ARNOLD OBERSCHELP

Prinzipien des Aufbaus von Syntax und Semantik formaler Sprachen

Formale Sprachen sind zuerst von Logikern eingeführt worden, um mit ihrer Hilfe logische Begriffe zu explizieren und mathematische Theorien zu analysieren, sie werden jedoch auch in zunehmendem Maße auf die Linguistik angewendet. Um einen Eindruck von formalen Sprachen zu geben und auch damit die nachfolgende Betrachtung nicht in der Luft hängt, sollen in einem ersten Teil einige Beispiele vorgestellt werden. Die Beispiele sind so gewählt, daß sie einerseits möglichst einfach sind, andererseits aber doch wichtige Typen formaler Sprachen repräsentieren. Im zweiten Teil werden einige sehr allgemeine Bemerkungen über formale Sprachen angeschlossen. In den beiden letzten Teilen entwerfen wir – verdeutlicht jeweils an den Beispielen – eine Skizze einer allgemeinen Syntax und Semantik formaler Sprachen, deren Anwendungsbereich weit über die im engeren Sinne logischen Sprachen hinausreicht.

I. Beispiele

Beispiel (1)

Das *Alphabet* von L_1 bestehe aus den Buchstaben A, B. Die *wohlgeformten Ausdrücke* von L_1 seien genau die Wörter der Form:

$$A^n B A^n B A^n ,$$

wobei n eine natürliche Zahl ($\geqslant 0$) sei und A^n die n-fache Verkettung des Symbols A mit sich selbst sei: $A^n =_{Df} \underbrace{A A \ldots A}_{n\text{-mal}}$. Als Grenzfall sei: $A^0 =_{Df} \Lambda$ (leeres Wort).

Beispiele für Ausdrücke:

BB, $ABABA$, $AABAABAA$ u.ä.

Keine Ausdrücke sind $ABABAA$, $AABBAAA$.

Eine *Interpretation* von L_1 wird nicht angegeben.

Beispiel (2)

Das *Alphabet* von L_2 soll aus den Dezimalziffern $0,1,2, \ldots, 8,9$ bestehen. *Ausdrücke* sind 0 und alle nichtleeren Wörter über dem Alphabet, die nicht mit 0 anfangen.

Ausdrücke sind z.B. *0*, *157*, *90000*,
keine Ausdrücke sind: *00*, *00157*, *010*.
Interpretation: Jeder Ausdruck von L_2 ist Name einer natürlichen Zahl.

Beispiel (3)

Das *Alphabet* von L_3 enthalte die lateinischen Großbuchstaben *A*, *B*,*C*, ..., *X*, *Y*, *Z* sowie ein Zwischenraumzeichen␣ . Die *Ausdrücke* mögen aus den intransitiven Verben *SINGT*␣ , *SCHREIT*␣ (genauer gesagt diesen Verbformen), dem Eigennamen *CARUSO*␣ , den Gattungsnamen *TENOR*␣ , *BABY*␣ und den Determinatoren und Artikeln *JEDER*␣ , *EIN*␣ , *KEIN*␣ , *DER*␣ sowie aus Nominalphrasen und Sätzen bestehen. Dabei entsteht eine Nominalphrase, wenn man hinter einen Determinator oder Artikel einen Gattungsnamen schreibt, und ein Satz, wenn man hinter einen Eigennamen oder eine Nominalphrase ein intransitives Verb schreibt. Das Zwischenraumzeichen lassen wir künftig weg und stellen es durch eine Lücke dar.

Sätze dieser formalen Sprache sind z.B.:

DER TENOR SINGT,
KEIN TENOR SCHREIT,
CARUSO SINGT,
JEDER BABY SCHREIT.

Keine Sätze sind:

DER CARUSO SINGT,
BABY SCHREIT.

Diese einfache Sprache enthält insgesamt nur 18 Sätze.

Interpretation: Wenn man von der Holperigkeit absieht, die dadurch entsteht, daß wir nur ein grammatisches Geschlecht berücksichtigt haben, so sind die Sätze von L_3 unmittelbar als Sätze der natürlichen Sprache lesbar. Man kann sie deshalb so verstehen, wie die entsprechenden Sätze der natürlichen Sprache. In Abschnitt IV kommen wir auf Interpretationen von L_3 zurück.

Beispiel (4)

Das *Alphabet* von L_4 bestehe aus den folgenden Symbolen:

$\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists, =$,
$(,)$,
$\leqslant, +,$
$x_0, x_1, x_2, x_3, \ldots$.

Die Zeichen der ersten Zeile sind die *logischen Symbole*, es folgen die *Gliederungsklammern*, die *Konstanten* und die *Variablen*.

Die *Ausdrücke* von L_4 sind in zwei syntaktische Kategorien eingeteilt, nämlich in *Terme* und *Formeln*.

Terme:

(a) Jede Variable ist ein Term.
(b) Wenn a, b Terme sind, so ist auch $(a + b)$ ein Term.

Formeln:

(a) Wenn a, b Terme sind, so sind $a = b$, $a \leqslant b$ Formeln.
(b) Wenn ϕ, ψ Formeln sind und v Variable ist, so sind

$$\neg\phi,\ (\phi \wedge \psi),\ (\phi \vee \psi),\ (\phi \rightarrow \psi),\ (\phi \leftrightarrow \psi),$$
$$\forall v\ \phi,\ \exists v\ \phi \quad \text{Formeln.}$$

Ein Vorkommen einer Variablen v in einem Ausdruck ist *frei* genau dann, wenn es nicht innerhalb einer Teilformel der Art $\forall v\ \phi$ bzw. $\exists v\ \phi$ ist, andernfalls ist das Variablenvorkommen *gebunden*.

Beispiele für Formeln:

ϕ_1 sei $\forall x_0\ \forall x_1\ (x_0 \leqslant x_1 \rightarrow \exists x_2 (x_0 + x_2) = x_1)$,

ϕ_2 sei $\forall x_0\ \forall x_1\ (\exists x_2 (x_0 + x_2) = x_1 \rightarrow x_0 \leqslant x_1)$.

In beiden Formeln kommen keine Variablen frei vor, solche Formeln heißen *Aussagen*.

