Origins of our Theory of Computation on Abstract Data Types at the Mathematical Centre, Amsterdam, 1979-80

J V Tucker
Department of Computer Science, University of Wales Swansea,
Singleton Park, Swansea, Wales, SA2 8PP

J I Zucker
Department of Computing and Software, McMaster University,
Hamilton, Ontario, Canada, L8S 4L7

To Jaco de Bakker
With gratitude, admiration and affection

1. Introduction

In 1979 the authors (hereafter JVT and JIZ) began our work together on developing a theory of computation that works for *any* data. We were members of Jaco de Bakker's research group at the Mathematical Centre. There we learnt about the semantics of programs and specifications, and about proof systems for program verification. There we found the mathematical form of our theory and its first application in computing. We will recall those times, describe the beginnings of our research programme, record some of Jaco's influence, and mention some present developments.

Our theory is about computation over *arbitrary* data types. It combines the algebraic theory of data with different programming language constructs and their semantics. Data is modelled using many sorted universal algebras and specified using equations or conditional equations. Programming constructs of many kinds control and extend the operations and tests of the algebra to compute sets and functions on the data. The theory is a generalisation of the now classical theory of computable functions on natural numbers or strings.

In 1979 the theories of programming language semantics and data types were both young. From the beginning, we have aimed to create a comprehensive mathematical theory for computing functions and sets, and to develop plenty of applications. In fact we have aimed to match, and in some areas supersede, classical computation theory based on natural numbers or strings. How much we have succeeded is for another occasion to judge. Something of a progress report was contained in *J W de Bakker, 25 Jaar Semantiek: Liber Amicorum* in 1989. We include our current bibliography for interest. We are still working on both theoretical and applied problems and there are many

In: F de Boer, M van der Heijden, P Klint, J Rutten (eds.), *Liber Amicorum, J W de Bakker,* CWI, Amsterdam, 197-221

problems in our research plans awaiting their turn. For the past few years we have been particularly focussed on computational problems on discrete and continuous data types and the analysis of that interface.

At the heart of our work is our theory of computable functions and semicomputable sets on many sorted algebras. Our theory is *abstract* in the sense that computations are independent of the representations of data and so are isomorphism invariants. It is *general* in the sense that it covers all data types - since all data can be modelled by many sorted algebras. Thus, from our theory of computable functions on algebras we can derive computability theories on particular classical data types, like number systems (e.g., natural numbers, real numbers, and complex numbers) and syntax (e.g., strings, terms). We can also derive computability theories for mathematical constructions (e.g., spaces of continuous functions) and general axiomatic classes of data types (e.g., semigroups, groups, rings, fields, vector spaces, Banach and Hilbert spaces). We can add storage structures (e.g., arrays, stacks and queues) and notions like time (e.g., timed data streams and waveforms) and location (e.g., addresses and co-ordinate systems). Over the years we have considered all of these examples of many sorted algebras.

Recently, we published a 200 page survey (Tucker and Zucker [2000]) of our theory, based on one of the simplest models of computation that we have studied in depth, namely the **while** programming language and its extensions. This survey contains a detailed history of the subject and account of the principal theorems. It also has a taste of our current research problems, namely to use the general theory to create a theory of computation on topological data and apply it in modelling, specification and programming.

Our first publication was *Program correctness over abstract data types with error-state semantics* (Tucker and Zucker [1988]). It was a research monograph based on Jaco's book *The Mathematical Theory of Program Correctness* (De Bakker [1980]). It generalised and extended Jaco's work on semantics and proof systems for program correctness. Its last chapter was on computable functions on many sorted algebras.

2. At The Mathematical Centre 1978-79

2.1 Background

In 1978, the Mathematical Centre, or MC, was at the Tweede Boerhaavestraat, next to the old Amstel Brewery. Its Director was Aad van Wijngaarden. A lot of excellent people in Computer Science had worked there almost from its inception in 1946. The Director had joined the MC in 1947 and begun work on the construction of the first Dutch computers (the ARRA, ARMAC and X1). Among his 15 doctoral students and former colleagues at the MC were most of the Dutch professoriate in the period. In 1978 three of his students were at the MC: Jaco, Hans van Vliet and Dick Grune. Peter van Emde Boas had recently moved to Amsterdam University but was frequently to be met at the MC. In 1978, the MC had a mature and strong computer science research culture.

Jaco joined the MC in 1964 and was awarded his doctorate under van Wijngaarden in 1967. Jaco was a theoretical computer scientist with a fine international reputation as a pioneer in the field of programming language semantics. He was also experienced in organising and encouraging this new scientific field through the foundation of the *European Association for Theoretical Computer Science* (EATCS), journals like Elsevier's *Theoretical Computer Science*, and international summer schools at the MC. He was Head of the Computer Science Department, which was small but first-rate.

While doing research, Jaco was also engaged in writing his graduate textbook and research monograph *The Mathematical Theory of Program Correctness*. This book was to give a self-contained account of the semantics of imperative programming language constructs. These constructs were introduced one by one, in small languages designed to focus on the computational ideas behind the construct. The operational and denotational styles of semantics were introduced and proved equivalent. Furthermore, for the constructs and their languages, a Floyd-Hoare logic was given to prove partial correctness assertions. Each logic was proved sound and complete with respect to its semantics. All computations, specifications and proof systems involved natural numbers and Booleans *only*.

At the time defining the meaning of programming constructs and languages was in some circles an important methodological problem. Program verification had to seen against the backcloth of axiomatic semantics. This was the idea that one could specify the meaning of a programming language by postulating axioms and rules for a logic to prove the correctness of programs. This was a high-level approach to defining the meaning of a programming language and one centred on the user of the language.

Euclid had axiomatised geometry circa 300 BC, and axiomatic foundations for the rest of mathematics had been created in the 19th and early 20th centuries. The conception was simple enough, but the implementation was exciting and hugely ambitious. The possibility of an axiomatic semantics for a programming language first appears in Floyd [1967] and is beautifully explored in Hoare [1968]. Axiomatic approaches to other programming issues had appeared: for example, van Wijngaarden [1966] for computer arithmetic, and De Bakker [1968]. The whole enterprise would now be seen as part of the search for verification methods for software (Jones [1994]).

In the 1970s proof rules were being invented for constructs that had no independent semantics. Faulty rules were not uncommon. The problem of correctness was recognised as a fundamental and pervasive problem and was being attacked by formal methods, but the first formal methods lacked theoretical foundations. Jaco's book was part of the research effort into providing these foundations.

Assisting Jaco at the MC was Arie de Bruin, who was working on his PhD thesis under Jaco's supervision (De Bruin [1986]). Former colleagues in semantics at the MC had moved on but were in regular contact, including Willem Paul de Roever (at Utrecht) and Krzysztof Apt (at Rotterdam).

