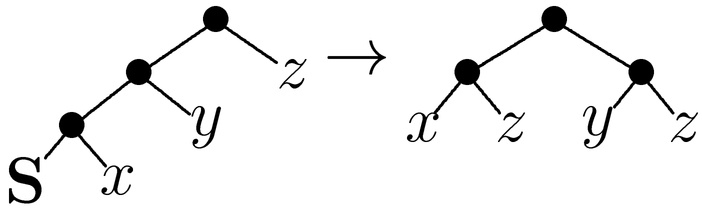
The Combinator S



vom Rat der Fakultät für Mathematik und Informatik
der Friedrich-Schiller-Universität Jena genehmigte

Dissertation

zur Erlangung des akademischen Grades
doctor rerum naturalium (Dr. rer. nat.)

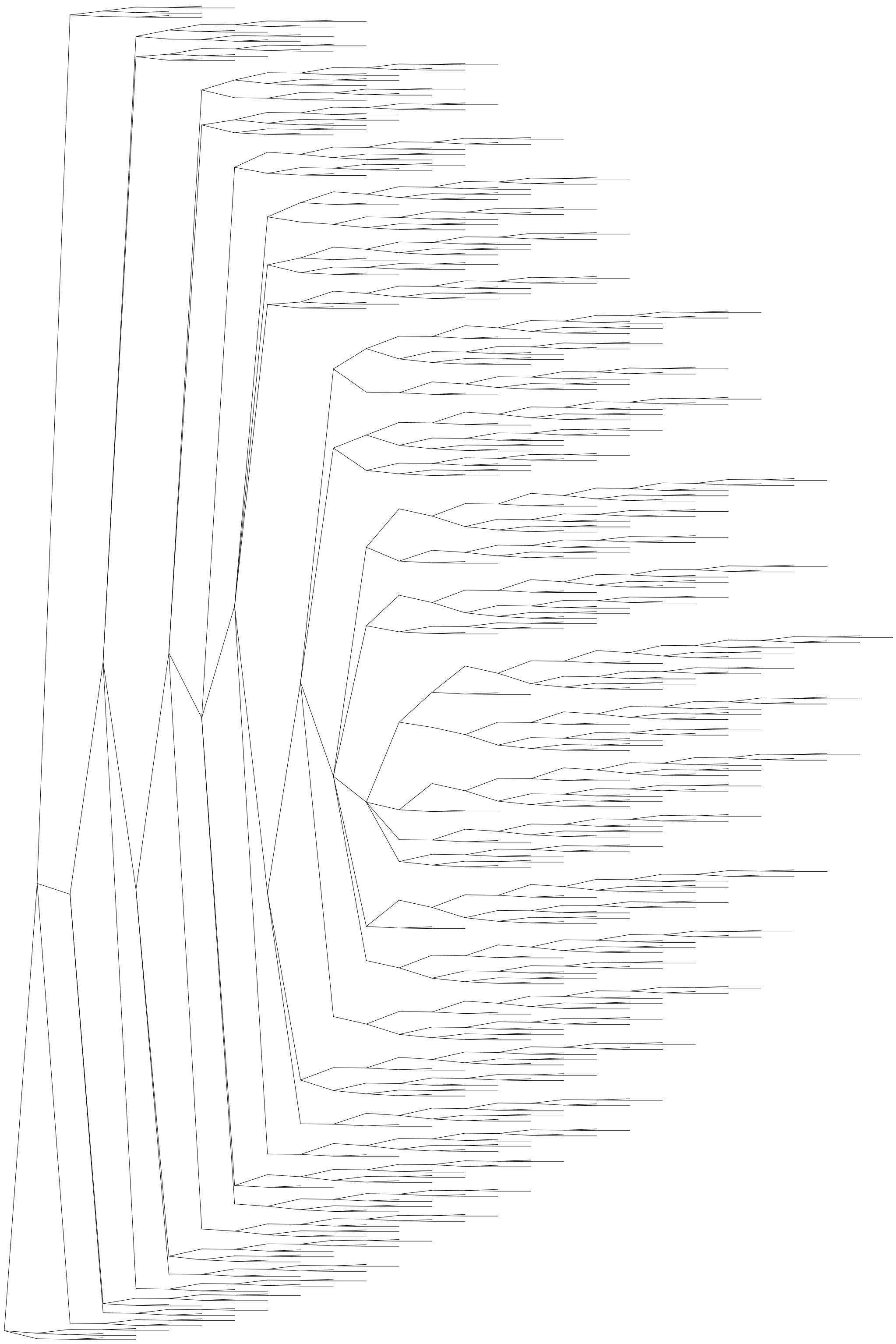
von Diplom-Mathematiker Johannes Waldmann
geboren am 3. Oktober 1967 in Gera

Gutachter:

1. Prof. Dr. Hans-Dietrich Hecker, Jena
2. Prof. Dr. Jan Willem Klop, Amsterdam
3. Prof. Dr. Horst Reichel, Dresden

Tag des Rigorosums: 5. Februar 1998

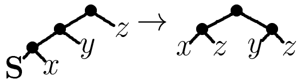
Tag der öffentlichen Verteidigung: 20. Februar 1998



S (S S S) S (S (S S) S)
after 50 leftmost innermost reductions

Introduction

This thesis analyzes the rewriting system $\mathbf{CL}(\mathbf{S})$: combinatory logic with the single combinator



We show that rational tree languages can be used in order to describe interesting properties of $\mathbf{S}$-terms.