Interpretationen: Eine Interpretation von L_4 wird gegeben durch eine *L_4-Struktur*, d.h. eine nichtleere Menge als *Individuenbereich*, sowie eine *zweistellige Relation* darauf als Interpretation von $\leqslant$ und eine *zweistellige Verknüpfung* darauf als Interpretation von +. Für die Interpretation von Ausdrücken mit freien Variablen muß noch eine *Belegung* dieser Variablen mit Individuen als Werten gegeben sein. Eine solche Struktur und Belegung (falls erforderlich) induziert dann eine *Bewertung* der Ausdrücke, wobei Terme durch Individuen und Formeln durch Wahrheitswerte W, F bewertet werden. Die Definition der Bewertungen wird hier nicht gegeben, sie greift auf ein metatheoretisches Verständnis der logischen Symbole zurück (siehe Abschnitt IV).

Beispiele für Interpretationen:

(a) Der Individuenbereich sei $\mathbb{N}$ (Menge der natürlichen Zahlen), und $\leqslant$, + mögen die übliche arithmetische Bedeutung auf $\mathbb{N}$ haben. Bei dieser Interpretation werden ϕ_1, ϕ_2 wahr.

(b) Der Individuenbereich sei $\mathbb{R}$ (Menge der reellen Zahlen) und $\leqslant$, + mögen die übliche arithmetische Bedeutung auf $\mathbb{R}$ haben. Bei dieser Interpretation ist ϕ_1 wahr und ϕ_2 falsch.

(c) Der Individuenbereich bestehe aus den höheren Beamten einer gewissen Behörde. $\leqslant$ werde durch die Relation zwischen Beamten, jünger oder gleich alt zu sein, interpretiert. + werde durch die Funktion interpretiert, die je zwei Beamten den rangniedrigsten gemeinsamen Vorgesetzten zuordne (wobei, damit die Funktion stets definiert ist, jeder als Vorgesetzter von sich selbst aufgefaßt sei). In dieser Interpretation ist ϕ_1 i. allg. falsch. ϕ_2 ist wahr, wenn in der Behörde das Anciennitätsprinzip gewahrt ist.

Beispiel (5)

Das *Alphabet* von L_5 bestehe aus den folgenden Symbolen:

A, B, C, D, ..., X, Y, Z,	(Buchstaben)
0, 1, 2, ..., 8, 9,	(Ziffern)
; , ␣ .	(Semicolon und Leerzeichen).

Das Leerzeichen stellen wir wieder durch eine Lücke dar.

Die *Ausdrücke* sind in vier Kategorien eingeteilt, nämlich in *Variablen, Labels, Anweisungen* und *Programme.*

Variablen: Eine Variable ist ein Wort ohne Semikolon, das mit einem Buchstaben anfängt.

Labels: Ein Label ist dasselbe wie eine Variable.

Anweisungen: Wenn *v* Variable und *l* Label ist, so sind *READ v, PRINT v, LBL l, GOTO l, TEST v, ICR v, DCR v* Anweisungen.

Programme: Wenn *a* Anweisung und *p* Programm ist, so sind

a; , *pa;*

Programme.

Beispiel für ein Programm:

READ X1;READ X2;
LBL A;TEST X1;GOTO D;DCR X1;
LBL B;TEST X2;GOTO C;DCR X2;ICR Y;ICR Z;GOTO B;
LBL C;TEST Z;GOTO A;DCR Z;ICR X2;GOTO C;
LBL D;PRINT Y;

Interpretation: Die Variablen eines Programms bedeuten Register einer Maschine, die jeweils eine natürliche Zahl enthalten können und die vor Beginn einer Rechnung auf Null gestellt sind. Jede Anweisung bedeutet

einen Rechenschritt, und ein Programm bewirkt eine Rechnung der Maschine, bei der die Anweisungen in der aufgeschriebenen Reihenfolge durchgeführt werden, es sei denn, eine Sprunganweisung oder Verzweigungsanweisung legt etwas anderes fest.

Bei *READ v* wird eine extern bereitgestellte Zahl in das Register *v* eingelesen, bei *PRINT v* wird die Zahl im Register *v* nach außen abgegeben. *LBL l* ist ein Leerschritt, der nur eine Programmstelle für Sprunganweisungen markiert. Bei einer Sprunganweisung *GOTO l* wird als nächster Schritt der bei *LBL l* stehende Rechenschritt ausgeführt. Falls das Label nicht definiert ist, also mehrere Anweisungen *LBL l* oder gar keine solche Anweisung vorkommt, stoppt die Maschine, ebenso, wenn sie die letzte Anweisung des Programmes durchgeführt hat und diese keinen Rücksprung bewirkt hat. *TEST v* liefert eine Verzweigung: Wenn Register *v* die Zahl *0* enthält, wird die nächste, sonst die übernächste Anweisung ausgeführt. Die beiden letzten Anweisungen entsprechen den arithmetischen Operationen, die die Maschine direkt beherrscht: Bei *ICR v* wird die Zahl im Register *v* um *1* erhöht (Inkrementierung), bei *DCR v* wird sie um *1* erniedrigt, sofern sie noch nicht *0* ist und sonst bei *0* belassen (Dekrementierung).

Das angegebene Programm bewirkt, daß zwei Zahlen eingegeben und ihr Produkt ausgegeben wird, dann erfolgt Stop.

II. Allgemeine Bemerkungen

Diese Beispiele sollen genügen. Sie repräsentieren einige Haupttypen formaler Sprachen und können zur Illustration vieler allgemeiner Eigenschaften formaler Sprachen dienen.

Es sollen nun allgemeine Feststellungen über formale Sprachen folgen.

(A) Formale Sprachen stehen im Gegensatz zu natürlichen Sprachen, sie sind *künstlich*. Sie müssen in allen Einzelheiten geplant und explizit eingeführt werden. Das betrifft auch Interpretationen solcher Sprachen.

(B) Aus der Art ihrer Entstehung ergibt sich die Notwendigkeit einer *Metasprache*, in der man die formale Sprache, dann auch als *Objektsprache* bezeichnet, einführt und untersucht. Die Metasprache muß bereits inhaltlich verstanden werden. Als Metasprache wird oft eine natürliche Sprache genommen.

(C) Die explizite Einführung der formalen Sprachen ermöglicht es, daß diese Sprachen *präzise* und nicht vage sind. Es liegt z.B. eindeutig fest, ob eine Zeichenreihe wohlgeformt ist.

(D) Formale Sprachen werden *schriftlich* notiert. Während bei natürlichen Sprachen die mündliche Form oft als die eigentliche Sprache gilt und die schriftliche Form nur als (mehr oder weniger unvollkommene) Simulation angesehen wird, ist es bei formalen Sprachen umgekehrt. Die Realisierung der Ausdrücke als Zeichenreihen ist die einzige relevante Realisierung. Sprechweisen sind sekundär und dienen nur der besseren Verständigung über eine formale Sprache in der mündlichen Metasprache.

