 1[0] \quad 1 \quad \forall x \forall y (r(x, y) \wedge r(y, x) \rightarrow x \doteq y) \\ 2[0] \quad 1 \quad \forall x (r(x, m(f(x)))) \\ 3[0] \quad 1 \quad \forall x (r(x, f(m(x)))) \\ 4[0] \quad 1 \quad \forall x \forall y (r(x, y) \rightarrow r(m(y), m(x))) \\ 5[0] \quad 0 \quad m(f(m(a))) \doteq m(a) \end{array}$$

Übungsaufgabe 5.1.11

Betrachten wir die Regeln G'_l und G'_r , die genau wie G_l und G_r definiert sind, nur dass die Substitution σ nur *lokal*, d.h. nur auf die neue Formel, angewendet wird, nicht aber auf das restliche Tableau.

Finden Sie eine Formel, die nicht allgemeingültig ist, aber für die ein geschlossenes Tableau existiert, wenn G'_l und G'_r zugelassen werden.

Diese Regeln sind also **nicht korrekt**.

5.2 Sonstige Kalküle

5.2.1 Hilbertkalkül

Definition 5.19 (Hilbertkalkül)

Der Hilbertkalkül (für eine Signatur Σ) bezeichnet mit \mathbf{H} (genauer mit \mathbf{H}_Σ) wird durch die folgenden Axiome und Regeln gegeben:

(Wie in der Aussagenlogik sind die Regeln – insbesondere die Axiome – als Schemata angegeben. x, y, z, \dots stehen für Variablen, f für ein Funktionssymbol, p für ein Relationssymbol, t für einen Term, A, B, C stehen für Formeln.)

Axiome

Ax1	$A \rightarrow (B \rightarrow A)$	(Abschwächung)
Ax2	$(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$	(Verteilung von \rightarrow)
Ax3	$(\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A)$	(Kontraposition)
Ax4	$\forall x A \rightarrow \{x/t\}(A)$ wobei $\{x/t\}$ kollisionsfrei für A	(Instantiierung)
Ax5	$\forall x(A \rightarrow B) \rightarrow (A \rightarrow \forall x B)$ wobei $x \notin \text{Frei}(A)$	(\forall -Verschiebung)
Gl1	$x \doteq x$	(Reflexivität)
Gl2	$x \doteq y \rightarrow y \doteq x$	(Symmetrie)
Gl3	$x \doteq y \rightarrow (y \doteq z \rightarrow x \doteq z)$	(Transitivität)
Gl4	$x_1 \doteq y_1 \rightarrow (x_2 \doteq y_2 \rightarrow (\dots (x_n \doteq y_n \rightarrow f(x_1, \dots, x_n) \doteq f(y_1, \dots, y_n)) \dots))$	
Gl5	$x_1 \doteq y_1 \rightarrow (x_2 \doteq y_2 \rightarrow (\dots (x_n \doteq y_n \rightarrow p(x_1, \dots, x_n) \rightarrow p(y_1, \dots, y_n)) \dots))$	

Regeln

$$\text{Mp: } \frac{A, A \rightarrow B}{B} \quad (\text{Modus ponens})$$

$$\text{Gen: } \frac{A}{\forall x A} \quad (\text{Generalisierung})$$

Ableitbarkeit im Hilbertkalkül wird

bezeichnet mit $\vdash_{\mathbf{H}}$.

Wir erinnern daran, daß $M \models A$ für die semantische Folgerbarkeit von a aus M steht (siehe Definition 4.38 auf Seite 134).

Beispiel 5.20 (Eine Ableitung in \mathbf{H})

Wir schreiben in diesem Beispiel der Einfachheit halber \vdash statt $\vdash_{\mathbf{H}}$.

Sei $M = \{r(x, x), r(x, y) \wedge r(x, z) \rightarrow r(y, z)\}$. Wir wollen

$$M \vdash r(x, y) \rightarrow r(y, x)$$

zeigen. (Daselbe Beispiel wird auch in dem Buch [HW90], S. 82ff benutzt, allerdings mit einem anderen Kalkül.)

- 1 $M \vdash r(x, y) \wedge r(x, z) \rightarrow r(y, z)$ (Vor.)
- 2 $M \vdash \forall z(r(x, y) \wedge r(x, z) \rightarrow r(y, z))$ (Gen.)[1]
- 3 $\emptyset \vdash \forall z(r(x, y) \wedge r(x, z) \rightarrow r(y, z)) \rightarrow (r(x, y) \wedge r(x, x) \rightarrow r(y, x))$ (Ax4)
- 4 $M \vdash r(x, y) \wedge r(x, x) \rightarrow r(y, x)$ (MP)[2, 3]
- 5 $\emptyset \vdash (r(x, y) \wedge r(x, x) \rightarrow r(y, x)) \rightarrow$ (AL-Tautologie $A \rightarrow (B \rightarrow A \wedge B)$
 $(r(x, x) \rightarrow ((r(x, y) \wedge r(x, x) \rightarrow r(y, x)) \wedge r(x, x)))$
- 6 $M \vdash r(x, x) \rightarrow ((r(x, y) \wedge r(x, x) \rightarrow r(y, x)) \wedge r(x, x))$ (MP)[4, 5]
- 7 $M \vdash r(x, x)$ (Vor.)
- 8 $M \vdash (r(x, y) \wedge r(x, x) \rightarrow r(y, x)) \wedge r(x, x)$ (MP)[6, 7]
- 9 $\emptyset \vdash ((r(x, y) \wedge r(x, x) \rightarrow r(y, x)) \wedge r(x, x))$ (AL-Tautologie
 $\rightarrow (r(x, y) \rightarrow r(y, x))$
- 10 $M \vdash r(x, y) \rightarrow r(y, x)$ (MP)[8, 9]

Satz 5.21 (Vollständigkeit und Korrektheit von H)

Σ sei eine Signatur der PL1. Für jede Formelmengung $M \subseteq For_{\Sigma}$ und jede Formel $A \in For_{\Sigma}$ gilt:

1. $M \vdash_{\mathbf{H}} A \Rightarrow M \models A$
2. $M \models A \Rightarrow M \vdash_{\mathbf{H}} A$

5.2.2 Resolutionskalkül

Um unsere Darstellung möglichst knapp zu halten, verzichten wir auf die Behandlung der Gleichheit.

Im übrigen sei auf die Literatur verwiesen, insbesondere auf die beiden Bücher von [HK89] und [SA91]. Aber auch die Klassiker auf diesem Gebiet, [CL73] und [Lov78], sind auch heute noch mit Gewinn zu lesen.

Definition 5.22 (Klausel)

Ein **Literal** ist eine atomare oder eine negierte atomare Formel.

Eine **Klausel** ist eine endliche Menge von Literalen.

Die **leere Klausel** wird mit \square bezeichnet.

Eine Klausel wird interpretiert wie die Disjunktion ihrer Literale. Eine Menge von Klauseln wird, wie jede Menge von Formeln, interpretiert wie die (unendliche) Konjunktion der den Klauseln entsprechenden Disjunktionen. Alle auftretenden Variablen werden als universell quantifiziert interpretiert. Die leere Klausel, \square , erhält immer den Wahrheitswert F .

Definition 5.23 (Variante)

Ist A eine quantorenfreie Formel und σ eine Variablenumbenennung, dann heißt $\sigma(A)$ eine *Variante* von A .

Notation

Zu einem Literal L sei $\sim L$ das Literal

$$\sim L := \begin{cases} \neg L & \text{wenn } L \text{ ein Atom ist} \\ L' & \text{wenn } L = \neg L', L' \text{ Atom, ist.} \end{cases}$$

Zu einer Klausel C sei $\sim C := \{\sim L \mid L \in C\}$.

Definition 5.24 (Resolutionsregel)

Über der Menge der Klauseln (über der prädikatenlogischen Signatur Σ) ist die *Resolution* die folgende Regel

$$\frac{C_1 \cup K_1 \quad C_2 \cup K_2}{\mu(C_1 \cup C_2)}$$

wobei gilt:

- C_1, C_2, K_1, K_2 sind Klauseln
- $K_1, K_2 \neq \square$
- $Var(C_1 \cup K_1) \cap Var(C_2 \cup K_2) = \emptyset$
- μ ist allgemeinsten Unifikator von $K_1 \cup \sim K_2$.

Eine Klausel C heißt *Resolvente* zweier Klauseln, wenn sie durch Anwendung von Res auf diese entsteht.

Definition 5.25 (Ableitbarkeit im Resolutionskalkül)

Sei M eine Klauselmenge.

1. Mit $Res(M)$ bezeichnen wir die Menge aller Resolventen von Klauseln aus M . Genauer:

$$Res(M) = \{B \mid \text{es gibt Varianten } C_1, C_2 \text{ von Klauseln aus } M, \text{ so daß } B \text{ eine Resolvente von } C_1, C_2 \text{ ist.}\}$$

2. $R^0(M) = M$
3. $R^{n+1}(M) = Res(R^n) \cup R^n$
4. $M \vdash_{\mathbf{R}} A$ gilt genau dann, wenn es ein n gibt mit $A \in R^n(M)$

Beispiel 5.26 (Anwendung der Resolutionsregel)

Gegeben seien die beiden Klauseln

$$\{p(x), q(f(x)), q(f(g(c)))\} \text{ und } \{r(y, z), \neg q(z), \neg q(f(y))\}$$

Zur Vorbereitung der Anwendung der Resolutionsregel zerlegen wir die Klauseln in komplementäre Klauseln K_1 und K_2 und die jeweiligen Restklauseln C_1 und C_2 .

$$\begin{aligned} K_1 &= \{q(f(x)), q(f(g(c)))\} & C_1 &= \{p(x)\} \\ K_2 &= \{\neg q(z), \neg q(f(y))\} & C_2 &= \{r(y, z)\} \end{aligned}$$

Die Menge $K_1 \cup \sim K_2$ ist also in diesem Fall $\{q(z), q(f(y)), q(f(x)), q(f(g(c)))\}$ mit $\sigma = \{x/g(c), y/g(c), z/f(g(c))\}$ als allgemeinstem Unifikator. Die Resolvente ist daher $\{p(g(c)), r(g(c), f(g(c)))\}$.

Am häufigsten wird die Resolutionsregel für einelementige komplementäre Klauseln angewandt. Diese spezielle Form heißt **binäre Resolution**.

Beispiel 5.27 (Binäre Resolution)

Gegeben seien die beiden Klauseln $\{p(x), q(f(x))\}$ und $\{r(y, c), \neg q(f(c))\}$.

$$\begin{aligned} K_1 &= \{q(f(x))\} & C_1 &= \{p(x)\} \\ K_2 &= \{\neg q(f(c))\} & C_2 &= \{r(y, c)\} \end{aligned}$$

Die Menge $K_1 \cup \sim K_2$ ist also in diesem Fall $\{q(f(x)), q(f(c))\}$ und unifizierbar durch den allgemeinsten Unifikator $\sigma = \{x/c\}$. Die Resolvente ist daher $\{p(c), r(y, c)\}$.

Satz 5.28 (Korrektheit und Vollständigkeit von R)

Sei K eine Menge von Klauseln im Vokabular Σ . Dann gilt

1. Gilt $M \vdash_{\mathbf{R}} \square$, dann ist M nicht erfüllbar.
2. Ist M nicht erfüllbar, dann folgt $M \vdash_{\mathbf{R}} \square$

Beweis

zu 1: Wir zeigen für eine beliebige Klauselmenge M für alle $n \geq 0$

wenn $R^n(M)$ erfüllbar ist, dann ist auch $R^{n+1}(M)$ erfüllbar

Daraus folgt dann:

wenn $M \vdash_{\mathbf{R}} \square$ dann $\square \in R^n(M)$ für ein n ,
dann $R^n(M)$ unerfüllbar für ein n ,
dann $R^0(M) = M$ unerfüllbar.

Kommen wir zum Beweis der induktiven Behauptung. Sei dazu (D, I) ein Modell von $R^n(M)$. Da alle Variablen als implizit universell quantifiziert interpretiert werden, gilt für alle Klauseln $K \in R^n(M)$ und alle Variablenbelegungen β $(D, I, \beta) \models K$. Wer streben an, für jede Resolvente C von Klauseln aus $R^n(M)$ auch $(D, I, \beta) \models C$ für jedes β nachzuweisen. Damit wären wir dann auch fertig.

Sei also C eine Resolvente von $C_1, C_2 \in R^n(M)$. Genauer heißt das nach der allgemeinen Definition einer Resolvente, daß $C_1 = C_{1,1} \cup K_1$, $C_2 = C_{2,1} \cup K_2$, μ ein allgemeinsten Unifikator von $K_1 \cup \sim K_2$ und $C = \mu(C_{1,1} \cup C_{2,1})$. Nach Voraussetzung gilt für eine beliebige Variablenbelegung β

$$(D, I, \beta) \models \mu(C_{1,1}) \vee \mu(K_1) \quad \text{und} \quad (D, I, \beta) \models \mu(C_{2,1}) \vee \mu(K_2)$$

Gilt $(D, I, \beta) \models \mu(C_{1,1})$ dann gilt um so mehr $(D, I, \beta) \models \mu(C_{1,1}) \vee \mu(C_{2,1})$ und wir sind fertig. Somit bleibt noch der Fall $(D, I, \beta) \models \mu(K_1)$ zu betrachten. Da μ Unifikator von $K_1 \cup \sim K_2$ ist, folgt daraus $(D, I, \beta) \not\models \mu(K_2)$ und damit muß nach Voraussetzung $(D, I, \beta) \models \mu(C_{2,1})$ gelten. Was wieder $(D, I, \beta) \models C$ nach sich zieht.

Das obige Argument mit dem Unifikator kann man sich klar machen, wenn man den einfachsten Fall $K_1 = \{L_1\}$, $K_2 = \{\neg L_2\}$ betrachtet mit $\mu(L_1) = \mu(L_2)$.

Bemerkungen: Da Variablen implizit als universell quantifiziert angesehen werden macht es für das vorliegende Argument keinen Unterschied, ob wir ein $C \in R^n(M)$ betrachten oder einer Variante C' für ein $C \in R^n(M)$. Offensichtlich spielte die Variablendisjunktheitsbedingung aus der Definition einer Resolvente für obigen Beweis keine Rolle. Ebenso die Forderung, daß die K_i nicht trivial sind.

zu 2: Die Beweisstrategie zielt darauf ab, den Vollständigkeitsbeweis für die aussagenlogische Resolution, Satz 3.19, zu benutzen. Wir beginnen mit der Beobachtung, daß aus der vorausgesetzten Unerfüllbarkeit von M nach dem Herbrandschen Satz (Satz 4.69, Implikation von (1) nach (3)), auch die Unerfüllbarkeit der Menge \bar{M} aller Grundinstanzen von M folgt. Wir ersetzen in \bar{M} jedes atomare Formel durch ein aussagenlogisches Atom - natürliche alle Vorkommen einer atomaren Formel durch dasselbe Atom. Die entstehende Menge aussagenlogischer Formel M_0 ist ebenfalls nicht erfüllbar. Warum? Aus einer erfüllenden Belegung I für M_0 könnten wir ein Herbrand-Modell \mathcal{H} für \bar{M} bauen, indem wir $\mathcal{H} \models p(t_1, \dots, p_n)$ genau dann setzen wenn $I(A) = \mathbf{1}$ gilt für das der atomaren Formel $p(t_1, \dots, p_n)$ zugeordnete aussagenlogische Atome A gilt. Nach Satz 3.19 kann man mit aussagenlogischer Resolution aus M_0 die leere Klausel \square herleiten. Die Frage, ob man aus einer aussagenlogischen Herleitung eine prädikatenlogische Herleitung gewinnen kann, ist

unter dem Namen *lifting* bekannt. Die Antwort ist natürlich positiv. Genauer werden wie zeigen:

Lifting Lemma

für jedes n und jede Klausel $C_0 \in R_0^n(M_0)$
gibt es eine Klausel $C \in R^n(M)$ und eine Substitution σ , so daß
 C_0 aus $\sigma(C)$ durch die oben beschriebene Ersetzung
von Grundliterals durch aussagenlogische Atome entsteht.

Wir wissen bisher schon, daß es ein n gibt mit $\square \in R_0^n(M_0)$. Nach dem Lifting Lemma gibt es dann eine Klausel $C \in R^n(M)$ und eine Substitution σ mit $\sigma(C) = \square$. Dann kann nur $C = \square$ sein und wir sind fertig. Es bleibt also nur noch das Lifting Lemma zu beweisen.

Der Beweis geschieht durch Induktion über n . Für $n = 0$ reduziert sich die Behauptung des Lemmas auf die Definition von M_0 . Für den Schritt von n nach $n + 1$ betrachten wir zwei Klauseln $C_{1,0}, C_{2,0} \in R_0^n(M_0)$ und eine ihrer Resolventen $C_0 \in R_0^{n+1}(M_0)$. Etwa $C_0 = \{L_1^0, \dots, L_k^0\}$ mit $C_{1,0} = \{L_1^0, \dots, L_k^0, L^0\}$ und $C_{2,0} = \{L_1^0, \dots, L_k^0, \neg L^0\}$. Nach Induktionsvoraussetzung gibt es $C_1, C_2 \in R^n(M)$ und Substitutionen σ_1, σ_2 mit $\sigma_1(C_1) = C_{1,0}$ und $\sigma_2(C_2) = C_{2,0}$. Wir unterschlagen hier die Ersetzung von Grundliterals durch aussagenlogische Atome. Das ist für den Beweis unerheblich. Wir können außerdem annehmen, daß C_1 und C_2 keine gemeinsamen Variablen haben. Denn einerseits läßt sich eine Variablenumbenennung die C_1 und C_2 variablendisjunkt macht durch entsprechende Änderungen von σ_1 und σ_2 kompensieren andererseits ist die Bildung von Varianten der Ausgangsklauseln eines Resolutionsschritts explizit erlaubt und die Variablendisjunktheit sogar gefordert. Wegen der Variablendisjunktheit von C_1 und C_2 können wir σ_1 und σ_2 zu einer Substitution μ zusammenfassen mit $\mu(C_1) = C_{1,0}$ und $\mu(C_2) = C_{2,0}$. Schauen wir uns C_1 und C_2 etwas genauer an $C_1 = C_{1,1} \cup C_{1,2}$ und $C_2 = C_{2,1} \cup C_{2,2}$ mit $\mu(C_{1,2}) = \mu(\sim C_{2,2}) = L$ und $C_0 = \mu(C_{1,1}) \cup \mu(C_{2,1})$. Man beachte, daß $C_{1,2}$ und $C_{2,2}$ keine Einerklauseln sein müssen. Die allgemeiner Form der prädikatenlogischen Resolution fordert das auch nicht. Wir haben noch eine kleine Klippe zu überwinden. Wir wissen, daß μ ein Unifikator von $C_{1,2}$ und $\sim C_{2,2}$ ist. Der Resolutionsschritt wird mit einem allgemeinsten Unifikator μ_a durchgeführt, wobei etwa $\mu = \mu' \circ \mu_a$ gilt. Das Ergebnis der prädikatenlogischen Resolution ist $C = \mu_a(C_{1,1} \cup C_{2,1})$ mit $C \in R^{n+1}(M)$. Somit gilt $\mu'(C) = \mu(C_{1,1} \cup C_{2,1}) = C_0$ und wir sind fertig. ■

Arbeiten mit dem Resolutionskalkül

Der Resolutionskalkül ist ein Widerlegungskalkül, d.h. der Beweis einer logischen Folgerbarkeit wird in einen Nachweis der Widersprüchlichkeit umgeformt. Wir fassen die Vorgehensweise zusammen.

Gegeben: Eine endliche Menge $\{B_1, \dots, B_n\} \subseteq \text{For}_\Sigma$ und eine Formel $A \in \text{For}_\Sigma$.

Gesucht: Für den Fall, daß $\{B_1, \dots, B_m\} \models A$ ein Beweis dieser Tatsache unter Verwendung von **R**.

Verfahren:

1. Bilde $E := \{Cl_{\forall}B_1, \dots, Cl_{\forall}B_m, \neg Cl_{\forall}A\}$
2. Bringe jede Formel in E in Pränex-Normalform, erhalte E' ,
3. Bringe jede Formel in E' in Skolem-Normalform, erhalte E'' . (Die Matrizen der Formeln sind in KNF:)
4. Lasse die \forall -Präfixe weg; schreibe die verbleibenden Konjunktionen von Disjunktionen (von Literalen) als Mengen von Klauseln; bilde die Vereinigung dieser Klauselmengen; erhalte \overline{E} .
5. Versuche, aus \overline{E} mittels Variantenbildung von Klauseln und Resolution die leere Klausel herzuleiten.

Zu Schritt 3 beachte man, daß die bei der Skolemisierung neu eingeführten Funktionssymbole (über ganz E' !) paarweise verschieden sind. (Denn bei Einführung eines jeden wird die Signatur erweitert.)

Die Variantenbildung in Schritt 5 hat den Zweck, die für die Anwendung der Resolution erforderliche Variablendisjunktheit von Klauseln herzustellen.

Das nächste Beispiel zeigt die Notwendigkeit der Variantenbildung für die Vollständigkeit des Resolutionskalküls.

Beispiel 5.29

Die Menge der beiden Klauseln $\{p(x)\}$ und $\{\neg p(f(x))\}$ hat sicherlich kein Modell, da sie der prädikatenlogischen Formel $(\forall x p(x)) \wedge (\forall x \neg p(f(x)))$ entspricht. Aber $p(x)$ und $p(f(x))$ sind nicht unifizierbar, also ist auch die leere Klausel so nicht herleitbar.

Weitere Beispiele zum Resolutionskalkül

Wir wollen die folgende logische Folgerung beweisen:

Beispiel 5.30

$$\forall x \forall y (x \subseteq y \leftrightarrow \forall u (u \epsilon x \rightarrow u \epsilon y)) \models \forall x \forall y \forall z (x \subseteq y \wedge y \subseteq z \rightarrow x \subseteq z)$$

Es handelt sich dabei um eine Aussage aus der elementaren Mengenlehre zur Transitivität der Teilmengenbeziehung. Bemerkenswert ist vielleicht, daß die Transitivität allein aus der Definition der Teilmengenrelation gefolgert werden soll, ohne zusätzliche mengentheoretische Axiome über die ϵ -Relation.

Bevor wir mit einem Beweis im Resolutionskalkül beginnen können, müssen wir die Prämisse und die Negation der Behauptung in Klauselnormalform transformieren. Dazu zerlegen wir die Prämisse in die beiden Teile

$$\forall x \forall y (x \subseteq y \rightarrow \forall u (u \epsilon x \rightarrow u \epsilon y))$$

$$\forall x \forall y (\forall u (u \epsilon x \rightarrow u \epsilon y) \rightarrow x \subseteq y)$$

Benutzen wir die Relationszeichen *conteg* und *memb* anstelle der Infixzeichen \subseteq und ϵ , so läßt sich die erste Formel direkt in die Klausel

$$\{\neg \text{conteg}(x, y), \neg \text{memb}(u, x), \text{memb}(u, y)\}$$

umschreiben.

Die zweite Formel wird nach Elimination von \rightarrow zunächst zu

$$\forall x \forall y (\exists u (u \epsilon x \wedge \neg u \epsilon y) \vee x \subseteq y)$$

und nach Skolemisierung zu

$$\forall x \forall y ((f(x, y) \epsilon x \wedge \neg f(x, y) \epsilon y) \vee x \subseteq y)$$

Nach Anwendung des Distributivgesetzes entstehen die beiden Klauseln:

$$\{\text{memb}(f(x, y), x), \text{conteg}(x, y)\}$$

$$\{\neg \text{memb}(f(x, y), y), \text{conteg}(x, y)\}$$

Die Negation der Behauptung führt zu

$$\exists x \exists y \exists z (x \subseteq y \wedge y \subseteq z \wedge \neg x \subseteq z)$$

und nach Einführung von Skolemkonstanten zu den drei Einerklauseln:

- $coneq(a, b)$
- $coneq(b, c)$
- $\neg coneq(a, c)$

Der Resolutionsbeweis (wir lassen zur Vereinfachung die Mengenklammern weg)

- | | | |
|------|---|----------------|
| (1) | $\neg coneq(x, y), \neg memb(u, x), memb(u, y)$ | [Vor.] |
| (2) | $memb(f(x, y), x), coneq(x, y)$ | [Vor.] |
| (3) | $\neg memb(f(x, y), y), coneq(x, y)$ | [Vor.] |
| (4) | $coneq(a, b)$ | [\neg Beh.] |
| (5) | $coneq(b, c)$ | [\neg Beh.] |
| (6) | $\neg coneq(a, c)$ | [\neg Beh.] |
| (7) | $\neg memb(u, a), memb(u, b)$ | [4,1] |
| (8) | $\neg memb(u, b), memb(u, c)$ | [5,1] |
| (9) | $\neg memb(f(a, c), c)$ | [6,3] |
| (10) | $memb(f(a, c), a)$ | [6,2] |
| (13) | $memb(f(a, c), b)$ | [7,10] |
| (19) | $memb(f(a, c), c)$ | [8,13] |
| (20) | \square | [19,9] |

Dieser Beweis wurde von dem automatischen Beweissystem OTTER gefunden. Die Lücken in der Nummerierung weisen darauf hin, daß zwischendurch Klauseln erzeugt wurden, die zum Beweis nichts beitragen.

Die binäre Resolutionsregel alleine liefert keinen vollständigen Kalkül, wie das nächste Beispiel zeigt.

Beispiel 5.31

Die Menge bestehend aus den beiden Klauseln

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

ist sicherlich unerfüllbar. Die binäre Resolutionsregel liefert, selbst bei beliebiger Wiederholung, als Resolventen nur Varianten der Klausel

$$\{P(y), \neg P(v)\}$$

und nie die leere Klausel. Entscheidend für dieses Phänomen ist der Teil der Definition 5.24, der die Variablendisjunktheit der Elternklauseln verlangt.

Zusammen mit der **Faktorisierungsregel**

$$\frac{\{L_1, \dots, L_k, L_{k+1}, \dots, L_n\}}{\{\sigma(L_1), \dots, \sigma(L_k)\}},$$

wobei σ so gewählt ist, daß $\sigma(L_k) = \sigma(L_{k+1}) = \dots = \sigma(L_n)$, entsteht wieder ein vollständiger, häufig eingesetzter Kalkül.

Auch das zweite Beispiel eines Resolutionsbeweises stammt aus der elementaren Mengenlehre. Es soll bewiesen werden

Beispiel 5.32

$$\forall x \forall y (x \subseteq y \wedge y \subseteq x \rightarrow x = y).$$

Als Voraussetzungen stehen dazu zur Verfügung

- die Definition der Teilmengenrelation

$$\forall x \forall y (x \subseteq y \leftrightarrow \forall u (u \in x \rightarrow u \in y))$$

und

- die Definition der Gleichheit (in der Mengenlehre ist die Gleichheit eine definierte Relation)

$$\forall x \forall y (x = y \leftrightarrow \forall u (u \in x \leftrightarrow u \in y))$$

Die Transformation liefert zunächst die schon bekannten Klauseln

- (1) $\neg \text{conteq}(x, y) \vee \neg \text{memb}(u, x) \vee \text{memb}(u, y)$
- (2) $\text{memb}(f(x, y), x) \vee \text{conteq}(x, y)$
- (3) $\neg \text{memb}(f(x, y), y) \vee \text{conteq}(x, y)$

Ersetzen wir das Infixzeichen $=$ durch eq , so liefert die Definition der Gleichheit die Klausel

- (4) $eq(x, y) \vee \text{memb}(g(x, y), x) \vee \text{memb}(g(x, y), y)$
- (5) $eq(x, y) \vee \neg \text{memb}(g(x, y), x) \vee \neg \text{memb}(g(x, y), y)$
- (6) $\neg eq(x, y) \vee \text{memb}(u, x) \vee \text{memb}(u, y)$
- (7) $\neg eq(x, y) \vee \neg \text{memb}(u, x) \vee \neg \text{memb}(u, y)$

mit der neuen Skolemfunktion $g(x, y)$. Die Negation der Behauptung schließlich führt zu den Klauseln

- (8) $\text{conteq}(a, b)$
- (9) $\text{conteq}(b, a)$
- (10) $\neg eq(a, b)$

mit den Skolemkonstanten a, b, c

(11)	$\neg memb(x, a) \vee memb(x, b)$	[8,1]
(12)	$\neg memb(x, b) \vee memb(x, a)$	[9,1]
(18)	$memb(g(a, x), b) \vee eq(a, x) \vee memb(g(a, x), x)$	[11,4]
(23)	$\neg memb(g(x, b), a) \vee eq(x, b) \vee \neg memb(g(x, b), x)$	[11,5]
(28)	$memb(g(a, b), b) \vee eq(a, b)$	[Fak 18]
(29)	$\neg memb(g(a, b), a) \vee eq(a, b)$	[Fak 23]
(61)	$memb(g(a, b), b)$	[28,10]
(62)	$memb(g(a, b), a)$	[61,12]
(69)	$eq(a, b)$	[29,62]
(70)	\square	[69,10]

Der Beweis wurde wieder mit dem Beweiser OTTER gefunden, der nicht die Mengennotation verwendet und außerdem die Faktorisierungsregel (Fak) benutzt. Man erhält aus dem obigen Beweis einen Beweis ohne Faktorisierung, wenn man die Mengenschreibweise benutzt und die Beweisschritte (28) und (29) einfach wegläßt. Aus (18) und (10) entsteht direkt (61), ebenso kommt man von (23) und (62) direkt zu (69).

Bemerkung

Die Behandlung der Gleichheit erfolgt prinzipiell durch Einbeziehen der Axiomenmenge G1 in den Kalkül. Für das genauere Verfahren (Paramodulation) sei auf die Literatur verwiesen.

5.2.3 Ein Sequenzenkalkül

Es gibt verschiedene Tableaunkalküle, in 5.1 wurde einer von ihnen vorgestellt. Es gibt auch verschiedene Sequenzenkalküle, vgl. die Bücher von [SA91] und [Gal86]. Wie im Falle der Aussagenlogik dargestellt, besteht eine enge Verwandtschaft zwischen Tableau- und bestimmten Sequenzenkalkülen, die in der Prädikatenlogik erhalten bleibt: Von Tableaux kann man zu Blocktableaux übergehen, und von diesen zu Beweisbäumen für Sequenzen. Man erhält auf diese Weise einen zum Tableauxkalkül 5.1 gehörigen Sequenzenkalkül (der allerdings die Eigenschaft hat, daß eine pfadschließende Substitution auf den gesamten Beweisbaum anzuwenden ist, diesen also ändert.)

Wir betrachten gleich Formeln, die auch das \doteq -Zeichen enthalten können.

Die beiden folgenden Definitionen sind die prädikatenlogische Fortsetzung der aussagenlogischen Definitionen 3.40) und 3.41.

Definition 5.33 (Sequenz)

Eine *Sequenz* wird notiert als eine Folge zweier endlicher Mengen prädikatenlogischer

Formeln getrennt durch das Symbol \rightarrow :

$$\Gamma \rightarrow \Delta.$$

Γ wird Antezedent und Δ Sukzedent genannt. Sowohl links wie rechts vom Sequenzenpfeil \rightarrow kann auch die leere Folge stehen.

Definition 5.34 (Auswertung von Sequenzen)

Sei \mathcal{D} eine prädikatenlogische Struktur und β eine Variablenbelegung:

$$val_{\mathcal{D},\beta}(\Gamma \rightarrow \Delta) = val_{\mathcal{D},\beta}(\bigwedge \Gamma \rightarrow \bigvee \Delta)$$

Es gelten die üblichen Vereinbarungen für leere Disjunktionen und Konjunktionen.

Definition 5.35 (Der Sequenzenkalkül S)

Die Regeln des Sequenzenkalküls sind in den Abbildungen 5.1, 5.2 und 5.3 zusammengestellt.

axiom

$$\frac{}{\Gamma, F \rightarrow F, \Delta}$$

not-left

$$\frac{\Gamma, \rightarrow F, \Delta}{\Gamma, \neg F \rightarrow \Delta}$$

not-right

$$\frac{\Gamma, F \rightarrow \Delta}{\Gamma \rightarrow \neg F, \Delta}$$

impl-left

$$\frac{\Gamma \rightarrow F, \Delta \quad \Gamma, G \rightarrow \Delta}{\Gamma, F \rightarrow G \rightarrow \Delta}$$

impl-right

$$\frac{\Gamma, F \rightarrow G, \Delta}{\Gamma \rightarrow F \rightarrow G, \Delta}$$

and-left

$$\frac{\Gamma, F, G \rightarrow \Delta}{\Gamma, F \wedge G \rightarrow \Delta}$$

and-right

$$\frac{\Gamma \rightarrow F, \Delta \quad \Gamma \rightarrow G, \Delta}{\Gamma \rightarrow F \wedge G, \Delta}$$

or-left

$$\frac{\Gamma, F \rightarrow \Delta \quad \Gamma, G \rightarrow \Delta}{\Gamma, F \vee G \rightarrow \Delta}$$

or-right

$$\frac{\Gamma \rightarrow F, G, \Delta}{\Gamma \rightarrow F \vee G, \Delta}$$

Abbildung 5.1: Aussagenlogische Sequenzenregeln

all-left

$$\frac{\Gamma, \forall xF, F(X/x) \rightarrow \Delta}{\Gamma, \forall xF \rightarrow \Delta}$$

wobei X eine neue Variable ist.

ex-right

$$\frac{\Gamma \rightarrow \exists xF, F(X/x), \Delta}{\Gamma, \rightarrow \exists xF, \Delta}$$

wobei X eine neue Variable ist.

all-right

$$\frac{\Gamma \rightarrow F(f(x_1, \dots, x_n)/x), \Delta}{\Gamma \rightarrow \forall xF, \Delta}$$

wobei f ein neues Funktions-
symbol ist und x_1, \dots, x_n alle
freien Variablen in $\forall xF$.

ex-left

$$\frac{\Gamma, F(f(x_1, \dots, x_n)/x) \rightarrow \Delta}{\Gamma, \exists xF \rightarrow \Delta}$$

wobei f ein neues Funktions-
symbol ist und x_1, \dots, x_n alle
freien Variablen in $\exists xF$.

Abbildung 5.2: Prädikatenlogische Sequenzenregeln

identity

$$\overline{\Gamma \rightarrow s = s, \Delta}$$

symmetry-right

$$\frac{\Gamma \rightarrow s = t, \Delta}{\Gamma \rightarrow t = s, \Delta}$$

symmetry-left

$$\frac{\Gamma, s = t \rightarrow \Delta}{\Gamma, t = s \rightarrow \Delta}$$

eq-subst-right

$$\frac{\Gamma, s = t \rightarrow F(t), \Delta}{\Gamma, s = t \rightarrow F(s), \Delta}$$

eq-subst-left

$$\frac{\Gamma, F(t), s = t \rightarrow \Delta}{\Gamma, F(s), s = t \rightarrow \Delta}$$

Abbildung 5.3: Sequenzenregeln für die Gleichheit

Die Definition eines Beweisbaum ist wörtlich dieselbe wie im aussagenlogischen Fall, Definition 3.43, mit einer kleinen Änderung in der Definition eines abgeschlossenen Beweisbaums.

Definition 5.36 (Beweisbaum)

Ein Ableitungsbaum ist ein Baum, dessen Knoten mit Sequenzen markiert sind und für jeden Knoten n die folgende Einschränkung erfüllt:

1. ist n_1 der einzige Nachfolgerknoten von n und sind $\Gamma \rightarrow \Delta$ und $\Gamma_1 \rightarrow \Delta_1$ die Markierungen von n und n_1 , dann gibt es eine Sequenzenregel

$$\frac{\Gamma_1 \rightarrow \Delta_1}{\Gamma \rightarrow \Delta}$$

2. besitzt n die beiden Nachfolgerknoten n_1 und n_2 und sind $\Gamma \rightarrow \Delta$, $\Gamma_1 \rightarrow \Delta_1$ und $\Gamma_2 \rightarrow \Delta_2$ die Sequenzen an den Knoten n , n_1 und n_2 dann gibt es eine Sequenzenregel

$$\frac{\Gamma_1 \rightarrow \Delta_1 \quad \Gamma_2 \rightarrow \Delta_2}{\Gamma \rightarrow \Delta}$$

Wir nennen einen Beweisbaum *geschlossen* oder *vollständig* wenn er zusätzlich noch die folgende Bedingung erfüllt:

3. es gibt eine Substitution σ , so daß für die Markierung A jedes Knoten n , der keinen Nachfolgerknoten hat, $\sigma(A)$ ein Axiom ist. Dazu zählen auch die beiden Gleichheitsaxiome.

Man beachte daß zunächst $A \equiv p(s) \rightarrow p(t)$ kein Axiom zu sein braucht. Ist σ aber ein Unifikator von s und t dann ist $\sigma(A) \equiv p(\sigma(s)) \rightarrow p(\sigma(t))$ zu einem Axiom wird.

Definition 5.37

Für endliche Formelmengen $\Delta, \Gamma \subseteq For_\Sigma$ sagen wir, daß die Sequenz $\Gamma \rightarrow \Delta$ in \mathbf{S} *ableitbar* ist, genau dann, wenn ein geschlossener Beweisbaum mit Wurzelmarkierung $\Gamma \rightarrow \Delta$ existiert.

In Symbolen schreiben wir dafür auch $\Gamma \vdash_S \Delta$.

Satz 5.38 (Korrektheit des Sequenzenkalküls)

Für endliche Formelmengen $\Delta, \Gamma \subseteq For_\Sigma$ ohne freie Variablen gilt:

$$\text{wenn } \Gamma \vdash_S \Delta, \text{ dann } \models \bigwedge \Gamma \rightarrow \bigvee \Delta$$

Beweis: Sei also \mathcal{T} ein geschlossener Beweisbaum für die Formel A über der Voraussetzungs­menge M . \mathcal{T} hat die Form $\sigma(\mathcal{T}_0)$ für eine geeignete Substitution σ .

Der Beweis wird durch strukturelle Induktion über den Beweisbaum geführt.

Axiome: Im einfachsten Fall besteht \mathcal{T} nur aus einer einzigen Sequenz, die dann ein Axiom sein muß. Es können die folgenden Fälle auftreten:

$$\text{axiom} \quad \frac{}{\Gamma', F \rightarrow F, \Delta'}$$

$$\text{identity} \quad \frac{}{\Gamma', \rightarrow s = s, \Delta'}$$

Der weitere Beweis gliedert sich in Fallunterscheidungen nach der ersten Regelanwendung in \mathcal{T}_0 und verläuft nach dem folgenden Muster. Sei

$$\frac{\Gamma_1 \rightarrow \Delta_1 \quad \Gamma_2 \rightarrow \Delta_2}{\Gamma \rightarrow \Delta}$$

die erste Regelanwendung auf \mathcal{T}_0 . Nach Induktionsvoraussetzung wissen wir daß

$$\bigwedge \sigma(\Gamma_1) \rightarrow \sigma(\Delta_1) \text{ und } \bigvee \sigma(\Gamma_2) \rightarrow \sigma(\Delta_2)$$

allgemeingültig sind. Daraus müssen wir auf die Allgemeingültigkeit von $\bigwedge \sigma(\Gamma) \rightarrow \bigvee \sigma(\Delta)$ schließen.

Die aussagenlogischen Regeln werden wie im Beweis von Satz 3.46 abgehandelt. Wir führen einige der restlichen Fälle ausführlich vor.

all-right Regel

$$\frac{\Gamma \rightarrow F(f(x_1, \dots, x_n)/x), \Delta'}{\Gamma \rightarrow \forall x F, \Delta'}$$

wobei f ein neues Funktionssymbol ist und x_1, \dots, x_n alle freien Variablen in $\forall x F$.

Um den Beweisbaum \mathcal{T} zu erhalten müssen wir noch die Substitution σ auf jede Sequenz anwenden. Als Induktionsvoraussetzung haben wir dann die Allgemeingültigkeit von

$$\bigwedge \sigma(\Gamma) \rightarrow F(\sigma(f(x_1, \dots, x_n))/x) \vee \bigvee \sigma(\Delta')$$

zur Verfügung. Daraus sollen wir die Allgemeingültigkeit von

$$\bigwedge \sigma(\Gamma) \rightarrow \sigma(\forall x F) \vee \bigvee \sigma(\Delta')$$

herleiten.

all-left Regel

$$\frac{\Gamma, \forall xF, F(X/x) \rightarrow \Delta'}{\Gamma, \forall xF \rightarrow \Delta'}$$

wobei X eine neue Variable ist.

Die restlichen Fälle bleiben dem Leser als Übungsaufgabe überlassen. ■

Satz 5.39 (Vollständigkeit des Sequenzenkalküls)

Es seien $M \subseteq For_\Sigma, A \in For_\Sigma$. Dann gilt

$$M \models A \Rightarrow M \vdash_S A$$

Der Beweis benutzt die Vollständigkeit des Hilbert-Kalküls ohne Gleichheit, indem Beweise im Hilbert-Kalkül zu Beweisen im Sequenzenkalkül transformiert werden. Die Transformation geschieht über die Länge des Hilbert-Kalkülbeweises. Es sei A_1, \dots, A_n mit $A_n = A$ ein Beweis von A über M .

- a) $A_n \in M$: Die Sequenz $\rightarrow A_n$ wird durch die (*einfügen*)-Regel sofort bewiesen.
- b) A_n Axiom des Hilbert-Kalküls: Die Ableitungen dieser Axiome im Sequenzenkalkül bleiben dem Leser als Übungsaufgabe überlassen.
- c) A_n ist durch die Modus-Ponens-Anwendung auf A_i und A_j mit $i < n$ und $j < n$ entstanden, wobei $A_j = A_i \rightarrow A_n$ ist. Dann erhält man $\rightarrow A_n$ im Sequenzenkalkül durch Anwendung der (*Schnitt*)-Regel (mit A_i).

$$\frac{A_i \rightarrow A_n \quad \rightarrow A_i, A_n}{\rightarrow A_n}$$

$\rightarrow A_i, A_n$ erhält man aus einem Beweis für A_i .

$A_i \rightarrow A_n$ erhält man durch Anwendung der (*Schnitt*)-Regel (mit $A_i \rightarrow A_n$).

$$\frac{A_i \rightarrow A_n, A_i \rightarrow A_n \quad A_i \rightarrow A_i \rightarrow A_n, A_n}{A_i \rightarrow A_n}$$

$A_i \rightarrow A_i \rightarrow A_n, A_n$ läßt sich durch einen Beweis für $\rightarrow A_i \rightarrow A_n$ beweisen und

$A_i \rightarrow A_n, A_i \rightarrow A_n$ durch die (*→ links*)-Regel.

- d) A_n ist durch die Anwendung der Gen-Regel auf A_i mit $i < n$ entstanden. Dann beweist man $\rightarrow A_n$ im Sequenzenkalkül durch Anwendung der (*→ rechts*)-Regel und die verbleibende offene Prämisse analog zu dem Beweis von A_i , der nach Induktionsvoraussetzung existiert.

Bemerkung 5.40

Wollen wir beweisen, daß eine Formel A logisch folgt aus einer Menge M von Voraussetzungen, so hatten wir bisher nur die Möglichkeit betrachtet, die Sequenz $M \rightarrow A$ herzuleiten. Für unendliche Voraussetzungenmengen M ist das keine Lösung. Für diesen Fall führen wir auf die folgende Regel ein:

$$(einfügen_M): \frac{Cl_{\forall} B, \Gamma \rightarrow \Delta}{\Gamma \rightarrow \Delta}$$

mit $B \in M$

Wenn M endlich ist (z. B. $M = \{B_1, \dots, B_n\}$), kann man wiederum abkürzend einen Beweisbaum erstellen, in dessen Wurzel die Sequenz $Cl_{\forall} B_1, \dots, Cl_{\forall} B_n \rightarrow A$ steht und auf die Einfügeregel verzichten.

Die (*Schnitt*)-Regel ist nicht wirklich notwendig, man kann sie auch weglassen, der Kalkül bleibt vollständig (siehe z.B. [Gal86]).

Ohne Schnitt lassen sich dann Tableaux- und Sequenzenbeweise einfach ineinander überführen, wenn man die unterschiedliche Variablensubstitution (γ -Regel bzw. (\forall right) und (\exists left)) entsprechend berücksichtigt (siehe auch 3.46 Seite 92).

5.2.4 Übungsaufgaben

Übungsaufgabe 5.2.1

Aufgabe zum Hilbertkalkül

Unter Verwendung von Ax3 und Ax4 (siehe Def. 5.19) zeige man

$$\vdash_H Cl_{\forall} \neg A \rightarrow \neg Cl_{\forall} A.$$

Übungsaufgabe 5.2.2

Was würde sich ändern wenn man in Definition 5.24 nicht verlangen würde, daß $C_1 \cup K_1$ und $C_2 \cup K_2$ variablendisjunkt sind? Vollständigkeit, Korrektheit, keins von beiden?

Übungsaufgabe 5.2.3

In Fortsetzung der vorangegangenen Übungsaufgabe 5.2.2 betrachten wir - nur für diese Übungsaufgabe - die folgende geänderte Version der Definition von $Res(M)$ aus Definition 5.25:

$$Res'(M) = \{B \mid \text{es gibt Klauseln } C_1, C_2 \text{ aus } M, \\ \text{so daß } B \text{ eine Resolvente von } C_1, C_2 \text{ ist.}\}$$

Gegenüber der offiziellen Definition ist die Variantenbildung, d.h. die Umbenennung der Variablen in C_1, C_2 weggefallen.

Wie wird dadurch Korrektheit und Vollständigkeit des Kalküls beeinflusst?

Übungsaufgabe 5.2.4

Die in den Abbildungen 5.1 und 5.2 vorgestellten Regeln beschreiben den Sequenzkalkül, der unter dem Namen *System G* bekannt ist, siehe etwa [Gal86, Chapter 5]. In der mathematischen Logik und Beweistheorie ist eine Variante des Kalküls, das *System LK*, siehe [Gal86, Chapter 6] wesentlich bekannter. Es entsteht aus unserem System, indem die Regeln *and – left* und *or – right* jeweils durch die folgenden beiden Regeln ersetzt werden

LK-and-left-1

$$\frac{\Gamma, F \rightarrow \Delta}{\Gamma, F \wedge G \rightarrow \Delta}$$

LK-or-right-1

$$\frac{\Gamma \rightarrow F, \Delta}{\Gamma \rightarrow F \vee G, \Delta}$$

LK-and-left-2

$$\frac{\Gamma, G \rightarrow \Delta}{\Gamma, F \wedge G \rightarrow \Delta}$$

LK-or-right-2

$$\frac{\Gamma \rightarrow G, \Delta}{\Gamma \rightarrow F \vee G, \Delta}$$

Zum Kalkül *LK* gehört außerdem noch eine sogenannte *Schnittregel*, auf die wir hier allerdings nicht eingehen wollen. In dem berühmten *Schnitteliminationssatz* kann man beweisen, daß jeder Beweis in *LK* unter Benutzung der Schnittregeln in einen Beweis ohne Schnittregel transformiert werden kann. Da der Kalkül *LK* nicht die Mengenschreibweise benutzt, braucht man noch sogenannte strukturelle Regeln. Für unsere bescheidenen Zwecke brauchen wir nur die Verdoppelungsregel:

$$\frac{\Gamma, F, F \rightarrow \Delta}{\Gamma, F \rightarrow \Delta}$$

Leiten Sie in dem Kalkül *LK* die folgenden Sequenzen ab

- 1 $F \wedge G \rightarrow F \vee G$
- 2 $F \wedge G \rightarrow F \wedge G$
- 3 $F \wedge \neg F \rightarrow$

5.3 Weitere Anmerkungen zur Prädikatenlogik erster Ordnung

5.3.1 Andere Notationsformen

- Wechsel zwischen Präfix-, Infix-, Postfix-, Suffix-Notation

- Attributnotation von Argumentstellen („Objekte“)
- Anlehnungen an die natürliche Sprache. Aktions- und Situationslogiken, Conceptual Graphs

5.3.2 Metaresultate

Eine Konsequenz der Korrektheit und Vollständigkeit von **H** ist der

Satz 5.41

(Kompaktheit der PL1)

Für beliebige $M \subseteq For_\Sigma$, $A \in For_\Sigma$ gilt:

$$M \models A \Leftrightarrow E \models A \text{ für eine endliche Teilmenge } E \text{ von } M.$$

Als Spezialfall und unmittelbares Korollar ergibt sich hieraus der

Satz 5.42

(Endlichkeitssatz) Eine Menge $M \subseteq For_\Sigma$ hat genau dann ein Modell, wenn jede endliche Teilmenge von M ein Modell hat.

Satz 5.43 (Nichtcharakterisierbarkeit der Endlichkeit)

Es sei Σ eine Signatur der PL1. Dann gibt es keine Formelmengemenge $M \subseteq For_\Sigma$, so daß für jede Interpretation (D, I) über Σ gilt:

$$(D, I) \text{ ist Modell von } M \Leftrightarrow D \text{ ist endlich.}$$

Beweis

Angenommen, es gäbe so ein M . Es seien x_0, x_1, \dots paarweise verschiedene Variable, und für jedes $n \in \mathbb{N}$, $n \geq 1$,

$$A_n := \exists x_0 \exists x_1 \dots \exists x_n \left(\bigwedge_{i,j \leq n, i < j} \neg x_i \doteq x_j \right).$$

Offensichtlich gilt für jede Interpretation (D, I) :

$$(D, I) \text{ ist Modell von } A_n \Leftrightarrow \#D > n.$$

Wir betrachten

$$M \cup \{A_n \mid n \in \mathbb{N}, n \geq 1\}.$$

Jede endliche Teilmenge E dieser Menge hat ein Modell, nämlich, wenn m maximal ist mit $A_m \in E$, ein beliebiges (D, I) mit $\#D = m + 1$. Nach dem Endlichkeitssatz müßte es also ein Modell von $M \cup \{A_n \mid n \in \mathbb{N}, n \geq 1\}$ geben. Das aber kann nicht sein, denn

- M hat nach Voraussetzung genau die (D, I) mit endlichem D
- $\{A_n | n \in \mathbb{N}, n \geq 1\}$ hat genau die (D, I) mit unendlichem D

zu Modellen.

■

5.3.3 Sorten

Bei der Axiomatisierung komplizierter Bereiche und Begriffe bedient man sich mit Erfolg eines zusätzlichen syntaktischen Hilfsmittels: Man verwendet *Sorten*, um die zu betrachtenden Gegenstandsbereiche von vornherein als *aufgeteilt* oder *typisiert* oder eben *sortiert* aufzufassen. Das ist schon von einfachen Beispielen aus der Mathematik her geläufig. Etwa unterscheidet man in der Geometrie Punkte, Geraden und Ebenen voneinander, in der Numerik und in den Programmiersprachen werden verschiedene Zahlentypen unterschieden: natürliche Zahlen, ganze Zahlen, rationale, reelle Zahlen usw.; in der linearen Algebra schließlich werden Skalare, Vektoren, Matrizen voneinander abgegrenzt. In allen Fällen soll dabei eine intendierte Klassifikation des Gegenstandsbereichs schon syntaktisch vorweggenommen werden.

Die Ausdrucksstärke der Logik wird durch die Hinzunahme von Sorten nicht erhöht. Eine Formel über einer sortierten Signatur läßt sich in eine ohne Sorten, aber mit zusätzlichen Prädikatsymbolen, in der Weise übersetzen, daß die erste genau dann ein Modell hat, wenn das für die zweite zutrifft. Doch kommt man bei größeren Anwendungen wohl kaum ohne Sorten aus. Sie zu verwenden,

- entspricht den Prinzipien des Entwurfs von Informatiksystemen;
- macht größere Spezifikationen lesbarer (oder überhaupt erst lesbar);
- gestattet ein *modulares* Vorgehen, insbesondere beim gleichzeitigen Arbeiten mit verschiedenen Spezifikationen.

Beispiel 5.44
(Lineare Algebra)

Signatur:

- Die Menge der Sorten bestehe aus S (für Skalar) und V (für Vektor),
- die Funktion $+$ habe die Sortierung VVV (Vektoraddition),

- \cdot habe die Sortierung SVV (Skalarmultiplikation),
- Konstanten $0^V, a_1, a_2, a_3$ der Sorte V und
- eine Konstante 0^S .

Die lineare Unabhängigkeit der Vektoren a_1, a_2, a_3 wird dann durch die Formel:

$$\forall x_1^S \forall x_2^S \forall x_3^S (x_1^S \cdot a_1 + x_2^S \cdot a_2 + x_3^S \cdot a_3 = 0^V \rightarrow (x_1^S = 0^S \wedge x_2^S = 0^S \wedge x_3^S = 0^S))$$

ausgedrückt.

Die Eigenschaft von a_1, a_2, a_3 , ein Erzeugendensystem zu sein, wird beschrieben durch:

$$\forall x^V \exists x_1^S \exists x_2^S \exists x_3^S (x^V = x_1^S \cdot a_1 + x_2^S \cdot a_2 + x_3^S \cdot a_3).$$

Man kann noch ein übriges tun, um die prädikatenlogische Notation der üblichen anzunähern. Man vereinbart, daß zwei unmittelbar hintereinander geschriebene Terme, wovon der erste die Sorte S und der zweite die Sorte V besitzt, als Skalarmultiplikation gelesen werden sollen und kann den Punkt \cdot weglassen. Außerdem ist die Kennzeichnung von Variablen der beiden Sorten durch Superskripte umständlich. Man vereinbart griechische Buchstaben für Skalare und lateinische Buchstaben für Vektoren zu verwenden. Dann liest sich die Basiseigenschaft von a_1, a_2, a_3 als:

$$\begin{aligned} \forall \lambda_1 \forall \lambda_2 \forall \lambda_3 (\lambda_1 a_1 + \lambda_2 a_2 + \lambda_3 a_3 = 0^V \rightarrow (\lambda_1 = 0^S \wedge \lambda_2 = 0^S \wedge \lambda_3 = 0^S)) \\ \forall x \exists \lambda_1 \exists \lambda_2 \exists \lambda_3 (x = \lambda_1 a_1 + \lambda_2 a_2 + \lambda_3 a_3) \end{aligned}$$

Das Beispiel zur linearen Algebra zeigt noch ein weiteres Phänomen einer benutzerfreundlichen Darstellung von Formeln auf. Üblicherweise würde man in den obigen Formeln keinen notationellen Unterschied machen zwischen der skalaren und der vektoriellen 0, denn schließlich läßt sich aus dem Zusammenhang erkennen, was gemeint ist. Man sagt, daß 0 ein *überladenes Symbol* ist und wir eine *überladene Notation (overloading)* verwenden. In den Beispielformeln kommt nur die Vektoraddition vor, käme auch die Addition in dem Skalarenkörper vor, so würden wir auch für $+$ eine überladene Notation benutzen.

Definition 5.45

Eine *ordnungssortierte Signatur* Σ ist ein Tupel

$$\Sigma = (S_\Sigma, \leq, F_\Sigma, P_\Sigma, \alpha_\Sigma),$$

wo $S_\Sigma, F_\Sigma, P_\Sigma$ paarweise disjunkte Mengen von Symbolen sind und ferner gilt:

1. S_Σ besteht aus endlich vielen *Sortensymbolen*;
2. \leq ist eine partielle Ordnung auf der Menge S_Σ ;
3. F_Σ, P_Σ sind wie früher die Mengen der Funktions- bzw. Prädikatsymbole;
4. α_Σ ist jetzt nicht mehr einfach die Stelligkeitsfunktion, sondern gibt zu jedem Funktions- bzw. Prädikatsymbol seine Sortierung an:

$$\alpha_\Sigma : F_\Sigma \cup P_\Sigma \rightarrow S_\Sigma^*$$

wobei $\alpha_\Sigma(f) \in S_\Sigma^+$, wenn $f \in F_\Sigma$.

$\alpha_\Sigma(f) = Z_1 \dots Z_n Z'$ bedeutet: f ist ein Funktionssymbol für Funktionen, welche n -Tupeln von Elementen der Sorten Z_1, \dots, Z_n (respective) ein Element der Sorte Z' zuordnen.

$\alpha_\Sigma(p) = Z_1, \dots, Z_n$ bedeutet: p ist gedacht zur Notation von Mengen von n -Tupeln von Elementen respective der Sorten Z_1, \dots, Z_n . Im Falle $n = 0$ ist (wie früher) p ein aussagenlogisches Atom.

Man nimmt ferner an, daß auch die Menge der Variablen sortiert ist: Zu jedem $Z \in S_\Sigma$ gibt es unendlich viele Variable der Sorte Z , und für verschiedene Sorten sind diese Variablenmengen disjunkt.