Among the cognoscenti in 1979, research in concurrency was recognised as a large, exciting and *open* area. Willem Paul had visited Tony Hoare in Belfast and become interested in CSP. The paper Owicki and Gries [1976] was taken up in new work by Willem Paul and Krzysztof. Their seminars and pre-prints were attracting a great deal of attention locally and internationally (Apt, Francez, and De Roever [1980]). Much later they were to write their own books Apt and Olderog [1991], Francez [1992], De Roever et al [2001]

Although Jaco's book was completed at the time concurrency research was taking off semantically, he did not attempt to include parallelism. In Chapter 7 (page 258), on *Nondeterminsitic Statements*, Jaco gives as the first of three reasons for the study of nondeterministic choice

$$S_1 \square S_2$$

its role in the arbitrarily interleaving semantics of parallel execution. However, he noted that "… the present status of the research on the semantics and proof theory of parallelism is, in our opinion, not yet such that it justifies a treatment on the textbook level…"

Indeed, it all looked awfully difficult to JIZ and JVT at the time.

2.2 Our Collaboration

JIZ joined Jaco's group in September 1978, moving from Dirk van Dalen's logic group at Utrecht. JIZ was an experienced logician and proof theorist, and was settled in the Netherlands having held appointments in Amsterdam and Eindhoven. JIZ's colleagues during his three years with Dirk's group in Utrecht (1975-1978) had included two of Dirk's doctoral students, Jan Willem Klop and Jan Bergstra, who would later have a profound impact on Dutch computer science.

JVT joined in January 1979 from Oslo where he had been a member of the group of Jens Erik Fenstad on generalised recursion theory. JVT's colleagues during his two years with Jens Erik included Dag Normann, Viggo Stoltenberg-Hansen and Johan Moldestad. He had written to Jaco describing his research and inquiring if there were any visiting posts now or in the future. It turned out that Krzysztof was leaving the MC for Rotterdam and there was a vacancy. Jaco invited JVT to come for a year. They had not yet met.

JIZ and JVT shared the end office in the little "temporary" building in the courtyard of the MC. Accommodation at the Tweede Boerhaavestraat was tight. From the ground floor of the main building, you passed into the courtyard, beside the windows of the punch card room, and into a single storey prefab with offices on one side of a corridor only, stocked with computer scientists. Our office was at the far end. There were two grey metal desks in the centre of the room, at which we sat facing one another. This was ideal for conversation. JIZ sat with his back to the window, his papers and notes surrounded by 3B Staedtler pencils and erasers.

The collaboration got off to an excellent start. JVT was impressed by JIZ's wide intellectual interests, especially his knowledge of logic, philosophy and linguistics, partly represented by the excellent collection of books which filled the grey metal bookcases in our office. JIZ was impressed, then and later, by JVT's facility for social interaction, based on a natural curiosity about other people, and an ability to draw them out about their own ideas, and discern connections with his work - a talent which led to a number of fruitful collaborations with different researchers.

JIZ and JVT had a lot in common. We were devoted to logic, loved literature, and delighted in conversation. JIZ was deeply committed to music and JVT enthralled by fine art. We were both at the MC to learn, too. Our logical expertise had given us a close connection with theoretical computer science. From different directions – proof theory and computability theory - we were now simply moving into the subject. Jaco's department was the first computer science department either of us had worked in. We were by no means the first or the last logicians to enter the subject through joining Jaco's department!

Our immediate next door neighbours were Hans van Vliet and Dick Grune who were involved in the later stages of the Algol 68 project. They kept a coffee percolator in their room. They were frequently interrupted when in residence by serious coffee drinkers like us, and by the noise of our animated conversations. Hans and Dick both worked for long periods on the computers in a terminal room in the main building. Their neighbours were Paul Klint and Lambert Meertens, also often in the terminal room. JIZ was, probably, the only inhabitant of the courtyard who did not smoke at that time.

Other theoreticians at the MC were Paul Vitanyi and Peter J Asveld, who were working on complexity theory and formal languages in the main building. High up in the main building was Jaco's office, where all good bosses should be.

Our working days had a pattern. JIZ arrived early and JVT arrived late, though always with time to do something before the morning visit of Mien, the tea lady, which signalled the start of our daily conversation. JVT spent at least an hour exploring the library every day. For the pleasure of browsing, no library compared with the MC's. Each day we went out to lunch in a small sandwich and coffee shop in the Weesperplein, often with Paul and Peter. The talk was largely of theoretical computer science. The afternoon was very much for conversation.

In January 1979 we started to discuss two subjects of interest to us, a little each day. The topics seemed quite independent.

1. *Logics for program verification* and the semantics of imperative constructs, based on Jaco's approach, and the soundness and completeness of Floyd-Hoare logics, based on Steve Cook's approach.

2. The *generalisation of computability theory* from natural numbers to arbitrary algebraic and relational structures based on various models, notably the finite algorithmic procedures of Harvey Friedman, which was a generalised register machine approach.

It took until Spring 1979 to see the natural connections between these subjects and ask some obvious questions.

JIZ was pursuing various research topics in logic, including work on phenomenology and logic with Robert Tragesser. He was also working closely with Jaco on the later stages of his book. Through reading the book closely, he became interested in the expressiveness of the assertion language in the proofs of completeness. He was working out a self-contained proof of the expressiveness of first order logic for pre- and post-conditions for computation by recursive procedures on the natural numbers, to be incorporated in the book as an appendix. This was quite an exercise and gave him the opportunity to dig into the technicalities of the relevant semantical and logical issues. It demanded more frequent wanderings up and down the corridor than many of the problems we have worked on since.

JVT was already working on generalising computability theory. At Oslo he had written some papers on computability theory on algebraic structures. Theorems in Moldestad, Stoltenberg-Hansen and Tucker [1980a,1980b] bridged the gap between abstract register machine computations (a la Harvey Friedman and John C Shepherdson) and (i) axiomatic computation theories and (ii) fixed point methods in higher type recursion. The results had been included in the monograph Fenstad [1980]. Recently, in Fenstad [2002], Jens Erik has reflected on that period of generalised recursion theory, both historically, and in the light of some of our more recent results.

JVT had been introduced to Harvey Friedman's paper in his first year of graduate studies at Bristol, in a course given by John Shepherdson in January 1974. However, his PhD had been on computable universal algebras, groups and fields under John P Cleave. Computable algebras of various kinds had been studied by M O Rabin and A I Mal'cev. JVT had started to work on John Cleave's ideas for a method of analysing the computability of uncountable topological algebras (specifically the classical matrix groups) using generalised enumerations from Baire space, but had found the Mal'cev theory for countable algebras under-developed. Cleave did not publish his approach and, quite independently, Klaus Weihrauch later created an extraordinary theory of computation for topological spaces using enumerations from Baire and Cantor space starting in Kreitz and Weihrauch [1985] and recently described in Weihrauch [2000].

In the first half of 1979 he was finishing a paper with Viggo Stoltenberg-Hansen on the undecidability of the roots of unity problem in computable fields. He was also actively learning about programming language theory and semantics. In the lunchtime meetings with Paul and Peter the subject of complexity theory often featured. JVT and Peter started to work together on complexity results for abstract models of computation on universal algebras (Asveld and Tucker [1982]).

In February 1979, JVT began regular meetings with Jan Bergstra. They had become friends through Jan's visits to Oslo, when he was active in higher type recursion theory. Discussions were focussed on algebraic specification methods for data types and logical foundations for program specification and correctness. As their research developed, the pattern was established of JVT visiting Leiden or Utrecht, usually on a Tuesday, to discuss a variety of subjects, and, especially, the week's progress on their investigations.