(E) Formale Sprachen sind häufig sehr *einfach*. Die Syntax vieler Sprachen kann man in wenigen Minuten lernen. Zwar gibt es auch Programmiersprachen, deren Syntax viele Seiten umfaßt, aber sie reichen nicht an die Komplexität natürlicher Sprachen heran.

(F) Entsprechend ihrer geringen Komplexität sind formale Sprachen *nicht universell*, sondern jeweils nur *für spezielle Zwecke* entworfen und brauchbar.
L_1 ist eine Art Spielzeugsprache für die Werkstatt eines formalen Syntaktikers. Allerdings ist L_1 so trivial auch nicht, wie es aussieht: L_1 ist nicht kontextfrei.
Die Sprache L_2 ist ein Bezeichnungssystem für natürliche Zahlen. L_2 steht hier als Repräsentant für *Nomenklaturen* wie sie z.B. auch für chemische Verbindungen, für Elementarteilchen, für die biologische Taxonomie üblich sind.
L_3 ist eine formale Sprache, die ein *Fragment* der natürlichen Sprache simulieren soll. L_3 möge auch selbst als Fragment bezeichnet werden. Derartige Fragmente (meist wesentlich umfangreicher) sind in großer Zahl aufgestellt worden.
L_4 ist eine prädikatenlogische Sprache aus der großen Familie der *logischen Sprachen*, die zur Explizierung logischer Begriffe dienen.
L_5 ist eine einfache *Programmiersprache* und dient der Kommunikation mit einem Computer.

(G) Der Zweck formaler Sprachen ist gewöhnlich *nicht* die Simulation der Oberflächenform natürlicher Sprachen. Deshalb ist es nicht als Nachteil der Sprachen anzusehen, wenn sie sich äußerlich stark von natürlichen Sprachen unterscheiden. Aber natürlich ist die Fremdartigkeit auch keine besondere Tugend. Auch formale Sprachen werden von Menschen benutzt. Deshalb sollte man Bezeichnungen verwenden, die die Benutzung durch den Menschen unterstützen und den Zweck erkennen lassen. Dem dient z.B., daß man in L_5 Variablen und Labels unterscheidet, obwohl es sich um dieselben Zeichenreihen handelt, und daß man einen *mnemotechni-*

schen Code verwendet, also *"PRINT"* einen Druckbefehl, *"GOTO"* einen Sprungbefehl bezeichnet usw. Auch die Verwendung von $\forall, \exists$ für Allquantor und Existenzquantor ist eine auf menschliche Benutzer zugeschnittene Bezeichnung. Gute Bezeichnungen sind für den Erfolg formaler Sprachen sicherlich wichtig. Für eine theoretische Betrachtung formaler Sprachen wird dieser metapragmatische Aspekt allerdings völlig ausgespart. Doch muß man einräumen, daß suggestive Bezeichnungen auch manchmal als Ersatz für eine explizit formulierte Semantik dienen.

Bei Fragmenten vom Typ des Beispiels (3) geht es natürlich um die Simulation natürlicher Sprachen. Dabei ist aber unbedingte grammatische Korrektheit nicht das einzige Adäquatheitskriterium. Es mag andere Zwecke geben, die höher rangieren (etwa daß das Fragment nicht zu kompliziert werden soll).

(H) Formale Sprachen sind rein syntaktisch definiert. Man könnte sagen, daß die *Syntax* unabhängig von der Semantik ist und daß sie systematisch *Priorität vor der Semantik* hat. Damit meinen wir, daß die Semantik keine Rolle spielt bei der Festlegung, was die wohlgeformten Ausdrücke sind.

Natürlich wird man gewöhnlich bereits von einer intuitiven Semantik bei der Aufstellung einer formalen Sprache geleitet, z.B. so, daß man einen entsprechenden mnemotechnischen Code wählt (siehe G). Aber die syntaktischen Definitionen können ohne theoretischen Rückgriff auf die Semantik gegeben werden. Die Semantik muß dagegen die vorliegende (syntaktische) formale Sprache voraussetzen, vielfach macht sie dann die intuitiven semantischen Vorstellungen explizit.

(I) Syntax und Semantik sind beide unabhängig von der *Pragmatik*, die nur in rudimentärer Weise behandelt wird. Viele Aspekte der tatsächlichen Verwendung formaler Sprachen spielen theoretisch gar keine Rolle. So ist z.B. die Aussage

$$(\ldots((\phi_1 \wedge \phi_1) \wedge \phi_1) \ \ldots \wedge \phi_1)$$

wobei etwa ϕ_1 10^{10}mal miteinander konjunktiv verknüpft sei), ebensogut eine wohlgeformte Aussage, wie ϕ_1 alleine und auch in gleicher Weise wahr oder falsch in Interpretationen wie ϕ_1 selbst. Aber natürlich ist diese Aussage weniger interessant und brauchbar, als ϕ_1. Auch könnte man die Variablen *X1, X2, Y, Z* des letzten Beispiels ebensogut mit *ABBCX5X, ABCX5CX, BLABLA, CCC* bezeichnen und die Labels *A, B, C* durch *HIHI, HAHA, BUMBUM* ersetzen. Für die Maschine, die nach dem Programm rechnen soll,

wäre das völlig nebensächlich. Für den Anwender der Sprache wäre es unpraktisch oder unseriös.

III. Syntax

Wir wollen jetzt genauer auf die Syntax formaler Sprachen eingehen.

Alphabete

Zu jeder formalen Sprache gehört ein *Alphabet.* Das ist eine nichtleere Menge von Objekten, die *Symbole* oder *Zeichen* heißen. Die Symbole können graphische Muster sein wie Buchstaben, Ziffern, Interpunktionszeichen, mathematische Zeichen u.ä. Dabei muß man ein Symbol von einem *Vorkommen* des Symbols in einer Zeichenreihe und dieses wieder von einer konkreten Manifestation des Symbols an einer Stelle des materiellen Trägers der Sprache, einen sog. *Token* unterscheiden. So gibt es nur einen Großbuchstaben *A*, der in dem Wort *PAPA* zweimal vorkommt und von dem in diesem Satz drei Token zu sehen sind.

