Logic and Machines: Turing Tradition at the Logic School of Münster

Egon Börger[1] and Rainer Glaschick[2]

[1] boerger@di.unipi.it
[2] rainer@glaschick-pb.de

Abstract. We describe the influence Turing's early work on the *Entscheidungsproblem* had in the *Schule von Münster* where it triggered the investigation of intimate links between logic and computation years before they became a main research theme in the science of computing. We report in particular on the work in the *Turingraum* where illustrative physical models of some outstanding (in particular universal) Turing and related machines were built some of which found their way into *teaching algorithmic thinking* in primary and secondary schools. We also point out how the investigation of various logical models of computation, initiated in Münster in the middle of the 1960s, led in the 1990s in a natural way to the practical but mathematically rigorous *Abstract State Machines Method* for a well-documented stepwise development of executable code by constructing logical models which successively introduce the necessary details to implement an initial rigorous requirements model.

1 Introduction

The story of this paper starts with two events in 1936 which involved Alan Turing and Heinrich Scholz.

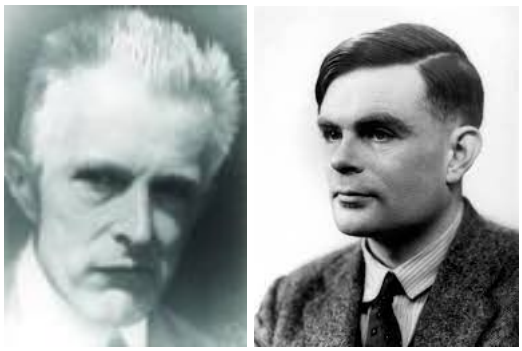


Fig. 1. Heinrich Scholz and Alan Turing

A. In 1936 the Logic School of Münster (*Schule von Münster* [174, p.15], see also [220]) became institutionally visible, namely when **Heinrich Scholz**,

working since 1928 as professor of Philosophy at the University of Münster with a growing focus on mathematical logic and foundational issues, succeeded to establish as an independent part of the Philosophisches Seminar a "Logistische Abteilung" which eventually in 1950 became the *Institut für mathematische Logik und Grundlagenforschung* (Institute of Mathematical Logic and Foundational Research) at the mathematics department of the university.[1]

The group of 10 doctoral students who worked with Scholz between 1930 and 1953 in Münster distinguished itself by strong *interdisciplinary interests* and an *openness of mind towards applications of logic* in various fields, attracted and stimulated by the extraordinary wide range of interests and knowledge of its founder. These properties have been honored by the two followers as head of the school and director of the institute, namely **Hans Hermes** (1953-1966)[2] and **Dieter Rödding** (1966-1984, see [32]), as well as by Scholz' last student **Gisbert Hasenjaeger** who worked with Scholz as student, assistant and Dozent from 1945 to 1962 when he moved from Münster to the University of Bonn to establish there yet another successful Department of Logic and Basic Research which he directed until his retirement in 1984 (see [174, pp.261–262], [265] and https://rclab.de/hasenjaeger/publikationen_von_gisbert_hasenjaeger).

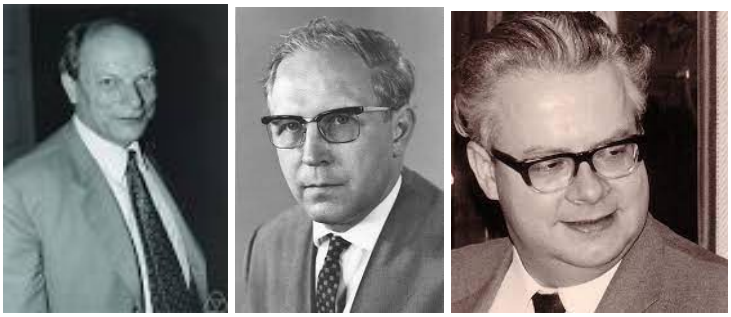


Fig. 2. Hans Hermes, Gisbert Hasenjaeger and Dieter Rödding

In this context also two later associates should be mentioned who contributed to the work of the school in set theory, proof theory and recursion theory resp. in model theory, namely **Wilhelm Ackermann**, who joined the group as honorary professor from 1953 to his death in 1962,[3] and **Wolfram Schwabhäuser**, who submitted his doctoral thesis in 1960 in Berlin [269], under the supervi-

[1] For the detailed historical development of the institute see [14].

[2] Hermes moved in 1966 to the University of Freiburg where he founded (and directed until his retirement in 1977) a new group of logic, sometimes referred to as the *Freiburger Logikschule*, which too excelled by a wide spectrum of scientific interests including applications of logic to computing. See [266].

[3] In computer science Ackermann is known mainly due to the scheme he invented to define recursive functions which 'grow faster than any primitive recursive function'

sion of Scholz' former student Karl Schröter, and his Habilitation in 1965 at the University of Münster, worked in 1965/66 with Tarski at the University of Berkeley, joined in 1966 Hasenjaeger's logic group in Bonn and since 1973 until his early death in 1985 worked as professor for theoretical computer science at the University of Stuttgart.

Fig. 3. Wilhelm Ackermann and Wolfram Schwabhäuser

B. In 1936 **Alan Turing** wrote (and published in [17]) his epochal paper where he

- defined a class of idealized machines, nowadays called *Turing machines* (TMs) which he argues to model *computability*, i.e. to capture every calculation, performed by a human using pencil and paper, 'which could naturally be regarded as computable' [17, p.230] (showing also that it is equivalent to Church's notion of *effective calculability* defined in [5]),
- proved that however some simple machine properties (e.g. whether a given machine once started will ever print some given symbol) cannot be computed by any Turing machine,
- derived from the preceding impossibility result a negative answer to Hilbert's question about the *Entscheidungsproblem* (decision problem of first order logic), as did Church in [4] using his concept of effective calculability,
- showed that there are so-called *universal* Turing machines, i.e. single (programmable) machines which can do ('simulate' in a precise mathematical sense) the work of any other Turing machine.

C. The two events are intimately related [127]: Scholz and his school promptly recognised the epochal significance of Turing's solution to Hilbert's fundamental *Entscheidungsproblem*, which at the time was considered to be 'the main problem of mathematical logic' [73, Ch.11,12]; they understood that there

and therefore are not primitive recursive [6]; see [150] for an account of his deep contributions to logic.

is an intimate connection between abstract machines and logic languages to describe the machine behaviour, between logical expressivity and computational machine power.

In fact, Scholz presented Turing's freshly published work in a seminar talk at the just founded logic institute, a seminar that has been called the world's first seminar on computer science [9].[4] Hans Hermes who at the time was a doctoral student at the institute[5] published right away in 1937 a first paper [142] where he builds upon Turing machines. When in 1947, after the war and after the Habilitation in mathematics at the University of Bonn, Hermes returned as Dozent to Münster he started regular lectures on the subject (since 1949) and in 1952 and 1954 wrote two other papers [144,145] where he adapted in a natural way Turing's definition of computation by a finiteness condition that better fits computing decision problems (instead of numbers). This led him directly to the halting problem. The definitions eventually went into his computability book of 1961 [147]. Together with Martin Davis' computability book of 1958 [211] these two early Turing machine based textbooks (which both went through various editions and had each a follower [186,212]) formed the mind of generations of logic and computer science students concerning the basic concepts of computability and undecidability.

1.1 Content of the paper

In this paper we illustrate how the early encounter between Turing and the Logic School of Münster triggered there for half a century (until Rödding's early death in 1984) a thorough investigation of the relations between logic and (in particular machine-based) concepts and methods of computation, years before they became a major theme for the new science of computing.[6]

- In Sect. 2 we report on the early Turing reception in Münster.
- In Sect. 3 we report on the work where the School of Münster enriched the classical recursion theory by machine-based classifications of recursive functions and by investigating links between machine computations and logical

[4] Note that *ibidem* (see also https://en.wikipedia.org/wiki/Heinrich_Scholz) it is also reported that Scholz was the only scientist worldwide outside the inner Cambridge circle who asked Turing for a reprint of his decision problem paper [17], by the way later also of [16]. How much Turing appreciated this interest can be seen from https://ivv5hpp.uni-muenster.de/u/cl/. The authors of [252] found letters of 1952/3 where Scholz tried (without success) to arrange for Turing a visit to Münster.

[5] His doctoral dissertation on "Eine Axiomatisierung der allgemeinen Mechanik" [143] was submitted there in 1938.

[6] In this paper we do not mention further the well-known research activities of members of the school in traditional areas of logic, in particular set theory, model theory, and proof theory. Nor do we mention further the numerous and intensive contacts the members of the institute maintained with their colleagues in other logic centers in Europe and the US.

and algorithmic decision problems, work which contributed to the emerging machine-based complexity theory in theoretical computer science.

- In Sect. 4 we report on Hasenjaeger's and Rödding's work to 'materialize' in hardware illustrative models of some outstanding computing machines, built in what in the institute was called the *Turingraum*. We explain also the effect this practical work had on the theoretical development of new models of computation. In Sect. 9 we provide the underlying technical details.
- In Sect. 5 we report on Rödding's and his students' work on modular decompositions of different kinds of automata which found various applications in other disciplines, among them an outstanding and rather intuitive dynamic way to teach fundamental mathematical and algorithmic concepts to students of elementary and high schools (see Sect. 6).
- In Sect. 7 we point to some alternative concepts of computation which have been developed in the School of Münster since the 1960s. They were about computing with non-numerical objects and structures, like terms, trees, graphs (networks of automata), topological structures, etc. Since the late 1980s this theme found a renewed interest in theoretical computer science and eventually led to the discovery of Abstract State Machines (ASMs). We shortly explain how this concept, in addition to its theoretical interest for logic and complexity theory, led in the 1990s to the development (far away from Münster) of the *ASM Method* which enhances the practice of rigorous software design and analysis.
- In Sect. 8 we summarize the institutional impact the Logic School of Münster (together with the School of Freiburg) had on the advancement of mathematical logic, in particular in Germany, by contributing to and taking inspiration from the practice and the theory of computing.
- In an appendix (Sect. 10) we show the genealogy of the School of Münster.

2 Early Turing Reception in Münster

The presentation of Turing's paper [17] to the logic seminar in Münster triggered three early publications that used Turing machines (in 1937, 1952 and 1954), written by Hermes who in 1937 was doctoral student of Scholz. In each of these papers which were addressed to a general educated public Hermes used Turing's definitions but adapted them to the theme each paper proposed to explain. The main modification with respect to Turing's original paper concerned the concept of computation or run of a Turing machine. Hermes tailored it by a finiteness condition (together with requiring runs to start with some input and to terminate with some output) that fits the definition of decision procedures, replacing the computations of the more complex circle-free machines Turing had tailored for computing infinite 0-1-sequences that represent real numbers in binary decimal notation.[7] The finiteness condition led Hermes in a natural way to the halting problem which is of lower arithmetical complexity than Turing's circle-freeness

[7] Turing notably used binary fractional numbers in a time when the rest of the world thought decimal, except two years later Zuse and four years later Atanasoff [20].

problem. The definitions went into the lectures Hermes delivered on the subject regularly since 1949 (see the early lecture notes [146] published in 1955 and the computability book [147]) and became part of common usage.

2.1 Turing Machines to Compute 'Definite' Predicates

In [142] Hermes saw a possibility to contribute to a rational discussion concerning the foundational question whether in mathematics only 'definite' predicates should be allowed, as claimed at the time by intuitionists and constructivists in opposition to the 'classical' understanding of mathematics. A predicate P (over the natural numbers or over words of any given alphabet) was considered to be *definite* if one can indicate a general procedure to decide for each argument x in a finite number of steps whether P is true for x or not.

Hermes used Turing machines to replace the intuitive understanding of 'a general decision procedure' by a precise mathematical concept. Therefore he proposed to define P as definite if (our wording) there is a Turing machine M such that for every argument x,

1. *M started with input x*
2. eventually *yields a result*, namely *yes*/*no* if and only if $P(x)$ is true/false.

This became the standard definition of what nowadays we call a *decidable* predicate P or also a predicate whose decision problem is *algorithmically solvable* (see for example [211, p.69, Def.2.1]). The two features 1. and 2. had to be integrated into the conceptual framework Turing tailored to compute numbers [17].

Ad 1. To start a computation not with the empty tape—as Turing requires for computing a number—but to 'supply the machine with a tape on the beginning of which is written some input' is used by Turing in two places (without naming it by a definition): for the construction of a universal machine [17, p.241] and for proving the unsolvability of the *Entscheidungsproblem* [17, p.259]. Hermes uses this input mechanism without changing the 0-1-representation of the input on the tape.

Ad 2. To yield a result necessitates some way to determine when the computation reaches a point where it is ready to provide a definite output (read: the answer whether P holds for the input x or not). Turing refers to such a feature (without defining it) where in an indirect proof he speaks about computations of the hypothetical machine $\mathcal{D}$ that decides circle-freeness and considers "when it has reached its verdict" [17, p.247] for the given input. Hermes makes this explicit by introducing a special symbol, say H, such that M, once started,

- after a finite number of steps will print the symbol H,
- up to this step has printed a 0-1-sequence that is interpreted to yield the result of the computation,
- from this moment on will never print any more any digit 0 or 1.

Note that the third condition—which guarantees that the "verdict" concerning the input question has been pronounced and will not change any more—turns such machines into circular (maybe not even halting) machines, exactly

those Turing was not interested in. We see here that the PhD student Hermes paid some attention to not change anything in Turing's definitions (neither of TM-programs nor of their computations) but to only adapt the *interpretation* of specific TM-runs to make them fit for 'deciding a property in finitely many steps', instead of computing an infinite 0-1-sequence. In Sect. 2.2 we will see that 15 years later, at the age of 40, Hermes replaces this veiled way to trick a Turing machine to serve as a decision procedure by a more streamlined and explicit definition of decidability of predicates and analogously computability of functions.

2.2 Entscheidungsmaschinen and Halting Problems

In his second Turing machine paper [144] Hermes simplified the stipulations made in [142] and made them more explicit, adding a termination convention to obtain a conceptually simple concept of what he called *Entscheidungsmaschine* (decision procedure): input/output Turing machines which, started with some input, terminate their computation after a finite number of steps and provide as result a yes/no answer to the input related question.

