MARCUS KRACHT

The Mathematics of Language

STUDIES IN GENERATIVE GRAMMAR 63

MOUTON DE GRUYTER

The Mathematics of Language

by

Marcus Kracht

Mouton de Gruyter
Berlin · New York 2003

Mouton de Gruyter (formerly Mouton, The Hague)
is a Division of Walter de Gruyter GmbH & Co. KG, Berlin.

The series Studies in Generative Grammar was formerly published by
Foris Publications Holland.

Kracht, Marcus.
The mathematics of language / by Marcus Kracht.
p. cm. – (Studies in generative grammar ; 63)
Includes bibliographical references and index.
ISBN 3-11-017620-3 (alk. paper)
1. Mathematical linguistics. I. Title. II. Series.
P138 .K73 2003
410'.1'51–dc21

2003016857

∞ Printed on acid-free paper which falls within the guidelines
of the ANSI to ensure permanence and durability.

ISBN 3-11-017620-3

Bibliographic information published by Die Deutsche Bibliothek

Die Deutsche Bibliothek lists this publication in the Deutsche
Nationalbibliografie; detailed bibliographic data is available in the
Internet at <http://dnb.ddb.de>.

Cover design: Christopher Schneider, Berlin.
Printed in Germany.

The Mathematics of Language

Studies in Generative Grammar 63

Editors

Henk van Riemsdijk
Harry van der Hulst
Jan Koster

Mouton de Gruyter
Berlin · New York

Was dann nachher so schön fliegt . . .
wie lange ist darauf rumgebrütet worden.

Peter Rühmkorf: *Phönix voran*

Preface

The present book developed out of lectures and seminars held over many years at the Department of Mathematics of the Freie Universität Berlin, the Department of Linguistics of the Universität Potsdam and the Department of Linguistics at UCLA. I wish to thank in particular the Department of Mathematics at the Freie Universität Berlin as well as the Freie Universität Berlin for their support and the always favourable conditions under which I was allowed to work. Additionally, I thank the DFG for providing me with a Heisenberg–Stipendium, a grant that allowed me to continue this project in between various paid positions.

I have had the privilege of support by Hans–Martin Gärtner, Ed Keenan, Hap Kolb and Uwe Mönnich. Without them I would not have had the energy to pursue this work and fill so many pages with symbols that create so much headache. They always encouraged me to go on.

Lumme Erilt, Greg Kobele and Jens Michaelis have given me invaluable help by scrupulously reading earlier versions of this manuscript. Further, I wish to thank Helmut Alt, Christian Ebert, Benjamin Fabian, Stefanie Gehrke, Timo Hanke, Wilfrid Hodges, Gerhard Jäger, Makoto Kanazawa, Franz Koniecny, Thomas Kosiol, Ying Lin, Zsuzsanna Lipták, István Németi, Terry Parsons, Alexis–Manaster Ramer, Jason Riggle, Stefan Salinger, Ed Stabler, Harald Stamm, Peter Staudacher, Wolfgang Sternefeld and Ngassa Tchao for their help.

Los Angeles and Berlin, September 2003 *Marcus Kracht*

Introduction

This book is — as the title suggests — a book about the mathematical study of language, that is, about the description of language and languages with mathematical methods. It is intended for students of mathematics, linguistics, computer science, and computational linguistics, and also for all those who need or wish to understand the formal structure of language. It is a mathematical book; it cannot and does not intend to replace a genuine introduction to linguistics. For those who are not acquainted with general linguistics we recommend (Lyons, 1968), which is a bit outdated but still worth its while. For a more recent book see (Fromkin, 2000). No linguistic theory is discussed here in detail. This text only provides the mathematical background that will enable the reader to fully grasp the implications of these theories and understand them more thoroughly than before. Several topics of mathematical character have been omitted: there is for example no statistics, no learning theory, and no optimality theory. All these topics probably merit a book of their own. On the linguistic side the emphasis is on syntax and formal semantics, though morphology and phonology do play a role. These omissions are mainly due to my limited knowledge. However, this book is already longer than I intended it to be. No more material could be fitted into it.