Man nimmt oft an, daß das Alphabet *endlich* ist und die Symbole geordnet sind (*alphabetische Ordnung*). Doch läßt man auch unendliche Alphabete zu (siehe L_4). Ein unendliches Alphabet läßt sich oft durch ein endliches ersetzen, wenn man Symbole des unendlichen Alphabetes durch Wörter über dem endlichen Alphabet ersetzt. So könnte man in L_4 die Variablen x_0, x_1, x_2, ... aus dem Alphabet entfernen und stattdessen die Symbole *x*, *0*, *1*, *2*, ..., *8*, *9* aufnehmen. Die Variablen ließen sich dann durch Wörter der Art *xW* ersetzen, wobei *W* ein Ausdruck von L_2 ist. Die Reduktion auf endliche Alphabete ist für uns nicht von Belang, wir wollen deshalb auch unendliche Alphabete zulassen. Ferner brauchen die Symbole auch keine graphischen Muster zu sein, es dürfen beliebige Objekte sein, z.B. auch Zahlen, Mengen u.ä.

Die Symbole werden oft unterteilt in *logische Symbole, Konstanten, Variablen, Hilfszeichen, Terminalzeichen* u.ä. Doch müssen solche Festsetzungen für jede Sprache besonders getroffen werden. Sie sind oft rein konventioneller Art. Z.B. ist die Grenze zwischen logischen Symbolen und Hilfssymbolen fließend. Variablen und Konstanten tauchen oft nicht bei den Symbolen, sondern bei den Zeichenreihen auf. Allgemein läßt sich nur sagen, daß Symbole das Material sind, aus dem Zeichenreihen aufgebaut sind.

Zeichenreihen, Wörter

Es sei jetzt ein Alphabet A gegeben.

Durch *Verketten von Symbolen* aus A kann man *Zeichenreihen* oder *Wörter* über A bilden. Die Zeichenreihe, die durch Verkettung der Symbole $\sigma_0, \sigma_1, \ldots, \sigma_n$ (in dieser Reihenfolge) entsteht, werde durch $\sigma_0\,\sigma_1 \ldots \sigma_n$ angegeben. Wir sagen, daß darin σ_i an der Stelle *i vorkommt* und daß das Wort die *Länge n+1* hat. So kann man aus den Symbolen *P, A, P, A* das Wort *PAPA* der Länge 4 bilden, in dem *P* an den Stellen 0 und 2 und *A* an den Stellen 1 und 3 vorkommt.

Ein besonderes Wort ist das *leere Wort*, das wir mit Λ bezeichnen und das als einziges die Länge 0 hat.

Die *Menge aller Wörter* über A sei

$$A^* .$$

Die Verkettung der Symbole induziert eine *Verkettung der Wörter:* Wenn W_1 das Wort $\sigma_0 \ldots \sigma_n$ und W_2 das Wort $\tau_0 \ldots \tau_m$ ist, so sei $W_1\,W_2$ das Wort $\sigma_0 \ldots \sigma_n\ \tau_0 \ldots \tau_m$. Die Verkettung ist assoziativ, deshalb schreiben wir keine Klammern. Verkettung mit dem leeren Wort bringt keine Änderung, deshalb kann Λ nach Belieben zugefügt oder weggelassen werden.

Die wichtigste Eigenschaft der Verkettungsoperation ist es, daß die durch sie gewonnenen Wörter der Bedingung der *eindeutigen Lesbarkeit* genügen, d.h. Wörter $\sigma_0 \ldots \sigma_n$ und $\tau_0 \ldots \tau_m$ sind genau dann gleich, wenn $n = m$ und $\sigma_i = \tau_i$ für $i = o,\ldots, n$ ist.

Wenn die Symbole graphische Muster sind, so gibt man die Verkettung durch Hintereinanderschreiben von Token an. Dabei ist allerdings Vorsicht geboten, damit die eindeutige Lesbarkeit gewahrt bleibt. Es seien z.B. in einem Alphabet drei Symbole $\sigma_0, \sigma_1, \sigma_2$ vorhanden, die folgendermaßen aussehen: $*\,o, *, o$. Dann entsteht durch Hintereinanderschreiben von σ_0, σ_1 und von $\sigma_1, \sigma_2, \sigma_1$ beidemale $*o*$ obwohl $\sigma_0\,\sigma_1$ und $\sigma_1\sigma_2\sigma_1$ als verschiedene Zeichenreihen aufzufassen sind. Eindeutige Lesbarkeit ist genau dann gegeben, wenn die Zeichenreihen umkehrbar eindeutig den endlichen Folgen von Symbolen im mathematisch-mengentheoretischen Sinne entsprechen.

Die Ausdrücke formaler Sprachen sollen Zeichenreihen über einem Alphabet sein. Man beachte, daß darin bereits eine Normierung enthalten ist, nämlich in der *streng linearen Aufeinanderfolge* der Symbole. In der mathematischen Formelschreibweise kommen auch hoch und tief gestellte Indizes, Exponenten, Quotientenschreibweisen $\frac{a}{b}$ u.ä. nicht lineare

Schreibweisen vor. Bei einer Darstellung durch Zeichenreihen muß das alles linearisiert werden, z.B. $a /. b$ statt $\frac{a}{b}$, $a**b$ statt a^b usw.

Syntaktische Kategorien

Eine formale Sprache L wird dadurch gegeben, daß gewisse Wörter über einem Alphabet A ausgezeichnet werden als *wohlgeformte Zeichenreihen* oder auch *Ausdrücke*. Die Definition der Ausdrücke geschieht durch eine *Grammatik*. Wir wollen eine Grammatik in dem Stil angeben, wie er in der Logik verbreitet ist und auch in den Beispielen verwendet wurde. Wir reden von einer *rekursiven Definition* der Ausdrücke. Dabei sind gewisse Ausdrücke explizit zu geben durch die *Rekursionsanfänge*. Ferner sind Regeln zu geben, wie man aus bereits erhaltenen Ausdrücken weitere Ausdrücke gewinnt, diese Regeln sind die *Rekursionsvorschriften*. Es gibt durchaus noch andere Darstellungsformen einer Grammatik, etwa die Backus-Naur-Form und die Form einer Phrasenstrukturgrammatik. Gewöhnlich gibt es mehrere Arten von Ausdrücken. Als erstes Bestimmungsstück wählen wir deshalb eine Menge

$$CAT\ ,$$

die wir die Menge der *syntaktischen Kategorien* nennen. Wir wollen die Beispiele betrachten.