Definition 5.46 (Sortierte Interpretationen)

Eine *Interpretation* über der sortierten Signatur $\Sigma = (S_\Sigma, F_\Sigma, P_\Sigma, \alpha_\Sigma)$ ist ein Paar (D, I) , wo gilt:

$D = \{D_Z \mid Z \in S_\Sigma\}$ ist eine Familie nichtleerer (den Sorten $Z \in S_\Sigma$ entsprechender) Mengen,

I ordnet jedem Funktions- und Prädikatsymbol eine sortengerechte Bedeutung zu:

$$I(f) : D_{Z_1} \times \dots \times D_{Z_n} \rightarrow D_{Z'} \text{ wenn } \alpha(f) = Z_1 \dots Z_n Z'$$

$$I(p) \subseteq D_{Z_1} \times \dots \times D_{Z_n} \text{ wenn } \alpha(p) = Z_1 \dots Z_n.$$

Definition 5.47 (Sortierte Terme)

Die Sortierung wird auf Terme übertragen:

- Ist x eine Variable der Sorte Z , dann ist x Term von der Sorte Z

- Sind t_1, \dots, t_n Terme der Sorten respective Z_1, \dots, Z_n und f ein Funktionssymbol mit $\alpha_f = Z_1 \cdots Z_n Z'$, dann ist $f(t_1, \dots, t_n)$ ein Term der Sorte Z' .

Entsprechend geschieht die sortengerechte Definition der Formeln.

Es lassen sich nun die weiteren, bisher behandelten Begriffe der Prädikatenlogik übertragen, und die entsprechenden Resultate gelten. Insbesondere müssen Substitutionen sortengerecht sein, d. h.: x und $\sigma(x)$ sind von derselben Sorte. Entsprechend ergeben sich Bedingungen für Homomorphismen zwischen Interpretationen.

Man kann zusätzlich auf den Sorten eine Ordnungsrelation einführen und gelangt dann zu *ordnungssortierten Logiken*. Das ermöglicht eine flexiblere Termbildung und Unifikation, als bei der obigen „strikten“ Sortierung möglich wäre.

Das Arbeiten mit sortierten und ordnungssortierten Logiken hat sich in vielfacher Weise gewinnbringend auf das automatische Beweisen ausgewirkt.

5.3.4 Übungsaufgaben

Übungsaufgabe 5.3.1

Sei Σ die leere Signatur, d.h. das einzige Prädikatszeichen in Formeln $A \in For_\Sigma$ ist das Gleichheitszeichen \doteq .

Zeigen Sie: Es gibt kein $A \in For_\Sigma$, welches genau die Interpretationen (D, I) mit unendlichem D zu Modellen hat.

Übungsaufgabe 5.3.2

Es sei Σ eine Signatur der PL1, so daß P_Σ das zweistellige Prädikatsymbol \equiv enthält. Dann gibt es kein $G \subseteq For_\Sigma$, so daß \doteq in keinem $A \in G$ auftritt und für alle Interpretationen (D, I) über Σ gilt:

$$(D, I) \text{ ist Modell von } G \Leftrightarrow I(\equiv) = \{(d, d) \mid d \in D\}$$

(d. h. $I(\equiv)$ ist die Gleichheit auf D).

Das heißt, wenn man die Gleichheit nicht als logisches Zeichen zur Verfügung hat, dann kann man sie auch nicht definieren, gleichgültig wie man Σ wählt.

5.4 Anwendung

5.4.1 Programmverifikation und -Synthese

5.4.2 Verifikation eines Schaltkreises

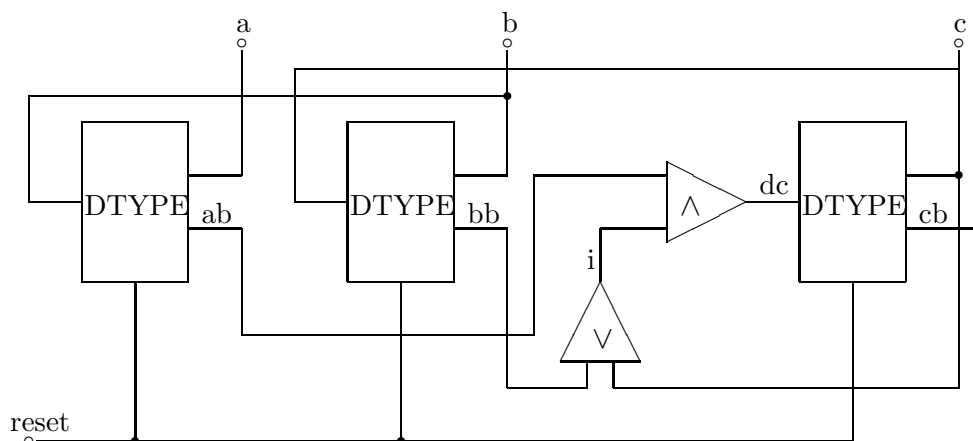


Abbildung 5.4: 3-Bit-Ring-Zähler

Abbildung 5.4 zeigt einen Schaltkreis für einen 3-Bit Ring Zähler. An den Ausgängen a, b, c sollen in aufeinanderfolgenden Taktzeitpunkten die Wertetripel $000, 001, 011, 111, 110, 100, 000$ usw. auftreten, wenn kein reset erfolgt. Die abgebildete Schaltung ist aus der Arbeit [SGS⁺92] entnommen. Dort wird die Schaltung in der Hardwarebeschreibungssprache FUNNEL beschrieben und zur logischen Herleitung das auf ordnungssortierter Algebra basierende Beweissystem 2OBJ benutzt. Wir wollen in diesem Abschnitt vorführen, wie Darstellung und Inferenz direkt in der Prädikatenlogik aussehen.

Das Schaltelement $DTYPE(d, reset, q, qb)$ hat vier Ein/Ausgänge, deren Abhängigkeiten durch die folgenden Formeln definiert ist:

Der Quantor $\forall t$ soll dabei über alle Zeitpunkte laufen und $next(t)$ bezeichnet den nächsten Zeitpunkt nach t . In dem 3-Bit Zähler kommen die folgenden Instanzen dieses Schaltelements vor:

$$\begin{aligned} &DTYPE(b, reset, a, ab) \\ &DTYPE(c, reset, a, bb) \\ &DTYPE(dc, reset, c, cb) \end{aligned}$$

was zur folgenden prädikatenlogischen Beschreibung führt:

$$\begin{aligned}
&\forall t(a(\text{next}(t)) \leftrightarrow b(t) \wedge \neg \text{reset}(t)) \\
&\forall t(ab(t) \leftrightarrow \neg(a(t))) \\
&\forall t(b(\text{next}(t)) \leftrightarrow c(t) \wedge \neg \text{reset}(t)) \\
&\forall t(bb(t) \leftrightarrow \neg(b(t))) \\
&\forall t(c(\text{next}(t)) \leftrightarrow dc(t) \wedge \neg \text{reset}(t)) \\
&\forall t(cb(t) \leftrightarrow \neg(c(t))) \\
&\forall t(i(t) \leftrightarrow (bb(t) \vee c(t))) \\
&\forall t(dc(t) \leftrightarrow (ab(t) \wedge i(t))) \\
&\forall t(q(\text{next}(t)) \leftrightarrow d(t) \wedge \neg \text{reset}(t)) \\
&\forall t(qb(t) \leftrightarrow \neg(q(t)))
\end{aligned}$$

Die Eigenschaften der Schaltung, die man durch formale Ableitung aus den Axiomen verifizieren möchte, sind:

$$\begin{aligned}
&\forall t(\text{reset}(t) \rightarrow \neg a(\text{next}(t)) \wedge \neg b(\text{next}(t)) \wedge \neg c(\text{next}(t))) \\
&\forall t(\neg \text{reset}(t) \wedge \neg a(t) \wedge \neg b(t) \wedge \neg c(t) \rightarrow \neg a(\text{next}(t)) \wedge \neg b(\text{next}(t)) \wedge \\
&c(\text{next}(t))) \\
&\forall t(\neg \text{reset}(t) \wedge \neg a(t) \wedge \neg b(t) \wedge c(t) \rightarrow \neg a(\text{next}(t)) \wedge b(\text{next}(t)) \wedge c(\text{next}(t))) \\
&\forall t(\neg \text{reset}(t) \wedge \neg a(t) \wedge b(t) \wedge c(t) \rightarrow a(\text{next}(t)) \wedge b(\text{next}(t)) \wedge c(\text{next}(t))) \\
&\forall t(\neg \text{reset}(t) \wedge a(t) \wedge b(t) \wedge c(t) \rightarrow a(\text{next}(t)) \wedge b(\text{next}(t)) \wedge \neg c(\text{next}(t))) \\
&\forall t(\neg \text{reset}(t) \wedge a(t) \wedge b(t) \wedge \neg c(t) \rightarrow a(\text{next}(t)) \wedge \neg b(\text{next}(t)) \wedge \neg c(\text{next}(t))) \\
&\forall t(\neg \text{reset}(t) \wedge a(t) \wedge \neg b(t) \wedge \neg c(t) \rightarrow \neg a(\text{next}(t)) \wedge \neg b(\text{next}(t)) \wedge \neg c(\text{next}(t)))
\end{aligned}$$

5.4.3 Anwendung in der Mathematik

Mathematische Probleme eignen sich besonders gut als Testprobleme für automatische Beweiser: sie sind schon in einer abstrakten Formulierung gegeben und können leicht und direkt durch prädikatenlogische Formeln kodiert werden. Die in anderen Situationen erforderliche Formalisierung des Anwendungsbereichs wurde durch die jahrtausende lange Entwicklungsgeschichte der Mathematik bereits vorweggenommen. Deswegen war es von den Anfängen an sehr beliebt automatischen Beweiser mit mathematischen Problemen zu füttern. Am Anfang konnten die Programme nur Beweise, die schon lange bekannt und meist trivial waren, nachvollziehen. Allmählich konnten Maschinen auch mathematische Fragen beantworten, auf die noch kein Mensch bisher eine Antwort gefunden hatte. Die Entgegnung, es handle sich dabei um Fragen, für die sich kein Mathematiker je ernsthaft interessiert habe, war in den Anfangsjahren nicht ganz unbegründet. Das änderte sich aber spätestens am 10. Dezember 1996 als den Lesern der *New York Times* auf der Titelseite ein ungewöhnliche Schlagzeile in die Augen sprang. Die mathematische Vermutung, daß jede Robbins Algebra eine Boole'sche Algebra

ist wurde von einer Maschine bewiesen. Über 60 Jahre lang hatten sich auch namhafte Mathematiker daran die Zähne ausgebissen, jetzt war die Lösung in einer gemeinsamen Anstrengung der beiden automatischen Beweiser Otter und EQP an den Argonne National Laboratories gefunden worden, [McC96]. Wir wollen uns mit diesem Problem und seine Lösung etwas ausführlicher beschäftigen.

Definition 5.48

Eine Algebra mit einer einstelligen Funktion \neg und einer zweistelligen Funktion \vee heißt eine *Robbins Algebra*, wenn sie die folgenden Axiome erfüllt

R1 $x \vee y = y \vee x$

R2 $(x \vee y) \vee z = x \vee (y \vee z)$

R3 $\neg[\neg(x \vee y) \vee \neg(x \vee \neg y)] = x$

Die Beschäftigung mit diesem Axiomensystem war motiviert durch vorangegangene Arbeiten von E. Huntington. Er hatten in seinen Arbeiten [Hun33b] und [Hun33a] nachgewiesen, daß die Formeln **H1** - **H3** eine Axiomatisierung der Klasse der Boole'schen Algebren liefern

H1 $x \vee y = y \vee x$

H2 $(x \vee y) \vee z = x \vee (y \vee z)$

H3 $\neg(\neg x \vee y) \vee \neg(\neg x \vee \neg y) = x$

Der Versuch Axiom **H3** durch das doch sehr ähnlich aussehende **R3** zu ersetzen wird Herbert Robbins zugeschrieben. Einem weiteren Publikum wurde die Robbinsche Vermutung, daß die Axiome **R1** bis **R3** ein Axiomensystem für die Boole'schen Algebren liefern, durch die Veröffentlichung in [HMT71][p. 245].

Einige erste Schritte zum Beweis dieser Vermutung sind einfach. Jede Boole'sche Algebra erfüllt offensichtlich die Robbinschen Axiome. Die interessante Frage ist also: ist jede Robbins Algebra eine Boole'sche Algebra?

Lemma 5.49

In jeder Robbins Algebra gilt

$$\forall x \exists z (\neg z = x)$$

Beweis: Zu einem gegebenen x wähle man $z = \neg(x \vee x) \vee \neg(x \vee \neg x)$. Die Behauptung folgt jetzt mit Axiom **R3**. ■

Die zunächst erzielten Resultate, damit meine ich die vor der Lösung erzielten, waren typischerweise von der Form

wenn eine Robbins Algebra eine zusätzliche Bedingung B erfüllt, dann ist sie schon eine Boolesche Algebra.

Hier sind zwei Beispiel nach diesem Muster.

Lemma 5.50

ist in einer Robbins Algebra die Formel

$$\forall x, y \neg x = \neg y \rightarrow x = y$$

wahr, dann ist sie eine Boolesche Algebra.

Beweis: Ersetzt man x durch $\neg x$ in **R3**, so erhält man

$$\neg[\neg(\neg x \vee y) \vee \neg(\neg x \vee \neg y)] = \neg x$$

Aus der Voraussetzung des Lemmas folgt daraus

$$[\neg(\neg x \vee y) \vee \neg(\neg x \vee \neg y)] = x$$

Das ist aber genau das Huntingtonsche Axiom **H3**. ■

Satz 5.51

Jede endliche Robbins Algebra ist eine Boolesche Algebra.

Beweis: Nach Lemma 5.49 ist die Funktion $f(x) = \neg x$ in jeder Robbins Algebra surjektiv. Ist die Algebra endlich, dann ist F sogar injektiv. Also ist die Voraussetzung von Lemma 5.50 erfüllt. ■

Das nächste Lemma ist eine Spur komplizierter zu beweisen. Er wurde zuerst von dem Theorembeweiser Otter gefunden. Die kommentierten Otterbeweise sind in [Win90], für den menschlichen Leser kaum nachvollziehbar, enthalten.

Lemma 5.52

Falls es in einer Robbins Algebra \mathcal{R} ein Element a gibt, so daß

$$\forall x(a \vee x = x)$$

gilt, dann ist \mathcal{R} eine Boolesche Algebra.

Beweis: Wir erwähnen Anwendungen der Kommutativität und Assoziativität nicht.

Wir zeigen zunächst, daß in \mathcal{R} gilt

$$\forall x(\neg(\neg x \vee \neg\neg x) = a) \quad (5.1)$$

Wir substituieren dazu in dem Robbinschen Axiom R3 a für x und x für y , das ergibt

$$\neg(\neg(a \vee x) \vee \neg(\neg x \vee a)) = a$$

Durch zweimalige Anwendung der Voraussetzung entsteht direkt 5.1. Nächstes Teilziel ist die Gleichung

$$\forall x\neg\neg(x \vee \neg\neg x) = \neg\neg x \quad (5.2)$$

Um das einzusehen substituieren wir in R3 wieder x für y , aber diesmal $\neg\neg x$ für x :

$$\neg[\neg(\neg\neg x \vee x) \vee \neg(\neg\neg x \vee \neg x)] = \neg\neg x$$

Auf der linken Seite können wir die zweite Disjunktion nach 5.1 durch a ersetzen und erhalten

$$\neg[\neg(\neg\neg x \vee x) \vee a] = \neg\neg x$$

woraus mit der vorausgesetzten Eigenschaft von a jetzt sofort 5.2 folgt. Das nächste Etappenziel ist

$$\forall x\neg\neg\neg x = \neg x \quad (5.3)$$

Wir beginnen wieder mit R3 und nehmen diesmal die Ersetzungen $x \Rightarrow \neg x$ und $y \Rightarrow \neg\neg x$ vor:

$$\neg[\neg(\neg x \vee \neg\neg x) \vee \neg(\neg x \vee \neg\neg\neg x)] = \neg x$$

Wegen 5.1 können wir die erste Disjunktion durch a ersetzen:

$$\neg[a \vee \neg(\neg x \vee \neg\neg\neg x)] = \neg x$$

Nach Eigenschaft von a folgt

$$\neg\neg(\neg x \vee \neg\neg\neg x) = \neg x$$

Benutzen wir 5.2 mit $\neg x$ substituiert für x , so ergibt sich 5.3. Wir behaupten, daß dann auch

$$\forall x \neg\neg = x \tag{5.4}$$

Ist nämlich ein beliebiges x gegeben, so besorge man sich nach Lemma 5.49 ein y mit $\neg y = x$. Damit ist die rechte Seite von 5.4 jetzt $\neg\neg\neg y$ nach 5.3 gleich $\neg y$ und damit auch gleich x . Das Gesamtergebnis folgt jetzt unmittelbar aus Übungsaufgabe 5.1.9. ■

Lemma 5.53

Wenn eine der beiden folgenden Bedingungen in einer Robbins Algebra erfüllt sind, dann ist sie schon eine Boolesche Algebra.

1. $\exists x(\neg(x \vee x) = \neg x)$
2. $\exists x, y(\neg(x \vee y) = \neg y)$

Das Lemma wurde ebenfalls in [Win90] mit Unterstützung durch das automatische Beweissystem *Otter*. Der Beweis der Robbinschen Vermutung gelang, in dem der Gleichungsbeweiser EQP die zweite Bedingung des Lemmas aus den Axiomen herleitete.

Weitere Teilergebnisse wurden berichtet in [Win92].

Eine umfangreiche Liste mathematischer Theoreme, die am Argonne National Laboratory mit Hilfe automatischer Beweiser gezeigt wurden findet man auf einer speziellen Webseite.

5.4.4 Semantische Technologien

Das Ziel Semantischer Technologien (engl. semantic technologies) ist es elektronische Informationen, wie sie z.B. im *world wide web* aber auch in firmeneigenen Netzen (*intranets*) präsent sind, genauer zu beschreiben. Eine genauere Beschreibung soll das Auffinden von Informationen, die Zusammenführung von Informationen aus unterschiedlichen Quellen erleichtern und die Erschließung impliziten Wissens ermöglichen. Bekannt geworden ist dieses Thema unter dem Stichwort *semantic web*. Da aber inzwischen die Skepsis wächst,

ob die anvisierten Ziele im weltweiten Netz erreichbar sind und sich zudem auch andere Anwendungsfelder auftun, benutzen wir lieber die weniger eingeschränkte Bezeichnung *semantische Technologien*.

Ein zentraler Bestandteil semantischer Technologien ist die 2004 vom W3C (World Wide Web Consortium) standardisierte Wissensrepräsentationssprache OWL. Das Akronym OWL steht dabei, etwas verdreht für *web ontology language*. Genauer besehen gibt es drei Varianten von OWL: OWL Full, OWL DL und OWL Lite. Wir konzentrieren uns hier auf OWL DL. Die umfangreichere Sprache OWL Full ist erstens nicht entscheidbar und wirft noch ungeklärte Probleme in ihrer Semantik auf. OWL DL kann interpretiert werden als eine entscheidbare Teilmenge der Prädikatenlogik erster Stufe. Bevor wir den gesamten Sprachumfang von OWL DL betrachten, schauen wir uns den einfachsten, aber schon typischen Kern davon an. In der Forschung wird diese Sprache mit \mathcal{ALC} bezeichnet. Die Grundbegriffe dabei sind

1. *Konzepte*, darunter kann man sich in erster Näherung Mengen oder einstellige Prädikate vorstellen,
2. *Rollen*, das sind im Wesentlichen binäre Relationen und
3. *Konstanten*, die Objekte, oder wie man in diesem Zusammenhang sagt, Ressourcen bezeichnen.

Die volle Version von OWL DL, in der Forschung mit $\mathcal{SHOIQ}(D)$ bezeichnet, wird später erklärt.

Definition 5.54 (\mathcal{ALC} -Ausdrücke)

Zu einem vorgegebenen Vokabular $V = \mathbf{CURUN}$, von Konzepten \mathbf{C} , Rollen \mathbf{R} und Konstanten \mathbf{N} definieren wir die Menge der \mathcal{ALC} -Konzeptausdrücke

1. jedes Konzeptsymbol C aus \mathbf{C} ist ein \mathcal{ALC} -Konzeptausdruck,
2. \top und \perp sind \mathcal{ALC} -Konzeptausdrücke,
3. sind C_1, \dots, C_k \mathcal{ALC} -Konzeptausdrücke, dann sind auch $C_1 \sqcap \dots \sqcap C_k$, $C_1 \sqcup \dots \sqcup C_k$ und $\neg C_1$ \mathcal{ALC} -Konzeptausdrücke,
4. ist C ein \mathcal{ALC} -Konzeptausdruck, R ein Rollensymbol aus \mathbf{R} , dann sind auch $\exists R.C$ und $\forall R.C$ \mathcal{ALC} -Konzeptausdrücke.

\mathcal{ALC} -Aussagen werden durch die beiden folgenden Regeln definiert.

5. ist C ein Konzeptausdruck und $c \in \mathbf{N}$ eine Konstante, dann ist $C(c)$ eine \mathcal{ALC} -Aussage,
6. ist R ein Rollensymbol und sind $c_1, c_2 \in \mathbf{N}$ Konstanten, dann ist $R(c_1, c_2)$ eine \mathcal{ALC} -Aussage,

Beispiel 5.55

Für dieses Beispiel benutzen wir Begriffe, die in einem juristischen Informationssystem relevant sein könnten.

1. $\mathbf{C} = \{U, E, AE\}$.
Die Buchstaben stehen als Abkürzungen:
 U für Urteil
 E für Einspruch
 AE für abgelehnten Einspruch
2. $\mathbf{R} = \{R, S\}$.
Die Buchstaben stehen als Abkürzungen:
 R für ist Revision von
 S für ist Einspruch zu
3. $\mathbf{N} = \{Az5.55, Az2020, E0417\}$.

Beispiel eines \mathcal{ALC} -Konzeptausdrucks

$$C_1 = U \sqcap \exists R.U \sqcap \forall S.AE$$

Intuitiv steht das Konzept C_1 für alle Urteile, die Urteile zu einem Revisionsverfahren sind, zu denen alle Einsprüche abgelehnt wurden.

Der \mathcal{ALC} -Ausdruck $C_1(Az5.55)$ soll sagen, daß es sich bei der durch $Az5.55$ bezeichneten Ressource um ein Urteil handelt, das die an C_1 gestellten Filterbedingungen erfüllt.

Der \mathcal{ALC} -Ausdruck $R(Az5.55, Az2020)$ soll sagen, daß Urteil $Az5.55$ ein Revisionsurteil zu $Az2020$ ist und $S(Az5.55, E0417)$ besagt, daß die durch $E0417$ bezeichnete Ressource ein Einspruch gegen das Urteil $Az5.55$ ist.

Man sieht, daß in \mathcal{ALC} auf die Benutzung von Variablen verzichtet wird. In dem Ausdruck $\exists R.C$ wird nicht gesagt *es gibt eine Rolle*. Das würde auch keinen Sinn machen, da R für einen im Vokabular festgelegten Rollenbezeichner steht. Der Ausdruck $\exists R.C$ kann vielmehr in natürlicher Sprache umschrieben werden als *es gibt einen Rollenfüller für die Rolle R der ein Element des Konzeptes C ist*.

Wir geben eine präzise Semantikdefinition für \mathcal{ALC} -Konzepte und -Ausdrücke, indem wir sie in prädikatenlogische Formeln übersetzen. Wir übernehmen dabei das Vokabular V unverändert. Nur fassen wir jetzt die Symbole in \mathbf{C} als einstellige, die in \mathbf{R} als zweistellige Prädikatszeichen auf, während die Zeichen in \mathbf{N} direkt als Konstantensymbole auftreten.

Bei der Transformation in Prädikatenlogik wird die unterdrückte Variable explizit. Jedem \mathcal{ALC} -Konzeptausdruck C ordnen wir eine prädikatenlogische Formel $C^0(x)$ zu, welche genau die freie Variable x enthält.

C (\mathcal{ALC} – Konzeptausdruck)	$C^0(x)$ (prädikatenlogische Formel)
$C \in \mathbf{C}$	$C(x)$
\top	$\mathbf{1}$
\perp	$\mathbf{0}$
$C_1 \sqcap C_2$	$C_1(x) \wedge C_2(x)$
$C_1 \sqcup C_2$	$C_1(x) \vee C_2(x)$
$\neg C$	$\neg C(x)$
$\exists R.C$	$\exists y(R(x, y) \wedge C^0(y/x))$
$\forall R.C$	$\forall y(R(x, y) \rightarrow C^0(y/x))$
A (\mathcal{ALC} – Aussage)	A^0 (prädikatenlogische Formel)
$C(c)$	$C^0(c)$
$R(c, d)$	$R(c, d)$

Im Unterschied zum Vorgehen in der Prädikatenlogik wird bei den meisten Konzeptlogiken angenommen, daß verschiedene Konstantensymbole auch verschiedene Objekte bezeichnen. Zu der Übersetzung eines \mathcal{ALC} -Konzeptausdrucks C muß man also noch die Ungleichungen $\neg(c_1 = c_2)$ für je zwei verschiedene Konstantensymbole $c_1, c_2 \in N$ hinzunehmen.

Definition 5.56 (*SHOIQ*-Ausdrücke)

Zu einem vorgegebenen Vokabular $V = \mathbf{C} \cup \mathbf{R} \cup \mathbf{N}$, von Konzepten \mathbf{C} , Rollen \mathbf{R} und Konstanten \mathbf{N} definieren wir:

1. die Menge der *SHOIQ*-Konzeptausdrücke
 - (a) jedes Konzeptsymbol C aus \mathbf{C} ist ein *SHOIQ*-Konzeptausdruck,
 - (b) \top und \perp sind *SHOIQ*-Konzeptausdrücke,
 - (c) sind C_1, \dots, C_k *SHOIQ*-Konzeptausdrücke, dann sind auch $C_1 \sqcap \dots \sqcap C_k, C_1 \sqcup \dots \sqcup C_k$ und $\neg C_1$ *SHOIQ*-Konzeptausdrücke,
 - (d) sind c_1, \dots, c_n in \mathbf{N} , dann ist $\{c_1, \dots, c_n\}$ ein *SHOIQ*-Konzeptausdruck,

- (e) ist C ein \mathcal{SHOIQ} -Konzeptausdruck, R ein \mathcal{SHOIQ} -Rollenausdruck, dann sind auch $\exists R.C$ und $\forall R.C$ \mathcal{SHOIQ} -Konzeptausdrücke.
- (f) ist C ein \mathcal{SHOIQ} -Konzeptausdruck, $R \in \mathbf{R}$ ein einfaches Rollensymbol, dann sind auch $\leq nR.C$ und $\geq nR.C$ \mathcal{SHOIQ} -Konzeptausdrücke.

2. die Menge der \mathcal{SHOIQ} -Rollenausdrücke

- (a) jedes $R \in \mathbf{R}$ ist ein \mathcal{SHOIQ} -Rollenausdruck,
- (b) für jedes $R \in \mathbf{R}$ ist R^- ein \mathcal{SHOIQ} -Rollenausdruck,

3. die Menge der \mathcal{SHOIQ} -Aussagen

- (a) für $c_1, c_2 \in \mathbf{N}$ sind $c_1 = c_2$ und $c_1 \neq c_2$ \mathcal{SHOIQ} -Aussagen,
- (b) sind $c_1, c_2 \in \mathbf{N}$ und ist R ein \mathcal{SHOIQ} -Rollenausdruck, dann ist $R(c_1, c_2)$ eine \mathcal{SHOIQ} -Aussage,
- (c) sind C_1, C_2 \mathcal{SHOIQ} -Konzeptausdrücke, dann ist $C_1 \sqsubseteq C_2$ eine \mathcal{SHOIQ} -Aussage,
- (d) für $R_1, R_2 \in \mathbf{R}$ ist $R_1 \sqsubseteq R_2$ eine \mathcal{SHOIQ} -Aussage,
- (e) für $R \in \mathbf{R}$ ist $\text{trans}(R)$ eine \mathcal{SHOIQ} -Aussage.
- (f) für $R \in \mathbf{R}$ sind $\text{func}(R)$ und $\text{invfunc}(R)$ \mathcal{SHOIQ} -Aussagen.

Dabei heißt ein Rollensymbol R *einfach*, wenn es nicht durch die Aussagen $\text{trans}(R)$ als transitiv erklärt wurde und das auch auch für keine Unterrolle $R' \sqsubseteq R$ der Fall ist.

Die Semantik von \mathcal{SHOIQ} geben wir wieder durch Übersetzung in die Prädikatenlogik, wobei in der folgenden Tabelle nur die neu hinzugekommenen Konstrukte aufgeführt sind.

C (\mathcal{SHOIQ} – Konzeptausdruck)	$C^0(x)$ (prädikatenlogische Formel)
$\{c_1, \dots, c_n\}$	$x = c_1 \vee \dots \vee x = c_n$
$\leq nR.C$	$\exists \leq n y (R(x, y) \wedge C^0(y/x))$
$\geq nR.C$	$\exists \geq n y (R(x, y) \wedge C^0(y/x))$
R (\mathcal{SHOIQ} – Rollenausdruck)	$R^0(x)$ (prädikatenlogische Formel)
$R \in \mathbf{R}$	$R(x, y)$
R^-	$R(y, x)$
A (\mathcal{ALC} – Aussage)	A^0 (prädikatenlogische Formel)
$C_1 \sqsubseteq C_2$	$\forall x (C_1^0(x) \rightarrow C_2^0(x))$
$R_1 \sqsubseteq R_2$	$\forall x \forall y (R_1^0(x, y) \rightarrow R_2^0(x, y))$
$\text{trans}(R)$	$\forall x \forall y \forall z (R(x, y) \wedge R(y, z) \rightarrow R(x, z))$
$\text{func}(R)$	$\forall x \forall y \forall z (R(x, y) \wedge R(x, z) \rightarrow y = z)$
$\text{invfunc}(R)$	$\forall x \forall y \forall z (R(y, x) \wedge R(z, x) \rightarrow y = z)$

Dabei wurde die folgenden Abkürzungen benutzt:

$$\begin{aligned} \exists^{\leq n} x F(x) &\leftrightarrow \exists x_1 \dots \exists x_n (\bigwedge_{1 \leq i < j \leq n} x_i \neq x_j \wedge \forall x (F(x) \rightarrow \bigvee_{1 \leq i \leq n} x = x_i)) \\ \exists^{\geq n} x F &\leftrightarrow \exists x_1 \dots \exists x_n (\bigwedge_{1 \leq i < j \leq n} x_i \neq x_j \wedge F(x_i)) \end{aligned}$$

Das bisher gesagte gibt zwar den logischen Kern des mit dem Schlagwort *semantic web* beschriebenen Gebiets wieder. Es bleibt aber eine Menge von Details ungesagt, z.B. eine genaue Beschreibung des aktuellen Zustands der Standardisierungen des *World Wide Web Consortiums* W3C, eine genauere Darstellung des Zusammenhang der hier skizzierten Logiken mit XML und RDF (Resource Description Framework), Algorithmen für Entscheidungsverfahren für *ALC* und *SHOIQ* und schließlich ein Überblick über die verfügbaren Implementierungen. Hinweise zu diesen Themen finden man z.B. in den beiden Lehrbüchern [AvH04] und [HKRS08].

Auf einen Aspekt allerdings soll hier doch noch kurz eingegangen werden. Woher das Vokabular einer prädikatenlogischen Sprache kommt ist für die Behandlung der logischen Eigenschaften absolut unerheblich. In der Praxis dagegen und für Unterstützungswerkzeuge ist das ein sehr entscheidender Punkt. In Kapitel 6 wird die Spezifikationsprache OCL eingeführt. Das Vokabular, aus dem OCL Ausdrücke aufgebaut werden, stammt aus dem beteiligten UML Diagramm und der OCL Bibliothek. Für die semantischen Technologien ist die Festlegung eines Vokabular eine der großen Herausforderungen, schließlich sollten die Annotationen im gesamten *World Wide Web* gelesen und vielleicht auch verstanden werden können. Die Beispiele von OWL Ausdrücken, die wir bisher gezeigt haben sind in *abstrakter Syntax* geschrieben. Für den elektronischen Austausch wird RDF Syntax benutzt, die wiederum von der XML Syntax abgeleitet ist. Daher beginnt ein Dokument, das eine OLW Ontologie enthält mit einer Deklaration der benutzen *Namensräume* (engl. name spaces), wie in XML durch die Angabe von URI *Uniform Resource Identifier*. Die Grundidee dahinter ist, Vokabulare im Netz zu veröffentlichen, vielleicht sogar mit einigen Erklärungen zur intendierten Bedeutung, und darauf zu hoffen, daß eine einheitliche Benutzung über einen möglichst großen Personenkreis sich einstellt.

Wir betrachten hier das beliebte Beispielvokabular *FOAF* (Friend of a Friend). Das FOAF Vokabular stellt Ausdrucksmittel bereit, um Dinge zu beschreiben die Menschen auf ihre *home pages* stellen. Genauere Informationen finden man unter <http://xmlns.com/foaf/spec/>. In abstrakter Notation betrachten wir die Aussagen

$(Person \sqcap \exists name.\{\text{Dan Brickley}\} \sqcap \exists homepage.\{hp1\})(p1)$

und die Klassendefinition

$ProfileDokument = \top \sqcap \exists maker.\{p1\} \sqcap \exists primaryTopic.\{p1\}$

Dabei sind *Person*, *ProfileDokument* Konzeptsymbole, *name*, *homepage* *maker* *primaryTopic* Rollenbezeichner und *p1*, *hp1* und *Dan Brickley* Namen. Hier sind wir etwas großzügig. Genauer ist *Dan Brickley* ein Literal im Datentyp Zeichenkette und *hp1* eine URI. Wie die beiden Ausdrücke in RDF Notation aussehen ist in Abbildung 5.5 zu sehen.

```
1 <rdf:RDF
2   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3   xmlns:foaf="http://xmlns.com/foaf/0.1/"
4   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
5
6 <foaf:Person rdf:nodeID="p1">
7   <foaf:name>Dan Brickley</foaf:name>
8   <foaf:homepage
9     rdf:resource="http://rdfweb.org/people/danbri"/>
10 </foaf:Person>
11
12 <foaf:PersonalProfileDocument rdf:about="">
13   <foaf:maker rdf:nodeID="p1"/>
14   <foaf:primaryTopic rdf:nodeID="p1"/>
15 </foaf:PersonalProfileDocument>
16
17 </rdf:RDF>
```

Abbildung 5.5: RDF Dokument zum FOAF Vokabular

In Zeile 3 wird das eben genannte Vokabular importiert. Die anderen vorkommenden Rollen, *nodeId* und *resource about* stammen aus den in den in Zeile 2 und 4 deklarierten Namensräumen.

Es bleibt abzuwarten, ob oder mit welchen Änderungen dieses Vorgehen erfolgreich sein wird. Hier ein kritisches Zitat aus [DJK04]

Allerdings zeigt die Erfahrung, daß es organisatorisch und politisch nahezu aussichtslos ist eine Übereinkunft hinsichtlich eines global gültigen Vokabulars zu etablieren, auf das alle Kommunikationspartner zurückgreifen. Um die Anforderung der losen Kopplung aus der Service-orientierten Architektur (SOA) nicht aufgeben zu müssen, wird daher ein Konzept benötigt, mit dem es möglich ist, eigenständig lokal definierte Vokabulare gegeneinander abgleichen zu können.

5.4.5 Übungsaufgaben

Übungsaufgabe 5.4.1

In dem Lehrbuch [TZ71, Seite 20] sind, unter vielen anderen, die folgenden Gleichungen aus der elementaren Mengenlehre als Übungsaufgaben aufgeführt

- 1 $A \cap (B \setminus C) = (A \cap B) \setminus (A \cap C)$
- 2 $A \cap (B \setminus C) = (A \cap B) \setminus C$
- 3 $A \cup (B \setminus C) = (A \cup B) \setminus (C \setminus A)$
- 4 $A \cup (B \setminus C) = (A \cup B) \setminus ((A \cap B) \setminus C)$

Hierbei steht $A \setminus B$ für die mengentheoretische Differenz von A und B , d.h. $A \setminus B = \{a \in A \mid a \notin B\}$.

Formalisieren Sie die vier mengentheoretische Gleichungen als Formeln der Logik erster Stufe und beweisen Sie deren Gültigkeit mit einem automatischen Theorembeweiser, z.B. mit dem KeY-Beweiser.

Achtung! eine der Gleichungen ist nicht korrekt.

Übungsaufgabe 5.4.2

Die folgenden Informationen stammen aus [GA02].

Die Boolesche Algebra ist die Theorie der Mengen. Mengen kann man als einstellige Relationen auffassen. Gibt es auch eine Theorie der zweistelligen, der binären Relationen? Der erste der dazu einen Beitrag veröffentlichte war de Morgan [dM64]. Einen ersten Höhepunkte erreichte die Theorie durch das 649 Seiten starke Buch von E.Schröder [Sch95]. Alfred Tarski stellte dann in [Tar41] den Versuch einer Axiomatisierung vor. Wir stellen die Axiome in der Form vor, wie sie in [CT51] veröffentlicht wurden. Dazu betrachten wir eine beliebige Menge M und definieren für binäre Relationen a, b auf M die folgenden Operationen

- $(a \vee b)(x, y) \Leftrightarrow a(x, y) \text{ oder } b(x, y)$
- $(a \wedge b)(x, y) \Leftrightarrow a(x, y) \text{ und } b(x, y)$
- $(\neg a)(x, y) \Leftrightarrow \text{nicht } a(x, y)$
- $(a; b)(x, y) \Leftrightarrow \text{es gibt } z \text{ mit } a(x, z) \text{ und } b(z, y)$
- $a^{-1}(x, y) \Leftrightarrow a(y, x)$
- $id(x, y) \Leftrightarrow x = y$
- für alle x, y gilt $1(x, y)$
- für alle x, y gilt $\neg 0(x, y)$

Tarskis Axiome sind

1. $((M \times M), \vee, \wedge, \neg, 0, 1)$ ist eine Boolesche Algebra, d.h. es gelten die folgenden Axiome

$a \vee (b \vee c)$	$= (a \vee b) \vee c$	<i>associativity</i>
$a \wedge (b \wedge c)$	$= (a \wedge b) \wedge c$	<i>associativity</i>
$a \vee b$	$= b \vee a$	<i>commutativity</i>
$a \wedge b$	$= b \wedge a$	<i>commutativity</i>
$a \vee (a \wedge b)$	$= a$	<i>absorption</i>
$a \wedge (a \vee b)$	$= a$	<i>absorption</i>
$a \vee (b \wedge c)$	$= (a \vee b) \wedge (a \vee c)$	<i>distributivity</i>
$a \wedge (b \vee c)$	$= (a \wedge b) \vee (a \wedge c)$	<i>distributivity</i>
$a \vee \neg a$	$= 1$	<i>complements</i>
$a \wedge \neg a$	$= 0$	<i>complements</i>

2. $(a; (b; c)) = (a; b); c$ (associativity of ;)
3. $a; (b \vee c) = (a; b) \vee (a; c)$ (distributivity of ; over \vee)
 $(a \vee b)^{-1} = a^{-1} \vee b^{-1}$ (distributivity of $^{-1}$ over \vee)
4. $a; id = a$ (identity law)
5. $(a^{-1})^{-1} = a$ (first involution law)
6. $(a; b)^{-1} = b^{-1}; a^{-1}$ (second involution law)
7. $(a^{-1}; \neg(a; b)) \subseteq \neg b$ (the cyclic axiom)

Man beachte, daß auch 7 sich, wie alle anderen Axiome, als Gleichung schreiben lässt: $(a^{-1}; \neg(a; b)) \vee \neg b = \neg b$

Formulieren Sie die Definitionen der Operatoren \vee , \wedge , \neg , $+$, $;$ und $^{-1}$ und die Gleichungen in einer Logik erster Stufe und beweisen Sie, daß die Gleichungen aus den Operationsdefinitionen folgen.

Übungsaufgabe 5.4.3

Beweisen Sie mit Hilfe eines automatischen Theorembeweislers die folgenden Aussagen.

Notation und Definitionen wie in Aufgabe 5.4.2 und außerdem

$$a + b = (a \wedge \neg b) \vee (b \wedge \neg a)$$

1. $a + (b + c) = (a + b) + c$
2. $a \vee b = a + b + (a \wedge b)$
3. $a + a = 0$
4. $(a \vee b) + (a \wedge b) = a + b$

5. $1 + a = \neg a$

Übungsaufgabe 5.4.4

Das ist eine etwas kompliziertere Aufgabe, die wieder mit der Notation und den Definitionen aus Aufgabe 5.4.2 arbeitet. Beweisen Sie, daß eine Relation a genau dann funktional ist, wenn $a^{-1}; a \subseteq id$ gilt

Zur Definition einer funktionalen Relation siehe Def. 1.1 auf Seite 1.

Übungsaufgabe 5.4.5

Die folgende Definition der ganzzahligen Division ist der Java Language Description entnommen. Geändert wurden die Teile, die sich auf die Endlichkeit der ganzen Zahlen in Java beziehen, so daß die folgende Definition für die üblichen mathematischen ganzen Zahlen \mathbb{Z} zu lesen ist. Außerdem wird vereinbart, daß die Relation $jdiv(n, 0) = q$ für beliebige Zahlen $n, q \in \mathbb{Z}$ wahr ist. Diese Art mit der Division durch 0 umzugehen ist unter dem Namen *underspecification* bekannt. Dabei wird angenommen, daß $div(n, 0)$ einen Wert in \mathbb{Z} hat über den aber nichts bekannt ist.

Integer division rounds toward 0. That is, the quotient produced for integer operands n and d is an integer value q whose magnitude is as large as possible while satisfying $|d \times q| \leq |n|$; moreover, q is positive when $|n| \geq |d|$ and n and d have the same sign, but q is negative when $|n| \geq |d|$ and n and d have opposite signs.

1. Finden Sie eine Formel ϕ der Prädikatenlogik erster Stufe mit freien Variablen n, d, q so daß $\phi \leftrightarrow q = jdiv(n, d)$ in den ganzen Zahlen gilt.
2. Zeigen Sie:
 $\forall x, y (y * jdiv(x, y) = x \vee y * jdiv(x, y) = x + 1)$
3. $\forall x, y (y * jdiv(x, y) = x \leftrightarrow \exists z (x = 2 * z))$ (x ist gerade)
4. $\forall x, y (y * jdiv(x, y) = x + 1 \leftrightarrow \exists z (x = 2 * z + 1))$ (x ist ungerade)

Übungsaufgabe 5.4.6

Es gibt Alternativen zu der ganzzahligen Division, wie sie in der vorangegangenen Aufgabe 5.4.5 betrachtet wurde. In Python wird *Runden nach $-\infty$* benutzt, d.h. $div(n, d)$ wird berechnet, in dem zu nächst die n/d als Dezimalzahl berechnet wird und dann nach $-\infty$ gerundet wird. Aus $5/2 = 2,5$ wird $div(5, 2) = 2$ und aus $-5/2 = -2,5$ wird $div(-5, 2) = -3$.

1. Geben Sie eine Formel ϕ der Prädikatenlogik, so daß $\forall n \forall d \forall q (div(n, d) = q \leftrightarrow \phi)$ gilt.
2. Drücken Sie $jdiv$ aus Aufgabe 5.4.5 durch div aus.

5.5 Reduktionssysteme

Um die Gültigkeit einer Gleichung in einer Gleichungstheorie algorithmisch behandeln zu können, müssen wir die Gleichheit in ihrer *gerichteten* Version benutzen, E wird ein *Termersetzungssystem*.

Die den Termersetzungssystemen zugrunde liegende Idee einer eindeutigen Normalform und die schrittweise Normalisierung eines symbolischen Ausdrucks ist so elementar, daß sie in vielen Zusammenhängen in unterschiedlichen Ausprägungen eine Rolle spielt. Als übergreifenden Begriff für alle diese Ausprägungen wollen wir allgemeiner die *Reduktionssysteme* einführen. Wir werden uns dann auf spezielle Reduktionssysteme konzentrieren, eben auf die Termersetzungssysteme. In diesem Fall sind die betrachteten symbolischen Ausdrücke Terme der Prädikatenlogik und die Umformung geschieht durch Ersetzung von Teiltermen. Aber zunächst stellen wir einige Resultate vor, die sich schon auf der abstrakten Ebene allgemeiner Reduktionssysteme beweisen lassen.

Einführung

Definition 5.57

(Reduktionssysteme) Ein **Reduktionssystem** (D, \succ) besteht aus einer nicht-leeren Menge D und einer beliebigen, binären Relation \succ auf D .

Definition 5.58

(Notation) Ist (D, \succ) ein Reduktionssystem, dann benutzen wir die folgenden Bezeichnungen:

- \rightarrow die reflexive, transitive Hülle von \succ
- $\xrightarrow{+}$ die transitive Hülle von \succ
- \leftrightarrow die reflexive, transitive, symmetrische Hülle von \succ

Beliebige Reduktionssysteme sind viel zu allgemein, als daß ihr Studium irgend etwas Interessantes hervorbringen könnte. Es sind die folgenden Definitionen, die Anlaß zu einer nicht trivialen Theorie geben,

Definition 5.59

1. Ein Reduktionssystem (D, \succ) heißt **konfluent**, wenn für jedes Tripel $s, s_1, s_2 \in D$ mit $s \rightarrow s_1, s \rightarrow s_2$ ein $t \in D$ existiert mit $s_1 \rightarrow t$ und $s_2 \rightarrow t$.
2. (D, \succ) heißt **lokal konfluent**, wenn für alle $s, s_1, s_2 \in D$ mit $s \succ s_1, s \succ s_2$ ein $t \in D$ mit $s_1 \rightarrow t$ und $s_2 \rightarrow t$ existiert.

3. (D, \succ) heißt **noethersch** (oder **wohlfundiert** oder **terminierend**), wenn es keine unendlichen Folge $s_0 \succ s_1 \dots \succ s_i \succ \dots$ gibt.
4. Ein konfluentes und noethersches Reduktionssystem heißt **kanonisch**.
5. Ein Element $s \in D$ heißt **irreduzibel** (oder eine **Normalform**) in (D, \succ) , wenn kein $t \in D$ existiert mit $s \succ t$.
6. Sei $s \in D$. Ein Element $s_0 \in D$ heißt eine **Normalform für s** in (D, \succ) , wenn s_0 irreduzibel ist und $s \rightarrow s_0$ gilt.

Satz 5.60

Sei (D, \succ) ein kanonisches Reduktionssystem. Dann gilt:

1. Zu jedem $s \in D$ gibt es eine eindeutige Normalform. Diese bezeichnen wir mit $irr(s)$.
2. Für $s, t \in D$ gilt

$$s \leftrightarrow t \text{ gdw } irr(s) = irr(t)$$
3. (D, \succ) sei berechenbar im folgenden Sinne: Es gibt einen Algorithmus, der zu jedem $t \in D$ ein t' mit $t \succ t'$ liefert, wenn ein solches existiert, und andernfalls ausgibt „ t ist irreduzibel“, Dann ist die Relation \leftrightarrow entscheidbar.

Beweis:

Zu 1.: Zunächst erledigen wir die Eindeutigkeitsfrage. Angenommen es gäbe für $s \in D$ zwei Normalformen s_1, s_2 . Das heißt es gilt $s \rightarrow s_1$ und $s \rightarrow s_2$. Wegen der Konfluenz von (D, \succ) gibt es $t \in D$ mit $s_1 \rightarrow t$ und $s_2 \rightarrow t$. Das widerspricht der Irreduzibilität von s_1, s_2 . Um uns von der Existenz einer Normalform zu überzeugen, betrachten wir $s \in D$, setzen $s_0 = s$ und wählen ein s_{i+1} mit $s_i \succ s_{i+1}$, solange s_i nicht irreduzibel ist. Da (D, \succ) noethersch ist, wird nach endlich vielen Schritten ein irreduzibles s_i erreicht.

Zu 2.: Die Implikation von rechts nach links ist trivial. Gelte jetzt $s \leftrightarrow t$. Nach Definition des transitiven, symmetrischen Abschlusses gibt es eine Folge $s = s_0, s_1, \dots, s_n = t$, so daß für alle $0 \leq i < n$ entweder $s_i \succ s_{i+1}$ oder $s_{i+1} \succ s_i$ gilt. Der Nachweis von $irr(s) = irr(t)$ geschieht durch Induktion über n . Der Induktionsanfang $n = 0$, d.h. $s = t$ ist trivial. Sei also die Behauptung für Folgen der Länge $n - 1$ schon bewiesen. Also gilt $irr(s_1) = irr(t)$. Im Fall $s_0 \succ s_1$ gilt offensichtlich $irr(s_0) = irr(s_1)$, und wir sind fertig. Falls $s_1 \succ s_0$ gilt, folgt aus der Konfluenz, daß ebenfalls $irr(s_0) = irr(s_1)$ gelten muß.

Zu 3.: Zu gegebenem s, t wird wie folgt entschieden, ob $s \leftrightarrow t$. Beginnend mit $s, s_0 := s$, liefert der vorausgesetzte Algorithmus Elemente s_i mit $s_0 \succ s_1 \succ s_2 \succ \dots$, bis hierbei ein irreduzibles s_m erreicht ist. Da (D, \succ) noethersch ist, tritt das auf jeden Fall ein und wird durch „ s_m ist irreduzibel“ mitgeteilt, ferner gilt $s_m = irr(s)$. Entsprechend erhält man $irr(t)$ aus t . Nach (2) ist $s \leftrightarrow t$ genau dann, wenn $irr(s) = irr(t)$. ■

Eine wesentliche Stütze für die Theorie der Reduktionssysteme ist das nachfolgende Lemma, welches sagt, daß für noethersche Systeme aus der lokalen Konfluenz auch schon die Konfluenz folgt. Wir besprechen zunächst das in seinem Beweis verwendete Beweisprinzip der „noetherschen Induktion“.

Lemma 5.61

(Noethersche Induktion) Für ein noethersches Reduktionssystem (D, \succ) gilt das folgende Beweisprinzip der *Noetherschen Induktion*:

Es sei $X \subseteq D$, so daß für alle $a \in D$ gilt

$$\{b \mid a \succ b\} \subseteq X \Rightarrow a \in X.$$

Dann ist $X = D$.

Beweis:

Angenommen $X \neq D$. Sei $a_0 \in D \setminus X$. Nach Annahme über X gilt $\{b \mid a_0 \succ b\} \not\subseteq X$.

Es gibt also ein a_1 mit

$$a_0 \succ a_1, a_1 \notin X.$$

Nach Annahme über X gilt wieder $\{b \mid a_1 \succ b\} \not\subseteq X$ und es gibt ein a_2 mit

$$a_1 \succ a_2, a_2 \notin X.$$

Fährt man in dieser Weise fort, so erhält man eine unendliche Folge $(a_i)_{i \in \mathbb{N}}$ mit $a_i \succ a_{i+1}$ für alle i . Das ist ein Widerspruch, denn (D, \succ) war als noethersch vorausgesetzt. ■

Hiermit zeigen wir das

Lemma 5.62

Wenn (D, \succ) ein noethersches und lokal konfluentes Reduktionssystem ist, dann ist (D, \succ) konfluent, d. h. kanonisch.

Beweis:

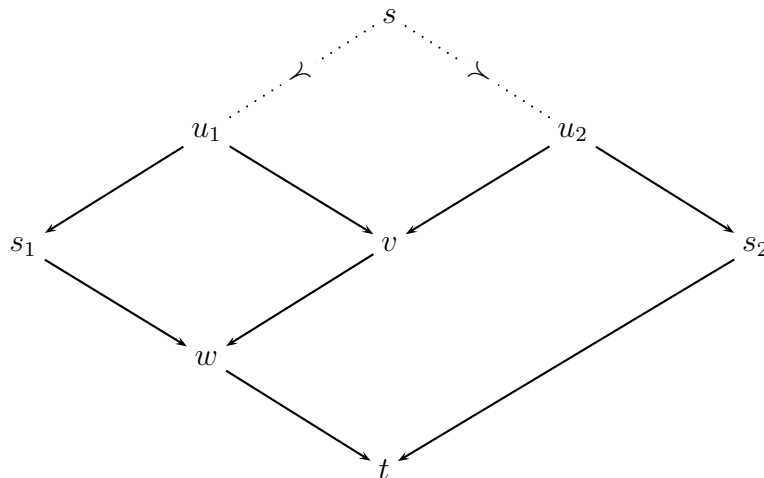
Wir verwenden noethersche Induktion bezüglich der Menge $\text{Confl} :=$

$$\{s \mid \text{Für alle } s_1, s_2 \text{ mit } s \rightarrow s_1, s \rightarrow s_2 \text{ existiert ein } t \text{ mit } s_1 \rightarrow t, s_2 \rightarrow t\}$$

Dazu müssen wir also zeigen, daß für alle s gilt:

$$\{s' \mid s \succ s'\} \subseteq \text{Confl} \Rightarrow s \in \text{Confl}$$

Es seien s, s_1, s_2 gegeben mit $s \rightarrow s_1, s \rightarrow s_2$. Im Falle $s = s_1$ oder $s = s_2$ ist man fertig. (Etwa: $s_1 = s \rightarrow s_2$). Sei also $s \neq s_1, s \neq s_2$. Also existieren u_1, u_2 mit $s \succ u_1 \rightarrow s_1$ und $s \succ u_2 \rightarrow s_2$. Wegen der lokalen Konfluenz von (D, \succ) existiert ein v mit $u_1 \rightarrow v, u_2 \rightarrow v$. Nach Voraussetzung („Induktionsannahme“) liegt u_1 in Confl . Also gibt es ein w mit $s_1 \rightarrow w$ und $v \rightarrow w$. Entsprechend schließen wir aus der Induktionsannahme $u_2 \in \text{Confl}$, daß ein Term t existiert mit $s_2 \rightarrow t$ und $w \rightarrow t$. Wir haben $s_1 \rightarrow t$ und $s_2 \rightarrow t$ und somit $s \in \text{Confl}$, was zu beweisen war. ■



Von lokaler zu uneingeschränkter Konfluenz

Beispiele

Polynomreduktion

Definition 5.63

- Ein **Potenzprodukt** in den Unbestimmten X_1, \dots, X_n über einem Körper K ist ein Ausdruck der Form

$$X_1^{e_1} * \dots * X_n^{e_n}$$

mit natürlichen Zahlen e_j ,

- Ein **Monom** in den Unbestimmten X_1, \dots, X_n über K ist ein Ausdruck der Form

$$c * pp,$$

wobei $c \neq 0$ ein Element aus k ist und pp ein Potenzprodukt.

- Ein **Polynom** in den Unbestimmten X_1, \dots, X_n über K ist ein Ausdruck der Form

$$m_1 + \dots + m_k$$

mit Monomen m_i . Die Menge aller Polynome über K bildet mit den naheliegenden Produkt- und Summendefinition den Polynomring $K[X_1, \dots, X_n]$.

Eine Ordnungsrelation \prec auf der Menge aller Monome heißt **zulässig**, wenn für beliebige Monome m, m_1, m_2 gilt:

1. aus $m_1 \prec m_2$ folgt $m_1 * m \prec m_2 * m$ und
2. $1 \preceq m$

Die lexikographische Ordnung der Monome ist ein typisches Beispiel einer zulässigen Ordnungsrelation.

Beispiel 5.64 (Polynomreduktion)

Sei $B \subseteq K[X_1, \dots, X_n]$. Die Reduktionsrelation \succ_B auf $K[X_1, \dots, X_n]$, die Polynomreduktion für B , wird wie folgt definiert.

$f \succ_B g$ gilt genau dann, wenn

- das größte Monom in f ist $m = c_1 * pp_1$ für $c \in K$ und ein Potenzprodukt pp_1 und
- es gibt ein Polynom $h \in B$ mit größtem Monom $u = c_2 * pp_2$ mit $pp_1 = v * pp_2$ und
- $g = f - c_1 * c_2^{-1} * v * h$

Ein konkretes Beispiel für die Polynomreduktion

Sei $B = \{h_1 = xy^2 - x, h_2 = x - y^3\}$ $f = x^7y^2 + x^3y^2 - y + 1$.

Für $h = h_1$ haben wir in der Notation des Beispiels $c_1 = c_2 = 1$, $pp_1 = x^7y^2$, $pp_2 = xy^2$ und $v = x^6$.

Für $g = f - c_1 * c_2^{-1} * v * h = x^7 + x^3y^2 - y + 1$ gilt dann

$$f \succ_B g$$

Das Reduktionssystem $(K[X_1, \dots, X_n], \succ_B)$ ist stets noethersch. Es muß nicht immer konfluent sein. Aber für jede Menge B gibt es eine Menge G , die dasselbe Ideal in dem Ring $K[X_1, \dots, X_n]$ erzeugt wie B , so daß \succ_G konfluent ist. G läßt sich aus B berechnen, z.B. mit dem Buchbergerschen Algorithmus.