Technically this is achieved by adding to Turing machine programs at least one dedicated termination instruction $(q, a, b, move, halt)$ whose execution in state q when reading a leads the machine to the successor state *halt* in which the machine will stop (read: has in its program no quintuple $(halt, ...)$ to execute). Then one gets the definition of an *Entscheidungsmaschine* E we all became used to and explained already above, i.e. E decides P if and only if for every x the machine if started with input x halts after a finite number of steps and its computation result is 1 (e.g. in its current working field) if and only if $P(x)$ is true, otherwise the result is 0.

The historically interesting fact one can observe here is that with such a simple halting convention (of a kind everywhere used today) Hermes proves in half a page, using a standard diagonalization argument, the undecidability of various forms of what nowadays is called the *halting problem*, thus considerably simplifying Turing's proof of the undecidability of the circularity property [17, pp.246–247]. The same year also Kleene [249, p.382] has proved the undecidability of a halting problem.

2.3 Universality of Programmable Computers

In his third Turing machine paper [145] Hermes sets out to show why programmable computers, as already available in 1954, are universal in the sense that one can construct a programmable computer such that for every computing machine its computational power is included in the computational power of that computer. This mathematical endeavor requires a mathematical definition of a) 'computing machines' and of b) the 'computational power' of programmable computers and computing machines.

To turn b) into a mathematical concept Hermes views (computations of) computing machines as computing functions over natural numbers (p.42)[8] so that their computational power can be defined by the class of number theoretic functions they can compute. For a) Hermes assumes Turing's thesis that every computable function is a Turing computable function, relying upon the accumulated evidence available already in 1954 that the numerous attempts to mathematically define computability in various ways eventually all led to Turing computable functions. Therefore to construct a universal programmable computer it suffices to construct an idealized computer which can be appropriately programmed to simulate every given Turing machine M; the idealization refers to the assumption that the computer has infinite memory (to store the program code for M and to simulate the computation of M for any input).

This is what Hermes does in [145], introducing a certain number of normalizations of Turing machines and their computations to simplify the construction of the *computer* which, if appropriately programmed, for any given Turing machine M simulates the work of M. Some of these normalizations and variations thereof went into [147] and are common usage nowadays.

Fig. 4. The three Computability and Logic books by Hermes

1. The first simplification concerns the programs: instead of quintuples Hermes considers flowcharts of 5 basic component-TMs whose connection structure

[8] The underlying mathematical abstraction is that computing machines can handle arbitrarily large natural numbers, thus requiring that the memory is infinite. In [17, p.231] this appears in the form that a Turing machine "may also change the square which is being scanned, but only by shifting it one place to right or left", assuming that to the right and to the left of any square the mathematically idealized machine always finds another square. If one prefers to see memory as potentially infinite, there must be an operation that allows one to import any time new memory elements (enough pencil and paper for "a man in the process of computing"[17, 231]) from somewhere.

can easily be encoded by corresponding links (gotos) between subprograms of the *computer* program, one for each basic component machine. Obviously it is assumed that there is enough memory (unbounded register content) to store the given flowchart. For the connection in the flowchart visualization each component has exactly one entry, the Test component has two exits yes/no, all the other components have at most one exit:

- R-ight machine,
- L-eft machine,
- M-ark machine (prints $*$ in the working field),
- N-ull machine (prints blank in the working field),
- T-est machine (whether working field is marked or not)

2. Simplification of the tape: tape with a left end and infinite to the right, only two symbols $*$, *blank*.
3. The following normalizations concern the concept of computation.
   - Initialization: exactly one component is distinguished as the currently-to-be-executed one. The initial tape is $\bar{n}$ (n a natural number) with the working position on the rightmost marked field; $\bar{n}$ is the tape beginning with $blank * \ldots *$ (n occurrences of $*$) with completely blank rest of the right-infinite tape.
   - Stop criterion: after having executed a component without exit, the machine halts. The final tape is $\bar{y}$ where $y = f(n)$ and f is the computed function. Only total functions are considered.

.

With these normalizations of TMs it is easy to encode an M-configuration into the memory of a programmable computer as follows:

- place the sum of all 2^k for all marked fields k into a register, say $reg(tape)$,
- place the number of the current working field into a register, say $workPos$.

Thus it only remains to construct for each basic component machine c a program $pgm(c)$ the *computer* will execute to simulate the behaviour of c, using a small number of memory locations.

Remark. Historically two facts are worth to be noticed:

- Hermes uses (in 1954) a number register for the encoding of the tape. The needed idealization with respect to physical computers is that such a register can hold arbitrarily large numbers.[9]
- It is only a small step from the flowcharts of 5 basic TM-components

$$R, L, M, N, T$$

to Rödding's structured programming Turing machines, called *Turing operators* [78]. The goto-structure of the flowchart is replaced by an algebraic

[9] Assuming that adding 1 to a register content n yields a register content $n + 1$ corresponds to Turing's assumption that each move to the right or to the left of a current tape square finds yet another tape square.

program structure that is defined using concatenation and iteration; otherwise stated the test component T is normalized to appear only at the beginning and end of an interated subprogram M so that it has the form

$$(M)_*$$

denoting to iterate M until the symbol $*$ appears in the working field. The same for $(M)_{blank}$. These operators (and even more their register operator relatives, see Sect. 3.2) simplify a lot the construction of Turing or register machines to compute recursive functions [78,81,30].

3 From Recursion Theory to Complexity Theory

In Münster, the path from recursion theory to complexity theory has been opened by two lines of research which have been pursued already before theoretical computer science became an academic discipline:

- description of machine computations by logical formulae relating computation power of the machines (measured in terms of the structure and computational strength of elementary machine operations and of computation time and/or space) and logical expressivity of the formulae (Sect. 3.1),
- machine-based characterization of hierarchies of recursive functions relating computational means to mathematical expressivity (definability schemes classified mainly in terms of the nesting of forms of recursion in traditional equational definitions of functions) (Sect. 3.2).

3.1 Machine-Characterization of Logical Decision Problems

The study of logical and algorithmic decision problems was part of the logic curriculum and a major thread of logic research in Münster. The subject was treated in regular courses (see the lecture notes [146,79,80,87])[10], diploma (master) and doctoral theses [176,177,86], articles [110,175,154,151,152,237,188,190,48,72] and monographies [7,44]. We refer here to just a few characteristic examples, for a complete Annotated Bibliography concerning the classical *Entscheidungsproblem* see [44].

Two lines of research characterized the traditional approach to the *Entscheidungsproblem*: develop algorithms for decidable cases [7] and reduce the general

[10] The institute had the tradition, initiated by Scholz (see [153, p.45]), to make lecture notes available for many courses. See for example [120,8]. The first author knows from the time when he was responsible for the library of the institute that this library contains many historically valuable lecture notes and also reprints of papers from famous logicians. The lecture notes often contained new research results that were not published elsewhere. For example, in [80] an elegant and simple proof for the sharp Kahr reduction class was developed, based upon a geometrical representation of computations of register machines with two registers in the Gaussian quadrant. This proof is used in [44, Ch.3.1]. Other examples are mentioned below. The lecture notes tradition has been maintained by Hasenjaeger also in Bonn, see for example [122].

problem to the decision problem of smaller and smaller classes of formulae (*reduction classes*) [181]. Following Büchi [172], stronger and stronger reduction classes were obtained by refining Turing's method to formalize the computations of machines (but also of other algorithmic systems or games) of a given class by logic formulae of a given class such that the machine behaviour of interest is equivalent to the decision (usually the satisfiability) problem of the corresponding logical formula. Since the 1980s, initiated in [157,115], this method is applied also to determine lower bounds for the complexity of decidable cases.

Establishing such relations between machine-computation-power and logical expressivity then became a major theme of automata and complexity theory in theoretical computer science. To mention just one outstanding example of a complexity variation of Turing's result: the NP-*completeness* of the satisfiability problem of propositional logic formulae in conjunctive normal form proved in [250] is a polynomial-time-restricted version of the Σ_1-*completeness* of the decision problem of predicate logic proved by [17] and in fact can be shown (see [33, Sect.2.3.1]) to be an instance of a general parameterized scheme (which has been developed in [29,28]) for logical implementations of machine programs.

The scheme relates algorithmic systems and their logical descriptions in such a way that numerous recursion and complexity theoretical properties are easily carried over from combinatorial to logical decision problems [27,26]. This includes the theoretically especially interesting case of Prolog programs where for one object—a Horn formula—different interpretations and their complexity properties are related, namely the computational properties of its program interpretation (e.g. Turing universality) and the logical properties of its purely logical interpretation (e.g. Σ_1-completeness of decision problems) [48,34]. The scheme can also be instantiated (see [44, Sect.2.2.2] for details) to prove Fagin's characterization of NP as a class of generalized first-order spectra [228], a result which marks the origin of Descriptive Complexity Theory where systematically logic languages are designed to capture computationally-defined complexity classes [219], a branch of finite model theory to which the **Logic School of Freiburg** contributed considerably [84] (see the lecturers from the *Coloquio sobre Logica Simbolica* at the Centro de Calculo de la Universidad Complutense in Madrid (19.02. – 21.02.1975) in Fig. 5 with Hermes and his former students and colleagues in Freiburg Dieter Ebbinghaus (Münster 1967) and Jörg Flum (Freiburg 1969)).

The very notion of spectra—for any given formula the set of the cardinalities of its finite models—and the question how to characterize them (called *Spektralproblem* and formulated in [159]) have been discovered by Hasenjaeger, Markwald and Scholz when they saw Trachtenbrot's undecidability result for finite satisfiability [260]. In 1971, applying Büchi's machine description technique from [172] to register machines (as done in Rödding's 1968 lecture [79]) and using the Rödding hierarchy $DSPACE(f_n)$ of functions—defined in [77] in terms of n-fold exponential time-bounded deterministic register machine computations, starting with the Grzegorczyk class E_2 and contained in and exhausting the

Fig. 5. J. Prida, H. Hermes, J. Flum, D. Ebbinghaus, E. Börger, unknown (Madrid 1975, names listed from right to left)

Grzegorczyk class E_3[11]—Rödding and Schwichtenberg [238] provided an elegant n-th order logical description of n-fold exponential time-bounded register machine computations with the result that n-th-order spectra form a strict subelementary hierarchy and exhaust the class of Kalmár-elementary sets. The result strengthened the early discovery, established using different (model-theoretic and number-theoretic) means by Asser [117] (as student of Schröter a scientific grand child of Scholz) and Mostowski [15], that first-order spectra reside between the classes E_2 and E_3 of the Grzegorczyk hierarchy of primitive recursive functions. The result rediscovered most of Bennett's unpublished thesis [171] which was based on a sophisticated analysis of the complexity of schemes to define number theoretic functions and relations and not on the computational complexity of machine programs that compute such functions.

Independently, in [180] a similar (but technically more involved) formalization of bounded computations of *non-deterministic* Turing machines by 'spectrum-automata' is developed which yields a characterization of first-order spectra by sets which are exponential-time acceptable by non-deterministic Turing machines, result proved independently also in [227]. Christen, using a sophisticated refinement of Rödding's computation time classification of subelementary functions [77] for register machines working over words, instead of numbers, has completed the Rödding-Schwichtenberg result in his doctoral thesis [64] to the final characterization of n-th order spectra as the class of $NTIME(f_n)$-sets of positive integers, where $NTIME(f)$ is the class of sets which are accepted by a

[11] For the Grzegorczyk hierarchy see [30, Ch.C].

non-deterministic Turing machine in time $f(c|x|)$ (for some constant c and input length $|x|$) and the n-fold exponential functions f_n are defined by $f_1(x) = 2^x$ and $f_{n+1}(x) = 2^{f_n(x)}$. Note that also the critical part of this characterization can be proved by a simple instantiation of the computation description scheme developed in [29] (see [44, pp.52–53] for the technical details).

Another line of research on the spectral representation of predicates appears in a series of papers by Deutsch (see the list in the Annotated Bibliography in [44]) where he uses normal forms of recursively enumerable sets he had developed in his doctoral thesis [213] to sharpen numerous reduction classes by restricting the interpretation of the unique occuring binary predicate symbol to an ϵ-like relation over transitive sets.

Numerous interesting questions about spectra of (fragments of) first-order or higher-order logics are still today without answer. For a survey of the extensive later developments of the Spektralproblem in theoretical computer science and logic see the detailed and very informative survey [11].

3.2 Machine-Characterization of Recursive Functions

The second major thread of research in Münster which influenced later developments in complexity theory was the machine-based characterization of hierarchies of recursive functions, relating mathematical definability schemes (see [230,194]) to computation time and/or resource consumption needed to compute functions by restricted Turing-like machines, in particular (structured) register machines (which Rödding discovered in 1959/60 before they appeared in the two—for the study of decision problems most influential—papers [215,251], see Sect. 4 and [125,234,231]).

Fig. 6. László Kalmár and Dieter Rödding (Lecture in Münster, 1970)

Rödding's doctoral thesis [75] on Kalmár's class of elementary functions triggered much further work on this theme in Münster, in particular [83,232,77], the doctoral dissertations [161,158,94,139] and [162,163]. See also [82, Ch.V.5].

Putting together the work done in Münster (which includes Heinermann's investigation of recursion classes in his doctoral dissertation [264]) with [12,243,10,19] yields equivalent characterizations of the Grzegorczyk hierarchy of classes of primitive recursive functions by measuring the nesting of bounded recursion, the nesting of recursion, the growth bound by branches of the Ackermann function [6], the nesting of loops and the computation time bound of register machines to compute the functions (see the Grzegorczyk-hierarchy theorem in [82, Ch.V3] and [30, C.II.1]).

A side remark: computer scientists may be interested in the observation that the use of structured register machines (called register operators, with an analogous definition of Turing operators, see [78,234],[81, Ch.I.4] and their use in [30]) also allowed to sharpen the famous Böhm-Jacopini characterization of computable functions (see [62]) which has played a role for the development of the concept of structured programming.[12]

The two books shown in Fig. 7 document with proofs, further references and detailed historical comments the way computing machine concepts led from recursion theory to complexity theory and from the study of highly unsolvable decision problems to the study of the exact complexity of their decidable subcases, development to which the Logic School of Münster contributed considerably, as has been outlined in this section.