(1) Wir nehmen zwei Kategorien: S (Sätze), A (A-Wörter) und setzen also

$$CAT_1 =_{Df} \{S,A\}.$$

(2) Wir nehmen zwei Kategorien ZW (Zahlwörter), ZF (Ziffernfolgen) und setzen also

$$CAT_2 =_{Df} \{ZW, ZF\}.$$

(3) Wir nehmen sechs Kategorien, nämlich V (intransitive Verben), E (Eigennamen), N (Nomina, Gattungsnamen), Det (Determinatoren und Artikel), NP (Nominalphrasen), S (Sätze) und setzen also

$$CAT_3 =_{Df} \{V, E, N, Det, NP, S\}.$$

(4) Wir nehmen fünf Kategorien: 0 (Formeln), 1 (Terme), 2 (Variablen), 3 (zweistellige Funktionszeichen), 4 (zweistellige Relationszeichen) und setzen also:

$$CAT_4 =_{Df} \{0,1,2,3,4\}.$$

(5) Wir wählen sechs Kategorien, nämlich B (Buchstaben), Z (Ziffern), V (Variablen), L (Labels), A (Anweisungen), P (Programme).

Es sei somit:

$$CAT_5 =_{Df} \{B, Z, V, L, A, P\}.$$

Man sieht, daß wir mehr Kategorien eingeführt haben, als wir in Abschnitt I vorgestellt haben. Bei genauer Ausführung der Syntax kann es vorkommen, daß die ursprünglich ins Auge gefaßten Kategorien besser um Hilfskategorien erweitert oder in Unterkategorien zerlegt werden.

Man beachte, daß eine Kategorie nicht dasselbe ist wie die Menge der Ausdrücke dieser Kategorie. Kategorien sind nur Indizes, durch die man gewisse Mengen von Wörtern kennzeichnet.

Ausdrücke

Der wesentliche Teil der Grammatik besteht darin, für jede Kategorie t eine Menge E_t der *Ausdrücke der Kategorie t* zu definieren.

Die Definition soll rekursiv sein.
Als *Rekursionsanfänge* ist zu jeder Kategorie t die Menge B_t der *Basisausdrücke der Kategorie t* explizit anzugeben.

Das wollen wir für unsere Beispiele tun.

(1) *Es sei* $B_S = \emptyset$, $B_A = \{\Lambda\}$.

Es gibt also keine Basisausdrücke der Kategorie S, das leere Wort ist der einzige Basisausdruck der Kategorie A.

(2) *Es sei* $B_{ZW} = \{0\}$, $B_{ZF} = \{\Lambda\}$.

Basiszahlwort ist das Wort, das nur aus der Ziffer *0* besteht, Basisziffernfolge ist das leere Wort.

(3) *Es sei* $B_V = \{SINGT\text{␣}, SCHREIT\text{␣}\}$, $B_E = \{CARUSO\text{␣}\}$,
$B_N = \{TENOR\text{␣}, BABY\text{␣}\}$, $B_{Det} = \{JEDER\text{␣}, EIN\text{␣}, KEIN\text{␣}, DER\text{␣}\}$,
$B_{NP} = B_S = \emptyset$.

Zu den Kategorien *NP* und *S* gibt es also keine Basisausdrücke.

(4) *Es sei* $B_0 = \emptyset$, $B_1 = B_2 = \{x_i \mid i \in \mathbb{N}\}$, $B_3 = \{+\}$, $B_4 = \{\leqslant\}$.

Es gibt also keine Basisausdrücke der Kategorie *0*.
Die Variablen bilden die Basisausdrücke der Kategorien *1* und *2*.
B_3 enthält nur das +-Zeichen, B_4 nur das $\leqslant$-Zeichen.

(5) *Es sei* $B_B = \{A, B, C, \ldots, X, Y, Z\}$
$B_Z = \{0, 1, 2, \ldots, 8, 9\}$
$B_V = B_L = B_A = B_P = \emptyset$.

Die Basisausdrücke der Kategorie B sind also genau die Buchstaben, die der Kategorie Z die Ziffern, die anderen Basiskategorien sind leer.

Nach der Betrachtung der Beispiele kehren wir jetzt wieder zur allgemeinen Theorie zurück.

Man verlangt zunächst natürlich, daß die Basisausdrücke insbesondere Ausdrücke sind. Man fordert für $t \in CAT$:

(*) $B_t \subseteq E_t$.

Ferner gibt man *Rekursionsvorschriften* an, um aus bereits gewonnenen Ausdrücken weitere Ausdrücke zu erhalten.

Dazu hat man gewisse *syntaktische Verknüpfungen* auf Zeichenreihen: $F : A^* \times ... \times A^* \longrightarrow A^*$ und *syntaktische Regeln*, die besagen, daß F angewendet auf $z_1,...,z_n$ mit $z_i \in E_{t_i}$ $(i = 1,...,n)$ einen Ausdruck aus E_t ergibt. Wir schreiben dann auch:

(**) *Wenn* $z_1 \in E_{t_1},...,z_n \in E_{t_n}$, so $F(z_1,...,z_n) \in E_t$.

Die Mengen E_t der Ausdrücke der Kategorie t (für $t \in CAT$) sind dann definiert als die eindeutig bestimmten kleinsten Mengen mit (*) (die also die Basisausdrücke enthalten) und mit (**) (die unter den syntaktischen Verknüpfungen gemäß den syntaktischen Regeln abgeschlossen sind). Dabei müssen die syntaktischen Verknüpfungen und Regeln so gewählt sein, daß die Ausdrücke eindeutig lesbar – besser analysierbar – sind. Bei syntaktisch ambigen Sprachen wird auf diese letzte Forderung verzichtet.

Gewöhnlich führt man für die Mengen E_t besondere Bezeichnungen ein und benutzt auch besondere syntaktische Variablensorten.

Wir wollen unsere Beispiele betrachten.

(1) *Wenn* $z \in E_A$, *so* $zA \in E_A$.
Wenn $z \in E_A$, *so* $zBzBz \in E_S$.

Die syntaktische Verknüpfung der ersten Regel ist das Anhängen des Buchstabens A, die syntaktische Verknüpfung der zweiten Regel fügt den Buchstaben B zwischen drei Exemplare des Argumentes z ein.

Die erste Regel ist in dem Sinne *echt rekursiv*, als das Ergebnis einer Anwendung der Regel wieder als Ausgangspunkt für eine neue Anwendung der Regel genommen werden kann. Daraus ergibt sich, daß die Sprache L_1 unendlich viele Ausdrücke hat.

(2) *Wenn* $z \in E_{ZF}$, *so* $z0, z1, z2,...,z8, z9 \in E_{ZF}$.
Wenn $z \in E_{ZF}$, *so* $1z, 2z,...,8z, 9z \in E_{ZW}$.

(3) *Wenn* $x \in E_{Det}$, $y \in E_N$, *so* $xy \in E_{NP}$.
Wenn $x \in E_E$, $y \in E_V$, *so* $xy \in E_S$.
Wenn $x \in E_{NP}$, $y \in E_V$, *so* $xy \in E_S$.

