Hochschule Wismar
University of Technology, Business and Design
Fachbereich Wirtschaft

Hochschule Wismar
University of Technology, Business and Design
Department of Business

Uwe Lämmel

Der moderne Frege

Heft 01 / 2004

W D P

Wismarer Diskussionspapiere / Wismar Discussion Papers

Der Fachbereich Wirtschaft der Hochschule Wismar, Fachhochschule für Technik, Wirtschaft und Gestaltung bietet die Studiengänge Betriebswirtschaft, Management sozialer Dienstleistungen, Wirtschaftsinformatik und Wirtschaftsrecht an. Gegenstand der Ausbildung sind die verschiedenen Aspekte des Wirtschaftens in der Unternehmung, der modernen Verwaltungstätigkeit im sozialen Bereich, der Verbindung von angewandter Informatik und Wirtschaftswissenschaften sowie des Rechts im Bereich der Wirtschaft.

Nähere Informationen zu Studienangebot, Forschung und Ansprechpartnern finden Sie auf unserer Homepage im World Wide Web (WWW): http://www.wi.hs-wismar.de/.

Herausgeber: Prof. Dr. Jost W. Kramer
Fachbereich Wirtschaft
Hochschule Wismar
University of Technology, Business and Design
Philipp-Müller-Straße
Postfach 12 10
D – 23966 Wismar
Telefon: ++49 / (0)3841 / 753 441
Fax: ++49 / (0)3841 / 753 131
e-mail: j.kramer@wi.hs-wismar.de

ISSN 1612-0884
ISBN 3-910102-42-5

JEL-Klassifikation C60, C80

Inhaltsverzeichnis

Vorwort

Gottlob Frege ist einer der größten (der größte?) Wissenschaftler Mecklenburg-Vorpommerns.

Als Mathematiker und Philosoph hat er die moderne mathematische Logik begründet, dazu Beiträge zur Mengenlehrer, Axiomatisierung sowie der Sprachphilosophie geleistet. Viele seiner Leistungen haben Spuren hinterlassen, die noch heute wirken, und dass nicht nur in der Mathematik und Philosophie sondern insbesondere auch in der Informatik.

Diese Schrift möge dazu beitragen, den Namen Gottlob Frege und sein Schaffen stärker in das öffentliche Bewusstsein zu rücken.

1. Gottlob Frege

Im November 1848 wurde Gottlob Frege in Wismar (Mecklenburg) geboren. „Wer ist Frege?“, werden viele fragen:

> *„dt. Mathematiker und Philosoph. ... Mitbegründer der modernen formalen Logik; leistete auch wichtige Vorarbeiten für die Sprachphilosophie.“*[1]
>
> *„German mathematician and logician, who founded modern mathematical logic. Working on the borderline between philosophy and mathematics ... Frege discovered, on his own, the fundamental ideas that have made possible the whole modern development of logic and thereby invented an entire discipline.“*[2]

Frege wird mit großen Philosophen wie Aristoteles oder Leibniz verglichen. Diese Namen sind geläufig, aber wer kennt Frege?

Gottlob Frege studierte in Jena und Göttingen Mathematik, Physik und Chemie sowie Philosophie. Im Jahre 1873 promovierte er in Jena und wurde ein Jahr später dort Professor, wo er bis 1917 lehrte. 1926 starb Frege in Bad Kleinen ca. 15km südlich seines Geburtsortes. Was hat diesen Wissenschaftler, der von seinen Kollegen in Jena eher schlecht beurteilt wurde, so berühmt werden lassen, dass seine Werke auch 100 Jahre später noch oder wieder veröffentlicht und gelesen werden?

Dieser Beitrag will insbesondere die Ideen des Mathematikers und Philosophen Gottlob Frege herausstellen, die heute ihre Anwendung in der Informatik finden. Informatik, eine zu Lebzeiten Freges unbekannte Wissenschaftsdis-

---

1 Meyers neues Lexikon in 10 Bänden, Meyers Lexikon Verlag 1993.

2 Encyclopædia Britannica, Chicago 1997, Micropaedia Bd. 4, S. 968-969: *„Deutscher Mathematiker und Logiker, der die moderne mathematische Logik begründetet. Frege arbeitete an der Grenze zwischen Philosophie und Mathematik ... und entwickelte, allein, die fundamentalen Ideen, die die gesamte Entwicklung der modernen Logik erst ermöglichten und begründetet somit eine ganze Disziplin.“*

ziplin hat zwei starke Wurzeln: Auf der technischen Seite die Elektrotechnik/Elektronik; auf der theoretischen Seite die Mathematik insbesondere die mathematische Logik. Freges Arbeiten stellten den entscheidenden Schritt zur Herausbildung der mathematischen Logik dar. Im ersten Teil wollen wir die Geschichte der Logik umreißen, um dann Freges Werk und die heutigen Auswirkungen aufzuzeigen.

Freges Versuche zur Beschreibung der Mathematik auf der Grundlage der Logik führten ihn auf das Problem der Zahldefinition. Er benutzte dazu die Mengenlehre. Ein kleiner Exkurs in die damals diskutierten Probleme schließt sich an. Aufgezeigt wird dabei auch, wie in der Informatik einige dieser Probleme heute umgangen werden.

Die Begründung der mathematischen Logik ging einher mit einer Trennung der Logik von der natürlichen Sprache. Die einem Satz innewohnende Bedeutung (Logik) trennte Frege deutlich von der sprachlichen Hülle (Zeichen). Syntax und Semantik spielen heute in der Programmierung eine große Rolle. Konzepte zur Beschreibung der Syntax von Programmiersprachen gehören mittlerweile zu den Klassikern in der Informatik. Die Semantik möglichst formal zu fassen, gehört immer noch zu den aktuellen Untersuchungen. Kapitel 4 beleuchtet dieses Gebiet.