The main mathematical background is algebra and logic on the semantic side and strings on the syntactic side. In contrast to most introductions to formal semantics we do not start with logic — we start with strings and develop the logical apparatus as we go along. This is only a pedagogical decision. Otherwise, the book would start with a massive theoretical preamble after which the reader is kindly allowed to see some worked examples. Thus we have decided to introduce logical tools only when needed, not as overarching concepts.

We do not distinguish between natural and formal languages. These two types of languages are treated completely alike. I believe that it should not matter in principle whether what we have is a natural or an artificial product. Chemistry applies to naturally occurring substances as well as artificially produced ones. All I will do here is study the structure of language. Noam Chomsky has repeatedly claimed that there is a fundamental difference between natural and nonnatural languages. Up to this moment, conclusive evidence for this claim is missing. Even if this were true, this difference should

not matter for this book. To the contrary, the methods established here might serve as a tool in identifying what the difference is or might be. The present book also is not an introduction to the theory of formal languages; rather, it is an introduction to the mathematical theory of linguistics. The reader will therefore miss a few topics that are treated in depth in books on formal languages on the grounds that they are rather insignificant in linguistic theory. On the other hand, this book does treat subjects that are hardly found anywhere else in this form. The main characteristic of our approach is that we do not treat languages as sets of strings but as algebras of signs. This is much closer to the linguistic reality. We shall briefly sketch this approach, which will be introduced in detail in Chapter 3.

A **sign** σ is defined here as a triple $\langle e, c, m \rangle$, where e is the **exponent of** σ, which typically is a string, c the **(syntactic) category of** σ, and m its **meaning**. By this convention a string is connected via the language with a set of meanings. Given a set Σ of signs, e **means** m **in** Σ if and only if (= iff) there is a category c such that $\langle e, c, m \rangle \in \Sigma$. Seen this way, the task of language theory is not only to say which are the legitimate exponents of signs (as we find in the theory of formal languages as well as many treatises on generative linguistics which generously define language to be just syntax) but it must also say which string can have what meaning. The heart of the discussion is formed by the principle of compositionality, which in its weakest formulation says that the meaning of a string (or other exponent) is found by homomorphically mapping its analysis into the semantics. Compositionality shall be introduced in Chapter 3 and we shall discuss at length its various ramifications. We shall also deal with Montague Semantics, which arguably was the first to implement this principle. Once again, the discussion will be rather abstract, focusing on mathematical tools rather than the actual formulation of the theory. Anyhow, there are good introductions to the subject which eliminate the need to include details. One such book is (Dowty *et al.*, 1981) and the book by the collective of authors (Gamut, 1991b). A **system of signs** is a partial algebra of signs. This means that it is a pair $\langle \Sigma, M \rangle$, where Σ is a set of signs and M a finite set, the set of so–called **modes (of composition)**. Standardly, one assumes M to have only one nonconstant mode, a binary function $\bullet$, which allows one to form a sign $\sigma_1 \bullet \sigma_2$ from two signs σ_1 and σ_2. The modes are generally partial operations. The action of $\bullet$ is explained by defining its action on the three components of the respective signs. We give a

simple example. Suppose we have the following signs.

$$\begin{aligned} \text{`runs'} &= \langle \texttt{runs}, v, \rho \rangle \\ \text{`Paul'} &= \langle \texttt{Paul}, n, \pi \rangle \end{aligned}$$

Here, v and n are the syntactic categories *(intransitive) verb* and *proper name*, respectively. π is a constant, which denotes an individual, namely Paul, and ρ is a function from individuals to the set of truth values, which typically is the set $\{0,1\}$. (Furthermore, $\rho(x) = 1$ if and only if x is running.) On the level of exponents we choose word concatenation, which is string concatenation (denoted by $^\frown$) with an intervening blank. (Perfectionists will also add the period at the end...) On the level of meanings we choose function application. Finally, let $\circ$ be a partial function which is only defined if the first argument is n and the second is v and which in this case yields the value t. Now we put

$$\langle e_1, c_1, m_1 \rangle \bullet \langle e_2, c_2, m_2 \rangle := \langle e_1 {}^\frown \square {}^\frown e_2, c_1 \circ c_2, m_2(m_1) \rangle$$

Then `'Paul' • 'runs'` is a sign, and it has the following form.

$$\text{`Paul'} \bullet \text{`runs'} := \langle \texttt{Paul runs}, t, \rho(\pi) \rangle$$

We shall say that this sentence is true if and only if $\rho(\pi) = 1$; otherwise we say that it is false. We hasten to add that `'Paul' • 'Paul'` is *not* a sign. So, $\bullet$ is indeed a partial operation.