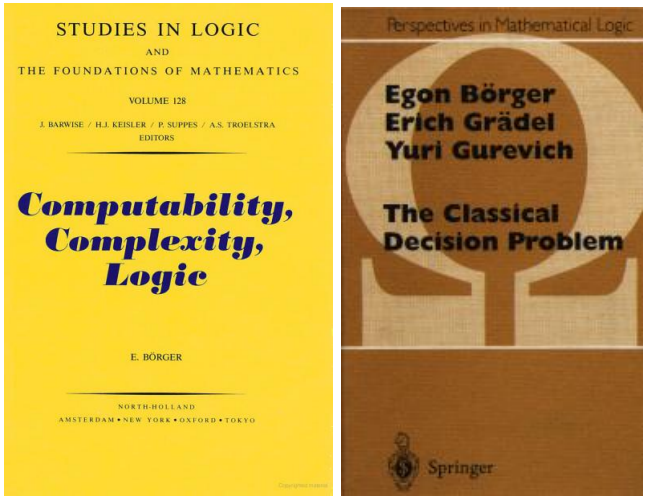


Fig. 7. Logic and Machines: From Decision Problems to Complexity

[12] For a generalization of the Boehm-Jacopini characterization from Turing machines to Abstract State Machines see [53, Proposition 5].

4 Turingraum

Copeland remarks in [71, p.54] that

> the mathematical representation of a Turing machine must not be confused with the thing that is represented – namely, an idealized physical machine

and concludes his chapter in the book by stating that

> What Turing described in 1936 was not an abstract mathematical notion, but a solid three dimensional machine containing (as he said) wheels, levers, and paper tape...

Turing was continuously engaged in practical projects (see Hodges [13]):

- in 1939, designed and built a mechanical machine to calculate roots of Riemann's zeta function (p.155),
- in 1944, invented and built a speech scrambler from valves (p.273),
- in 1945, designed the soft- and hardware for the Pilot ACE (p.333),
- in 1949, contributed a hardware random number generator for the Manchester Ferranti machine (p.402).

Fig. 8. Entrance to Turingraum in Münster (around 1962).

In the same spirit—combining theoretical work with the desire to build tangible theoretically well-founded machines—Hasenjaeger, soon joined by his doctoral student Rödding, pursued the goal to *materialize* (as Hasenjaeger wrote in

[125, p.182]) the theoretical concept of a universal Turing machine by some real physical machines one could also use to demonstrate the concept in lectures on the subject.[13] In the late 1950s Hasenjaeger and Rödding, supported by Hermes,[14] transfered these activities from Hasenjaeger's home to a dedicated room in the logic institute that was named *Turingraum* (Fig.8).[15] Here running physical models of small though computationally universal machines were built with simplest means until Rödding's early death in 1984; the same year Hasenjaeger retired but continued his materialization work at home.[16]

This materialization endeavor was triggered by a talk Friedrich Bauer gave in the middle of the 1950s at the logic institute in Münster where he presented his electro-mechanical model *Stanislaus* to evaluate algebraic terms in parenthesis-free notation. Upon Hermes' suggestion Hasenjaeger constructed for use in the institute a specimen of that machine, for details see Sect. 9.1.

This work soon led to the idea to build a physically running small universal Turing machine. As Hasenjaeger points out in [125] the "bottleneck was the materialization ... of a TURING tape" that was rewriteable.[17] The starting idea was to exploit the following ideas by Moore and Wang, of which Hasenjaeger only mentioned the first two ones:

- the reduction of the number of states in Moore's universal machine obtained by introducing a separate program tape plus an additional auxiliary tape (3-tape machine with only 15 states and 2 symbols [217]),
- Wang's observation [165] that erasing (overwriting) is not necessary; read and write-on-blank operations suffice (together with the move operations to the right/left) for a universal machine.
- Wang's proposal to use instructions instead of encoded state tables, which—at least in hindsight—was another step towards small and quick state machines.

[13] In [123, Part 1] Hasenjaeger speaks about physical 'models whose behaviour can be followed in "human" dimensions'.

[14] Already in his 1952 paper Hermes invites the technically interested reader to think about "how one can realize a Turing machine in practice"[144, footnote 5, p.185].

[15] The enlarged part below the room number shows that Turing's name was used as room name. The *Turingraum* together with a *Fregeraum* (reserved for the work on the Frege edition [156,116]) and two rooms for guest researchers formed a Dependance of the logic institute [268], given the lack of space in the castle which hosted the mathematical institutes until the end of the 1960s; at that time the *Turingraum* moved together with the Institute for Mathematical Logic to the new math building in the Einsteinstrasse.

[16] After Rödding's death the *Turingraum* has been disbanded by the new direction of the institute, the material was abandoned without further notice in a lumber room. To save it from greater damage it was quickly brought to Walburga Rödding (see [268]) who preserved it over the time, until in 2011, she and Hasenjaeger's family donated all physical artefacts from Hasenjaeger's legacy to the Heinz Nixdorf MuseumsForum in Paderborn (`http://www.hnf.de/`).

[17] Nota bene that Turing's paper suggests to use two tapes (even and odd fields) and to distinguish between erase and overwrite operations.

Out of the many artefacts that survived, the second author has been able to reconstruct Hasenjaeger's Mini-Wang machine, a Universal Turing Machine with only 4 states, 2 symbols and 3 tapes: a read-only program tape, a non-erasable working tape and a counter tape that is used to implement instruction skips. For the technical and historical details of this and related Turingraum machines see Sect. 9. Here we notice only that the effort to build universal but operationally surprisingly simple and running physical machines by no means lacked its scientific output:

- The remarkably small and physically executable Mini-Wang turned out to be efficiently universal among the dozens of conceptual universal Turing machines in the literature (see [218]). With hindsight one can also say that the investigation of computationally universal and with respect to a variety of parameters 'small' machines made the role explicit that different data, operations and architectural features (besides input/output mechanisms and stop criteria) play for the realization of the notion of computation:[18] number of symbols, states, tapes, the data type of and operations on tape contents (e.g. counters, stacks, read-only tapes, tapes with multiple parallel reads, cyclic shift registers), other topological structures than tapes (see in particular the work of Ottmann, Priese and Kleine Büning on universal machines we discuss in Sect. 5), etc. Hasenjaeger's abstract definition of register components of a net of machines in [119] looks like a presentiment of some particular classes of Abstract State Machines we discuss in Sect. 7.
- Rödding was led by the Q-tapes (counter tape) used in the Wang machines to the invention of register machines before their appearance in [215] and [251].

In fact, from the very beginning Rödding (who had enrolled at the university of Münster in 1956) got involved in Hasenjaeger's work and the creation of the Turingraum as working place for the construction of illustrative running machine models. When Rödding saw the use of a counter tape in the Wang machine he had the idea that counters alone could suffice to compute every partial recursive function.[19] He worked this out and presented the result to Hermes' Logic Seminar in Münster (see [125, p.184]), namely the definition of register machines (later

[18] This is related to Kleene's normal form theorem [248] that there are primitiv recursive functions in, out, $step$, $stop$ such that every partial recursive function f has the iterative form $f(x) = out \circ (step)_{stop} \circ in(k, x)$ for some k, where $(step)_{stop}$ denotes the iteration of the $step$ function until the $stop$ criterion becomes true (see the proof in [30, p.41]). Bruno Buchberger characterized the four component functions whose composition $out \circ (step)_{stop} \circ in$ defines a Gödel numbering (of the n-ary partial recursive functions for some n). These characterizations (see [30, Sect.BIII3] for proofs and references) show that one can design universal machines whose iterative component functions are of any a priori given (whether low or high) complexity, independently of each other.

[19] During the demonstration of some Turingraum machines at the Drei-Generationen-Kolloquium in Münster (see[46]) Hasenjaeger told the first author that he had asked Rödding whether one can represent sequences of natural numbers on 0-1-tapes of a universal Turing machine cheaply, in such a way that only an a priori fixed number

published in [78,234] with new results on structured programming normal forms) and the proof that every n-ary partial recursive function can be computed by a register machine with $n + 2$ registers and with prime number encoding even with only 2 registers, well before these results appeared in the famous papers [215,251].

The register machine concept turned out to be rather useful (see [169] for its role to pave the way for the Abstract State Machines concept, see Sect. 7). Rödding himself and his students made heavy use of them for an analysis of the computational power of numerous combinatorial systems, of the complexity of decision problems, of recursive functions, etc., as described in Sect. 3,5,6. Hasenjaeger used the elegant register machine proof by Jones and Matijasevich [179] for the theorem on exponential diophantine representation of enumerable sets in connection with his universal Turing machines to obtain a simple exponential diophantine predicate that is universal for the recursively enumerable sets [124].

So not surprisingly also register machines were a Turingraum theme, together with other components of Rödding's automata networks described in Sect. 5. See also Hasenjaeger's 'materialization' of (a general scheme of) register machines using SIMULOG instead of an electro-mechanical model [123]. See in particular the register machine materializations performed at the university of Osnabrück (see [99,63]) where these models have been applied with success for teaching algorithmic thinking at primary and secondary schools (see Sect. 6).

5 Networks of Machines

In the late 1960s until his early death Rödding together with a group of students analysed construction principles for (finite as well as infinite) automata and developed a theory for the modular decomposition of sequential automata by networks over a few simple basic automata [255,223,256,224,187,259,192]. Rödding's register operators (see [78,234]) appear here as nets with a particular graphical structure that visualizes the structured programming control. The theory made its way into the two textbooks [30,184] shown in Fig. 9.

Applications were found not only in computation theory [242,189,166,246,105] and logic [236,257,183]—where the inclusion of register components among the basic automata permitted to represent functionals of finite type, resulting in novel characterizations of the partial recursive functions (for the type 0 case), of the combinators K and S, of recursors, etc.— but also in theoretical biology [201], economics and systems theory [239,240,202,241,204,178], fault-tolerant switching theory [209,200,59] and Didactics of Mathematics (see Sect. 6).

Defining analogous nets of Asynchronous Parallel Automata (APA nets) [60] led to interesting results about concurrency [206,208,207]. The Turing spirit is particularly present in the applications of the theory of automata networks to the

of 1's appear on the tape. The answer was yes by representing n as distance of a unique occurrence of 1 to the left end of the infinite-to-the-right tape $0^n1000\ldots$, i.e. a register where move-to-the-right means $+1$ and move-to-the-left -1.

Fig. 9. Two Books with a Chapter on Rödding's Automata Nets

construction of small alternative models of universal (Turing complete) computational systems, e.g. asynchronous cellular spaces [199], multi-dimensional Turing machines [254,193],[184, Sect.14.7] and 2-dimensional Thue systems [203], see also [205] and the comparative analysis in [267].

Further references appear in the survey [235], in the cited papers and in the two textbooks of the years 1985 [30, Sect.CIV3-4] and 2000 [184, Sect.14.6-7].

6 Computational Networks in Didactics of Computing

Rödding's work with register operators and networks of automata has triggered two particularly interesting didactical applications that have been elaborated by Elmar Cohors-Fresenborg and his group at the university of Osnabrück for teaching computational concepts in primary and secondary schools.

Introduction of n-ary functions by register machines. The first of these two applications is based upon the discovery presented in [95] that the register machine model of computation can be used with success to introduce in school the mathematical concept of multi-variable functions. Based upon various teaching experiments this idea has been elaborated first for teaching to high-school students (see [97], a book that according to Wikipedia and [262, p.40] has influenced the construction of the Know-how computer `https://en.wikipedia.org/wiki/WDR_paper_computer`, see also [101,124]); further experiments showed that the method can be adapted for secondary [103] (age 12–13) and even late primary [102] (age 10) school level.

From the very beginning of this work various specimens of a dedicated register machine model have been built to visually illustrate the computations so that the students can play with the machines. One copy of the first of these register machines [63] was purchased in 1976 for the Turingraum. It has been used in lectures by Rödding and by Ottmann (in Karlsruhe [68]). The physics

laboratory of the University of Osnabrück developed later versions on the basis of microprocessors offering an output mechanism to an external TV screen; these machines have been used with success in numerous schools in Germany and are part of the mathematical didactics study program at the University of Cologne (see https://mathedidaktik.uni-koeln.de/mitarbeiterinnen/prof-dr-inge-schwank/forschungs-und-lehrprojekte). Since 1982 also a simulation on PCs is available (see https://mathedidaktik.uni-koeln.de/mitarbeiterinnen/prof-dr-inge-schwank/forschungs-und-lehrprojekte/registermaschine/english-englisch).

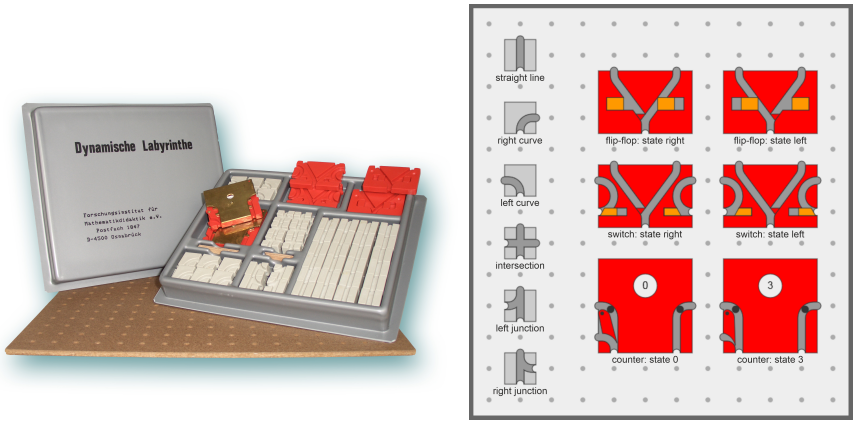


Fig. 10. The 9 Automata Construction Kit Bricks

Automata construction kit for elementary schools. In 1973 Elmar Cohors-Fresenborg started to exploit Rödding's networks of automata for didactical purposes (see [96]). The basic idea was to enable kids by a construction kit—consisting of bricks which are placed on a baseboard—to realize the computations of an automaton as walks through a net of basic components some of which perform a control action and others an operation on some data. For a simple to visualize but computationally universal concept of data, data operations and control, number register components (counters) came in handy with only two elementary operations +1, -1. Only two counters are needed; to realize their underlying Finite State Machine control they can be connected to an automata net of only two types of basic control components—flip-flop and switch—plus trivial support components like straight lines, curves, junctions, etc. (see [97]). The resulting *Automata Mazes*[20] construction kit (see Fig. 10,11) and its later software versions (see https://mathedidaktik.uni-koeln.de/mitarbeiterinnen/prof-dr-inge-schwank/forschungs-und-lehrprojekte/

[20] In German called *Dynamische Labyrinthe*, literally translated Dynamic Labyrinths, available via https://www.bildungsserver.de/onlineressource.html?onlineressourcen_id=10147.

automatentheorie-dynamische-labyrinthe) were applied with great success in elementary schools to teach algorithmic thinking in terms of counters plus typical railway net control! The offered teaching material that supports also self-study [100] has been translated to various languages including English, Chinese, and Dutch. It helps kids to realize algorithms as the result of non-verbal, action-oriented, motor thinking, an important form of mathematical reasoning pointed out already by van der Waerden in [263]. The didactical exploitation of Rödding's automata nets is applied with success also for very early training of particularly talented kids (see [247] and https://mathedidaktik.uni-koeln.de/fileadmin/home/ischwank/dynlab/maschinenintelligenz_mathematisieren_dynlab.pdf).