Frege war ein Minimalist. Ob das seinen gesamten Lebensstil betrifft, wie seine überlieferten jährlichen Wanderungen von Jena nach Wismar vermuten lassen, sei einmal dahingestellt. Auf seinem Arbeitsgebiet stellte er sich das Ziel, die Logik und darauf aufbauend die Mathematik auf so wenige als nur irgend mögliche Grundannahmen zurückzuführen. Damit regte Frege Arbeiten auf dem Gebiet der Axiomensysteme, der Axiomatisierbarkeit von Theorien an. Axiomensysteme werden heute auch in der praktischen Informatik zur Konstruktion von Software benutzt

Mit diesen verschiedenen Blickwinkeln auf das Werk Freges soll ein Eindruck vermittelt werden, welchen gewaltigen Einfluss auf die Entwicklung der Logik, der Mathematik und damit auch auf das Entstehen der Informatik die Ideen von Gottlob Frege hatten. Informatik, die Verarbeitung von Informationen mittels Computer, setzt formale Beschreibungen voraus. Freges Werk kann man als einen kräftigen Baustein im Fundament der Informatik betrachten. Jede Software, die wir anwenden, wurde in einer Programmiersprache formuliert. Der Übersetzer für diese Programmiersprache hantiert formal mit Syntax und Semantik. Der formulierte Algorithmus nutzt die Logik für die Programmstruktur. Formale Spezifikationen für insbesondere sicherheitskritische Software basieren auf Axiomensystemen, wobei Eigenschaften der Software dann formal (unter Nutzung der Logik) bewiesen werden. Die Künstliche Intelligenz, ein Teilgebiet der Informatik, wäre ohne die Darstellung von Sachverhalten in Form logischer Aussagen kaum handlungsfähig. Frege hat für all diese Anwendungen Vorarbeiten geleistet, sein Hauptverdienst besteht

dabei darin, die mathematische Logik, die wir heute so vielfältig nutzen, auf den Weg gebracht zu haben.

2. Frege und die Entwicklung der Logik

2.1. Die Logik vor Frege

Die Geschichte der Logik beginnt bei Aristoteles, der bereits Schlussregeln aufstellte, die es erlauben, aus zwei Aussagen eine dritte Aussage, die Schlussfolgerung formal abzuleiten. Man geht dabei von bestimmt strukturierten Sätzen aus. Die Form des Schließens wird Syllogismus genannt. Es gibt vier Figuren, die die Struktur der Sätze festlegen:

M – P	P – M	M – P	P – M	M	- Mittler
S – M	S – M	M – S	M – S	P	- Prädikat
S – P	S – P	S – P	S – P	S	- Subjekt

Die Sätze werden nun unter Verwendung von Quantifizierungen, wie „Alle“ (Jeder), „Kein“, „Einige“ oder „Einige ... nicht“ gebildet:

Alle Vögel haben Federn.	Mittler M:	Vögel	Einige Vögel haben Federn.
Alle Spatzen sind Vögel.	Prädikat P:	haben Federn	Alle Spatzen sind Vögel
Alle Spatzen haben Federn	Subjekt S:	Spatzen.	Einige Spatzen haben keine Federn.

Um unsinnige Schlussfolgerungen (Beispiel rechts) auszuschließen, wurden für jede der vier Figuren die Modi aufgezählt, die einen korrekten Schluss ergeben.

1. Figur: Barbara, Celarent, Ferio, Darii
2. Figur: Cesare, Baroco, Festimo, Camestres
3. Figur: Bocardo, Felapton, Disamis, Darapti,Darisi, Ferison
4. Figur: Dimaris, Fesapo, Fresion, Cameres, Bamalip

Für die Bezeichnung der Modi verwendete man Namen, deren Vokale die notwendigen Quantifizierungen bestimmen: Es steht ein „a“ für „Alle“, ein „e“ für „Kein“, ein „i“ für „Einige“ und ein „o“ für Einige nicht“. Das linke Beispiel ist ein Syllogismus der ersten Figur mit dem Modus „Barbara“: Alle drei Aussagen haben die Form „Alle ...“ (a). Das Beispiel oben rechts würde man mit iai bezeichnen, aber es gibt keinen entsprechenden Modus für die erste Figur.

Zum Trainieren versuchen wir einmal den Modus „Ferison“ der dritten Figur.

(e) Keine ungerade Zahl ist durch 2 teilbar.
(i) Einige ungerade Zahlen sind durch 5 teilbar.

Daraus kann nun formal die Schlussfolgerung gebildet werden:

(o) Einige durch 5 teilbare Zahlen sind nicht durch 2 teilbar.

Es sei darauf verwiesen, dass vier dieser Modi aus der Sicht der mathematischen Logik nur dann korrekte Schlüsse bilden, wenn man Subjekte, Prädikate und Mittler benutzt, die wirklich existieren, also nicht einer leeren Menge entsprechen. Das folgende Beispiel verdeutlicht den Sachverhalt anhand der 3. Figur mit dem Modus „Darapti“:

Alle grünen Hunde haben 5 Beine.
Alle grünen Hunde sind Haustiere.
Einige Haustiere haben 5 Beine.

Jahrhundertelang war man der Meinung, dass das Gebiet der Logik mit dem Konzept der Syllogismen erschöpfend beschrieben ist. Im 19. Jahrhundert kam dann durch George Boole und danach durch Gottlob Frege aus unterschiedlichen Beweggründen und mit unterschiedlichen Ergebnissen Bewegung in die Entwicklung der Logik.

Das Ziel George Booles bestand darin, die Gesetze der Arithmetik auf die Logik anzuwenden, um so mit der Logik rechnen zu können. Ebenso wie Zahlen mittels arithmetischer Operationen miteinander verknüpft werden, gedachte er logische Werte miteinander zu verknüpfen. Dabei übernahm er die Symbole aus der Arithmetik mit folgender Zuordnung:

1 - wahr; 0 - falsch; + - oder; * - und

Dabei gilt die Festlegung, dass 1+1 wieder 1 ergibt. Die Negation eines Wertes A lässt sich leicht als 1-A beschreiben. Man sieht leicht, dass das Verwenden der arithmetischen Operationen die korrekte Verknüpfung von Wahrheitswerten erlaubt: 1*0=0 (wahr und falsch ergibt falsch)

Frege bezeichnete Booles Herangehen treffend als „rechnende Logik“. Booles Logik ist die Aussagenlogik. Sätze werden immer als Ganzes, als Aussagen betrachtet, denen ein Wahrheitswert „wahr“ oder „falsch“ zugeordnet werden kann: „Die Zahl 3 ist eine gerade Zahl.“ Mit den Wahrheitswerten kann dann gerechnet werden. Aussagen, wie sie in den Syllogismen verwendet werden oder auch die folgende Aussage, können mittels der Booleschen Logik nicht dargestellt werden:

„Alle Zahlen, die sowohl durch 2 teilbar als auch durch 3 teilbar sind, sind auch durch 6 teilbar.“

Der Name Boole hat sowohl Eingang gefunden in die Mathematik als auch in die Informatik. Funktionen, deren Argumente bzw. Funktionswerte ausschließlich die Werte 0 oder 1 sind, werden als Boolesche Funktionen bezeichnet. Der Informatiker kennt den Datentyp „boolean“ aus vielen Programmiersprachen. Dieser setzt sich aus den beiden Wahrheitswerten TRUE und FALSE und den zugehörigen logischen Operationen zusammen, die meist allerdings mit Worten wie „AND“, „OR“ bezeichnet werden. Jede Program-