Eine leicht zu lesende Einführung in dieses Gebiet findet man in [Buc85].

β -Reduktion

Beispiel 5.65

(β -Reduktion im λ -Kalkül)

Für zwei λ -Terme M, N ist die β -Reduktion \succ_β definiert durch: $M \succ_\beta N$ genau dann, wenn

- ein Teiltermvorkommen $(\lambda x M_1)N_1$ in M gibt und
- n entsteht aus M , indem $(\lambda x M_1)N_1$ ersetzt wird $M_1[x \leftarrow N_1]$,
- wobei $M_1[x \leftarrow N_1]$ aus M_1 entsteht, indem jedes freie Vorkommen von x ersetzt wird durch N_1 .

Die β -Reduktion auf der Menge aller λ -Terme ist konfluent, siehe z. B. [Bar84], Section 3.2. Die β -Reduktion ist nicht noethersch, so hat z.B. der Term $(\lambda x(xx))\lambda x(xx)$ keine Normalform.

Wortersetzung

Beispiel 5.66

(Semi-Thue-Systeme) Sei R eine Menge von Paaren (r, s) von Wörtern über einem Alphabet Σ , d.h. $r, s \in \Sigma^*$. Die Relation \succ_R auf der Menge Σ^* aller Wörter über Σ ist definiert durch:

$u \succ_R v$ gdw es gibt $(r, s) \in R$ und $z, y \in \Sigma^*$, so daß $u = xry$ und $v = xsy$.
 (S, R) heißt ein **Semi-Thue-System** (string rewriting system).

Im Unterschied zu den jetzt zu besprechenden Termersetzungssystemen treten in Semi-Thue-Systemen keine Variablen auf und damit keine Substitutionen. Außerdem ist die interne Struktur von Wörtern wesentlich ärmer als die interne Struktur von Termen. Einen umfassenden Überblick über Semi-Thue-Systeme findet man in [Boo87].

5.6 Termersetzungssysteme

Wir kehren zurück zu unserem Anliegen, in einer Gleichungstheorie E die Gültigkeit einer Gleichung festzustellen und lesen, wie schon zu Beginn von Abschnitt 5.5 bemerkt, zu diesem Zweck die Gleichheit „gerichtet“.

Definition 5.67

1. Zu einer Menge E von Gleichungen über einer Signatur Σ definieren wir für Terme t, s :

$$s \xrightarrow{1}_E t$$

genau dann, wenn es eine Gleichung $l \doteq r \in E$ gibt und eine Substitution σ , so daß gilt:

- $\sigma(l)$ ist Unterterm von s
- t entsteht aus s , indem dort der Unterterm $\sigma(l)$ an genau einer Stelle ersetzt wird durch $\sigma(r)$.

Man beachte, daß nur die linke Seite durch die rechte ersetzt wird

2. \rightarrow_E ist die reflexive, transitive Hülle von $\xrightarrow{1}_E$
3. \leftrightarrow_E ist die reflexive, transitive, symmetrische Hülle von $\xrightarrow{1}_E$

Wenn das zugehörige Gleichungssystem klar ist, schreiben wir einfach $\xrightarrow{1}$, \rightarrow , \leftrightarrow .

Die allgemeinen Betrachtungen in 5.5 werden jetzt dazu spezialisiert, daß für eine vorgegebene endliche Menge E von Gleichungen

\succ die Relation $\xrightarrow{1}_E$ ist,

wie sie in Definition 5.67 eingeführt wurde.

Definition 5.68

(Termersetzungssysteme)

Ist E eine endliche Menge von Gleichungen über der Signatur Σ , dann nennen wir das Reduktionssystem $(\text{Term}_\Sigma, \xrightarrow{1}_E)$ ein *Termersetzungssystem*. Da dieses durch Σ und E eindeutig bestimmt ist, sprechen wir kürzer vom *Termersetzungssystem* (Σ, E) .

Beispiel 5.69

Ein einfaches kanonisches Termersetzungssystem E_{GBT}

$$\begin{array}{ll} 0 \wedge x = 0 & 1 \wedge x = x \\ x \wedge 0 = 0 & x \wedge 1 = x \\ 0 \vee x = x & 1 \vee x = 1 \\ x \vee 0 = x & x \vee 1 = 1 \end{array}$$

Wir schreiben kurz E anstelle von E_{GBT} .

Für jeden variablenfreien Booleschen Term t gilt $t \rightarrow_E 0$ oder $t \rightarrow_E 1$.

Als speziellen Fall von Satz 5.60 haben wir jetzt den

Satz 5.70

(Σ, E) sei ein kanonisches Termersetzungssystem.

1. Zu jedem Term t gibt es genau einen irreduziblen Term $\text{irr}(t)$ mit $t \rightarrow_E \text{irr}(t)$.
2. Für beliebige Terme s, t gilt:

$$E \models s \doteq t \Leftrightarrow \text{irr}(s) = \text{irr}(t).$$

3. Die Gültigkeit einer Gleichung in der Theorie von E ist entscheidbar.

Wenn unser vorgelegtes System (Σ, E) kanonisch ist, dann ist unser Problem also gelöst: Für eine gegebene Gleichung $s \doteq t$ läßt sich entscheiden, ob $E \models s \doteq t$. Leider ist das i.a. nicht der Fall (wir wissen ja, daß es unentscheidbare Gleichungstheorien gibt). Auch können wir i.a. einem Termersetzungssystem nicht ansehen, ob es noethersch ist bzw. ob es konfluent ist, beides ist i.a. unentscheidbar.

Kapitel 6

Die Objektbeschreibungssprache OCL

In diesem Abschnitt wollen wir die Objektbeschreibungssprache, (engl. Object Constraint Language, OCL) vorstellen.

6.1 Historie und Motivation

Im Jahre 1996 rief die von der Object Management Group (OMG) eingesetzte Arbeitsgruppe (task force, wie so etwas im Amerikanischen genannt wird) für objekt-orientierte Analyse und Entwurf zur Einreichung von Vorschlägen auf. IBM und ObjecTime Limited reichten einen gemeinsamen Entwurf ein, dessen wichtigster Aspekt die Aufnahme der Objektbeschreibungssprache, OCL, in den UML Standard war. Dieser Vorschlag wurde angenommen und in UML 1.1 integriert. OCL wurde von Jos Warmer entwickelt als eine Sprache zur Modellierung von Geschäftsprozessen in der IBM und baut auf der Syntropy Methode von Steve Cook und John Daniels auf.

Die Objektbeschreibungssprache, OCL, ist jetzt Teil des UML Standards und kann benutzt werden zur Spezifikation jeder Art von Einschränkungen, z.B. Vor- und Nachbedingungen, Invarianten, Filter für die Objekte in den verschiedenen UML- Diagrammen.

OCL ist eine reine Beschreibungssprache (engl. expression language) im Gegensatz zu einer Programmiersprache. Die Ausdrücke von OCL, in welchem Teil einer UML Spezifikation sie auch auftreten, haben garantiert keine Seiteneffekte. Auch wenn eine OCL Formel benutzt wird um den Zustand nach der Ausführung einer Methode zu beschreiben, also als *post condition* benutzt wird, hat sie selbst keine Auswirkung auf den Zustand. OCL dient als Modellierungssprache. Mit OCL kann man keine Programme schreiben oder den Kontrollfluß verändern. Was in OCL aufgeschrieben wird muß nicht einmal ausführbar sein. Wichtig ist auch die Atomizität von OCL: während der Auswertung einer OCL Formeln können sich die darin vorkommenden Objekte nicht ändern. Schließlich ist OCL gedacht als eine formale Sprache. Alle darin vorkommenden Konstrukte haben eine formal definierte Bedeutung.

Die Motivation für die Einführung einer Objektbeschreibungssprache, ist die Tatsache, daß in der objekt-orientierten Modellierung die graphischen Diagramme, wie z.B. die Klassendiagramme, nicht präzise genug sind für eine unzweideutige Spezifikation. Es ist erforderlich zusätzliche Einschränkungen (constraints) von den Objekte eines Modells zu fordern. Solche Einschränkungen wurden häufig in natürlicher Sprache angegeben. Aber die Erfahrung hat gezeigt, daß sich dabei Unklarheiten nicht vermeiden lassen.

Die Entwickler von OCL sind angetreten mit dem Anspruch eine formale

Sprache zu entwickeln, die auch von Personen ohne mathematische Ausbildung verstanden und benutzt werden kann. Darin sehen sie den Unterschied zu anderen formalen Spezifikationsprachen, wie z.B. Z oder VDM.

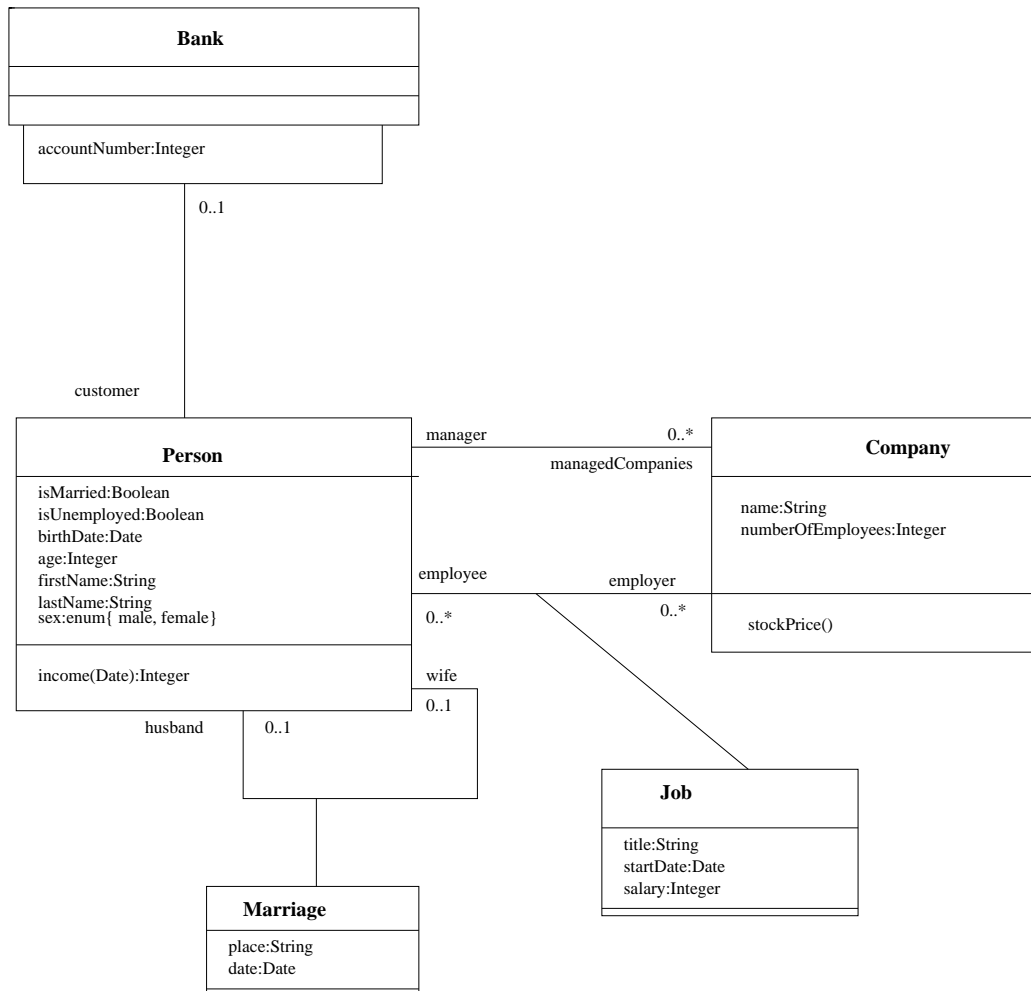


Abbildung 6.1: Beispiel 1 eines Klassendiagramms

6.2 Einführung in OCL

Die erste wichtige Information zum Erlernen der Objektbeschreibungssprache OCL ist ihre Koppelung an ein UML Diagramm. Eine OCL Formel macht überhaupt nur Sinn, wenn ein UML Diagramm vorliegt, auf das sie sich beziehen kann. In Abbildung 6.1 ist das UML Klassendiagramm aus [OMG97a]

wiedergegeben. In Abbildung 6.4 finden Sie das Referenzbeispiel, das in dem Buch [WK99] durchgängig verwendet wird.

Innerhalb eines UML Diagramms können OCL Ausdrücke überall auftreten, wo Einschränkungen auftreten können. Im Englischen heißt dieser Teil der UML-Sprache *constraint*, was wir mit *Einschränkungen* übersetzen wollen. Einschränkungen sind ein sehr flexibel einsetzbares Sprachelement. Sie können an jedes Modellelement von UML oder auch an mehrere gemeinsam angeheftet werden. Die syntaktische Form von Einschränkungen ist völlig frei, abgesehen davon, daß sie in geschweifte Klammern eingeschlossen sein sollen. Sie können in natürlicher Sprache aufgeschrieben werden oder in mehr oder weniger formalen Notationen. Die allgemeiner Erklärung, wie sie z.B. in [WK99] gegeben wird, liest sich so

A constraint is a restriction on one or more values of (part of) an object-oriented model or system

Der UML Standard bietet als eine Möglichkeit einer formalen Sprache zur Formulierung von Einschränkungen die Objektbeschreibungssprache OCL an, die wir hier vorstellen wollen. Beispiele für Einschränkungen sind in der Abbildungen 6.2, in natürlicher Sprache, und 6.3 in formaler Notation, zu sehen.

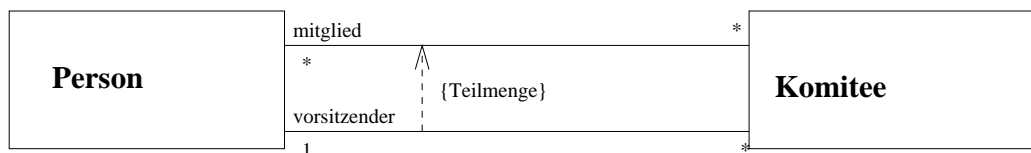


Abbildung 6.2: Eine Einschränkung in natürlicher Sprache

Die Einschränkung in Abb. 6.3 ist in ein Kommentarkästchen geschrieben worden, das an zwei Relationen angehängt ist. In OCL muß man sich für eine Klasse entscheiden, an der eine Einschränkung angreift. Diese Klasse heißt der *Kontext* des OCL Ausdrucks. Wir folgen der in [WK99] eingeführten Praxis und stellen den Kontext, den unterstrichenen Klassennamen, dem OCL Ausdruck voran. Die Einschränkung aus Abbildung 6.3 notieren wir dann als

OCL Beispiel 1

Person

arbeitgeber = vorgesetzter.arbeitgeber **or** vorgesetzter→isEmpty

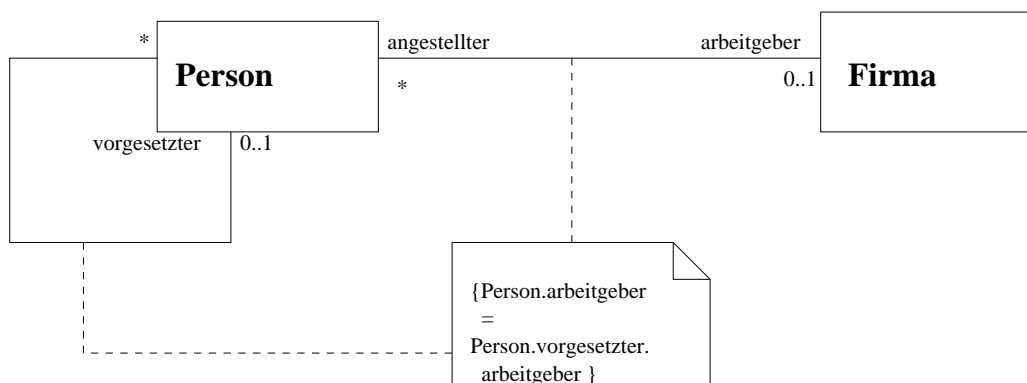


Abbildung 6.3: Eine formal notierte Einschränkung

Diese Einschränkung wird wie folgt interpretiert: für jede Instanz p der Kontextklasse, hier also der Klasse *Person*, betrachtet man die Menge $M_1(p)$, der mit p assoziierten Firmen. $M_1(p)$ kann hier die leere Menge oder eine Einermenge sein. Um die rechte Seite der Gleichung zu berechnen, betrachtet man zuerst die Menge $M_0(p)$ derjenigen Personen, die Vorgesetzte von p sind. Wieder kann $M_0(p)$ die leere Menge oder eine Einermenge sein. In einem zweiten Schritt betrachtet man die Menge $M_2(p)$ aller Firmen, die Arbeitgeber einer Person aus $M_0(p)$ sind. Im vorliegenden Fall kann $M_2(p)$ wieder einer Einermenge oder die leere Menge sein. Die Einschränkung `arbeitgeber = vorgesetzter.arbeitgeber` ist wahr, wenn $M_1(p) = M_2(p)$ für jedes p aus der Klasse gilt.

Innerhalb eines OCL Ausdrucks kann auf die Kontextklasse mit dem Namen *self* bezug genommen werden. Die obige Einschränkung ist gleichbedeutend mit

OCL Beispiel 2

Person

`self.arbeitgeber = self.vorgesetzter.arbeitgeber or self.vorgesetzter→isEmpty`

Für ein weiteres Beispiel wählen wir die Assoziationsklasse (association class) *membership* aus Abbildung 6.4 als Kontext.

OCL Beispiel 3

membership

`card.owner = customer`

Die Interpretation ist wie eben beschrieben. Neu hier ist, daß die Assoziationsrelation zwischen *LoyaltyProgram* und *Customer* nur auf einer Seite einen

Rollennamen aufweist. Die allgemeine Regel in einem solchen Fall schreibt vor, daß man einen fehlenden Rollennamen durch die Klasse ersetzen kann, die an dem betreffenden Ende der Assoziation steht.

Bisher sind in OCL Ausdrücken nur Assoziationsrelationen und deren Rollenbezeichner vorgekommen. In dem folgenden Beispiel, das sich auf das Diagramm in Abbildung 6.1 bezieht, kommt auch ein Attribut vor.

OCL Beispiel 4

Company
numberOfEmployees > 0

Der Auswertung des Ausdrucks `numberOfEmployees` ist, entsprechend dem Typ des Attributs, eine ganze Zahl.

Der nächste Ausdruck zeigt, daß auch auf Methoden des UML-Modells zugegriffen werden kann:

OCL Beispiel 5

CustomerCard
validFrom.isBefore(goodThru)

hierbei ist

`isBefore(t:Date):Boolean`

eine Operation der Klasse *Date*, die wir hier nicht wiedergegeben haben.

Wie wir schon gesehen haben kann einem Objekt der Kontextklasse durch eine Folge von durch Punkten getrennten Rollen-, Klassen-, Attribut-, Methodenbezeichnern ein Objekt eines wohldefinierten Typs zugeordnet werden. Als Typen können alle UML Typen, vordefinierte und benutzerdefinierte, auftreten sowie einige in OCL eingebaute Typen. Wir wollen zunächst den Typ *Gesamtheit* (im Englischen Original *collection* genannt) betrachten. Gesamtheit (*collection*) wird in OCL verwendet als gemeinsamer Oberbegriff für

- Menge (set)
Jedes Element kommt nur einmal vor.
- Multimenge (bag)
Elemente können mehrfach auftreten.
- Folge (sequence)
Elemente können mehrfach auftreten und sind in einer Reihenfolge angeordnet.

Auf die Objekte vom Type *Gesamtheit* kann man vordefinierte Operationen anwenden, die man durch das Zeichen \rightarrow von dem restlichen Ausdruck trennt. Beispiele sind

OCL Beispiel 6

LoyaltyProgram
serviceLevel \rightarrow size = 2

OCL Beispiel 7

Customer
programm \rightarrow size = card \rightarrow select(valid = true) \rightarrow size

OCL Beispiel 8

LoyaltyProgram
partners.deliveredServices \rightarrow forAll(
 pointsEarned = 0 and pointsBurned = 0)
 implies membership.loyaltyAccount \rightarrow isEmpty

Abbildung 6.5 zeigt eine (fast) vollständige Übersicht der Operatoren, die für alle Gesamtheiten anwendbar sind. Weitere Beispiele für mengentheoretische Operatoren sind

OCL Beispiel 9

Membership
program.serviceLevel \rightarrow includes(actualLevel)

OCL Beispiel 10

ServiceLevel
loyaltyProgram.partners \rightarrow
 includesAll(availableServices.programPartner)

Die bisher betrachteten Einschränkungen können alle als *Invarianten* aufgefasst werden. Invarianten sind ein Beispiel für *Stereotypen* in UML. Im allgemeinen sind Stereotypen benutzerdefinierte Erweiterungen der Modellobjekte. Neben den Invarianten gibt es in UML noch die vordefinierten Stereotypen *Vorbedingung* und *Nachbedingung* (precondition und postcondition). Diese können eingesetzt werden z.B. zur Spezifikation von Methoden in Klassendiagrammen, zur Angabe von Ausführungsvoraussetzungen in Ereignisdiagrammen (event diagrams) oder zur Beschreibung von Zustandsübergängen in Zustandsübergangendiagrammen (state transition diagrams). Wir betrachten hier nur Beispiele für den ersten Fall.

OCL Beispiel 11

LoyaltyProgramm::enroll(c : Customer)

```
pre   : not customer→includes(c)
post  : customer = customer@pre→including(c)
post  : membership→select(customer = c)→forAll(
        loyaltyAccount→notEmpty() and
        loyaltyAccount.points = 0 and
        loyaltyAccount.transactions→isEmpty
    )
```

OCL Beispiel 12

Transactions::program():LoyaltyProgram

```
post  : result = self.card.membership.program
```

In dem Dokument [OMG97b] wird genau beschrieben welche Kombinationen der Modellelemente von UML ein korrektes UML Modell darstellen. Dies geschieht im wesentlichen durch Angabe eines Metamodells, d.h. die Beschreibungsmittel von UML werden benutzt um die UML Syntax festzulegen. So gibt es z.B. eine Metaklasse *Assoziationen*, deren Instanzen dann die Assoziationen eines UML Diagramms sind. In Abbildung 6.6 ist ein (kleiner) Ausschnitt aus dem UML Metamodell aus [OMG97b] wiedergegeben. Das Interessante ist nun, daß Diagramme dieser Art nicht ausreichen um hinreichend genau zu beschreiben, was man haben möchte. Deswegen enthält das Standarddokument [OMG97b] zusätzliche Einschränkungen notiert in der OCL Sprache. Einige Beispiel davon wollen wir uns anschauen.

OCL Beispiel 13

Associations

```
self.connection→forAll( r1, r2 | r1.name = r2.name implies r1 = r2)
```

OCL Beispiel 14

Associations

```
self.connection→select(aggregation <> #none)→size <=1
```

OCL Beispiel 15

AssociationClass

```
self.allConnections→forAll( ar | ar.type <> self)
```

wobei der zusätzliche Operator allConnections definiert ist durch

```
allConnections = self.connection→union(self.supertype→
    select(s | s.oclIsKindOf(association))→
    collect( a : Association | a.connection))→ asSet
```


Das *iterate* Konstrukt ist eine Besonderheit von OCL, die sie wesentlich von anderen, logik-basierten Spezifikationsprache unterscheidet.

Falls

- S, T OCL Typen sind,
- x eine Variable vom Typ S , y eine Variable vom Typ T , $y \neq x$,
- t ein OCL Ausdruck in dem y nicht vorkommt,
der Typ von t muß eine Gesamtheitstyp von Elementen des Typs S sein,
- t_0 ein Ausdruck vom Typ T , in dem weder x noch y vorkommen und
- u ein OCL Ausdruck vom Typ T ,

dann ist auch

$$t \rightarrow \mathbf{iterate}(x; y = t_0 \mid u)$$

ein OCL Ausdruck und zwar vom Typ T .

Um die Interpretation eines **iterate** Ausdrucks

$$m = I_\beta(\mathbf{iterate}(x : S; y : T = t_0 \mid u)t)$$

zu berechnen geht man wie folgt vor:

Sei $\{a_1, \dots, a_n\} = I_\beta(t)$ die Interpretation von t .

Wir definieren m_i für $0 \leq i < n$:

1. $m_0 = I_\beta(t_0)$
2. $m_{i+1} = I_{\beta^{i+1}}(u)$
where

$$\beta^{i+1}(z) = \begin{cases} \beta(z) & \text{falls } z \neq x, y \\ a_{i+1} & \text{falls } z = x \\ m_i & \text{falls } z = y \end{cases}$$

Schließlich ist $m = m_n$.

Überzeugen Sie sich, da $t \rightarrow \mathbf{iterate}(x : S; y : Bool = true \mid y \wedge u)$ gleichbedeutend ist mit $t \rightarrow \mathbf{forAll}(x : S \mid u)$.

6.3 Übersetzung in Prädikatenlogik

Wenn wir in OCL formalisierte Einschränkungen in die Sprache der Prädikatenlogik übersetzen wollen, müssen wir uns zuerst auf ein Vokabular (eine Signatur) einigen. Da sich OCL Ausdrücke auf ein UML Diagramm beziehen, brauchen wir Relations- und Funktionszeichen, welche die darin vorkommenden Klassen, Assoziationen, Attribute und Methoden in adäquater Weise bezeichnen. Betrachten wir zuerst das einfache Beispiel aus Abb.6.3. Um die darin enthaltene Information ausdrücken zu können führen wir ein:

Kurzform	Name	Bedeutung
$p(x)$	$Person(x)$	x ist ein Objekt der Klasse $Person$
$fa(x)$	$Firma(x)$	x ist ein Objekt der Klasse $Firma$
$vor(x, y)$	$vorgesetzter(x, y)$	die Objekte x, y aus der Klasse $Person$ sind assoziiert
$abg(x, y)$	$arbeitgeber(x, y)$	die Firma y ist Arbeitgeber der Person x
$ags(x, y)$	$angestellter(x, y)$	die Person x ist Angestellter der Firma y

Jeder Klasse des UML Modells wurde ein einstelliges Relationszeichen mit dem gleichen Namen zugeordnet. Alternativ dazu hätte man jeder Klasse auch einen Typ (oder Sorte) zuordnen können. Da wir jedoch bisher getypte (sortierte) Sprachen und Kalküle nur unzureichend behandelt haben, war dieser Weg hier nicht so attraktiv.

Jeder Assoziation wird ein zweistelliges Relationszeichen zugeordnet. Obwohl es im UML Standard vorgesehen ist, werden Assoziationen selbst selten mit einem Namen versehen. Im Gegensatz dazu tragen AssoziationsEnden häufig einen Namen, der als Rollennamen in einer Assoziation aufgefasst wird. Die Benennung der Relationen ist somit ziemlich willkürlich. Wir haben versucht uns in diesem Beispiel an den Rollennamen zu orientieren. Die Vorgesetztenrelation zwischen Personen trägt nur einen Rollenbezeichner, den wir dann auch als Namen für die ganze Relation, $vor(x, y)$, übernommen haben. Für die Assoziation zwischen Personen und Firmen sind beide Rollennamen vorhanden, was zu den beiden Relationszeichen $abg(x, y)$ und $ags(x, y)$ führt. Wegen

$$\forall x, y (abg(x, y) \leftrightarrow ags(y, x))$$

ist diese Bezeichnungsweise jedoch redundant und man sollte sich auf eine der beiden Namen festlegen. Grundsätzlich ist es möglich eine Assoziation A auch durch eine Funktion $f_A(x)$ in einer prädikatenlogischen Struktur zu modellieren, wobei für ein Objekt b der Wert $f_A(b)$ die Menge aller mit b assoziierten Objekte sein müsste. Die Tatsache, daß man bei einer Modellierung durch Funktionen Objekte einer mengentheoretisch höheren Stufe betrachten müsste und die Unvereinbarkeit der Hintereinanderausführung von Funktio-

nen mit dem *flattening* Konzept von OCL haben uns von dieser Möglichkeit Abstand nehmen lassen. Denkbar ist jedoch Assoziationen zwischen Klassen A und B , die am B -Ende mit der Multiplizität 1 gekennzeichnet sind, als Funktionen von A nach B zu modellieren.

In dem UML Modell aus Abb. 6.3 kommen keine Attribute vor. Es sei jedoch hier schon vorweggeschickt, daß wir Attribute in ein- oder mehrstellige Funktionen übersetzen werden.

Die Einschränkung aus Beispiel 1 liest sich in prädikatenlogischer Repräsentation als

$$\forall x, y, u_1, u_2(\\ p(x) \wedge p(y) \wedge fa(u_1) \wedge fa(u_2) \wedge \\ agb(x, u_1) \wedge vor(x, y) \wedge abg(y, u_2) \rightarrow u_1 = u_2)$$

Eine etwas übersichtlichere äquivalente Möglichkeit ist

$$\forall x, y(vor(x, y) \rightarrow p(x) \wedge p(y)) \\ \forall x, u(abg(x, u) \rightarrow p(x) \wedge fa(u)) \\ \forall x, y, u_1, u_2(agb(x, u_1) \wedge vor(x, y) \wedge abg(y, u_2) \rightarrow u_1 = u_2)$$

membership ist eine Assoziationsklasse. Ihre Objekte sind also Paare (lp, c) , wobei lp ein *LoyaltyProgram* ist und c ein Kunde (customer). Durch Assoziationen werden mit jedem solchen Paar weitere Attribute verbunden. Wir betrachten die Paare in *membership* als eigenständige Objekte m , denen durch die Funktionen $pr(m)$ und $c(m)$ ihre erste bzw. zweite Komponente zugeordnet wird. Die weiteren Assoziationen zu m werden durch Relationen modelliert. Die Relation $card(m, c)$ zeigt an, daß c eine Karte ist, die der Mitgliedschaft m zugehört. Es könnte ja mehrere Karte für eine Mitgliedschaft geben, gültige, ungültige, verlorene und neuausgestellte. Als Alternative zu dem hier gewählten Vorgehen, hätte man auch alle drei Assoziationen der Mitgliedschaftsklasse zu einer einzigen, dann fünfstelligen, Relationen $Mem(x, y, z, u, v)$ zusammenfassen können. Da mehr als zweistellige Relationen für den Menschen schwer zu durchschauen sind haben wir diesen Weg nicht beschritten.

Übersetzung von Beispiel 3

membership

card.owner = customer

$$\forall x, y(m(x) \wedge card(x, y) \rightarrow own(y) = c(x))$$

Übersetzung von Beispiel 5

CustomerCard

validFrom.isBefore(goodThru)

$$\forall x(cc(x) \rightarrow from(x) < thru(x))$$

Übersetzung von Beispiel 6

LoyaltyProgram

serviceLevel→size = 2

$$\begin{aligned} \exists u, w & \quad (issl(u) \wedge issl(w) \wedge u \neq w \wedge \\ \forall x, y & \quad (sl(x, y) \wedge lp(x) \wedge issl(y) \rightarrow \\ & \quad (y = u \vee y = w)) \quad \wedge \\ \forall x \exists z_1, z_2 & \quad (sl(x, z_1) \wedge sl(x, z_2) \wedge z_1 = u \wedge z_2 = w)) \end{aligned}$$

In diesem Beispiel klingt an, daß man sich vielleicht doch eine Erweiterung der Prädikatenlogik wünscht, in der es einfacher ist über die Kardinalität endlicher Mengen zu reden. Im vorliegenden Fall muß nur ausgedrückt werden, daß die Menge der *servicelevels* genau zwei ist. Das geht noch. Würde statt zwei die Zahl, sagen wir, sieben, auftreten, wäre eine direkte Kodierung in Prädikatenlogik zwar immer noch möglich, aber nicht mehr praktikabel. Im vorliegenden Fall könnte man sich auch eine andere Variante vorstellen. Man führt Konstanten *gold* und *silber* ein, die Objekte der Klasse *ServiceLevel* bezeichnen, die eindeutig durch die entsprechenden Namen gekennzeichnet sind. Dann ist die folgende Übersetzung sinnvoll

$$\begin{aligned} \forall x, y & \quad (sl(x, y) \wedge lp(x) \wedge issl(y) \rightarrow \\ & \quad (y = gold \vee y = silver)) \quad \wedge \\ \forall x & \quad (sl(x, gold) \wedge sl(x, silver)) \end{aligned}$$

Übersetzung von Beispiel 7

Customer

program→size = card→select(valid = true)→ size

$$\forall x(isCustomer(x) \rightarrow size(\{y \mid cp(x, y)\}) = size(\{y \mid cards(x, y) \wedge valid(y)\}))$$

Übersetzung von Beispiel 8

LoyaltyProgram

partners.deliveredServices→forAll(

pointsEarned = 0 and pointsBurned = 0)

implies membership.loyaltyAccount→isEmpty

Erste Formulierung, in welcher die mengentheoretische Schreibweise noch beibehalten wurde:

$$\begin{aligned} \forall x(lp(x) \rightarrow & (\forall s \in \{z \mid \exists y(pp(x, y) \wedge serv(y, z))\} \\ & (pointsBurned(s) = 0 \wedge pointsEarned(s) = 0) \\ \rightarrow & \\ & \{w \mid \exists u(m(u) \wedge pr(u, x) \wedge lA(u, w))\} = \emptyset)) \end{aligned}$$

Zweite Formulierung ohne mengentheoretische Schreibweise:

$$\begin{aligned} \forall x(lp(x) \wedge & \forall y, z(pp(x, y) \wedge serv(y, z) \rightarrow \\ & (pointsBurned(z) = 0 \wedge pointsEarned(z) = 0)) \\ \rightarrow & \\ & \neg \exists u, w(m(u) \wedge pr(u, x) \wedge lA(u, w)) \end{aligned}$$

6.4 Übungsaufgaben

Übungsaufgabe 6.4.1

Übersetzen Sie die OCL Einschränkung aus Beispiel 4

Company

numberOfEmployees > 0

in eine prädikatenlogische Formel.

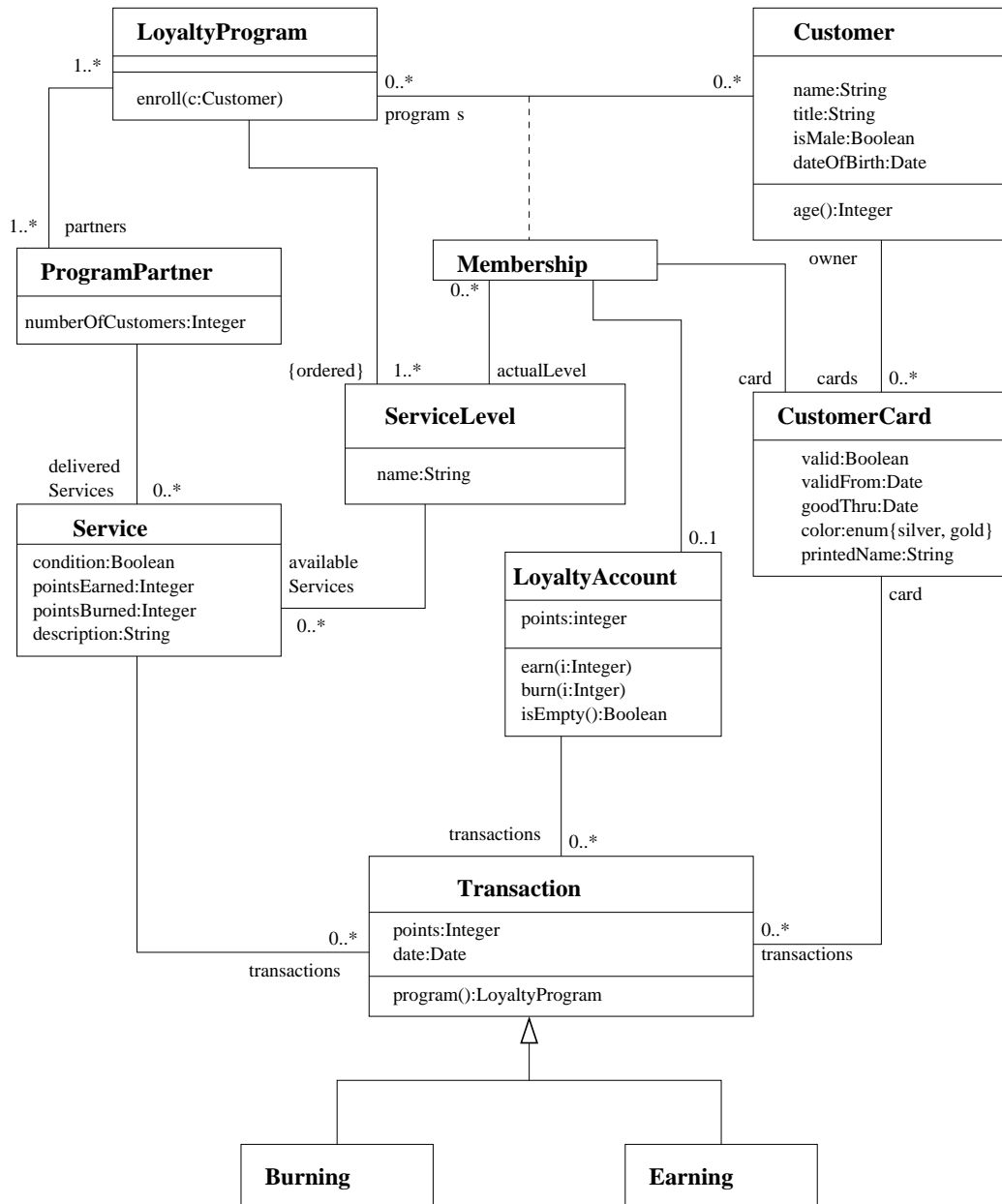


Abbildung 6.4: Beispiel 2 eines Klassendiagramms

Sei M ein Objekt vom Typ *Gesamtheit*.

Operation	Erklärung
size	Anzahl der Elemente von M
count(o)	Anzahl der Vorkommen von o in M
includes(o)	Wahr, wenn o ein Element von M ist
includesAll(c)	Wahr, wenn c eine Teilmenge von M ist
isEmpty	Wahr, wenn M kein Element enthält
notEmpty	Wahr, wenn M ein Element enthält
iterate(exp)	exp wird für jedes Element aus M ausgewertet. Der Ergebnistyp hängt von exp ab
exists(exp)	Wahr, wenn exp auf mindestens ein Element aus M zutrifft
forAll(exp)	Wahr, wenn exp auf alle Elemente von M zutrifft
select($e \mid exp$)	$\{m \in M \mid exp(m/e) \text{ ist wahr} \}$
collect($e \mid exp$)	$\{exp(m/e) \mid m \in M\}$

Abbildung 6.5: Operationen, die für alle Gesamtheiten anwendbar sind

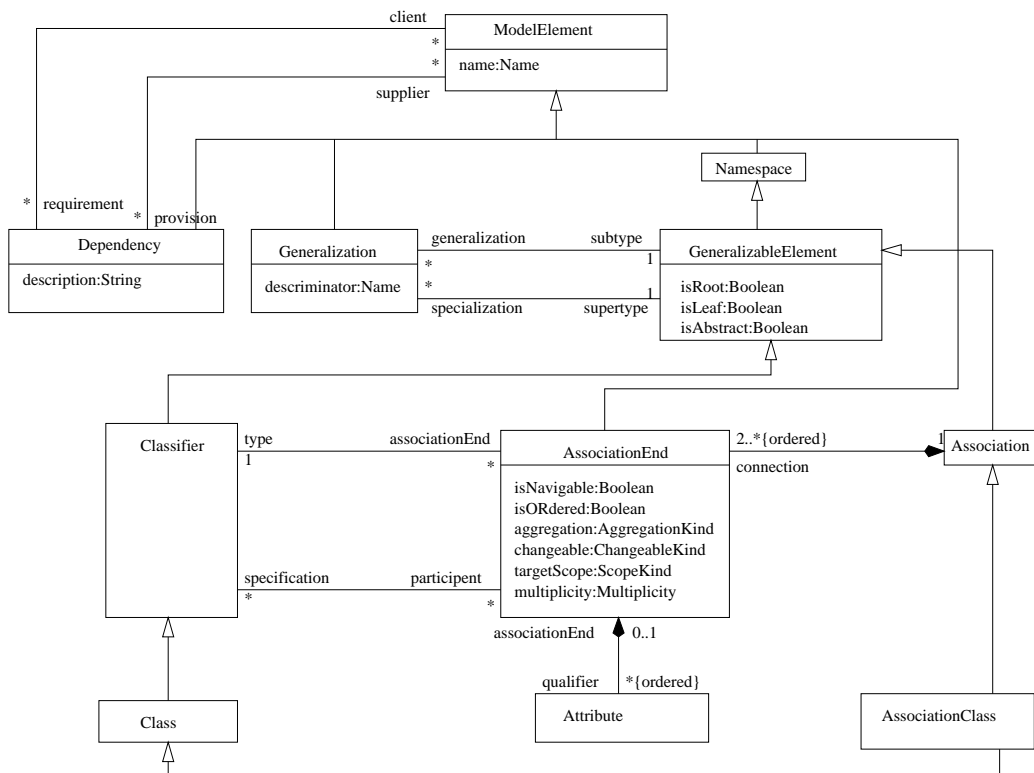


Abbildung 6.6: Ausschnitt aus dem UML Metamodell

Kurzform	Name	Bedeutung
$m(x)$	$membership(x)$	x ist ein Objekt der Assoziationsklasse <i>Membership</i>
$pr(x)$	$program(x)$	Funktion, die jedem Objekt der Klasse <i>membership</i> das Programm zuordnet, zu dem es gehört
$c(x)$	$customer(x)$	Funktion, die jedem Objekt der Klasse <i>membership</i> den Kunden zuordnet, zu dem es gehört
$isC(x)$	$isCustomer(x)$	x ist ein Objekt der Klasse <i>Customer</i>
$card(m, c)$	$card(m, c)$	Relationen zwischen einer Mitgliedschaft m
$cards(c, cc)$	$cards(c, cc)$	Relationen zwischen einem Customer und einer CustomerCard
$own(c)$	$owner(x)$	und einer zugehörigen Karte c Funktion, die jeder Karte ihren Eigentümer zuordnet
$from(x)$	$validFrom(x)$	Funktionen, die gleichnamige Attribute der Klasse <i>Card</i> bezeichnet
$thru(x)$	$goodThru(x)$	Relation zur Bezeichnung der Booleschwertigen Methode <i>isBefore(d)</i> aus der Klasse <i>Date</i>
$d_1 < d_2$	$isBefore(d_1, d_2)$	
$cc(x)$	$CustomerCard(x)$	x ist Objekt der Klasse <i>CustomerCard</i>
$lp(x)$	$LoyaltyProgram(x)$	x ist ein Objekt der Klasse <i>LoyaltyProgram</i> (Treueprogramm)
$issl(x)$	$ServiceLevel(x)$	x ist ein Objekt der Klasse <i>ServiceLevel</i>
$sl(x, y)$	$serviceLevel(x, y)$	Im Programm x gibt es der Servicelevel y
$<_{sl}$		Ordnungsrelation auf den Servicelevels, die einem Treueprogramm zugeordnet sind.

Abbildung 6.7: Vokabular zur logischen Modellierung des UML Diagramms aus Abb.6.4

Kapitel 7

Modale Aussagenlogik

7.1 Einführung

Die folgende Darstellung orientiert sich u.a. an den Büchern [Pop94, HR00, FM99],

Im Unterschied zur klassischen Logik, in der nur die Wahrheit einer Aussage von Bedeutung ist, spielt in der modalen Logik die Art und Weise, der Modus, in der eine Aussage wahr ist eine ausschlaggebende Rolle. Beispielsweise kann eine Aussage

- notwendigerweise wahr, zufälligerweise wahr
- heute, gestern oder morgen wahr sein
- geglaubt werden, zum Wissen einer Person gehören,
- oder vor/nach einer Aktion wahr, nach Ausführung eines Programms wahr sein.

Wir beginnen mit zwei Einführungsbeispielen.

Das Puzzle von den drei Weisen Das folgende Puzzle ist in vielen Varianten weit verbreitet:

Drei Weisen werden Hüte aufgesetzt, jedem genau einen. Die Hüte sind entweder weiß oder schwarz, und jedem ist bekannt, daß mindestens ein schwarzer Hut mit dabei ist. Jeder Beteiligte sieht, welche Hüte die anderen beiden aufsitzen haben und soll erschließen, welchen Hut er aufsitzen hat, natürlich ohne in einen Spiegel zu schauen, den Hut abzunehmen oder ähnliches. Nach einer Weile sagt der erste Weise: „Ich weiß nicht, welchen Hut ich auf habe.“ Nach einer weiteren Pause des Nachdenkens sagt der zweite: „Ich weiß auch nicht, welchen Hut ich auf habe.“ „Dann“, sagt der dritte, „weiß ich, daß ich einen schwarzen Hut auf habe.“

In Abbildung 7.1 ist zunächst die Menge aller möglichen Welten zu sehen. Das Tripel (b,w,b) z.B. steht dabei für die Welt, in der der erste Weise einen schwarzen Hut auf hat („b“ für black), der zweite Weise einen weißen und der dritte wieder einen schwarzen Hut auf hat. Die Welt (w,w,w) kommt nicht, da sie durch die Spielregeln ausgeschlossen ist. Aber die Abbildung zeigt noch mehr. Sie zeigt welche Welten jeder der drei Weisen für möglich hält. Das hängt natürlich ab, von der Welt in der er sich befindet. Stellen wir uns vor,

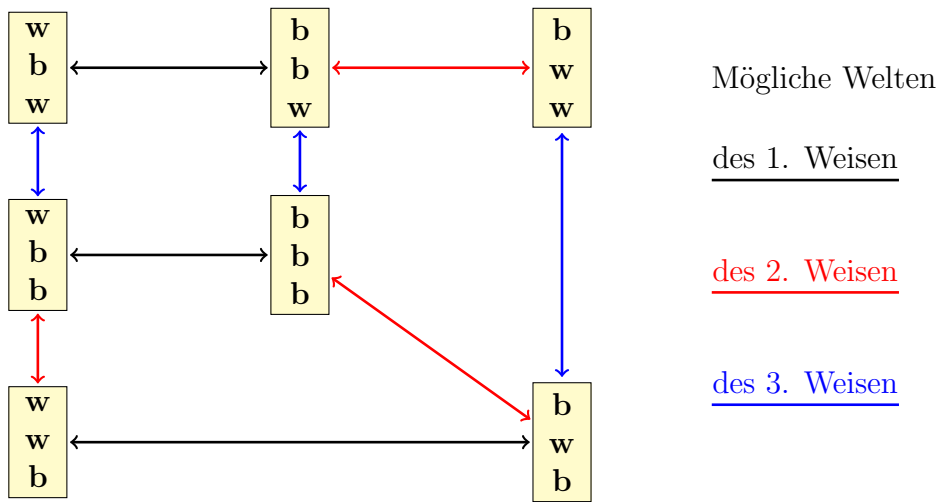


Abbildung 7.1: Mögliche Welten im Drei-Weisen-Beispiel

der erste Weise befindet sich in der Welt (w,b,b). Er sieht zwei schwarze Hüte und hält zu Beginn die beiden Welten (w,b,b) und (b,b,b) für möglich. Das ist in Abbildung 7.1 durch Pfeile gekennzeichnet, wobei Pfeile, die in den eigenen Zustand zurückverweisen, der Übersichtlichkeit halber weggelassen wurden.

Nach dem Geständnis des ersten Weisen, er wisse nicht welche Farbe sein Hut hat, ist allen Beteiligten klar, daß die Welt (b,w,w) ausgeschlossen werden kann. Denn in dieser Welt sieht der erste Weise zwei weiße Hüte und weiss sofort, daß er den schwarzen auf haben muß. In der Abbildung 7.1 kann man diese Argumentation auch direkt ablesen. Aus Welt (b,w,w) führt kein schwarzer Pfeil, sie ist für den ersten Weise ohne Alternative. Die neue Situa-

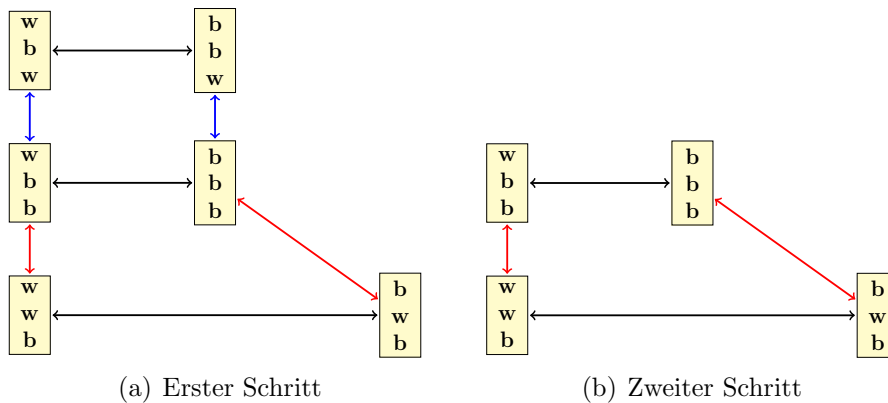


Abbildung 7.2: Reduktion der möglichen Welten

tion ist in Abbildung 7.2(a) zu sehen. Nach der Äußerung des zweiten Weisen, wissen wir und alle Mitspieler, daß die Welt (w,b,w) nicht vorkommen kann. Der zweite Weise hätte sofort gewußt, daß er den schwarzen Hut auf hat. In Abbildung 7.2(a) gibt es aber noch eine zweite Welt, die für der zweiten Weisen ohne Alternative ist, nämlich (b,b,w) . Denn hätte er in der Situation $(b,?,w)$ einen weißen Hut auf, so hätte der erste Weise wissen müssen, daß er den einzigen schwarzen auf hat. Nach Elimination dieser beiden Welten ist die Situation in Abbildung 7.2(b) erreicht. Man sieht, daß dabei nur Welten vorkommen, in denen der dritte Weise einen schwarzen Hut auf hat.

Der Graph in Abbildung 7.1 kann als Modell für das Wissen der drei Akteure gedeutet werden. Die Aussage

in der Welt s weiß der i -te Weise die Aussage A

wird dabei modelliert durch

in jeder für den i -te Weisen von s ausgesehen möglichen Welt gilt A .

In modallogischer Notation wird diese Aussage geschrieben als

$$s \models \Box_i A$$

Vereinbaren wir, daß die Boolesche Variable B_i wahr ist in einer Welt, in der der i -te Weise einen schwarzen Hut auf hat und entsprechend für W_j dann gilt in Abbildung 7.1 z.B.

$$\begin{array}{ll} (w, b, w) \models \Box_1 B_2 & (w, b, w) \models \Box_1 W_3 \\ \text{nicht } (w, b, w) \models \Box_1 B_1 & (b, w, w) \models \Box_1 B_1 \end{array}$$

Konfliktfreie Zugriffskontrolle Ein beliebiger Algorithmus, der den konfliktfreien Zugriff mehrerer Prozesse auf eine kritische Ressource regelt, ist der sogenannte *bakery algorithm*. Der Name kommt von der in amerikanischen Bäckereien und *delicatessen shops* üblichen Praxis, daß der Kunde beim Eintreten eine Nummer zieht und dann wartet bis seine Nummer die kleinste unter den noch Wartenden ist. Dann ist er an der Reihe. Bei uns kennt man dieses Verfahren aus manchen Arztpraxen. Im Gegensatz zu dem namensgebenden Beispiel geht der Algorithmus nicht davon aus, daß es ein zentrales Gerät gibt, welches die Nummern verteilt. Das Problem, welches der *bakery* Algorithmus löst, ist im Englischen unter dem Namen *mutual exclusion* bekannt. Dieser Algorithmus ist überaus beliebt als Lehrbuchbeispiel und in einfache Fallstudien.

In der Beschreibung des Algorithmus gehen wir davon aus, daß jeder Prozess sich in einem der Zustände `idle`, `trying` oder `critical` befindet. Außerdem besitzt jeder Prozeß ein *ticket*, eine Nummer, die ist eine natürliche Zahl größer 0. Prozesse, die noch nicht ihre Absicht bekundet haben, auf die kritische Ressource zuzugreifen, haben die Nummer 0. Wir können uns Prozesse

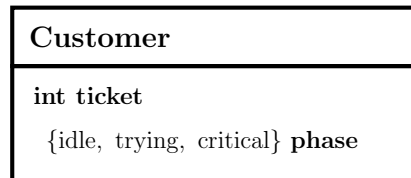


Abbildung 7.3: Klassenspezifikation für Prozesse

vorstellen als Instanzen der Klasse *Customer*, siehe Abb.7.3. Der Algorithmus kann jetzt durch die folgenden Zustandsübergangsregeln beschrieben werden. Die Protokollregeln aus Tabelle 7.1 sehen aus, wie die Zustandsübergänge

try:	if phase = idle	then phase := trying ticket := max of all other tickets + 1
enter:	if phase = trying and ticket less than all other tickets	then phase := critical
leave	phase = critical	then phase := idle ticket := 0

Tabelle 7.1: Zustandsübergänge des *bakery* Protokolls

eines Automaten. Der einzig wichtige Unterschied besteht darin, daß Automaten in ihrer üblichen Form nur endliche viele Zustände haben dürfen. Das können wir erreichen, indem wir die Anzahl der beteiligten Prozesse und die maximale Nummer beschränken. Bei zwei Prozessen und maximaler Nummer 8 erhalten wir einen Automaten mit $576 = (3 * 8)^2$ Zuständen. Daß einige dieser Zustände nicht erreichbar sind, soll uns im Augenblick nicht interessieren. 576 Zustände sind für ein Lehrbuchbeispiel zu viel. Wir wollen daher zunächst die Nummern ganz ignorieren. Für zwei Prozesse bleiben dann noch 9 Zustände. Abbildung 7.4 zeigt diese Zustände einschließlich der

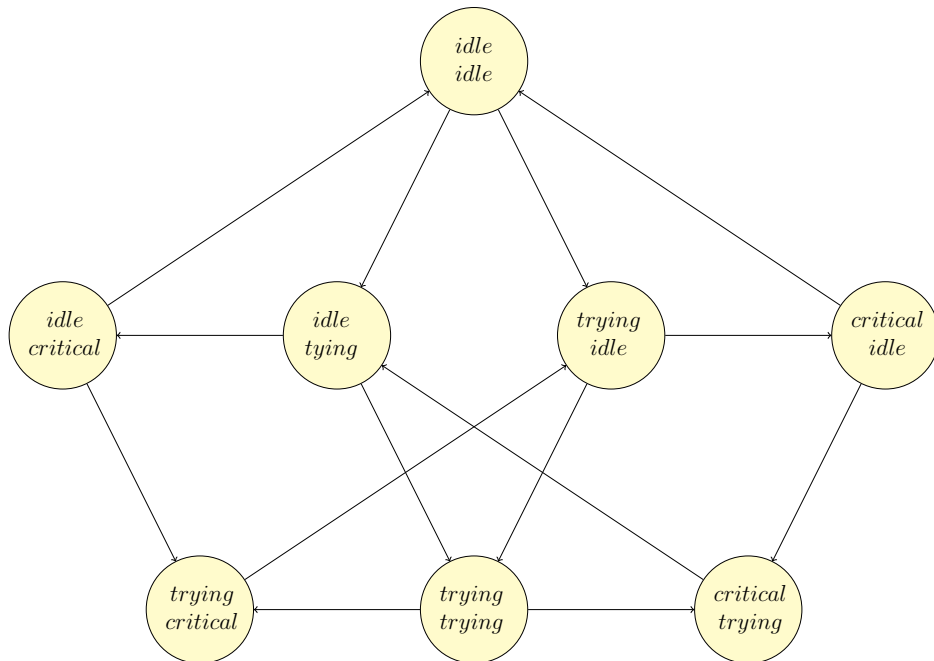


Abbildung 7.4: Zustandsübergänge im *bakery* Protokoll ohne Nummern

möglichen Übergänge. Der Zustand, daß beide Prozesse im kritischen Bereich sind, kommt nicht vor. Es ist die Aufgabe einer Protokollverifikation sicherzustellen, daß das gezeigte Modell mit einer Realisierung übereinstimmt. Eine Eigenschaft der Struktur in Abbildung 7.4, die für das Protokoll wichtig sein könnte, ist die Beobachtung, daß beide Prozess, wenn sie die Phase *trying* erreicht haben, immer in höchstens zwei Schritten in die kritische Phase kommen können, aber nicht müssen. Zur Formalisierung solche Eigenschaften ist ebenfalls die modale Logik geeignet. Die Booleschen Variablen $i.idle$, $i.trying$, $i.critical$ seien wahr in einem Zustand s , wenn in s der i -te Prozess in der angegebenen Phase ist. Dann ist die folgende Formel

$$1.trying \rightarrow (\diamond 1.critical \vee \diamond \diamond 1.critical)$$

in jedem Zustand wahr. Der modale Operator \diamond funktioniert dabei so, daß eine Aussage $\diamond A$ in einem Zustand s wahr ist, wenn es einen von s aus in einem Schritt erreichbaren Zustand t gibt, in dem A wahr ist.