Algebraic methods for the specification of data types had been developed in the programming methodology literature by Barbara Liskov and Stephen Zilles, and by John Guttag. Extensive case studies showed that there were many ways to use many sorted equations, conditional equations, and other formulae to axiomatise the operators of data types and this raised many questions as to which were best. Not surprisingly, some methods were expressed through examples only. The algebraic specification techniques were first analysed mathematically by the ADJ Group, as part of their general research programme on initial algebra semantics (Goguen, Thatcher, Wagner and Wright [1977]). The study of the scope and limits of the methods led to a deep collaboration between JVT and Jan that uncovered intimate connections between computability, specification and verification. The first of many MC Reports they devoted to these subjects appeared in 1979 (see Bergstra and Tucker [1979 a, b, c]. The theory of data types had taken root in The Netherlands.

The algebraic approach to data types was also discussed at length in the office at the MC. It gave a new perspective to the search for a general theory of computable functions on algebras and raised the question what is the theory of specification and correctness on data types other than the natural numbers. JIZ began to actively study abstract computability theory. He wondered if the whole of Jaco's approach could be generalised to arbitrary structures. He had met many sorted structures through the interest of his supervisor, Sol Feferman, at Stanford. The many sortedness was exactly right for modelling data. The computability theory showed that the assertion languages could no longer be first order because they could not express even simple computational conditions, but that a weak second order assertion language (allowing finite sequences or arrays) would be adequate. JIZ suggested we write an MC Tract to see how the theory looked. He also suggested we might look at errors. We agreed and proposed the idea to Jaco who was enthusiastic.

We planned out our monograph keeping closely to the pattern of the first five chapters of Jaco's book. JIZ would write the three correctness chapters and JVT would write the introduction to the project and a final chapter on computability on many sorted algebras. It was an extension of Jaco's approach, designed to allow applications to any data.

At the end of September 1979, JIZ left The Netherlands for Bar Ilan University in Israel. Our plans were clear and it was agreed that he would make return visits to Amsterdam to work on the end of one book (Jaco's) and the start of another (our own). JVT was left working on a paper on generalised computability theory and its applications to abstract algebra. It was the published version of his invited lecture at the Association for

Symbolic Logic Summer Meeting at Leeds in 1979. This was the last of the register machine work that he had started in Oslo (Tucker [1980]).

Ralph Johan Back (Helsinki) took JIZ's desk and new conversations started up, this time on programming methodology, refinement and logic. JVT learnt a great deal of contemporary thinking on software construction from Ralph. Whist Ralph introduced him to several research problems JVT was too slow a pupil to write to a joint paper.

The following year, 1980, saw the move of the MC from the Tweede Boerhaavestraat to the Kruislaan. At lunch this was longed for by Ralph (used to the quality and modernity of Scandinavian interiors) and bemoaned by Paul and JVT (used to the MC). One lunchtime we upped and went to see the new building before its completion to prove our respective points. However the debate descended into speculations on how to get there, where the sandwich shops were, and where our offices might be. When the MC moved we still managed to get there each day, there was a canteen, and Ralph and JVT were in the same office.

That summer brought the seventh *International Colloquium on Automata, Languages and Programming* (ICALP) to Noordwijkerhout, organised by Jaco and others. With it came visitors to Amsterdam, including Jim Thatcher who had been working out the mathematical theory of data types with Joe Goguen, Eric Wagner and Jesse B Wright in one fine paper after another for several years. On September 1, came the retirement of Aad van Wijngaarden, and changes in the organisation of the MC. Ralph returned to Helsinki but the conversations continued, JVT visiting him in Helsinki in 1981. Subsequently, Ralph has influenced the development of formal methods for software through his deep study of refinement, starting in his thesis and MC Tract, Back [1980], and most recently described in his book, Back and von Wright [1998]. He has also had a strong effect on Finnish computer science, of course.

How was our *Program correctness over abstract data types with error-state semantics* written? Slowly. We received a lot of patient encouragement and support from Jaco over the nine years it took to publish the book in 1988. We were occupied with other research problems: for example, JIZ and Jaco were working on concurrent processes (see below), and JVT and Jan Bergstra continued their work on data type specifications and Hoare logic. There were disruptions. In January 1981, JVT returned for a brief period to Bristol University in the UK. In October 1981 he settled at Leeds University. After three years, JIZ moved from Bar Ilan to SUNY at Buffalo, New York.

However, technology, too, moved on. The IBM typewriters that had produced Jaco's book were passing into history. Our book was produced by JIZ using the Ditroff text processor under UNIX with a QMS Lasergrafix 1200, the first laser printer to be used at Buffalo's computer science department, funded by his grant from National Science Foundation. From that time to this day, we write `tt` and `ff` for Booleans, in fond memory of Jaco's typewriter notation for 𝕥 and 𝕗 in his book (using 3B Staedtler pencils, of course).

3. After Jaco's Book

When Jaco's book was finished, there was an obvious interesting question for everyone in his circle. What was Jaco going to do next? There was a definite sense of achievement in the completion and appearance of the book in 1980. In it the basic technical issues of semantic modelling, and its connection with logics for program verification, were answered clearly, rigorously and in detail. For imperative programming, the book sorted out technical issues that had been causing problems since the emergence of semantic modelling and formal verification a decade earlier. This sense of achievement can also be found in Krzysztof's survey paper *Ten years of Hoare Logic* (Apt [1981]). The basic approach to semantics went something like this:

- design constructs that require semantic models
- design syntax-directed logics for the constructs
- prove soundness and completeness.

If you wanted to do this properly then you had know Jaco's book.

In terms of research, as the book came to completion, it seemed to us there were some obvious avenues for further theoretical research on semantics and verification:

1. The mathematical investigation of Hoare logic, since little was known about its proof theory and model theory. This was taken up in a series of papers in the 1980s at the hands of a few authors including Ed Clarke, Ernst-Rüdiger Olderog, Jan Bergstra and JVT.

2. The investigation of proof systems for communication and concurrency. This had started in earnest was being taken up by many researchers.

3. The extension of Hoare logic with abstract data types, modules, classes and other encapsulation and architectural ideas, and with errors and exceptions. This was taken up by several authors including us.

Jaco was primarily interested in denotational semantics, and he abandoned Hoare logic. He had followed the research on concurrency and was intrigued by the abstract process approach of Robin Milner, newly expounded in Milner [1980]. The process analysis of concurrency was exciting. Uninterpreted actions could be manipulated by operations and defined by fixed point formulae, but the method of making a calculus such as CCS appealed less. Jaco had also been taken with the work of Maurice Nivat on infinite trees and strings (Nivat [1979]). It contained a beautiful metric space treatment of the idea of specifying a set of strings or trees, and was related to Nivat and Arnold's metric space theory of nondeterministic programs. Jaco had already considered trees in De Bakker [1977]. Trees and metric spaces were central to Ruurd Kuiper's first work on semantics at the MC, written under Jaco's guidance (Kuiper [1981]).

The idea that a process is "like" an infinite tree was a starting point for applying metric techniques to the specification and semantics of concurrent systems. This seemed a new

and promising project: it combined simple abstract notions of process with established mathematical tools. Surely it would lead to a denotational semantics for concurrency and new insights into concurrent processes? It certainly did.