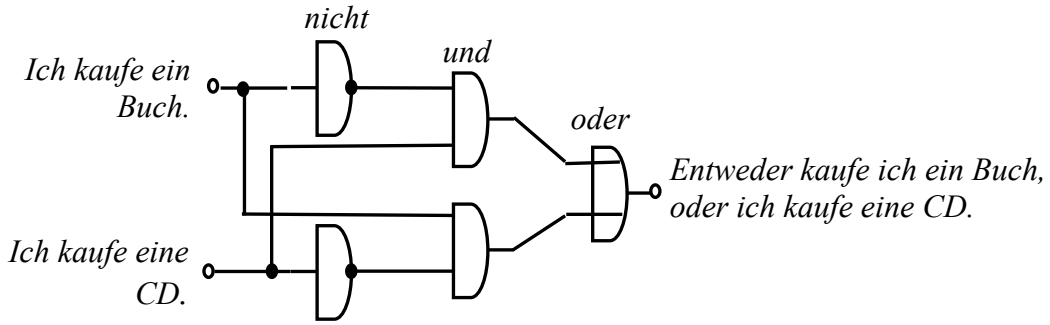
miersprache benutzt den Datentyp „boolean“, auch wenn ältere Sprachen keinen logischen Datentyp kennen. Jede Programmiersprache kennt Vergleichsausdrücke in Anweisungen, wie IF-THEN-ELSE oder WHILE-DO bzw. REPEAT-UNTIL:

```
WHILE a<>b DO
 IF a>b THEN a:=a-b ELSE b:=b-a;
```

In der Programmiersprache C z. B. wird jeder Wert ungleich 0 als „wahr“ interpretiert, die 0 wie oben als „falsch“. Leider sind die arithmetischen Operationen damit nicht ganz logisch korrekt. 1+ -1 =0 wäre dann zu interpretieren als wahr und wahr ergibt falsch. Solche Ungereimtheiten haben der Beliebtheit der Sprache C allerdings keinen Abbruch getan.

Nicht nur in der Programmierung ist die Boolesche Logik lebendig, im Schaltkreisentwurf benutzt man die Schaltalgebra, eine direkt auf die boolesche Logik zurückgehende Technik, wobei für die logischen Operationen Symbole eingeführt werden. Die Verbindungen deuten die Datenleitungen im Schaltkreis an:

Abbildung 1: Beispiel Schaltalgebra



2.2. Freges Beitrag

Frege hatte ein anderes Ziel für seine Arbeiten. Er wollte eine Möglichkeit schaffen, wissenschaftliche (mathematische) Beweisführungen nachvollziehbar, d. h. wiederum selbst beweisbar zu machen. Grundlage dafür muss eine von den Widrigkeiten der natürlichen Sprache losgelöste Darstellung von Sachverhalten sein. Darauf kann man dann formale Schlussmechanismen anwenden, die dann gewährleisten, dass wieder wahre Aussagen entstehen. In diesem Sinne sah er die Logik als Fundament an, auf dem sich die Mathematik aufbauen lassen sollte. In seiner auch heute wieder verlegten Begriffsschrift stellt Frege seine Logik vor, die auf drei Pfeilern steht:

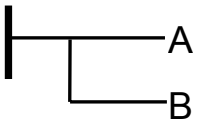
- Implikation (Wenn-Dann),
- Negation und
- Allquantifizierung (Für alle gilt).

Die Implikation stellt Frege an den Anfang:

A wird bejaht, und B wird bejaht.
A wird bejaht, und B wird verneint.
A wird verneint, und B wird bejaht.
A wird verneint, und B wird verneint.

Frege zählt die Möglichkeiten der Bewertung zweier „*beurtheilbarer Inhalte*“ auf und legt fest. Wird nun der dritte dieser Fälle verneint und alle anderen bejaht, so wird diese Beziehung wie folgt dargestellt:

Abbildung 2: Die Implikation in der Darstellung Freges:



Der waagerechte Strich ist dabei der Inhaltsstrich: ——*A*,
der Inhalt A: ——*ungerade(3)*
Versehen mit dem senkrechten Strich, dem Urteilsstrich │, erhalten wir dann ein Urteil, eine Aussage:

Es ist wahr, dass A gilt.
Es ist wahr, dass 3 eine ungerade Zahl ist.

Die in Abbildung 2 dargestellte Verknüpfung zweier Inhalte entspricht der Implikation, der Wenn-Dann-Beziehung: Wenn B dann A.

Etwas abweichend vom umgangssprachlichen sind die Fälle, in denen B falsch ist, dann aber die Gesamtaussage wahr ist:

Wenn ein Hund fünf Beine hat, dann scheint die Sonne.

Umgangssprachlich eher als unsinnig einzustufen, ist diese Aussage aus Sicht der mathematischen Logik wahr. Die Schlussfolgerung „Wenn B Dann A.“ ist nur dann falsch, wenn B wahr und A falsch ist.

Die von Frege eingeführte graphische Darstellung logischer Ausdrücke stellt einen enormen Fortschritt dar, erlaubt es doch, Sachverhalte losgelöst von der natürlichen Sprache formal darzustellen. Gleichzeitig sind Gedankengänge auf diese Weise darstellbar und auch nachvollziehbar. Allerdings verliert Freges grafische Darstellung schon für kleine Ausdrücke schnell ihre Übersichtlichkeit. Daher hat sich diese Notation nicht durchgesetzt.

Die grafische Notation Freges wird anhand einiger logischer Ausdrücke il-

lustriert. Dazu wird der heute üblichen Schreibweise die grafische Darstellung Freges gegenübergestellt:

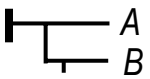
$\neg$ A  *Es gilt nicht A. (A gilt nicht.)*

Frege hat kein besonderes Symbol für die UND-Verknüpfung eingeführt. Somit muss diese Verknüpfung allein durch die Implikation und Negation ausgedrückt werden. Die Darstellung Freges lässt sich mittels der DeMorganschen Regel ableiten:

$(A \wedge B)$ ist identisch mit $(\neg (\neg A \vee \neg B))$, was wiederum identisch ist mit $(\neg (B \rightarrow \neg A))$, einem Ausdruck, der nun ausschließlich die Negation und Implikation verwendet.