7.2 Syntax und Semantik

Definition 7.1 (Syntax der modalen Aussagenlogik)

Sei Σ eine Menge aussagenlogischer Atome $\Sigma = \{P_0, P_1, \dots\}$ dann ist die Menge $mFor0_\Sigma$ (bzw. kürzer $mFor0$), der *Formeln der modalen Aussagenlogik*, definiert durch

1. $\mathbf{1} \in mFor0_\Sigma, \mathbf{0} \in mFor0_\Sigma$
2. $\Sigma \subseteq mFor0_\Sigma$
3. Mit $A, B \in mFor0_\Sigma$ liegen ebenfalls in $mFor0_\Sigma$: $\neg A, (A \circ B)$ für $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$
4. für $A \in mFor0_\Sigma$ gilt auch $\Box A \in mFor0_\Sigma$ und $\Diamond B \in mFor0_\Sigma$

Man liest $\Box A$ als *Box A* oder *A ist notwendig*. $\Diamond A$ wird gelesen als *Diamond A* oder *A ist möglich*.

Die modallogischen Operatoren lassen sich nicht mehr als Boole'sche Funktionen deuten, eine *reine* Wahrheitswertsemantik ist daher nicht mehr möglich. Für die Angabe einer Semantik ist, wie durch die Beispiele des vorangegangenen Unterkapitels nahegelegt wird, eine Menge von Zuständen notwendig. Anstelle einer Interpretation der aussagenlogischen Variablen im Falle der klassischen Aussagenlogik, braucht man in der Modallogik für jeden Zustand eine Belegung der aussagenlogischen Variablen. Als letztes kommt noch hinzu, daß die Zustände nicht beziehungslos nebeneinander stehen, sondern durch eine Zugänglichkeitsrelation miteinander verbunden sind. All diese Angaben werden in dem Begriff einer *Kripke-Struktur* zusammengefasst.

Definition 7.2 (Kripke- Struktur)

Eine *Kripke-Struktur* $\mathcal{K} = (S, R, I)$ über Σ ist gegeben durch

- eine nichtleere Menge S von Zuständen
- eine Relation $R \subseteq S \times S$
- eine Interpretationsfunktion $I: (\Sigma \times S) \rightarrow \{W, F\}$

In der Modallogik nennt man Zustände gelegentlich auch *mögliche Welten*.

Zustände und Zugänglichkeitsrelation zusammen, ohne die Interpretation I , nennt man einen *Kripke-Rahmen*, $\mathcal{R} = (S, R)$.

Definition 7.3 (Auswertung der Formeln)

Sei $\mathcal{K} = (S, R, I)$, $s \in S$ ein Zustand, dann wird der Wahrheitswert einer Formel A im Zustand s mit $val_{(\mathcal{K},s)}(A)$ bezeichnet und wie folgt induktiv definiert:

$$\begin{aligned}
val_s(\mathbf{0}) &= \mathbf{F} \\
val_s(P) &= I(P)(s) \text{ f\u00fcr Atome } P \\
val_s(\neg A) &= \begin{cases} \mathbf{F} & \text{falls } val_s(A) = \mathbf{W} \\ \mathbf{W} & \text{falls } val_s(A) = \mathbf{F} \end{cases} \\
val_s(A \circ B) &= \begin{cases} \text{Wie in der Aussagenlogik} \\ \text{(wobei } \circ \text{ eine aussagenlogische Verkn\u00fcpfung ist)} \end{cases} \\
val_s(\Box A) &= \begin{cases} W & \text{falls f\u00fcr alle } s' \in S \text{ mit } sRs' \text{ gilt } val_{s'}(A) = \mathbf{W} \\ \mathbf{F} & \text{sonst} \end{cases} \\
val_s(\Diamond A) &= \begin{cases} W & \text{falls ein } s' \in S \text{ existiert mit } sRs' \\ & \text{und } val_{s'}(A) = \mathbf{W} \\ \mathbf{F} & \text{sonst} \end{cases}
\end{aligned}$$

Wir sagen \mathcal{K} ist ein *Modell* der Formel A , wenn $val_{(\mathcal{K},s)}(A) = W$ f\u00fcr alle $s \in S$.

Wenn \mathcal{K} aus dem Kontext ersichtlich ist, schreiben wir $val_s(A)$ anstelle von $val_{(\mathcal{K},s)}(A)$, wie in der vorangegangenen Definition schon geschehen.

Wir benutzen gleichberechtigt die alternative Schreibweisen:

$$\begin{aligned}
(\mathcal{K}, s) \models A &\Leftrightarrow val_{(\mathcal{K},s)}(A) = \mathbf{W} \\
\text{wenn } \mathcal{K} \text{ aus dem Kontext bekannt ist auch:} \\
s \models A &\Leftrightarrow val_s(A) = \mathbf{W} \\
\mathcal{K} \models A &\Leftrightarrow \text{f\u00fcr alle } s \in S \text{ gilt } (\mathcal{K}, s) \models A
\end{aligned}$$

Bemerkung 7.4

Bei den Zust\u00e4nden kommt es *nicht* etwa nur auf ihre Auswirkung auf val an, d.h. es kann durchaus in einer Kripke Struktur $\mathcal{K} = (S, R, I)$ zwei verschiedene Welten $s_1, s_2 \in S$ geben, so da\u00df f\u00fcr alle Atome p gilt $val_{s_1}(p) = val_{s_2}(p)$.

Definition 7.5

$$\begin{aligned}
A \text{ allgemeing\u00fcltig} &\Leftrightarrow val_{(\mathcal{K},s)}(A) = W \\
&\text{f\u00fcr alle } \mathcal{K} = (S, R, I) \text{ und alle } s \in S \\
A \text{ erf\u00fcllbar} &\Leftrightarrow \text{es gibt eine Kripke-Struktur } \mathcal{K} = (S, R, I) \\
&\text{und ein } s \in S \text{ mit } val_{(\mathcal{K},s)}(A) = W.
\end{aligned}$$

Man überzeugt sich leicht, daß wie in der klassischen Aussagenlogik gilt: A allgemeingültig $\Leftrightarrow \neg A$ unerfüllbar.

Lemma 7.6 (Allgemeingültige modale Formeln)

Alle Instanzen der folgenden modallogischen Schemata sind allgemeingültig.

1. Jede Tautologie ist allgemeingültig
2. $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$
3. $(\Box A \wedge \Box B) \leftrightarrow \Box(A \wedge B)$
4. $(\Box A \vee \Box B) \rightarrow \Box(A \vee B)$
5. $\Box A \leftrightarrow (\neg \Diamond \neg A)$
6. $\Diamond A \leftrightarrow (\neg \Box \neg A)$
7. $(\Diamond A \vee \Diamond B) \leftrightarrow \Diamond(A \vee B)$
8. $\Diamond(A \wedge B) \rightarrow (\Diamond A \wedge \Diamond B)$
9. $(\Box(A \rightarrow B) \wedge \Box(B \rightarrow A)) \leftrightarrow \Box(A \leftrightarrow B)$
10. $\Box(A \rightarrow B) \rightarrow (\Diamond A \rightarrow \Diamond B)$

Beweise

zu 1: Diese Aussage ist offensichtlich richtig. Wir geben eine Beispiel:

Beispiel 7.7

Die modallogische Formel (A und B seien aussagenlogische Atome)

$$((\underbrace{\Box A \wedge \neg \Box B}_{P_1}) \rightarrow \underbrace{\Box A}_{P_2}) \rightarrow ((\neg(\underbrace{\Box A \wedge \neg \Box B}_{P_1}) \rightarrow \underbrace{\Box A}_{P_2}) \rightarrow \underbrace{\Box A}_{P_2})$$

ist eine Instanz des aussagenlogischen Schemas:

$$(P_1 \rightarrow P_2) \rightarrow ((\neg P_1 \rightarrow P_2) \rightarrow P_2)$$

zu 2: Sei $\mathcal{K} = (S, R, I)$ eine beliebige Kripke-Struktur und s ein beliebiger Zustand in S . Wir wollen $s \models \Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$ zeigen. Dazu genügt es zu zeigen, daß aus der Annahme $s \models \Box(A \rightarrow B)$ folgt $s \models \Box A \rightarrow \Box B$. Dazu genügt es wiederum aus der zusätzlichen Annahme $s \models \Box A$ auf $s \models \Box B$ zu schließen. Nach der Semantikdefinition besagen die beiden Voraussetzungen, daß für alle s' mit $R(s, s')$ gilt

$$\begin{aligned} s' &\models A \rightarrow B \\ s' &\models A \end{aligned}$$

Daraus folgt für alle s' mit $R(s, s')$ auch $s' \models B$, d.h. $s \models \Box B$, wie gewünscht.

zu 3: Die Definition dafür, daß die linke Seite der Äquivalenz in einem Zustand s eine Kripke-Struktur gilt lautet:

für alle s' mit $R(s, s')$ gilt $s' \models A$ und
für alle s' mit $R(s, s')$ gilt $s' \models B$

Die rechte Seite führt zu

für alle s' mit $R(s, s')$ gilt $s' \models A \wedge B$

Die Äquivalenz ist jetzt klar ersichtlich.

zu 4: Aus

für alle s' mit $R(s, s')$ gilt $s' \models A$ oder
für alle s' mit $R(s, s')$ gilt $s' \models B$

folgt sicherlich

für alle s' mit $R(s, s')$ gilt $s' \models A \vee B$

Die umgekehrte Implikation ist nicht allgemeingültig, siehe Abb.7.5(a).

zu 5: Nach Einsetzen der Semantikdefinition für beide Seiten erhalten wir

für alle s' mit $R(s, s')$ gilt $s' \models A$

für die linke Seite und

es gibt kein s' mit $R(s, s')$, so daß $s' \models \neg A$ gilt.

für die rechte Seite. Offensichtlich sagen diese beiden Aussagen dasselbe.

zu 6: Folgt direkt aus 5. Damit meinen wir folgendes: Für $\neg A$ anstelle von A (schließlich ist A eine Schemavariablen) lautet 5 $\Box \neg A \leftrightarrow \neg \Diamond \neg \neg A$. Setzt man das in der rechten Seite von 6 ein, so ergibt sich $\Diamond A \leftrightarrow \neg \neg \Diamond \neg \neg A$. Nach den offensichtlichen Vereinfachungen also die Tautologie $\Diamond A \leftrightarrow \Diamond A$.

Nach demselben Muster kann man aus 5 und 3 auf 7 schließen. Ebenso folgt 8 aus 5 und 4. Die Äquivalenz 9 schließlich ist eine direkte Konsequenz aus 3.

zu 10: Hier müssen wir noch einmal auf die Semantikdefinition zurückgreifen. Die linke Seite liefert

$$\text{für alle } s' \text{ mit } R(s, s') \text{ gilt } s' \models A \rightarrow B$$

Um daraus auf die rechte Seite zu schließen nehmen wir $s \models \Diamond A$ an, d.h.

$$\text{es gibt ein } s' \text{ mit } R(s, s') \text{ und } s' \models A$$

Zusammengenommen erhalten wir

$$\text{es gibt ein } s' \text{ mit } R(s, s') \text{ und } s' \models B$$

. Was die Definition für $s \models \Diamond B$ ist.

■

Die beiden letzten Beispiele in Lemma 7.6 zeigen außerdem, daß einer der beiden Operatoren \Box , \Diamond entbehrlich ist. Es bilden also etwa $\{\mathbf{0}, \rightarrow, \Box\}$ eine Basis für die modale Aussagenlogik.

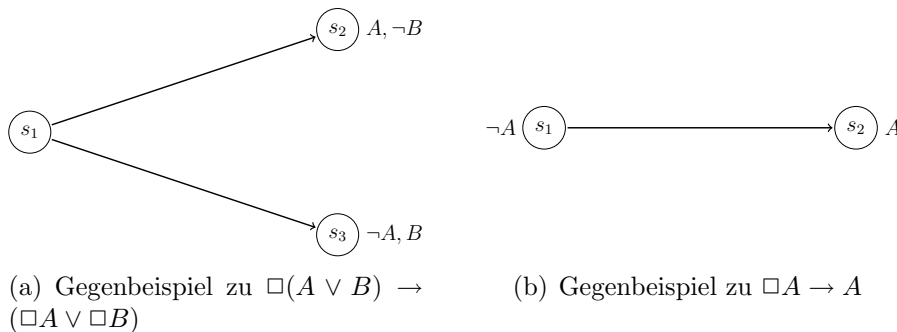


Abbildung 7.5: Gegenbeispiele

Definition 7.8

Sei A eine Formel und Γ eine Menge von Formeln der modalen Aussagenlogik.

1. A ist eine logische Folgerung aus Γ , in Symbolen $\Gamma \vdash A$, gdw für alle Kripke-Strukturen \mathcal{K} und jede Welt s von \mathcal{K} gilt: wenn $(\mathcal{K}, s) \models \Gamma$ dann auch $(\mathcal{K}, s) \models A$
2. A ist allgemeingültig wenn $\emptyset \vdash A$

Eine Variante wird als Definition 7.15 in Übungsaufgabe 7.5.2 betrachtet.

7.3 Eigenschaften von Kripke-Rahmen

Die Formel $\Box A \rightarrow A$ ist nicht allgemeingültig, siehe Abb.7.5(b) für ein Gegenbeispiel. Die Formel ist dagegen wahr in allen Kripke-Strukturen $\mathcal{K} = (S, R, I)$ mit reflexivem Kripke-Rahmen (S, R) . Wir nennen die gesamte Kripke-Struktur oder den ganzen Kripke-Rahmen reflexiv, wenn R eine reflexive Relation auf S ist.

Das ist nur die Spitze eines Eisbergs. Der Begriff der Allgemeingültigkeit kann auf die verschiedensten Teilklassen von Kripke-Strukturen relativiert werden.

Lemma 7.9 (Relative Allgemeingültigkeit modaler Formeln)

In den folgenden Formeln ist A eine aussagenlogische Variable.

1. In der Klasse aller reflexiven Kripke-Strukturen ist allgemeingültig:
 $\Box A \rightarrow A$
2. In der Klasse aller transitiven Kripke-Strukturen ist allgemeingültig:
 $\Box A \rightarrow \Box \Box A$
3. In der Klasse aller symmetrischen Kripke-Strukturen ist allgemeingültig:
 $A \rightarrow \Box \Diamond A$
4. In der Klasse aller dichten Kripke-Strukturen ist allgemeingültig:
 $\Box \Box A \rightarrow \Box A$
Dabei heißt eine Kripke-Struktur (S, R, I) dicht, wenn für alle $t_1, t_2 \in S$ mit $R(t_1, t_2)$ eine Welt $t_3 \in S$ existiert mit $R(t_1, t_3)$ und $R(t_3, t_2)$.
5. In der Klasse aller partiell funktionalen Kripke-Strukturen ist allgemeingültig:
 $\Diamond A \rightarrow \Box A$
Dabei heißt eine Kripke-Struktur (S, R, I) partiell funktional, wenn für alle $s, t_1, t_2 \in S$ mit $R(s, t_1)$ und $R(s, t_2)$ folgt $t_1 = t_2$.
6. In der Klasse aller endlosen Kripke-Strukturen ist allgemeingültig:
 $\Box A \rightarrow \Diamond A$
Dabei heißt eine Kripke-Struktur (S, R, I) endlos, wenn für jedes $s \in S$ ein t existiert mit $R(s, t)$.

Beweise

zu 1: Gilt $s \models \Box A$, dann gilt für alle s' mit $R(s, s')$ nach Definition $s' \models A$. Wegen der Reflexivität von R gilt insbesondere $R(s, s)$. Also $s \models A$.

zu 2: Gelte $s \models \Box A$, d.h. für alle s' mit $R(s, s')$ gilt $s' \models A$. Wir wollen $s \models \Box\Box A$ zeigen. Seien also beliebige s_1, s_2 gegeben mit $R(s, s_1)$ und $R(s_1, s_2)$, dann müssen wir $s_2 \models A$ zeigen. Wegen der Transitivität von R gilt aber unter diesen Voraussetzungen schon $R(s, s_2)$ und damit nach Voraussetzung $s_2 \models A$.

zu 3: Gelte $s \models A$. Wir wollen $s \models \Box\Diamond A$ zeigen. Dazu betrachten wir eine beliebige Welt t mit $R(s, t)$ und suchen ein s' mit $R(t, s')$ und $s' \models A$. Da R als symmetrisch vorausgesetzt ist, leistet $s' = s$ das Gewünschte.

zu 4: Gelte $t_1 \models \Box\Box A$, d.h. für alle t_3, t_2 mit $R(t_1, t_3)$ und $R(t_3, t_2)$ gilt $t_2 \models A$. Wir wollen zeigen, daß auch $t_1 \models \Box A$ gilt, d.h. für alle t_2 mit $R(t_1, t_2)$ ist $t_2 \models A$ zu zeigen. Wegen der Dichtheit gibt es ein t_3 mit $R(t_1, t_3)$ und $R(t_3, t_2)$, so daß aus der Voraussetzung tatsächlich $t_2 \models A$ folgt.

zu 5: Gelte $s \models \Diamond A$, d.h. es gibt t mit $R(s, t)$ und $t \models A$. Wir müssen $s \models \Box A$ nachweisen. Sei dazu t' eine beliebige Welt mit $R(s, t')$. Gilt $t' \models A$? Wegen der partiellen Funktionalitätseigenschaft muß $t = t'$ gelten, und wir sind fertig.

zu 6: Gilt $s \models \Box A$, dann gilt für alle t mit $R(s, t)$ definitionsgemäß $t \models A$. Das schließt im allgemeinen auch den Fall ein, daß kein t mit $R(s, t)$ existiert. Wegen der vorausgesetzten Endlosigkeit gibt es aber in der vorliegenden Situation mindestens ein solches t . Somit ist $s \models \Diamond A$ gezeigt.

■

Für die verschiedenen Klassen von Kripke-Strukturen und die dazugehörigen Logiken hat sich eine bizarre wenig systematische Bezeichnungskonvention etabliert:

die Klasse **K** aller Kripke-Strukturen

die Klasse **T** aller reflexiven Kripke-Strukturen

die Klasse **A4** aller transitiven Kripke-Strukturen

die Klasse **S4** aller reflexiven und transitiven Kripke-Strukturen

die Klasse **S5** aller transitiven und symmetrischen Kripke-Strukturen.

Die in Lemma 7.9 aufgelisteten Formeln sind so gewählt, daß sie die Klasse von Kripke-Strukturen, für die sie allgemeingültig sind, in einem gewissen Sinne charakterisieren. Die naive Vermutung, daß für eine Kripke-Struktur $\mathcal{K} = (S, R, I)$ mit $s \models \Box A \rightarrow \Box\Box A$ für alle $s \in S$, schon die Transitivität von (S, R) folgen würde, ist falsch. Ganz unabhängig von (S, R) wird $s \models \Box A \rightarrow \Box\Box A$ wahr, wenn wir nur $I(A, s) = W$ wählen für alle $s \in S$. Die folgende Definition beschreibt den richtigen Zusammenhang.

Definition 7.10

Sei \mathbf{R} eine Klasse von Kripke-Rahmen, und A eine Formel der Modallogik.

A charakterisiert die Klasse \mathbf{R} genau dann, wenn für alle Kripke-Rahmen (S, R) gilt

$$\begin{array}{l} \text{für alle } I \text{ gilt } (S, R, I) \models A \\ \text{gdw} \\ (S, R) \in \mathbf{R} \end{array}$$

Lemma 7.11 (Charakterisierungen)

In den folgenden Formeln ist A eine aussagenlogische Variable.

1. Die Formel $\Box A \rightarrow A$ charakterisiert die Klasse der reflexiven Kripke-Rahmen.
2. Die Formel $\Box A \rightarrow \Box\Box A$ charakterisiert die Klasse der transitiven Kripke-Rahmen.
3. Die Formel $A \rightarrow \Box\Diamond A$ charakterisiert die Klasse der symmetrischen Kripke-Rahmen.
4. Die Formel $\Box\Box A \rightarrow \Box A$ charakterisiert die Klasse der dichten Kripke-Rahmen.
5. Die Formel $\Diamond A \rightarrow \Box A$ charakterisiert die Klasse der partiell funktionalen Kripke-Rahmen.
6. Die Formel $\Box A \rightarrow \Diamond A$ charakterisiert die Klasse der endlosen Kripke-Rahmen.

Beweise In allen Beweisen ist eine Implikationsrichtung schon durch Lemma 7.9 abgedeckt. Wenn z.B. (S, R) transitiv ist, dann gilt für alle I schon $(S, R, I) \models \Box A \rightarrow \Box\Box A$. Im folgenden wird immer nur die verbleibende Richtung behandelt.

zu 1: Wir wollen zeigen, daß jeder Kripke-Rahmen (S, R) , für den $(S, R, I) \models \Box A \rightarrow A$ für alle I gilt, reflexiv ist. Wir benutzen hier, wie in allen nachfolgenden Fällen, Beweis durch Widerspruch: Angenommen (S, R) ist nicht reflexiv, dann gibt es ein I , so daß $(S, R, I) \not\models \Box A \rightarrow A$. Wenn (S, R) nicht reflexiv ist, dann gibt es ein $s_0 \in S$ mit $\neg R(s_0, s_0)$. Wir definieren eine Interpretation I durch

$$I(A, s) = \begin{cases} \mathbf{W} & \text{falls } s \neq s_0 \\ \mathbf{F} & \text{falls } s = s_0 \end{cases}$$

Nach dieser Definition gilt für $\mathcal{K} = (S, R, I)$ einerseits $(\mathcal{K}, s_0) \not\models A$ und andererseits $(\mathcal{K}, s_0) \models \Box A$. Also insgesamt $(\mathcal{K}, s_0) \not\models \Box A \rightarrow A$.

zu 2: Dieser Teil läuft, wie alle folgenden, nach demselben Muster ab, wie der erste. Wir fassen uns deswegen etwas kürzer.

Wenn ein Rahmen (S, R) nicht transitiv ist, dann gibt es $s_1, s_2, s_3 \in S$ mit $R(s_1, s_2)$ und $R(s_2, s_3)$ aber $\neg R(s_1, s_3)$. Wir definieren eine Interpretation I durch

$$I(A, s) = \begin{cases} \mathbf{W} & \text{falls } R(s_1, s) \\ \mathbf{F} & \text{falls } \textit{sonst} \end{cases}$$

Nach dieser Definition folgt sofort $s_1 \models \Box A$. Außerdem gilt $s_3 \models \neg A$. Da s_3 in zwei R -Schritten von s_1 aus erreichbar ist, haben wir auch $s_1 \models \neg \Box \Box A$. Insgesamt $s_1 \models \neg(\Box A \rightarrow \Box \Box A)$.

zu 3: Wenn ein Rahmen (S, R) nicht symmetrisch ist, dann gibt es $s_1, s_2 \in S$ mit $R(s_1, s_2)$ und $\neg R(s_2, s_1)$. Wir definieren eine Interpretation I durch

$$I(A, s) = \begin{cases} \mathbf{W} & \text{falls } s = s_1 \\ \mathbf{F} & \text{falls } \textit{sonst} \end{cases}$$

Trivialerweise gilt $s_1 \models A$. Für s_2 gibt es entweder überhaupt kein s mit $R(s_2, s)$ oder für alle s mit $R(s_2, s)$ gilt $s \models \neg A$. Das heißt aber $s_2 \models \neg \Diamond A$ und somit auch $s_1 \models \neg \Box \Diamond A$. Zusammengenommen erhalten wir den Widerspruch $s_1 \models \neg(A \rightarrow \Box \Diamond A)$.

zu 4: Ist (S, R) kein dichter Kripke-Rahmen, dann gibt es t_1, t_2 mit $R(t_1, t_2)$, so daß kein t_3 existiert mit $R(t_1, t_3)$ und $R(t_3, t_2)$. Wir definieren eine Interpretation I durch

$$I(A, s) = \begin{cases} \mathbf{W} & \text{falls } s = t_1 \text{ und es gibt } s_1, s_2 \\ & \text{mit } R(t_1, s_1) \text{ und } R(s_1, s_2) \\ \mathbf{F} & \text{falls } \textit{sonst} \end{cases}$$

Unmittelbar aus der Definition von I folgt $s \models \Box\Box A$. Andererseits gilt $t_2 \models \neg A$, also auch $t_1 \models \neg\Box A$.

zu 5: Ist (S, R) nicht partiell funktional, dann gibt es s_1, s_2, s_3 mit $R(s_1, s_2)$, $R(s_1, s_3)$ und $s_2 \neq s_3$. Wir definieren eine Interpretation I durch

$$I(A, s) = \begin{cases} \mathbf{W} & \text{falls } s = s_1 \\ \mathbf{F} & \text{falls sonst} \end{cases}$$

Somit gilt $s_1 \models \Diamond A$ und $s_1 \models \neg\Box A$. Insgesamt also $s_1 \models \neg(\Diamond A \rightarrow \Box A)$

zu 6: Ist (S, R) nicht endlos, dann gibt es $s_1 \in S$ mit $\neg R(s_1, s)$ für alle $s \in S$. Für jede beliebige Formel C gilt dann offensichtlich $s_1 \models \Box C$ und $s_1 \models \neg\Diamond C$. Insbesondere $s_1 \models \neg(\Box A \rightarrow \Diamond A)$. ■

7.4 Entscheidbarkeit modaler Logiken

Die Erfüllbarkeit einer Formel der klassischen Aussagenlogik kann in endlichen vielen Schritten entschieden werden. Schlimmstenfalls probiert man alle möglichen Interpretationen der vorkommenden aussagenlogischen Variablen aus. Um die Erfüllbarkeit einer Formel A der modalen Aussagenlogik nachzuweisen muß man eine Kripke-Struktur $\mathcal{K} = (S, R, I)$ angeben, so daß A in einer Welt $s \in S$ wahr ist. Ein einfaches Durchprobieren aller Möglichkeiten ist nicht mehr möglich, da die Anzahl der Welten in S beliebig groß sein kann. Wir wollen in diesem Unterabschnitt zeigen, daß die meisten modalen Aussagenlogiken dennoch entscheidbar sind. Der entscheidende Schritt besteht darin, eine obere Schranke für die Anzahl der möglichen Welten in einer erfüllenden Kripke-Struktur aus der Größe Formel A herzuleiten.

Das zentrale Konzept auf dem Weg dorthin ist die Filtration.

Definition 7.12 (Filtration)

Sei $\mathcal{K} = (G, R, v)$ eine Kripke Struktur und Γ eine Menge von Formeln der modalen Aussagenlogik.

Die Äquivalenzrelation \sim_Γ auf G wird definiert durch:

$$g \sim_\Gamma h \text{ gdw } (\mathcal{K}, g) \models A \Leftrightarrow (\mathcal{K}, h) \models A \text{ für alle } A \in \Gamma$$

Für $g \in G$ bezeichnen wir mit $[g]$ die Äquivalenzklasse von g bezüglich \sim_Γ , i.e. $[g] = \{h \in G \mid g \sim_\Gamma h\}$. Die Kripke Struktur $\mathcal{K}_\Gamma = (G_\Gamma, R_\Gamma, v_\Gamma)$ ist wie folgt definiert

$$\begin{aligned} G_\Gamma &= \{[g] \mid g \in G\} \\ [g]R_\Gamma[h] &\Leftrightarrow \exists g_0 \in [g]\exists h_0 \in [h](g_0R_\Gamma h_0) \\ v_\Gamma([g], p) &= v(g, p) && \text{für } p \in \Gamma \\ v_\Gamma([g], p) &= \text{beliebig} && \text{sonst} \end{aligned}$$

\mathcal{K}_Γ heißt eine *Filtration* von \mathcal{K} bezüglich Γ .

Lemma 7.13

1. Sei Γ eine Menge von Formeln der modalen Aussagenlogik, abgeschlossen unter Teilformeln, dann gilt für alle $A \in \Gamma$ und alle $g \in G$

$$(\mathcal{K}, g) \models A \text{ gdw } (\mathcal{K}_\Gamma, [g]) \models A$$

2. Ist Γ endlich mit $\#\Gamma = n$, dann ist $\#G_\Gamma \leq 2^n$.

Beweis:

(1) Der Beweis geschieht durch Induktion über den Formelaufbau von A . Ist A eine Aussagenvariable, dann folgt die Behauptung aus der Definition von v_Γ und der Äquivalenzrelation \sim_Γ . Wir betrachten nur noch den Induktionsschritt von B auf $\Box B$. Gelte $(\mathcal{K}_\Gamma, [g]) \models \Box B$. Dann gilt für alle $h \in G$ mit $[g]R_\Gamma[h]$ auch $(\mathcal{K}_\Gamma, [h]) \models B$. Nach Induktionsvoraussetzung folgt damit für alle $h \in G$ mit $[g]R_\Gamma[h]$ auch $(\mathcal{K}, h) \models B$ und damit auch für alle $h \in G$ mit $gR_\Gamma h$ ebenfalls $(\mathcal{K}, h) \models B$, also auch $(\mathcal{K}, g) \models \Box B$. Gelte jetzt umgekehrt $(\mathcal{K}, g) \models \Box B$. Wir betrachten $g, h \in G$ mit $[g]R_\Gamma[h]$. Nach Definition gibt es $g_0, h_0 \in G$ mit $g_0 \sim_\Gamma g$, $h_0 \sim_\Gamma h$ und $g_0R_\Gamma h_0$. Nach Definition von \sim_Γ gilt auch $(\mathcal{K}, g_0) \models \Box B$ und damit auch $(\mathcal{K}, h_0) \models B$. Woraus mit der Definition von \sim_Γ wieder $(\mathcal{K}, h) \models B$ folgt. Nach Induktionsvoraussetzung erhalten wir weiter $(\mathcal{K}_\Gamma, [h]) \models B$. Insgesamt ist damit für alle $[h] \in G_\Gamma$ mit $[g]R_\Gamma[h]$ die Gültigkeit von $(\mathcal{K}_\Gamma, [h]) \models B$ gezeigt und damit $(\mathcal{K}_\Gamma, [g]) \models \Box B$

(2) Enthält Γ n Formeln, so kann es höchstens 2^n Äquivalenzklassen von \sim_Γ geben. ■

Satz 7.14

Die modale Aussagenlogik **K** ist entscheidbar, d.h. es gibt einen Algorithmus, der für jede Formel A entscheidet, ob A eine **K**-Tautologie ist oder nicht.

Beweis: Sei Γ die endliche Menge aller Teilformeln von $\neg A$. Wenn $\neg A$ überhaupt erfüllbar ist, dann nach Lemma 7.13 auch in einer Kripke Struktur mit höchstens 2^n Welten, wobei n die Kardinalität von Γ ist. Alle Kripke Strukturen mit höchstens 2^n Welten lassen sich endlich aufzählen, und es ist für jede dieser Strukturen entscheidbar, ob $\neg A$ in ihr wahr ist oder nicht. ■

7.5 Übungsaufgaben

Übungsaufgabe 7.5.1

Zeigen Sie die folgenden Behauptungen:

1. In der Klasse aller reflexiven Kripke-Strukturen sind allgemeingültig:
 $A \rightarrow \Diamond A$
2. In der Klasse aller transitiven Kripke-Strukturen sind allgemeingültig:
 $\Diamond \Diamond A \rightarrow \Diamond A$
3. In der Klasse aller symmetrischen Kripke-Strukturen sind allgemeingültig:
 $\Diamond \Box A \rightarrow A$.
4. In der Klasse aller transitiven und symmetrischen Kripke-Strukturen sind allgemeingültig:
 $\Diamond \Box A \rightarrow \Box A$

Übungsaufgabe 7.5.2

Die in Definition 7.8 definierte Folgerungsbeziehung wird genauer die *lokale* Folgerungsbeziehung genannt. Für die Zwecke dieser Übungsaufgabe schreiben wir deutlicher $\Gamma \vdash^L A$ anstelle von $\Gamma \vdash A$. Es gibt noch eine zweite Möglichkeit eine Folgerungsbeziehung, die *globale* Folgerungsbeziehung, zu definieren.

Definition 7.15

Sei A eine Formel und Γ eine Menge von Formeln der modalen Aussagenlogik.

A ist eine globale logische Folgerung aus Γ , in Symbolen $\Gamma \vdash^G A$, gdw für alle Kripke-Strukturen \mathcal{K} gilt:

wenn für jede Welt s von \mathcal{K} gilt $(\mathcal{K}, s) \models \Gamma$
dann gilt für jede Welt s auch $(\mathcal{K}, s) \models A$

1. Zeigen Sie, daß $\emptyset \vdash^L A$ genau dann gilt wenn $\emptyset \vdash^G A$ gilt.

2. Zeigen Sie, daß aus $\Gamma \vdash^L A$ stets $\Gamma \vdash^G A$ folgt.
3. Finden Sie ein Beispiel, so daß $\Gamma \vdash^G A$ gilt aber nicht $\Gamma \vdash^L A$.

Übungsaufgabe 7.5.3

Sei $\mathcal{K} = (S, R, I)$ eine Kripke-Struktur und $T \subseteq S$ eine Teilmenge von S mit der folgenden Abschlußeigenschaft

$$\text{für alle } s_1, s_2 \text{ mit } s_1 \in T \text{ und } R(s_1, s_2) \text{ gilt } s_2 \in T$$

Wir definieren eine Teilstruktur $\mathcal{K}_T = (T, R_T, I_T)$ von \mathcal{K} wobei R_T und I_T die Einschränkungen von R und I auf die Teilmenge T sind, d.h. für $s_1, s_2 \in T$ gilt:

$$\begin{aligned} R_T(s_1, s_2) & \text{ gdw } R(s_1, s_2) \\ I_T(P, s_1) & \text{ gdw } I(P, s_1) \end{aligned}$$

für jedes aussagenlogische Atom P .

Zeigen Sie für jedes $s \in T$ und für jede modallogische Formel A

$$(\mathcal{K}, s) \models A \text{ gdw } (\mathcal{K}_T, s) \models A$$

Wir geben zur Illustration ein Beispiel für eine Teilmenge $T \subseteq S$ mit der gewünschten Abschlußeigenschaft. Sei dazu $s_0 \in S$. Dann sei S_0 die Menge der von s_0 aus via R erreichbaren Welten:

$$S_0 = \{s \in S \mid \text{es gibt } n \text{ und es gibt } s_i \text{ für } 0 \leq i \leq n \\ \text{mit } R(s_i, s_{i+1}) \text{ für } 0 \leq i < n \text{ und } s_n = s\}$$

Übungsaufgabe 7.5.4

Ist A eine modallogische Formel so stehe $\Box^n A$ für die Formel $\underbrace{\Box \dots \Box}_n A$ und

A^* für die Formelmengemenge $\{\Box^n A \mid 0 < n\}$

Zeigen Sie:

$$A \vdash^G B \text{ gdw } A^* \vdash^L B$$

Übungsaufgabe 7.5.5

Definition 7.16

Sei A eine modallogische Formel. Die *negationsduale* Formel B zu A wird erhalten, in dem jedes aussagenlogische Atom p in A durch $\neg p$ ersetzt wird.

Zeigen Sie: Ist A eine charakterisierende Formel für die Klasse \mathbf{R} von Rahmen, dann ist auch die negationsduale Formel B eine charakterisierende Formel für \mathbf{R} .

Übungsaufgabe 7.5.6

Bestimmen Sie die negationsdualen Formeln zu

1. $\Box A \rightarrow A$
2. $\Box A \rightarrow \Box \Box A$
3. $A \rightarrow \Box \Diamond A$
4. $\Diamond A \rightarrow \Box A$

Übungsaufgabe 7.5.7

Wir nennen einen Kripke-Rahmen (S, R) *Euklidisch* wenn er die Formel $\forall x \forall y \forall z (R(x, y) \wedge R(x, z) \rightarrow R(y, z))$ erfüllt.

Beweisen Sie, daß die modallogische Formel $\Diamond p \rightarrow \Box \Diamond p$ die Klasse der Euklidischen Rahmen charakterisiert.

Übungsaufgabe 7.5.8

Wir nennen einen Kripke-Rahmen (S, R) *schwach konfluent* wenn er die Formel $\forall x \forall y ((\exists z (R(x, z) \wedge R(y, z))) \rightarrow \exists z (R(x, z) \wedge R(y, z)))$ erfüllt.

Finden Sie eine modallogische Formel, welche die Klasse der schwach konfluenten Rahmen charakterisiert.

Offensichtlich ist jede reflexive, symmetrische Relation R auch schwach konfluent, die Umkehrung muß jedoch bei weitem nicht gelten.

Kapitel 8

Temporale Logik

In den bisher betrachteten Logiken wird davon ausgegangen, daß die Wahrheit oder Falschheit von Aussagen unveränderlich ist. Besonders augenfällige Situationen, in denen diese Annahme nicht zutrifft, sind Situationen, die, vielleicht sogar sehr kurzfristigen, zeitlichen Änderungen unterworfen sind. Temporale Logiken, die wir in diesem Kapitel behandeln werden, sind speziell darauf zugeschnitten. Es gibt sehr viele Varianten temporaler Logiken. Die größte Bedeutung haben aussagenlogische Temporallogiken. Wir wollen uns hier mit einem einfachen Beispiel begnügen, mit der linearen temporalen Logik, LTL. Eine umfassendere Darstellung findet man in dem Skriptum P. SCHMITT „Nichtklassische Logiken“.

Die temporale Logik LTL, die wir betrachten wollen, ist eine spezielle modale Logik. Das gesamte Rahmenwerk aus dem vorangegangenen Kapitel 7.1 wird wieder benutzt, insbesondere wird es in LTL wieder modale Operatoren geben, sogar mehr als bisher, und zur Erklärung der Semantik werden wieder Kripke-Strukturen $\mathcal{K} = (S, R, I)$ benutzt. Im allgemeinen werden für die Semantik von LTL sogenannte *Zeitstrukturen* verwendet.

Definition 8.1

Eine Zeitstruktur ist von der Form $\mathcal{T} = (S, R, I)$, wobei R eine strikte Quasiordnung (siehe Def.1.1 Punkt 1.1) ist. Wir schreiben dann auch suggestiver $<$ anstelle von R und nennen die Elemente s der Menge S Zeitpunkte.

Mit Hilfe von Zeitstrukturen lässt sich auch ein kontinuierlicher Zeitverlauf modellieren. Als abstrakte Zeitpunkte kann man z.B. die Menge der reellen oder rationalen Zahlen benutzen. Zeitstrukturen müssen nicht notwendigerweise einen Anfangspunkt haben. Das ist hilfreich, wenn man nicht nur unbegrenzt viele zukünftige Zeitpunkte, sondern auch unbegrenzt viele Zeitpunkte in der Vergangenheit betrachten möchte.

Wir sagen Zeitpunkt t ist ein unmittelbarer Nachfolger des Zeitpunktes s , in Zeichen $s \prec t$, wenn $s < t$ gilt und keinen Zeitpunkt s_0 existiert mit $s < s_0 < t$. In Zeitstrukturen muß nicht jeder Zeitpunkt einen unmittelbaren Nachfolger besitzen.

Omega-Strukturen sind der Spezialfall von Zeitstrukturen, in denen die Quasiordnung (S, R) gleich $(\mathbb{N}, <)$ ist. Details werden in der nachfolgenden Definition 8.3 gegeben.

8.1 Lineare Temporale Logik (LTL)

Definition 8.2 (LTL-Formeln)

Sei Σ eine Menge aussagenlogischer Atome. Die Menge $LTLFor$ (bzw. genauer $LTLFor_{\Sigma}$ falls erforderlich) der LTL-Formeln ist definiert durch

1. $\Sigma \subseteq LTLFor$
2. $\mathbf{1}, \mathbf{0} \in LTLFor$
3. Liegen A, B in $LTLFor$, dann auch alle aussagenlogischen Kombinationen von A und B .
4. für $A, B \in LTLFor$ gilt auch
 - (a) $\Box A \in LTLFor$ und
 - (b) $\Diamond B \in LTLFor$ und
 - (c) $A \mathbf{U} B \in LTLFor$
 - (d) $X A$

Die Symbole \Box , \Diamond , X und \mathbf{U} heißen temporale Modaloperatoren. Die Notation für die temporalen Modaloperatoren ist leider nicht einheitlich. So finden man auch die folgenden Varianten

$G A$	für	$\Box A$
$F A$	für	$\Diamond A$
$A \text{ until } B$	für	$A \mathbf{U} B$
$\circ A$	für	$X A$

Für die Semantik von LTL benutzen wir *omega-Strukturen*.

Definition 8.3

Eine *omega-Struktur* $\mathcal{R} = (\mathbb{N}, <, \xi)$ für eine aussagenlogische Signatur P besteht aus der geordneten Menge der natürlichen Zahlen

$$(\mathbb{N}, <)$$

interpretiert als Menge abstrakter Zeitpunkte und einer Funktion

$$\xi : \mathbb{N} \rightarrow 2^P$$

mit der Intention

$$p \in \xi(n) \Leftrightarrow \text{in } \mathcal{R} \text{ ist } p \text{ zum Zeitpunkt } n \text{ wahr}$$

Für $\xi : \mathbb{N} \rightarrow 2^P$ und $n \in \mathbb{N}$ steht ξ_n für das bei n beginnende Endstück von ξ . In Zeichen

$$\xi_n(m) = \xi(n + m)$$

Inbesondere gilt $\xi_0 = \xi$.

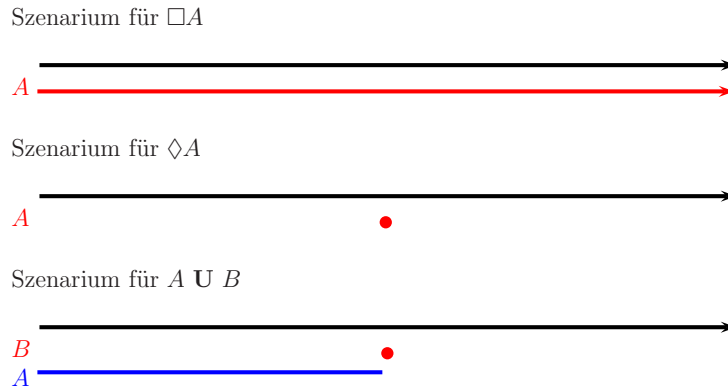


Abbildung 8.1:

Definition 8.4 (Semantik von LTL-Formeln)

Sei $\mathcal{R} = (\mathbb{N}, <, \xi)$ eine omega-Struktur und A eine *LTL* Formel.

$\xi \models p$	gdw	$p \in \xi(0)$	(p ein AL Atom)
$\xi \models op(A, B)$		für aussagenlogische Kombinationen $op(A, B)$ von A und B wie üblich	
$\xi \models \Box A$	gdw	für alle $n \in \mathbb{N}$ gilt $\xi_n \models A$	
$\xi \models \Diamond A$	gdw	es gibt ein $n \in \mathbb{N}$ mit $\xi_n \models A$	
$\xi \models A \cup B$	gdw	es gibt $n \in \mathbb{N}$ mit $\xi_n \models B$ und für alle m mit $0 \leq m < n$ gilt $\xi_m \models A$	
$\xi \models X A$	gdw	$\xi_1 \models A$	

Beispiel 8.5

$\xi \models \Diamond X p$	gdw	$\xi_n \models X p$	für ein $n \in \mathbb{N}$
	gdw	$(\xi_n)_1 \models p$	für ein $n \in \mathbb{N}$
	gdw	$\xi_{n+1} \models p$	für ein $n \in \mathbb{N}$

$\xi \models X \Diamond p$	gdw	$\xi_1 \models \Diamond p$	
	gdw	$(\xi_1)_n \models p$	für ein $n \in \mathbb{N}$
	gdw	$\xi_{1+n} \models p$	für ein $n \in \mathbb{N}$

Die beiden Beispiele zeigen auch, daß $\Diamond X p \leftrightarrow X \Diamond p$ in allen omega-Strukturen wahr ist.

In Abbildung 8.1 ist der Versuch zu sehen, die Bedeutung der temporalen Operatoren graphisch darzustellen.

Das nächste Lemma zeigt, daß die beiden Operatoren \Box und \Diamond in der Definition 8.2 hätten eingespart werden können.

Lemma 8.6

Die folgenden Äquivalenzen gelten in allen Zeitstrukturen:

$$\begin{aligned} \Diamond A &\leftrightarrow \mathbf{1} \mathbf{U} A \\ \Box A &\leftrightarrow \neg(\mathbf{1} \mathbf{U} \neg A) \end{aligned}$$

Beweis: Einfach. ■

Dieses Lemma zeigt, daß man in LTL allein mit dem Operator \mathbf{U} auskommen kann. In der Literatur treten zusätzlich auch die Operatoren \mathbf{U}_w und \mathbf{V} auf, deren Semantik wir uns kurz anschauen wollen. \mathbf{U}_w heißt der *schwache until-Operator* während für \mathbf{V} kein gebräuchlicher Name existiert.

Definition 8.7 (Zusätzliche Operatoren)

$\xi \models A \mathbf{U}_w B$ gdw für alle $n \in \mathbb{N}$ gilt $\xi_n \models (A \wedge \neg B)$ oder es gibt $n \in \mathbb{N}$ mit $\xi_n \models B$ und für alle m mit $0 \leq m < n$ gilt $\xi_m \models A$

$\xi \models A \mathbf{V} B$ gdw $\xi \models B$ und für alle $n \in \mathbb{N}$ gilt falls $\xi_n \models \neg B$ dann gibt es ein m mit $0 \leq m < n$ und $\xi_m \models A$

Siehe Abbildung 8.2 für eine Visualisierung des temporalen Operators \mathbf{V} .

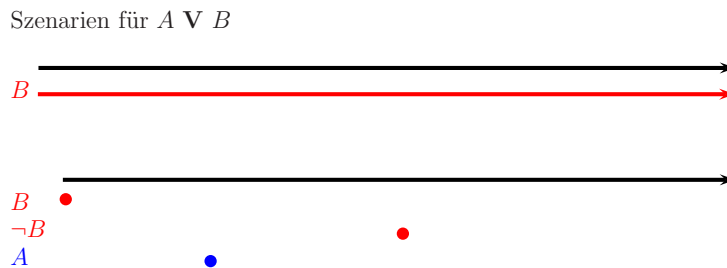


Abbildung 8.2:

Lemma 8.8

Für alle omega-Strukturen gilt:

1. $A \mathbf{U} B \leftrightarrow A \mathbf{U}_w B \wedge \diamond B$
2. $A \mathbf{U}_w B \leftrightarrow A \mathbf{U} B \vee \square(A \wedge \neg B)$
3. $A \mathbf{V} B \leftrightarrow \neg(\neg A \mathbf{U} \neg B)$
4. $A \mathbf{U} B \leftrightarrow (B \vee (A \wedge X(A \mathbf{U} B)))$
5. $A \mathbf{V} B \leftrightarrow (B \wedge A) \vee (B \wedge X(A \mathbf{V} B))$

Beweise:

zu 1 Folgt direkt aus den Definitionen.

zu 2 Einfach.

zu 3 Wir inspizieren die rechte Seite. Es gibt zwei Möglichkeiten, daß eine \mathbf{U} -Aussage in einer omega-Struktur ξ fehlschlagen kann. Erstens, wenn zu keinem Zeitpunkt das zweite Argument wahr wird und zweitens, wenn es zwar einen Zeitpunkt gibt, zu dem das zweite Argument wahr ist, aber für jeden solchen Zeitpunkt t zwischen jetzt und t ist nicht immer das erste Argument wahr. Für die vorliegende Formel ergeben sich die beiden folgenden Möglichkeiten

1. für alle n gilt $\xi_n \models B$
2. für alle mit n mit $\xi_n \models \neg B$ gibt es ein m mit $0 \leq m < n$ und $\xi_m \models A$.

Somit gilt auf jeden Fall $\xi_0 = \xi \models B$. Im ersten Fall ist das klar. Im zweiten Fall würde $\xi_0 \models \neg B$ zu dem Widerspruch führen, daß ein m existieren soll mit $0 \leq m < 0$. Damit ist die erste Forderung von $\xi \models A \mathbf{V} B$ gezeigt. Der zweite Teil der Definition von $\xi \models A \mathbf{V} B$ folgt unmittelbar.

Der Beweis der umgekehrten Implikation ist leicht nachzuvollziehen.

zu 4 Wir betrachten ein ξ mit $\xi \models A \mathbf{U} B$. Wir unterscheiden die beiden Fälle $\xi \models B$ und $\xi \models \neg B$. Im ersten Fall kann man nichts weiter schließen, $\xi \models B$ reicht ja schon aus um $\xi \models A \mathbf{U} B$ wahr zu machen. Im zweiten Fall muß auf jeden Fall $\xi \models A$ gelten und $\xi_1 \models A \mathbf{U} B$, was gleichbedeutend mit $\xi \models X (A \mathbf{U} B)$. Das erledigt die Implikatin von links nach rechts. Die inverse Implikation folgt ebenso einfach.

zu 5 Nehmen wir zunächst $\xi \models A \mathbf{V} B$ an. Falls $\xi \models B \wedge X (A \mathbf{V} B)$ gilt dann gilt trivialerweise die recht Seite. Betrachten wir also den Fall $\xi \models \neg(B \wedge X (A \mathbf{V} B))$. Das heißt $\xi \models \neg B \vee \neg X (A \mathbf{V} B)$. Nach Annahme gilt $\xi \models A \mathbf{V} B$, woraus auf jeden Fall $\xi \models B$ folgt. Unsere Annahme reduziert sich damit auf $\xi \models \neg X (A \mathbf{V} B)$. Das kann aus zwei Gründen passieren

1. $\xi_1 \models \neg B$
2. $\xi_1 \models B$, es gibt $n > 1$ mit $\xi_n \models \neg B$ und es gibt kein $1 \leq m < n$ mit $\xi_m \models A$.

Im ersten Fall muß es wegen $\xi \models A \mathbf{V} B$ ein j geben mit $0 \leq j < 1$ und $\xi_j \models A$. Für j bleibt also nur $j = 0$, d.h. $\xi \models A$. Im zweiten Fall folgt wieder aus $\xi \models A \mathbf{V} B$ die Existenz eines j mit $0 \leq j < n$ mit $\xi_j \models A$. Wieder bleibt unter den Annahmen dieses Falles $j = 0$ als einzige Möglichkeit. Da schließlich noch $\xi \models B$ aus $\xi \models A \mathbf{V} B$ folgt haben wir tatsächlich $\xi \models (B \wedge A)$ gezeigt.

Gelte jetzt die rechte Seite $\xi \models (B \wedge A) \vee (B \wedge X (A \mathbf{V} B))$. Es ist einfach zu sehen, daß sowohl aus $\xi \models (B \wedge A)$ als auch aus $\xi \models (B \wedge X (A \mathbf{V} B))$ folgt $\xi \models A \mathbf{V} B$.

■

8.1.1 Übungsaufgaben

Übungsaufgabe 8.1.1

Wir betrachten die folgende Definition eines neuen temporalen Operators \mathbf{U}^+ :

$\xi \models A \mathbf{U}^+ B$ gdw es gibt $n \in \mathbb{N}$ mit $0 < n$ und $\xi_n \models B$ und für alle m mit $0 < m < n$ gilt $\xi_m \models A$

1. Drücken Sie \mathbf{U}^+ im Vokabular von LTL aus.
2. Definieren Sie \mathbf{U} und X durch \mathbf{U}^+ .

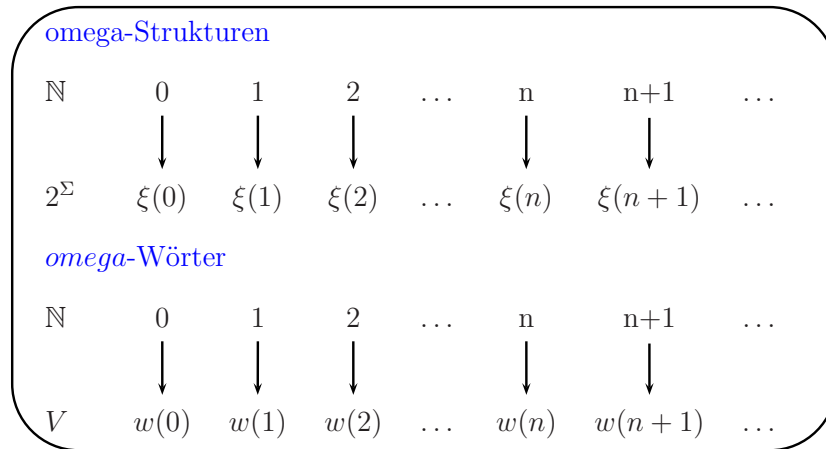


Abbildung 8.3: Korrespondenz zwischen omega-Strukturen und omega-Wörtern

Übungsaufgabe 8.1.2

Sei p ein aussagenlogisches Atom. Geben Sie eine LTL-Formel A_{2p} , so daß für jedes ξ gilt

$$\xi \models A_{2p} \quad \text{gdw} \quad (p \in \xi(n) \Leftrightarrow n \text{ ist gerade})$$

Übungsaufgabe 8.1.3

Zeigen Sie, daß die folgenden LTL-Formeln in allen Omega-Strukturen wahr sind

1. $\neg(B \mathbf{U} C) \leftrightarrow \neg C \mathbf{U}_w (\neg C \wedge \neg B)$
2. $\neg(B \mathbf{U}_w C) \leftrightarrow \neg C \mathbf{U} \neg B$
3. $B \mathbf{U} C \leftrightarrow C \vee (B \wedge X (B \mathbf{U} C))$
4. $\diamond B \leftrightarrow (B \vee X \diamond B)$
5. $\square B \leftrightarrow (B \wedge X \square B)$

Übungsaufgabe 8.1.4

Definition 8.9

Sei ξ eine omega-Struktur zur aussagenlogischen Signatur Σ . Ein *Stotter-schritt* ist ein Paar von aufeinander folgenden Zeitpunkten, $n, n+1$, welche dieselben Atome aus Σ erfüllen, i.e. für alle $p \in \Sigma$ gilt $\xi(n) \models p \Leftrightarrow \xi(n+1) \models p$

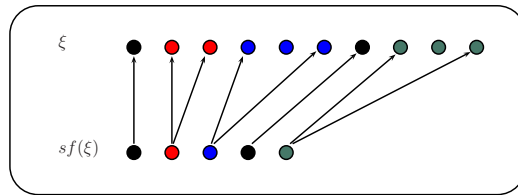


Abbildung 8.4: Die Abbildung an

p . Das ist nach Definition 8.3 gleichbedeutend mit $\xi(n) = \xi(n+1)$. Mit $sf(\xi)$ bezeichnen wir die omega-Struktur, die aus ξ entsteht, indem alle Stotter-schritte entfernt werden. Wir müssen allerdings eine Ausnahme machen für den Fall, daß von einer Stelle an in ξ nur noch Stotter-schritte auftreten. Diese bleiben dann erhalten.

In Formeln heißt das: Es gibt eine Funktion $an : \mathbb{N} \rightarrow 2^{\mathbb{N}}$, so daß für alle n gilt

1. für $n_1 < n_2$ gilt für alle $k_1 \in an(n_1)$, $k_2 \in an(n_2)$ stets $k_1 < k_2$
2. für alle n und alle $k_1, k_2 \in an(n)$ gilt $\xi(k_1) = \xi(k_2) = sf(\xi)(n)$,
3. für alle n für die $an(n)$ endlich ist gilt für alle $k_1 \in an(n)$, $k_2 \in an(n+1)$ stets $\xi(k_1) \neq \xi(k_2)$,
4. für alle n , so daß $an(n)$ unendlich ist gilt $an(n) = an(m)$ für alle $n \geq m$,
5. $\bigcup_{n \in \mathbb{N}} an(n) = \mathbb{N}$

Zwei omega-Strukturen ξ_1 und ξ_2 heißen *stotteräquivalent* falls $sf(\xi_1) = sf(\xi_2)$ gilt.

Zeigen Sie, daß für zwei stotteräquivalent omega-Strukturen ξ_1 , ξ_2 und jede temporallogische Formel F , in der der Operator X nicht vorkommt, gilt:

$$\xi_1 \models F \quad \text{gdw} \quad \xi_2 \models F$$

Man sagt in diesem Fall, F ist eine stotterinvariante Formel.