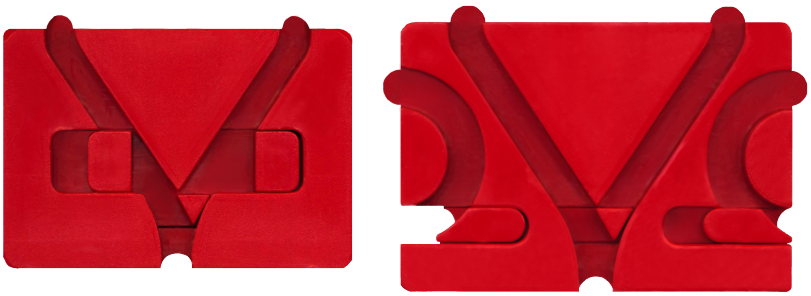


Fig. 11. flip-flop and switch bricks (in left position)

Functional versus predicative thinking. The two papers [98,69] report on extensive experiments with pupils who were observed during the construction of register machine programs or automata mazes. The result of these experiments led Inge Schwank to the conjecture that there are two forms of mental problem representation [245]: a *functional* one that is focussed on process runs, i.e. on the dynamic succession of process steps and their effect, and a *predicative* one where the attention is focussed on structural process elements and their relations. This difference has been experimentally confirmed by an analysis of brain currents and eye movements of test persons who were observed during a problem solving activity [216,104]; in [67] it is reported that these differences match the differences one can observe in how pupils approach algebraic or geometrical phenomena. In [70] Cohors-Fresenborg reports the rather interesting discovery of a similar difference in the mental representation managers have of business processes (see also [106,107]). It would be interesting to know whether Schwank's observation can be experimentally confirmed to also explain the two different system specification approaches advocated energetically by two camps of theoretical computer scientists, namely the declarative and the operational approach.

7 Computation on Structures Leading to ASMs

Since the middle of the 1960s various studies in Münster investigated computational concepts for objects which differ from numbers or words, using appropriate basic operations which work on those objects directly, without encoding. Early examples are the definition and characterization of primitive-recursive functions over hereditarily finite sets [76,233] and over sets of terms [111,112].[21] In the doctoral dissertation [94] register machines which operate on binary trees using a few natural basic operations are introduced to define and investigate the complexity of subelementary resp. elementary classes of functions over binary trees. The lecture notes [82] contain a chapter on generalizations of computable functions including a proposal for an axiomatic recursion theory along the lines of [109].

Later also Rödding's theory of networks of automata described in Sect. 5 led in a natural way to alternative computation models of topologically arranged automata or substitution systems [203,254,258,187,193] and asynchronous cellular spaces [199].

During those years two other logicians proposed to study computations over arbitrary relational (also called Tarski) structures [108,141]. Only much later were register machines used which operate on real (instead of natural) numbers, on rings, fields [197] or on finite relations over a fixed universe [65, Sect.4].[22] The query computability concept resulting from the last cited work played a major role in database theory. See [169] for a detailed historical analysis.

In 1982 the question appears whether one can replace Turing machines by a computation model over structures that captures the complexity class PTIME [66]. Three years later this question is extended in [134] to whether a computation model over structures can be defined which captures *every* sequential computational device, thus generalizing Turing's Thesis. The real breakthrough came with the idea,[23] nota bene conceived by a logician in an attempt to solve an epistemological question, to define a) a small number of elementary operations one can apply in any structure and b) a few composition (read: program construction) schemes to create composite sets of those operations, such that they suffice to define ('simulate') for every given algorithm its behaviour directly in terms of runs of an appropriate composition of those operations in the algorithm's 'natural' Tarski structures. This has been achieved in [135] by a simple mathematical definition of an arguably comprehensive algorithmic language. It lifts Finite State Machines (FSMs) and Turing Machines (TMs), which work over words, to **Abstract State Machines (ASMs)** which work directly over

[21] Note that in 1965 Hermes defined a logic of terms [149]. Much later the decision problem of this (and the pure ϵ-logic) has been in investigated in Freiburg [185], see [44, Ch.5.3].

[22] The query language definition in [65] is given in terms of variables y_i (which can be viewed as registers, each containing a finite relation) and a few appropriate basic operations which can be viewed as operating on register contents.

[23] With hindsight one is tempted to say 'simple idea', but as often happens with scientific discoveries the simple thoughts are the more difficult ones to find.

structures of *whatever* type (read: signature), using as basic action only guarded *term* (NOT only variable!) assignments **if** *condition* **then** $f(t_1, \ldots, t_n) := t$ and as composition scheme bounded synchronous parallelism.[24]

This operational definition of machines with arbitrarily abstract 'actions' broke with the at that time still main-stream declarative thinking in logic and theoretical computer science, but to the first author who had studied Tarski's Wahrheitsbegriff paper [21]—reading material for the introductory predicate logic course in Münster—and has been formed in the machine-based tradition in Münster, it quickly became clear that this definition enables to **mathematically support abstraction the way it is needed and used in the practice of computing** by engineers of software-intensive systems. In fact, ASMs permit to rigorously define algorithmic systems as computational models at whatever desired level of abstraction and to mathematically relate runs of an abstract and a refined machine (read: a machine that implements abstractions by more details) for a verification or test of the correctness of this 'implementation step'. These—by their abstract nature virtual—machines are not axiomatized by logical formulae but are computational models which drive the stepwise execution of the guarded-action-rules that capture the intended dynamic system behavior ('an evolution of states').

Fig. 12. Three Books on the ASM Method

The idea to use ASMs for a fully documented modular design, analysis and implementation of software-intensive systems by

- starting with appropriate rigorous but abstract requirements models (called *ground model ASMs* [36]) one can inspect like pseudo- (but semantically well-

[24] Other constructs, like **forall**, **choose**, **Call**, **let**, and other composition schemes, like asynchronous parallelism, can be easily integrated where useful. The work with ASMs revealed that the language can easily be tailored to the needs of domain specific applications. Note that the terms $f(t_1, \ldots, t_n)$ in guarded assignments are a general form of 'array variables' where the indeces are not just numbers, but arbitrary terms with possibly updatable values.

defined) code for its *appropriateness* with respect to the usually informally presented original requirements, and
- adding the implementation details by stepwise *ASM refinements* one can submit to test suits and to mathematical analysis for a *correctness check* [37][25]

triggered the **development of the ASM method** [54,253,50] (see Fig. 12), which is well-founded by its roots in logic and contributes effectively to the practice of computing, notably providing a system description and documentation technique of the kind Harel and Pnueli asked for in 1985 [137, p.480]:

> A natural, comprehensive, and understandable description of the behavioural aspects of a system is a must in all stages of the systems development cycle, and, for that matter, after it is completed too.

During the decade 1988-1998 the ASM community worked to make the ASM method fit for serious practical applications in a variety of computer science fields. A commented ASM bibliography for this period [47] counted 128 scientific contributions, which five years later became more than four hundred (see the commented bibliography in [88,54]); for references and the developments since then see `https://www.abz-conf.org/methods/asm`.

The **comprehensiveness of the concept of ASMs** is nowadays supported also theoretically, namely by numerous forms of Turing-like theses one can prove from natural axioms for appropriate classes of ASMs, characterizing for example sequential algorithms [136] (including a proof of Turing's Thesis for computable functions over the natural numbers) and their extensions to synchronous parallel algorithms [22,23,113], concurrent algorithms [51], recursive algorithms [52], reflective algorithms [182], etc. This work suggests the development of a realistic (theoretically well-founded and in the practice of software engineering helpful) complexity theory which is based not any more on Turing-like machines but on machines working directly over structures, avoiding extraneous encodings, see the very interesting recent survey paper [244].

In this respect it is interesting to note that half a century ago, in [275, p.6], the engineer Zuse critically remarked that much of research in theoretical computer science at the time disregards the badly needed dialogue between theoreticians and practitioners and that a logical algorithmic language is needed which helps the practitioner, in other words which is useful to reliably design, construct and analyze implementations. This is what the ASM method achieves, exploiting the machine-based pseudo-code-like yet abstract and semantically well-founded language of ASMs to design and analyse dynamic system behaviour.

8 Institutional Impact of the School of Münster

The Logic School of Münster had a strong impact not only on the scientific progress of mathematical logic as described above, but also on the institutional

[25] The concept of ASM refinement is not declarative but supports the direct description of system dynamics at different levels of abstraction. See [54] for details.

development of the discipline at German universities, especially in computer science departments where many positions became available for researchers in computational logic. We mention a few examples of influential science management activities of members of the group (besides those mentioned already in the chapters above).

Hermes acted as co-founder of the *Archiv für Mathematische Logik und Grundlagen der Mathematik* (1950) (later renamed to **Archive for Mathematical Logic**), as founding member of the *Deutsche Vereinigung für mathematische Logik und für Grundlagenforschung der exakten Wissenschaften* (**DVMLG**) (1962) and for many years has been co-editor of the *Journal of Symbolic Logic*. Together with Kurt Schütte from Munich he created and organized for many years the influential *Hermes-Schütte-Tagung* for mathematical logic in Oberwolfach. Schröter, who from 1936 to 1948 worked as student [195], assistant[196] and Dozent in Münster, established in 1950 the Institute for Mathematical Logic at the (now called Humboldt-) University of Berlin and in 1955 founded together with Asser the *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* (in 1991 renamed to **Mathematical Logic Quarterly**), acting as its editor until 1977.

Fig. 13. Hilbert/Ackermann and Scholz/Hasenjaeger Books

Members of the group wrote **influential books and textbooks**, some of them translated to other languages, on the main areas of logic. To mention a few: edition of the Frege Nachlass [156,116] (work that had been started by Scholz and his student Bachmann in the middle of the 1930s, has been destroyed by the fire of the university library during the bombardment of Münster on March 25, 1945, and has been taken up again by Hermes in the 1960s), textbooks or book chapters on logic [73,155,160,121,148,30,191,85] and computability [147,186,97,30,184], monographies on various subjects, e.g. term logic [149], the

Entscheidungsproblem [7],[44][26] (see also the first systematic textbook treatment of the *Entscheidungsproblem* in [73, Ch.12] which covers the results known at that time), metamathematics of geometry [270], proof theory [164].

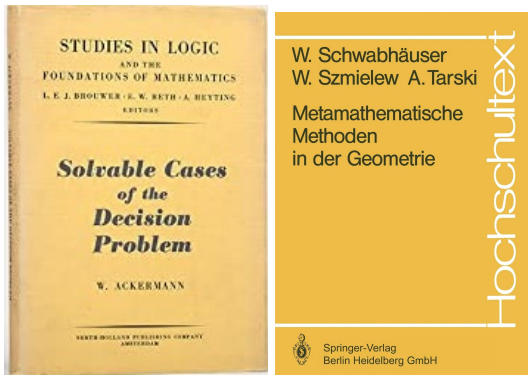


Fig. 14. Influential books by Ackermann and Schwabhäuser

Some numbers reveal the impact the computational focus of logic propagated by the *Schule von Münster* had on the **institutional growth** of the discipline in computer science departments of German universities. According to the Mathematics Genealogy Project [221] (see the list in the appendix)

- the school's first generation consists of 10 doctoral students,

[26] The first author has often been asked why it took [44] 25 years to appear. In 1972, on an invitation by the series editor Gert Müller, he had started to work on a volume in Springer's Lecture Notes in Mathematics with the material of his 1972/73 lectures in Münster on the *Classical Decision Problem* [87]. He stopped the project when in 1973 a doubt was expressed about a possible conflict with a book announced by Burton Dreben (Harvard); that book [74] came out in 1979 together with its companion book [198]. A critical analysis, performed with Yuri Gurevich during his visits to Münster and Dortmund (1978, 1983, 1985), of the complex machinery of Herbrand expansions used throughout in [74,198] and of the missing investigation of the algorithmic complexity of decidable cases eventually led to a new proposal of a comprehensive complexity account of the classical decision problem which was accepted by Gert Müller for Springer's Perspectives in Mathematical Logic, edited by the Ω-group. After various tergiversations by Gurevich to find another author, eventually Börger wrote Part I (on undecidable classes), updated the annotated bibliography he had compiled in the 1970s and attracted Erich Grädel to join the project. Grädel's work on the complexity of decidable subclasses of logical theories in his doctoral dissertation [129,130,131] and his postdoc research in Pisa (Spring 1988 – Fall 1989) [132,133] made him the ideal person to complete the book by writing Part II (on decidable classes and their algorithmic complexity), except the Shelah class which has been written by Gurevich. The appendix (on tiling problems) has been written by C. Allauzen and B. Durand.

- its second generation consists of 81 doctoral students at the universities of Kiel, Berlin, Münster and Bonn (Bachmann 26, Schröter 16, Hermes 28, Hasenjaeger 11)
- D. Rödding, Scholz' second successor as director of the institute, supervised 15 doctoral students from 1966 until his early death in 1984.[27]

Alltogether in three generations Scholz has the extraordinary number of 1,625 descendants (Schröter 260, Bachmann 314, Hermes 392, Hasenjaeger 371, Rödding 246, Schwabhäuser 42).