For many years after JIZ had left Amsterdam he visited the Netherlands every summer to work with Jaco at the CWI. Sometimes JIZ and JVT would meet at the CWI; sometimes he would travel on to Leeds. In addition, Jaco and JVT were guests of JIZ at Bar Ilan University and of the Weizmann Institute in Israel in the summer of 1981, following the eighth ICALP meeting at Akko that year. It was a relaxed, pleasant and productive visit. We continued working on our book, and Jaco and JIZ developed the ideas leading to the paper De Bakker and Zucker [1982], which helped to lay the foundation for the broad subject of concurrent process theory.

On subsequent visits by JIZ to the MC and CWI, this subject of topological process theory was developed much further by Jaco, JIZ and other colleagues, including Joost Kok, John-Jules Meyer, Ernst-Rüdiger Olderog, and Jan Rutten, resulting in the papers De Bakker and Zucker [1983 a,b], De Bakker, Meyer and Zucker [1983], De Bakker, Kok, Meyer, Olderog and Zucker [1985], De Bakker, Meyer, Olderog and Zucker [1988], and Rutten and Zucker [1992].

The origins of this subject is discussed van Breugel [1999] and its effects recorded in the books De Bakker and Rutten [1992] and De Bakker and De Vink [1996].

4. Concurrency, Domain Representability and Computability on Topological Data Types

The metric space theory of processes was influential in the development of a popular concrete approach to the study of computability on topological algebras, that of using domains to represent the algebras.

Jan Bergstra and Jan Willem Klop started working on process algebra after a lecture by Jaco in Utrecht in June 1982. They tackled the open problem he posed of solving unguarded recursion equations in the topological model of De Bakker and Zucker [1982]. Their solution was this: in the case of a finite set of atomic actions, they created the axiomatic system *Process Algebra* PA for processes. The theory PA had an initial algebra $A_{\square}$ and a system of projections A_n that modelled the execution of processes for n steps, for $n = 1, 2, \ldots$. These projections were also models of PA and the algebras formed an inverse sequence with inverse or projective limit A_∞, which was again a model of PA. They proved that all recursion equations have solutions in all the A_n and so in the A_∞. Since the A_∞ can be embedded in the De Bakker-Zucker model of processes, the problem was solved.

The theory PA was extended to the axiomatic system called *Algebra of Communicating Processes* ACP, and it was soon recognised that these axiomatisations were independent

and important new approaches to the theory of concurrent processes. Early on, in 1983, Jan Bergstra made one of his regular visits to JVT on holiday at his home in Ogmore-by-Sea in Wales, and he explained the theory in some detail. It was immediately exciting: mathematically, the theory combined the abstract conception of processes, the beautiful algebraic axioms of PA and ACP which were algebraic specifications of processes, and applications of mathematical techniques from the huge range available in universal algebra and topology. It was clear there could be many operators and axiom systems. Therefore, it seemed to be important that the subject follow general algebraic principles, i.e., it should use the basics of universal algebra, equational axiomatisations, homomorphisms, etc. This first encounter led to a few joint papers, including Bergstra and Tucker [1985] which attempted to give a clean account of the ideas in a strict algebraic style; added axioms (e.g., standard concurrency); proved a Milner expansion theorem for ACP; and attempted to use homomorphisms to model top-down design. JVT was sold on process algebra in one afternoon, between lunch and high tea, and has followed with great pleasure the development of the subject ever since.

In 1983, JVT was also continuing his longstanding collaboration with Viggo Stoltenberg-Hansen (now in Uppsala) on computable algebra. They wanted to develop a smooth and general treatment of computability in topological algebras, especially rings and fields, about which they knew a lot (Stoltenberg-Hansen and Tucker [1999a]). This problem had been encountered by JVT when working with John Cleave in 1974. Unknown to them, Klaus Weihrauch was also seeking a general method of analysing computations. In fact, Klaus considered a connection between metrics and partially ordered spaces in Weihrauch [1981], but abandoned this direction in favour of his theory of Baire space enumerations.

At their next meeting in Leeds, Viggo explained how he had been attracted to the study of local rings and their completions. There a local ring R with a maximal ideal M is used to create a inverse sequence of rings R/M^n and an inverse limit R_∞. This inverse limit was an uncountable ultrametric space and, in particular, a completion. What can be computed in local rings and their completions? We thought about computing with Cauchy sequences of rationals and with the Baire space of all functions on the natural numbers.

Viggo had also taken the brave step (for a mathematician) of offering a course on domain theory at Uppsala. There was a lot to learn about the connections between order and topology, fixed-point theory, and, of course, effective domains. But, by taking a select path into the subject, it did appear to be a generalisation of the key elements of higher type recursion theory.

On visits like this Viggo and JVT simply talked most of the time. Conversations about local rings and domains were interleaved many times each day and began to converge. They worked out how to build representations of complete local rings using total elements in algebraic domains. This allowed them to apply the theory of effective domains to analyse computability in the ring automatically.

At the same time JVT was keen to introduce a third new subject to their repertoire. He explained the algebraic theory of processes based on ACP with its use of equations, initial models and inverse limit model, which was another ultrametric space and completion. They saw that the methods for representing the completion of the local rings also applied to the inverse limit model of ACP and to many other sorts of completions, including infinite trees.

They wrote out the domain representation method for topological universal algebras, and formulated the general approach of analysing the effective content of topological algebras. They formulated the general problem of finding (computable) solutions of equations in topological algebras, inspired by the idea of giving equational specifications of (computable) processes. How were the domain theoretic and metric space based methods related for topological algebras in general, and for process algebras in particular?

Viggo and JVT embarked on a research programme to represent different sorts of topological algebras using domains. The theoretical starting point was this:

inverse limits of algebras were ultrametric algebras, and conversely,

and our domain representation methods worked beautifully for ultrametric algebras. In fact, the typical situation was exactly like that of the inverse limit model of processes: there was an initial algebra and a family of congruences $\equiv_n$ that led to a countably indexed inverse limit.

Viggo and JVT ultimately wrote up the work on the local rings and the general approach in Stoltenberg-Hansen and Tucker [1985] and circulated it widely as the first preprint of the newly launched *Centre for Theoretical Computer Science* at Leeds University. It was later published as Stoltenberg-Hansen and Tucker [1988]. They also wrote up their investigations in papers on ultrametric universal algebras and the solution of finite and infinite systems of equations using domains and fixed points (Stoltenberg-Hansen and Tucker [1991,1993]). The idea was to show that

(i) there was an equivalence of approaches to concurrency based on process algebras; process calculi; metric space methods and, indeed, domain theory; and
(ii) these equivalences could be derived from the principles of universal algebra.

They gave applications to process algebra and infinite synchronous concurrent algorithms. Domain representability, and results such as the fact that the Banach Contraction Mapping Theorem was derivable from the Least Fixed Point Theorem via a domain representation, were included in Viggo's book on domains (Stoltenberg-Hansen, Lindstrom and Griffor [1994]).

Now, although the real numbers were invaluable in exploring the abstract idea of completion they do not themselves an inverse limit. Viggo and JVT broadened the general domain representation method and showed that the functions on the real numbers that were effective in a natural domain representation of the real numbers were the computable functions of Grzegorczyk and Lacombe defined in the 1950s (see below). This result was published later in Stoltenberg-Hansen and Tucker [1995].