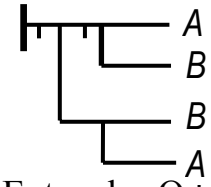
$A \wedge B.$  *A gilt und B gilt.*

In ähnlicher Weise wie vorher lässt sich aus $(A \vee B)$ der Ausdruck $(\neg B \rightarrow A)$ ableiten.

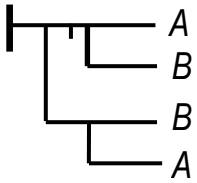
$A \vee B.$  Es gilt A oder B.

Sieht die folgende Formel Freges auf den ersten Blick auch recht kompliziert aus, so offenbart sie hier eine zugrundeliegende Struktur, die häufig für Beweise benutzt wird: $(A \rightarrow B) \wedge (B \rightarrow A)$

Man vergleiche dazu die Darstellung Freges für die UND-Verknüpfung.

$A \leftrightarrow B.$  Es gilt A genau dann wenn B gilt.

Für das Entweder-Oder hat sich kein bestimmtes Zeichen in der heutigen Logik durchgesetzt. In der Informatik wird deshalb oft „xor“ benutzt. Unter Beachtung, dass „Entweder-Oder“ die Negation von „Genau-Dann-Wenn“ ist, sollte das Diagramm aus dem vorherigen leicht ableitbar sein.

A xor B.  Es gilt entweder A oder B.

Sieht man von der Unterscheidung zwischen Inhalt und Urteil ab, so lassen sich alle gezeigten Beispiele ebenso mit der Boolschen Logik ausdrücken. Eine entscheidende Neuerung in der Logik Freges war die Einführung einer Symbolik für die Formulierung „Für alle gilt...“. Werden Sätze mit Variablen gebildet, so sind diese i. a. keine Aussagen. Dem Satz „X ist eine Primzahl“,

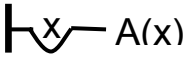
kann kein Wahrheitswert zugeordnet werden, es ist keine Aussage. Aussagen lassen sich ableiten, indem man die Variable durch Werte ersetzt:

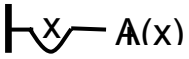
„899 ist eine Primzahl.“

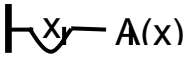
oder indem die Variable quantifiziert wird:

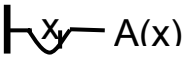
„Für alle X gilt,...“, *„Für ein X gilt...“*, *„Für kein X gilt....“* .

Frege erkannte, dass sich alle Formen, die in den Syllogismen verwendet werden (alle, einige, kein, einige nicht), sich auf eine Form zurückführen lassen, wenn man dazu die Negation zu Hilfe nimmt:

Für alle x gilt A(x). 

Für kein x gilt A(x).  *(Für alle x gilt nicht A(x).)*

Für einige x gilt A(x).  *(Es gilt nicht, dass für alle x nicht A(x) gilt.)*

Für einige x gilt A(x) nicht.  *(Es gilt nicht, dass für alle x gilt A(x).)*

Damit war es erstmals möglich, auch komplexere Aussagen formal zu fassen. Ein weiterer Beitrag Freges wird in den obigen Diagrammen bereits deutlich. Frege übernahm das aus der Arithmetik bekannte Konzept der Funktionen mit Argumenten *f(x) = 3x+4* in die Logik. Als Beispiel mögen einige bereits angeführte Sätze dienen: Der Satz „*X ist eine Primzahl*.“ kann unter Verwendung der Funktion „*ist_primzahl*“ formal dargestellt werden als: *ist_primzahl(x)*. Frege erkannte aber auch die damit verbundenen Probleme. Wie kann der Inhalt des Satzes *„x ist größer als 7“* dargestellt werden? Zwei Möglichkeiten springen ins Auge: *größer_als_7(x)* oder auch als zweistellige Funktion: *größer(x,7)*. Für einen Satz wie *„Hamburg hat mehr Einwohner als Wismar“* lassen sich dann noch weitere Möglichkeiten angeben. Frege bemerkte dazu:

„*Indem man einen Ausdruck in dieser Weise veränderlich denkt, zerfällt derselbe in einem bleibenden Bestandtheil, der die Gesamtheit der Beziehungen dargestellt, und in das Zeichen, welches durch andere ersetzbar gedacht wird, und welches den Gegenstand bedeutet, der in diesen Beziehungen sich befindet. Den ersten Bestandtheil nenne ich Function, den letzteren Argument.*“[3]

Die Frage nach der Art der Abstraktion kann nicht eindeutig beantwortet werden. Es liegt am Geschick des Bearbeiters, eine günstige Zerlegung in Funktion und Argumente zu erkennen. Das gilt damals wie heute.

Frege begründete mit seinen hier vorgestellten Ideen die mathematische

---

[3] Frege, G.: Begriffsschrift. Eine der arithmetischen nachgebildete Formelsprache des reinen Denkens. Halle 1879, zitiert nach Mayer, V: Gottlob Frege. München 1996, S. 58.

Logik, genauer den heute als Prädikatenlogik erster Stufe bezeichneten Kalkül.

2.3. Die Logik nach Frege

Nach Frege hat der italienische Mathematiker Peano eine Notation für die Logik vorgestellt, die auch heute noch Verwendung findet. Man verwendet dabei die Symbole $\wedge$ (und), $\vee$ (oder), $\neg$ (nicht), $\rightarrow$ (Implikation),$\forall$ (Für alle), $\exists$ (es existiert ein).

In der ersten Hälfte des letzten Jahrhunderts gab es eine enorme Entwicklung in der Mathematik, die letztlich auf den Arbeiten Freges aufbaut. Logik und Mathematik wurden in ein einheitliches Konzept gebracht. Es wurden Fragestellungen wie die nach der Berechenbarkeit von Funktionen, der Axiomatisierbarkeit von Theorien bearbeitet. Die gerade auch für die Informatik wichtige Fragestellung: „Was ist berechenbar?“, wurde in den dreißiger Jahren durch die Arbeiten von Alan Turing, Markov und Church beantwortet.

Betrachtet man die Verbindung von Logik und Informatik, so gab es Mitte der 60er Jahre einen weiteren Impuls. A. Robinson stellte einen Mechanismus vor, die sogenannte Resolution, mit dem es möglich wurde, aus gewissen Aussagen (Klauseln) neue Aussagen automatisch abzuleiten. Damit waren die theoretischen Voraussetzungen gegeben, den Computer nicht nur für das Rechnen, englisch wunderschön als „number crunching“ bezeichnet, sondern auch für das Bearbeiten logischer Probleme, z. B. dem Beweisen von Aussagen einzusetzen.