Übungsaufgabe 8.1.5

Zeigen Sie, daß die Äquivalenzen 1 bis 3 aus Lemma 8.8 auch für alle Zeitstrukturen gelten.

Übungsaufgabe 8.1.6

In der Definition 8.4 von $\xi \models A$ auf Seite 263 wird in den rekursiven Schritten die omega-Struktur ξ geändert, sie wird z.B. ersetzt durch ξ_n . Eine Alternative dazu wäre die Definition von $(\xi, n) \models A$ mit der intuitiven Bedeutung, die Formel A ist wahr zum Zeitpunkt n in der omega-Struktur ξ . Wir haben uns für die Textversion entschieden, weil sie geringfügig weniger Schreibaufwand verursacht. Wie würde die rekursive Definition für die Alternative $(\xi, n) \models A$ aussehen?

Übungsaufgabe 8.1.7

Finden Sie eine LTL Formel F , die genau dann in einer omega-Struktur ξ wahr ist, wenn für jeden Zeitpunkt t_0 , in dem p wahr ist, gilt:

1. Es gibt einen Zeitpunkt t_1 an dem q wahr ist und zwischen t_0 und t_1 ist q nicht wahr, und
2. es gibt einen Zeitpunkt t_1 an dem r wahr ist und zwischen t_0 und t_2 ist r nicht wahr, und
3. $t_1 < t_2$ ist.

Übungsaufgabe 8.1.8

Finden Sie eine LTL Formel F , die genau dann in einer omega-Struktur ξ wahr ist, wenn gilt

1. nach jedem Zeitpunkt t_0 , in dem p gilt, kommt ein Zeitpunkt t_1 mit $t_0 < t_1$, in dem q wahr ist und
2. gilt ein einem Zeitpunkt t_0 die Aussage p , dann gibt es bis zum nächsten Zeitpunkt t_1 , $t_0 < t_1$ zu dem q wahr ist, genau einen Zeitpunkt, in dem r wahr ist.

Kapitel 9

Stärkere Logiken

9.1 Prädikatenlogik zweiter Ordnung

Bemerkung 9.1

Die Prädikatenlogik der zweiten Ordnung, kurz PL2, unterscheidet sich dadurch von der PL1, daß es nun auch Variable (über die man quantifizieren kann) für Funktionen und Prädikate gibt. In der vollen Ausbaustufe hätte man also zu jeder Stelligkeit von Prädikaten unendlich viele Prädikatenvariable, und entsprechend Funktionsvariable. Zusätzlich kann die Signatur fest zu interpretierende Symbole für Operatoren verschiedenen Typs enthalten: z. B. für Funktionale, die Funktionen auf Funktionen abbilden; für Relationen (bestimmter Stelligkeit) über Relationen (bestimmter Stelligkeiten) und/oder Funktionen (bestimmter Stelligkeiten), etc.. Wir werden das hier nicht weiter ausführen. Es zeigt sich nämlich, daß die kennzeichnenden Phänomene der zweiten Stufe bereits bei einem eher bescheidenen sprachlichen Rahmen auftreten (und trivialerweise bei Erweiterung erhalten bleiben).

Wir beschränken uns im folgenden auf Sprachen zweiter Ordnung, bei denen zusätzlich zur Ausstattung der PL1 noch Variable für einstellige und für zweistellige Prädikate (Relationen) vorhanden sind. (Insbesondere gibt es keine Funktionsvariablen und keine Signatursymbole „von höherem Typ“.)

Definition 9.2

(Syntax der PL2 (in der hier betrachteten Ausbaustufe))

Sonderzeichen:

Wie in der PL1, also: $(,), \doteq, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists$.

Variable:

$Var = Ivar \cup Mvar \cup Rvar$ (paarweise disjunkt)

Ivar: unendlich viele *Individuenvariable* v_0, v_1, \dots
Notation: x, y, z, \dots

Mvar: unendlich viele *Mengenvariable* oder
einstellige Prädikatvariable M_0, M_1, \dots
Notation: X, Y, Z, \dots

Rvar: unendlich viele *Relationenvariable* oder
zweistellige Prädikatvariable R_0, R_1, \dots
Notation: U, V, \dots

Signatur

$\Sigma = (F_\Sigma, P_\Sigma, \alpha_\Sigma)$ wie in der PL1

Terme

$Term_\Sigma$ wie in der PL1

atomare Formeln:

$s \doteq t$ für Terme s, t

$p(t_1, \dots, t_n)$ für $p \in P_\Sigma$, $\alpha_\Sigma(p) = n$, $t_1, \dots, t_n \in \text{Term}_\Sigma$

$X(t)$ für Mengenvariable X und Terme t

$U(s, t)$ für Relationenvariable U und Terme s, t

Formeln

For_Σ^2 (kurz For_Σ oder For) enthält genau

- alle atomaren Formeln
- true, false
- mit $A, B \in For_\Sigma^2$, $x \in Ivar$, $X \in Mvar$, $U \in Rvar$
auch: $\neg A$, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, $(A \leftrightarrow B)$, $\forall xA$,
 $\exists xA$, $\forall XA$, $\exists XA$, $\forall UA$, $\exists UA$

Die Begriffe „freie Variable“, „gebundene Variable“, „Substitution“, „kollisionsfreie Substitution“, „Präfix“, „Allabschluß“, „Existenzabschluß“ u.ä. werden entsprechend der PL1 gebildet.

Definition 9.3

(Semantik der PL2 (in der hier betrachteten Ausbaustufe))

Zu einer Interpretation (D, I) (wie in der PL1) sind jetzt neben Belegungen $\beta : Ivar \rightarrow D$ auch Belegungen $\gamma : Mvar \rightarrow P(D)$ und $\delta : Rvar \rightarrow P(D \times D)$ zu betrachten (P : Potenzmenge). Zu $D, I, \beta, \gamma, \delta$ ist $val_{D, I, \beta, \gamma, \delta}$ entsprechend dem Vorgehen in der PL1 definiert.

Auf Termen: $val_{D, I, \beta, \gamma, \delta}(t) = val_{D, I, \beta}(t)$ wie in der PL1 (hängt nicht ab von γ, δ).

Auf Formeln:

$$\left. \begin{array}{l} val_{D, I, \beta, \gamma, \delta}(p(t_1, \dots, t_n)) \\ val_{D, I, \beta, \gamma, \delta}(s \doteq t) \end{array} \right\} \begin{array}{l} \text{Wie in der PL1} \\ \text{(Hängt nicht von } \gamma, \delta \text{ ab)} \end{array}$$

$$\begin{array}{ll} val_{D, I, \beta, \gamma, \delta}(X(t)) = W & \Leftrightarrow val_{D, I, \beta, \gamma, \delta}(t) \in \gamma(X) \\ val_{D, I, \beta, \gamma, \delta}(U(s, t)) = W & \Leftrightarrow (val_{D, I, \beta, \gamma, \delta}(s), val_{D, I, \beta, \gamma, \delta}(t)) \in \delta(U) \end{array}$$

$$\left. \begin{array}{l} val_{D, I, \beta, \gamma, \delta} \text{ auf } \underline{\text{true}}, \underline{\text{false}}, \\ \neg A, (A \wedge B), (A \vee B), (A \rightarrow B), (A \leftrightarrow B), \forall xA, \exists xB \end{array} \right\} \text{Wie in der PL1}$$

$$val_{D, I, \beta, \gamma, \delta}(\forall XA) = W \Leftrightarrow \text{Für jede Modifikation } \gamma_X^M \text{ von } \gamma \text{ gilt } val_{D, I, \beta, \gamma_X^M, \delta}(A) = W$$

$$val_{D,I,\beta,\gamma,\delta}(\forall UA) = W \Leftrightarrow \text{Für jede Modifikation } \delta_U^R \text{ von } \delta \text{ gilt } val_{D,I,\beta,\gamma,\delta_U^R}(A) = W$$

Existenzoperatoren: Entsprechend mit „Es gibt ...“.
(Modifikationen sind entsprechend der Begriffsbildung bei Belegungen von Individuenvariablen definiert.)

(D, I) heißt *Modell* von $A : \Leftrightarrow val_{D,I,\beta,\gamma,\delta}(A) = W$ für alle β, γ, δ . (D, I) ist Modell einer Formelmengemenge $M : \Leftrightarrow (D, I)$ ist Modell jeder Formel in M .

$M \models A : \Leftrightarrow$ Jedes Modell von M ist Modell von A

A *allgemeingültig* : $\Leftrightarrow \models A$ (d. h. $\emptyset \models A$)

A *erfüllbar* : $\Leftrightarrow \neg A$ ist nicht allgemeingültig.

(D.h.: Es gibt $D, I, \beta, \gamma, \delta$ mit $val_{D,I,\beta,\gamma,\delta}(A) = W$)

Bemerkung 9.4

Auf der zweiten Stufe kann es Formeln geben, in denen kein Signatursymbol und auch nicht \doteq auftritt, z. B. $X(y)$. Ist eine solche Formel geschlossen, dann hängt ihr Wahrheitswert bei einer Interpretation (D, I) (nicht nur nicht von Belegungen, sondern auch) höchstens von D , nicht von I ab. Z.B. ist $\exists X \exists y X(y)$ allgemeingültig, $\forall X \exists y X(y)$ unerfüllbar, $\exists X (\exists y X(y) \wedge \exists y \neg X(y))$ gilt genau bei den (D, I) mit $\#D \geq 2$.

Satz 9.5

In der PL2 ist die Gleichheit durch eine Formel charakterisierbar.

Beweis

Mit $G := \forall X (X(x) \leftrightarrow X(y))$ gilt $\models x \doteq y \leftrightarrow G$.

Satz 9.6

Durch das Peano'sche Axiomensystem $P1, \dots, P5$ in der PL2 sind die natürlichen Zahlen mit Addition und Multiplikation bis auf Isomorphie eindeutig charakterisiert.

Beweis

Das allgemeine Induktionsschema

$$\forall X ((X(O) \wedge \forall y (X(y) \rightarrow X(S(y)))) \rightarrow \forall y X(y))$$

hat genau die durch O und S termerzeugten Interpretationen zu Modellen.

Übungsaufgabe 9.1.1

Über einer endlichen Signatur ist die Termerzeugtheit durch eine Formel der PL2 charakterisierbar. (Man orientiere sich an obigem Beispiel.)

Satz 9.7

In der PL2 ist die Endlichkeit einer Interpretation durch eine Formel charakterisierbar.

Beweis

Es gilt

(D, I) endlich

\Leftrightarrow Jede injektive Funktion $F : D \rightarrow D$ ist auch surjektiv

\Leftrightarrow Für jede Relation $R \subseteq D \times D$: Wenn R der Graph einer injektiven Funktion ist, dann ist diese auch surjektiv

Als charakterisierende Formel wählen wir

$$\begin{aligned} Fin := \forall U \quad & ((\forall x \exists y U(x, y) \wedge \forall x \forall y \forall z (U(x, y) \wedge U(x, z) \rightarrow y \doteq z) \\ & \wedge \forall x \forall y \forall z (U(x, z) \wedge U(y, z) \rightarrow x \doteq y)) \\ & \rightarrow \forall y \exists x U(x, y)) \end{aligned}$$

Satz 9.8

Die PL2 ist nicht kompakt. D. h.:

- Aus $M \models A$ braucht i. a. nicht zu folgen:
 $E \models A$ für eine endliche Teilmenge E von M
- Es kann sein, daß jede endliche Teilmenge einer Formelmenge M ein Modell hat, diese selbst aber nicht.

Beweis

Im Beweis der Nichtcharakterisierbarkeit des Begriffs „endlich“ auf der ersten Stufe (Satz 5.43) wurde von der PL1 nur verwendet, daß sie kompakt ist. Wäre die PL2 kompakt, dann ließe sich dieser Beweis wörtlich übertragen, im Widerspruch zu Satz 9.7.

Satz 9.9

Für die PL2 kann es keinen korrekten und vollständigen Kalkül geben.

Beweis

Der Begriff der Ableitbarkeit aus einem Kalkül K ist stets kompakt, d. h.

$$M \vdash_K A \Rightarrow E \vdash_K A \text{ für eine endliche Teilmenge } E \text{ von } M.$$

Die Existenz eines korrekten und vollständigen Kalküls stünde also im Widerspruch zu Satz 9.8

9.1.1 Übungsaufgaben

Übungsaufgabe 9.1.2

Gegeben sei eine Signatur Σ mit einem zweistelligen Relationszeichen r , einem einstellige Relationszeichen v und Konstantenzeichen a und b . Für eine Σ -Struktur $\mathcal{G} = (G, R, c_a, c_e, V)$, dabei sei R die Interpretation von r , ($R = I(r)$) und $I(a) = c_a$, $I(b) = c_b$, und $I(v) = V$. definieren wir:

Definition 9.10

1. Ein *R-Pfad* ist eine Folge a_1, \dots, a_n ($n \geq 1$) von Elementen von G , so daß für alle $1 \leq i < n$ gilt $R(a_i, a_{i+1})$ und $a_1 = c_a$ und $a_n = c_e$.
2. Wir nennen eine Teilmenge $V \subseteq G$ eine *Verbindungs Menge*, wenn es einen *R-Pfad* a_1, \dots, a_n gibt, so daß für alle $1 \leq i \leq n$ gilt $a_i \in V$.
3. Eine *Verbindungs Menge* heißt eine *minimale Verbindungs Menge*, wenn jede echte Teilmenge $V_0 \subset V$ keine *Verbindungs Menge* ist.

Geben Sie eine Σ -Formel ϕ_{VM} der Logik zweiter Stufe an, so daß $(G, R, c_a, c_e, V) \models \phi_{VM}$ genau dann gilt, wenn V eine *minimale Verbindungs Menge* ist.

9.2 Dynamische Logik

Bemerkung 9.11

(Modale Prädikatenlogik 1. Ordnung).

...

Bemerkung 9.12

(Dynamische Logik)

Die dynamische Logik (DL) ist eine Modallogik, jedoch mit einer Semantik, bei der für den Übergang zwischen Zuständen (Welten) nicht eine einzige, feste Relation gegeben ist, sondern gleichzeitig unendlich viele Relationen. Welche solche Relation jeweils zu wählen ist, läßt sich der auszuwertenden Formel entnehmen. Formeln können nämlich *Programme* einer bestimmten Programmiersprache enthalten, und zu jedem Programm gehört eine Übergangsrelation zwischen Zuständen. Dabei sind die Zustände in einer von der Programmiersprache (und der genaueren Aufbereitung der Semantik) abhängigen Weise eng verknüpft mit den Belegungen der Variablen. Man kann dann auf diese Weise auch reiche und komplizierte Programmiersprachen axiomatisch erfassen. Auf die Vielfalt dieser Möglichkeiten gehen wir hier nicht ein, sondern behandeln die Grundidee anhand einer besonders einfachen Programmiersprache, nämlich „elementarem PASCAL“, einer Sprache

mit Zuweisung, Verzweigung und while-Schleife. Hier kann man als Zustände einfach die Variablenbelegungen selber wählen. Auch handelt es sich um eine deterministische Sprache; in allgemeineren Versionen ist die DL auch für Sprachen mit indeterministischen Programmkonstrukten betrachtet worden. Die DL geht (in etwas anderer Version als hier dargestellt) zurück auf HARREL, sie wurde wesentlich ausgearbeitet in [Gol82].

Definition 9.13

(Syntax der DL)

Sonderzeichen: Wie in der PL1, zusätzlich $[,], <, >, \underline{\text{skip}}, \underline{\text{abort}}, :=, ;, \underline{\text{if}}, \underline{\text{then}}, \underline{\text{else}}, \underline{\text{while}}, \underline{\text{do}}$

$\left. \begin{array}{l} \text{Variable, } Var \\ \text{Signatur, } \Sigma \\ \text{Terme, } Term_{\Sigma} \\ \text{atomare Formeln} \end{array} \right\}$	wie in der PL1
--	----------------

Boole'sche Ausdrücke, $Bexp_{\Sigma}$, sind die quantorenfreien PL1-Formeln über Σ . Sie werden aus den atomaren Formeln und true, false durch die aussagenlogischen Operatoren gebildet.

Programme, $Prog_{\Sigma}$:

- skip und abort sind Programme
- Mit $x \in Var$ und $t \in Term_{\Sigma}$ ist $(x := t)$ ein Programm (Zuweisung)
- Sind π, ρ Programme und ist ϵ ein Boole'scher Ausdruck, dann sind auch Programme:
 $(\pi; \rho)$ (Hintereinanderausführung)
if ϵ then π else ρ (Verzweigung)
while ϵ do π (Schleife)

(Man beachte, daß wegen der Klammerung der Hintereinanderausführung eine schließende Klammer für Verzweigung bzw. Schleife unnötig ist.)

Formeln, For_{Σ} :

- Alle Boole'schen Ausdrücke sind Formeln
- Mit $A, B \in For_{\Sigma}, x \in Var, \pi \in Prog_{\Sigma}$ sind auch Formeln:
 $\neg A, (A \wedge B), (A \vee B), (A \rightarrow B), (A \leftrightarrow B),$
 $\forall x A, \exists x A, [\pi] A, \langle \pi \rangle A.$

Definition 9.14

(Semantik der DL)

Zu einer Interpretation (D, I) über Σ und einem Zustand β , d.h. einer Belegung der Variablen, sind $val_{D,I,\beta}(t)$ für Terme t und $val_{D,I,\beta}(\epsilon)$ für Boole'sche Ausdrücke ϵ definiert wie in der PL1.

Es sei $Sta := (Var \rightarrow D)$ die Menge der Zustände. Die *Semantik der Programme* wird festgelegt durch die Definition einer Relation $\rho(\pi) \subseteq Sta \times Sta$ für jedes $\pi \in Prog_\Sigma$. ($\rho(\dots)$ hängt von (D, I) ab, aus Gründen der einfacheren Notation schreiben wir aber einfach $\rho(\pi)$ statt $\rho(\pi)_{D,I}$.)

$\rho(\underline{abort})$ ist die leere Relation

$$\begin{aligned} \beta\rho(\underline{skip})\gamma & :\Leftrightarrow \beta = \gamma \\ \beta\rho(x := t)\gamma & :\Leftrightarrow \gamma = \beta_x^{val_{D,I,\beta}(t)} \\ \beta\rho(\pi_1; \pi_2)\gamma & :\Leftrightarrow \text{Es gibt } \delta \in Sta \text{ mit } \beta\rho(\pi_1)\delta \text{ und } \delta\rho(\pi_2)\gamma \\ \beta\rho(\underline{if} \ \epsilon \ \underline{then} \ \pi_1 \ \underline{else} \ \pi_2)\gamma & :\Leftrightarrow (val_{D,I,\beta}(\epsilon) = W \text{ und } \beta\rho(\pi_1)\gamma) \text{ oder} \\ & (val_{D,I,\beta}(\epsilon) = F \text{ und } \beta\rho(\pi_2)\gamma) \\ \beta\rho(\underline{while} \ \epsilon \ \underline{do} \ \pi)\gamma & :\Leftrightarrow \text{Es gibt } n \in \mathbb{N} \text{ und Zustände } \beta_0, \dots, \beta_n \text{ mit:} \\ & - \beta_0 = \beta, \beta_n = \gamma \\ & - \forall i \text{ mit } 0 \leq i < n : val_{D,I,\beta_i}(\epsilon) = W \text{ und } \beta_i\rho(\pi)\beta_{i+1} \\ & - val_{D,I,\beta_n}(\epsilon) = F \end{aligned}$$

Zu gegebenem π und β gibt es stets höchstens ein γ mit $\beta\rho(\pi)\gamma$: Die Relation $\rho(\pi)$ ist rechtseindeutig und hätte auch als partielle Funktion formuliert werden können. Das würde sich ändern, wenn man auch indeterministische Programme zuläßt.

$\beta\rho(\pi)\gamma$ besagt, daß π , beginnend im Zustand β , terminiert und zum Zustand γ führt. π , beginnend in β , terminiert genau dann, wenn es ein solches γ gibt. Etwa terminiert abort in keinem Zustand, ebenso while true do ρ (für beliebiges ρ). Das Programm while ϵ do skip terminiert genau bei Beginn in Zuständen β mit $val_{D,I,\beta}(\epsilon) = F$ (und dann nach 0 Schritten).

Formeln

Für Boole'sche Ausdrücke ϵ ist nach Annahme $val_{D,I,\beta}(\epsilon)$ bereits definiert.

Durch strukturelle Induktion wird $val_{D,I,\beta}$ für Formeln

$$\neg A, (A \wedge B), (A \vee B), (A \rightarrow B), (A \leftrightarrow B), \forall x A, \exists x A$$

definiert entsprechend dem Vorgehen in der PL1.

$$val_{D,I,\beta}([\pi]A) = W :\Leftrightarrow \text{Für jedes } \gamma \text{ mit } \beta\rho(\pi)\gamma \text{ gilt } val_{D,I,\gamma}(A) = W$$

$$val_{D,I,\beta}(\langle \pi \rangle A) = W :\Leftrightarrow \text{Es gibt } \gamma \text{ mit } \beta\rho(\pi)\gamma \text{ und } val_{D,I,\gamma}(A) = W$$

Man bemerke, daß (in unserem Fall von deterministischen Programmen) gilt: $val_{D,I,\beta}([\pi]A) = W$ genau dann, wenn: Falls ein γ existiert mit $\beta\rho(\pi)\gamma$, dann ist für dieses (dann eindeutig bestimmte) γ $val_{D,I,\gamma}(A) = W$.

(D, I) heißt *Modell* von A , wenn $val_{D,I,\beta}(A) = W$ für alle Zustände β . Entsprechend sind wie in der PL1 erklärt: Modell einer Formelmenge M , allgemeingültig, erfüllbar, sowie die Folgerbarkeitsbeziehung \models .

Bemerkung 9.15

(Zur Semantik der DL)

$\langle \dots \rangle$ ist entbehrlich, denn es gilt für beliebige D, I, β und beliebige π, A :

$$val_{D,I,\beta}(\langle \pi \rangle A) = val_{D,I,\beta}(\neg[\pi]\neg A),$$

d. h. $\langle \rangle$ läßt sich stets durch $\neg[\]\neg$ ersetzen.

Die *partiellen Korrektheitsaussagen der HOARE-Logik*

$A\{\pi\}B$, d.h.

„Wenn in einem Zustand A gilt, und wenn π , ab diesem Zustand rechnend, terminiert, dann gilt im erreichten Zustand B “

lassen sich in der DL wiedergeben. Es ist nämlich $A\{\pi\}B$ gleichbedeutend mit

$$A \rightarrow [\pi]B.$$

Das ist die *partielle Korrektheit* von π bzgl. der *Vorbedingung* A und der *Nachbedingung* B . Entsprechend läßt sich die *totale Korrektheit* ausdrücken:

$$A \rightarrow \langle \pi \rangle B$$

besagt (d.h. wird zu W ausgewertet unter genau denjenigen D, I, β , für welche gilt): Falls A gilt (d.h. $val_{D,I,\beta}(A) = W$), terminiert π und im dann erreichten Zustand gilt B ($val_{D,I,\gamma}(B) = W$ für dieses γ).

Insbesondere besagt $[\pi]A$ (gleichbedeutend mit $\underline{\text{true}} \rightarrow [\pi]A$), daß immer, wenn π terminiert, hernach A gilt. Und $\langle \pi \rangle A$ besagt: π terminiert immer, und hernach gilt A .

Ferner: $A \rightarrow \langle \pi \rangle \underline{\text{true}}$ besagt: Wenn A gilt (d.h. rechnend ab Zuständen, in denen A zu W ausgewertet) terminiert π . $\langle \pi \rangle \underline{\text{true}}$ besagt, daß π terminiert (d.h. $val_{D,I,\beta}(\langle \pi \rangle \underline{\text{true}}) = W \Leftrightarrow \pi$ in Interpretation (D, I) und rechnend ab β terminiert). $\neg\langle \pi \rangle \underline{\text{true}}$, d.h. $[\pi] \underline{\text{false}}$, besagt: π terminiert nicht.

Man beachte, daß die DL im Gegensatz zu den HOARE'schen Korrektheitsaussagen eine *volle* Logik ist, sie ist abgeschlossen gegenüber aussagenlogischen Operatoren, Quantoren und Programmeinfügungen.

Satz 9.16

Die natürlichen Zahlen mit Addition und Multiplikation lassen sich in der DL bis auf Isomorphie eindeutig charakterisieren.

Beweis

Die Formel

$$\forall x \langle (y := 0; \text{while } \neg y \doteq x \text{ do } y := S(y)) \rangle \text{true}$$

drückt die Termerzeugtheit durch Terme über O, S aus. Zusammen mit $Pe1, \dots, Pe4$ charakterisiert sie also das Modell (\mathbb{N}, I_0) bis auf Isomorphie eindeutig

Satz 9.17

Die DL ist nicht kompakt.

Beweis

Über einer Signatur mit zwei Konstanten a, b und einem einstelligen Funktionssymbol f bilden wir die Formeln A_0, A_1, \dots :

$$\begin{aligned} A_0 &= \neg y \doteq b \\ A_{n+1} &= [y := f(y)]A_n. \end{aligned}$$

A_n sagt über die Schleife while $\neg y \doteq b$ do $y := f(y)$, daß sie nicht nach genau n Schritten terminiert.

Ferner sei für $n = 0, 1, \dots$:

$$B_n = [y := a]A_n.$$

B_n besagt: Nach Initialisierung von y mit a hat die Schleife while $\neg y \doteq b$ do $y := f(y)$ nicht die Eigenschaft, nach genau n Schritten zu terminieren (und damit von a aus durch genau n Anwendungen von f b zu erreichen).

Schließlich sei C die Formel

$$\langle y := a; \text{while } \neg y \doteq b \text{ do } y := f(y) \rangle \text{true}.$$

C sagt, daß die Schleife bei Initialisierung von y mit a terminiert.

Jede endliche Teilmenge E von $\{B_n \mid n \in \mathbb{N}\} \cup \{C\}$ hat ein Modell, nämlich (\mathbb{N}, I) mit $I(f)(n) = n + 1$, $I(a) = 0$ und $I(b) = \max\{B_n \mid n \in \mathbb{N}\} + 1$.

Aber $\{B_n \mid n \in \mathbb{N}\} \cup \{C\}$ hat kein Modell. Denn in einem solchen, (D, I) ,

- müßte man einerseits, beginnend mit $I(a)$, durch sukzessives Anwenden von $I(f)$ das Element $I(b)$ von D in endlich vielen Schritten erreichen (Modell von C),

- andererseits gäbe es kein n , so daß man von $I(a)$ aus durch n Anwendungen von $I(f)$ das Element $I(b)$ erreicht (Modell von $\{B_n \mid n \in \mathbb{N}\}$).

Korollar 9.18

Für die DL gibt es keinen korrekten und vollständigen Kalkül

Bemerkung 9.19

(Analyse der While-Schleife)

Wie das Beispiel im Beweis des Satzes 9.17 verdeutlicht, ist die „natürliche Semantik“ der While-Schleife der Grund für die Nichtkompaktheit der DL. Wir analysieren diese Semantik etwas genauer. Das eröffnet den Zugang zu einem „infinitären Kalkül“. Bei dieser (nicht mehr „konstruktiven“) Ausweitung unseres bisherigen Kalkülbegriffs bleibt dennoch ein Teil dessen erhalten, was man sich unter der Arbeit mit einem Kalkül vorstellt.

Definition 9.20

Zu $\epsilon \in Bexp_\Sigma, \pi \in Prog_\Sigma, A \in For_\Sigma$ definieren wir für alle $n \in \mathbb{N}$ Formeln $Loop_n(\epsilon, \pi, A)$ wie folgt.

$$Loop_0(\epsilon, \pi, A) = \neg\epsilon \rightarrow A$$

$$Loop_{n+1}(\epsilon, \pi, A) = \epsilon \rightarrow [\pi]Loop_n(\epsilon, \pi, A)$$

Korollar 9.21

Zu ϵ, π, A sowie einer Interpretatin (D, I) und einem Zustand β sei β_0, β_1, \dots die eindeutig bestimmte Folge von Zuständen mit $\beta_0 = \beta, \beta_i \rho(\pi)\beta_{i+1}$. Dann gilt für jedes n :

$$val_{D,I,\beta}(Loop_n(\epsilon, \pi, A)) = W$$

$$\Leftrightarrow \text{Wenn } val_{D,I,\beta_i}(\epsilon) = W \text{ für alle } i < n \text{ und } val_{D,I,\beta_n}(\epsilon) = F, \text{ dann}$$

$$val_{D,I,\beta_n}(A) = W.$$

Beweis

Leichte Induktion nach n .

$Loop_n(\epsilon, \pi, A)$ besagt also: Wenn die Schleife while ϵ do π nach genau n Schritten terminiert, dann gilt hernach A .

Lemma 9.22

Für beliebige D, I, β gilt

$$val_{D,I,\beta}([\text{while } \epsilon \text{ do } \pi]A) = W$$

$$\Leftrightarrow \text{Für alle } n \in \mathbb{N} : val_{D,I,\beta}(Loop_n(\epsilon, \pi, A)) = W$$

Beweis

Wir zeigen: $val_{D,I,\beta}([\text{while } \epsilon \text{ do } \pi]A) = W$ gdw.: Für alle n ist das (zu $val_{D,I,\beta}(Loop_n(\epsilon, \pi, A)) = W$ äquivalente) Kriterium aus Korollar 9.21 erfüllt.

Gelte $val_{D,I,\beta}([\underline{\text{while}} \ \epsilon \ \underline{\text{do}} \ \pi]A) = W$, und sei $n \in \mathbb{N}$ gegeben. Wenn die Schleife nicht terminiert, d.h. $val_{D,I,\beta_i}(\epsilon) = W$ für alle i , dann $val_{D,I,\beta_n}(\epsilon) = W$, das Kriterium ist erfüllt. Wenn für ein m gilt $val_{D,I,\beta_m}(\epsilon) = F$ und $val_{D,I,\beta_i}(\epsilon) = W$ für alle $i < m$, dann (wegen $val_{D,I,\beta}([\underline{\text{while}} \ \epsilon \ \underline{\text{do}} \ \pi]A) = W$) $val_{D,I,\beta_m}(A) = W$. Falls $n < m$: $val_{D,I,\beta_n}(\epsilon) = W$. Falls $n > m$: Nicht gilt für alle $i < n$, daß $val_{D,I,\beta_i}(\epsilon) = W$ (für $i = m$ nicht). Falls $n = m$: $val_{D,I,\beta_n}(A) = W$. In allen drei Fällen ist also das Kriterium für n erfüllt.

Gelte nun umgekehrt $val_{D,I,\beta}([\underline{\text{while}} \ \epsilon \ \underline{\text{do}} \ \pi]A) = F$. Für das dann existierende m mit $val_{D,I,\beta_m}(\epsilon) = F$ und $val_{D,I,\beta_i}(\epsilon) = W$ für $i < m$ gilt $val_{D,I,\beta_m}(A) = F$. Für $n = m$ ist das Kriterium also nicht erfüllt.

Bemerkung 9.23

(Infinitäre Kalküle)

Aus Lemma 9.22 folgt

$$\{Loop_n(\epsilon, \pi, A) \mid n \in \mathbb{N}\} \models [\underline{\text{while}} \ \epsilon \ \underline{\text{do}} \ \pi]A.$$

Es ist also sozusagen

$$\frac{\{Loop_n(\epsilon, \pi, A) \mid n \in \mathbb{N}\}}{[\underline{\text{while}} \ \epsilon \ \underline{\text{do}} \ \pi]A}$$

für beliebiges ϵ, π, A eine „korrekte Regel“. Es ist keine Regel in unserem bisherigen Sinne, da sie unendlich viele Prämissen hätte. Wir bezeichnen sie als *infinitäre* Regel im Gegensatz zu den uns bekannten *finitären* Regeln. Eine infinitäre Regel kann nicht in der bekannten Weise zum Ableiten verwendet werden. Denn da eine Ableitung stets nur endlich lang ist, können niemals an einer bestimmten Stelle alle Prämissen der Regel produziert worden sein.

Also wird, abstrakter, zu einem „Kalkül“ K , der auch infinitäre Regeln enthalten kann, die Theoremmenge $Th_K(M)$ definiert als die kleinste Formelmengung, die M und die Axiome enthält, und die gegen die Regeln abgeschlossen ist. (Letzteres bedeutet: Wenn alle Prämissen einer Instanz der Regel in der Menge liegen, dann auch die Conclusio.) Man definiert $M \vdash_K A :\Leftrightarrow A \in Th_K(M)$.

Bemerkung 9.24

(Ein infinitärer Kalkül für die DL)

Auf der Suche nach einem korrekten und vollständigen *infinitären Kalkül* für die DL (d.h. einem solchen mit infinitären Regeln) muß man die obige infinitäre Regel noch etwas verallgemeinern, nämlich so, daß die Formeln $Loop_n(\epsilon, \pi, A)$ auch innerhalb bestimmter anderer Formeln als „Approximationen“ von $[\underline{\text{while}} \ \epsilon \ \underline{\text{do}} \ \pi]A$ verwendet werden dürfen. Z.B. wäre

$$\frac{\{B \rightarrow Loop_n(\epsilon, \pi, A) \mid n \in \mathbb{N}\}}{B \rightarrow [\underline{\text{while}} \ \epsilon \ \underline{\text{do}} \ \pi]A}$$

eine entsprechende Regel. Die Analyse des Sachverhalts liefert eine bestimmte Klasse zulässiger solcher Zusammensetzungen. Sie ist definiert durch eine Klasse \mathbf{S} von *zulässigen Schemata* oder *zulässigen Formen*. Das sind syntaktische Gebilde, die eine *Metavariablen* (einen Platzhalter) X enthalten, so daß jeweils durch Ersetzung von X durch eine Formel (an allen Stellen, und dann überall durch dieselbe Formel) eine Formel entsteht. Wir schreiben $\Phi(X)$ für ein solches Schema, und entsprechend $\Phi(A)$ für das Ersetzungsergebnis von X durch A . Die Klasse \mathbf{S} der *zulässigen Formen* ist dann definiert durch

- $X \in \mathbf{S}$
- mit $\Phi(X)$ enthält \mathbf{S} auch $B \rightarrow \Phi(X)$, $\forall x\Phi(X)$, $[\pi]\Phi(X)$ (für Formeln B , Variable x , Programme π).

Anmerkung (\mathbf{S} als Klasse *stetiger* Formen)

...

Hiermit definiert man nun die folgende infinitäre Regel, die *Omega-Iteration*:

$$OI: \frac{\{\Phi(Loop_n(\epsilon, \pi, A) \mid n \in \mathbb{N}\}}{\Phi([\underline{\text{while}} \ \epsilon \ \underline{\text{do}} \ \pi]A)} \quad \text{für } \Phi \in \mathbf{S}.$$

Es gilt nun der folgende

Satz (Goldblatt 1982)

Für die DL gibt es einen korrekten und vollständigen infinitären Kalkül, der neben OI nur finitäre Regeln enthält.

(Dabei haben wir die Axiome auch als Regeln, nämlich nullstellige Regeln, aufgefaßt.)

Es bleibt die Frage, wie man hiermit verfährt; denn „in normaler Weise ableiten“ kann man ja mit einer infinitären Regel nicht. Eine naheliegende Heuristik besteht darin, zu versuchen, die unendlich vielen Formeln $\Phi(Loop_n(\epsilon, \pi, A))$, $n \in \mathbb{N}$, durch Induktion über n zu beweisen. Zu diesem Zweck wird eine Zählerstruktur (Terme über O, S) in die Signatur fest eingebaut und die Formeln $\Phi(Loop_n(\epsilon, \pi, A))$ werden insgesamt durch eine Formel mit einer Zählervariablen wiedergegeben. Man kann ein zugehöriges Induktionsschema angeben und hat hiermit eine finitäre Approximation an eine Instanz von OI . Man erhält auf diese Weise einen korrekten finitären Kalkül. Dieser ist notwendigerweise unvollständig, jedoch durchaus leistungsfähig.

Kapitel 10

Automaten

10.1 Endliche Automaten

10.1.1 Deterministische endliche Automaten

Definition 10.1 (Endlicher Automat (finite-state-automaton))

Ein endlicher Automat ist gegeben durch

- eine endliche Menge S von Zuständen
- eine Menge VT von terminalen Zeichen
- einer Übergangsfunktion $\delta : S \times VT \rightarrow S$

Außerdem ist ein Element $s_0 \in S$ als Anfangszustand ausgezeichnet und eine nichtleere Teilmenge $S_1 \subseteq S$ als Menge von Endzuständen festgelegt.

Jeder endliche Automat EA akzeptiert eine Menge von Wörtern $L(EA)$ auf die folgende Weise:

Ein gegebenes Wort $w = a_1 \dots a_k$ denken wir uns auf das Eingabeband des EA geschrieben. Im Anfangszustand s_0 liest der Automat den ersten Buchstaben a_1 und geht in den Zustand $\delta(s_0, a_1) = s_i$ über. In diesem Zustand wird nur der zweite Buchstabe von w eingelesen und der nächste Zustand $\delta(s_i, a_2) = s_j$ erreicht. Das Wort gehört zu $L(EA)$, wird also von EA akzeptiert, wenn nach Einlesen des letzten Buchstabens a_k von w ein Endzustand aus S_1 erreicht wird.

Formal wird $L(EA)$ definiert, indem man die Übergangsfunktion δ zu einer Funktion

$$\delta : S \times VT^* \rightarrow S$$

fortsetzt.

Definition 10.2 (Fortsetzung von δ)

Der Funktionswert $\delta(s, w)$ wird induktiv über die Länge von w wie folgt definiert:

$$\begin{aligned} \delta(s, \varepsilon) &= s \\ \delta(s, aw_1) &= \delta(s', w_1) \text{ wobei } \delta(s, a) = s' \end{aligned}$$

Definition 10.3 (L(EA))

$$L(EA) = \{w \in VT^* \mid \delta(s_0, w) \in S_1\}$$

Varianten:

Gelegentlich wird in Beispielen die Übergangsfunktion nicht für alle Paare (s, a) definiert. Wird während der Abarbeitung eines Wortes w eine Situation (s, a) erreicht, für die $\delta(s, a)$ nicht definiert ist, so gilt w als nicht akzeptiert. Ein endlicher Automat, so daß $\delta(s, a)$ für alle $s \in S$ und $a \in VT$ definiert ist, heißt ein **vollständiger endlicher Automat**.

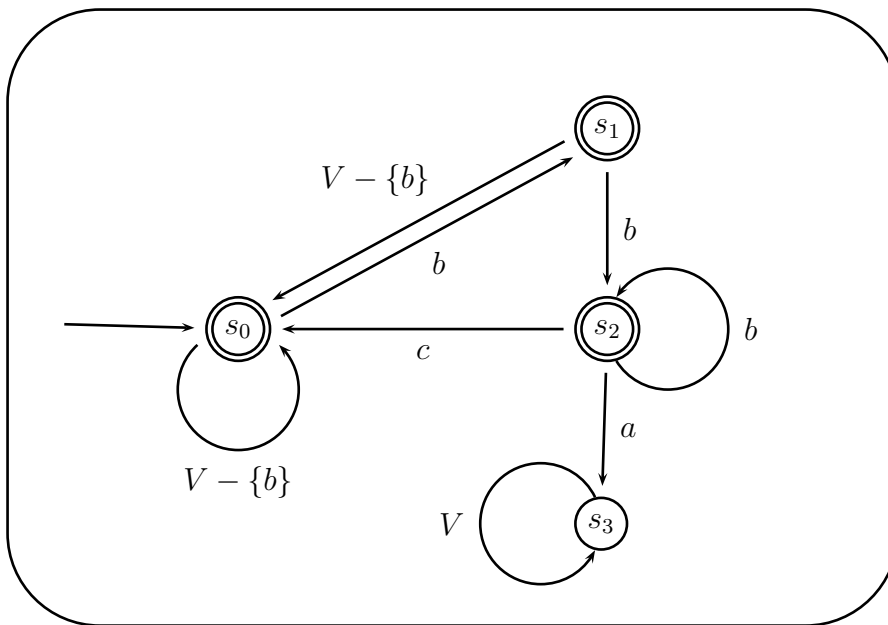


Abbildung 10.1: Der Automat N_{bba}

Beispiel 10.4

Der endliche Automat N_{bba} , siehe Abbildung 10.1, ist gegeben durch

$$S = \{s_0, s_1, s_2, s_3\}$$

$$V = \{a, b\}$$

$$S_1 = \{s_0, s_1, s_2\}$$

$$L(N_{bba}) = \{w \in \{a, b\}^* \mid bba \text{ ist kein Teilwort von } w\}$$

Beispieldurchläufe: Das Wort $bbca$ wird von N_{bba} akzeptiert, $cbbac$ nicht.

10.1.2 Nichtdeterministische endliche Automaten

Definition 10.5 (Nichtdeterministische endliche Automaten)

Ein nichtdeterministischer endlicher Automat wird durch die folgenden Bestimmungsstücke gegeben:

- eine endliche Menge S von Zuständen,
- ein Alphabet V ,
- ein Anfangszustand $s_0 \in S$,
- eine Übergangsfunktion $\delta : S \times V \rightarrow Pot(S)$,
- eine Menge $F \subseteq S$ von Finalzuständen,

Die Änderung gegenüber den deterministischen endlichen Automaten besteht also darin, daß die Übergangsfunktion δ als Werte Mengen von Zuständen annimmt:

Die Fortsetzung von δ zu einer Funktion

$$\delta : S \times V^* \rightarrow Pot(S)$$

wird jetzt wie folgt definiert:

$$\begin{aligned}\delta(s, \varepsilon) &= \{s\} \\ \delta(s, aw_1) &= \{s' : \text{es gibt } s_1 \in S \text{ mit } s_1 \in \delta(s, a) \\ &\quad \text{und } s' \in \delta(s_1, w_1)\}\end{aligned}$$

Die von einem nichtdeterministischen endlichen Automaten NEA akzeptierte Sprache $L(NEA)$ wird jetzt durch

$$L(NEA) = \{w \in V^* : \delta(s_0, w) \cap F \neq \emptyset\}$$

definiert.

Varianten: In der angegebenen Definition kann ein nichtdeterministischer endlicher Akzeptor im Zustand s , ohne einen Buchstaben zu lesen, nur nach s übergehen, $\delta(s, \varepsilon) = \{s\}$. Häufig begegnet man der Variante, die für $\delta(s, \epsilon)$ eine beliebige Teilmenge von S zuläßt. Übergänge von s zu $s_1 \in \delta(s, \varepsilon)$ nennt man dann spontane Übergänge.

Definition 10.6 (Endliche Automaten mit spontanen Übergängen)

Ein endlicher Automat mit spontanen Übergängen wird durch die folgenden Bestimmungsstücke gegeben:

- eine endliche Menge S von Zuständen,
- ein Alphabet V ,
- ein Anfangszustand $s_0 \in S$,
- eine Menge $F \subseteq S$ von Finalzuständen,
- eine Übergangsfunktion $\delta : S \times (V \cup \{\varepsilon\}) \rightarrow Pot(S)$

Um die Fortsetzung der Transitionsfunktion δ für diese Variante endlicher Automaten zu beschreiben benötigt man die folgende Hilfsfunktion:

Definition 10.7

Sei $A = (S, V, s_0, \delta, F)$ ein endlicher Automat mit spontanen Übergängen, dann ist die Funktion

$$\varepsilon-cl : Pot(S) \rightarrow Pot(S)$$

für $I \subseteq S$ definiert als:

$\varepsilon-cl(I)$ ist die kleinste Teilmenge $J \subseteq S$ mit

1. $I \subseteq J$
2. für alle $s \in J$ gilt $\delta(s, \varepsilon) \subseteq J$.

Die Bezeichnung $\varepsilon-cl$ soll an $\varepsilon-closure$ erinnern.

Die Fortsetzung von

$$\delta : S \times (V \cup \{\varepsilon\}) \rightarrow Pot(S)$$

zu

$$\bar{\delta} : S \times V^* \rightarrow Pot(S)$$

kann jetzt definiert werden als:

$$\begin{aligned}\bar{\delta}(s, \varepsilon) &= \varepsilon-cl(\delta(s, \varepsilon)) \\ \bar{\delta}(s, w_1a) &= \varepsilon-cl(\{s' : \text{es gibt } s_1 \in \bar{\delta}(s_1, w_1) \text{ so, daß } s' \in \delta(s, a)\})\end{aligned}$$

Der folgende Satz ist von grundlegender Bedeutung für die Theorie endlicher Automaten und wird auch in allen einführenden Texten behandelt. Da wir später, auch auf Details des Satzes, zurückgreifen wollen, ist er auch hier noch einmal explizit aufgeführt.

Satz 10.8

Zu jedem nichtdeterministischen endlichen Automaten $\mathcal{A} = (S, V, s_0, \delta, F)$ gibt es einen deterministischen endlichen Automaten $\mathcal{B} = (Q, V, q_0, \Delta, G)$ mit

$$L(\mathcal{A}) = L(\mathcal{B})$$

Dabei kann \mathcal{A} spontane Übergänge enthalten und muß auch nicht vollständig sein.

Beweis: Die neuen Zustände sind Teilmengen von alten Zuständen, genauer

$$\begin{aligned} Q &= \{X \subseteq S \mid \varepsilon - cl(X) = X\} \\ q_0 &= \varepsilon - cl(\{s_0\}) \\ \Delta(X, a) &= \varepsilon - cl(\bigcup_{s \in X} \delta(s, a)) \\ G &= \{X \in Q \mid F \cap X \neq \emptyset\} \end{aligned}$$

Offensichtlich ist \mathcal{B} deterministisch.

Man zeigt nun leicht, durch Induktion über die Länge des Wortes $w \in V^*$,

$$\bar{\delta}(s_0, w) = \Delta(q_0, w)$$

■

Wir beweisen noch zwei Lemmata, die einerseits für sich genommen ein Licht auf das Konzept eines endlichen Automaten werfen, andererseits als Hilfsmittel in späteren Beweisen willkommen sind.

Lemma 10.9

Zu jedem nichtdeterministischen endlichen Automaten \mathcal{A} gibt es einen nichtdeterministischen endlichen Automaten $\mathcal{B} = (Q_B, V, q_0^B, \delta_B, F_B)$ mit $L(\mathcal{A}) = L(\mathcal{B})$, so daß für alle $q \in Q_B$ gilt $q_0^B \notin \delta_B(q, x)$ für alle $x \in V$.

Mit anderen Worten, der Anfangszustand wird nur am Anfang betreten und ist dann nie wieder erreichbar. Das beinhaltet, daß es keinen Übergang von q_0^B zu sich selbst gibt.

Beweis: Wir beginnen mit $\mathcal{A} = (Q_A, V, q_0^A, \delta_A, F_A)$. Es sei q_0^B ein neuer Zustand und $Q_B = Q_A \cup \{q_0^B\}$. Unverändert bleibt $F_B = F_A$, falls $s_0^A \notin F_A$ und $F_B = F_A \cup \{q_0^B\}$ sonst. Für die Übergangsfunktion setzen wir bleibt zu nächst unverändert und $\delta_B(q, x) = \delta_A(q, x)$ für $q \in Q_A, x \in V$. Neu hinzukommt $\delta_B(q_0^B, x) = \delta_A(q_0^A, x)$ Die Bedingung $q_0^B \notin \delta_B(q, x)$ ist für alle $q \in Q_B$ offensichtlich erfüllt. Außerdem gilt für jedes $w \in V^*$ die Gleichheit $\delta_B(q_0^B, w) = \delta_A(q_0^A, w)$. Besteht w nur aus einen Buchstaben, so ist das gerade die Definition von $\delta_B(q_0^B, x)$. Da aber im weiteren Verlauf der Abarbeitung des Wortes w der Anfangszustand q_0^B nie wieder betreten wird, kann keine Unterschied zwischen $\delta_B(q_0^B, w)$ und $\delta_A(q_0^A, w)$ auftreten. Das sichert, daß $w \in L(\mathcal{B})$ genau dann gilt, wenn $w \in L(\mathcal{A})$ gilt. ■

Die Finalzustände eines endlichen Automaten spielen genau genommen eine zweifache Rolle. Zum einen zeigt ihr Erreichen an, daß ein Wort akzeptiert wird, zum anderen sind sie auch Zwischenzustände, nach denen die Aktion des Automaten weitergeht. Das folgende Lemma zeigt, daß man sich ohne Verlust an Ausdrucksstärke auf Automaten beschränken könnte, in denen Endzustände ihren Namen auch verdienen.

Lemma 10.10

Zu jedem nichtdeterministischen endlichen Automaten \mathcal{A} mit $\varepsilon \notin L(\mathcal{A})$ gibt es einen nichtdeterministischen endlichen Automaten $\mathcal{B} = (Q_B, V, q_0^B, \delta_B, F_B)$ mit $L(\mathcal{A}) = L(\mathcal{B})$, so daß für alle $q \in F_B$ und alle $x \in V$ gilt $\delta(q, x) = \emptyset$.

Beweis: Wir beginnen mit $\mathcal{A} = (Q_A, V, q_0^A, \delta_A, F_A)$. Für jeden Finalzustand $q \in F_A$ wählen wir ein Symbol q^e , sodaß alle neuen q^e untereinander verschieden sind und auch nicht in Q_A vorkommen. Wir setzen $Q_B = Q_A \cup \{q^e \mid q \in F_A\}$. Endzustände in \mathcal{B} sind nur die neuen Kopien der alten Endzustände, d.h. $F_B = \{q^e \mid q \in F_A\}$. Der Anfangszustand bleibt unverändert, $s_0^B = s_0^A$, und die Übergangsfunktion wird festgesetzt zu

$$\begin{aligned} \delta_B(s, x) &= \delta_A(s, x) \cup \{q^e \mid q \in \delta_A(s, x)\} && \text{falls } s \notin F_B \\ \delta_B(s, x) &= \emptyset && \text{falls } s \in F_B \end{aligned}$$

Durch Induktion über die Länge des Wortes w zeigt man, daß für jedes $w \in V^*$ die Gleichung $\delta_B(s_0^B, w) = \delta_A(s_0^A, w) \cup \{q^e \mid q \in \delta_A(s_0^A, w)\}$ gilt. Da $F_B \cap \delta_B(s_0^B, w) = \emptyset$ genau dann gilt, wenn $F_A \cap \delta_A(s_0^A, w) = \emptyset$ gilt, erhalten wir, wie gewünscht, $L(\mathcal{A}) = L(\mathcal{B})$.

Hier folgt der induktive Beweis von

$$\delta_B(s_0^B, w) = \delta_A(s_0^A, w) \cup \{q^e \mid q \in \delta_A(s_0^A, w)\}$$

Für den Anfangsfall $w = \varepsilon$ ergibt sich nach Definition von δ

$$\delta_B(s_0^B, \varepsilon) = \{s_0^B\} = \{s_0^A\} = \delta_A(s_0^A, \varepsilon) \cup \{q_0^{A,e} \mid q_0^A \in F_A\}$$

Die Gleichung würde stimmen wenn der Anfangszustand von \mathcal{A} keine Endzustand wäre. Das ist aber gerade durch die Voraussetzung $\varepsilon \notin L(\mathcal{A})$ garantiert.

Im Induktionsschritt nehmen wir die Gültigkeit der zu beweisenden Gleichung für das Wort w an und wollen sie für das Wort wa zeigen.

$$\begin{aligned} s \in \delta_B(s_0^B, wa) &\Leftrightarrow \exists s'(s' \in \delta_B(s_0^B, w) \wedge s \in \delta_B(s', a)) \\ &\Leftrightarrow \exists s'(s' \in \delta_A(s_0^A, w) \cup \{q^e \mid q \in \delta_A(s_0^A, w)\} \\ &\quad \wedge s \in \delta_B(s', a)) \\ &\Leftrightarrow \exists s'(s' \in \delta_A(s_0^A, w) \wedge s \in \delta_B(s', a)) \end{aligned}$$

Die erste Äquivalenz folgt aus der Definition von δ_B , die zweite nutzt die Induktionsvoraussetzung für $\delta_B(s_0^B, w)$ und die dritte Zeile rührt von der Beobachtung, daß $s' \in \{q^e \mid q \in \delta_A(s_0^A, w)\}$ nicht auftreten kann, da anderenfalls $\delta_B(s', a) = \emptyset$ wäre. Setzen wir die Definition von δ_B ein, erhalten wir:

$$s \in \delta_B(s_0^B, wa) \Leftrightarrow \exists s'(s' \in \delta_A(s_0^A, w) \wedge s \in \delta_A(s', a) \cup \{q^e \mid q \in \delta_A(s', a)\})$$

Woraus jetzt leicht folgt

$$s \in \delta_B(s_0^B, wa) \Leftrightarrow s \in \delta_A(s_0^A, wa) \cup \{q^e \mid q \in \delta_A(s_0^A, wa)\}$$

Wäre $\varepsilon \in L(\mathcal{A})$, so wäre s_0^A ein Endzustand in \mathcal{A} gewesen. Nach der obigen Konstruktion ist s_0^A zwar auch noch der Anfangszustand von \mathcal{B} , aber kein Finalzustand mehr. Die in der Formulierung des Lemmas gemachte Einschränkung an \mathcal{A} ist also notwendig. ■

Definition 10.11 (Produktautomat)

Seien $\mathcal{A} = (S_A, V, s_0^A, \delta_A, F_A)$ und $\mathcal{B} = (S_B, V, s_0^B, \delta_B, F_B)$ zwei endliche Automaten über dem gemeinsamen Alphabet V , dann ist der Produktautomat $\mathcal{C} = (S_C, V, s_0^C, \delta_C, F_C) = \mathcal{A} \times \mathcal{B}$ gegeben durch:

$$\begin{aligned} S_C &= \{(s_1, s_2) \mid s_1 \in S_A, s_2 \in S_B\} \\ s_0^C &= (s_0^A, s_0^B) \\ \delta_C((s_1, s_2), a) &= \{(t_1, t_2) \mid t_1 \in \delta_A(s_1, a), t_2 \in \delta_B(s_2, a)\} \\ F_C &= \{(s_1, s_2) \mid s_1 \in F_A, s_2 \in F_B\} \end{aligned}$$

Lemma 10.12

$$L(\mathcal{A} \times \mathcal{B}) = L(\mathcal{A}) \cap L(\mathcal{B})$$

Beweis Ist in jedem Einführungstext in die Theorie endlicher Automaten enthalten. ■

10.1.3 Reguläre Ausdrücke

Neben den Operationen, die für beliebige Mengen erklärt werden können, wie Vereinigung (\cup), Durchschnitt (\cap), relative Komplementbildung (\setminus) betrachten wir hier Operationen die nur auf Mengen von Wörtern Sinn machen:

Definition 10.13 (Operationen mit Wortmengen)

Seien $L, L_1, L_2 \subseteq V^*$.

1. $L_1L_2 = \{w_1w_2 \mid w_1 \in L_1, w_2 \in L_2\}$,
2. $L^* = \{w_1 \dots w_n \mid n \geq 0, w_i \in L\}$

Manche Autoren verwenden für die Konkatenation von Wörtern w_1, w_2 und Sprachen L_1, L_2 die Notation $w_1 \cdot w_2$ bzw. $L_1 \cdot L_2$

Definition 10.14 (Reguläre Ausdrücke)

Die Menge Reg_V der regulären Ausdrücke über einem Alphabet V wird induktiv definiert durch:

1. $\emptyset \in Reg_V$,
2. $\varepsilon \in Reg_V$,
3. für jedes $a \in V$ ist $a \in Reg_V$,
4. für $t \in Reg_V$ gilt auch $(t)^* \in Reg_V$,
5. für $t_1, t_2 \in Reg_V$ gilt auch $(t_1t_2) \in Reg_V$ und $(t_1 + t_2) \in Reg_V$.

Definition 10.15 (Semantik regulärer Ausdrücke)

Durch die folgende Vorschrift wird jedem regulären Ausdruck t über V eine Menge $S(t)$ von Wörtern in V^* zugeordnet.

1. $S(\emptyset) = \emptyset$,

2. $S(\varepsilon) = \{\varepsilon\}$,
3. $S(a) = \{a\}$,
4. $S((t)^*) = S(t)^*$,
5. $S((t_1 t_2)) = S(t_1)S(t_2)$ und $S((t_1 + t_2)) = S(t_1) \cup S(t_2)$.

Wir benutzen im folgenden stillschweigend die Assoziativität der Konkatenation und von $+$ um in regulären Ausdrücken Klammern einzusparen, also $(a + b + c)$ anstelle von $((a + b) + c)$.

Satz 10.16

1. Für jeden regulären Ausdruck t ist $S(t)$ eine reguläre Sprache.
2. Zu jedem endlichen Automaten A gibt es einen regulären Ausdruck t mit $S(t) = L(A)$.