The domain representation method for topological universal algebras is now widely known and used when looking at computation in analysis and its applications (see Edelat [1997]). It is interesting to note the power of the oral tradition in the origins of this domain approach to computable topological data types. The general forms of domain representations for universal algebras were revealed through conversations on concurrency theories between Jaco and JIZ, Jan and Jan Willem, Jan and JVT, and JVT and Viggo.

5. A Taste of our Current Research

Notions of computability on the natural numbers and strings have long been known to agree. Since 1936, many models of computation have been developed and proved to be equivalent. The theory of computable functions on natural numbers is stable. It is largely independent of data representations (e.g., computability on binary numbers is equivalent to that on decimal numbers) and programming constructs. It possesses many elegant and efficient models with which to work on applications. And, thanks to the theory of the arithmetic hierarchy, the connection between computation and specification in first order logic is clear and beautiful. It is truly worthy of its name Classical Computability Theory.

Our research has sought to analyse the concepts and mathematical ideas in classical computability by examining them in the more general setting of an arbitrary many sorted algebra and applying them in particular cases, such as algebras of real numbers. Currently, computing with topological algebras is our main occupation. Here is a taste emphasising the real numbers.

5.1 Computability on topological algebras

There are many approaches to defining computability on topological and metric algebras that are technically different and have different agendas. Until recently, there was no sign of a stable theory, only a rather confusing range things to do and ways to do them. The rather basic question,

What are the computable functions on topological algebras?

did not have a general, widely agreed and understood answer, even for the real numbers; an elegant solution to this fundamental problem seemed even further away.

The computability theories in the literature may be divided usefully into two kinds:

1. *abstract computability theories*, in which computations are independent of data representations; and
2. *concrete computability theories,* in which computations depend on some chosen data representations.

Abstract computation theories are based on "programs" that use the basic operations and tests of the algebra only. They may have different control or specification constructs, such as **goto**, recursion or parallelism. The programming languages that are used in semantical studies, like Jaco's book, are all examples of abstract models of computation. One simply has to ask the question: What does this language compute?

As we showed in Chapter 4 of Tucker and Zucker [1988], many abstract models of computation of different kinds have been defined and shown to be equivalent. Subsequently, we analysed several more and found a family of abstract models better suited to specification rather than computation. Thus, the theory of abstract computation is quite stable. For a recent comprehensive introduction to abstract computation, including a new survey of its origins in the 1950s and principal literature, see our Tucker and Zucker [2000]. There we used **while-array** programs over these algebras, the primary mathematical model of imperative programming. Abstract computation theories are designed to compute on *all* many sorted algebras and so can be used to develop computability theories for particular algebras such as *rings and fields of real numbers* (see, e.g., Blum et al [1998]) and *topological and metric algebras* (Tucker and Zucker [1999a]).

Concrete computation theories are designed to analyse computability in terms of classical recursion theory on natural numbers via chosen representations of data and spaces. Some general approaches are:

Effective metric spaces	Moschovakis [1964];
Computable sequence structures for Banach spaces	Pour El and Richards [1989],
Type 2 enumerations	Weihrauch [2000],
Algebraic domain representations	Stoltenberg-Hansen and Tucker [1988, 1995],
Continuous domain representations	Edalat [1995],
Numbered spaces	Spreen [1998].

The equivalence of most of these concrete approaches is proved for certain topological algebras in Stoltenberg-Hansen and Tucker [1999b]. Whilst the study of concrete computability is not so well understood, it seems to be stable on the real numbers. The general concrete models have all been shown to define the *Grzegorczyk-Lacombe (GL) computable functions* on the real numbers, first formulated in the 1950s and the basis of early studies in computable analysis.

It is the connection between the abstract and concrete theories that has been a problem. Concrete models can compute a lot more or a lot less, depending on the selection of the algebraic operations. Only recently, it has been shown that, surprisingly, notions of

(i) continuity and partiality,
(ii) limit processes and approximation, and
(iii) nondeterminism and multivaluedness,

are necessary to bridge the gap between them for general classes of metric algebras: see Brattka [1996, 1999] and Tucker and Zucker [1999, 200?]). Let us amplify these points in turn.

Ad (i): Abstract computability theories are dependent entirely on the operations and tests of the algebra. Since the choice of operations is unlimited, it is possible to choose operations that are not concretely computable. A typical mistake is to allow tests like

$$= \text{ or } <$$

as total functions in the algebra. These operations are discontinuous. It follows from conceptual analysis and different versions of Ceitin's Theorem that concrete computable functions are always continuous on the real numbers. Hence, such basic tests can only be introduced as *partial operations*.

Ad (ii): Abstract models compute outputs that are contained in the subalgebra generated by the input. However, in a metric algebra they can computably approximate data outside this subalgebra. For example, on the field of real numbers, $\sqrt{x}$ cannot be computable by **while-array** programs because $\sqrt{x}$ is not in the subfield $\langle 2 \rangle = Q$, the rationals. It is, however, computably approximable.

Ad (iii): Concrete models can compute single-valued selection functions that are continuous in the topology of the *representations* of the real numbers, but *not* in the topology of the real numbers themselves. Abstract models cannot compute such single-valued functions.

In our Tucker and Zucker [200?a] we extended the language of **while-array** programs with the *non-deterministic assignment statement*

$$x := \textbf{choose } z\text{: } \textbf{nat} \mid b(z, y).$$

where b is a Boolean-valued procedure. This produces *countable* nondeterministic choice. This construct escaped Jaco's book but was caught in Apt and Olderog [1991]. We first encountered the idea in our old office in the MC, where Ralph Back was looking at nondeterministic assignments, in the early days of his extensive theory of program refinement (Back [1980a, 1980b, 1998]).

Let us describe some recent extensions to our recent work in this area Tucker and Zucker [1999, 2002, 200?a]

In computing with the real numbers we often use the assumption of global uniform continuity, as it simplifies considerably technical definitions of the computability of functions on spaces. In metric algebras, compactness implies that continuous functions are uniformly continuous. We weaken this assumption by considering the broader class of functions that are uniformly continuous in pieces, by localising the uniformities, necessary for computability, using families of open sets

$$(U, V_0, V_1, V_2 \ \ldots) \quad \text{such that} \quad U = \bigcup V_i$$

called *open exhaustions* This leads to the notion of *effective local uniform continuity*. Exhaustions are an obvious and standard way of extending computability notions via localisation. The resulting theorems have applications to the study of functions on *all* of n-space $\mathbf{R}^n \to \mathbf{R}^m$, rather than on a compact cube $[a, b]^n \to \mathbf{R}^m$.