Logic and Machines [46] (and more generally *Logic and Computation Theory* [31]) reflect the new horizon of scientific challenges the School of Münster discovered for the interaction of mathematical logic and computer science, in the spirit of the mathematician Turing and interestingly also of the engineer Zuse who submitted a manuscript [274] for a doctoral dissertation to Scholz.[28] *Logic and Machines* is the title of the Proceedings [46] of an **international symposium** which was organized in May 1983 in Münster—Hasenjaeger named it the *Drei-Generationen-Tagung*—to let reseachers come together who are interested in the cross-pollination between Logic and Computer Science. During this symposium Hasenjaeger and Rödding showed and explained to the participants some of their *Turingraum* machines.

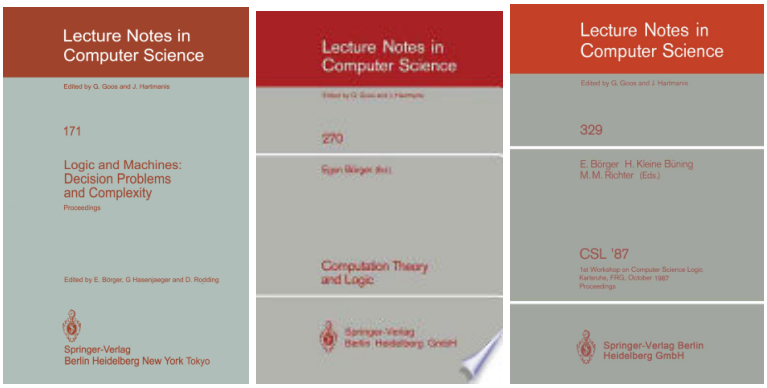


Fig. 15. Books that Marked the Path to CSL

[27] See the list in [32] which corrects the one of the Mathematics Genealogy Project. It includes [170,199,178,59], see also https://www.uni-muenster.de/FB10/historie/anhangD.pdf. It mentions also five of Rödding's Diplom (Master) students who later have written a PhD thesis at other universities.

[28] Scholz wrote a positive evaluation, but due to the war and post-war conditions the PhD procedure did not reach its natural end. See [140, p.2]. Scholz showed great interest in Zuse's work; when Hans Lohmeyer, a former student of Scholz, worked with Zuse in Berlin he brought Scholz there for a visit (see [275, p.62]).

This was at a time when those who in Germany tried to bring logic and computing together experienced a strong resistence from a group of short-sighted professors of mathematical logic who did not understand the potential that computing held in store for their discipline.

The success of this symposium (with over 50 participants from 9 European countries and the US) gave the first author the idea to institutionalize such a forum by forming an annual **Computer Science Logic conference series**. After Rödding's unexpected death he attracted Hans Kleine Büning (one of Rödding's doctoral students) and Michael Richter (one of Hermes' early doctoral students in Freiburg) to join the endeavour. This was a year before the ACM/IEEE Symposium on *Logic in Computer Science* (LiCS, https://lics.siglog.org/) was launched in 1986. However, due to the adverse personal interest of somebody—who in 1985 after Rödding's death cut off the Turing tradition at the institute[29] (but obviously could not stop the strong development of computational logic in the scientific world) and triggered the first author's move to the University of Pisa—the conference series could not start in 1986 and not at the logic institute in Münster, but only a year later at the Institut für angewandte Informatik und Formale Beschreibungsverfahren in Karlsruhe.

The first seven years are documented in [40,41,42,43,39,38,56]. In 1992, during a Dagstuhl seminar [55] which has been attended by logicians and computer scientists from 14 countries the first author proposed to transform the CSL conference series into the annual gathering of a **European Association for Computer Science Logic** (EACSL), see [35] and for the complete list of CSL conferences and Proceedings https://dblp.uni-trier.de/db/conf/csl/index.html. Notably, when the EACSL upon Makowsky's proposal created an annual award for an outstanding dissertation in the area of logic in computer science, this distinction was named after Ackermann to stand for logic and computation. We quote from [18, p.VIII]:

> Together with ... LiCS, CSL counts as one of the most prestigious conferences in theoretical computer science focusing on the connections between logic and computing.

In this connection it is also interesting to remark that in the year 2000, a special conference subseries has been created that is devoted to "the foundational interconnections between *Logic and Computational Complexity*" (see https://www.cs.swansea.ac.uk/lcc/).

Also the development of the ASM method has been supported by a series of **International ASM Workshops**, co-founded in 1994 and until 2007 steered by the first author together with Yuri Gurevich. From the very beginning, many of these meetings were held as part of larger computer science conferences to easen the integration of the ASM method into current system engineering environments—ASM 1994 as part of the IFIP World Congress in Hamburg [222], ASM 1998 as part of the GI-Jahrestagung in Magdeburg [261], ASM 1999 as part of the Formal Methods Europe conference FME'99 in Toulouse, ASM 2001 as

[29] See the documentation in [25].

Fig. 16. Books that Marked the Start of ABZ Conferences

part of Eurocast'01 in Las Palmas [229,45]. Some more Proceedings have been published in [273,1,91,89,271,93,90,92].

During a Dagstuhl Seminar on rigorous methods for software construction and analysis [167] it became clear that state-based methods like ASMs, Z (Zermelo), B (Bourbaki) and others, all of which are based upon logic and set theory, have a lot of commonalities but also differences which should be clarified to enable practitioners to combine such approaches where this can help to develop and analyse reliable software for complex software-intensive systems. This led the first author to propose to Jean-Raymond Abrial, the creator of both the Z and the B (and Event-B) method and the leader of the community [2,3], to merge the ASM Workshops with the regular meetings of the Z and B user groups into a forum where common methods and ideas are investigated to reach a fruitful integration. This led to the establishment of the the biennual international **ABZ-Conference series** (`https://www.abz-conf.org/`) which has been launched in 2008 in London [49] and since then continues to be held regularly in Europe and Canada [49,214,173,272,24,61,210,226,225].

9 Analysis of Turingraum Artefacts

As already explained in Chap.4, Hasenjaeger had created several artefacts to show the *materialization* of theoretical concepts, and many of these still exist in various conditions by the effort of his family and W. Rödding. They are now

located in the Heinz Nixdorf MuseumsForum (HNF) in Paderborn, due to the initiative of its founding director Norbert Ryska. Whithout his engagement, in particular the *Mini-Wang* would still be unknown. Only the latter one has been analysed quite thoroughly and found to be still working; it is a universal Turing machine with (only) four states and three binary tapes, only one of them could be modified in a write-only-once manner. The other machines will be described here roughly; some of them might justify more deeper analysis. Also, for some artefacts their use is still unknown.

Documentation on these machines is often not existant; a few hints are in some of Hasenjaeger's publications. Several relevant texts appear in his paper legacy [118], but normally they are undated and difficult to match with the artefacts in the HNF.

The follwing comments on (some of) the artefacts now owned by the HNF are in hopefully historical order.

9.1 Kasimir

In his article [125] Hasenjaeger wrote that in 1956 F.L. Bauer from Munich reported in the *Logistisches Seminar* in Münster about an electromechanical model to evaluate parenthesis-free logical expressions, see [114]). His machine is in the *Deutsches Museum* in Munich, but currently not displayed.

Hans Hermes asked Hasenjaeger if he would like to build a similar machine ([125, p.182]):

> H. Hermes suggested that I should make a specimen of STANISLAUS for our institute, and F.L. BAUER sent me a blueprint of his version.

Bauer replied with the above mentioned blueprint and more information about his solution:

> Es sei:
> R die Anzahl von verschiedenen Variablen,
> S die Anzahl von verschiedenen logischen Operation einschlielich der Negation und der Identität, die Sie mit einem Formelrechner behandeln wollen.
> M die höchste Anzahl der in einer Formel vorkommenden Variablen und Operationszeichen.
> Dann benötigen Sie:
> 2 M Relais mit je einem Ruhe und Arbeitskontakt (für die Logik).
> 2 M Tastenstreifen mit je R+S Feldern, je Taste etwa 5 Ruhe und Arbeitskontakte;
> Einen doppelten Satz von je M Relais, deren Kontaktbestückung von 1 bis M/2 linear anwächst und ebenfalls linear abfällt (Für die Wegeschaltung)

This means, that the number of relays is quadratic with the number of terms. Hasenjaeger characterizes his machine KASIMIR in the above paper:

Its main features were:
(a) a three position switch for each place of the formula,
(b) a pair of relays to be activated by (a) for the shifts,
(c) a 10 position switch for each place of the formula, determining
(c1) the actual binary connective (ther are just 10 non-trivial ones) in connective-position of (a)
(c2) the subscript of the variable in variable-position of (a)
(d) two relays for each place of the formula for the actual connective
(e) but no realisation of a unary functor.
Being different from STANISLAUS our model needed a different name; I think `KASIMIR` was taken after K. AJDUKIEWICZ [1935].

Nothing had been published about *KASIMIR*, although it looks like the number of relays was only linear with the number of terms. It has not yet been studied if this is possible at all; so a complete reverse engineering of the machine might be valuable.

Fig. 17. Kasimir central part

Of the machine, two specimens of the first version survived (See Fig.17):

- one apparently complete machine (provided 2011 by W. Rödding)
- one badly preserved frame (found 2012 in Hasenjaeger's home in Plettenberg)

In Hasenjaeger's paper legacy [118, number 084] some notes may be found, at least for a later version (1977) with a different technology. A deeper analysis of the first version has been done by the second author.[30]

9.2 The "Alte Wang"

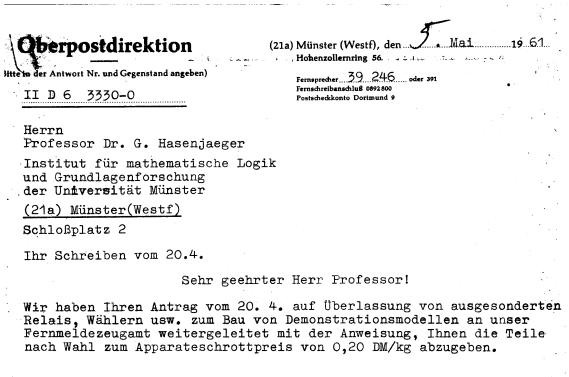
When Hasenjaeger learned about Moore's and Wang's proposals for practical Turing machines (see Chap.4), he started the work on his first Turing machine *materialization*.

This initiated the creation of a machine that could be programmed by short cables with 2.6mm plugs used in toy trains. It was originally called the *Wang*, and later *Alte Wang* (old Wang), see Fig.18.

Fig. 18. Alte Wang

Used were discarded relays[31] that were sold for their material price to educational institutions by the German post office, see Fig.19, which had a major office (*Oberpostdirektion*, OPD) in Münster.

[30] https://rclab.de/hasenjaeger/kasimir
[31] Flachrelais 28, i.e. flat relay, produced from 1928 on for the German post office



Oberpostdirektion

(21a) Münster (Westf), den 5. Mai 19 61

Hohenzollernring 56.

(Bitte in der Antwort Nr. und Gegenstand angeben)

Fernsprecher 39 246 oder 391
Fernschreibanschluß 0892800
Postscheckkonto Dortmund 9

II D 6 3330-0

Herrn
Professor Dr. G. Hasenjaeger

Institut für mathematische Logik
und Grundlagenforschung
der Universität Münster

(21a) Münster(Westf)

Schloßplatz 2

Ihr Schreiben vom 20.4.

Sehr geehrter Herr Professor!

Wir haben Ihren Antrag vom 20. 4. auf Überlassung von ausgesonderten Relais, Wählern usw. zum Bau von Demonstrationsmodellen an unser Fernmeldezeugamt weitergeleitet mit der Anweisung, Ihnen die Teile nach Wahl zum Apparateschrottpreis von 0,20 DM/kg abzugeben.

Fig. 19. OPD reply

As tapes, Hasenjaeger used 35mm perforated paper film used for contact prints.[32] Like ordinary 35mm film, it was perforated on both sides, so it could be split longitudinally, which Hasenjaeger did probably himself. [33] Hasenjaeger used paper film, as it was easier to punch than transparent celloloid film (used e.g. in Zuse's Z3), as the second authors experience in running the machine showed.

The tape drives were made in the workshop of the physics department in a quite professional way, see Fig.20.

In the middle is the slot through which the film guided, and at its bottom is the gear wheel that uses the perforation to move the tape many times without slack. On the left side is the stepper mechanism, that can move the tape left or right. On the right side is the sense and punch mechanmism. The punch is the outer tube, operated by the magnet with the heavy block on its armature.[34] To determine if there is a punched hole, a sense pin is moveable within the punch tube. It is normally kept away to allow free move of the tape. Activating the lower magnet, the sense pin is released and its position thereafter sensed by the leaf spring contact left to the magnet.

Hasenjaeger did not mention in [125] a very important feature of Wang's solution: Instead of encoding of a state table with an elaborate state machine to scan each line of the encoded table for a match, and then follow the actions, instructions are used like in stored progam computers, where the machine language is not a state table. Just the universal machine itself uses a state table, as in modern computers the microprogram that interpretes the instructions from

[32] At that time, often a (black-and-white) film was given to a drugstore to develop and produce 1:1 contact copies on this paper film, which did cost only a fraction of an enlarged copy of each picture.

[33] using a tool which is no longer known

[34] The obvious purpose is to provide enough mass to punch, even if such magnets have a much larger force short before closing anyhow.

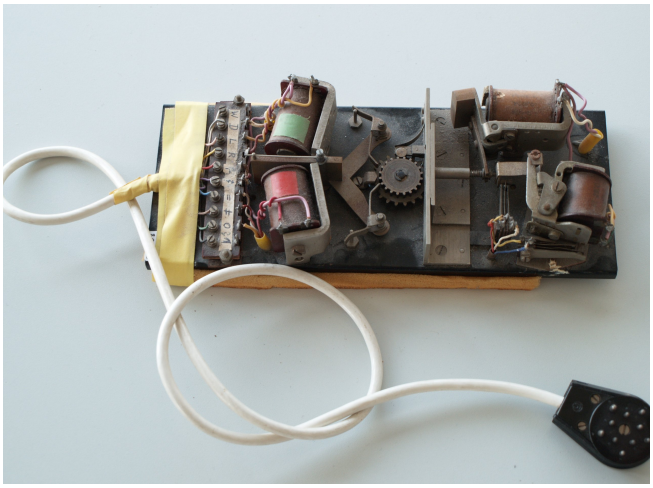


Fig. 20. Tape Drive *Alte Wang*

memory is close to a state table. This allowed a very compact program encoding (see example for the later *Mini-Wang* on page 39), otherwise such a machine would run too slow for even the smallest example.