Beweis:

Für 1. siehe z.B. [Kfo88], Proposition 2, Seite 34 oder [HU79], Theorem 2.3, Seite 30.

Für 2. siehe z.B. [Kfo88], Proposition 12, Seite 38 oder [HU79], Theorem 2.4, Seite 33.

■

Reguläre Ausdrücke sind in vielen Bereichen der Informatik ein alltägliches Hilfsmittel geworden. Als ein Beispiel dafür wollen wir etwas ausführlicher auf die Benutzung regulärer Ausdrücke im GNU Emacs Editor eingehen, siehe z.B. [Sta88] oder online help für emacs. Während einer Editiersitzung kann man das Kommando `C-M-s` eingeben, worauf in der Kommandozeile `Regexp I-search:` erscheint. Hinter dem Doppelpunkt kann man jetzt einen regulären Ausdruck t in der emacs-Syntax (Abschnitt 13.5 in [Sta88]) eingeben. Der Cursor springt dann unmittelbar auf die erste, von der aktuellen Cursorposition aus gesehen, Zeichenkette im Textpuffer die in $S(t)$ liegt. Das Kommando `C-s` liefert das nächste Vorkommen einer Zeichenkette in $S(t)$. Abgesehen von der Nützlichkeit dieser Kommandos ist es auch instruktiv zu sehen, wie das theoretische Konzept regulärer Ausdrücke in praktisches umgesetzt wird.

Die erste Frage die zu beantworten ist lautet: Was ist das Alphabet V_{emacs} ? Antwort V_{emacs} besteht aus den 256 Zeichen des erweiterten ASCII Zeichensatzes. Dazu gehören auch Zeichen, die man normalerweise nicht auf dem

Bildschirm zu sehen bekommt, z.B. das ASCII Zeichen mit dem Oktalcode 012. Dieses Zeichen heißt *LF*, für line feed. Man sieht es nicht selbst auf dem Bildschirm, sondern nur seine Auswirkung, daß eben an dieser Stelle eine neue Zeile beginnt. Aber *LF* kann genau so in einem regulären Ausdruck benutzt werden, wie jedes andere Zeichen aus V_{emacs} auch. An dieser Stelle sollte man wissen, wie nicht-graphische Zeichen von emacs dargestellt werden und wie man sie eingeben kann ohne ein eventuell damit verbundenes Editor Kommando auszulösen. Das kann man in [Sta88] in den Abschnitten 2 und 4.1 nachlesen. Ein interessanteres Problem tritt auf, wenn wir nach Zeichenketten suchen wollen, die mit einem x beginnen auf welches eine gerade Anzahl von y 's folgt. Wir würden gern den regulären Ausdruck $x(yy)^*$ eingeben. Die Klammern sind notwendig um den gewünschten regulären Ausdruck von dem ungewünschten $(xyy)^*$ oder $xy(y)^*$ zu unterscheiden. Andererseits sind die beiden Klammern (und) Zeichen in V_{emacs} . Gibt man nach dem Prompt `Regexp I-search: x(yy)*` ein so findet man alle Ausdrücke der Form $x(yy)\dots$ mit einer beliebigen Anzahl schliessender Klammern. Dieses Problem wird gelöst, wie an vielen anderen Stellen in der Informatik auch, indem man V_{emacs} unterteilt in Sonderzeichen und normale Zeichen. An dieser Stelle geht die Einfachheit des theoretischen Konzepts verloren unter vielfachen Ausnahme- und Sonderregeln, Kompatibilitätsrücksichten und historischen Gewohnheiten. In der emacs Syntax regulärer Ausdrücke sind zunächst einmal die folgenden neun Zeichen Sonderzeichen:

`$, ^, ., *, +, ?, [,], \`

Wir betrachten zunächst `\`. Es hat zwei Funktionen. Erstens dient es dazu, die Sonderzeichen zu normalen Zeichen zu machen, d.h. für den regulären Ausdruck `\$` gilt $S(\$) = \{\$\}$ und konsequenterweise auch $S(\backslash) = \{\backslash\}$. Zweitens macht `\` aus einigen normalen Zeichen Sonderzeichen. Hier ist eine Liste einiger auf diese Weise gebildeter Sonderzeichen:

`\|, \(), \), ', ` , \b, \B, \<, \>, \w, \W, \scode, \Scode`
und `\z` für jede Ziffer z .

Wir wollen nur die wichtigsten dieser Sonderzeichen erklären. Eine vollständige Beschreibung findet man in [Sta88] in Abschnitt 13.5. Für reguläre emacs Ausdrücke t_1, t_2 gilt $S(t_1|t_2) = S(t_1) \cup S(t_2)$, d.h. `|` in regulären emacs Ausdrücken entspricht dem `+` in den regulären Ausdrücken aus Definition 10.14. Die Sonderzeichen `\(` (und `\)`) dienen der Klammerung. Der oben erwähnte reguläre Ausdruck $x(yy)^*$ wird also in emacs Syntax geschrieben als `x\ (yy\)*`.

Hiermit ist auch gleich verraten, daß der Stern * in den regulären emacs Ausdrücken dieselbe Bedeutung hat wie in unseren regulären Ausdrücken. Da auch die Konkatenation von regulären emacs Ausdrücken mit der üblichen Semantik möglich ist, entspricht als jedem regulären Ausdruck nach Definition 10.14 ein regulärer emacs Ausdruck mit derselben Bedeutung. Wir richten im folgenden unser Augenmerk auf die Frage, ob reguläre emacs Ausdrücke vielleicht ausdrucksmächtiger sind als unsere regulären Ausdrücke.

Das Sonderzeichen . besitzt als seine Interpretation $S(\cdot)$ die Menge aller Zeichen mit der einzigen Ausnahme des Zeichens LF . Da kann man in der Syntax aus Definition 10.14 auch hinschreiben als eine Summe mit 255 Summanden. Die Sonderzeichen + und ? sind wie * postfix Operatoren, wobei + die übliche Bedeutung, $t+ \equiv t(t)^*$, hat und $t? \equiv (\varepsilon + t)$. Die Sonderzeichen ^ und \$ suchen nach dem leeren Wort am Anfang bzw. am Ende einer Zeile. Genauer gilt für einen regulären emacs Ausdruck t $S(\hat{t}) = \{w \in S(t) \mid w \text{ kommt am Anfang einer Zeile vor}\}$ und $S(t\$) = \{w \in S(t) \mid w \text{ kommt am Ende einer Zeile vor}\}$.

Die Sonderzeichen \[und \] erlauben eine bequeme Beschreibung endlicher Mengen von Zeichen. Es gilt $S([a_1 \dots a_k]) = \{a_1, \dots, a_k\}$, jedenfalls dann, wenn die a_i keine Sonderzeichen enthalten. Allerdings gelten innerhalb der [...] Paare ganz andere Sonderzeichenregeln als bisher. Hier sind nämlich

], ^, -

die einzigen Sonderzeichen. Warum] ein Sonderzeichen ist, ist klar. Das Minuszeichen - wird benutzt um einen Bereich aufeinanderfolgender Zeichen zu beschreiben, z.B. $S([a - z]) =$ Menge aller Kleinbuchstaben. Schließlich ist $S([\hat{a}_1 \dots a_k]) = \{z \in V_{emacs} \mid z \neq a_1 \text{ und } z \neq \dots \text{ und } z \neq a_k\}$. Was macht man, wenn man eines der drei Sonderzeichen in einem [...] Paar verwenden möchte? Anstelle von - schreibt man ---,] ist kein Sonderzeichen, wenn es unmittelbar nach [steht und ^ ist nur dann ein Sonderzeichen, wenn es unmittelbar nach [steht. Es gibt noch einige kleinere Subtilitäten, aber auf die brauchen wir hier nicht einzugehen.

Mancher Leser mag sich gefragt haben, was in dem oben erwähnten Prompt **Regexp I-search**: wohl das I bedeutet? Es steht für *inkrementell*. Die Suche nach der ersten Zeichenkette, die zu dem regulären Ausdruck gehört, erfolgt inkrementell. Tippt man als regulären Ausdruck a ein, so springt der Cursor zum ersten im Puffer vorkommenden a . Tippt man jetzt noch ein b dazu, so springt der Cursor zum ersten Vorkommen der Zeichenkette ab . Wie verhält sich die Suche bei Eingabe von $a \setminus b$? Da waren sich die Implementierer nicht immer einig. In meiner emacs Version beginnt die Suche nach Eingabe von

$a \setminus |$ nochmal von vorn, so daß auch die bs gefunden werden, die vor dem ersten a im Text stehen.

10.1.4 Übungsaufgaben

Übungsaufgabe 10.1.1

Zeigen Sie, daß es zu jedem endlichen Automaten EA einen vollständigen endlichen Automaten EAc gibt mit $L(EA) = L(EAc)$. Kann EAc immer gebildet werden, ohne die Menge der Zustände von EA zu vergrößern?

Übungsaufgabe 10.1.2

Zeigen Sie, daß es zu jedem nichtdeterministischen endlichen Automaten NEA mit spontanen Übergängen einen nichtdeterministischen endlichen Automaten NEA_0 ohne spontane Übergänge gibt mit $L(NEA) = L(NEA_0)$. ([Hopcroft, Ullman 79], p. 24.)

10.2 Omega Automaten

Manche Automaten besitzen keine terminalen Zustände, man möchte im Gegenteil sogar, daß sie nicht terminieren. Ein Beispiel dafür sind Sender- und Empfängerautomaten für Übertragungsprotokolle. Das führt auf der theoretischen Seite zur Einführung unendlich langer Wörter.

Definition 10.17 (Unendliche Wörter)

Sei V ein (weiterhin endliches) Alphabet.

Mit V^ω bezeichnen wir die Menge der unendlich langen Wörter mit Buchstaben aus V .

Für $n \in \mathbb{N}$ bezeichnet $w(n)$ den n -ten Buchstaben in w und $w \downarrow (n)$ das endliche Anfangstück $w(0) \dots w(n)$ von w .

Wir nennen ein Wort $w \in V^\omega$ manchmal auch ein ω -Wort über V .

Wer Schwierigkeiten hat, sich eine unendlich lange Zeichenkette vorzustellen, der interpretiere ein Wort $w \in V^\omega$ als eine Funktion $w : \mathbb{N} \rightarrow V$, von den natürlichen Zahlen in das Alphabet. Die Präzisierung beantwortet auch die Frage des mit der Theorie der unendlichen Mengen vertrauten Lesers, welchen Grad der Unendlichkeit wir meinen: unendliche Wörter sind Folgen vom Ordnungstyp ω .

Man beachte außerdem, daß das leere Wort ε nicht in V^ω vorkommt.

Definition 10.18 (Operationen)

1. Ist $K \subseteq V^*$ eine Menge (endlicher) Wörter, so bezeichnet K^ω die Menge der unendlichen Wörter der Form

$$w_1 \dots w_i \dots \text{ mit } w_i \in K \text{ für alle } i$$

2. Die Verkettung zweier unendlich langer Wörter macht keinen Sinn, es ist jedoch möglich einem unendlich langen Wort ein endliches vorzuschalten. Ist $K \subseteq V^*$ und $J \subseteq V^\omega$, dann setzen wir:

$$KJ = \{w_1w_2 \mid w_1 \in K, w_2 \in J\}$$

3. Ist $K \subseteq V^*$ eine Menge endlicher Wörter, dann ist

$$\vec{K} = \{w \in V^\omega \mid w \downarrow (n) \in K \text{ für unendlich viele } n\}$$

Manche Autoren benutzen $\lim(K)$ anstelle von \vec{K} .

10.2.1 Büchi Automaten

Definition 10.19 (Büchi Automaten)

Sei $\mathcal{A} = (S, V, s_0, \delta, F)$ ein nicht deterministischer endlicher Automat. Für ein ω -Wort $w \in V^\omega$ nennen wir eine Folge s_0, \dots, s_n, \dots eine *Berechnungsfolge* (Englisch *run*) für w , wenn für alle $0 \leq n$ gilt

$$s_{n+1} \in \delta(s_n, w(n))$$

Eine Berechnungsfolge s_0, \dots, s_n, \dots heißt eine *akzeptierte Berechnungsfolge* wenn in ihr unendlich viele Finalzustände vorkommen.

Die von \mathcal{A} akzeptierte ω -Sprache wird definiert durch

$$L^\omega(\mathcal{A}) = \{w \in V^\omega \mid \text{es gibt eine akzeptierte Berechnungsfolge für } w\}$$

Wir nennen eine Menge L von ω -Wörtern **ω -regulär**, wenn es einen endlichen Automaten \mathcal{A} gibt mit $L^\omega(\mathcal{A}) = L$.

Gelegentlich braucht man auch den Begriff einer Berechnungsfolge unabhängig davon, welches Wort damit assoziiert ist.

Eine Berechnungsfolge für den Automaten $\mathcal{A} = (S, V, s_0, \delta, F)$ ist eine Folge s_1, \dots, s_n, \dots von Zuständen, so daß für alle $0 \leq n$ ein $a \in V$ existiert mit $s_{n+1} \in \delta(s_n, a)$. Eine Berechnungsfolge heißt akzeptierend, wenn unendliche viele der s_i Endzustände sind.

Man beachte, daß eine Berechnungsfolge mit s_1 beginnt. Das ist eine technische Detail, das später einige Vereinfachungen mit sich bringt.

Ein Büchi Automat \mathcal{A} unterscheidet sich also in nichts von einem endlichen Automaten, nur die Definition, wann \mathcal{A} ein ω -Wort akzeptiert ist neu.

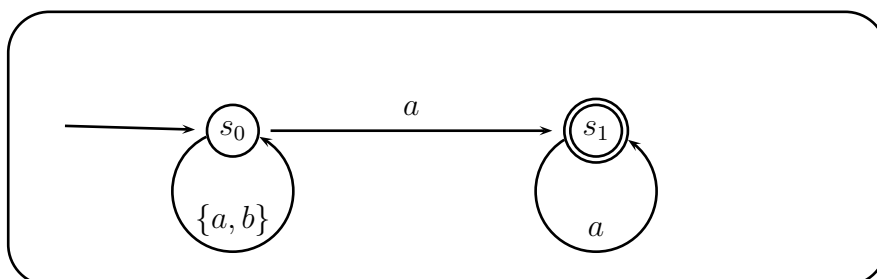


Abbildung 10.2: Der Büchi-Automat \mathcal{N}_{afin}

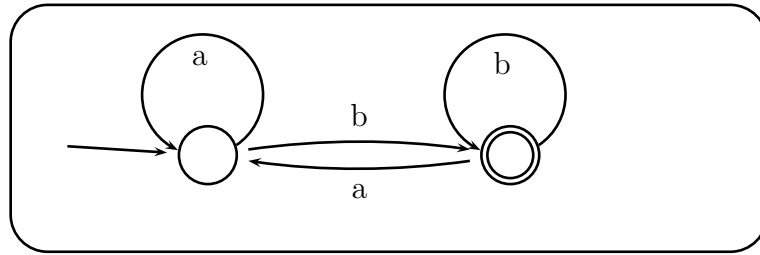


Abbildung 10.3: Büchi Automat für $(a^*b^+a)^\omega + (a^*b^+a)^*b^\omega$

Satz 10.20 (Entscheidbarkeit)

Die Frage, ob für einen Büchi-Automaten \mathcal{B} die Menge der akzeptierten Wörter nicht leer ist, d.h. $L^\omega(\mathcal{B}) \neq \emptyset$, ist entscheidbar.

Beweis: Um $L^\omega(\mathcal{B}) \neq \emptyset$ zu zeigen muß man nur einen erreichbaren Endzustand $q_f \in F$ finden, der auf einer Schleife liegt, d.g. q_f ist erreichbar von q_f ausgehend. Diese Frage kann sogar in linearer Zeit beantwortet werden, z.B. durch den Tarjan-Paige algorithmus [Tar72].

■

Das folgende Lemma fasst zusammen, was über den Zusammenhang zwischen $L(\mathcal{A})$ und $L^\omega(\mathcal{A})$ gesagt werden kann

Lemma 10.21

Sei \mathcal{A} ein endlicher Automat und $K = L(\mathcal{A})$. Dann gilt

1. $L^\omega(\mathcal{A}) \subseteq \vec{K}$
2. Falls \mathcal{A} deterministisch ist gilt sogar $L^\omega(\mathcal{A}) = \vec{K}$

Beweis:

zu 1: Nach Definition ist für $w \in L^\omega(\mathcal{A})$ gibt es eine Berechnungsfolge $\rho(w)$, so daß die Menge $F_w = \{n \in \mathbb{N} \mid \rho(w)(n) \in F\}$ unendlich ist. Für alle $n \in F_w$ gilt $\rho(w)(n) \in F$ und daher $w \downarrow (n) \in K$. Also $w \in \vec{K}$.

zu 2: Sei jetzt umgekehrt $w \in \vec{K}$. Dann ist die Menge $R_w = \{n \in \mathbb{N} \mid w \downarrow (n) \in K\}$ unendlich. Für jedes $n \in R_w$ gibt es also eine Berechnungsfolge s_n von \mathcal{A} für $w \downarrow (n)$. Da \mathcal{A} deterministisch ist gibt es für jedes n nur eine Möglichkeit für s_n . Insbesondere ist s_n jeweils ein Anfangsstück von s_{n+1} .

Somit erhalten wir im Limes eine unendliche Berechnungsfolge s für w , die unendlich oft einen Endzustand durchläuft.

■

Für nicht-deterministische Automaten gilt Teil 2 von Lemma 10.21 nicht notwendigerweise. Wir betrachten dazu den Automaten N_{afin} in Abbildung 10.2. Es gilt $L^\omega(N_{afin}) = \{w \in \{a, b\}^\omega \mid \text{in } w \text{ kommt } a \text{ nur endlich oft vor}\}$ und $L(N_{afin}) = \{w \in \{a, b\}^* \mid w \text{ endet auf } b\}$. Man sieht leicht, daß $L^\omega(N_{afin}) \neq Lim(L(N_{afin}))$.

Korollar 10.22

Eine Sprache $L \subseteq V^\omega$ wird von einem deterministischen Büchi Automaten akzeptiert, gdw es eine reguläre Sprache $K \subseteq K^*$ gibt mit $L = \vec{K}$.

Beweis:

Wird L von einem deterministischen Büchi Automaten akzeptiert, so kann L nach Lemma 10.21 auch in der angegebenen Weise dargestellt werden. Nehmen wir jetzt umgekehrt an, daß $L = \vec{K}$ für eine reguläre Sprache $K \subseteq V^*$ gilt. Dann gibt es einen deterministischen endlichen Automaten \mathcal{A} mit $K = L(\mathcal{A})$. Man überzeugt sich leicht, daß in diesem Fall $\vec{K} = L^\omega(\mathcal{A})$ gilt.

■

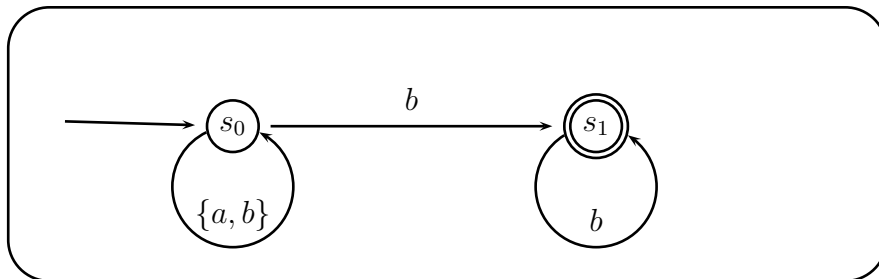


Abbildung 10.4: Der Büchi-Automat \mathcal{N}_{bfin}

Korollar 10.23

Es gibt Sprachen $L \subseteq V^\omega$, die von einem nicht deterministischen Büchi Automaten akzeptiert werden, aber von keinem deterministischen.

Beweis:

Sei $V = \{a, b\}$ und L die von dem Automaten \mathcal{N}_{afin} in Abbildung 10.2 akzeptierte Sprache:

$$L = \{w \in V^\omega \mid w(n) = a \text{ nur für endlich viele } n\}$$

Wir zeigen, daß L nicht in der Form $L = \vec{K}$ für eine reguläre Menge $K \subseteq V^*$ darstellbar ist. Wir nehmen an, das wäre doch der Fall und versuchen einen Widerspruch herzuleiten.

Es gibt ein $k_1 > 0$, so daß $b^{k_1} \in K$, da $b^\omega \in L$. Als nächstes muß es dann auch ein $k_2 > 0$ geben, so daß $b^{k_1}ab^{k_2} \in K$ liegt, weil $b^{k_1}ab^\omega \in L$ gilt. In dieser Weise fortfahrend gibt es $k_i > 0$, so daß $b^{k_1}ab^{k_2}a \dots ab^{k_i} \in K$ gilt für alle i . Nach Annahme über die Darstellbarkeit von L folgt daraus auch $b^{k_1}ab^{k_2}a \dots ab^{k_i}a \dots \in L$ im Widerspruch zur Definition von L . ■

10.2.2 Abschlußigenschaften

Satz 10.24

Sind L_1, L_2 ω -reguläre Sprachen und ist K eine reguläre Sprache, dann ist auch

1. $L_1 \cup L_2$ ω -regulär,
2. K^ω ω -regulär, falls $\varepsilon \notin K$,
3. KL_1 ω -regulär,
4. $V^\omega \setminus L_1$ ω -regulär,
5. $L_1 \cap L_2$ ω -regulär.

Beweis:

zu 1. Seien zwei ω -reguläre Sprachen $L_1, L_2 \subseteq V^\omega$ gegeben und $\mathcal{A}_i = (Q_i, V, s_0^i, \delta_i, F_i)$ Büchi-Automaten mit $L_i = L_i^\omega(\mathcal{A}_i)$. Wir können ohne Beschränkung der Allgemeinheit annehmen, daß $Q_1 \cap Q_2 = \emptyset$

Wir konstruieren einen Büchi-Automaten $\mathcal{A} = (Q, V, s_0, \delta, F)$ wie folgt, wobei s_0 ein neuer Zustand ist, der weder in Q_1 noch in Q_2 vorkommt.

$$\begin{aligned} Q &= Q_1 \cup Q_2 \cup \{s_0\} \\ \delta(q, a) &= \delta_i(q, a) && \text{falls } q \in Q_i \\ \delta(s_0, a) &= \delta_1(s_0^1, a) \cup \delta_2(s_0^2, a) \\ F &= F_1 \cup F_2 \end{aligned}$$

Man zeigt leicht, daß $L^\omega(\mathcal{A}) = L_1 \cup L_2$.

zu 2. Sei $\mathcal{A} = (Q_A, V, s_0^A, \delta_A, F_A)$ ein nichtdeterministischer endlicher Automat mit $L(\mathcal{A}) = K$. Wegen Lemma 10.9 können wir $s_0^A \notin \delta_A(q, x)$ für alle $q \in Q_A$ und $x \in V$ annehmen.

Wir definieren einen neuen Automaten $\mathcal{B} = (Q_B, V, s_0^B, \delta_B, F_B)$ durch:

$$\begin{aligned} Q_B &= Q_A \\ s_0^B &= s_0^A \\ \delta_B(q, a) &= \delta_A(q, a) \cup \{s_0^A\} \quad \text{falls } F_A \cap \delta_A(q, a) \neq \emptyset \\ \delta_B(q, a) &= \delta_A(q, a) \quad \text{sonst} \\ F_B &= \{s_0^B\} \end{aligned}$$

Wir müssen uns überzeugen, daß $L^\omega(\mathcal{B}) = K^\omega$ gilt.

Betrachten wir dazu zunächst ein $w \in L^\omega(\mathcal{B})$. Sei $q_1 \dots q_i \dots$ eine akzeptierende Berechnungsfolge für w . Nach Definition eines Büchi-Automaten muß der einzige Endzustand von \mathcal{B} unendlich oft vorkommen. Seien $n_1, \dots, n_k \dots$ die Positionen, sodaß $q_{n_k} = s_0^B$. Offensichtlich ist $n_1 = 0$. Mit w_k bezeichnen wir das Teilwort $w(q_{n_k}), w(q_{n_k+1}), \dots, w(q_{n_{k+1}-1})$ von w . Nach Einlesen eines Teilwortes w_k ist der Automat \mathcal{B} im Zustand q_0^B . Das kann wegen der Definition von δ_B und der Voraussetzung an \mathcal{A} nur der Fall sein wenn $F_A \cap \delta_A(s_0^A, w_k) \neq \emptyset$ gilt. Das heißt für alle k gilt $w_k \in L(\mathcal{A})$. Daraus folgt $w \in K^\omega$.

Sei jetzt umgekehrt w ein Wort aus K^ω . Also $w = w_1 \dots w_i \dots$ mit $w_i \in K$. Es gibt somit für jedes i akzeptierende Berechnungsfolgen $q_{n_i}, q_{n_i+1}, \dots, q_{n_{i+1}}$ für w_i in $L(\mathcal{A})$. Nach Definition einer endlichen akzeptierenden Berechnungsfolge muß $q_{n_{i+1}} \in F_A$ gelten. Nach Definition von \mathcal{B} ist auch $q_{n_i}, q_{n_i+1}, \dots, q_{n_{i+1}-1}, s_0^B$ eine Berechnungsfolge für w_i in \mathcal{B} . Setzen wir diese Berechnungsfolgen hintereinander erhalten wir eine Berechnungsfolge für w , in der der Finalzustand q_0^B an jeder Position n_i , und damit unendlich oft, auftritt. Also $w \in L(\mathcal{B})$. ■

- zu 3. Übungsaufgabe 10.2.1
- zu 4. zurückgestellt auf später, Satz 10.30
- zu 5. folgt unmittelbar aus 1. und 4. ■

Satz 10.25

Eine Teilmenge $L \subseteq V^\omega$ ist eine ω -reguläre Menge von ω -Wörtern, genau dann, wenn L eine endliche Vereinigung von Mengen der Form

$$JK^\omega$$

für reguläre Mengen $J, K \subseteq V^*$ ist, wobei $\varepsilon \notin K$.

Beweis:

Sei $\mathcal{A} = (Q, V, s_0, \delta, F)$ ein Büchi-Automat mit $L^\omega(\mathcal{A}) = L$. Für $p, q \in Q$ sei

$$L_{p,q} = \{w \in V^* \mid q \in \delta(p, w)\}$$

Offensichtlich ist jedes $L_{p,q} \subseteq V^*$ eine reguläre Menge. Man überzeugt sich leicht, daß

$$L = \bigcup_{p \in F} L_{s_0,p} L_{p,p}^\omega$$

gilt. Die umgekehrter Richtung der Behauptung folgt direkt aus Satz 10.24. ■

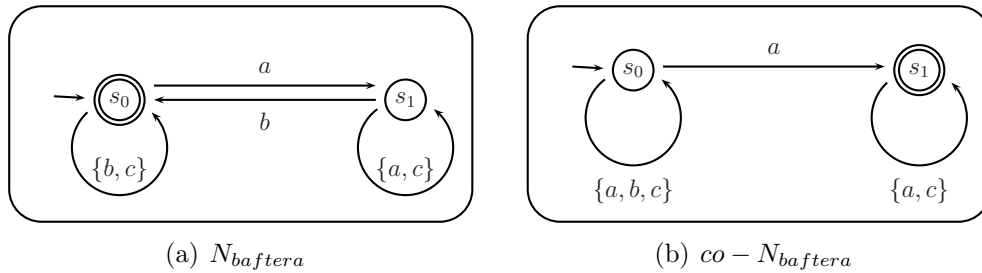


Abbildung 10.5: Beispiel zur Komplementbildung

In Abbildung 10.5 ist ein Paar komplementärer Büchi-Automaten zu sehen. Das Beispiel stammt aus [Tho90]. Es gilt $L^\omega(N_{bftera}) = \{w \in \{a, b, c\}^\omega \mid \text{nach jedem } a \text{ kommt irgendwann ein } b\}$. $L^\omega(co - N_{bftera})$ ist dazu offensichtlich die Komplementärmenge, d.h. $\{w \in \{a, b, c\}^\omega \mid \text{es gibt ein } a \text{ nach dem kein } b \text{ mehr vorkommt}\}$. Da es zu jedem Büchi-Automaten einen Komplementärautomaten gibt, d.h. die Abgeschlossenheit ω -regulärer Sprachen unter Komplementbildung in V^ω , muß noch bewiesen werden. Der Beweis bedarf einiger Vorbereitung. Eine zentrale Rolle spielt dabei die folgende Äquivalenzrelation.

Definition 10.26

Sei $\mathcal{A} = (Q, V, s_0, \delta, F)$ ein Büchi-Automat. Dann definieren wir für $p, q \in Q$ und $u, w \in V^*$

1. $L_{p,q} = \{w \in V^* \mid q \in \delta(p, w)\}$
(wie im Beweis von Satz 10.25)
2. $L_{p,q}^F = \{w = a_0 \dots a_k \in V^* \mid \text{es gibt eine Folge von Zuständen } q_0, \dots, q_k \text{ mit}$

q_0	$=$	p	
q_k	$=$	q	
q_{i+1}	\in	$\delta(q_i, a_i)$	für alle $0 \leq i < k$
q_i	\in	F	für ein $0 \leq i \leq k$ }

3. $u \equiv_A w$ gdw
für alle $p, q \in Q$ gilt
 $u \in L_{p,q} \Leftrightarrow w \in L_{p,q}$
 $u \in L_{p,q}^F \Leftrightarrow w \in L_{p,q}^F$

Offensichtlich gilt stets $L_{p,q}^F \subseteq L_{p,q}$ und falls $p \in F$ oder $q \in F$ auch $L_{p,q}^F = L_{p,q}$. Ebenso leicht sieht man, daß \equiv_A eine Äquivalenzrelation ist.

Lemma 10.27

Die Äquivalenzrelation \equiv_A hat folgende Eigenschaften:

1. für alle $p, q \in Q$ sind $L_{p,q}^F$ und $L_{p,q}$ reguläre Mengen,
2. für $u \in V^\omega$ gilt für die Äquivalenzklasse M_w

$$M_w = \bigcap_{(p,q) \in P} L_{p,q} \cap \bigcap_{(p,q) \notin P} \sim L_{p,q} \cap \bigcap_{(p,q) \in R} L_{p,q}^F \cap \bigcap_{(p,q) \notin R} \sim L_{p,q}^F$$

wobei $P = \{(p, q) \in Q^2 \mid w \in L_{p,q}\}$ und $R = \{(p, q) \in Q^2 \mid w \in L_{p,q}^F\}$

3. \equiv_A besitzt endlich viele Äquivalenzklassen,
4. jede Äquivalenzklasse von \equiv_A ist eine reguläre Menge,
5. sind U, V Äquivalenzklassen von \equiv_A , dann folgt aus $UV^\omega \cap L^\omega(\mathcal{A}) \neq \emptyset$ schon $UV^\omega \subseteq L^\omega(\mathcal{A})$

Beweise:

zu 1 Für $L_{p,q}$ wurde das schon im Beweis von Satz 10.25 benutzt. Wegen $L_{p,q}^F = \bigcup_{f \in F} L_{p,f} L_{f,q}$ folgt auch die Regularität von $L_{p,q}^F$.

zu 2 folgt direkt aus der Definition von \equiv_A .

zu 3 unmittelbare Konsequenz aus 2.

zu 4 folgt aus 1, 2 und den Abschlusseigenschaften regulärer Mengen.

zu 5 Sei $w \in UV^\omega \cap L^\omega(\mathcal{A})$ gegeben und u ein weiteres Wort aus UV^ω . Es ist $u \in L^\omega(\mathcal{A})$ zu zeigen.

Wegen $w \in UV^\omega$ können wir w wie folgt zerlegen:

$$w = w_1 w_2 \dots w_n \dots$$

wobei $w_1 \in U$ und $w_i \in V$ für alle $i > 1$. Wegen $w \in L^\omega(\mathcal{A})$ gibt es eine Berechnungsfolge σ für w in \mathcal{A} in der unendlich viele Zustände aus F auftreten. Wir interessieren uns für bestimmte Stützpunkte in der Folge σ ; mit q_n bezeichnen wir den Zustand in σ , der nach Einlesen des Teilwortes w_{n-1}

und vor Beginn des Einlesens von w_n eingenommen wird. Insbesondere sei $q_0 = s_0$ der Anfangszustand. Sei schließlich $u \in UV^\omega$, genauer

$$u = u_1 u_2 \dots u_n \dots$$

mit $u_1 \in U$ und $u_i \in V$ für alle $i > 1$. Da U und V nach Voraussetzung \equiv_A Äquivalenzklassen sind gilt für alle i

$$w_i \equiv_A u_i$$

Wir konstruieren eine Berechnungsfolge ρ für u wie folgt. Dabei wird ρ die Zustände q_n in derselben Reihenfolge durchlaufen, wie σ , aber welche Zustände auf dem Weg von q_n nach q_{n+1} auftreten ist von vornherein noch nicht festgelegt, ja es ist zunächst noch nicht einmal klar ob ein solcher Weg gefunden werden kann. Nach Definition der q_i gilt auf jeden Fall $w_n \in L_{q_{n-1}, q_n}$. Wegen $w_n \equiv_A u_n$ gilt dann auch $u_n \in L_{q_{n-1}, q_n}$. Man kann also den Weg von q_{n-1} nach q_n in ρ interpolieren. Kommt in σ ein Finalzustand zwischen q_{n-1} und q_n vor, so gilt $w_n \in L_{q_{n-1}, q_n}^F$ und wegen $w_n \equiv_A u_n$ auch $u_n \in L_{q_{n-1}, q_n}^F$. Das Teilstück von ρ zwischen q_{n-1} und q_n kann demnach auch so gewählt werden, daß darin ein Finalzustand vorkommt. Das zeigt, daß ρ eine akzeptierende Berechnungsfolge für u ist und damit $u \in L^\omega(\mathcal{A})$. ■

Satz 10.28 (Satz von Ramsey)

Sei P_1, \dots, P_k eine endliche Partition aller zweielementigen Teilmenge der Menge \mathbb{N} der natürlichen Zahlen. Dann gibt es eine unendliche Teilmenge $T \subseteq \mathbb{N}$ und ein i mit $1 \leq i \leq k$, so daß alle zweielementigen Teilmengen von Elementen aus T in P_i liegen.

In Formeln:

aus $P_1 \uplus \dots \uplus P_k = \mathbb{N}^{[2]}$ folgt die Existenz einer unendlichen Menge $T \subseteq \mathbb{N}$, so daß $T^{[2]} \subseteq P_i$ für ein i .

Beweis: Siehe etwa [Mon76] p.448 oder [Fel78] p. 228 oder die Originalarbeit [Ram29]. Wichtig bei der Ausgangssituation des Ramseyschen Satzes ist, daß für zwei natürliche Zahlen $n, m \in \mathbb{N}$ nicht $(n, m) \in P_i$ und $(m, n) \in P_j$ für $i \neq j$ auftreten kann. In der vorliegenden Formulierung wurde das dadurch ausgeschlossen, daß anstelle von geordneten Paaren ungeordnete Zweiermengen betrachtet wurden. Häufig wird das Problem auch dadurch gelöst, daß nur Paare (n, m) mit $n \leq m$ betrachtet werden.

Lemma 10.29

Zu jedem ω -Wort $w \in V^\omega$ gibt es eine \equiv_A -Äquivalenzklassen U und V mit $w \in UV^\omega$.

Beweis:

Seien U_1, \dots, U_n alle Äquivalenzklassen von \equiv_A und bezeichne für $i < j$ $w_{i,j}$ das endliche Teilwort $w(i) \dots w(j-1)$ von w . Dann wird durch

$$P_r = \{\{i, j\} \in \mathbb{N}^{[2]} \mid [w_{i,j}]_{\equiv_A} = U_r\}$$

eine Partition von $\mathbb{N}^{[2]}$ definiert. Nach dem Satz von Ramsey gibt es eine unendliche Teilmenge $T \subseteq \mathbb{N}$ und ein i mit $T^{[2]} \subseteq P_i$. Sei i_0 das kleinste Element in T und $U = [w \downarrow (i_0)]_{\equiv_A}$, dann gilt offensichtlich

$$w \in UU_i^\omega$$

■

Satz 10.30

Ist $L \subseteq V^\omega$ eine ω -reguläre Menge, dann auch $V^\omega \setminus L$

Beweis

Sei \mathcal{A} ein Büchi Automat mit $L^\omega(\mathcal{A}) = L$.

$$V^\omega \setminus L = \bigcup_{(U,V) \in \mathcal{R}} UV^\omega$$

dabei gilt $(U, V) \in \mathcal{R}$ genau dann, wenn $UV^\omega \cap L = \emptyset$.

■

10.2.3 Varianten von Büchi Automaten

Wir beginnen mit der einfachsten Modifikation von Definition 10.19 für die wir nicht einmal einen neuen Namen erfinden wollen. Die modifizierten Automaten sind von der Form $\mathcal{C} = (S, V, S_0, \delta, F)$ mit einer Menge S_0 von Anfangszuständen anstelle eines einzigen Anfangszustands s_0 .

Lemma 10.31

Zu jedem Büchi Automaten $\mathcal{C} = (S, V, S_0, \delta, F)$ mit einer Menge von Anfangszuständen gibt es einen Büchi Automaten \mathcal{A} mit einem einzigen Anfangszustand und

$$L^\omega(\mathcal{C}) = L^\omega(\mathcal{A})$$

Beweis: Sei $S_0 = \{s_1, \dots, s_k\}$. Wir setzen $\mathcal{C}_i = (S, V, s_i, \delta, F)$. Offensichtlich gilt $L^\omega(\mathcal{C}) = \bigcup_{i=1}^k L^\omega(\mathcal{C}_i)$. Die Existenz von \mathcal{A} folgt jetzt aus Abgeschlossenheit ω -regulärer Mengen unter Vereinigung. ■

Definition 10.32 (Erweiterte Büchi Automaten)

Ein erweiterter Büchi Automat

$$\mathcal{A} = (S, V, s_0, \delta, F_1, \dots, F_n)$$

unterscheidet sich von einem (normalen) Büchi Automaten nur dadurch, daß er statt einer einzigen Menge F von Finalzuständen endliche viele solcher Mengen F_1, \dots, F_n enthält.

Eine Berechnungsfolge w wird akzeptiert, wenn es eine Berechnungsfolge s für w gibt, die für jedes j , $1 \leq j \leq n$ unendlich viele Zustände aus F_j enthält. Die von \mathcal{A} akzeptierte ω -Sprache kann dann definiert werden als:

$$L^\omega(\mathcal{A}) = \{w \in V^\omega \mid \text{es gibt eine Berechnungsfolge } s \text{ für } w, \\ \text{so daß für jedes } j, 1 \leq j \leq n, \text{ die Menge } \{i \mid s_i \in F_j\} \text{ unendlich ist.}\}$$

Das Konzept eines erweiterten Büchi Automaten macht in vielen Fällen das Leben (für den theoretischen Informatiker) etwas einfacher, an Ausdruckstärke wird gegenüber den einfachen Büchi Automaten nichts gewonnen:

Lemma 10.33

Zu jedem erweiterten Büchi Automaten \mathcal{A}_e gibt es einen einfachen Büchi Automaten \mathcal{A} mit

$$L^\omega(\mathcal{A}_e) = L^\omega(\mathcal{A})$$

Beweis: Dieselbe Konstruktion, die in dem Beweis von Lemma 11.2 benutzt werden wird, kann auch benutzt werden um zu einem erweiterten Automaten $\mathcal{A}_e = (S_e, V, s_0^e, \delta_e, F_1, F_2)$ einen äquivalenten einfachen $\mathcal{A} = (S, V, s_0, \delta, F)$ zu konstruieren. Diese Konstruktion kann man dann n -mal wiederholen um auch den allgemeinen Fall abzudecken. ■

Definition 10.34 (Müller Automaten)

Ein Müller Automat $\mathcal{M} = (S, V, s_0, \delta, \mathcal{F})$ ist ein endlicher Automat $\mathcal{M} = (S, V, s_0, \delta)$ ohne Angabe von Endzuständen, aber stattdessen mit einer Menge \mathcal{F} nicht leerer Endzustandsmengen, d.h. für alle $F \in \mathcal{F}$ gilt $F \subseteq S$ und

$F \neq \emptyset$.

Ist $\sigma = s_1, \dots, s_n, \dots$ eine Zustandsfolge, so bezeichnet $In(\sigma)$ die Menge der Zustände, die unendlich oft in σ vorkommen, also

$$In(\sigma) = \{s \in S \mid \text{es gibt unendlich viele } n \text{ mit } s_n = s\}$$

Die von einem Müller Automat $\mathcal{M} = (S, V, s_0, \delta, \mathcal{F})$ akzeptierte ω -Sprache wird definiert durch:

$$L^\omega(\mathcal{M}) = \{w \in V^\omega \mid In(\sigma) \in \mathcal{F} \text{ für eine Berechnungsfolge } \sigma \text{ für } w\}$$

Lemma 10.35

Die Klasse der von nichtdeterministischen Büchi Automaten akzeptierten ω -Sprachen stimmt überein mit der von nichtdeterministischen Müller Automaten akzeptierten ω -Sprachen.

Beweis: Für die einfache Richtung, siehe Übungsaufgabe 10.2.9.

Für die umgekehrte Richtung gehen wir von einem Müller Automaten $\mathcal{M} = (S, V, s_0, \delta, \mathcal{F})$ aus und suchen einen Büchi Automaten \mathcal{A} mit $L^\omega(\mathcal{M}) = L^\omega(\mathcal{A})$. Wir werden den Automaten \mathcal{A} nicht im einzelnen konstruieren, sondern seine Existenz aus den bisher bewiesenen Sätzen herleiten. Als erstes betrachten wir den Spezialfall, daß $\mathcal{F} = \{F_0\}$ ist mit $F_0 = \{a_1, \dots, a_n\}$. Seien b_1, \dots, b_m die anderen Buchstaben im Alphabet V . Nach Lemma 10.33 gibt es einen Büchi Automaten \mathcal{A}_0 mit

$$L^\omega(\mathcal{A}_0) = \{w \in V^\omega \mid \text{es gibt eine Berechnungsfolge } \sigma \text{ in } \mathcal{A}_0 \text{ für } w \\ \text{mit } F_0 \subseteq in(\sigma)\}$$

Außerdem können wir Büchi Automaten \mathcal{A}_j konstruieren mit $L^\omega(\mathcal{A}_j) = \{w \in V^\omega \mid b_j \text{ kommt in } w \text{ nur endlich oft vor}\}$. Für den *Durchschnittsautomaten* (nach Satz 10.24 (5)) \mathcal{A} von $\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_m$ gilt dann $L^\omega(\mathcal{A}) = L^\omega(\mathcal{M})$.

Ist im allgemeinen Fall $\mathcal{F} = F_0, \dots, F_k$. Dann liefert das bisherige Verfahren für $1 \leq i \leq k$ Büchi Automaten \mathcal{B}_i mit $L^\omega(\mathcal{B}_i) = L^\omega((S, V, s_0, \delta, \{F_i\}))$. Für den *Vereinigungsautomaten* (nach Satz 10.24 (1)) \mathcal{B} von $\mathcal{B}_1, \dots, \mathcal{B}_k$ gilt dann $L^\omega(\mathcal{B}) = L^\omega(\mathcal{M})$. ■

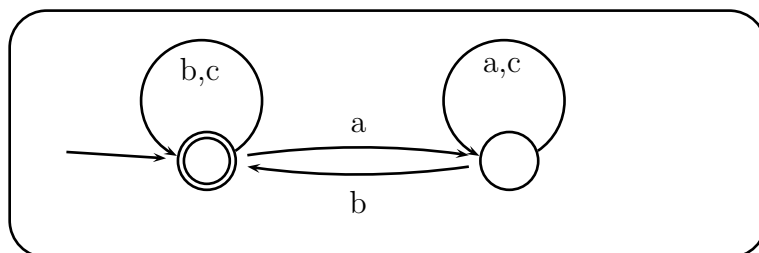
10.2.4 Übungsaufgaben

Übungsaufgabe 10.2.1

Ist L eine ω -reguläre Sprache und K eine reguläre Sprache, dann ist auch KL eine ω -reguläre Sprache.

Übungsaufgabe 10.2.2

Bestimmen Sie die von dem folgenden Büchi Automaten akzeptiert ω -Sprache in dem Alphabet $V = \{a, b, c\}$.



Übungsaufgabe 10.2.3

Der in Abbildung 10.2 gezeigte endliche Automat \mathcal{A} ist nichtdeterministisch.

1. Berechnen Sie den äquivalenten deterministischen endlichen Automaten \mathcal{A}_1 .
2. Bestimmen Sie $L^\omega(\mathcal{A}_1)$.

Übungsaufgabe 10.2.4

Sei \mathcal{A} ein endlicher Automat, so daß für jeden Endzustand q_f und jeden Buchstaben x des Vokabulars $\delta_{\mathcal{A}}(q_f, x) = \emptyset$ gilt. Nach Lemma 10.10 stellt diese Annahme keine Einschränkung der Allgemeinheit dar. Wir benutzen K als Abkürzung für $L(\mathcal{A})$.

Der Automat \mathcal{B} sei wie folgt definiert:

$$\begin{aligned}
 Q_{\mathcal{B}} &= Q_{\mathcal{A}} \\
 s_0^{\mathcal{B}} &= s_0^{\mathcal{A}} \\
 \delta_{\mathcal{B}}(q, x) &= \delta_{\mathcal{A}}(q, x) && \text{falls } q \notin F_{\mathcal{A}}, x \in V \\
 \delta_{\mathcal{B}}(q, x) &= \delta_{\mathcal{A}}(q, x) \cup \delta_{\mathcal{A}}(s_0^{\mathcal{A}}, x) && \text{falls } q \in F_{\mathcal{A}}, x \in V \\
 F_{\mathcal{B}} &= F_{\mathcal{A}}
 \end{aligned}$$

Zeigen Sie, $L^\omega(\mathcal{B}) = K^\omega$.

Übungsaufgabe 10.2.5

Zeigen Sie, daß in der vorangegangenen Übungsaufgabe, Nummer 10.2.4, die Voraussetzung an den Automaten \mathcal{A} unverzichtbar sind.

Übungsaufgabe 10.2.6

Finden Sie endliche Automaten $\mathcal{A}_1, \mathcal{A}_2$, so daß für die akzeptierten Sprachen $K_i = L(\mathcal{A}_i)$ gilt

$$K_1 \cap K_2 = \emptyset \quad \text{aber} \quad \vec{K}_1 \cap \vec{K}_2 \neq \emptyset$$

Übungsaufgabe 10.2.7

Finden Sie einen deterministischen Büchi Automaten \mathcal{A} , so daß das Komplement der von \mathcal{A} akzeptierten Wörter nicht von einem deterministischen Büchi Automaten akzeptiert wird.

Übungsaufgabe 10.2.8

In der Definition 10.19 eines Büchi Automaten wurde nicht geklärt, ob spontane Übergänge zugelassen sind oder nicht. Zeigen Sie: zu jedem endlichen Automaten \mathcal{A} gibt es einen Automaten \mathcal{B} ohne spontane Transitionen mit

$$L^\omega(\mathcal{A}) = L^\omega(\mathcal{B})$$

Übungsaufgabe 10.2.9

Finden Sie zu einem gegebenen (nicht deterministischen) Büchi-Automaten $\mathcal{A} = (S_A, V, s_0^A, \delta_A, F)$ einen Müller-Automaten $\mathcal{M} = (S_M, V, s_0^M, \delta_M, \mathcal{F})$ mit

$$L^\omega(\mathcal{A}) = L^\omega(\mathcal{M})$$

Übungsaufgabe 10.2.10

Zeigen Sie, daß es zu jedem Müller Automaten \mathcal{M} über dem Alphabet V einen komplementären Müller Automaten \mathcal{M}_c gibt, d.h.

$$L^\omega(\mathcal{M}_c) = V^\omega \setminus L^\omega(\mathcal{M})$$

Zusammen mit Lemma 10.35 erhalten wir damit einen alternativen Beweis zu Satz 10.30 für die Abgeschlossenheit ω -regulärer Mengen unter Komplementbildung .

Übungsaufgabe 10.2.11

Sei $\mathcal{A} = (S, V, s_0, \delta, F)$ ein endlicher Automat. Für ein Wort $w \in V^\omega$ definieren wir induktiv Mengen $Q_n(w)$ von Zuständen:

$$\begin{aligned} Q_0(w) &= \{s_0\} \\ Q_{n+1}(w) &= \bigcup_{s \in Q_n(w)} \delta(s, w(n)) \end{aligned}$$

Schließlich sei

$$L^{alt}(A) = \{w \in V^\omega \mid \text{für unendlich viele } n \text{ gilt } Q_n(w) \cap F \neq \emptyset\}$$

Frage: Gilt

$$L^{alt}(A) = L^\omega(A)$$

für alle A .

Übungsaufgabe 10.2.12

Das Alphabet V enthalte mindestens die beiden Buchstaben a und b . Die Menge L_{a3b} bestehe genau aus den Wörtern $w \in V^\omega$, so daß für jedes n mit $w(n) = a$ gilt $w(n+1) = b$ oder $w(n+2) = b$ oder $w(n+3) = b$.

Finden Sie einen Büchi Automaten \mathcal{B}_{a3b} mit

$$L^\omega(\mathcal{B}_{a3b}) = L_{a3b}.$$

Übungsaufgabe 10.2.13

Das Alphabet V enthalte mindestens die beiden Buchstaben a und b . Finden Sie einen Büchi Automaten $\mathcal{B}_{a\infty \rightarrow b\infty}$, so daß $w \in L^\omega(\mathcal{B}_{a\infty \rightarrow b\infty})$ genau dann gilt, wenn w die Bedingung erfüllt:

Wenn a in w unendlich oft vorkommt, dann kommt auch b in w unendlich oft vor.

Diese Eigenschaft kann als Grundmuster für Fairnessbedingungen angesehen werden, wenn durch den Buchstaben a symbolisiert werden soll, daß eine Aktion ausführbar ist und b symbolisiert, daß diese Aktion tatsächlich ausgeführt wird.

Übungsaufgabe 10.2.14

Wir stellen uns die symbolische Modellierung eines Systems vor, in dem eine Aktion A ausführbar sein kann oder nicht. Die Ausführbarkeit von A soll durch die Boolesche Variable a beschrieben werden. Weiterhin kann in dem betrachteten System die Aktion A tatsächlich ausgeführt werden oder nicht. Die Ausführung von A soll durch die Boolesche Variable b beschrieben werden. Es ist nicht ausgeschlossen, daß a und b gleichzeitig wahr sind. Zur Beschreibung dieses Systems in unserem bisherigen formalen Rahmen betrachten wir das Alphabet V mit den Buchstaben $w = \{\}$, $x = \{a\}$, $y = \{b\}$,

$z = \{a, b\}$. Eine Ausführungsfolge des Systems wollen wir *schwach fair* (engl. weakly fair) nennen, wenn b unendlich oft wahr wird, falls von einer Stelle an a immer wahr ist. Entsprechend nennen wir ein Wort $w \in V^\omega$ *schwach fair* wenn aus der Existenz einer Zahl n mit $w(m) \in \{x, z\}$ für alle $m \geq n$ folgt, daß es unendliche viele $k \geq n$ gibt mit $w(k) \in \{y, z\}$. Finden Sie einen Büchi Automaten $\mathcal{B}_{w\text{fair}}$, so daß $L^\omega(\mathcal{B}_{w\text{fair}})$ genau die Menge der schwach fairen Wörter in V^ω ist.

Übungsaufgabe 10.2.15

Zu jedem Automatenalphabet V assoziieren wir die Signatur Σ_V , welche für jeden Buchstaben $a \in V$ ein ein-stelliges Prädikatszeichen P_a enthält. Zusätzlich enthält Σ_V das zwei-stellige Relationszeichen $<$ und das ein-stellige Funktionszeichen $s()$.

Es gibt einen natürlichen Zusammenhang zwischen Wörtern $w \in V^\omega$ und der Struktur \mathcal{M}_w , die wie folgt definiert ist:

$$\begin{aligned} \text{Universum } \mathcal{M}_w &= \mathbb{N} \text{ Menge der natürlichen Zahlen} \\ < &= \text{übliche Ordnung auf } \mathbb{N} \\ s^{\mathcal{M}_w}(n) &= n + 1 \\ P_a^{\mathcal{M}_w}(n) \text{ ist wahr} &\text{ gdw } w(n) = a \end{aligned}$$

Beachten Sie, daß die ersten drei Teile der Definition von \mathcal{M}_w nicht von w abhängen.

Wir fixieren ein Alphabet V , das mindestens a, b, c enthält. Finden Sie eine Formel F der Logik zweiter Stufe, so daß für alle $w \in V^\omega$ gilt $\mathcal{M}_w \models F$ genau dann, wenn $w \in \{ab, bbc\}^\omega$.

Übungsaufgabe 10.2.16

Mit der Notation aus Aufgabe 10.2.15:

1. Finden Sie eine Formel der Prädikatenlogik erster Stufe F_1 , so daß für alle $w \in V^\omega$ gilt: $\mathcal{M}_w \models F_1$ gdw $w \in L_{a3b}$, wobei L_{a3b} in Aufgabe 10.2.12 definiert wurde.
2. Finden Sie eine Formel der Prädikatenlogik erster Stufe F_2 , so daß für alle $w \in V^\omega$ gilt: $\mathcal{M}_w \models F_2$ gdw w liegt in der in Aufgabe 10.2.13 definierten Sprache.
3. Dasselbe für die in Aufgabe 10.2.15 definierte Sprache. Die gesuchte Formel soll F_3 heißen.

Kapitel 11

Modellprüfung

11.1 Einführung

Um einen Einstieg in die Probleme und grundlegenden Konzepte temporaler Logiken zu finden, betrachten wir in diesem Abschnitt in aller Ausführlichkeit ein Beispiel aus dem Gebiet der Verifikation von Kommunikationsprotokollen. Bei der Ausarbeitung dieses Materials habe ich viel dem dem Buch [Hol04, Chapter 14] profitiert. Holzmann, und damit auch die folgende Darstellung, orientiert sich an dem ITU Standard SS7 (Signaling System 7) für öffentliche Telefonnetze (public-switched telephone network). Fairerweise muß man jedoch eingestehen, daß hier nur ein minimaler Teil von SS7 überhaupt zur Sprache kommt.

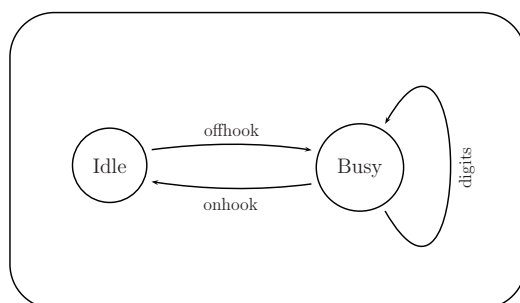


Abbildung 11.1: Einfaches Modell eines Telefonteilnehmers

In Abbildung 11.1 ist ein einfaches Modell eines Telefonteilnehmers (engl. subscriber) gezeigt. Die einzigen *Aktionen* in dem Modell sind *Hörer abnehmen* (offhook), *Hörer auflegen* (onhook) und *Wählen* (digits). Es wird nicht unterschieden ob der Teilnehmer selbst anruft oder angerufen wird. Wir beschränken uns in dem ganzen Szenarium auf den Teilnehmer, der das Gespräch initiiert. Nach dem der Benutzer den Hörer abgehoben und gewählt hat wird auch nicht weiter unterschieden, was passiert. Kommt eine Verbindung zustande? Ist die gerufene Nummer besetzt? Nimmt der Teilnehmer am anderen Ende der Leitung ab? Das alles ist in dem einen Zustand *Busy* zusammengefasst.