We extend our various general notions to the localised notion using exhaustions. We can show that for connected exhaustions, on certain algebras, **while** approximation, **while-array** approximation, and polynomial approximation are equivalent. On choosing a *particular* total algebra of real numbers A_{real}, these three notions can be shown to coincide with standard GL-computability on **R** (c.f. Tucker and Zucker [1999]). Next we extend our main bridging theorem in Tucker and Zucker [200?a] to the localised case. We can show that for effectively locally uniformly continuous functions, and a wide class of metric algebras, approximation by **while-array** programs with countable choice is equivalent to a simple general notion of effectively trackable in a concrete Moschovakis-like computational model. Combining these results in the special case of real numbers we obtain:

Theorem *Let* f: $\mathbf{R}^n \to \mathbf{R}^m$ *be a function that is effectively locally uniformly continuous on an exhaustion. Then the following are equivalent:*

1. f *is GL computable on* **R**.
2. f *is "effectively trackable" on* **R**.
3. f *is locally polynomial approximable on* A_{real} .
4. f *is locally* **while array** *approximable on* A_{real} .
5. f *is locally* **while array** *with countable choice approximable on a partial algebra* B_{real}

This, together with other equivalences between concrete models and GL-computability, gives us a stable foundation for the idea of a locally computable function based on exhaustions. Next we consider the specification of these functions.

5.2 Specifications

In the theory of data, abstract data types are modelled by many sorted algebras and homomorphisms, and are specified axiomatically by equations and conditional equations. Most of the theory has been developed for data types that are discrete and countable, since they are the data types for which exact digital computation is possible. Jan Bergstra and JVT began a investigation that revealed surprising equivalences between computability, algebraic specification methods, and term rewriting; see, for instance Bergstra and Tucker [1983,1987,1995], Meseguer, Moss and Goguen [1992], Khoussainov [1998]; and surveys such as Meseguer and Goguen [1985] and Stoltenberg-Hansen and Tucker [1995].

Now, data types containing continuous data can be modelled by *topological* many sorted algebras and continuous homomorphisms, or - to take a more restricted class, closer to examples - by many sorted topological algebras that are also *metric spaces*. There are two questions we need to answer:

What methods exist to axiomatically specify functions on topological algebras?

and

Can all computable functions be specified?

The theory of topological data types is in its infancy. As we have seen there are *many* approaches to computability theory on general and specific spaces, and *few* approaches to specification theory. In Tucker and Zucker [2002] we have studied these questions.

Algebraic specification methods characterise functions as the solutions of systems of algebraic formulae that are unique in some sense. By algebraic formulae, we mean equations

$$t(X) = t'(X)$$

or conditional equations

$$t_1(X) = t_1'(X) \wedge \ldots \wedge t_k(X) = t'_k(X) \Rightarrow t(X) = t'(X),$$

or other formulae, based on these, and enjoying some algebraic properties or customised to the particular algebraic context. For example, in working in metric algebras, we have as standard the sort of real numbers, so we will adapt the formulae to include (i) inequalities between reals and (ii) localisation via exhaustions. Taking (i) and (ii) together, we form specifications using *localised conditional equations and inequalities*. We can then prove that algebraic specifications can specify all computable functions on metric algebras. We say that the algebraic specification $(\Omega^+, C(n))$ is a *universal specification* if as n varies over the natural numbers the specification defines all the computably approximable functions over all metric Ω algebras. In the case of the real numbers we can derive:

Theorem *There is a finite universal specification* $(\Omega^+, C(\mathbf{n}))$*, consisting of conditional equations and inequalities over* Ω^+*, that defines all the locally GL computable functions on* $\mathbf{R}$.

From this it is easy to prove technical results with the informal meaning:

Theorem *If a deterministic finite dimensional dynamical system has a model that can be simulated to any degree of accuracy by an algorithm then there exists an algebraic specification that uniquely defines that algorithmic model. Indeed, for each n, there is an algebraic specification that uniformly captures all algorithmic models with n-dimensional state spaces.*

These results will appear in Tucker and Zucker [200?b].

6. Conclusion

Semantics, Verification and Process are some of the Big Ideas to emerge in Computer Science. Jaco has played an important pioneering role in the development of the modern

field of Semantics of Computation from its earliest days, and has made big contributions to fields of Verification and Processes. His work will be studied with profit in the years to come.

Data is another Big Idea. The idea of working out a computability theory for data in general, and applying it, seemed a basic task in 1979 and is more so even now. Jaco's ideas and techniques in semantics shaped our joint research and, of course, our research with others. Topological data types are fundamental in modelling physical systems and are the characteristic data types of analogue processing. What is the relationship between computations with continuous data and with discrete data? The semantical theory is developing, but little is known.

Nor are the books over! We intend to write a comprehensive graduate textbook and monograph on abstract computability theory. Some of basic ideas and approaches of both Jaco's and our book have found their way in to the undergraduate textbook Tucker and Stephenson [200?].

The Netherlands is one of the most important countries in the world for research in Computer Science. Amsterdam is the most important city in the world for the theoretical research on programming and specification. Jaco is the primary architect of this city's outstanding reputation though his research, leadership and organisation at the MC and CWI for the past 38 years.

In our own case we worked with Jaco in our formative years as computer scientists and met with him many times in the years that followed. Through regular visits to Amsterdam and the hospitality of Jaco and Angeline, at home and on excursions, we have become firm and loyal friends. We owe him a great deal. There are so many scientists who owe the key steps in their progress to Jaco.

7. References

K R Apt, N Francez, W-P de Roever, A proof system for communicating sequential processes, *ACM Transactions on Programming Languages and Systems*, 2 (1980) 359-385.

K R Apt and E-R Olderog, Verification of sequential and concurrent programs, Springer Verlag, New York, 1991.

P R J Asveld and J V Tucker, Complexity theory and the operational structure of algebraic programming systems, *Acta Informatica,* 17 (1982), 451-476.

C A R Hoare, An axiomatic basis for computer programming, *Communications of the ACM*, 12 (1969), 576-580, 583.

R J R Back, Semantics of unbounded nondeterminism, in J W de Bakker and J van Leeuwen (eds.) *Automata, Languages and Programming, Seventh Colloquium, Noordwijkerhout, 1980,* Springer Lecture Notes in Computer Science 81, Springer Verlag, Berlin, 1980, 51–63. a

R J R Back, *Correctness Preserving Program Refinements: Proof Theory and Applications*, Mathematical Centre Tracts 131, Mathematical Centre, Amsterdam, 1980. b

R J R Back and J von Wright, *Refinement Calculus: A Systematic Introduction*, Graduate Texts in Computer Science, Springer Verlag, 1998.

J W de Bakker, Axiomatics of simple assignment statements, Report 94, Mathematish Centrum, Amsterdam, 1968.

J W de Bakker, *Recursive Procedures*, Tract 24, Mathematish Centrum, Amsterdam, 1973.

J W de Bakker, Semantics of infinite processes using generalised trees, in J Gruska (ed.), *Mathematical Foundations of Computer Science, 6th Colloquium*, Lecture Notes in Computer Science, 53, Springer, Berlin, 1977, 240-252.

J W de Bakker, *The Mathematical Theory of Program Correctness*, Prentice Hall International, London,1980.

J W de Bakker and J J M M Rutten, *Ten years of concurrency semantics*, World Scientific, Singapore Amsterdam, 1992.

J W de Bakker and E de Vink, *Control flow semantics*, MIT Press, 1996.

J W de Bakker and J I Zucker, Processes and the denotational semantics of concurrency, *Information and Control*, 54 (1982), 70-120. Reprinted, with errata, in De Bakker and Rutten [1992], 28-80.