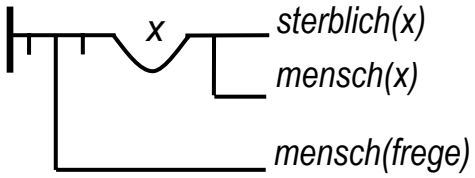
Anfang der 70er Jahre folgte dann der praktische Schritt durch R. Kowalski und A. Colmerauer, die die Programmiersprache PROLOG- PROgramming in LOGic, entwickelten. In der Begriffswelt der Sprache PROLOG „programmiert“ man, indem die gegebene Situation mittels Aussagen (Klauseln) beschrieben wird. Dann kann das System zum Beweisen von Aussagen zu dieser Situation benutzt werden. Dabei ermittelt das System auch die Wertebelegung von Variablen, die zu einer wahren Aussage führen. Nehmen wir dazu ein Beispiel, welches in ähnlicher Weise schon von Frege benutzt wurde:

(1) Alle Menschen sind sterblich.
(2) Frege ist ein Mensch.

Eine Behauptung, die PROLOG auf der Basis von (1) und (2) beweisen kann ist: Frege ist sterblich. Übersetzen wir diese Aussagen einmal sowohl in die Peano und Frege Notation als auch in die PROLOG-Sprache, so erhalten wir:

Peano: $\forall x\,(mensch(x) \rightarrow sterblich(x)) \wedge mensch(frege)$

Frege:



PROLOG: *sterblich(X) :- mensch(X).* *(Eine Regel: X ist sterblich, falls X ein Mensch ist.)*

mensch(frege). *(Ein Fakt gilt immer ohne Bedingung: Frege ist ein Mensch.)*

Die automatisch verarbeitbare Form von Aussagen ist die Klauselform, eine Form, deren Grundstruktur eine Implikation ist, wobei nur ein positives Literal[4] auf der DANN-Seite stehen darf. Im Bedingungsteil dürfen mehrere Literale mittels UND verknüpft werden. Alle auftretenden Variablen (mit Großbuchstaben bezeichnet) sind a priori allquantifiziert. Schade eigentlich, dass Frege keine Notation für die UND-Verknüpfung in sein Konzept eingebaut hat. Seine Notation ist der Klauselnotation der Sprache PROLOG sehr ähnlich. Die Frage, „Existiert ein X für das gilt, X ist sterblich?“ wird nun formuliert als:

?- sterblich(X).

?- sterblich(X) *bedeutet* $\exists$ X sterblich(X*) und wir erhalten als Negation:*

$\neg \exists$ X sterblich(X*), was gleichbedeutend ist mit:* $\forall$ X $\neg$ sterblich(X).

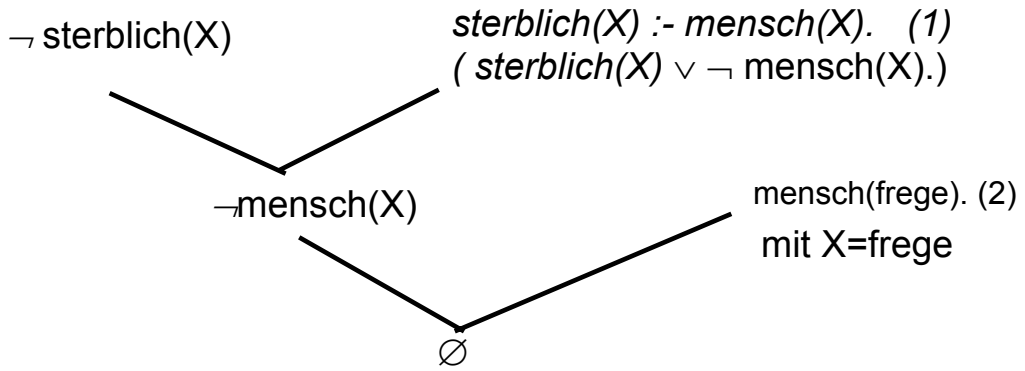
Diese Form wird nun mit den vorhandenen Aussagen verknüpft, siehe Abbildung 3.

Mittels der Belegung „frege“ für die Variable X wird die leere Menge von Klauseln, ein Widerspruch in der Klauselmenge, abgeleitet. Da das Ergebnis mit der Negation der Behauptung zustande kam, kann umgekehrt geschlussfolgert werden, dass die Behauptung *sterblich(X)* für *X=frege* richtig ist. Das System kann Behauptungen beweisen. Frege wäre sicherlich begeistert von derartigen Möglichkeiten gewesen. Voraussetzung dafür ist die Formalisierung von Sachverhalten, eine Formalisierung die von ihm eingeführt hat.

---

[4] Literal: Teil einer logischen Formel, der weder UND bzw. ODER Verknüpfungen noch Quantifizierungen der Variablen enthält. Beispiele sind: mensch(X), groesser(7,Y), farbe(blau), oma(anne,eva), ...

Abbildung 3: Resolutionsbeweis



Die Entwicklung der Logik ist aber auch heute bei weitem noch nicht abgeschlossen. In der Künstlichen Intelligenz versucht man, menschliches Wissen im Computer abzubilden und damit zu arbeiten: Im täglichen Leben haben wir es aber oft genug mit Wahrheiten zu tun, die nicht ewig gelten, sondern temporär sind: *„Anne ist eine Schülerin“*, ist eine Aussage, die nur begrenzt gültig ist. Die temporäre Logik bietet zusätzliche Operatoren zur logischen Beschreibung zeitlich begrenzter Sachverhalte.

Zum Glück ist unsere Welt nicht nur schwarz-weiß oder falsch-wahr. Die klassische Frage dazu lautet: Wann ist ein Mensch alt? Sind Sie ein alter Mensch? Bin ich alt? Mein Kind sagt ja, mein Vater nein. Die unscharfe Logik (fuzzy-Logik) hat bereits Einzug gehalten in die Steuerung von Geräten, die wir täglich benutzen, sei es zur Steuerung des Wasserverbrauches einer Waschmaschine oder die Scharfeinstellung einer Kamera.