Ein zweiter Bestandteil des Kommunikationsmodells ist die lokale Vermittlungstelle. Ein Automat dafür ist in Abbildung 11.2 zu sehen. In dem Abschnitt 10.1 über Automaten hatten wir ein abstraktes Alphabet V betrachtet, das zur Markierung von Kanten diente. Der Kantenmarkierungen in Abbildung 11.2 sind dagegen um Einiges konkreter. Ausdrücke der Form $tpc?xxx$ besagen, daß die Vermittlungsanlage die Nachricht xxx über den Kanal tpc empfangen hat. Folgendes Vokabular für Nachrichten steht zur Verfügung:

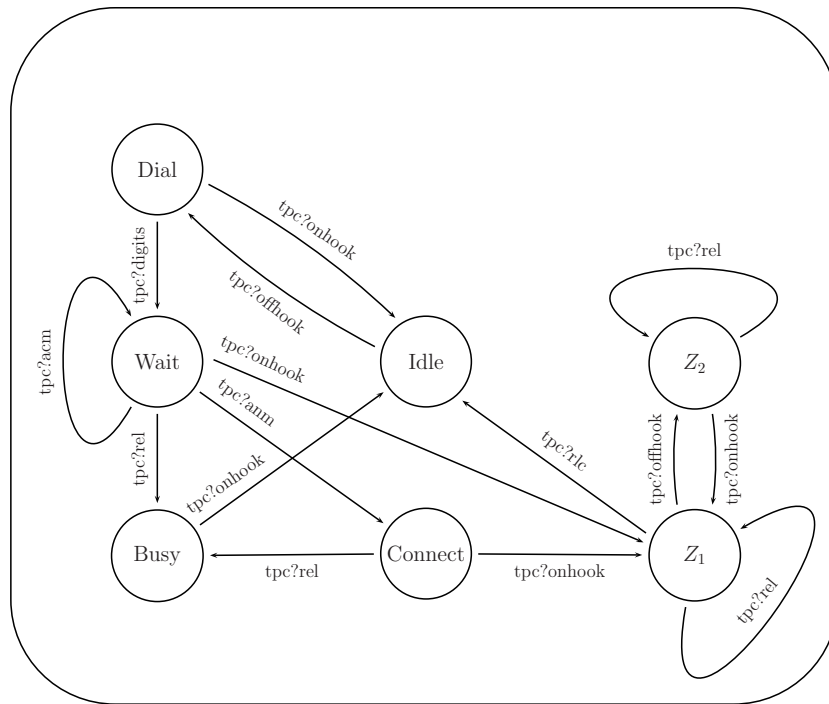


Abbildung 11.2: Einfacher Automat einer Telefonvermittlung

Nachricht	Bedeutung
offhook	Hörer wurde abgehoben
onhook	Hörer wurde aufgelegt
digits	Nummer wurde gewählt
iam	initial address message
acm	address complete message
rel	Einleitung des Verbindungsabbaus (release)
anm	End-zu-Endverbindung hergestellt
rlc	Quittierung des Verbindungsabbaus

Woher die empfangenen Nachrichten kommen kann man aus dem Automaten nicht erkennen. Das weiss die Anlage in Wirklichkeit auch nur, weil zu den Nachrichten noch Datenfelder gehören, in denen entsprechende Information geliefert wird. In unserem Szenarium können wir aber erklären woher die Nachrichten kommen. Die ersten drei **offhook**, **onhook** und **digits** kommen vom Teilnehmer. Wenn er den Hörer abhebt, geht bei der Vermittlungsanlage die Nachricht **tpc?offhook** ein. In dem Kommunikationsprozess sind neben den bisher genannten Akteuren, dem Teilnehmer und der lokalen Vermittlungsstelle, weiterhin beteiligt, ein Zielteilnehmer, eine Zielvermittlungsstelle

und auch Zwischenknoten des Verbindungsweges. Die Nachrichten *acm*, *anm*, *rel* und *rlc* werden von der Zielvermittlungsstelle an die lokale Vermittlung geschickt. Die *initial address message* wird, in unserem Szenarium, von der lokalen Vermittlung an die Zielvermittlungsstelle geschickt. In Abbildung 11.2 sind nur die Bedingungen für die Ausführung eines Zustandsübergangs eingetragen (engl. *guards*), die Aktionen, die zusätzlich zur Zustandsänderung passieren sind der Übersichtlichkeit halber weggelassen worden.

Zwei typische Abläufe des Vermittlungsautomaten ist in Abbildung 11.3 zu sehen. 1. Beispiel: Zustand ist *idle*, Teilnehmer hebt Hörer ab, Zustand ist *dial*, Teilnehmer wählt, Zustand ist *wait*, der gewünschte Teilnehmer wird erreicht und nimmt seinerseits den Hörer ab, Zustand ist *connect*, der Zielteilnehmer legt den Hörer auf, Zustand ist *busy*, Teilnehmer legt den Hörer auf, Zustand ist *idle*. Der Beispielablauf 2 entsteht, wenn der das Gespräch initiiierende Teilnehmer zuerst den Hörer auflegt.

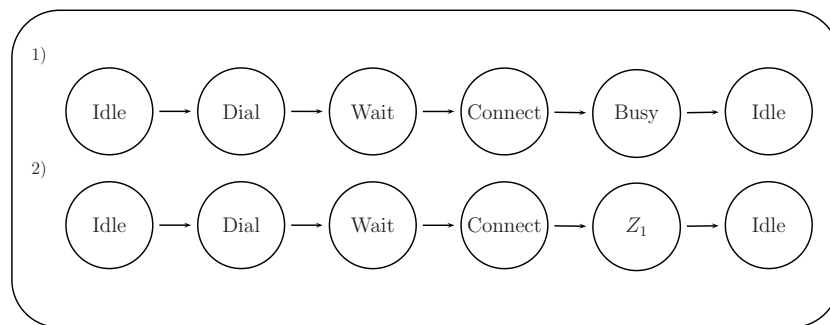


Abbildung 11.3: Beispielablauf des Automaten aus Abb.11.2

Eine wichtige Aufgabe der formalen Analyse von Kommunikationsprotokollen ist es Aussagen über aller möglichen Abläufe eines Systems zu machen. Zum Beispiel, daß zwischen zwei Auftreten des Zustand *connect* mindestens einmal der Zustand *idle* vorkommen muß. Wie sollen wir Abläufe formal beschreiben? Wir fassen dazu die Kreise in Abbildung 11.3 als abstrakte Zeitpunkte auf. Abstrakt deswegen, weil wir nicht messen wollen wieviel Zeit vergeht um von einem Zeitpunkt zum anderen zu kommen. Die zeitliche Ordnung in vorher und nachher und die Tatsache, das jeder Zeitpunkt einen unmittelbar folgenden hat sind hier ausschlaggebend. In Abbildung 11.3 ist nur ein endliches Anfangsstück eines möglichen Ablaufs gezeigt. Der Vermittlungsautomat hört ja nicht auf zu arbeiten. Deswegen wollen wir auch in unserer Formalisierung annehmen, daß die Folge der abstrakten Zeitpunkte keinen letzten Zeitpunkt aufweist. Mit anderen Worten haben wir uns gerade auf die Ordnungsstruktur der natürlichen Zahlen, $(\mathbb{N}, <)$, festgelegt. Für jeden der sieben Zustände des Automaten aus Abbildung 11.2 führen wir ein

aussagenlogisches Atom ein und können dann einen Ablauf beschreiben als eine Kopie der natürlichen Zahlen, aufgefasst als Folge abstrakter Zeitpunkte, wobei zu jedem genau eines der aussagenlogischen Atome wahr ist. Es kann sinnvoll sein noch weitere aussagenlogische Atome einzuführen, z.B. in unserem Szenarium ein Atom OH , das in einem Zeitpunkt wahr ist, wenn zu diesem Zeitpunkt der Hörer des anrufenden Teilnehmers abgehoben ist. In dem Abschnitt werden wir auch zeigen, wie die temporale aussagenlogische Sprache LTL aus Abschnitt 8.1 zur Formulierung temporaler Eigenschaften benutzt werden kann. Insbesondere werden *omega-Strukturen* als Beschreibungsstruktur benutzt.

11.2 Büchi Automaten und LTL

Wir kommen nun zu einem wichtigen Punkt in der Entwicklung der temporalen Logik: Wir wollen einen Zusammenhang herstellen zwischen endlichen Automaten auf der einen Seite und der temporalen Logik LTL auf der anderen.

Wir erinnern zunächst daran, daß wir in Definition 10.19 die Menge $L^\omega(\mathcal{A})$ der von \mathcal{A} akzeptierten unendlichen Wörter definiert hatten. Die in solchen Wörtern vorkommenden Buchstaben aus dem Alphabet V , die auch als Kantenmarkierungen des Automaten auftreten, hatten bisher keine besondere Bedeutung. Das soll sich jetzt ändern. Sei Σ die Menge aller aussagenlogischen Atome einer temporalen Logik, siehe Abbildung 8.3. Wir setzen jetzt das Automatenvokabular $V = 2^\Sigma$. Ein ω -Wort $\xi \in V^\omega$ ist jetzt gleichzeitig eine omega-Struktur. Jetzt können wir das nächste Ziel, realisiert durch Satz 11.3, formulieren. Zu jeder LTL-Formel A wollen wir einen Büchi-Automaten \mathcal{A}_B finden mit

$$L^\omega(\mathcal{A}_B) = \{\xi \in V^\omega \mid \xi \models A\}$$

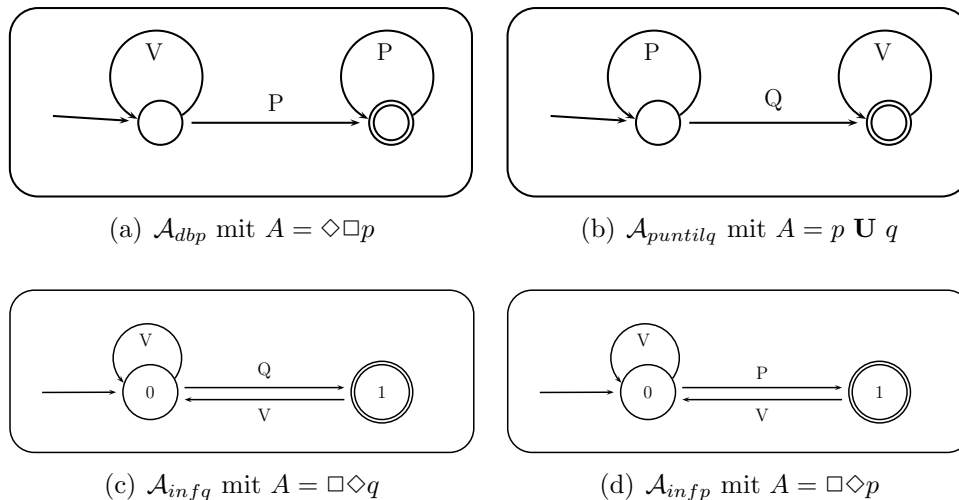


Abbildung 11.4: Automat für LTL-Formel A

Beispiel 11.1

Sei Σ ein Menge aussagenlogischer Atome mit $p, q \in \Sigma$, $V = 2^\Sigma$ das entsprechende Automatenvokabular und $P = \{b \in V \mid p \in b\}$, $Q = \{b \in V \mid q \in b\}$.

Dann gilt für die Automaten aus der Abbildung 11.4:

$$\begin{aligned}
L^\omega(\mathcal{A}_{dbp}) &= \{\xi \in V^\omega \mid \xi \models \diamond \square p\} \\
L^\omega(\mathcal{A}_{puntilq}) &= \{\xi \in V^\omega \mid \xi \models p \mathbf{U} q\} \\
L^\omega(\mathcal{A}_{infp}) &= \{\xi \in V^\omega \mid \xi \models \square \diamond p\} \\
L^\omega(\mathcal{A}_{infpq}) &= \{\xi \in V^\omega \mid \xi \models \square \diamond q\}
\end{aligned}$$

Lemma 11.2

Seien $\mathcal{A}_1 = (S_1, V, s_1^0, \delta_1, F_1), \mathcal{A}_2 = (S_2, V, s_2^0, \delta_2, F_2)$ Büchi-Automaten, C_1, C_2 LTL-Formeln mit $\mathcal{A}_1 \models C_1$ und $\mathcal{A}_2 \models C_2$. Dann gibt es einen Büchi-Automaten \mathcal{C} mit

$$\mathcal{C} \models C_1 \wedge C_2$$

Wir benutzen gelegentlich die Notation $\mathcal{C} = \mathcal{A}_1 \cap \mathcal{A}_2$.

Beweis Nach Satz 10.24 ist die Richtigkeit des Lemmas schon klar. Der Beweis von Satz 10.24 war jedoch indirekt geführt worden. Wir wollen hier eine explizite Konstruktion von \mathcal{C} aus \mathcal{A}_1 und \mathcal{A}_2 vorführen. Wir beginnen mit einem Beispiel. Ein erster Versuch könnte sein, für \mathcal{C} das direkte Produkt von \mathcal{A} und \mathcal{B} einzusetzen. Das direkte Produkt von \mathcal{A}_{infp} und \mathcal{A}_{infpq} ist in Abbildung 11.5 zu sehen.

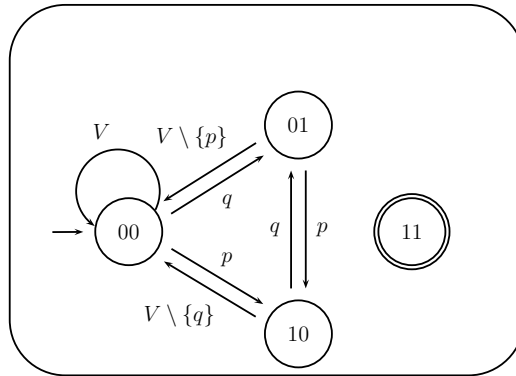


Abbildung 11.5: Erster Versuch eines Automaten für $(\square \diamond p) \wedge (\square \diamond q)$

Man sieht, daß dieser Automat überhaupt kein ω -Wort akzeptiert. In Abbildung 11.6 ist der korrekte Automat zu sehen. Wir haben die beiden unerreichbaren Zuständen dieses Automaten gleich ganz weggelassen.

Im allgemeinen kann $\mathcal{C} = (S, V, s^0, \delta, F)$ wie folgt konstruiert werden.

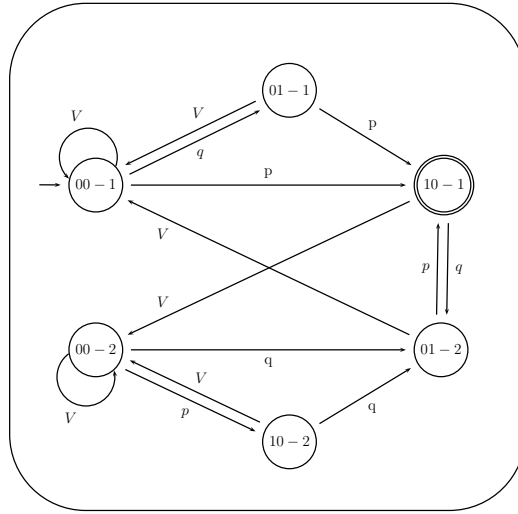


Abbildung 11.6: Ein Automaten für $(\square \diamond p) \wedge (\square \diamond q)$

$$S = S_1 \times S_2 \times \{1, 2\}$$

$$s^0 = (s_1^0, s_2^0, 1)$$

$$F = F_1 \times S_2 \times \{1\}$$

für alle $s_1 \in S_1, s_2 \in S_2, i \in \{1, 2\}$

$$(t_1, t_2, i) \in \delta^0((s_1, s_2, i), a) \Leftrightarrow t_1 \in \delta_1(s_1, a) \text{ und } t_2 \in \delta_2(s_2, a)$$

falls $s_1 \in F_1$

$$(t_1, t_2, 2) \in \delta^1((s_1, s_2, 1), a) \Leftrightarrow t_1 \in \delta_1(s_1, a) \text{ und } t_2 \in \delta_2(s_2, a)$$

falls $s_2 \in F_2$

$$(t_1, t_2, 1) \in \delta^2((s_1, s_2, 2), a) \Leftrightarrow t_1 \in \delta_1(s_1, a) \text{ und } t_2 \in \delta_2(s_2, a)$$

$$\delta((s_1, s_2, 1), a) = \delta^0((s_1, s_2, 1), a) \cup \delta^1((s_1, s_2, 1), a)$$

$$\delta((s_1, s_2, 2), a) = \delta^0((s_1, s_2, 2), a) \cup \delta^2((s_1, s_2, 1), a)$$

Die Idee dahinter ist, den Produktautomaten zweimal zu benutzen. Die Zustände von Teil 1 sind alle Tripel $(s_1, s_2, 1)$ mit $s_i \in S_i$, die Zustände von Teil 2 sind entsprechend die Tripel $(s_1, s_2, 2)$. Innerhalb einer der beiden Teile sind auf jeden Fall die Übergänge des Produktautomaten möglich, das wird durch δ^0 definiert. Ist s_1 ein Endzustand des ersten Automaten, so gibt es zusätzlich Transitionen von $(s_1, s_2, 1)$ in den 2. Teil von \mathcal{C} . Das beschreibt δ^1 . Ist s_2 ein Endzustand des zweiten Automaten, dann gibt es außerdem Transitionen von $(s_1, s_2, 2)$ in den ersten Teil. Das wird durch δ^2 beschrieben.

Fortsetzung des Beweises von Lemma 11.2 Ist $r = r_1, r_2, \dots, r_n \dots$ eine Berechnungsfolge des Automaten \mathcal{C} , wobei $r_n = (s_n^1, s_n^2, j_n) \in S$, dann ist $s^1 = s_1^1, s_2^1, \dots, s_n^1 \dots$ eine Berechnungsfolge für \mathcal{A}_1 und $s^2 = s_1^2, s_2^2, \dots, s_n^2 \dots$

eine Berechnungsfolge für \mathcal{A}_2 . Ist r eine akzeptierende Berechnungsfolge, dann muß eine Endzustand $(s, t, 1)$ mit $s \in F_1$ unendlich oft als r_n vorkommen. Damit wäre auf jeden Fall s^1 eine akzeptierende Berechnungsfolge für \mathcal{A}_1 . Wird aber ein Zustand $(s, t, 1)$ erreicht, dann ist der nächste Zustand von der Form $(s', t', 2)$. Von Zuständen mit der Endziffer 2 kommt man zu einem Zustand mit der Endziffer 1 nur wieder zurück, indem ein Zustand der Form $(s, t, 2)$ mit $t \in F_2$ durchlaufen wird. Daraus folgt, daß auch in s^2 ein Endzustand aus F_2 unendlich oft vorkommen muß. ■

Satz 11.3

Zu jeder LTL-Formel B gibt es einen Büchi-Automaten \mathcal{A}_B mit

$$L^\omega(\mathcal{A}_B) = \{\xi \in V^\omega \mid \xi \models B\}.$$

Beweis: Wir fixieren eine Menge P von aussagenlogischen Atomen und eine LTL-Formel $B \in LTLFor(P)$. In der Formel B dürfen die temporalen Operatoren \mathbf{U} , \mathbf{V} und X vorkommen. Wir können annehmen, daß B in Negationsnormalform vorliegt, d.h. daß Negationszeichen \neg nur for atomaren Formeln vorkommt. Wir geben explizit einen erweiterten Büchi Automaten $\mathcal{A}_B = (V, S, S_0, \delta, \mathcal{F})$ mit eine Menge von Anfangszuständen an, so daß

$$L^\omega(\mathcal{A}_B) = \{\xi \in V^\omega \mid \xi \models B\}$$

gilt. Der Rest des Beweises wird dann durch die Lemmata 10.31 und 10.33 erledigt.

Wir kommen zur Definition \mathcal{A}_B

Vokabular $V = 2^P$

Zustände Sei $subF(B)$ die Menge aller Teilformeln von B .

$$S = \left\{ s \subseteq subF(B) \mid \begin{array}{l} \mathbf{0} \notin s \\ \text{wenn } (C_1 \wedge C_2) \in s \text{ dann } C_1 \in s \text{ und } C_2 \in s \\ \text{wenn } (C_1 \vee C_2) \in s \text{ dann } C_1 \in s \text{ oder } C_2 \in s \end{array} \right\}$$

Anfangszustände $S_0 = \{s \in S \mid B \in s\}$

Übergangsfunktion Für $s, t \in S$ und $a \in 2^P$ gilt $t \in \delta(s, a)$ wenn alle folgenden Bedingungen erfüllt sind

1. Für alle $p \in P$ mit $p \in s$ gilt $p \in a$.
2. Für alle $p \in P$ mit $\neg p \in s$ gilt $p \notin a$.
3. Falls $X A \in s$ dann $A \in t$.
4. Falls $A \mathbf{U} B \in s$, dann gilt $B \in s$ oder $A \in s$ und $A \mathbf{U} B \in t$.
5. Falls $A \mathbf{V} B \in s$, dann $(B \in s \text{ und } A \in s)$ oder $B \in s$ und $A \mathbf{V} B \in t$.

Endzustandsmengen Seien $E_i = A_i \mathbf{U} B_i$ für $1 \leq i \leq k$ alle Formeln der Form $A \mathbf{U} B$, die in $\text{sub}F(B)$ vorkommen. Dann ist $\mathcal{F} = \{\mathcal{F}_1, \dots, \mathcal{F}_k\}$ mit

$$\mathcal{F}_i = \{s \in S \mid E_i \notin s \text{ oder } E_i \in s \text{ und } B_i \in s\}$$

Damit ist die Definition des Automaten \mathcal{A}_B abgeschlossen und wir gehen nun daran zu beweisen, daß er die gewünschten Eigenschaften hat.

Behauptung 1 $L^\omega(\mathcal{A}_B) \subseteq \{\xi \in V^\omega \mid \xi \models B\}$

Sei $\xi \in L^\omega(\mathcal{A}_B)$ und $r = r_0, \dots, r_n, \dots$ eine akzeptierende Berechnung des Automaten \mathcal{A}_B für ξ (siehe Definition 10.19). Dann zeigen wir für jede Formel $C \in \text{sub}F(B)$ und jedes $i \geq 0$

$$C \in r_i \Rightarrow \xi_i \models C$$

Da in jedem Fall $r_0 \in S_0$ gilt und B eine Element jedes Anfangszustands ist, folgt aus der Behauptung 1 insbesondere $\xi_0 = \xi \models B$.

Beweis von Behauptung 1 Durch strukturelle Induktion über die Komplexität der Formel C . Nach der Definition einer Berechnungsfolge gilt $r_{i+1} \in \delta(r_i, \xi(i))$, davon und von der obigen Definition der Transitionsrelation δ werden wir intensiven Gebrauch machen.

C ist Literal

Aus den Bedingungen 1 und 2 der Definition von δ folgt schon die Behauptung. Man muß sich an dieser Stelle erinnern, daß $p \in \xi(i)$ gleichbedeutend mit $\xi(i) \models p$ und $p \notin \xi(i)$ gleichbedeutend mit $\xi(i) \models \neg p$ ist.

$$C = C_1 \wedge C_2 \text{ oder } C = C_1 \vee C_2$$

In diesem Fall greifen wir auf die Definition der Menge der Zustände S zurück. Wenn $(C_1 \wedge C_2) \in r_i$ ($(C_1 \vee C_2) \in r_i$) dann wurde verlangt daß $C_1 \in r_i$ und $C_2 \in r_i$ ($C_1 \in r_i$ oder $C_2 \in r_i$) gilt. Nach Induktionsvoraussetzung folgt $\xi_i \models C_1$ und $\xi_i \models C_2$ ($\xi_i \models C_1$ oder $\xi_i \models C_2$) und damit auch $\xi_i \models C_1 \wedge C_2$ ($\xi_i \models C_1 \vee C_2$).

$$C = C_1 \text{ U } C_2$$

Wir behaupten, daß es ein $i \leq j$ gibt mit $C_2 \in r_j$ und $C_1 \in r_{j'}$ für alle j' mit $i \leq j' < j$. Nach Definition von δ könnte $C_2 \in r_i$ gelten und wir hätten j gefunden für $j = i$. Anderenfalls gilt $C_1 \in r_i$ und $C_1 \text{ U } C_2 \in r_{i+1}$ und wir können dasselbe Argumente für $i + 1$ anstelle von i wiederholen. Wenn wir in dieser Iteration auf ein erstes j stoßen mit $C_2 \in r_j$ dann haben wir es geschafft. Andererseits können wir noch nicht die Möglichkeit ausschließen, daß für alle $j \geq i$ gilt $C_2 \notin r_j$. Wir wir gesehen haben folgt daraus $C_1 \text{ U } C_2 \in r_j$ für alle $j \geq i$. An dieser Stelle erinnern wir uns daran, daß die akzeptierende Berechnung r unendlich oft die Finalmenge $\mathcal{F}_k = \{s \in S \mid C_1 \text{ U } C_2 \notin s \text{ oder } C_1 \text{ U } C_2 \in s \text{ und } C_2 \in s\}$ trifft. (Wir haben angenommen, daß $C_1 \text{ U } C_2 = E_k$.) Insbesondere gibt es ein j mit $i \leq j$ mit $r_j \in \mathcal{F}_k$. Beide Annahmen $C_1 \text{ U } C_2 \in r_j$ und $C_2 \notin r_j$ zusammengekommen, stehen dazu im Widerspruch. Also muß es j mit $j \geq i$ und $C_2 \in r_j$ geben. Wir wählen ein kleinstes j mit dieser Eigenschaft. Nach den bisher gemachten Annahmen folgt jetzt $C_2 \in r_j$ und $C_1 \in r_{j'}$ für $i \leq j' < j$. Nach Induktionsvoraussetzung folgt $\xi_j \models C_2$ und $\xi_{j'} \models C_1$ für alle $i \leq j' < j$. Also insgesamt $\xi_i \models C_1 \text{ U } C_2$.

$$C_1 \text{ V } C_2$$

Aus Teil 5 der Definition von δ folgt auf jeden Fall für alle $j \geq i$, wenn $C_1 \text{ V } C_2 \in r_j$, dann auch $C_2 \in r_j$. Somit haben wir auf jeden Fall schon einmal $C_2 \in r_i$ sicher. Aus der Induktionsvoraussetzung folgt damit $\xi_i \models C_2$. Gilt jetzt für ein $j + 1 > i$ die Aussage $\xi_{j+1} \models \neg C_2$, dann folgt mit der Induktionsvoraussetzung $C_2 \notin r_{j+1}$ und nach der obigen Überlegung auch $C_1 \text{ V } C_2 \notin r_{j+1}$. Nach Teil 5 muß daher $C_1 \in r_j$ gelten, was nach Induktionsvoraussetzung wieder $\xi_j \models C_1$ liefert. Insgesamt ist damit $\xi \models C_1 \text{ V } C_2$ gezeigt.

Behauptung 2 $\{\xi \in V^\omega \mid \xi \models B\} \subseteq L^\omega(\mathcal{A}_B)$

d.h. zu jedem $\xi \in V^\omega$ mit $\xi \models B$ gibt es eine akzeptierende Berechnung $r = r_0, \dots, r_n \dots$ von \mathcal{A}_B mit $r_{i+1} \in \delta(r_i, \xi(i))$ für alle $i \geq 0$.

Beweis von Behauptung 2 Wir setzen $r_i = \{C \in \text{subF}(B) \mid \xi_i \models C\}$. Man prüft leicht nach, daß für alle $i \geq 0$ gilt $r_i \in S$, $r_{i+1} \in \delta(r_i, \xi(i))$ und schließlich auch, daß $r = r_0, \dots, r_n, \dots$ eine akzeptierende Berechnungsfolge für \mathcal{A}_B . ■

Korollar 11.4

Erfüllbarkeit und Allgemeingültigkeit von LTL Formeln ist entscheidbar.

Beweis: Nach dem im vorangegangenen Satz beschriebenen Verfahren konstruiert man die Büchi Automaten \mathcal{A}_B und $\mathcal{A}_{\neg B}$. Es gilt

$$\begin{aligned} B \text{ ist erfüllbar} &\Leftrightarrow L^\omega(\mathcal{A}_B) \neq \emptyset \\ B \text{ ist allgemeingültig} &\Leftrightarrow L^\omega(\mathcal{A}_{\neg B}) = \emptyset \end{aligned}$$

Nach Satz 10.20 sind wir fertig. ■

11.2.1 Übungsaufgaben

Übungsaufgabe 11.2.1

Finden Sie einen Büchi-Automaten \mathcal{A}_V , so daß $L^\omega(\mathcal{A}_V)$ genau die omega-Strukturen sind, die $p \mathbf{V} q$ erfüllen.

Übungsaufgabe 11.2.2

Sei Σ eine aussagenlogische Signatur, mit $p, q, r \in \Sigma$. Finden Sie einen Büchi-Automaten \mathcal{B}_{UU} , der genau die omega-Strukturen (in der üblichen Weise als ω -Wörter repräsentiert) akzeptiert, welche die LTL-Formel $p \mathbf{U} q \mathbf{U} r$ erfüllen.

Übungsaufgabe 11.2.3

In Abb. 11.5 wird ein Automat gezeigt, den genau die ω -Wörter akzeptiert, in denen sowohl p als auch q unendlich oft vorkommen. Der Automat diene als Beispiel für die allgemeine Konstruktionsmethode aus Lemma 11.2. Es gibt wesentlich einfachere Büchi Automaten für diese Sprache. Finden Sie einen mit 3 Zuständen.

Übungsaufgabe 11.2.4

Im Beweis von Lemma 11.2 haben wir schon ein Gegenbeispiel gesehen zur Gleichung $L^\omega(\mathcal{A}_1 \times \mathcal{A}_1) = L^\omega(\mathcal{A}_1) \cap L^\omega(\mathcal{A}_1)$.

1. Finden Sie ein weiteres möglichst einfaches Gegenbeispiel.

2. Zeigen Sie, daß die Inklusion $L^\omega(\mathcal{A}_1 \times \mathcal{A}_1) \subseteq L^\omega(\mathcal{A}_1) \cap L^\omega(\mathcal{A}_1)$ immer gilt.
3. Zeigen Sie, daß $L^\omega(\mathcal{A}_1 \times \mathcal{A}_1) = L^\omega(\mathcal{A}_1) \cap L^\omega(\mathcal{A}_1)$ gilt, wenn in \mathcal{A}_1 jeder Zustand ein Endzustand ist.

11.3 Modellprüfung für LTL

11.3.1 Problemstellung

Generell wird eine Modellprüfungsaufgabe bestimmt durch die Angabe eines Modells \mathcal{M} und einer Aussagen ϕ . Die Lösung der Aufgabe besteht in der Antwort, ob ϕ für \mathcal{M} wahr ist oder nicht. Auf dieser allgemeinen Stufe lässt sich fast jedes Problem in eine Modellprüfungsaufgabe einkleiden. Wir konzentrieren uns hier auf den Spezialfall, daß Modelle durch Büchi-Automaten gegeben werden und die Aussagen LTL-Formeln sind. Mehr noch, es muß der in Abschnitt 8.1, Seite 317 beschriebene Zusammenhang zwischen dem Automatenvokabular und den LTL-Atomen gelten, d.h. ist Σ die Menge aller aussagenlogischen Atome einer temporalen Logik, dann ist das Automatenvokabular für die Kantenmarkierungen $V = 2^\Sigma$. Es bleibt noch genauer zu erklären, was es heißen soll, daß ein Büchi-Automat \mathcal{A} eine LTL Formel ϕ erfüllt. Damit meinen wir, daß für jedes von \mathcal{A} akzeptierte ω -Wort ξ gilt $\xi \models \phi$. Durch die spezielle Wahl des Automatenvokabulars haben wir sichergestellt, daß ξ eine ω -Struktur beschreibt. Wir führen dafür die neue Notation ein

$$\mathcal{A} \models \phi \quad \text{gdw} \quad \text{für alle } \xi \in L^\omega(\mathcal{A}) \text{ gilt } \xi \models \phi$$

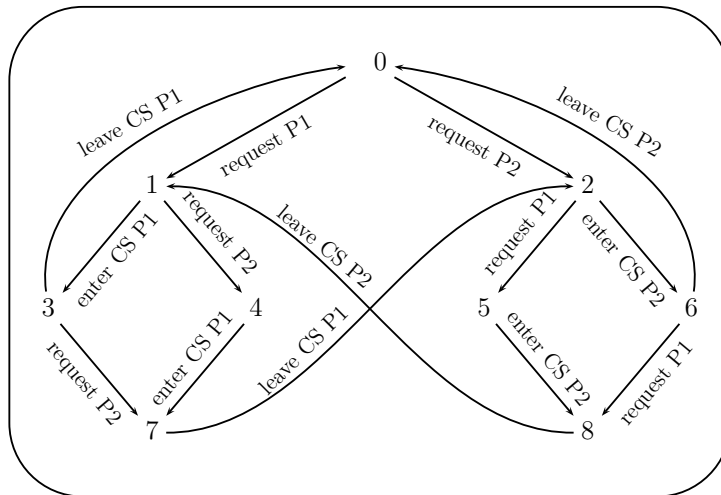


Abbildung 11.7: Ereignis-basiertes Automatenmodell

Als ein Beispiel für ein Modell betrachten wir Abbildung 11.7. Sie zeigt ein Modell zur Regelung des gemeinsamen Zugriffs mehrerer Prozesse, hier sind

es zwei, auf eine gemeinsame Resource. Das damit gelöste Problem ist im Englischen unter der Bezeichnung *mutual exclusion problem* bekannt und dient oft als akademisches Beispiel. Der Automat in Abbildung 11.7 trägt als Kantenmarkierungen Ereignisse, was das intuitive Verständnis sicherlich erleichtert. Für die formale Analyse mit Hilfe von LTL müssen wir allerdings einen etwas anderen Blickwinkel einnehmen. Wir brauchen als erstes eine Menge Σ aussagenlogischer Atome. Im vorliegenden Fall entscheiden wir uns für die folgende Auswahl, wobei in unserem Beispiel $i \in \{1, 2\}$:

- N_i Prozeß i befindet sich in einer nichtkritischen Region
- T_i Prozeß i befindet sich in der Anmeldephase
- C_i Prozeß i befindet sich in einer kritischen Region

Die Kantenmarkierungen des Automaten aus Abbildung 11.7 müssen jetzt entsprechend angepasst werden, um den Zusammenhang zwischen dem aussagenlogischen Vokabular Σ und den Aktionen des Automaten herzustellen. Als Automatenvokabular V soll, wie schon gesagt, $V = 2^\Sigma$ benutzt werden. Der Zusammenhang zwischen den ereignisorientierten Labels und den Markierungen aus V erhält man dadurch, daß für eine Kante von s_1 nach s_2 die Markierung mit dem Ereignis e ersetzt wird mit der Teilmenge von Atomen, die in s_2 nach Ausführung der Aktion e wahr sind. Das ergibt die in Abbildung gezeigten Kantenmarkierungen.

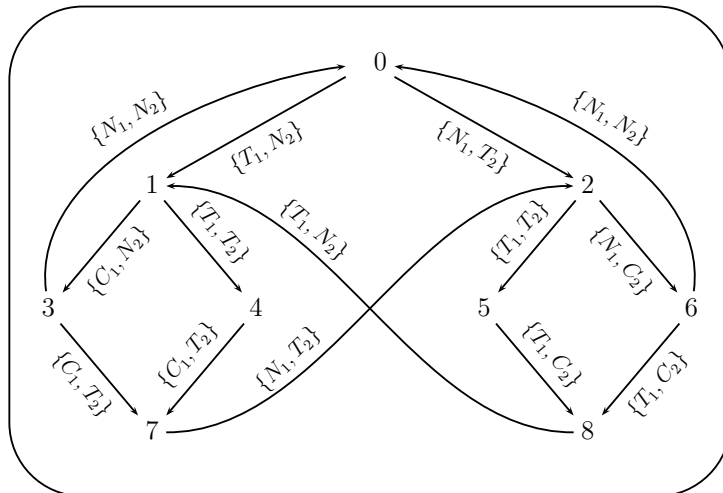


Abbildung 11.8: Aussagenbasiertes Automatenmodell

Wollte man das Beispiel mit dem vollständigen Automaten aus Abbildung 11.7 durchführen würde das Bild sehr schnell sehr undurchsichtig werden.

Wir beschränken uns daher auf den abgespeckten Automaten aus Abbildung 11.9. Das simplifiziert zwar die Aufgabenstellung bis zur Trivialität, aber der Grundgedanke der gesamten Vorgehensweise sollte so besser erkennbar sein.

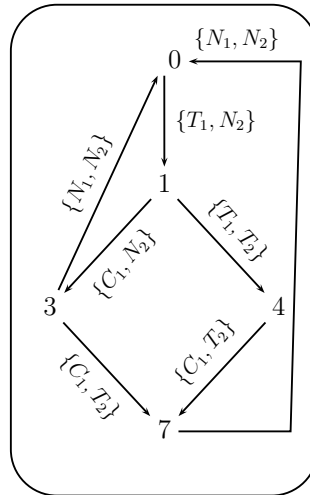


Abbildung 11.9: Reduziertes Automatenmodell \mathcal{A}_{me}

Nachdem das zu analysierende Automatenmodell feststeht, wenden wir uns den Eigenschaften zu, die wir gerne verifizieren möchten. Eine naheliegende Forderung wäre, daß jeder Prozeß, der sich für die Nutzung der exklusiven Resource anmeldet, schließlich auch den Zugang erhält. Das läßt sich als LTL-Formel über der Signatur Σ ausdrücken als

$$\mathcal{A}_{me} \models \Box(T_i \rightarrow \Diamond C_i) \quad ?$$

Wir erinnern noch einmal daran was $\mathcal{A}_{me} \models \Box(T_i \rightarrow \Diamond C_i)$ bedeutet. Für jede akzeptierte Berechnungsfolge s gilt für die omega-Struktur ξ_s , die s zugeordnet ist $\xi_s \models \Box(T_i \rightarrow \Diamond C_i)$. In dem einfachen Beispielautomaten \mathcal{A}_{me} ist die Menge der Finalzustände leer, also ist jede Folge von Zuständen, die den Kanten des Automaten folgt, eine akzeptierte Berechnungsfolge. Somit sind

$$\begin{aligned} s_1 &= 0 \ 1 \ 3 \ 0 \ 1 \ \dots \\ s_2 &= 0 \ 1 \ 3 \ 7 \ 0 \ \dots \\ s_3 &= 0 \ 1 \ 4 \ 7 \ 0 \ \dots \end{aligned}$$

Anfangsstücke akzeptierender Berechnungsfolgen. Bezeichnen wir mit ξ^i die omega-Struktur zu s_i dann sieht man leicht

$$\begin{aligned} \xi^1 &\models T_1 & \xi_1^1 &\models C_1 \\ \xi^2 &\models T_1 & \xi_1^2 &\models C_1 \\ \xi^3 &\models T_1 & \xi_2^3 &\models C_1 \end{aligned}$$

11.3.2 Methode

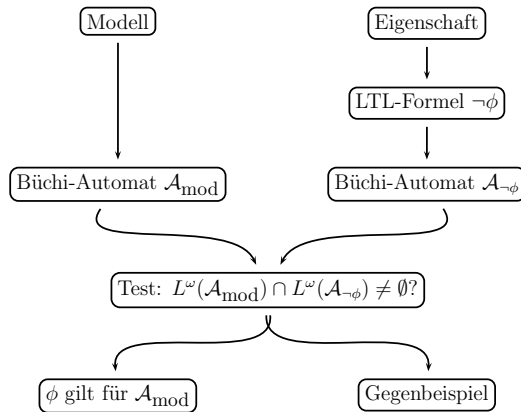


Abbildung 11.10: Übersicht über LTL Modellprüfung

Abbildung 11.10 zeigt schematisch den Ablauf eine Modellprüfung. Wir werden anhand des Beispiels aus Abschnitt 11.3.1 diesen Ablauf Schritt für Schritt nachvollziehen.

Wie in dem Übersichtsdiagramm in Abbildung 11.10 gezeigt, wird mit der Negation der zu verifizierenden Eigenschaft gearbeitet, d.h. in unserem Beispiel mit

$$\diamond(T_i \wedge \square \neg C_i)$$

Wir wollen uns hier auf die Aussage für den ersten Prozeß konzentrieren: $\diamond(T_1 \wedge \square \neg C_1)$.

Der nächste Schritt besteht darin, einen Büchi-Automaten \mathcal{B}_{me} zu konstruieren, der im Sinne von Satz 11.3, die LTL-Formel $\diamond(T_1 \wedge \square \neg C_1)$ wiedergibt. Das ist nach allem, was wir bisher schon gesehen haben, einfach, siehe Abbildung 11.11.

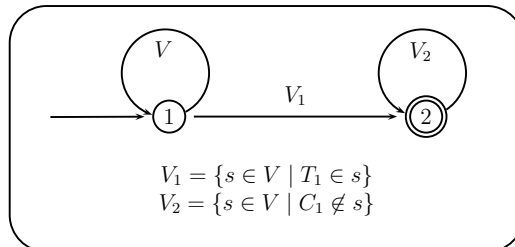


Abbildung 11.11: Büchi-Automat \mathcal{B}_{me} zu $\diamond(T_1 \wedge \square \neg C_1)$

Der Plan ist, zu zeigen, daß keine Berechnungsfolge des Automaten \mathcal{A}_{me} eine omega-Struktur ergibt, die $\diamond(T_1 \wedge \square \neg C_1)$ erfüllt. Das läuft darauf hinaus, daß kein ω -Wort aus V im Durchschnitt von $L^\omega(\mathcal{A}_{me})$ und $L^\omega(\mathcal{B}_{me})$ liegt. In dem vorliegenden Beispiel kann man den Produktautomat für diesen Test benutzen, weil $L^\omega(\mathcal{A}_{me} \times \mathcal{B}_{me}) = L^\omega(\mathcal{A}_{me}) \cap L^\omega(\mathcal{B}_{me})$ gilt. Im allgemeinen muß der in Lemma 11.2 konstruierte Automat für die Konjunktion von LTL-Formeln benutzt werden. Im vorliegenden Fall ist insbesondere (s_1, t_1) ein Endzustand des Produktautomaten, wenn sowohl s_1 als auch t_1 in den Faktorautomaten Endzustände sind. In dem Automaten \mathcal{A}_{me} ist jeder Zustand ein Endzustand. Wir haben das nicht extra durch Doppelkreise angezeigt. Das erklärt den in Abbildung 11.12 gezeigten Produktautomaten.

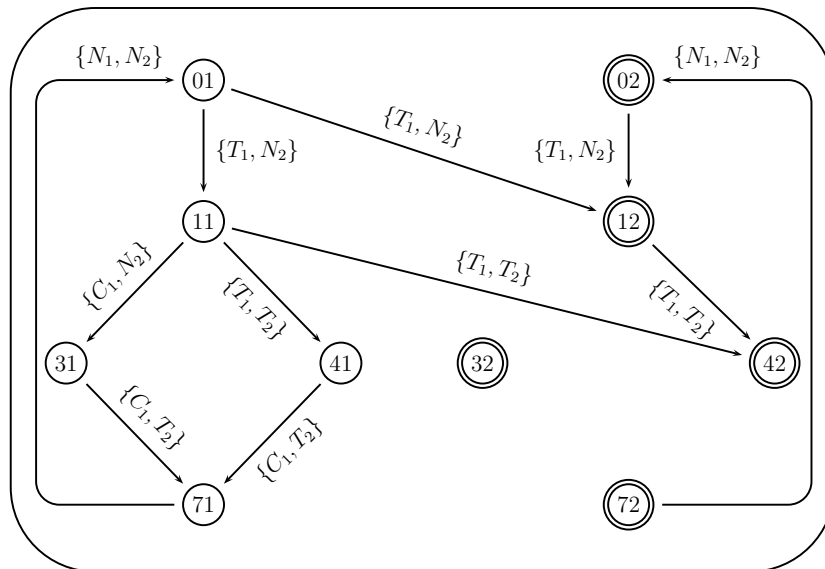


Abbildung 11.12: Produktautomat $\mathcal{A}_{me} \times \mathcal{B}_{me}$

Man sieht leicht, daß der Produktautomat keine Schleife enthält, die durch einen Endzustand geht, d.h. $L^\omega(\mathcal{A}_{me} \times \mathcal{B}_{me}) = \emptyset$

11.3.3 Übungsaufgaben

11.4 Aussagenlogische Modellprüfung (Zusatzstoff)

Dieses Kapitel stellt einen interessanten Zusammenhang her zwischen dem im Unterkapitel 11.3.1 vorgestellten Modellprüfungsproblem und aussagenlogischer Erfüllbarkeit, Definition 2.12. Es bildet die Grundlage für eine Alternative zur Modellprüfungsmethode aus Abschnitt 11.3.2, indem man Programme zur Lösung aussagenlogischer Erfüllbarkeitsprobleme, sogenannte SAT solver, einsetzen kann. Dieser Ansatz ist unter dem Namen *bounded model checking* bekannt geworden und wurde zuerst in der Arbeit [BCCZ99] vorgestellt.

Wir beginnen mit einer simplen notationellen Vorbereitung. Die Aussage, daß ein Büchi-Automat \mathcal{A} eine LTL-Formel A erfüllt, $\mathcal{A} \models A$, bedeutet, daß für jede akzeptierende Berechnungsfolge s von \mathcal{A} die zugeordnete omega-Struktur ξ (dargestellt als ein Wort über einem geeigneten Alphabet) die Formel erfüllt, d.h. daß $\xi \models A$ gilt. Bisher hatten wir nur $\xi \models A$ definiert. Das war auch aus Gründen einer modularen Vorgehensweise nützlich so; man konnte die Theorie der Logik LTL entwickeln ohne sich festzulegen, woher die omega-Strukturen kommen. Beschäftigt man sich allerdings ausschließlich mit Modellprüfungsaufgaben, wie wir das in diesem Kapitel tun, so macht es mehr Sinn direkt zu definieren, wann eine akzeptierende Berechnungsfolge $s = (s_i)_{i \geq 0}$ eine LTL-Formel A erfüllt. Man beachte, daß $\xi(n)$ die Markierung der Kante von s_n nach s_{n+1} ist. Die Definition von $\xi \models A$, Def.8.4, läßt sich jetzt trivialerweise umschreiben zu einer Definition von $s \models A$:

Definition 11.5 (Semantik von LTL für Berechnungsfolgen)

$s \models p$	gdw	$p \in \xi(0)$ (p ein AL Atom)
$s \models op(A, B)$		für aussagenlogische Kombinationen $op(A, B)$ von A und B wie üblich
$s \models \Box A$	gdw	für alle $n \in \mathbb{N}$ gilt $s_n \models A$
$s \models \Diamond A$	gdw	es gibt ein $n \in \mathbb{N}$ mit $s_n \models A$
$s \models A \mathbf{U} B$	gdw	es gibt $n \in \mathbb{N}$ mit $s_n \models B$ und für alle m mit $0 \leq m < n$ gilt $s_m \models A$
$s \models X A$	gdw	$s_1 \models A$

wobei s_n die Restfolge $(s_m)_{m \geq n}$ ist.

Wir werden im weiteren Verlauf dieses Abschnitt die Gültigkeit einer LTL-Formel für eine Berechnungsfolge, d.h. eine Aussage der Form $s \models A$, durch aussagenlogische Formeln ausdrücken. Die erste Schwierigkeit dabei ist die

Unendlichkeit der Berechnungsfolge s . Die Lösung kommt daher, daß wir beweisen können, daß es genügt, endliche, zyklische Berechnungsfolgen zu betrachten.

Definition 11.6 (Zyklische Berechnungsfolgen)

1. Eine endliche Berechnungsfolge s_0, \dots, s_k heißt *i-zyklisch*, falls $0 \leq i < k$ und $s_k = s_i$.
2. Eine endliche Berechnungsfolge s_0, \dots, s_k heißt *zyklisch*, falls sie *i-zyklisch* ist für ein i mit $0 \leq i < k$.
3. Eine *i-zyklische* Berechnungsfolge s_0, \dots, s_k heißt *akzeptierend*, wenn unter den Zuständen s_i, \dots, s_k mindestens ein Endzustand vorkommt.
4. Für eine *i-zyklische* Berechnungsfolge $s = s_0, \dots, s_k$ und eine LTL Formel A schreiben wir $s \models A$ anstelle von $s_0, \dots, s_{i-1}, (s_i, \dots, s_{k-1})^\omega \models A$.

Eine *i-zyklische* Berechnungsfolge s_0, \dots, s_k enthält schon die gesamte Information über die unendliche, schließlich periodische Berechnungsfolge $s_0, \dots, s_{i-1}, (s_i, \dots, s_{k-1})^\omega$. Insbesondere ist $s_0, \dots, s_{i-1}, (s_i, \dots, s_{k-1})^\omega$ akzeptierend, wenn s_0, \dots, s_k akzeptierend ist.

Lemma 11.7

Sei \mathcal{A} ein Büchi-Automat und A eine LTL-Formel.

Falls es eine akzeptierende Berechnungsfolge t gibt mit $t \models A$, dann gibt es auch eine endliche zyklische akzeptierende Berechnungsfolge s_e mit $s_e \models A$.

Beweis Für einfache Formeln, etwa $A \equiv \Box p$ oder $A \equiv \Diamond p$, kann man leicht sehen, daß die Aussage des Lemmas richtig ist. Denn ist t eine akzeptierende Berechnungsfolge, dann muß ein Endzustand s_F unendlich oft vorkommen. Die endliche Teilfolge t_e von t bis zum zweiten Auftreten (also dem ersten wiederholten Auftreten) von s_F erfüllt auch $t_e \models \Box p$, wenn schon $t \models \Box p$ gegolten hat. Gilt $t \models \Diamond p$, dann sucht man das kleinste i mit $t_i \models p$. Liegt t_i vor dem ersten Auftreten von s_F , kann man wie gerade beschrieben verfahren, anderenfalls bricht man die Folge t mit dem ersten Vorkommen von s_F nach t_i ab.

Es ist allerdings schwierig zu sehen, wie man daraus ein allgemeines Verfahren erhalten kann oder ein induktives Argument darauf aufbauen kann. Deswegen benutzen wir den folgenden Trick.

Sei \mathcal{B}_A der nach Satz 11.3 existierende Büchi-Automat, so daß $L^\omega(\mathcal{B}_A) = \{\xi \mid \xi \models A\}$. Sei weiterhin \mathcal{C} der Automat $\mathcal{A} \cap \mathcal{B}_A$ aus Lemma 11.2 mit der Eigenschaft $L^\omega(\mathcal{C}) = L^\omega(\mathcal{A}) \cap L^\omega(\mathcal{B}_A)$. Nach Annahme des Lemmas gilt also

$L^\omega(\mathcal{C}) \neq \emptyset$. Es gibt also eine akzeptierende Berechnungsfolge s' von \mathcal{C} , so daß für die omega-Struktur ξ' , die s' zugeordnet ist, gilt $\xi' \models A$. Nach der Akzeptanzbedingung für Büchi-Automaten gibt es einen Endzustand s_F von \mathcal{C} , der unendlich oft in s' vorkommt. Sei s'_i das erste und s'_k das zweite Vorkommen von s_F in s' und s die Folge $s'_1, \dots, s'_{i-1}(s'_i, \dots, s'_{k-1})^\omega$. Dann liegt auch s in $L^\omega(\mathcal{C})$. Insbesondere gilt für die s zugeordnete omega-Struktur ξ wieder $\xi \models A$.

Jetzt müssen wir etwas genauer in die interne Struktur von \mathcal{C} hineinschauen. Jeder Zustand s_j in der Folge s ist von der Form $s_j = (s_j^1, s_j^2, k_j)$ wobei s_j^1 ein Zustand von \mathcal{A} ist, s_j^2 ein Zustand von \mathcal{B}_A ist und $k_j \in \{1, 2\}$. Nach Konstruktion von \mathcal{C} ist s_F^1 ein Endzustand von \mathcal{A} und damit die Projektion von s auf die erste Koordinate, $s^1 = s_1^1, \dots, s_n^1, \dots$ eine akzeptierende Berechnungsfolge von \mathcal{A} . Die Projektion s^2 von s auf die zweite Koordinate ist ebenfalls eine akzeptierende Berechnungsfolge von \mathcal{B}_A . Als letztes bemerken wir, daß die omega-Struktur die s^1 und s^2 zugeordnet ist mit ξ übereinstimmt. Somit haben wir mit $s_e = s_1^1, \dots, s_k^1$ eine endliche zyklische Berechnungsfolge gefunden, welche die Behauptung des Lemmas erfüllt. ■

Wir können jetzt mit der aussagenlogischen Kodierung beginnen.

Lemma 11.8

Zu einem gegebenen Büchi-Automaten \mathcal{A} und eine LTL-Formel F gibt es für jedes $k \in \mathbb{N}$ eine aussagenlogische Formel M_k , so daß gilt

$$M_k \text{ ist erfüllbar } \text{ gdw } \text{ es gibt eine zyklische Berechnungsfolge } s \text{ der Länge } k \text{ mit } s \models F$$

Falls erforderlich schreiben wir anstelle von M_k genauer $M_k(\mathcal{A}, F)$.

Beweis Wir geben explizit ein Konstruktionsverfahren für M_k an. Diese Konstruktion ist zwar aufwendig, aber leicht nachvollziehbar. Wir werden deswegen keinen zusätzlichen Beweis führen, daß die angeführte Konstruktion tatsächlich leistet, was sie soll. Wir werden zusätzlich die einzelnen Schritte anhand des Automaten \mathcal{A}_{dbp} aus Abb.11.4(a) illustrieren.

Wir nehmen an, daß \mathcal{A} n Zustände hat, auf die wir mit ihrer Nummer verweisen, also Zustand 1, 5 oder allgemein Zustand n . Als erste Vorbereitung brauchen wir eine Binärkodierung der Zustandsnummern. Die dafür nötigen Booleschen Variablen seien c_1, \dots, c_m .

In dem Beispielautomaten \mathcal{A}_{dbp} gibt es nur zwei Zustände, also reicht eine Boolesche Variable c aus. $I(c) = \mathbf{0}$ bezeichnet den Anfangszustand $I(c) = \mathbf{1}$ den eindeutigen Endzustand.

Außerdem kommen in der Aufgabenstellungen die aussagenlogischen Variablen aus der Formel A vor. Diese seien $\Sigma = \{p_1, \dots, p_r\}$.

In unserem Beispiel sind p und q die einzigen aussagenlogischen Variablen.

Wir werden später noch einige weitere Hilfsvariablen einführen, über die wir aber reden, wenn es soweit ist.

Da die Formel M_k über Folgen von Zuständen der Länge k reden soll, kommen alle Variablen in k Kopien vor, also c_j^i mit $1 \leq i \leq k$ und $1 \leq j \leq m$ und p_j^i mit $1 \leq i < k$ und $1 \leq j \leq r$.

In unserem Beispiel wollen wir mit $k = 3$ rechnen, also gibt es die aussagenlogischen Variablen c^1, c^2, c^3, p^1, p^2 und q^1, q^2 .

Die Formel M_k wird in mehreren Teilen konstruiert

$$M_k \equiv \text{Init} \wedge \text{Trans} \wedge \bigvee_{1 \leq i < k} L_i$$

Ist I eine Interpretation der aussagenlogischen Variablen, dann fassen wir $I(c_1^i), \dots, I(c_m^i)$ für jedes $1 \leq i \leq k$ als Binärkodierung der Zahl n_i auf. So erhalten wir eine Folge von Zuständen $\pi = n_1, \dots, n_k$. Ebenso fassen wir $I(p_1^i), \dots, I(p_r^i)$ als Kodierung einer Teilmenge b_i der Variablen p_1, \dots, p_r auf. Die b_i sind also Buchstaben aus dem Vokabular V der Kantenmarkierungen des Automaten \mathcal{A} . Insgesamt erhalten wir eine Wort b_1, \dots, b_{k-1} der Länge $k - 1$. Die Formeln Init und Trans werden dafür sorgen, daß n_1, \dots, n_k eine Berechnungsfolge von \mathcal{A} ist mit der Folge b_1, \dots, b_{k-1} von Kantenmarkierungen.

Betrachten wir die Interpretation I_1, I_2 in unserem Beispiel

	c^1	c^2	c^3	p^1	p^2	q^1	q^2
I_1	0	1	1	1	1	1	0
I_2	0	1	1	1	0	0	1

I_1 und I_2 liefern die Zustandsfolge $\pi = 0, 1, 1$. I_1 liefert dazu die Buchstabenfolge $w_1 = \{p, q\}, \{p\}$ und I_2 die Folge $w_2 = \{p\}, \{q\}$. Man sieht, daß w_1 eine korrekte Kantenmarkierung für die Folge π ist, w_2 aber nicht.

Ist $d = d_1, \dots, d_m$ die Binärkodenummer eines Zustands von \mathcal{A} dann bezeichnen wir mit $S_d^i = S_{d_1, \dots, d_m}^i$ die aussagenlogische Formel $\bigwedge_{d_j=1} c_j^i \wedge \bigwedge_{d_j=0} \neg c_j^i$. Ist b ein Buchstabe des Kantenalphabets, also $b \subseteq \Sigma$, dann steht B_b^i für die Formel $\bigwedge_{p_j \in b} p_j^i \wedge \bigwedge_{p_j \notin b} \neg p_j^i$.