Term rewrite systems are *complicated* in the following sense: they constitute a model of computation that is Turing-complete. Therefore, a lot of questions are undecidable in general.

On the other hand, rational languages are *handy*: they can be described by finite machines (tree automata), and here a lot of questions are decidable and a lot of operations on rational languages are effectively computable.

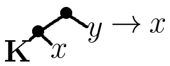
It is desirable to explore the borderline between these two realms: How far can we go in using rational languages when describing properties of term rewriting systems?

This thesis is intended to be a very detailed case study, yielding data that invites finding a more abstract description of the phenomena that are presented here.

We regard combinatory logic as a special term rewriting system. It acts on term algebras with some nullary function symbols (namely, the *combinators*) and exactly one binary function symbol (usually called *application*). Further, each combinator has an associated reduction rule whose left hand side must be of a simple shape.

Because of these restrictions, combinatory logic has the nice property that it is a *confluent* term rewrite system. Moreover, $\mathbf{CL}(\mathbf{S})$ is *non-erasing.*

Still, combinatory logic remains complicated in general because there are *complete sets* (called *bases*) of combinators that can simulate arbitrary rewrites, and therefore are computationally equivalent to Turing machines. A well known example is the two-element base $\{\mathbf{S}, \mathbf{K}\}$ with



If we now drop $\mathbf{K}$ from $\mathbf{CL}(\mathbf{S},\mathbf{K})$, we arrive at the system $\mathbf{CL}(\mathbf{S})$, which is no longer complete. Since $\mathbf{CL}(\mathbf{K})$ looks easy, we feel that $\mathbf{S}$ is responsible for the difficulty of $\mathbf{CL}(\mathbf{S},\mathbf{K})$. Therefore, $\mathbf{CL}(\mathbf{S})$ should be nearly as hard as $\mathbf{CL}(\mathbf{S},\mathbf{K})$. However some questions for $\mathbf{CL}(\mathbf{S})$ really *are* solvable, and we present these new results:

- $\mathbf{CL}(\mathbf{S})$ is top-terminating.
- $\mathbf{CL}(\mathbf{S})$ admits no ground loops.
- It is decidable whether an $\mathbf{S}$-term normalizes.
- The set of normalizing $\mathbf{S}$-terms is rational.

In this list, the last item has to be taken *cum grano salis* because it depends on a proof by computer that has only been partially completed.

This thesis also includes the description of the software system `RX`. It handles rational tree languages, and it has been used in finding and verifying results presented in this thesis.

Basically, `RX` acts as a desk top calculator that operates on rational languages. It is not tied to $\mathbf{CL}(\mathbf{S})$, but can do computations in arbitrary term algebras (with fixed arities).

On top of this, `RX` processes its input and output so that it co-operates nicely with text formatters as LaTeX[Lam94], in the spirit of a *literate programming* [Knu92] tool. For instance, `RX` had been used to format [Wal97].

Acknowledgments

The research presented in this thesis has been carried out at the Institut für Informatik at the Fakultät für Mathematik und Informatik at the Friedrich-Schiller-Universität Jena.

I am indebted to a lot of people for their assistance, encouragement and understanding during the time of my studies and PhD research.

Klaus Haberland's lectures greatly influenced my view of mathematics. I've always enjoyed discussing various topics in computer science with Traugott Schulmeiß.

Jan Willem Klop's article [Klo80] sparked my interest in combinators in general, and **S** in particular.

My supervisor, Hans-Dietrich Hecker, gave me the opportunity to follow my research interest, and the researchers from the Theoretical Computer Science group at Jena provided general support and encouragement.

Several term rewriters patiently answered and discussed questions on $\mathbf{CL}(\mathbf{S})$ with me. Especially I'd like to thank Henk Barendregt (Nijmwegen), Jan Willem Klop (CWI Amsterdam), Richard Statman (CMU Pittsburgh), for remarks on combinators; Dieter Hofbauer (TU Berlin), Alfons Geser (Tübingen), Vincent van Oostrom (VU Amsterdam), for general advice on term rewriting; Sophie Tison (LIFL, Lille), Thomas Genet (LORIA, Nancy), Florent Jacquemard (MPI Saarbrücken), Gregory Kucherov (LORIA, Nancy), for information on rational languages and rewriting.

Thomas Fröhlich (TU Ilmenau) provided me with some literature that wasn't available locally.

Alfons Geser spotted the reference to Zachos' paper [Zac78] "buried" in the bibliography of Wilfried Kurth's thesis [Kur90].

In September 1997, just before finishing writing this thesis, I visited the Research Group Theoretical Computer Science at Amsterdam University. I thank Jan Willem Klop and Vincent van Oostrom for hosting me, and providing a pleasant working environment.

I am very much indebted to Vincent van Oostrom for carefully reading the first version of this thesis.

Wolfgang Schwarz (Dresden) gave detailed comments on style and presentation.

This thesis, and some of the research described therein, was prepared by using free software: the Glasgow Haskell Compiler [Tea96], the Haskell B Compiler [Aug96], the nhc Haskell compiler [Röj96] the Hugs Haskell interpreter [Jon96], the Emacs text editor, the TeX and LaTeX text formatter, tools like BibTeX and `makeindex`, the GNU/Linux operating system.

I am indebted to Frank Thierbach's pioneering LaTeX enthusiasm and wisdom. The people at our faculty's Computing Services Center, notably Erik Braun and Jutta Sieron, proved to be experts at installing and maintaining software.

Contents