3. Logik und Mengenlehre

Frege verfolgte das Ziel, die Arithmetik mittels der Logik zu beschreiben. Fundamental dabei ist der Begriff der Zahl. War man bereits zuvor zur Auffassung gelangt, *„daß die einzelnen Zahlen am besten ... aus der Eins und der Vermehrung um eins abgeleitet werden“,*[5] so blieb doch die Erklärung für die Eins und die Vermehrung um eins offen. Seinem Ziel verbunden, sollte die Zahldefinition nur mit den vorhandenen Möglichkeiten der Mathematik und Logik definierbar sein. Frege gab als erster eine exakte Definition der Zahl Null unter Verwendung der Mengenlehre an. Damit war der Zahlbegriff erstmals formal gefasst: *„Null ist die Anzahl, welche dem Begriffe »sich selbst ungleich« zukommt.“* (Ebd., S. 107) Damit wird die Null als Mächtigkeit (bei Frege Begriffsumfang) der leeren Menge definiert, der Aussage selbst liegt der einfache (syntaktische) Gleichheitsbegriff zugrunde.

---

[5] Frege, G: Grundlagen der Arithmetik, Stuttgart 1987, S. 48.

Einen „Begriffsumfang“ versuchte Frege durch ein Axiom zu beschreiben. Dieses fünfte Axiom in seinem Werk „Grundgesetze der Arithmetik“ lautet:

Der Umfang eines Begriffes P ist gleich dem Umfang eines Begriffes Q, wenn alle Gegenstände x, die unter P fallen, auch unter Q fallen, und umgekehrt.

Frege zweifelte selbst etwas an diesem fünften Axiom, B. Russel zeigte dann auch die Widersprüchlichkeit: Es ist möglich einen Begriffsumfang wiederum als Gegenstand zu verwenden. Diese Überlegung hat als *Russelsche Mengenantinomie* Eingang in die Mathematik gefunden:

> *Man betrachte die Menge M aller derjenigen Mengen K, die sich nicht selbst als Element enthalten.*
> *Wohin gehört nun die Menge M selbst? Ist M Element von M?*
> *Ja, dann enthält es sich selbst als Element und würde demzufolge nicht zu M gehören! Gut, M ist kein Element von M. Dann erfüllt aber M die Mengenbedingung, M muss zu M gehören. Widerspruch!*

Und nun? Die Informatik benutzt die auf der Mengenlehre aufbauende Typtheorie. In der Programmierung werden Variable (Speicherplätze) benutzt, die einen bestimmten Typ (logische Struktur) besitzen. Schließlich muss man mit einer Zahl anders umgehen als mit einer Zeichenfolge. Ein modernes Typkonzept verhindert die Russelsche Antinomie, indem prinzipiell nur Elemente „niederer“ Stufe Elemente eines komplexeren Typs (einer Menge) sein dürfen:

```
TYPE
  TElement = INTEGER;
  TMenge = ARRAY[1..Cmax] OF TElement; { Typ: Menge ganzer Zahlen }
```

Die Programmiersprache erlaubt dann nicht, TElement unter Zuhilfenahme von TMenge zu definieren. Selbst Typsysteme, die variable Typen (Typvariable) erlauben, vermeiden die Antinomie, indem Hierarchien einen Zirkelschluss verhindern:

```
typevar  Element;
type   TMenge = list(Element);
          { Typ: Mengenelemente beliebig, aber nicht TMenge}
```

Auch hierbei stößt man also bei einigen Nachforschen auf die Wirkung der Arbeiten von Gottlob Frege.

4. Syntax und Semantik

Der Name Frege ist unter Philosophen weitaus breiter bekannt als unter Mathematikern oder gar Informatikern. Ursache dafür ist sicher einerseits, dass

die deutschen Mathematiker die Bedeutung lange nicht richtig einschätzten. Grund ist andererseits, dass Frege sich intensiv mit der Analyse der Sprache beschäftigte, um daraus das für die Logik wesentliche herauszuarbeiten. Damit drang er in ein Gebiet ein, welches traditionell von Philosophen bearbeitet wurde: Sprache und ihre Bedeutung.

Anders als viele Philosophen sieht Frege jedoch das Verhältnis zwischen Zeichen, Sätzen, deren Sinn und Bedeutung in erster Linie aus dem Blickwinkel des Mathematikers, oder besser des Logikers. Bereits in der Begriffsschrift argumentiert er für eine Trennung der Logik von der natürlichen Sprache. Ein und derselbe Sachverhalt lässt sich oft auf verschiedene Art und Weise formulieren, wird aber aus logischer Sicht keinen Unterschied darstellen:

Brasilien ist der Fußballweltmeister von 1994.
Brasilien hat die Fußballweltmeisterschaft 1994 gewonnen.
Die Fußball WM 1994 wurde von Brasilien gewonnen.

Frege unterscheidet nun weiter zwischen dem Sinn eines Satzes und der Satzbedeutung. Die Satzbedeutung ist ein Wahrheitswert. Ein Satz ist entweder wahr oder falsch. Damit haben alle wahren Sätze dieselbe Bedeutung, sie sind einfach nur wahr. Der Sinn eines Satzes muss wohl aber etwas mehr enthalten als nur einen logischen Wert. Dazu wird von den sprachlichen Eigenheiten abstrahiert. Alle drei Sätze lassen sich in dieselbe Struktur bringen:

fußballWM(1994,brasilien).

Eine andere Abstraktion wäre genauso denkbar:

wm(fußball,1994,brasilien).

Frege sah Unterschiede zwischen einem Satz, einem Eigennamen und einem Begriffswort. Letzterer ist ein in der Logik wichtiger Begriff, wie schon in Kapitel 3 deutlich wurde. In der Abbildung 4 werden neben dem Schema Freges Entsprechungen aus Programmiersprachen angeführt.[6]

In der Informatik haben wir es mit Algorithmen zutun, die in einer Programmiersprache abgefasst wurden. Das sind in erster Linie nur Zeichenfolgen, deren Struktur (Syntax) vom Übersetzerprogramm erkannt werden muss. Einige semantische Prüfungen können ebenfalls maschinell erledigt werden, so bspw. die Typkontrolle, die sicherstellt, dass man nicht Äpfel mit Birnen vergleicht oder verarbeitet: Die Anweisungsfolge `a:='hurra'; b:=2.75*a;` multipliziert eine reelle Zahl mit einem Text, ergibt somit keinen Sinn und ist ein normalerweise semantisch falscher Satz einer Programmiersprache.

---

[6] Zitiert nach Mayer, V.: Gottlob Frege, München 1996, S. 119.