J W de Bakker and J I Zucker, Compactness in semantics for merge and fair merge, in E Clarke and D Kozen (ed.), *Logics of Programs. Workshop, Carnegie-Mellon University, Pittsburgh, PA, June 1983*, Lecture Notes in Computer Science 164, Springer Verlag, 1983, 18-33. a

J W de Bakker and J I Zucker, Processes and a fair semantics for the ADA rendez-vous, in J Diaz (ed.), *Automata, Languages and Programming: Proceedings of the 10th International Colloquium on Logic, Automata and Programming, Barcelona, Spain, July 1983*, Lecture Notes in Computer Science 154, 1983, Springer Verlag. b

J W de Bakker, J-J Ch Meyer and J I Zucker, On infinite computations in denotational semantics, *Theoretical Computer Science*, 26 (1983), 53-82.

J W de Bakker, J N Kok, J J Ch Meyer, E R Olderog and J I Zucker, Contrasting themes in the semantics of imperative concurrency, in J W de Bakker et al. (eds.) *Current Trends in Concurrency (Overviews and Tutorials): Proceedings of the ESPRIT/LPC Advanced School in Concurrency, Noordwijkerhout, The Netherlands, June 1985*, Lecture Notes in Computer Science 224, Springer Verlag, (1985), 51-121.

J W de Bakker, J-J Ch Meyer, E R Olderog and J I Zucker, Transition systems, metric spaces, and ready sets in the semantics of uniform concurrency, *Journal of Computer and System Sciences*, 54 (1988), 158-224.

J A Bergstra, J Tiuryn and J V Tucker, Correctness theories and program equivalence, Stichting Mathematisch Centrum. Informatica, IW 119/79, Amsterdam 1979, 31 pp.

J A Bergstra and J V Tucker, On the adequacy of finite equational methods for data type specification, *ACM-SIGPLAN Notices*, 14.11 (1979) 13-18.

J A Bergstra and J V Tucker, Algebraic specifications of computable and semi-computable data structures, Stichting Mathematisch Centrum Informatica, IW 115/79, Amsterdam 1979, 24pp.

J A Bergstra and J V Tucker, A characterisation of computable data types by means of a finite, equational specification method, Stichting Mathematisch Centrum Informatica, IW 124/79, Amsterdam, 1979, 23 p.

J A Bergstra and J V Tucker, A natural data type with a finite equational final semantics specification but no effective equational initial semantics specification, *Bulletin of the European Association for Theoretical Computer Science,* 11 (1980) 23-33.

J A Bergstra and J V Tucker, A characterisation of computable data types by means of a finite equational specification method, in J W de Bakker and J van Leeuwen (eds.) *Automata, Languages and Programming, Seventh Colloquium, Noordwijkerhout, 1980,* Springer Lecture Notes in Computer Science 81, Springer Verlag, Berlin, 1980, pp. 76-90.

J A Bergstra and J V Tucker, The completeness of the algebraic specification methods for data types, *Information and Control,* 54 (1982) 186-200.

J A Bergstra and J V Tucker, Initial and final algebra semantics for data type specifications: two characterisation theorems, *SIAM Journal on Computing*, 12 (1983) 366-387.

J A Bergstra and J V Tucker, Top-down design and the algebra of communicating processes, *Science of Computer Programming*, 5 (1985) 171-199.

J A Bergstra and J V Tucker, Algebraic specifications of computable and semi-computable data types, *Theoretical Computer Science,* 50 (1987) 137-181.

J A Bergstra and J V Tucker, Equational specifications, complete term rewriting systems, and computable and semicomputable algebras, *Journal of ACM*, 42 (1995) 1194-1230.

V Brattka, Recursive characterisation of computable real-valued functions and relations, *Theoretical Computer Science*, 162 (1996), 45-77.

V Brattka, *Recursive and comptable operations over topological structures*, PhD thesis, Fachbereich Informatik, FernUniversität Hagen, 1999.

J Blanck, V Stoltenberg-Hansen and J V Tucker, Streams, stream transformers and domain representations, in B Möller and J V Tucker (eds.), *Prospects for hardware foundations*, Springer Lecture Notes in Computer Science, Vol 1546, 1998, 27-68.

J Blanck, V Stoltenberg-Hansen and J V Tucker, Domain representations of partial functions, with applications to spatial objects and constructive volume geometry, *Theoretical Computer Science*, 284 (2002), 207-240.

L Blum, F Cucker, M Shub and S Smale, *Complexity and real computation*, Springer Verlag, New York, 1998.

F van Breugel, De Bakker-Zucker Processes Revisited (Dedicated to Jaco de Bakker on the occasion of his 60th birthday.) Report CS-1999-05, York University, Toronto, November 1999. To appear in *Information and Computation.*

A Edalat, Domains for computation in mathematics, physics and exact real arithmetic, *Bulletin of Symbolic Logic*, 3 (1997) 401-452.

J E Fenstad, *Generalised recursion theory*, Springer Verlag, Berlin, 1980.

J E Fenstad, Computability theory: structure or algorithms, in W Sieg, R Somer, C Talcott (eds.), *Reflections on the foundations of mathematics: Essays in honour of Solomon Feferman*, Lecture Notes in Logic, volume 15, Association for Symbolic Logic, 2002, 188-213.

C B Jones The Search for Tractable Ways of Reasoning about Programs, Manchester University, UMCS-92-4-4, 1994.

R Kuiper, An operational semantics for bounded nondeterminism equivalent to a denotational one, J W de Bakker and J C Van Vliet (eds.), *Algorithmic languages*, North-Holland, 1981, 373-398.

R W Floyd, Assigning meanings to programs, *Proceedings AMS Symposium in Applied Mathematics* 19 (1967) 19-31.

N Francez, *Program verification*, Addison Wesley, 1991.

J A Goguen, J W Thatcher, E G Wagner and J B Wright, Initial algebra semantics and continuous algebras, *Journal ACM* 24 (1977), 68-95.

B Khoussainov, Randomness, computability, and algebraic specifications, *Annals of Pure and Applied Logic*, 91 (1998) 1-15.

C Kreitz and K Weihrauch, Theory of representations, *Theoretical Computer Science* 38 (1985) 35-53.

J Moldestad, V Stoltenberg-Hansen and J V Tucker, Finite algorithmic procedures and inductive definability, *Mathematica Scandinavica*, 46 (1980) 62-76. a

J Moldestad, V Stoltenberg-Hansen and J V Tucker, Finite algorithmic procedures and computation theories, *Mathematica Scandinavica*, 46 (1980) 77-94. b

K Meinke and J V Tucker, Universal algebra, in S Abramsky, D Gabbay and T Maibaum (eds.) *Handbook of Logic in Computer Science. Volume I: Mathematical Structures,* Oxford University Press, 1992, pp.189-411.

J Meseguer and J A Goguen, Initiality, induction and computability, in M Nivat and J Reynolds (eds.), *Algebraic methods in semantics*, Cambridge University Press, 1985.

J Meseguer, L Moss and J A Goguen, Final algebra, cosemicomputable algebras and degrees of unsolvability, *Theoretical Computer Science*, 100 (1992) 267-302.

M Nivat, Infinite words, infinite trees, infinite computations, in J W de Bakker and J van Leeuwen (ed.), *Foundations of Computer Science III, Part 2: Languages, Logic, Semantics*, Mathematical Centre Tracts vol.109, Mathematical Centre, Amsterdam, 1979, 3-52.