Ist $d_0 = d_1, \dots, d_m$ die Binärkodenummer des Anfangszustands von \mathcal{A} dann ist

$$\text{Init} \equiv S_{d_0}^1$$

Seien $(d_a^1, d_e^1, b^1), \dots, (d_a^K, d_e^K, b^K)$ alle Kanten des Automaten \mathcal{A} , aufgefasst als Übergang von dem Zustand mit der Binärkodierung d_a^j in den Zustand mit der Binärkodierung d_e^j und der Kantenmarkierung b^j für $1 \leq j \leq K$.

$$Trans \equiv \bigwedge_{1 \leq i < k \leq j \leq K} \bigvee (S_{d_a^i}^i \wedge S_{d_e^j}^{i+1} \wedge B_{b^j}^i)$$

Für unser Beispiel ergeben diese Definitionen $Init = \neg c^1$. Die Menge aller Kanten ist

$$\begin{aligned} & (0, 0, \{\}), (0, 0, \{p\}), (0, 0, \{q\}), (0, 0, \{p, q\}) \\ & (0, 1, \{p\}), (0, 1, \{p, q\}) \\ & (1, 1, \{p\}), (1, 1, \{p, q\}) \end{aligned}$$

$$\begin{aligned} Trans = & \bigwedge \\ & (\neg c^1 \wedge \neg c^2) \vee (\neg c^1 \wedge c^2 \wedge p^1) \vee (c^1 \wedge c^2 \wedge p^1) \\ & (\neg c^2 \wedge \neg c^3) \vee (\neg c^2 \wedge c^3 \wedge p^2) \vee (c^2 \wedge c^3 \wedge p^2) \end{aligned}$$

Dabei haben wir schon einige Vereinfachungen vorgenommen und z.B. $(\neg c^1 \wedge \neg c^2 \wedge p^1 \wedge q^1) \vee (\neg c^1 \wedge \neg c^2 \wedge \neg p^1 \wedge q^1) \vee (\neg c^1 \wedge \neg c^2 \wedge p^1 \wedge \neg q^1) \vee (\neg c^1 \wedge \neg c^2 \wedge \neg p^1 \wedge \neg q^1)$ äquivalent ersetzt durch $(\neg c^1 \wedge \neg c^2)$. Als weitere Vereinfachung könnte man $(c^1 \wedge c^2 \wedge p^1)$ weglassen, weil das im Widerspruch zu $Init$ steht.

Die Formeln L_i werden dafür sorgen, daß die aus einer erfüllenden Interpretation I extrahierte endliche Berechnungsfolge π i -zyklisch und akzeptierend ist und, was am wichtigsten ist, die Formel F erfüllt:

$$L_i = Z_i \wedge Akz_i \wedge erfF_i$$

Die ersten beiden Bestandteile sind einfach

$$\begin{aligned} Z_i & \equiv \bigwedge_{1 \leq j \leq m} c_j^k \leftrightarrow \bigwedge_{1 \leq j \leq m} c_j^i \\ Akz_i & \equiv Fin_i \vee Fin_{i+1} \dots \vee Fin_{k-1} \end{aligned}$$

wobei $F_i \equiv S_{d_1^f}^i \vee \dots \vee S_{d_R^f}^i$ wenn d_1^f, \dots, d_R^f die Binärkodes aller Finalzustände von \mathcal{A} sind.

Als letztes bleibt $erfF_i$ zu erklären. Die Erfüllbarkeit dieser Formel soll sicherstellen, daß die durch die erfüllende Belegung I bestimmte i -zyklische Berechnungsfolge π die Formel F erfüllt. An dieser Stelle brauchen wir neue aussagenlogische Variable. Wir sind zwar nur an der Gültigkeit von F an der Position 1 von π interessiert, es wird sich aber durch die Definition der temporalen Operatoren nicht vermeiden lassen, auch die Gültigkeit an den anderen Positionen $1 \leq j \leq k$ zu betrachten. Außerdem hängt die Gültigkeit auch von dem zyklischen *Rücksprungspunkt* i ab. Also brauchen wir für jede

Teilformel C , für jedes i , $1 \leq i < k$ und jedes j , $1 \leq j \leq k$ eine neue aussagenlogische Variable, die wir mit $[C]_i^j$ bezeichnen. Die definierenden Formeln lassen sich dann wie in Abbildung 11.13 aufschreiben. Die Zeilen für $\Box C$ und $\Diamond C$ sind eigentlich überflüssig. Es hilft aber beim Verstehen, wenn man zuerst die Definition von $\Box C$ und $\Diamond C$ gesehen hat, bevor die komplizierteren Operatoren $C_1 \mathbf{U} C_2$ und $C_1 \mathbf{V} C_2$ drankommen. Es gibt keine allgemeine Regel für die Negation. Es wird vorausgesetzt, daß die Formel F in Negationsnormalform vorliegt und daß die in der Definition von $[C_1 \mathbf{V} C_2]_i^j$ auf der rechten Seite auftretende Formel $\neg C_2$ in Negationsnormalform transformiert wird. Zu den aus der Abbildung 11.13 sich ergebenden aussagenlogischen Äquivalenzformeln kommt schließlich noch hinzu

$$erfF_i \leftrightarrow [F]_i^1$$

$$\begin{array}{ll}
[C]_i^j & \leftrightarrow p_l^j \quad \text{falls } C = p_l \in \Sigma \\
[C]_i^j & \leftrightarrow \neg p_l^j \quad \text{falls } C = \neg p_l \text{ mit } p_l \in \Sigma \\
[C_1 \wedge C_2]_i^j & \leftrightarrow [C_1]_i^j \wedge [C_2]_i^j \\
[C_1 \vee C_2]_i^j & \leftrightarrow [C_1]_i^j \vee [C_2]_i^j \\
[\Box C]_i^j & \leftrightarrow \bigwedge_{j \leq l < k} [C]_i^l & \text{falls } j \leq i \\
[\Box C]_i^j & \leftrightarrow \bigwedge_{j \leq l < k} [C]_i^l \wedge \bigwedge_{i \leq l < j} [C]_i^l & \text{falls } i < j \\
[\Diamond C]_i^j & \leftrightarrow \bigvee_{j \leq l < k} [C]_i^l & \text{falls } j \leq i \\
[\Diamond C]_i^j & \leftrightarrow \bigvee_{j \leq l < k} [C]_i^l \vee \bigvee_{i \leq l < j} [C]_i^l & \text{falls } i < j \\
[C_1 \mathbf{U} C_2]_i^j & \leftrightarrow \bigvee_{j \leq l < k} ([C_2]_i^l \wedge \bigwedge_{j \leq n < l} [C_1]_i^n) & \text{falls } j \leq i \\
[C_1 \mathbf{U} C_2]_i^j & \leftrightarrow \bigvee_{j \leq l < k} ([C_2]_i^l \wedge \bigwedge_{j \leq n < l} [C_1]_i^n) \vee \\
& \quad \bigvee_{i \leq l < j} ([C_2]_i^l \wedge \bigwedge_{j \leq n < k} [C_1]_i^n \wedge \bigwedge_{i \leq n < l} [C_1]_i^n) & \text{falls } i < j \\
[C_1 \mathbf{V} C_2]_i^j & \leftrightarrow [C_2]_i^j \wedge \bigwedge_{i \leq l < k} ([\neg C_2]_i^l \rightarrow \bigvee_{i \leq n < l} [C_1]_i^n) & \text{falls } j \leq i \\
[C_1 \mathbf{V} C_2]_i^j & \leftrightarrow [C_2]_i^j \wedge \bigwedge_{j \leq l < k} ([\neg C_2]_i^l \rightarrow \bigvee_{j \leq n < l} [C_1]_i^n) \wedge \\
& \quad \bigwedge_{i \leq l < j} ([\neg C_2]_i^l \rightarrow \bigvee_{i \leq n < l} [C_1]_i^n \vee \bigvee_{j \leq n < k} [C_1]_i^n) & \text{falls } i < j \\
[X C]_i^j & \leftrightarrow [C]_i^{j+1} & \text{falls } j < (k-1) \\
[X C]_i^{k-1} & \leftrightarrow [C]_i^i
\end{array}$$

Abbildung 11.13: Zyklische Semantik für LTL Formeln

Wir wollen jetzt L_1 und L_2 für unser Beispiel vorrechnen. Wir beginnen mit $Z_1 \equiv c^1 \leftrightarrow c^3$ und $Z_2 \equiv c^2 \leftrightarrow c^3$. Da der einzige Finalzustand von \mathcal{A}_{dbp} durch c gegeben ist erhalten wir $Akz_1 \equiv c^1 \vee c^2 \vee c^3$ und $Akz_2 \equiv c^2 \vee c^3$. Betrachten

wir für unser Beispiel die Formel $F \equiv \diamond \Box p$.

$$\begin{aligned}
[F]_1^1 &\equiv [\Box p]_1^1 \vee [\Box p]_1^2 \\
[\Box p]_1^1 &\equiv [p]_1^1 \wedge [p]_1^2 \\
&\equiv p^1 \wedge p^2 \\
[\Box p]_1^2 &\equiv [p]_1^2 \wedge [p]_1^1 \\
&\equiv p^2 \wedge p^1
\end{aligned}$$

Insgesamt also $[F]_1^1 \leftrightarrow p^1 \wedge p^2$

$$\begin{aligned}
[F]_2^1 &\equiv [\Box p]_2^1 \vee [\Box p]_2^2 \\
[\Box p]_2^1 &\equiv [p]_2^1 \wedge [p]_2^2 \\
&\equiv p^1 \wedge p^2 \\
[\Box p]_2^2 &\equiv [p]_2^2 \\
&\equiv p^2
\end{aligned}$$

Insgesamt also $[F]_2^1 \leftrightarrow p^2$.

Wir können jetzt die Berechnung der L_i für das Beispiel hinschreiben.

$$\begin{aligned}
L_1 &\leftrightarrow (c^1 \leftrightarrow c^3) \wedge (c^1 \vee c^2 \vee c^3) \wedge p^1 \wedge p^2 \\
&\leftrightarrow (c^1 \leftrightarrow c^3) \wedge (c^2 \vee c^3) \wedge p^1 \wedge p^2 \\
L_2 &\leftrightarrow (c^2 \leftrightarrow c^3) \wedge (c^2 \vee c^3) \wedge p^2 \\
&\leftrightarrow c^2 \wedge c^3 \wedge p^2
\end{aligned}$$

Das Endergebnis der gesamten Konstruktion M_3 ist in Abbildung 11.14 zu sehen. Diese Formel läßt sich weiter vereinfachen, wenn man ausnutzt, daß $Init = \neg c^1$ gelten muß. Durch diese Vereinfachungen ergibt sich $c^2 \rightarrow p^1$, was dann zu weiteren Vereinfachungen benutzt wird. Schließlich erhält man $M_3 \equiv \neg c^1 \wedge c^2 \wedge c^3 \wedge p^1 \wedge p^2$. Offensichtlich ist diese Formel erfüllbar. Was nach ihrer Konstruktion besagt, daß es eine Berechnungsfolge in \mathcal{A}_{dbp} gibt, für die $\diamond \Box p$ wahr ist. Wir wissen sogar, daß für jede Berechnungsfolge von \mathcal{A}_{dbp} diese Formel gilt.

Satz 11.9

Es gibt eine Berechnungsfolge eines Büchi-Automaten \mathcal{A} , welche die LTL-Formel F erfüllt genau dann, wenn es ein k gibt, so daß $M_k(\mathcal{A}, F)$ aussagenlogisch erfüllbar ist.

$$\begin{aligned}
& \neg c^1 \wedge \\
& (\neg c^1 \wedge \neg c^2) \vee (\neg c^1 \wedge c^2 \wedge p^1) \vee (c^1 \wedge c^2 \wedge p^1) \\
& \wedge \\
& (\neg c^2 \wedge \neg c^3) \vee (\neg c^2 \wedge c^3 \wedge p^2) \vee (c^2 \wedge c^3 \wedge p^2) \\
& \wedge (\\
& ((c^1 \leftrightarrow c^3) \wedge (c^2 \vee c^3) \wedge p^1 \wedge p^2) \\
& \vee \\
& (c^2 \wedge c^3 \wedge p^2) \\
&)
\end{aligned}$$

Abbildung 11.14: M_3 für den Beispielautomaten \mathcal{A}_{dbp} und $F \equiv \diamond \square p$

Beweis Folgt aus den Lemmata 11.8 und 11.7. ■

Das Verfahren der limitierten Modellprüfung (das ist unsere Übersetzung von *bounded model checking*) für einen Büchi Automaten \mathcal{A} und eine LTL-Formel F funktioniert nun so, daß für ein zunächst kleines k die aussagenlogische Erfüllbarkeit der Menge $M_k(\mathcal{A}, F)$ analysiert wird. Dazu gibt es leistungsfähige Implementierungen von Entscheidungsverfahren, sogenannte *SAT solver*. Wird eine Lösung gefunden, so haben wir das Ausgangsproblem gelöst. Wird dagegen festgestellt, daß $M_k(\mathcal{A}, F)$ unerfüllbar ist, kann man weiterfahren mit der Analyse der Erfüllbarkeit von $M_{k+1}(\mathcal{A}, F)$. In einer Situation, in der es keine Berechnungsfolge von \mathcal{A} gibt, die F erfüllt, terminiert dieses Verfahren zunächst nicht. Man kann aber leicht sehen, daß aus dem Beweis von Lemma 11.7 eine obere Schranke für k abgeleitet werden kann. Die kleinste obere Schranke für die Lemma 11.7 gilt, nennt man die Vollständigkeitsschranke (*completeness threshold*) des Problems. Ist für ein k das gleich oder grösser als die Vollständigkeitsschranke ist $M_k(\mathcal{A}, F)$ unerfüllbar, so wissen wir, daß es keine akzeptierte Berechnungsfolge von \mathcal{A} gibt, die F erfüllt.

Das limitierte Modellprüfungsverfahren reduziert nicht die Komplexität des zu lösenden Problems, es beruht weiterhin auf einem NP-vollständigen Verfahren. Experimente haben jedoch gezeigt, daß die Stärken des automaten-theoretischen Zugangs orthogonal liegen zu den Stärken des limitierten Modellprüfungsverfahrens, [BCC⁺03, CFF⁺01].

11.4.1 Übungsaufgaben

Übungsaufgabe 11.4.1

Berechnen Sie die Formeln L_1 , L_2 und M_3 für das in der Konstruktion zu Lemma 11.8 benutzte Beispiel für die Formel $F \equiv \Box \Diamond \neg p$.

11.5 Modellprüfung mit SPIN

Diese Abschnitt ist noch in Bearbeitung.

Das zentrale Konzept in den beiden vorangehenden Kapiteln 10 und 11 ist das Konzept eines Büchi-Automaten, siehe Definition 10.19. Für Beweise von allgemeinen Aussagen über Büchi-Automaten, z.B. Lemma 11.2 oder Satz 11.3, wurde jeweils auf diese Definition bezug genommen und dazu gibt es auch keine Alternative. Dagegen wurden Beispiele für konkrete Automaten bisher ausschließlich in graphischer Form gegeben. Das ist natürlich nur für kleine Automaten möglich, für größere, wie sie in realistischen Anwendungen auftreten, ist das nicht praktikabel.

```
1 /* Peterson 's solution to mutual exclusion - 1981 */
2 bool turn, flag[2];
3 byte ncrit;
4 active [2] proctype user()
5 {
6     assert(_pid == 0 || _pid == 1);
7     again: flag[_pid] = 1;
8     turn = _pid;
9     (flag[1 - _pid] == 0 || turn == 1 - _pid);
10
11     ncrit++;
12     assert(ncrit == 1);    /* critical section */
13     ncrit--;
14
15     flag[_pid] = 0;
16     goto again
17 }
```

Abbildung 11.15: *Mutual exclusion* für zwei Prozesse

Für dieses Problem gibt es eine Vielzahl von Lösungen. Wir wollen in diesem Abschnitt die Lösung, die G. Holtzmann für das SPIN System, [Hol04], benutzt vorstellen. Der Benutzer von SPIN muss die Automatenmodelle, die mit dem System analysiert werden sollen, in irgendeiner Form eingeben. Dazu dient die *Promela* Sprache (**P**rocess **M**eta **L**anguage). In Abbildung 11.5 ist ein einfaches Beispiel einer Prozeßbeschreibung in Promela zu sehen. Dieses Beispiel ist in den Testbeispielen enthalten, die mit dem SPIN System ausgeliefert werden. Es behandelt eine Lösung des konkurrierenden Zugriffs

auf eine kritische Ressource in dem einfachen Fall, daß es nur zwei Prozesse gibt.

Wir beginnen mit einer Erklärung von Promela, die es zumindest erlaubt den Code in Abb. 11.5 zu verstehen. Als erstes ist zu bemerken, daß Promela keine Programmiersprache ist, sondern eine Beschreibungssprache für Systeme mit nebenläufiger Systeme (engl. concurrent systems). Promela Programme beschreiben Zustandsübergangssysteme durch die Definition von Prozesstypen, *process types*, angezeigt durch das Schlüsselwort `proctype`. In dem vorliegenden Codestück wird der Prozesstyp `users()` in den Zeilen 4 - 17 deklariert. Mit der Deklaration eines Typs ist noch keine Aktion verbunden. Das geschieht erst durch die Ausführung des `run` Befehls. Damit überhaupt Aktionen ins Rollen kommen gibt es einen Prozess `init`, der bei der Initialisierung gestartet wird. In der Deklaration von `init` könnten dann z.B., `run users();` `run users();` stehen, womit zwei Prozesse vom Prozesstyp `users()` erzeugt würden. In dem vorliegenden Codebeispiel kommt keine `init` Deklaration vor. Stattdessen wird hier von einer Abkürzung Gebrauch gemacht, indem gleich bei der Prozedeklaration in Zeile 4 durch den Präfix `active [2]` zwei Prozesse des deklarierten Typs erzeugt werden. Die Prozesse werden in der Reihenfolge ihrer Aktivierung durchnumeriert. Auf die Nummer eines Prozesses, seine *process id*, kann mit der lokalen Variablen `_pid` zugegriffen werden. In dem betrachteten Code kommt das häufig vor, genauer in den Zeilen 6,7,8,9 und 15.

Neben der Prozesstypdeklaration von `users()` enthält der Code noch in den Zeilen 2 und 3 die Deklaration der globalen Variablen `turn`, `flag[2]`, `ncrit`. Dabei ist `flag` ein Feld (*array*) der Länge 2. Die Indizierung beginnt mit 0. Globale Variable können von jedem Prozess gelesen und geschrieben werden. Die Anweisungen im Rumpf der Prozesstypdeklaration von `users()` sind für einen Leser mit etwas Programmiererfahrung leicht zu verstehen, bis auf die Zeilen 6,9 und 12. Die `assert` Anweisungen in den Zeilen 6 und 9 werten den in der Klammer angegebenen Booleschen Ausdruck aus. Falls er wahr ist, geht die Ausführung mit der nächsten Zeile weiter, anderenfalls wird ein Fehler berichtet und die Simulation oder Verifikation wird abgebrochen. Anstelle von Booleschen Ausdrücken können auch ganzzahlige Ausdrücke verwendet werden, die als wahr interpretiert werden wenn ihr Wert > 0 ist. Anders verhält es sich mit dem Kommando in Zeile 9. Eigentlich steht in dieser Zeile ein Boolescher Ausdruck. Es gehört zu den Eigenarten von Promela, daß jeder Ausdruck auch anstelle einer Anweisung stehen kann. Ergibt die Auswertung den Wahrheitswert *wahr*, dann geht die Ausführung mit der nächsten Anweisung weiter. Anderenfalls wird die Ausführung des Prozesses *blockiert*. Der Prozess bleibt blockiert, bis durch die Aktionen anderer Prozesse der Boo-

lesche Ausdruck wahr wird. Die Einteilung in ausführbare (engl. executable oder enabled) und blockierte (engl. blocked) Befehle ist eines der wichtigsten Mittel, mit denen man in Promela asynchrone Kommunikation beschreiben kann (Ein zweites Mittel sind Kanäle (engl. channels), die in unserem Beispiel nicht vorkommen.)

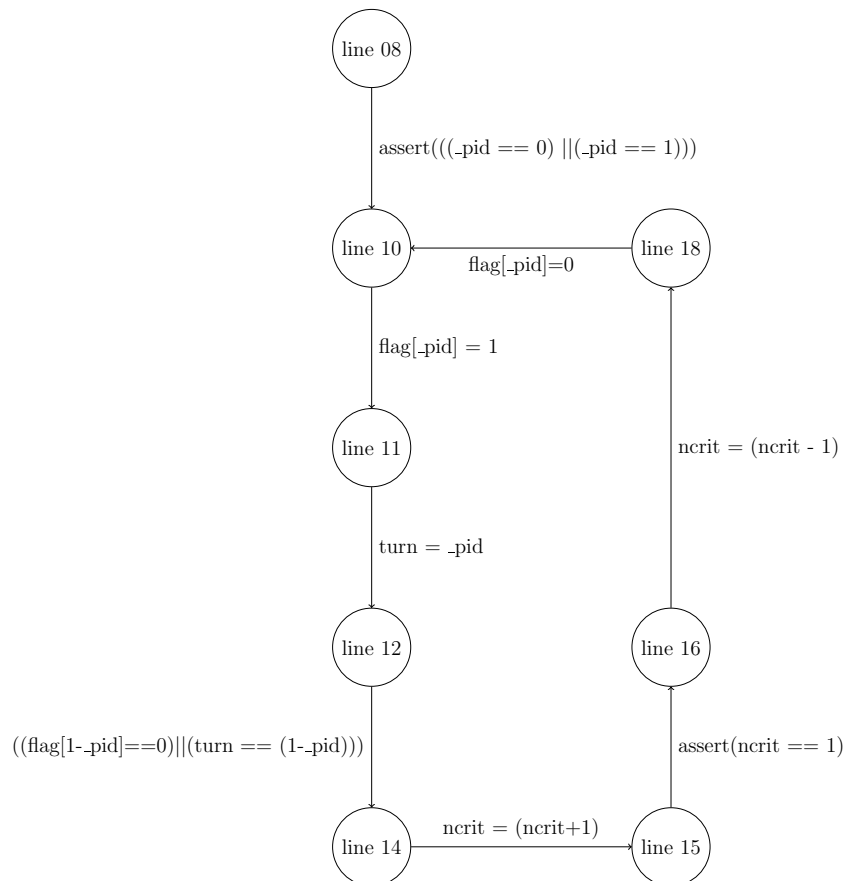


Abbildung 11.16: Generischer Automat zum *user* Prozess

SPIN bietet die Möglichkeit eine Prozesstypdeklaration zu visualisieren durch eine Art Automatenmodell. Die Visualisierung des `users()` Typs ist in Abb. 11.16 zu sehen. Man sieht, daß die Zustände durch die Nummern von Programmzeilen beschrieben werden. An den ausgehenden Kanten stehen die Anweisungen in der entsprechenden Zeile, die dann zum nächsten Zustand, d.h. zur dann erreichten Zeile führen. Diese Visualisierung vermittelt natürlich nur ein partielles Bild des Systems. Zum einen werden, der Übersichtlichkeit halber, die Werte der globalen Variablen nicht gezeigt, zum anderen fehlt

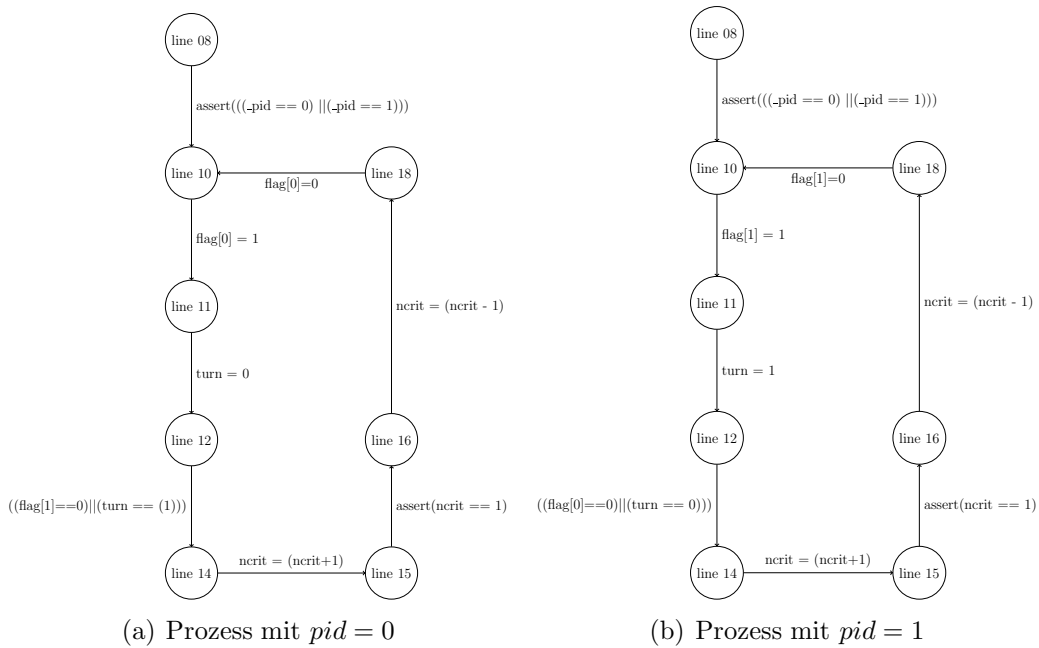


Abbildung 11.17: Instanzen des generischen Automaten aus Abb. 11.16

die Darstellung des Zusammenwirkens der beiden Prozesse. In Abb. 11.17 sind, was man in SPIN nicht sieht, die Automaten für die beiden Prozessinstanzen vom Typ `users()` dargestellt, einmal für $_pid = 0$ und einmal für $_pid = 1$. Das Zusammenwirken der beiden Prozesse wird durch einen Automaten dargestellt, der ohne Berücksichtigung der globalen Variablen `ncrit` schon 512 Zustände umfassen würde. In Abb. 11.18 ist eine Approximation dieses Automaten zu sehen. Dabei wurde der durch *Zeile 08* gegebene Zustand weggelassen. Ein Zustand des approximativen Produktautomaten wird, untereinander geschrieben, gegeben durch die Programmzeilennummer von Prozess 0, Programmzeilennummer von Prozess 1 und den Wert der globalen Variablen `turn`. Man überlegt sich leicht, daß die Werte der beiden anderen globalen Variablen sich eindeutig aus den beiden Programmzeilen erschließen lassen. Für die Variable `turn` ist das nicht der Fall. Hier kommt z.B. sowohl $(10, 11, 0)$ als auch $(10, 11, 1)$ vor. Um die Abbildung 11.18 nicht durch zu viele Kanten unübersichtlich zu machen, wurden Kanten die vom unteren Ende des Graphen zurück zum oberen Ende führen durch korrespondierende Farbgebung kenntlich gemacht und die verbindende Kante unterdrückt. Es sei noch bemerkt, daß das SPIN System den Gesamtautomaten zu einer Promela Beschreibung nicht vorab berechnet. Er wird *on the fly* berechnet: immer wenn der Suchalgorithmus ausgehend von einem bestimmten Systemzustand die nächsten erreichbaren Zustände braucht, werden diese aus dem

Promela Code ermittelt.

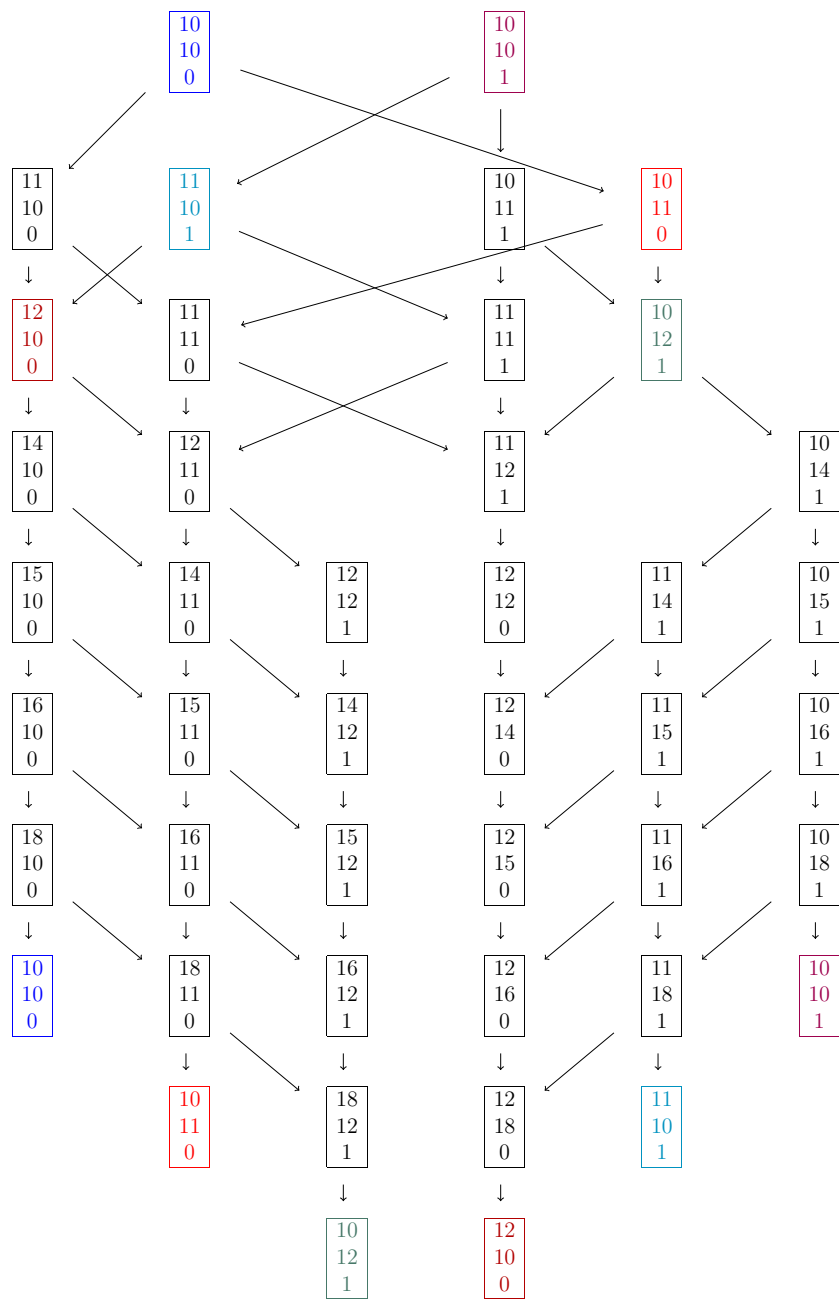


Abbildung 11.18: Ausschnitt des vollständigen Automaten zu Abb. 11.16

Kapitel 12

Lösungen

Gibt es selektiv in den Musterlösungen zu den wöchentlichen Übungsaufgaben auf der Webseite der Vorlesung <http://i12www.ira.uka.de/~mulbrich/teaching/formsys07/>.

Literaturverzeichnis

- [AvH04] Grigoris Antoniou and Frank van Harmelen. *A Semantic Web Primer*. Cooperative Information Systems. MIT Press, April 2004.
- [B98] Egon Börger. *Berechenbarkeit, Komplexität, Logik*. Vieweg Verlagsgesellschaft, 1998.
- [Bar84] H. Barendregt. *The Lambda Calculus*, volume 103 of *Studies in Logic*. North-Holland, Amsterdam, 1984.
- [BB92] M. Buro and Hans Kleine Büning. Report on a SAT competition. Technical Report 110, Universität Paderborn, nov 1992.
- [BCC⁺03] A. Biere, A. Cimatti, E. Clarke, O. Strichman, and Y. Zhu. Bounded model checking. *Advances in Computers*, 58, 2003.
- [BCCZ99] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without bdds. In *Proc. of the Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS99)*, LNCS. Springer-Verlag, 1999.
- [BJL88] Blair, Robert C. Jeroslow, and Lowe. Some results and experiments in programming techniques for propositional logic. *Computers and Operations Research*, 13(13):633–645, 1988.
- [BL92] Matthias Baaz and Alexander Leitsch. Complexity of resolution proofs and function introduction. *Annals of Pure and Applied Logic*, 57:181–215, 1992.
- [Boo87] R. Book. Thue systems as rewriting systems. *J. of Symbolic Computation*, 1 & 2:39–68, 1987.

- [Bry86] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, August 1986.
- [Buc85] Bruno Buchberger. Gröbner bases: An algorithmic method in polynomial ideal theory. In N. K. Bose, editor, *Recent Trends in Multidimensional Systems Theory*. Reidel Publ.Co., 1985.
- [Car58] Lewis Carrol. *Symbolic Logic*. Dover Publications NY, 1958.
- [CFF⁺01] F. Copt, L. Fix, R. Fraer, E. Giunchiglia, G. Kamhi, A. Tacchella, and M.Y. Vardi. Benefits of bounded model checking at an industrial setting. In *Proc. 12th Intl. Conference on Computer Aided Verification (CAV01)*, LNCS, pages 436–453. Springer, 2001.
- [Cha03] Patrice Chalin. Improving JML: For a safer and more effective language. In Keijiro Araki, Stefania Gnesi, and Dino Mandrioli, editors, *Proc. Formal Methods Europe, Pisa, Italy*, volume 2805 of *LNCS*, pages 440–461. Springer-Verlag, 2003.
- [Chu56] Alonzo Church. *Introduction to Mathematical Logic*, volume 1. Princeton University Press, 1956.
- [CL73] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, London, 1973.
- [CT51] L.H. Chin and A. Tarski. Distributive and modular laws in the arithmetic of relation algebras. *University of California Publications in Mathematics, New Series*, 1:341–384, 1951.
- [dB80] J. de Bakker. *Mathematical Theory of Program Correctness*. International Series in Computer Science. Prentice Hall, 1980.
- [DGHP99] Marcello D’Agostino, Dov .M. Gabbay, Reiner Hähnle, and Joachim Posegga, editors. *Handbook of Tableau Methods*. Kluwer Academic Publishers, 1999.
- [DJK04] Wolfgang Dostal, Mario Jeckle, and Werner Kriechbaum. Semantik und web services: Vokabulare und ontologien. *Java Spektrum*, 3:51 – 54, 2004.
- [dM64] A. de Morgan. On the syllogism, no.iv, and on the logic of relations. *Transactions of the Cambridge Philosophical Society*, 10:331–358, 1864.

- [Ede92] Elmar Eder. *Relative Complexities of First-Order Calculi*. Artificial Intelligence. Vieweg Verlag, 1992.
- [EFT92] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Einführung in die Mathematische Logik. 3., vollständig überarbeitete und erweiterte Auflage*. BI Wissenschaftsverlag, 1992.
- [EMC⁺99] Hartmut Ehrig, Bernd Mahr, Felix Cornelis, Martin Große-Rhode, and Philip Zeitz. *Mathematisch-strukturelle Grundlagen der Informatik*. Springer -Lehrbuch. Springer, 1999.
- [Fel78] Walter Felscher. *Naive Mengen und abstrakte Zahlen I*. B.I.Wissenschaftsverlag, 1978.
- [Fit90] Melvin C. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 1990.
- [FM99] Melvin Fitting and Richard L. Mendelsohn. *First-Order Modal Logic*, volume 277 of *Synthese Library*. Kluwer Academic Publishers, 1999.
- [GA02] Steven Givant and Hajnal Andréka. Groups and algebras of binary relations. *The Bulletin of Symbolic Logic*, 8(1):38–64, 2002.
- [Gal86] Jean H. Gallier. *Logic for Computer Science: Foundations of Automated Theorem Proving*. Harper and Row, New York, 1986.
- [Gol82] Robert Goldblatt. *Axiomatising the logic of computer programming*, volume 130 of *LNCS*. Springer-Verlag, 1982.
- [Gor88] M.J. Gordon. HOL: A proof generating system for higher-order logic. In Graham Birtwistle and P.A. Subrahmanyam, editors, *VLSI Specification and Synthesis*, pages 73–128. Kluwer Academic Publishers, 1988.
- [HK89] D. Hofbauer and R. D. Kutsche. *Grundlagen des maschinellen Beweisens*. Vieweg, 1989.
- [HKRS08] Pascal Hitzler, Markus Krtzsch, Sebastian Rudolph, and York Sure. *Semantic Web - Grundlagen*. eXamen.press. Springer-Verlag Berlin Heidelberg, 2008.
- [HMT71] L. Henkin, T. Monk, and A. Tarski. *Cylindrical Algebras, Part I*. North Holland Publ. Co, 1971.

- [Hol04] G.J. Holzmann. *The Spin Model Checker, Primer and Reference Manual*. Addison-Wesley, Reading, Massachusetts, 2004.
- [Hoo88] John N. Hooker. A quantitative approach to logical inference. *Decision Support Systems*, 4:45–69, 1988.
- [HR00] Michael R. Huth and Mark D. Ryan. *Logic in Computer Science. Modelling and reasoning about systems*. Cambridge University Press, 2000.
- [HS94] Reiner Hähnle and Peter H. Schmitt. The liberalized δ -rule in free variable semantic tableaux. *Journal of Automated Reasoning*, 13:211–221, 1994.
- [HU79] J. E. Hopcroft and J. D. Ullmann. *Introduction to Automata Theory*. Addison-Wesley, 1979.
- [Hun33a] E. Huntington. Boolean algebra. a correction. *Transactions of the AMS*, 35:557–558, 1933.
- [Hun33b] E. Huntington. New sets of independent postulates for algebra of logic. *Transactions of the AMS*, 35:274–304, 1933.
- [HW90] B. Heinemann and K. Weihrauch. *Logik für Informatiker*. B. G. Teubner Verlag, 1990.
- [JS74] N.D. Jones and A.L. Selman. Turing machines and the spectra of first-order formulas with equality. *J. Symbolic Logic*, 39:139–150, 1974.
- [Kfo88] Robert A. Molland Michael A. Arbib and A. J. Kfoury. *An Introduction to Formal Language Theory*. Texts and Monographs in Computer Science. Springer-Verlag, 1988.
- [LBR03] Gary T. Leavens, Albert L. Baker, and Clyde Ruby. Preliminary design of JML: A behavioral interface specification language for Java. Technical Report 98-06y, Iowa State University, Department of Computer Science, 2003. Revised June 2004.
- [Lov78] Donald W. Loveland. *Automated Theorem Proving. A Logical Basis*, volume 6 of *Fundamental Studies in Computer Science*. North-Holland, 1978.
- [McC96] William McCune. Robbins algebras are boolean. *Association for Automated Reasoning Newsletter*, 35:1–3, 1996.

- [McM03] K.L. McMillan. Interpolation and sat-based model checking. In F. Somenzi and W. Hunt, editors, *Computer Aided Verification (CAV03)*, 2003.
- [Men87] E. Mendelson. *Introduction to Mathematical Logic*. Wardsworth, Pacific Grove, Calif., 1987.
- [Min90] G. Mints. Gentzen-type systems and resolution rules, part 1: Propositional logic. In *Proc. COLOG-88, Tallin*, volume 417 of *Lecture Notes in Computer Science*, pages 198–231. Springer, 1990.
- [Mon76] J. Donald Monk. *Mathematical Logic*. Springer-Verlag, 1976.
- [OMG97a] OMG. Object constraint language specification, version 1.1. OMG Document ad970808, September 1997.
- [OMG97b] OMG. UML semantics, version 1.1. OMG Document ad970804, September 1997.
- [Pop94] Sally Popkorn. *First Steps in Modal Logic*. Cambridge University Press, 1994.
- [Ram29] F. P. Ramsey. On a problem of formal logic. *Proc. of London Math.Soc.*, 30:264–286, 1929.
- [Ric78] M. M. Richter. *Logikkalküle*, volume 43 of *Leitfäden der Angewandten Mathematik und Mechanik LAMM*. Teubner, 1978.
- [SA91] Volker Sperschneider and G. Antoniou. *Logic, a foundation for computer science*. Addison-Wesley, 1991.
- [Sch95] E. Schröder. *Vorlesungen über die Algebra der Logik, vol.III, Algebra und Logik der Relative*. B.G. Teubner, 1895. Nachdruck: Chelsea Publishing Company, New York, 1966.
- [Sch92] Peter H. Schmitt. *Theorie der logischen Programmierung*. Springer Verlag, 1992.
- [Sch00] Uwe Schöning. *Logik fr Informatiker*. Spektrum Akademischer Verlag, 2000.
- [SGS⁺92] V. Stavridou, J. A. Goguen, A. Stevens, S. M. Eker, S. N. Abo-neftis, and K. M. Hobley. FUNNEL and 2OBJ: Towards an integrated hardware design environment. In Stavridou et al. [SMB92], pages 197–223.

- [Sha38] C. E. Shannon. A symbolic analysis of relay and switching circuits. *AIEE Transactions*, 67:713–723, 1938.
- [SMB92] V. Stavridou, T. F. Melham, and R. T. Boute, editors. *Theorem provers in Circuit Design, Proceed. of Internat. Conf. Nijmegen*. IFIP Transactions A-10. IFIP, 1992.
- [Smu68] Raymond Smullyan. *First-Order Logic*. Springer, 1968.
- [Smu95] Raymond M. Smullyan. *First-Order Logic*. Dover Publications, New York, second corrected edition, 1995. First published in 1968 by Springer.
- [Sta88] Richard Stallman. GNU Emacs Manual, sixth edition, emacs version 18. Technical report, Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, 1988.
- [Tar41] Alfred Tarski. On the calculus of relations. *J. of Symbolic Logic*, 6:73–89, 1941.
- [Tar72] R.E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal of Computing*, 1:146–160, 1972.
- [Tho90] Wolfgang Thomas. Automata on infinite objects. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science. Vol. B : Formal Models and Semantics*, pages 135–192. Elsevier, Amsterdam, 1990.
- [TZ71] G. Takeuti and W.M. Zaring. *Introduction to Axiomatic Set Theory*. Undergraduate Texts in Mathematics. Springer Verlag, 1971.
- [Win90] S. Winker. Robbins algebra: Conditions that make a near-boolean algebra boolean. *J. Automated Reasoning*, 6(4):465–489, 1990.
- [Win92] S. Winker. Absorption and idempotency criteria for a problem in near-boolean algebras. *J. Algebra*, 153(2):414–423, 1992.
- [WK99] Jos Warmer and Anneke Kleppe. *The Object Constraint Language: Precise Modelling with UML*. Object Technology Series. Addison-Wesley, Reading/MA, 1999.

Index

- $(A \downarrow B)$, 18
- $(A|B)$, 18
- At_Σ , 109
- $For0$, 13
- $For0_\Sigma$, 12
- For_Σ , 110
- $H \cong G$, 39
- Kal , 57
- $M \models_\Sigma A$, 21
- Var , 109
- \square , 68, 178
- Σ , 12
- $\beta_x^d(y)$, 126
- \cong , 39
- $\ddot{A}qFor$, 51
- \doteq , 108
- \downarrow , 18
- \exists , 108
- $\mathbf{0}$, 12
- \forall , 108
- λ -Kalkül, β -Reduktion, 220
- \wedge , 12
- \leftrightarrow , 12
- \neg , 12
- \vee , 12
- \models , 21, 55, 58
- \models_Σ , 21
- \rightarrow , 12
- $\xrightarrow{1}_E$, 221
- $\mathbf{1}$, 12
- \vdash , 177
- $\vdash_{\mathbf{T}}$, 161
- $\vdash_{\mathbf{T}} 0$, 81
- $\vdash_{\mathbf{H}}$, 177
- $mFor0$, 246
- $mFor0_\Sigma$, 246
- sh -Formel, 33
- sh -Formel, normierte, 34
- sh -Graph, 35
 - reduziert, 38
- $sh(P_1, P_2, P_3)$, 33
- $shFor0$, 33
- $true$, 12
- val_I , 16
- $val_{D,I,\beta}$, 126
- \mathbf{H} , 177
- $\mathbf{H0}$, 60
- $\mathbf{H0}_\Sigma$, 60
- \mathbf{H}_Σ , 177
- $\mathbf{R0}$, 69
- $\mathbf{S0}$, 90
- \mathbf{T} , 161
- $\mathbf{T0}$, 81
- \mathcal{D} , 128
- $|$, 18
- $($, 12
- $)$, 12
- Äquivalenz
 - logische, 21
- ALC, 206
 - Ausdrücke, 206
- SHOIQ, 208
 - Ausdrücke, 208
- 1-Resolution, 95
- 3-Bit-Ring-Zähler, 200
- 3-KNF, 48

Abkömmlinge, 78
 Abkürzungen, 14
 ableitbar aus M in Kal , 57
 Ableitbarkeit, 55
 Ableitbarkeit in $\mathbf{S0}$, 91
 Ableitung aus M in Kal , 57
 Ableitung in \mathbf{T} , 161
 Ableitungsmechanismus, 60
 Abschlußregel, 161
 Abschwächung, 19
 Absorption, 20
 AL, 6
 Algebra, Lineare, 196
 Allabschluß, 112
 allgemeingültig, 18, 134
 Allquantor \forall , 108
 antisymmetrische Relation, 1
 Äquivalenzformeln, 51
 Äquivalenzrelation, 2
 Assoziativität, 20
 asymmetrische Relation, 1
 Atom, 13, 68, 109
 Atom, aussagenlogisches, 109
 Aussage, atomare, 13
 Aussagenlogik, 6
 Semantik, 16
 Syntax, 12
 temporale, 261
 Aussagevariable, 13
 Auswertung einer Formel, 16
 Auswertung von Termen und Formeln,
 126
 axiomatische Fundierung, 60
 Axiome, 56, 60
 Axiome, bereichsspezifische, 60
 Axiome, nichtlogische, 60

 Büchi Automaten, 297
 erweiterte, 306
 Basis, 17
 bedingter Term, 142

 Berechnungsfolge, 297
 i-zyklische, 331
 akzeptierende, 331
 akzeptierte, 297
 endliche, 331
 zyklische, 331
 bereinigte Formel, 144
 Beschreibungslogik
 Syntax, 206, 208
 Beweisbaum, 59, 90, 190
 geschlossener, 90, 190
 beweisen, 55
 Beweisführung, 56
 Beweistheorie, 55
 Beweistheorie (PL1), 156
 Boole'sche Funktionen, 17
 bounded model checking, 337
 Box \square , 68, 246

 collection, 228
 Conclusio, 56
 constraint, 226

 Davis-Putnam-Verfahren, 95
 De Morgan'sche Gesetze, 20
 Deduktionstheorem der AL, 61
 Diamond \diamond , 246
 Dirichletsches Schubfachprinzip, 74
 Disjunktionen, 26
 disjunktiver Normalform (DNF), 26
 Distributivität, 20
 DL, 275
 DNF, 26
 Doppelnegation, 20

 Einschränkungen, 226
 Endlichkeitssatz, 67, 195
 Entscheidbarkeit
 von \mathbf{K} , 256
 erfüllbar, 18, 134
 Erfüllbarkeitsproblem, 48
 Ersetzungstheorem, 22, 136

Ex falso quodlibet, 19
 Existenzabschluß, 112
 Existenzquantor \exists , 108
 Existenzquantoren beseitigen, 147

 Fairnessforderungen, 172
 Faktorisierungsregel, 185
 Filtration, 255
 First-Order n -Damen Problem, 139
 Folgerbarkeit, 134, 140
 Folgerbarkeit, semantische, 21
 folgern, 55
 Formel, 110
 negationsduale, 258
 Formel, atomare, 109
 funktionale Relation, 2
 Funktionssymbole $f \in F_{\Sigma}$, 109
 FUNNEL, 200

 gebundene Umbenennung, 137
 geordnete Resolution, 75
 Gesamtheit, 228
 geschlossen, 112
 globales Modell, 139
 Grundinstanz, 151
 Grundliteral, 153

 Hülle
 transitive, 3
 transitive, reflexive, 3
 transitive, reflexive, symmetrische,
 3
 Herbrand, Satz von, 150, 151
 Herbrand-Algebra, 150
 Herbrand-Interpretation, 150
 Hilbertkalkül (AL), 60
 Hilbertkalkül (PL1), 177
 Hintikka-Mengen, 168
 Hoare-Kalkül, 132
 Homomorphismus, 5
 Horn-Formel, 48
 Horn-Formel, definite, 54

 Horn-Klausel, 48
 Horn-Klausel, definite, 54

 Idempotenz, 19
 Individuenvariable, 108
 Induktion, Noethersche, 217
 inkonsistent
 wechselseitig, 24
 Instanz, 56
 Interpolante, 23
 Interpretation, 124
 sortierte, 198
 Interpretation I über Σ , 16
 Interpretationsfunktion, 125
 Invarianten, 229
 irreduzibel, 215
 irreflexive Relation, 1
 isomorph, 39
 Isomorphismus, 5

 Königs Lemma, 173
 Kalkül, 55
 abstrakter, 56
 Hilbert \sim , AL, 60
 Hilbert \sim , PL, 177
 infinitärer, 281
 Resolutions \sim , AL, 70
 Resolutions \sim , PL, 179
 Sequenzen \sim , AL, 90
 Sequenzen \sim , PL, 188
 Tableau \sim , AL, 81
 Tableau \sim , PL, 161
 kanonisch, 215
 Klammereinsparungsregeln, 14, 110
 Klausel, 26, 68, 178
 isolierte, 46
 leere, 68
 Mengen von \sim_n , 68
 \sim normalform, 26
 tautologische, 47
 Klauselmenge

minimal unerfüllbare, 47
 KNF, 26
 Koinzidenzlemma, 128
 kollisionsfreie Substitution, 113
 Kommutativität, 20
 Kompaktheit (PL1), 195
 Kompaktheit der Ableitbarkeit, 58
 Kompaktheit der DL, 279
 Kompaktheit der PL2, 274
 Kompositionslemma, 57, 61
 konfluent, 215
 konfluent, lokal, 215
 Kongruenzrelation, 5
 Konjunktionen, 26
 konjunktiver Normalform (KNF), 26
 Konstante, 109
 Konstante, logische, 13
 Konstantensymbol, 109
 Kontext, 226
 Kontraposition, 20
 Korrektheit der DL, 280
 Korrektheit der PL2, 274
 Korrektheit von \mathbf{R} , 180
 Korrektheit von $\mathbf{R0}$, 71
 Korrektheit von \mathbf{S} , 190
 Korrektheit von $\mathbf{S0}$, 92
 Korrektheit von \mathbf{T} , 164, 167
 Korrektheit von $\mathbf{T0}$, 82
 Korrektheit von \mathbf{H} , 178
 Korrektheit von Kal , 58
 Korrektheitsaussagen, partiellen, 278
 Kripke-Struktur, 246, 261

- dichte, 251
- endlose, 251
- partiell funktionale, 251
- reflexive, 251
- symmetrische, 251

 Krom-Formel, 48
 kurze konjunktive Normalform, 30
 kurze konjunktive Normalform, 28
 leere Klausel \square , 178
 Linear Time Logic, 262
 Literal, 26, 68, 153, 178

- Grund \sim , 153
- isoliertes, 46
- komplementäres, 46
- negatives, 153
- positives, 153

 Logik, dynamische, 275
 Logik, Hoare-, 278
 Logiken, ordnungssortiert, 199
 Logiken, stärkere, 271
 logisch äquivalent, 21, 135
 logische Äquivalenz, 21
 Logische Zeichen, 12
 LTL, 262

 Metaebene, 24
 Metasprache, 1
 mgu, 117
 Modaloperator

- temporaler, 262

 Modell, 16, 82, 133
 Modell von T über M , 165
 Modell-Lemma, 170
 Modellprüfung

- limitierte, 337

 most general unifier, 117

 Nachbedingung, 229
 Nachfolger, unmittelbarer, 261
 negationsduale Formel, 258
 Negationsnormalform, 144, 320
 Nichtcharakterisierbarkeit von „endlich“ in der PL1, 195
 Nichtdefinierbarkeit, 195
 noethersch, 215
 Normalform, 215
 Normalform für s , 215
 Normalformen, 26, 144
 Notation, uniforme, 77

Notationsvarianten, 194
 Nullsemantik, 175

 Objektebene, 24
 Objektsprache, 1
 omega-Struktur, 262
 Operator, logischer, 13
 Ordnung, 2
 lineare, 2
 partielle, 2
 strikte, 2
 totale, 2
 Ordnungsrelation, 2
 Otter, 185

 Paramodulation, 187
 Pfad, 160
 Pfad, geschlossener, 160
 pigeon hole principle, 74
 PL1, 108
 PL1, modale, 275
 PL2, 271
 Polynomreduktion, 219
 Prädikatenlogik, 105
 Prädikatenlogik (erster Ordnung), 108
 Prädikatenlogik zweiter Ordnung, 271
 Präfix, 14
 Prämisse, 56
 Pränex-Normalform, 144
 Prädikatssymbole $p \in P_\Sigma$, 109
 Primimplikant, 27, 44
 essentieller, 27
 Prioritätsregeln, 14
 Promela, 339

 Quasiordnung, 2

 Reduktionssysteme, 215
 Reduktionssysteme, 215
 reflexive Relation, 1
 Regel, n -stellige, 56
 Regelsystem, 55

 reguläres Tableau, 86
 Relation
 Äquivalenz \sim , 2
 antisymmetrische, 1
 asymmetrische, 1
 irreflexive, 1
 reflexive, 1
 symmetrische, 1
 transitive, 1
 Resolution, 68
 Resolution, binäre, 180
 Resolutionskalkül, 70
 Resolutionskalkül \mathbf{R} , 179
 Resolutionsregel, 179
 Resolvente, 69, 179
 Resoulution
 geordnete, 75
 Robbins Algebra, 202
 run, 297

 SAT, 19, 48
 Schaltkreis, 200
 Schaltkreisen, Beschreibung von, 100
 Schema, 56
 schließen, 55
 Selbstbeweis, 20
 Selbstimplikation, 19
 Semantik der Aussagenlogik, 16
 Semantik der DL, 277
 Semantik der PL1, 124
 Semantik der PL2, 272
 Semi-Thue-Systeme, 220
 Sequenz, 89, 187
 Sequenzen, Auswertung von, 89, 188
 Sequenzenkalkül, 90, 187
 Beweisbaum, 90, 190
 Sequenzenkalkül \mathbf{S} , 188
 Sequenzenkalküle, 89
 Shannon Normalform, 33
 Signatur, 12, 68, 109
 Signatur Σ , sortierte, 197

Skolem-Normalform, 145, 148
 Sonderzeichen der AL, 12
 Sorten, 196
 Sortensymbole, 197
 Spektrum $spec(A)$, 141
 Sprache, 56
 Stelligkeit, 109
 Stereotypen, 229
 Struktur, 124
 strukturelle Induktion, 13, 110
 subsect:PLResolution, 178
 Substitution, 112

- Grund-, 112
- kollisionsfreie, 113
- Komposition, 113
- Variablenumbenennung, 114

 Substitutionslemma für Formeln, 131
 Substitutionslemma für Terme, 130
 symmetrische Relation, 1
 syntaktischer Formalismus, 56
 Syntax der Aussagenlogik, 12
 Syntax der DL, 276
 Syntax der PL2, 271

Tableau, 158

- analytisches, 84
- aussagenlogisches, 79
- erschöpftes, 83
- geschlossenes, 80, 161
- reguläres, 86

 Tableukalkül **T**, 157, 161
 Tableukalkül (AL), 77
 Tautologie, 18, 135
 Teilformel, 14, 110
 Teilformel, echte, 14
 Teilterm, 110
 Temporale Aussagenlogik, 261
 Term, 109

- bedingter, 142

 Term $_{\Sigma}$, 109
 Termersetzungssystem, 215

Termersetzungssysteme, 221, 222
 Tertium non datur, 19
 The Lion and the Unicorn, 102
 transitive Hülle, 3
 transitive Relation, 1

unerfüllbar

- minimal, 47

 Unifikation, 115
 Unifikationsalgorithmus von Robinson, 118
 Unifikator, 115

- allgemeinster, 117

 unifizierbar, 115
 unifiziert, 115
 Uniforme Notation, 157
 unit resolution, 95
 unmittelbarer Nachfolger, 261
 Unterterm, 110

Variable, 111
 Variable, freie, 111
 Variable, gebundene, 111
 Variablenbelegung, 124, 126
 Variablenumbenennung, 114
 Variante, 179
 Verifikation, 200
 Vollständigkeit der DL, 280
 Vollständigkeit der PL2, 274
 Vollständigkeit von **H0**, 66
 Vollständigkeit von **R0**, 71
 Vollständigkeit von **S**, 192
 Vollständigkeit von **S0**, 92
 Vollständigkeit von **T**, 171
 Vollständigkeit von **T0**, 84
 Vollständigkeit von **H**, 178
 Vollständigkeit von *Kal*, 58
 Vollständigkeit von **R**, 180
 Vorbedingung, 229
 Vorzeichenformel, 77

Wahrheitstafel, 17

Wahrheitswerte, 16
wechselseitig inkonsistent, 24
Widerlegungskalkül, 68, 77, 183
Wirkungsbereich eines Präfix, 111
Wissensrepräsentation, 101
wohlfundiert, 215

Zeichen

 logische, 12
Zeitstruktur, 261