Man beachte, daß diese Regeln nicht in dem echten Sinne rekursiv sind, wie es oben erläutert wurde. Deshalb hat die Sprache L_3 nur endlich viele Ausdrücke. Zu allen Regeln gehört übrigens dieselbe syntaktische Verknüpfung, nämlich das Verketten der beiden Argumente x, y.

(4) Wir schreiben FML für E_0, synt. Variablen: ϕ, ψ,
TM für E_1, synt. Variablen: a, b und
VAR für E_2, synt. Variablen: v, w.

Die Regeln lauten:

Wenn $f \in E_3$ *und* $a, b \in TM$, *so* $(afb) \in TM$.
Wenn $a, b \in TM$, $\phi, \psi \in FML$, $v \in VAR$, $Q \in E_4$, *so*
$a = b$, aQb, $\neg\phi$, $(\phi \wedge \psi)$, $(\phi \vee \psi)$,
$(\phi \rightarrow \psi)$, $(\phi \leftrightarrow \psi)$, $\forall v\, \phi$, $\exists v\, \phi \in FML$.

Zu den Regeln muß eigentlich noch ein Zusatz gemacht werden, nämlich daß in $\forall v\, \phi$, $\exists v\, \phi$ die Variable v gebunden ist (s. u.).

Man beachte ferner, daß die logischen Zeichen nicht zu den Ausdrücken irgendeiner Kategorie gehören, sie kommen durch die Anwendung der syntaktischen Verknüpfungen herein.

(5) Wir schreiben BU für E_B, ZI für E_Z,
VAR für E_V, LBL für E_L, ANW für E_A,
PRG für E_P.

Die Regeln lauten:

Wenn $x \in BU$, $y \in ZI$, $v \in VAR$, *so* x, vx, vy, $v_{\sqcup} \in VAR$.
Wenn $x \in VAR$, *so* $x \in LBL$.
Wenn $v \in VAR$, $l \in LBL$, *so*
READ v, *PRINT* v, *LBL* l, *GOTO* l, *TEST* v,
ICR v, *DCR* $v \in ANW$.
Wenn $a \in ANW$, $p \in PRG$ *so* $a;$, $pa; \in PRG$.

Hierdurch ist wohl exemplarisch klar geworden, wie man die Ausdrücke formaler Sprachen festlegen kann.

Es muß nun noch die Rolle der Variablen erläutert werden.

Variablenbindung

Die Basisausdrücke in *logischen Sprachen* sind oft in *Variablen* und *Konstanten* unterteilt. Dabei ist im Augenblick unerheblich, daß diese oft als Zeichenreihen der Länge *1* aufgefaßt und mit entsprechenden Symbolen identifiziert werden. Die Variablen spielen eine Sonderrolle. Einerseits sind es auch Ausdrücke bestimmter Kategorien und können entsprechend beim Aufbau der Ausdrücke fungieren. Andererseits spielen die Variablen eine Sonderrolle als *gebundene Variablen.*

Eine syntaktische Regel kann auch fordern, daß die zugehörige syntaktische Verknüpfung an gewissen Stellen Variablen als Argument nimmt und daß einer jeden solchen Stelle gewisse der anderen Stellen als *Skopus* zugeordnet sind. Ein Variablenvorkommen in einer Stelle der zuerst genannten Art ist ein Vorkommen als *Operatorvariable* und dieses Vorkommen und jedes Vorkommen dieser Variablen in einem Ausdruck im Skopus, das noch nicht anderweitig gebunden ist, ist bezüglich des Operatorvariablenvorkommen *gebunden.* Ein Variablenvorkommen, das nicht gebunden ist, ist *frei.* In unseren Beispielen tritt Variablenbindung nur in L_4 beim Aufbau mit Hilfe der Quantoren $\forall$, $\exists$ auf: Das Vorkommen von v in $\forall v\, \phi$ bzw. $\exists v\, \phi$ unmittelbar hinter $\forall$ bzw. $\exists$ ist ein Vorkommen als Operatorvariable, die Formel ϕ steht im zugehörigen Skopus.

Eine syntaktische Verknüpfung zusammen mit einer syntaktischen Regel möge ein *Funktor* heißen, wenn dazu keine Skopusrelation gehört und somit beim zugehörigen Aufbau der Ausdrücke keine Variablenbindungen entstehen. Im anderen Fall reden wir von einem *Operator.*

Operatoren und gebundene Variablen sind für viele logische Sprachen charakteristisch. Sie treten auch in Programmiersprachen auf, allerdings noch nicht in der einfachen Sprache L_5, die keine Prozeduren kennt. Dagegen enthält das Fragment L_3 keine gebundenen Variablen (sogar überhaupt keine Variablen), und genauso ist es bei vielen anderen Fragmenten.

IV. Semantik

Wir wollen jetzt besprechen, wie man formale Sprachen interpretieren kann. Natürlich gibt es formale Sprachen, für die man gar keine Interpretationen sucht (siehe Beispiel (1)). Im allgemeinen aber wird der Zweck verfehlt, den man mit einer formalen Sprache verfolgt, wenn man keine Interpretationen angibt. Gleichwohl fehlt oft eine explizite Semantik, und man begnügt sich statt dessen mit dem intuitiven Verständnis der Ausdrücke, das durch umgangssprachliche Lesarten oder mnemotechnische

Codes suggeriert wird. In diesem Zustand befand sich z.B. lange Zeit die Modallogik, in der man formale Sprachen mit den Funktoren $\Box$, $\Diamond$ hatte mit den zugehörigen Lesarten "es ist notwendig, daß", "es ist möglich, daß" ohne eine darüber hinausgehende Semantik. Es bleibe dahingestellt, ob die jetzt übliche Kripke-Semantik für die Modallogik adäquat ist, aber jedenfalls ist es eine explizite semantische Theorie, mit der alternative Theorien konkurrieren müssen.

Die Semantik geht gewöhnlich so vor, daß den Ausdrücken einer formalen Sprache, oder jedenfalls gewissen Ausdrücken, *Denotate* zugeordnet werden. Eine solche Zuordnung nennen wir eine *Bewertung* oder *Interpretation* der Sprache.