J J M M Rutten and J I Zucker, A semantic approach to fairness, *Fundamenta Informaticae*, 16(1992), 1-38.

D Spreen, On effective topological spaces, *Journal of Symbolic Logic* 63 (1998) 185 – 221.

D Spreen, Representations versus numberings: On the relationship of two computability notions, *Theoretical Computer Science* 263 (2001), 473-499.

D Spreen and H Schulz, On the equivalence of some approaches to computability on the real line, in Keimel, K et al., (eds.) *Domains and Processes*, Proc. 1st Intern. Symp. on Domain Theory, Shanghai, China, 1999, Kluwer, Boston, 2001, 67-101.

W-P de Roever, F de Boer, U Hannemann, J Hooman, Y Lakhnech, M Poel, and J Zwiers, *Concurrency Verification: Introduction to Compositional and Noncompositional Methods*, Cambridge University Press, 2001.

V Stoltenberg-Hansen, I Lindstrom and E R Griffor, *Mathematical Theory of Domains*, Cambridge University Press, 1994.

V Stoltenberg-Hansen and J V Tucker, Computing roots of unity in fields, *Bulletin of the London Mathematical Society*, 12 (1980) 463-471.

V Stoltenberg-Hansen and J V Tucker, Complete local rings as domains, *Journal of Symbolic Logic*, 53 (1988) 603-624.

V Stoltenberg-Hansen and J V Tucker, Algebraic equations and fixed-point equations in inverse limits, *Theoretical Computer Science*, 87 (1991) 1-24.

V Stoltenberg-Hansen and J V Tucker, Infinite systems of equations over inverse limits and infinite synchronous concurrent algorithms in J W de Bakker, G Rozenberg, and W P de Roever (eds.) *Semantics - Foundations and applications*, Springer Lecture Notes in Computer Science 666, Springer Verlag, 1993, 531-562.

V Stoltenberg-Hansen and J V Tucker, Effective algebras, in S Abramsky, D Gabbay and T Maibaum (eds.) *Handbook of Logic in Computer Science. Volume IV: Semantic Modelling* , Oxford University Press, 1995, pp.357-526.

V Stoltenberg-Hansen and J V Tucker, Computable rings and fields, in E Griffor (ed.), *Handbook of Computability Theory*, Elsevier, 1999, pp.363-447. a

V Stoltenberg-Hansen and J V Tucker, Concrete models of computation for topological algebras, *Theoretical Computer Science*, 219 (1999) 347-378. b

J V Tucker, Computing in algebraic systems, in F R Drake and S S Wainer (eds.) *Recursion Theory, its Generalisations and Applications*, London Mathematical Society Lecture Note Series 45, Cambridge University Press, Cambridge, 1980, pp. 215-235.

J V Tucker, Applications of computability theory over abstract data types, in J W Klop (ed.) *J W de Bakker: 25 Jaar Semantiek. Liber Amicorum,* CWI Amsterdam, 1989, 421-432.

J V Tucker, Theory of computation and specification over abstract data types and its applications, in F L Bauer (ed.), Proceedings of NATO Summer School 1989 at Marktoberdorf, in *Logic, algebra and computation***,** Springer, 1991, pp.1-39.

J V Tucker and K Stephenson, *Data, syntax and semantics*, in preparation.

J V Tucker and J I Zucker, *Program correctness over abstract data types with error-state semantics*, CWI Tract 6, North-Holland, Amsterdam, 1988.

J V Tucker and J I Zucker, Horn programs and semicomputable relations on abstract structures, in G Ausiello, M Dezani-Ciancaglini, S Ronchi Della Rocca (eds.) *Automata, Languages and Programming, Sixteenth Colloquium, Stresa, 1989,* Springer Lecture Notes in Computer Science 372, Springer Verlag, Berlin, 1989, pp.745-760.

J V Tucker and J I Zucker, Toward a general theory of computation and specification over abstract data types, in S G Akl, F Fiala, and W W Koczkodaj (eds.), *Advances in computation and information, May 1990,* Canadian Scholars Press, 1990, pp.101-102. Also book republished in Springer Lecture Notes in Computer Science 468, Springer Verlag, Berlin, 1990, 129-133.

J V Tucker and J I Zucker, Examples of semicomputable sets of real and complex numbers, in J P Myers Jr and M J O'Donnell (eds.), *Constructivity in computer science***,** Springer Lecture Notes in Computer Science 613, Berlin, pp.179-198.

J V Tucker and J I Zucker, Projections of semicomputable relations on abstract data types, *International J of Foundations of Computer Science* 2 (1991) 267-296.

J V Tucker and J I Zucker, Deterministic and nondeterministic computation, and Horn programs, on abstract data types, *Journal of Logic Programming*, 13 (1992) 23-55.

J V Tucker and J I Zucker, Theory of computation over stream algebras, and its applications, in I M Havel and V Koubek (eds.) *Mathematical Foundations of Computer Science 1992, 17th International Symposium, Prague*, Springer Lecture Notes in Computer Science 629, Berlin, 62-80.

J V Tucker and J I Zucker, Provable computable selection functions on abstract structure*s,* in P Aczel, H Simmons and S S Wainer (eds.) *Proof theory*, Cambridge University Press, 1993, 277-306.

J V Tucker and J I Zucker, Computable functions on stream algebra*s,* in H Schwichtenberg (ed.), Proceedings of NATO Advanced Study Institute, International Summer School 1993 at Marktoberdorf, in *Proof and Computation***,** Springer, 1994, 341-382.

J V Tucker and J I Zucker, Computation by while programs on topological partial algebras, *Theoretical Computer Science*, 219 (1999) 379-421.

J V Tucker and J I Zucker, Computable functions and semicomputable sets on many sorted algebras, in S Abramsky, D Gabbay and T Maibaum (eds.) *Handbook of Logic for Computer Science Volume V*, Oxford University Press, 2000, 317-523.

J V Tucker and J I Zucker, Infinitary initial algebraic specifications for stream algebras, in W Sieg, R Somer, C Talcott (eds.), *Reflections on the foundations of mathematics: Essays in honour of Solomon Feferman*, Lecture Notes in Logic, volume 15, Association for Symbolic Logic, 2002, 234-253.

J V Tucker and J I Zucker, Abstract computability and algebraic specification, *ACM Transactions on Computational Logic,* 3 (2002) 279-333.

J V Tucker and J I Zucker, Abstract versus concrete models of computation on partial metric algebras, *ACM Transactions on Computational Logic,* accepted, 200?a.

J V Tucker and J I Zucker, Computable total functions, algebraic specification and dynamical systems, submitted, 200?b.

S S Wainer, J V Tucker and J I Zucker, Provably computable functions on abstract data types, in M Patterson (ed.) *Automata, Languages and Programming, Seventeenth Colloquium, Coventry, 1990,* Springer Lecture Notes in Computer Science 443, Springer Verlag, 1990, pp.660-673.

K W Weihrauch, *Computable analysis*, Springer Verlag, 2000.

A van Wijngaarden, Numerical analysis as an independent science, *BIT* 6 (1966), 66-81.