Abbildung 4: Freges Begriffswelt und Entsprechungen in Programmiersprachen

	Satz	**Eigenname**	**Begriffswort**	
	↓	↓	↓	
	Sinn eines Satzes (Gedanke)	Sinn des Eigennamens	Sinn des Begriffswortes	
	↓	↓	↓	
	Bedeutung (Wahrheitswert)	Bedeutung (Gegenstand)	Bedeutung → (Begriff)	Gegenstand, der unter den Begriff fällt
PROLOG:	Klausel (Regel/Fakt)	Konstante (Atom)	Literal (Menge der Variablenbelegung, für die das Literal WAHR wird)	
	sterblich(X) :- mensch(X)	*frege*	*mensch(X)*	
Programmiersprache:	Anweisung	Konstante	Datentyp	
	IF a>1.0 THEN...	*1.0*	*INTEGER*	

Die Bedeutung eines Programms, ist zum einen das Ergebnis der Übersetzung, das Programm im Maschinencode. Zum anderen ist die Bedeutung dann letztlich durch die bei der Ausführung des Programms erzeugten Ergebnisse gegeben.

Auch heute wird einerseits immer wieder die Trennung von Syntax und Semantik betont, andererseits ist jeder von uns an einer Vermischung von Syntax und Semantik interessiert. Die einfache Bedienbarkeit eines technischen Gerätes, die leichte Verständlichkeit einer Software entsteht gerade dadurch, dass die Syntax die Semantik bestmöglich wiedergibt.

Eine Forderung an einen guten Programmierstil, die oft leider nicht erfüllt wird, lautet:

Benenne eine Variable so, dass deren Bedeutung aus dem Namen ersichtlich ist.

Erkennen Sie die Bedeutung der beiden Programmstücke? In beiden Fällen enthält die Variable a am Ende den Wert des größten gemeinsamen Teilers der Eingabewerte a und b.

```
IF a=b  THEN ggT:=a
        ELSE IF a>b THEN ggT:= ggT(a-b,b)
                    ELSE ggT:= ggT(a,b-a);
```

```
REPEAT
p:=a MOD b;
a:=b;
b:=p;
UNTIL p=0;
```

Auf eine völlig andere Ausnutzung von Syntax und Semantik zielt das Programm Eliza von J. Weizenbaum7. Das Programm ermöglicht einen Dialog mit einem „Psychologen“. Das Programm ist der „Psychologe“. Das erstaunt etwas, funktioniert leidlich, wenn man sich an die Spielregeln hält: Eingabe fehlerfreier englischer Sätze:

Patient:	I am so sad.
Eliza:	*Why do you say you are so sad?*
Patient:	My mother is ill.
Eliza:	*Tell me something about your family.*
Patient:	My father is a teacher, my mother too.
Eliza:	*You seem to dwell on your family.*
Patient:	My mother is in a hospital.
Eliza:	*Are you hung up on your family?*
Patient:	No, I tell you, why I'm sad.
Eliza:	*Is it because you are sad that you came to me?*
Patient:	I hope you can help me.
Eliza:	*Earlier you said you are sad?*

Die Eingabe deutscher Sätze macht den Spuk deutlich: Eliza „versteht“ nichts, das Programm sucht nach bestimmten Worten (Zeichenfolgen) in der Eingabe und formuliert daraus eine Antwort.

Patient:	Meine Mutter ist krank.
Eliza:	Why do you say meine mutter ist krank?
Patient:	Sie liegt im Krankenhaus.
Eliza:	Is it because of your plans that you say sie liegt im krankenhaus?

Es ist also reinste Zeichenverarbeitung mit sehr wenig Bedeutung. Weizenbaum hatte keineswegs den Anspruch, einen Menschen nachzubilden, er wollte eher auf die Unsinnigkeit dieses Ansinnens aufmerksam machen.

---

7 Eliza ist als Bestandteil des Texteditors „Emacs“ in jedem UNIX-Betriebssystem vorhanden, z. B. auch im System Linux.

5. Axiome

Wie bereits erwähnt, war es Freges erklärtes Ziel, die Mathematik durch die Logik zu beschreiben. Er war auf der Suche nach einem minimalen Gerüst, aus dem sich alles weitere logisch einwandfrei nachvollziehbar ableiten lässt. Möglichst minimal, natürlich vollständig und widerspruchsfrei sollte diese Ausgangssituation sein: *Ein Axiomensystem.*

Freges Arbeiten haben Untersuchungen zur Formalisierung der Logik und Mathematik entscheidend beeinflusst, obwohl oder vielleicht gerade weil sein Gerüst nicht erfolgreich war. Wir sprechen heute davon, dass eine Theorie axiomatisierbar ist, wenn man ein Axiomensystem aufstellen kann, aus dem sich alle wahren Aussagen der Theorie ableiten lassen. Nach Frege wurde dann erkannt, dass sich nicht alles axiomatisieren lässt.

Wo ist der Einfluss auf die Informatik? Axiomensysteme spielen in der formalen Spezifikation von Software eine große Rolle. Eigenschaften der auf solche Weise spezifizierten Software lassen sich formal beweisen, was insbesondere für sicherheitskritische Anwendungen wichtig ist. Als Beispiel betrachten wir Spezifikationen auf der Basis funktionaler Programmiersprachen. Dabei wird in Funktionsdefinitionen programmiert. In Analogie zur Logik Freges schreiben wir hier ein Axiomensystem in einer funktionalen Sprache auf:

```
data Wahrwert = wahr
dec nicht : Wahrwert -> Wahrwert;
dec imp: Wahrwert X Wahrwert -> Wahrwert;
--- imp(wahr, x) <= x;
--- imp(x,wahr) <= wahr;
```

Dabei kann das Wort „wahr“ nicht weiter definiert werden, also wird es als Grundelement des Begriffs (Typ) Wahrwert definiert. „wahr“ ist somit eine Konstante, die als Argument für die Funktionen „nicht“ bzw. „imp“(-likation) benutzt werden kann. Frege bemerkte zum Wort „wahr“:

> *„Für die Logik kann das Wort „wahr” dazu dienen, ein solches (Ziel) kenntlich zu machen, in ähnlicher Weise wie „gut” für die Ethik und „schön” für die Ästhetik. ... Es wäre nun vergeblich, durch die Definition deutlicher zu machen, was unter „wahr” zu verstehen sei. ... Wahrheit ist offenbar etwas so Ursprüngliches und Einfaches, daß eine Zurückführung auf noch Einfacheres nicht möglich ist.“*[8]

Das Axiomensystem besteht somit aus der Festlegung von Sachverhalten, die

[8] Frege, G.: Logik [1897] in Schriften zur Logik, Akademie-Verlag, Berlin 1973.

sich nicht auf Einfacheres zurückführen lassen. Im Beispiel sind dies die Datentypen (Begriffe im Sinne Freges) und die Funktionsdefinitionen, die als Axiome fungieren.