The key construct is the free algebra generated by the constant modes alone. This algebra is called the **algebra of structure terms**. The structure terms can be generated by a simple context free grammar. However, not every structure term names a sign. Since the algebras of exponents, categories and meanings are partial algebras, it is in general not possible to define a homomorphism from the algebra of structure terms into the algebra of signs. All we can get is a partial homomorphism. In addition, the exponents are not always strings and the operations between them not only concatenation. Hence the defined languages can be very complex (indeed, every recursively enumerable language Σ can be so generated).

Before one can understand all this in full detail it is necessary to start off with an introduction into classical formal language theory using semi Thue systems and grammars in the usual sense. This is what we shall do in Chapter 1. It constitutes the absolute minimum one must know about these matters. Furthermore, we have added some sections containing basics from algebra,

set theory, computability and linguistics. In Chapter 2 we study regular and context free languages in detail. We shall deal with the recognizability of these languages by means of automata, recognition and analysis problems, parsing, complexity, and ambiguity. At the end we shall discuss semilinear languages and Parikh's Theorem.

In Chapter 3 we shall begin to study languages as systems of signs. Systems of signs and grammars of signs are defined in the first section. Then we shall concentrate on the system of categories and the so–called categorial grammars. We shall introduce both the Ajdukiewicz–Bar Hillel Calculus and the Lambek–Calculus. We shall show that both can generate exactly the context free string languages. For the Lambek–Calculus, this was for a long time an open problem, which was solved in the early 1990s by Mati Pentus.

Chapter 4 deals with formal semantics. We shall develop some basic concepts of algebraic logic, and then deal with boolean semantics. Next we shall provide a completeness theorem for simple type theory and discuss various possible algebraizations. Then we turn to the possibilities and limitations of Montague Semantics. Then follows a section on partiality and presupposition.

In the fifth chapter we shall treat so–called PTIME languages. These are languages for which the parsing problem is decidable deterministically in polynomial time. The question whether or not natural languages are context free was considered settled negatively until the 1980s. However, it was shown that most of the arguments were based on errors, and it seemed that none of them was actually tenable. Unfortunately, the conclusion that natural languages are actually all context free turned out to be premature again. It now seems that natural languages, at least some of them, are not context free. However, all known languages seem to be PTIME languages. Moreover, the so–called weakly context sensitive languages also belong to this class. A characterization of this class in terms of a generating device was established by William Rounds, and in a different way by Annius Groenink, who introduced the notion of a literal movement grammar. We shall study these types of grammars in depth. In the final two sections we shall return to the question of compositionality in the light of Leibniz' Principle, and then propose a new kind of grammars, de Saussure grammars, which eliminate the duplication of typing information found in categorial grammar.

The sixth chapter is devoted to the logical description of language. This approach has been introduced in the 1980s and is currently enjoying a revival. The close connection between this approach and the so–called constraint–programming is not accidental. It was proposed to view grammars not as

generating devices but as theories of correct syntactic descriptions. This is very far away from the tradition of generative grammar advocated by Chomsky, who always insisted that language contains a generating device (though on the other hand he characterizes this as a theory of competence). However, it turns out that there is a method to convert descriptions of syntactic structures into syntactic rules. This goes back to ideas by Büchi, Wright as well as Thatcher and Doner on theories of strings and theories of trees in monadic second order logic. However, the reverse problem, extracting principles out of rules, is actually very hard, and its solvability depends on the strength of the description language. This opens the way into a logically based language hierarchy, which indirectly also reflects a complexity hierarchy. Chapter 6 ends with an overview of the major syntactic theories that have been introduced in the last 25 years.