Using the *Alte Wang*, Rödding had the insight that a single mark on an otherwise entirely blank tape could be used to store a number without modifying the tape at all, by just advancing the tape the given number of steps. Effectively, a number could be added and subtracted, but the total only read destructively. As Hasenjaeger wrote ([125, p.184]):

> But I think the fact that one counter ... suffices ... did suggest to D. Rödding to consider the possiblility of computing any recursive function by using only a bounded number of counting registers.

No consolidated documentation of this machine was found in Hasenjaeger's paper legacy [118], only several undated notes that could belong to the *Alte Wang*.

9.3 The Mini-Wang

The *Old Wang* was quite flexible, but bulky. This might have been Hasenjaeger's motivation to build another, smaller machine, with a fixed behaviour and as few states as possible in the state machine. He called this machine the *Mini-Wang*, see Fig.21.

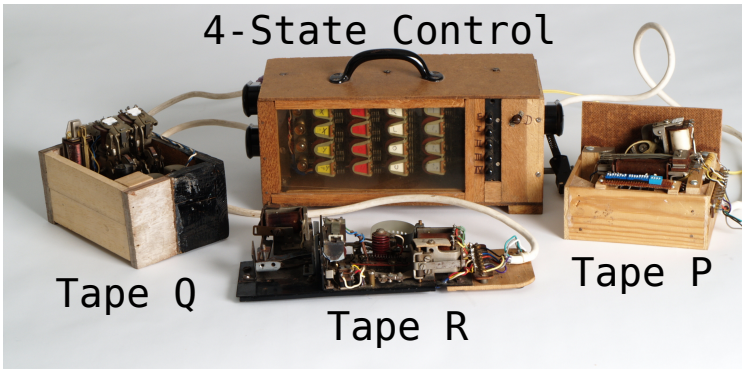


Fig. 21. Mini-Wang

It is a universal machine with a central fixed state machine and three tapes, labelled P, Q and R:

- Tape `P`, containing program instructions
- Tape `Q`, a pure counter tape
- Tape `R`, the result tape (also for initial values)

Tape `P` is the program tape for the (encoded) instructions. It is a cyclic tape of 18 bits, using a selector switch[35] to sense 18 blue little DIP-switches that represent the program.

Tape `Q` is the skip counter for the program tape and equivalent to a (cyclic) tape with only one mark. Two selector switches are connected back-to-back, so that their equal position can be sensed modulo the number of positions. One is used for forward movement, the other one for backward movements.

Tape `R` (Fig.22) is the working and result tape. It uses the same 35mm halved contact sheet paper film as in the *Alte Wang*, but a different method to mark the tape. Instead of punching a (round) hole in the middle, a notch is punched at the upper end, and a lever is released vertically to sense the notch, Fig.23.

In contrast to the tapes of the *Alte Wang*, this tape drive was apparently made by Hasenjaeger himself. Note that the sprocket wheel comes from Meccano, sold in Germany by *Märklin*.

[35] also known as Strowger switch after its inventor

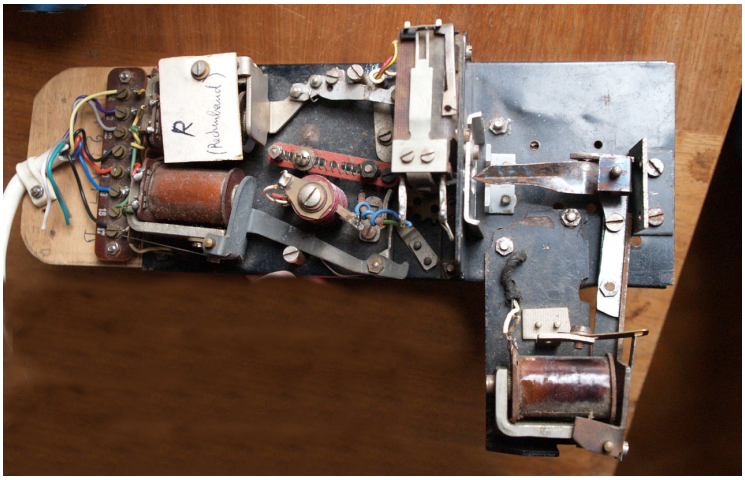


Fig. 22. Mini-Wang working tape

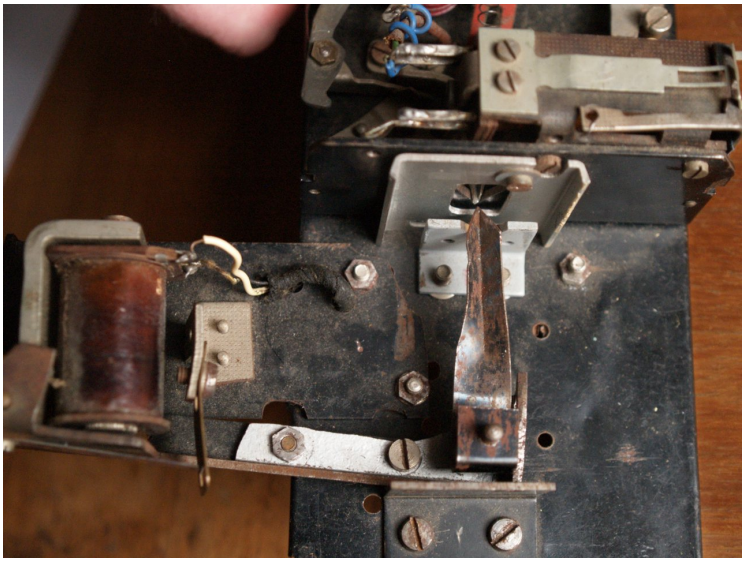


Fig. 23. Mini-Wang punch mechanism

The state machine uses 16 relays, which are not relays commonly used by the German post office.[36] The relays are properly orientated so that the gap between the contacts is vertical to avoid dust pile up.[37]

In Summer 2011, the second author reverse-engineered the machine to obtain the schematics shown in Fig.24. Four relays, labelled X, X', Y and Y' form two conventional flipflops for four states. Four relays are for the clock generator. The remaining 8 relays make up 4 Master-Slave flip-flops labelled R, L, O and U.

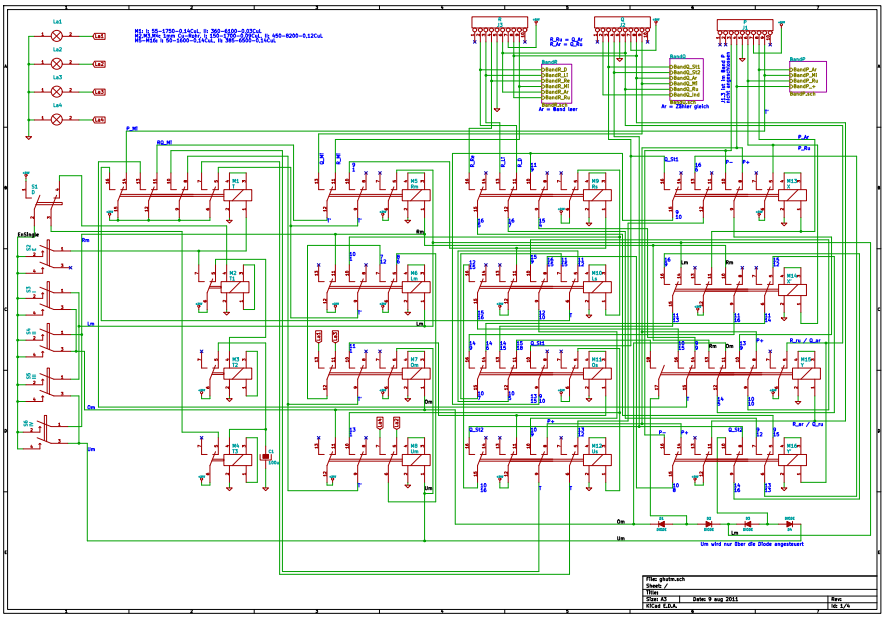


Fig. 24. Mini-Wang schematics

From the schematics, the state table was recovered, but found to be dubious, because the tape P would require repeats as forward jumps depending on the cyclicity of the tape.

[36] One reason might be that it is hard to find 16 relays with the same coils; in the second author's own collection of more than 200 post office relays, he was happy to find for a gray counter 5 such relays. Circuits were massively optimized for component count, ignoring the repair costs.

[37] This has been one of the major technological steps for larger reliability in German telephone exchanges using the Flachrelais 24. Zuse in his Z3 rebuild has the gap horizontally.

The recovered state table was:

```
Z       PQR     Z"      action

I       0._     II      P+
        0.*     II      P+
        1._     .       P+    M
        1.*     .       P+

II      0!.     III     P+ Q-
        00.     .       P+ Q+
        1!.     I       P+ Q- L
        10.     I       P+    R

III     0._     .       P+
        0.*     .       P+ Q+
        1._     I       P+
        1.*     IV      .  Q+

IV      0!.     .       P-
        00.     I       P+
        1!.     .       P- Q-
        10.     I       P+
```

As there was a unidirectional tape found with the others, in the action column there was originally no distintion between `P+` and `P-`. The instructions to be coded on tape `P` were

```
1       M   mark the tape
01      R   right move
001     L   left move
000<sup>n</sup>1   n   skip if the tape is marked
```

The conditional skip was originally assumed to be forward, using the cyclicity of the tape, which however made finding progammes rather difficult, as there are no neutral instructions (no-ops) to fill the tape.

The solution came when a bidirectional selector switch in its original boxing was found in Hasenjaeger's home in Plettenberg. Re-interpreting the schematics, it was clear that the machine was built for a bidirectional programme-tape, which was then created from the new switch and could be alternatively used, see Fig.25.

The Wang like encoding with variable length instructions requires in this case a minimum of 4 states to count the number of zeroes until it is clear that it is a skip. The `Q`-Tape may be used as a state extension and reduce the number of states to 3, which does not help to reduce the number of relays in practice.

A different punching tool was found and a small stock of already cut tape; both worked flawlessly at 24V, while using celloloid film required to increase the supply.

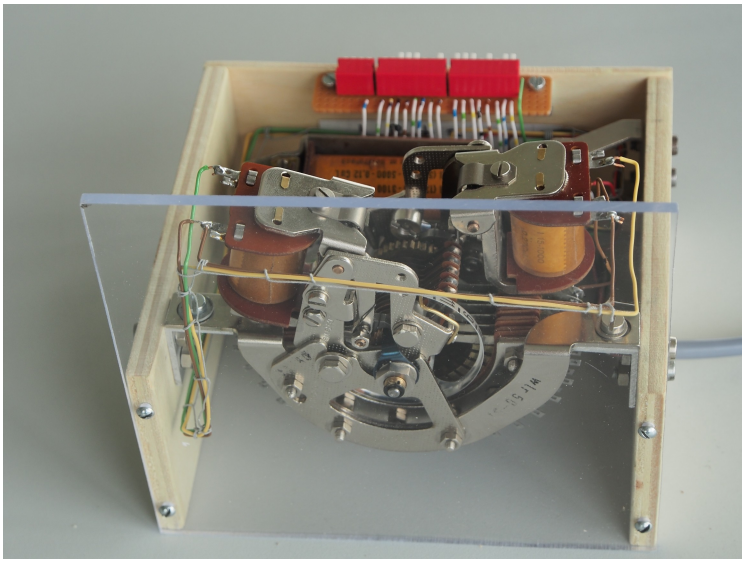


Fig. 25. Bidirectional progamme tape

So the above skip is finally a skip-back-if-marked, which corresponds to the repeat operator in Rödding's register machines, only the proper nesting is not enforced.

A programme to find the next space to the right and mark it (15 bits):

```
L   R  1     M 0
001 01 00001 1 0001
```

The first `L` is for the case that the tape is on a space. Then a tight loop moves right, checks for a mark, and if marked, repeats. If not marked, punches a mark, and a zero jump is a *dynamic stop*, which does jump infinitely, because the tape is marked.

More details are available. The formerly published descriptions (e.g. [127], [128]) are not up to date.[38]

To estimate the compactnesss of multi-tape Turing machines, the second author has proposed a Turing machine index ([126]) which is 24.0 (basic index) for this machine.

A proof that this machine is universal and also computationally efficient can be found in [218].

[38] Try [http://rclab.de/] for more details.

9.4 The RTL/70

A Box labelled *RTL/70* was found in his legacy in Plettenberg (see Fig.26), containing a base module and four pluggable modules, labelled `S`, `R`, `Q` and `P` Thus it looks like a 4-tape machine.

Fig. 26. RTL 70 in transport box and assembled

Used are early integrated digital circuits of the very early *Resistor-Transistor-Logic* (RTL) by Motorola (`HEP570` to `HEP584`), for which not datasheets were found in the WEB.

An early attempt to reverse engineer the machine and re-build it using SMD replacements for the RTL ICs, was abandoned. Later, some documentation and the corresponding Motorola catalog was found, but the task was not restarted.

It might be suspected that this was a machine materializing the 2-state machine described in Hasenjaegers 1984 report ([138]), but that machine used a set of R-tapes, and this machine in its tailored box seems to be complete and intended for demonstrations.

The machine is very interesting because it is compact; its relevance can only be determined if analysed more extensively. The TM index cannot be guessed as the machine features are too vague.

9.5 The 1984 machine

Hasenjaeger gave the state table of a 2-state machine with 2 bits per instruction in [138]. It uses a `P`- `Q`- and a set of `R`-tapes, that could be cyclically selected, a technique he had persued since the *Alte Wang*.

The instructions on the P-tape are uniformly 2 bits long:

- E: enlarge (increment) the current R-tape
- D: decrement the current R-tape
- C: cyclically change to the next R-tape
- F: for *if*: conditional jump

As two F-instructions in a row are useless, the jump distance is encoded by the corresponding number of contiguous F-instructions.