Wir wollen statt des Wortes Denotat nicht das Wort *Bedeutung* verwenden. Dieses wird nämlich meist in einer sehr umfassenden Weise verstanden, so daß in die Bedeutung eines Ausdrucks alle akzeptablen *Verwendungsweisen* des Ausdrucks eingehen. Denotate dagegen sind geeignete *Objekte*, die in einer passenden semantischen Metatheorie näher bestimmt werden, wobei man sich durch das intuitive Bedeutungskonzept bei der Auswahl der Denotate leiten läßt. Von der Metatheorie nehmen wir hier an, daß sie mengentheoretische Operationen gestattet. Es sei aber erwähnt, daß dieses in der Logik nicht unumstritten ist. Die konstruktive Logik und Mathematik vermeidet mengentheoretische Existenzannahmen. In der operativen Logik z.B. werden die Bedeutungen von Aussagen direkt (d.h. ohne Zwischenschaltung von Denotaten) durch ihre Verwendung in idealisierten Dialogen bestimmt.

Es soll jetzt ein Semantikkonzept skizziert werden, das zu dem im vorigen Abschnitt beschriebenen Syntaxkonzept paßt. Es sei also L eine formale Sprache mit den syntaktischen Kategorien CAT, wobei zu jeder Kategorie $t \in CAT$ eine Menge B_t von Basisausdrücken der Kategorie t und eine umfassende Menge E_t von Ausdrücken der Kategorie t gehört.

Für eine Interpretation von L ist dann zunächst einmal zu jeder Kategorie $t \in CAT$ eine Menge M_t der *möglichen Denotate* für Ausdrücke aus E_t festzulegen.

Wir wollen die Zuordnung, die jeder Kategorie t die Menge M_t zuordnet, als einen *semantischen Rahmen* bezeichnen. Jede Interpretation $\mathbf{J}$ über einen solchen Rahmen ordnet jedem Basisausdruck $\alpha \in B_t$ direkt ein Element

$$\alpha^{\mathbf{J}} \in M_t \quad \text{zu.}$$

Die weitere Bewertung der Ausdrücke erfolgt parallel zu ihrem syntaktischen Aufbau:

Es sei etwa *Fk* ein Funktor, der aus Ausdrücken der Kategorien $t_1,...,t_n$ Ausdrücke der Kategorie t aufbaut. Dann gehört dazu eine *semantische Verknüpfung* Fk^J von M_{t_1} x...x M_{t_n} in M_t. Und wenn $\alpha_i \in M_{t_i}$ und α_i^J bereits definiert sind *(i = 1,...,n)* und der Funktor *Fk* aus $\alpha_1,...,\alpha_n$ den Ausdruck α aufbaut, so sei $\alpha^J = Fk^J(\alpha_1^J,...,\alpha_n^J)$.

Es sei *Op* ein Operator, der aus Variablen der Kategorien $\tau_1,...,\tau_m$ und Ausdrücken der Kategorien $t_1,...,t_n$ Ausdrücke der Kategorie t aufbaue. Dann gehört dazu eine semantische Verknüpfung Op^J, die jeweils n Funktionen einer gewissen Art ein Element von M_t zuordnet. Diese Funktionen sind dabei *mögliche Wertverläufe*, die hier nicht genauer beschrieben werden sollen. Wenn α mit Hilfe von *Op* aus Variablen $v_1,...,v_m$ und Ausdrücken $\alpha_1,...,\alpha_n$ aufgebaut ist, so bestimmen die Werte von $\alpha_1,...,\alpha_n$ bei den Bewertungen, die aus J durch Abändern der Werte der zugehörigen Operatorvariablen entstehen, derartige Wertverläufe; und Op^J ergibt darauf angewendet α^J.

Für manche Zwecke werden die zugelassenen Interpretationen noch eingeschränkt, etwa durch Forderungen der Art, daß α^J für gewisse $\alpha \in B_t$ vorgeschriebene Werte hat, oder daß gewisse Fk^J, Op^J ein für allemal festgelegt werden. Auf diese Weise kann man erreichen, daß die zugelassenen Interpretationen etwa logische Begriffe respektieren oder in anderer Hinsicht standardmäßig sind.

Wir wollen jetzt zu den Beispielen zurückkehren und betrachten zunächst logische Sprachen. Als mögliche Denotate für Formeln benutzt man gewöhnlich die klassischen Wahrheitswerte W (Wahr) und F (Falsch), als mögliche Denotate für Individuenausdrücke (wie Individuenvariablen, Individuenkonstanten, Kennzeichnungen) benutzt man Individuen, d.h. Elemente eines vorgegebenen Individuenbereiches. In der intensionalen Logik treten auch komplexe Objekte als Denotate auf wie Propositionen, Individuenkonzepte u.ä., die man mit bestimmten Funktionen gleichsetzt, die mögliche Welten, Kontexte, Indizes o.ä. auf einfache Objekte (wie Wahrheitswerte oder Individuen) abbilden. Wir gehen jetzt in die Beispielsprache L_4. Dort haben wir fünf syntaktische Kategorien: *0* (Formeln), *1* (Terme), *2* (Variablen), *3* (zweistellige Funktionszeichen), *4* (zweistellige Relationszeichen). Die Mengen der möglichen Werte seien:

$M_0 = Df \{W,F\}$, M_1 sei eine nichtleere Menge (Individuenbereich),
$M_2 = Df M_1$, M_3 sei die Menge aller zweistelligen Verknüpfungen auf M_1, M_4 sei die Menge aller zweistelligen Relationen auf M_1.
Der semantische Rahmen ist also durch M_1, den Individuenbereich bestimmt.

Eine Interpretation J bildet B_t in M_t ab *(i = 0, 1,...,4).*
Dabei ist $B_0 = \emptyset$, $B_1 = B_2$ = Menge aller Variablen, B_3 enthält nur das +-Zeichen, B_4 das $\leqslant$-Zeichen. Die Interpretation besteht also zunächst einmal aus einer Belegung der Variablen und einer Bedeutung für + und einer Bedeutung für $\leqslant$. Die Fortsetzung von J auf alle Ausdrücke von L_4 wird dann eindeutig in der folgenden Weise vorgenommen:

Ein gewisser Funktor liefert aus dem Funktionszeichen + und Termen *a, b* den Term *(a + b).* Die zugehörige semantische Verknüpfung liefert aus $+^J$, a^J , b^J den Wert: $+^J$ *angewendet auf* a^J , b^J .

Ein weiterer Funktor liefert aus dem Relationszeichen $\leqslant$ und den Termen *a, b* die Formel $a \leqslant b$. Die zugehörige semantische Verknüpfung liefert aus $\leqslant^J$, a^J , b^J den Wert W genau dann, wenn $\leqslant^J$ *auf* a^J , b^J *zutrifft,* sonst F. Zu dem Funktor, der aus Termen *a, b* die Formel $a = b$ liefert, gehört folgende semantische Verknüpfung: Aus a^J , b^J macht sie W, wenn a^J *dasselbe Objekt wie* b^J *ist,* sonst F.