NOTATION. Some words concerning our notational conventions. We use typewriter font for true characters in print. For example: `Maus` is the German word for 'mouse'. Its English counterpart appears in (English) texts either as `mouse` or as `Mouse`, depending on whether or not it occurs at the beginning of a sentence. Standard books on formal linguistics often ignore these points, but since strings are integral parts of signs we cannot afford this here. In between true characters in print we also use so–called *metavariables* (placeholders) such as a (which denotes a single letter) and $\vec{x}$ (which denotes a string). The notation $\mathtt{c}_i$ is also used, which is short for the true letter `c` followed by the binary code of i (written with the help of appropriately chosen characters, mostly `0` and `1`). When defining languages as sets of strings we distinguish between brackets that appear in print (these are `(` and `)`) and those which are just used to help the eye. People are used to employ abbreviatory conventions, for example `5+7+4` in place of `(5+(7+4))`. Similarly, in logic one uses $\mathtt{p}_0\wedge(\neg\mathtt{p}_1)$ or even $\mathtt{p}_0\wedge\neg\mathtt{p}_1$ in place of $(\mathtt{p}_0\wedge(\neg\mathtt{p}_1))$. We shall follow that usage when the material shape of the formula is immaterial, but in that case we avoid using the true function symbols and the true brackets '`(`' and '`)`', and use '`⟨`' and '`⟩`' instead. For $\mathtt{p}_0\wedge(\neg\mathtt{p}_1)$ is actually *not* the same as $(\mathtt{p}_0\wedge(\neg\mathtt{p}_1))$. To the reader our notation may appear overly pedantic. However, since the character of the representation is part of what we are studying, notational issues become syntactic issues, and syntactical issues simply cannot be ignored. Notice that '⟨' and '⟩' are truly metalinguistic symbols that are used to define sequences. We also use sans serife fonts for terms in formalized and computer languages, and attach a prime to refer to its denotation (or meaning). For example, the computer code for a while–loop is written

semi–formally as while $i < 100$ do $x := x \times (x + i)$ od. This is just a string of symbols. However, the notation $\mathsf{see}'(\mathsf{john}', \mathsf{paul}')$ denotes the proposition that John sees Paul, not the sentence expressing that.

Contents

1	**Fundamental Structures**	1
1	Algebras and Structures	1
2	Semigroups and Strings	16
3	Fundamentals of Linguistics	29
4	Trees	43
5	Rewriting Systems	52
6	Grammar and Structure	66
7	Turing machines	80
2	**Context Free Languages**	95
1	Regular Languages	95
2	Normal Forms	103
3	Recognition and Analysis	117
4	Ambiguity, Transparency and Parsing Strategies	132
5	Semilinear Languages	147
6	Parikh's Theorem	160
7	Are Natural Languages Context Free?	165
3	**Categorial Grammar and Formal Semantics**	177
1	Languages as Systems of Signs	177
2	Propositional Logic	191
3	Basics of λ–Calculus and Combinatory Logic	207
4	The Syntactic Calculus of Categories	225
5	The AB–Calculus	239
6	The Lambek–Calculus	249
7	Pentus' Theorem	258
8	Montague Semantics I	269
4	**Semantics**	281
1	The Nature of Semantical Representations	281
2	Boolean Semantics	296
3	Intensionality	308
4	Binding and Quantification	323
5	Algebraization	332

6	Montague Semantics II	343
7	Partiality and Discourse Dynamics	354
5	**PTIME Languages**	367
1	Mildly–Context Sensitive Languages	367
2	Literal Movement Grammars	381
3	Interpreted LMGs	393
4	Discontinuity	401
5	Adjunction Grammars	414
6	Index Grammars	424
7	Compositionality and Constituent Structure	434
8	de Saussure Grammars	447
6	**The Model Theory of Linguistic Structures**	461
1	Categories	461
2	Axiomatic Classes I: Strings	470
3	Categorization and Phonology	485
4	Axiomatic Classes II: Exhaustively Ordered Trees	505
5	Transformational Grammar	515
6	GPSG and HPSG	529
7	Formal Structures of GB	540
Bibliography		555
Index		573

Chapter 1
Fundamental Structures

1. Algebras and Structures

In this section we shall provide definitions of basic terms and structures which we shall need throughout this book. Among them are the notions of *algebra* and *structure*. Readers for whom these are entirely new are advised to read this section only cursorily and return to it only when they hit upon something for which they need background information.