The state table has been rearranged for better readability:

```
S PRJ   rj s    Remark
                Group 1: sequential instructions
0 c.0   #. .    cycle tape
  d10   -. .    decrement register
  e.0   +. .    increment register
  f.0   .. 1    start jumping
                Group 2: executing a jump
0 c.1   .- .    decrement J for c
  d.1   .- .    decrement J for d
  f.1   .. .    skip  f
                Group 3: do not jump as R is zero
1 c0.   .. 0    terminated by c
  d0.   .. 0    terminated by d
  f0.   .. .    ignore f
                Group 4: collect jump distance
1 c1.   .. 0    found c
  d1.   .. 0    found d
  f1.   .+ .    count number of 'f's
```

According to the text, the programme tape is assumed to by cyclic. As this is apparently a Turing machine simulating a register machine, backward jumps would be more appropriate, but require a different state table, that might nevertheless still have 2 states.

Until now, the only purpose of this machine seems to demonstrate the use of *Jones-Matiyasevich-Masking* to formally describe such a machine, which succeeded in an astonishing short proof.

The TM index is not as small as it seems, because the states of the tape multiplexer must be multiplied with the two visible states. The result tape is a single binary tape for the basic index, and there are 3+2 operations, thus the basic TM index is 6*sqrt(8*5/2) = 26.8. Penalties for the cyclic programme tape would be necessary. With backward jumps, there are two more actions in the state table, so the basic TM index is 6*sqrt(8*7/2) = 31.7, with no penalties.

9.6 The TTL machines

When TTL logic ICs became available, they were much cheaper than Motorola's RTL logic, and thus Hasenjaeger switched to this technology.

He created a large amount of modules that are connected using cables with 2mm plugs, see Fig.27.

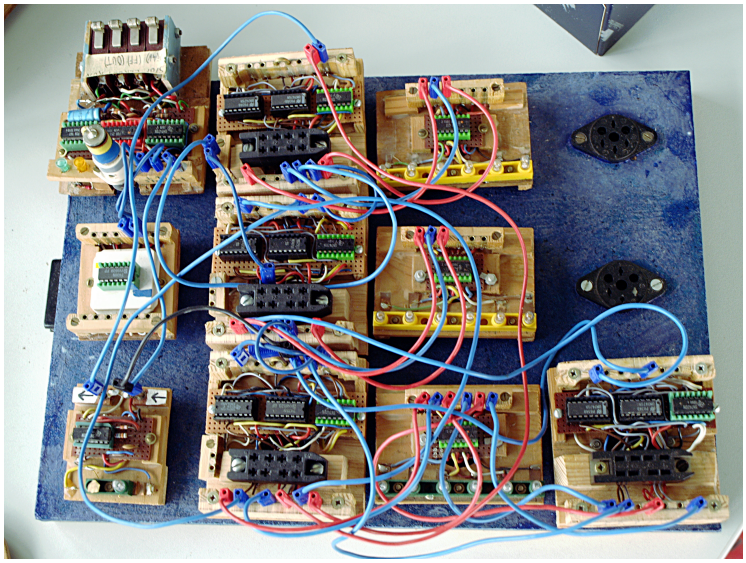


Fig. 27. TTL modules.jpg

Ground and power supply was from the bottom, thus the cables were only needed for signals.

No further analysis and inventory of the modules has been carried out so far.

9.7 More artefacts

A photo from the first *Turingraum* shows an object with old German telephone relays, (Fig. 28) which apparently is a shift register where the state is temporarily stored in capacitors in the front. Such shift registers could well serve as Turing tapes; if it is a ring counter, where only one relay is active ever, it is a Turing tape with just one mark, to be used as count register.

Similar artefacts are preserved but were not yet analyzed; some of these look like (another) relay shift register (see Fig. 29).

Fig. 28. Turingraum with relay shift register

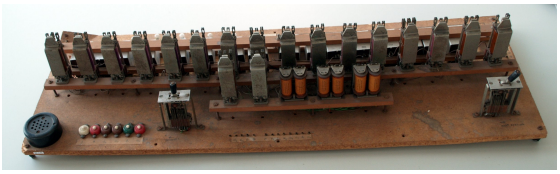


Fig. 29. Relay shift register

Acknowledgement. We thank the following persons who have helped with criticism, suggestions, information, pictures: Volker Claus, Peter Päppinghaus, Andreas Podelski, Walburga Rödding, Uwe Schöning, Inge Schwank. We are particularly thankful to Elmar Cohors-Fresenborg who pointed us to most of the material in Sect. 6, and to Jonathan Bowen who after the presentation of his historical analysis of the community that has developed itself around the Abstract State Machines method (see [58,57]) suggested to write a companion paper that analyses the influence Turing's epochal 1937 paper had on the *Schule von Münster*, a community formed by activities that are focussed on the relations between logic and computing science.

10 Appendix: The Genealogy of the School of Münster

The Logic School of Münster in half a century had 65 doctoral students, listed below, and 1,353 descendants (data (not completely reliable) from [221], consulted on August 4 and November 25, 2021, with slight corrections due to direct knowledge of the first author, see in particular [32]).

- **Heinrich Scholz**' 11 doctoral students in Münster (1,265 descendants):

Candidate	Year	Students	Descendants
Anna Holling	1930		
Friedrich Bachmann	1934	26	314
Walter Kinder	1935		
Hermann Schweitzer	1935		
Eugen Roth	1937		
Hans Hermes	1938	28	398
Shih-hua Hu	1939		
Karl Schröter	1941	16	263
Eduard Arens	1944		
Gisbert Hasenjaeger	1950	11	378
Werner Markwald	1952		

- **Hans Hermes**' 20 doctoral students in Münster.

Candidate	Year	Students	Descendants
Heinz Gumin	1954		
Arnold Oberschelp	1957	7	20
Walter Oberschelp	1958	20	257
Ludwig Brinkmann	1961		
Horst Burwick	1961		
Klemens Döpp	1961	3	13
Walther Heinermann	1961	1	1
Herbert Fiedler	1962		
Dieter Titgemeyer	1962		
Klaus Brockhaus	1963		
Paul Röver	1963		
Joachim Hornung	1964		
Laurent Larouche	1964		
Jürgen Genenz	1965		
Friedrich-Karl Mahn	1965		
Walburga Schwering	1965		
Giorgio Germano	1966	1	1
Joachim Bammert	1967		
Heinz-Dieter Ebbinghaus	1967	7	47
Klaus Rödding	1967		

- In Freiburg Hermes had the following 8 doctoral students: Reiner Durchholz (1968), Robert Kerkhoff (1968), Michael Richter (1968), Hubert Schwarz (1968), Jörg Flum (1969), Dieter Klemke (1970), Klaus Heidler (1973), Walther Kindt (1973).
  In total Hermes had 28 doctoral students and 398 descendants.
- **Gisbert Hasenjaeger**'s 11 doctoral students (378 descendants):

Candidate	Year	Students	Descendants
Dieter Rödding	1961	15	251
Ronald Jensen	1964	13	92
Ulrich Perret	1968		
Ibrahim Garro	1972		
Wilhelm Johannes Backhausen	1973		
Gerda Thieler-Mevissen	1974		
Tassilo von der Twer	1976		
Ralf Bülow	1980		
Dimitrios Christodoulakis	1980		
Peter Schroeder-Heister	1981	6	22
Emile Weydert	1988		

- **Dieter Rödding**'s 15 doctoral students (251 descendants):

Candidate	Year	Students	Descendants
Helmut Schwichtenberg	1968	16	25
Michael Deutsch	1968		
Thomas Ottmann	1971	22	78
Jürgen Bartnick	1971		
Egon Börger	1971	4	6
Elmar Cohors-Fresenborg	1971	11	14
Hansjürgen Brämik	1972		
Hans-Georg Carstens	1972	22	58
Helmut Müller	1974		
Lutz Priese	1974		
Peter Körber	1976		
Hans Kleine Büning	1977	9	13
Klaus-Peter Kniza	1980		
Joachim Müller	1982		
Anne Brüggemann-Klein	1985	2	2

References

1. A.Blass, E.Börger, and Y.Gurevich, editors. *Theory and Application of Abstract State Machines*. Schloss Dagstuhl, 2002. Seminar Report 336. https://www.dagstuhl.de/02101.
2. J.-R. Abrial. *The B-Book*. Cambridge University Press, Cambridge, 1996.
3. J.-R. Abrial. *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, Cambridge, 2010.
4. A.Church. A note on the Entscheidungsproblem. *J. of Symbolic Logic*, 1:40–41, 1936.
5. A.Church. An unsolvable problem of elementary number theory. *American J. of Mathematics*, 58:345–363, 1936.
6. W. Ackermann. Zum Hilbertschen Aufbau der reellen Zahlen. *Mathematische Annalen*, 99:118–133, 1928.
7. W. Ackermann. *Solvable Cases of the Decision Problem*. North–Holland, 1954.
8. W. Ackermann. Von den natürlichen zu den reellen Zahlen. Lecture Notes, Institut für math. Logik und Grundlagenforschung, Winter Term 1961/2.
9. A.Clausing. Heinrich Scholz' early interest in Turing's papers. https://ivv5hpp.uni-muenster.de/u/cl/. Consulted July 11, 2021.
10. A.Cobham. The intrinsic difficulty of functions. In *Proc.1964 Congress for Logic, Mathematics, and Philosophy of Science*, pages 24–30, 1964.
11. A.Durand, D. Jones, J. Makowsky, and M. More. Fifty years of the spectrum problem: survey and new results. *Bull. Symbol. Logic*, 18:505–553, 2012.
12. A.Grzegorczyk. Some classes of recursive functions. *Rozprawy Matematiyczne IV*, IV:3–45, 1953.
13. A.Hodges. *Alan Turing: The Enigma*. Simon and Schuster, 1983.
14. H.-C. S. am Busch and K.F.Wehmeier. "Es ist die einzige Spur, die ich hinterlasse". Dokumente zur Entstehungsgeschichte des Instituts für Mathematische Logik und Grundlagenforschung. In H.-C. S. am Busch and K.F.Wehmeier, editors, *Heinrich Scholz. Logiker, Philosoph, Theologe*, pages 93–101. Mentis (Paderborn), 2005.
15. A.Mostowski. Concerning a problem of H. Scholz. *Zeitschr. f. math. Logik u. Grundlagen d. Math.*, 2:210–214, 1956.
16. A.M.Turing. Computing machinery and intelligence. *Mind*, LIX(236):433–460, 1937. https://doi.org/10.1093/mind/LIX.236.433.
17. A.M.Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937. https://doi.org/10.1112/plms/s2-42.1.230.
18. A.Raschke, E.Riccobene, and K.-D.Schewe, editors. *Logic, Computation and Rigorous Methods*, volume 12750 of *LNCS*. Springer-Verlag, 2021. Essays Dedicated to Egon Börger on the Occasion of His 75th Birthday.
19. A.R.Meyer and D.M.Ritchie. Computational complexity and program structure. IBM Watson Research Center at Yorktown Heights, Research Report RE–1817, p.1-15, 1967.
20. J. V. Atanasoff. Computing machine for the solution of large systems of linear algebraic equations. In B. Randell, editor, *The Origins of Digital Computers*, pages 305–325. Springer-Verlag, 1973.
21. A.Tarski. Der Wahrheitsbegriff in den formalisierten Sprachen. *Studia Philosophica*, 1:261–405, 1936.

22. A. Blass and Y. Gurevich. Abstract State Machines capture parallel algorithms. *ACM Trans. Computational Logic*, 4(4):578–651, 2003.
23. A. Blass and Y. Gurevich. Abstract State Machines capture parallel algorithms: Correction and extension. *ACM Transactions on Computation Logic*, 9(3), 2008.
24. F. Boniol, V. Wiels, Y.Ait-Ameur, and K.-D. Schewe, editors. *ABZ 2014: The Landing Gear Case Study*, volume 433 of *Communications in Computer and Information Science*. Springer, 2014.
25. E. Börger. Brief an G. Hasenjaeger 04.10.1985. Anlage: von M. Richter verfasste Dokumentation zur Rödding-Nachfolge. See Hasenjaeger Nachlass, Deutsches Museum München, NL 288 / 149.
26. E. Börger. On the constrution of simple first-order formulae without recursive models. In *Proc. Coloquio sobra logica simbolica*, pages 9–24, Madrid, 1975. Universidad Complutense.
27. E. Börger. A new general approach to the theory of the many-one equivalence of decision problems of algorithmic systems. volume 25, pages 135–162, 1979. Also published as vol.30 of R.Kaerkes and L.Merkwitz and W.Oberschelp: Schriften zur Informatik und Angewandten Mathematik, RWTH Aachen.
28. E. Börger. Decision Problems in Predicate Logic. In G.Lolli, G.Longo, and A.Marcja, editors, *Logic Colloquium'82*, pages 263–301. North-Holland, Studies in Logic and the Foundations of Mathematics vol.112, 1984.
29. E. Börger. Spektralproblem and completeness of logical decision problems. In E. Börger, G. Hasenjaeger, and D.Rödding, editors, *Logic and Machines: Decision Problems and Complexity*, pages 333–356. Springer LNCS 171, 1984.
30. E. Börger. *Berechenbarkeit,Komplexität,Logik*. Vieweg Verlag Braunschweig, 1985. 2nd ed.1986, 3d extended edition 1991, engl.translation Computability, Complexity, Logic (vol. 128 of Studies in Logic and the Foundations of Mathematics, North-Holland 1989), italian transl. Computabilità, Complessità, Logica vol.1: Teoria delle Computazione, Serie di Informatica, Bollati Borighieri 1989.
31. E. Börger, editor. *Computation Theory and Logic. In memory of Dieter Rödding*. Springer LNCS 270, 1987.
32. E. Börger. D.Rödding: Ein Nachruf. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 89:144–148, 1987.
33. E. Börger. Logic as Machine: Complexity relations between programs and formulae. In E. Börger, editor, *Trends in Theoretical Computer Science*, pages 59–94, 1988. A survey of the main results was presented to the centenary Scholz-Festkolloquium held at the logic institute in Münster on February 8–9, 1985.
34. E. Börger. Complexity of logical decision problems. In G. Corsi, M. Chiara, and G. Ghirardi, editors, *Bridging the Gap: Philosophy, Mathematics, and Physics*, pages 71–86. Kluwer Academic Publisher, 1993.
35. E. Börger. Ten years of CSL conferences (1987-1997). *EATCS Bulletin*, 63:61–63, 1997.
36. E. Börger. The ASM ground model method as a foundation of requirements engineering. In N.Dershowitz, editor, *Verification: Theory and Practice*, volume 2772 of *LNCS*, pages 145–160. Springer-Verlag, 2003.
37. E. Börger. The ASM refinement method. *Formal Aspects of Computing*, 15:237–257, 2003.
38. E. Börger, H. Büning, G.Jäger, S. Martini, and M.Richter, editors. *CSL'92*, volume 702 of *Lecture Notes in Computer Science*. Springer, 1993.
39. E. Börger, H. Büning, G.Jäger, and M.Richter, editors. *CSL'91*, volume 626 of *Lecture Notes in Computer Science*. Springer, 1992.