Die Bewertung der mit Junktoren $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ aufgebauten Formeln übergehen wir und besprechen nur noch einen Quantorenschritt.

Dabei sei für $x \in M_2$ und eine Variable v J_x^v die Bewertung, die genau so gegeben ist wie J , bis auf den Umstand, daß v mit x belegt werde. Wir betrachten den Operator, der aus v und ϕ die Formel $\forall v\, \phi$ macht. Die zugehörige semantische Verknüpfung nimmt als Argumente Funktionen an, die jedem $x \in M_2$ den Wert $\phi^{J_x^v}$ zuordnen und hat als Wert W genau dann, wenn dieser Wertverlauf stets den Wert W hat, sonst F. Kurz gesagt: $\forall v\, \phi^J = W$ genau dann, wenn *für alle* $x \in M_2$ *gilt:* $\phi^{J_x^v} = W$.

In diesem Beispiel sind die Werte α^J der Basisausdrücke von der Logik her beliebig wählbar, während die zu den Funktoren und Operatoren gehörigen semantischen Verknüpfungen (die logischen Konstanten entsprechen) fest vorgeschrieben sind.

Das hier beschriebene semantische Konzept paßt gut für logische Sprachen. Es ist jedoch bemerkenswert, daß es auch für Fragmente vom Typ des Beispiels (3) paßt. Das ist von R. Montague entdeckt worden, der erstmals logische Sprachen und Fragmente in einheitlicher Weise syntaktisch und semantisch behandelt hat. Wir betrachten deshalb zum Abschluß die Beispielsprache L_3. Diese ist zwar syntaktisch trivial. Doch läßt sich in der Semantik die Montaguesche Interpretation der Quantoren demonstrieren, die keineswegs trivial ist.

Die Sprache L_3 hat die Kategorien: *S, E, N, V, NP, Det.* Wir legen zunächst einen semantischen Rahmen fest. Dabei sei $\mathbb{B} = \{W,F\}$ (Menge der Wahrheitswerte) und $\mathbb{D}$ eine nichtleere Menge (Individuenbereich).

Sodann sei:

$M_S = Df$ $\mathbb{B}$, $M_E = Df$ $\mathbb{D}$,
$M_N = M_V = Df$ *Menge aller Funktionen von* $\mathbb{D}$ *in* $\mathbb{B}$,
$M_{NP} = Df$ *Menge aller Funktionen von* M_V *in* $\mathbb{B}$,
$M_{Det} = Df$ *Menge aller Funktionen von* M_N *in* M_{NP}.

Für die Basisausdrücke fordern wir:

$CARUSO^J \in M_E$, $TENOR^J \in M_N$, $BABY^J \in M_N$, $SINGT^J \in M_V$, $SCHREIT^J \in M_V$. Von logischer Seite her werden die Werte dieser Basisausdrücke nicht weiter festgelegt, doch legen wir die Werte der anderen Basisausdrücke, die den Charakter von logischen Konstanten haben, fest.

$JEDER^J$ *sei diejenige Funktion aus* M_{Det}, *die jeder Funktion* $f : \mathbb{D} \rightarrow \mathbb{B}$ *diejenige Funktion* $g \in M_{NP}$ *zuordnet, die jeder Funktion* $h : \mathbb{D} \rightarrow \mathbb{B}$ *den Wert* W *genau dann zuordnet, wenn* h *an jeder Stelle, an der* f *den Wert* W *annimmt, auch den Wert* W *annimmt.*

EIN^J, $KEIN^J$ seien entsprechend definiert, wobei das in der oben stehenden Definition unterstrichene *"jeder"* durch *"einer"* bzw. *"keiner"* ersetzt wird.

DER^J *sei diejenige Funktion aus* M_{Det}, *die jeder Funktion* $f : \mathbb{D} \rightarrow \mathbb{B}$ *diejenige Funktion* $g \in M_{NP}$ *zuordnet, die jeder Funktion* $h : \mathbb{D} \rightarrow \mathbb{B}$ *den Wert* W *genau dann zuordnet, wenn* f *an genau einer Stelle* W *ist und* h *dort auch* W *ist, andernfalls ist* $g(h) = F$.

Schließlich setzen wir:

Wenn $\alpha \in E_{Det}$, $\beta \in E_N$, *so* $(\alpha\beta)^J = \alpha^J (\beta^J)$,
wenn $\alpha \in E_{NP}$, $\beta \in E_V$, *so* $(\alpha\beta)^J = \alpha^J (\beta^J)$,
wenn $\alpha \in E_E$, $\beta \in E_V$, *so* $(\alpha\beta)^J = \beta^J (\alpha^J)$.

Eigennamen haben wir direkt durch Individuen interpretiert, Verben und auch Nomina durch Funktionen von Individuen in Wahrheitswerte (der Funktionswert ist W, wenn die entsprechende Eigenschaft zutrifft). Man beachte aber, daß Nominalphrasen und Eigennamen ganz unterschiedlich interpretiert werden, obwohl sie sich syntaktisch bei der Bildung von Sätzen ganz gleichartig verhalten. Es ist ja auch semantisch nicht möglich, $JEDER\ TENOR^J$, $KEIN\ BABY^J$ mit einem Individuum gleichzusetzen, auf das dann etwa $SINGT^J$ zuträfe. Deshalb operiert nicht $SINGT^J$ auf $JEDER\ TENOR^J$ (wie es bei $CARUSO^J$ der Fall ist), vielmehr ist es umgekehrt:
$JEDER\ TENOR^J$ ist die Funktion, die $SINGT^J$ den Wert W zuordnet, wenn auf jedes Individuum, auf das $TENOR^J$ zutrifft, auch $SINGT^J$

zutrifft. *JEDER*^{J} ist dann die Funktion, die *TENOR*^{J} in diese Funktion *JEDER TENOR*^{J} überführt. Aus diesem Grunde ist die Interpretation von *JEDER* und analog die von *EIN, KEIN, DER* komplizierter, als die aller anderen Ausdrücke dieses Fragmentes.

Die Tatsache, daß Syntax und Semantik von logischen Sprachen und von formalisierten Fragmenten natürlicher Sprachen in völlig gleichartiger Weise entwickelt werden können, hat das gegenseitige Interesse von Linguistik und Logik aneinander in den letzten Jahren sehr belebt.