Nun beschäftigt sich die Informatik nicht damit, die Logik noch einmal zu definieren. Ein weiteres Axiomensystem zeigt einen in der Informatik häufig benutzten Datentyp, einen Keller. Die Eigenschaft eines Kellers besteht darin, dass man Elemente speichern (push) kann, aber immer nur das zuletzt gespeicherte Element wieder betrachten (top) bzw. auch entfernen (pop) kann.

```
typevar Elem;
data stack == emptystack ++ st(Elem X stack);
dec init : stack;
dec push: stack X Elem -> stack;
dec top : stack -> Elem;
dec pop : stack -> stack;
--- init <= emptystack;
--- push(x,e) <= st(e,x);
--- top(st(e,x)) <= e;
--- pop(st(e,x)) <= x;
```

Werden Programme auf die gezeigte Art und Weise aufbauend auf einem Axiomensystem entwickelt, so können dann Programmeigenschaften formal abgeleitet, somit bewiesen werden. Dies ist für sicherheitskritische Anwendungen von großer Bedeutung.

Auch hier wieder eine eher ungeahnte Verbindung zum Werk Freges.

6. Schlussfolgerung

Freges Ideen leben weiter, wenn man sich die Wirkung seiner Arbeiten bis in die heutige Zeit hinein vor Augen führt. Es bestand nicht der Anspruch, Freges Werk in seiner Gesamtheit zu würdigen. Vielmehr sollten insbesondere die Leistungen Freges dargestellt werden, die heute in der Informatik zu spüren sind:

Frege	Informatik
Begriffsschrift(Logik)	Logische Programmierung
Begriff(Menge)	Datentypen/Typtheorie
Grundgesetze der Arithmetik/Axiomensysteme	formale Spezifikationen/ funktionale Programmierung
Sinn und Bedeutung	Syntax und Semantik von Programmiersprachen

Es gibt wohl mehr als historische Gründe, sich mit dem Werk von Gottlob Frege auseinander zu setzen. Es bleibt zu hoffen, dass auch die Herausgeber deutscher Lexika die Bedeutung dieses großen deutschen Mathematikers und

Logikers entsprechend einschätzen. Während die Encyclopaedia Britannica in ihrer Ausgabe von 1997 Gottlob Frege eine ganze Seite widmet (George Boole wurde ein halbe Seite zuerkannt), wird Frege im Brockhaus[9] mit ganzen 12 Zeilen bedacht. Dafür wird George Boole im deutschen Lexikon ausführlicher (28 Zeilen) behandelt. Möge der Philosoph auch in eigenen Landen wieder zu gebührender Anerkennung gelangen.

Literaturverzeichnis

Albrecht, E./**Asser**, G. (Hrsg.): Wörterbuch der Logik, Leipzig 1983.

Clocksin, W. F./**Mellish**, C. S.: Programming in PROLOG, Berlin u. a. 1994.

Field, A. J./**Harrison**, P. G.: Functional Programming, Addison-Wesley, 1988.

Frege, G.: Begriffsschrift. Eine der arithmetischen nachgebildete Formelsprache des reinen Denkens, Halle 1879.

Frege, G.: Grundlagen der Arithmetik, Stuttgart 1987.

Frege, G.: Logik [1897] in Schriften zur Logik, Akademie-Verlag, Berlin 1973.

Mayer, V.: Gottlob Frege, München 1996.

Autorenangaben

Prof. Dr.-Ing. Uwe Lämmel
Grundlagen der Informatik / Künstliche Intelligenz
Hochschule Wismar, Fachbereich Wirtschaft
Philipp-Müller-Straße
Postfach 12 10
D – 23966 Wismar
Telefon: ++49 / (0)3841 / 753 617
Fax: ++49 / (0)3841 / 753 131
E-mail: u.laemmel@wi.hs-wismar.de
www.wi.hs-wismar.de/~laemmel/

[9] Der Brockhaus in 20 Bänden, Wiesbaden 1967, Bd. 3 bzw. Bd. 6.

WDP - Wismarer Diskussionspapiere / Wismar Discussion Papers

Heft 01/2003	Jost W. Kramer: Fortschrittsfähigkeit gefragt: Haben die Kreditgenossenschaften als Genossenschaften eine Zukunft?
Heft 02/2003	Julia Neumann-Szyszka: Einsatzmöglichkeiten der Balanced Scorecard in mittelständischen (Fertigungs-)Unternehmen
Heft 03/2003	Melanie Pippig: Möglichkeiten und Grenzen der Messung von Kundenzufriedenheit in einem Krankenhaus
Heft 04/2003	Jost W. Kramer: Entwicklung und Perspektiven der produktivgenossenschaftlichen Unternehmensform
Heft 05/2003	Jost W. Kramer: Produktivgenossenschaften als Instrument der Arbeitsmarktpolitik. Anmerkungen zum Berliner Förderungskonzept
Heft 06/2003	Herbert Neunteufel/Gottfried Rössel/Uwe Sassenberg: Das Marketingniveau in der Kunststoffbranche Westmecklenburgs
Heft 07/2003	Uwe Lämmel: Data-Mining mittels künstlicher neuronaler Netze
Heft 08/2003	Harald Mumm: Entwurf und Implementierung einer objektorientierten Programmiersprache für die Paula-Virtuelle-Maschine
Heft 09/2003	Jost W. Kramer: Optimaler Wettbewerb – Überlegungen zur Dimensionierung von Konkurrenz
Heft 10/2003	Jost W. Kramer: The Allocation of Property Rights within Registered Co-operatives in Germany
Heft 11/2003	Dietrich Nöthens/Ulrike Mauritz: IT-Sicherheit an der Hochschule Wismar
Heft 12/2003	Stefan Wissuwa: Data Mining und XML. Modularisierung und Automatisierung von Verarbeitungsschritten
Heft 13/2003	Bodo Wiegand-Hoffmeister: Optimierung der Sozialstaatlichkeit durch Grundrechtsschutz – Analyse neuerer Tendenzen der Rechtsprechung des Bundesverfassungsgerichts zu sozialen Implikationen der Grundrechte -
Heft 14/2003	Todor Nenov Todorov: Wirtschaftswachstum und Effektivität der Industrieunternehmen beim Übergang zu einer Marktwirtschaft in Bulgarien
Heft 15/2003	Robert Schediwy: Wien – Wismar – Weltkulturerbe. Grundlagen, Probleme und Perspektiven
Heft 16/2003	Jost W. Kramer: Trends und Tendenzen der Genossenschaftsentwicklung in Deutschland
Heft 01/2004	Uwe Lämmel: Der moderne Frege