We presuppose some familiarity with mathematical thinking, in particular some knowledge of elementary set theory and proof techniques such as induction. For basic concepts in set theory see (Vaught, 1995) or (Just and Weese, 1996; Just and Weese, 1997); for background in logic see (Goldstern and Judah, 1995). Concepts from algebra (especially universal algebra) can be found in (Burris and Sankappanavar, 1981) and (Grätzer, 1968), and in (Burmeister, 1986) and (Burmeister, 2002) for partial algebras; for general background on lattices and orderings see (Grätzer, 1971) and (Davey and Priestley, 1990).

We use the symbols $\cup$ for the union, $\cap$ for the intersection of two sets. Instead of the difference symbol $M \backslash N$ we use $M - N$. $\varnothing$ denotes the empty set. $\wp(M)$ denotes the set of subsets of M, $\wp_{fin}(M)$ the set of finite subsets of M. Sometimes it is necessary to take the union of two sets that does not identify the common symbols from the different sets. In that case one uses $+$. We define $M + N := M \times \{0\} \cup N \times \{1\}$ ($\times$ is defined below). This is called the **disjoint union**. For reference, we fix the background theory of sets that we are using. This is the theory ZFC (Zermelo Fraenkel Set Theory with Choice). It is essentially a first order theory with only two two place relation symbols, $\in$ and $=$. (See Section 3.8 for a definition of first order logic.) We define $x \subseteq y$ by $(\forall z)(z \in x \to x \in y)$. Its axioms are as follows.

1. *Singleton Set Axiom.* $(\forall x)(\exists y)(\forall z)(z \in y \leftrightarrow z = x)$.
   This makes sure that for every x we have a set $\{x\}$.

2. *Powerset Axiom.* $(\forall x)(\exists y)(\forall z)(z \in y \leftrightarrow z \subseteq x)$.
   This ensures that for every x the power set $\wp(x)$ of x exists.

3. *Set Union.* $(\forall x)(\exists y)(\forall z)(z \in y \leftrightarrow (\exists u)(z \in u \wedge u \in x))$.
   u is denoted by $\bigcup_{z \in x} z$ or simply by $\bigcup x$. The axiom guarantees its existence.

4. *Extensionality.* $(\forall x)(\forall y)(x = y \leftrightarrow (\forall z)(z \in x \leftrightarrow z \in y))$.

5. *Replacement.* If f is a function with domain x then the direct image of x under f is a set. (See below for a definition of *function*.)

6. *Weak Foundation.*

$$(\forall x)(x \neq \varnothing \rightarrow (\exists y)(y \in x \wedge (\forall z)(z \in x \rightarrow z \notin y)))$$

   This says that in every set there exists an element that is minimal with respect to $\in$.

7. *Comprehension.* If x is a set and φ a first order formula with only y occurring free, then $\{y : y \in x \wedge \varphi(y)\}$ also is a set.

8. *Axiom of Infinity.* There exists an x and an injective function $f : x \rightarrow x$ such that the direct image of x under f is not equal to x.

9. *Axiom of Choice.* For every set of sets x there is a function $f : x \rightarrow \bigcup x$ with $f(y) \in y$ for all $y \in x$.

We remark here that in everyday discourse, comprehension is generally applied to all collections of sets, not just elementarily definable ones. This difference will hardly matter here; we only mention that in monadic second order logic this stronger from of comprehension is expressible and also the axiom of foundation.

Full Comprehension. For every class P and every set x, $\{y : y \in x \text{ and } x \in P\}$ is a set.

Foundation is usually defined as follows

Foundation. There is no infinite chain $x_0 \ni x_1 \ni x_2 \ni \cdots$.

In mathematical usage, one often forms certain collections of sets that can be shown not to be sets themselves. One example is the collection of all finite sets. The reason that it is not a set is that for every set x, $\{x\}$ also is a set. The