40. E. Börger, H. Büning, and M.Richter, editors. *CSL'87*, volume 329 of *Lecture Notes in Computer Science*. Springer, 1988.
41. E. Börger, H. Büning, and M.Richter, editors. *CSL'88*, volume 385 of *Lecture Notes in Computer Science*. Springer, 1989.
42. E. Börger, H. Büning, and M.Richter, editors. *CSL'89*, volume 440 of *Lecture Notes in Computer Science*. Springer, 1990.
43. E. Börger, H. Büning, M.Richter, and W.Schönfeld, editors. *CSL'90*, volume 533 of *Lecture Notes in Computer Science*. Springer, 1991.
44. E. Börger, E.Grädel, and Y.Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer-Verlag, 1997. Second printing in "Universitext", Springer-Verlag 2001.
45. E. Börger and U. Glässer. Abstract State Machines 2001: New developments and applications. In E. Börger and U. Glässer, editors, *J. Universal Computer Science*, volume 7(11), pages 914–917. Springer-Verlag, 2001. Selected extended papers from 8th international ASM workshop.
46. E. Börger, G. Hasenjaeger, and D.Rödding, editors. *Logic and Machines: Decision Problems and Complexity*, volume 171. Springer LNCS, 1984.
47. E. Börger and J. Huggins. Abstract State Machines 1988–1998: Commented ASM bibliography. *Bull. EATCS*, 64:105–127, 1998.
48. E. Börger and U. Löwen. Logical decision problems and complexity of logic programs. *Fundamenta Informaticae*, 10:1–34, 1987.
49. E. Börger, M.Butler, J. P.Bowen, and P.Boca, editors. *Abstract State Machines, B and Z*, volume 5238 of *Lecture Notes in Computer Science*. Springer, 2008. First International Conference ABZ 2008.
50. E. Börger and A. Raschke. *Modeling Companion for Software Practitioners*. Springer, 2018. ISBN 978-3-662-56641-1. For Corrigenda and lecture material on themes treated in the book see http://modelingbook.informatik.uni-ulm.de.
51. E. Börger and K.-D. Schewe. Concurrent Abstract State Machines. *Acta Informatica*, 53(5), 2016. http://link.springer.com/article/10.1007/s00236-015-0249-7, DOI 10.1007/s00236-015-0249-7. Listed as Notable Article in ACM 21th Annual BEST OF COMPUTING, see www.computingreviews.com/recommend/bestof/notableitems.cfm?bestYear=2016.
52. E. Börger and K.-D. Schewe. A behavioral theory of recursive algorithms. *Fundamenta Informaticae*, 177(1):1–37, 2020. DOI 10.3233/FI-2020-1915.
53. E. Börger and J. Schmid. Composition and submachine concepts for sequential ASMs. In P. Clote and H. Schwichtenberg, editors, *Computer Science Logic (Proceedings of CSL 2000)*, volume 1862 of *Lecture Notes in Computer Science*, pages 41–60. Springer-Verlag, 2000.
54. E. Börger and R. F. Stärk. *Abstract State Machines. A Method for High-Level System Design and Analysis*. Springer, 2003.
55. E. Börger, Y.Gurevich, H. K. Büning, and M.Richter. Computer Science Logic. 1992. Dagstuhl Seminar Report 40 (9229), https://www.dagstuhl.de/fileadmin/files/Reports/92/9229.pdf.
56. E. Börger, Y.Gurevich, and K.Meinke, editors. *CSL'93*, volume 832 of *Lecture Notes in Computer Science*. Springer, 1994.
57. J. P. Bowen. ABZ 2021 conference report. *FACS FACTS*, 2021(2):65–70, July 2021.
58. J. P. Bowen. Communities and ancestors associated with Egon Börger and ASM. In A. Raschke, E. Riccobene, and K.-D. Schewe, editors, *Logic, Computation and Rigorous Methods*, volume 12750 of *Lecture Notes in Computer Science*, pages 96–120. Springer, 2021.

59. A. Brüggemann. *Stochastische Zuverlässigkeit fehlertoleranter Netzwerke*. PhD thesis, Inst.math.Logik und Grundlagenforschung, Universität Münster, 1985. The scientific advisor of this dissertation has been D.Rödding, but the thesis was submitted after Rödding's death so that the Mathematics Genealogy Project does not associate it with D.Rödding.
60. A. Brüggemann, L.Priese, D.Rödding, and R.Schätz. Modular decomposition of automata. In E.Börger, G.Hasenjaeger, and D.Rödding, editors, *Logic and Machines: Decision Problems and Complexity*, pages 198–236. Springer Lecture Notes in Computer Science vol.171, 1984.
61. M. J. Butler, K.-D. Schewe, A. Mashkoor, and M. Biro, editors. *Abstract State Machines, Alloy, B, TLA, VDM, and Z - 5th International Conference ABZ 2016*, volume 9675 of *Lecture Notes in Computer Science*, Linz (Austria), 2016. Springer.
62. C.Boehm and G.Jacopini. Flow diagrams, Turing machines and languages with only two formation rules. *Communications of the ACM*, 9:366–371, 1966.
63. C.Carstensen. Eine schaltalgebraische Realisierung von Registermaschinen. Prüfungsarbeit zur Ersten Staatsprüfung für das Lehramt an Realschulen, 1975.
64. C.Christen. *Spektren und Klassen elementarer Funktionen*. PhD thesis, ETH Zürich, 1974.
65. A. K. Chandra and D.Harel. Computable queries for relational data bases. *J. Computer and System Sciences*, 21:156–178, 1980.
66. A. K. Chandra and D.Harel. Structure and complexity of relational queries. *J. Computer and System Sciences*, 25:99–128, 1982. https://doi.org/10.1016/0022-0000(82)90012-5.
67. C.Kaune. Das Wissen um Unterschiede in den kognitiven Strukturen von Schülerinnen und Schülern als Erklärung von Unterrichtsbeiträgen. *Zentralblatt für Didaktik der Mathematik*, 35:102–109, 2003.
68. E. Cohors-Fresenborg. Registermaschine. email of Nov 1 to Egon Börger.
69. E. Cohors-Fresenborg. On the representation of algorithmic concepts. In F.Lowenthal and F.Vandamme, editors, *Pragmatics and Education*, Boston (MA), 1986. Springer. https://doi.org/10.1007/978-1-4757-1574-3_13.
70. E. Cohors-Fresenborg. Individual differences in cognitive structures and the effect on business reengineering. In *Proceedings of the IV European Congress of Psychology*, pages 153–160, Göttingen, 1996.
71. J. Copeland. Turing's great invention: the universal computing machine. In J. Copeland, J. Bowen, M. Sprevak, and R. Wilson, editors, *The Turing Guide*, 2017. DOI:10.1093/oso/9780198747826.003.0013.
72. M. Deutsch. Ein neuer Beweis und eine Verschärfung für den Reduktionstyp $\forall\exists\forall^{\infty}(0,1)$ mit einer Anwendung auf die spektrale Darstellung von Prädikaten. *Zeitschrift für math. Logik und Grundlagen der Math.*, 38:559–564, 1992.
73. D.Hilbert and W.Ackermann. *Grundzüge der theoretischen Logik*. Springer, 1928,1938. English translation of the 2nd edition: Principles of Mathematical Logic, Chelsea Publishing Company, New York (1950).
74. B. Dreben and W. Goldfarb. *The decision problem: solvable cases of quantificational formulas*. Addison-Wesley, 1979.
75. D.Rödding. *Darstellungen der (im Kalmár-Csillagschen Sinne) elementaren Funktionen*. PhD thesis, Inst. für math. Logik und Grundlagenforschung, Universität Münster, 1961. Presented at the International Congress of Math., Stockholm 1962, and published in Arch. Math. Logik Grundlagenforsch. 7 (1965) 139–158.

76. D.Rödding. Theorie der Rekursivität über dem Bereich der endlichen Mengen von endlichem Rang. Habilitationsschrift, Institut für math. Logik und Grundlagenforschung, 1964.
77. D.Rödding. Klassen rekursiver Funktionen. In M. H. Löb, editor, *Proceedings of the Summer School in Logic (Leeds 1967)*, pages 159–222, 1968.
78. D.Rödding. Einführung in die Theorie der berechenbaren Funktionen. Lecture Notes (written by E.Börger), Institut für math. Logik und Grundlagenforschung, 1969. Reviewed in Mathematical Reviews (number 56 # 15384a/b).
79. D.Rödding. Höhere Prädikatenlogik: Interpolationstheorem, Reduktionstypen. Lecture Notes (written by H.Schwichtenberg), Institut für math. Logik und Grundlagenforschung, 1969.
80. D.Rödding. Reduktionstypen der Prädikatenlogik. Lecture Notes (written by E.Börger), Institut für math. Logik und Grundlagenforschung, 1970. Reviewed in Mathematical Reviews 57 (number 2903) and Zentralblatt für Mathematik 267 (number 02034).
81. D.Rödding. Einführung in die Theorie der berechenbaren Funktionen I. Lecture Notes (written by P. Koerber), Institut für math. Logik und Grundlagenforschung, 1972. Summer Term 1972.
82. D.Rödding. Einführung in die Theorie der berechenbaren Funktionen II. Lecture Notes (written by P. Päppinghaus), Institut für math. Logik und Grundlagenforschung, 1973. Winter Term 1972/73.
83. D.Rödding. Klasseneinteilungen im Bereich der rekursiven Funktionen. Lecture Notes, Institut für math. Logik und Grundlagenforschung, Winter term 1964/65.
84. H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 1995.
85. H.-D. Ebbinghaus, J. Flum, and W.Thomas. *Mathematical Logic*. Springer, 1995.
86. E.Börger. *Reduktionstypen in Krom- und Hornformeln*. PhD thesis, Institut für math. Logik und Grundlagenforschung, Universität Münster, 1971.
87. E.Börger. Reduktionstypen der klassischen Prädikatenlogik, Teil 1. Lecture Notes, Institut für math. Logik und Grundlagenforschung, 1972.
88. E.Börger. The origins and the development of the ASM method for high level system design and analysis. *J.Universal Computer Science*, 8:2–74, 2002.
89. E.Börger, editor. *Abstract State Machines and high-level system design and analysis*, volume 336 (2–3) of *Theoretical Computer Science (Special Issue)*. Elsevier, 2005. ISSN 0304–3975. Selection of extended papers from ASM'03 (Taormina, Sicily).
90. E.Börger, editor. *The Abstract State Machines method*, volume 77 of *Fundamenta Informaticae (Special Issue)*. IOS Press, 2007. ISSN 0169–2968. Selection of extended papers from ASM'05 (Paris).
91. E.Börger, A.Gargantini, and E.Riccobene, editors. *Abstract State Machines 2003. Advances in Theory and Practice*, volume 2589 of *LNCS*. Springer, 2003. Contains Proceedings of 10th ASM Workshop (Taormina, Italy). For a selection of extended workshop papers see [89].
92. E.Börger and A.Prinz, editors. *Quo vadis Abstract State Machines?*, volume 14 (12) of *J. Universal Computer Science (Special Issue)*. 2008. Selection of extended papers from ASM'07 (Grimstadt, Norway).
93. E.Börger, D. Beauquier, and A. Slissenko. Proc. 12th international workshop on Abstract State Machines ASM'05. Université Paris 12 (France), 2005. For a selection of extended workshop papers see [90].
94. E.Cohors-Fresenborg. *Subrekursive Funktionsklassen über binären Bäumen*. PhD thesis, Inst.math.Logik und Grundlagenforschung, Universität Münster, 1971.

95. E.Cohors-Fresenborg. Berechenbare Funktionen und Registermaschinen Ein Beitrag zur Behandlung des Funktionsbegriffs auf konstruktiver Grundlage. *Didaktik der Mathematik*, 3:187 –209, 1973.
96. E.Cohors-Fresenborg. Dynamische Labyrinthe. *Didaktik der Mathematik*, 1:1–21, 1976.
97. E.Cohors-Fresenborg. *Mathematik mit Kalkülen und Maschinen*. Vieweg, Braunschweig, 1977.
98. E.Cohors-Fresenborg. Verschiedene Repräsentationen algorithmischer Begriffe. *Journal für Mathematikdidaktik*, 6:187–209, 1985.
99. E.Cohors-Fresenborg and B.Reimers. Ein Demonstrationsmodell für Registermaschinen. *Der Mathematische und Naturwissenschaftliche Unterricht (MNU)*, XXVIII, 1975.
100. E.Cohors-Fresenborg, D.Finke, and S.Schütte. Dynamische Labyrinthe. *Osnabrücker Schriften zur Mathematik*, 1979. English version 11–19, Dutch version 31–39, also translated to Chinese and Indonesian.
101. E.Cohors-Fresenborg, M. Griep, and I. Schwank. Registermaschinen und Funktionen–Ein Schulbuch zur Einführung des Funktionsbegriffs auf der Grundlage von Algorithmen. *Osnabrücker Schriften zur Mathematik*, 22, 1979.
102. E.Cohors-Fresenborg and C. Kaune. *Von Anweisungen zu Funktionen*. Forschungsinstitut für Mathematikdidaktik e.V., Osnabrück, 2012. 3d revised edition.
103. E.Cohors-Fresenborg, C. Kaune, and M. Griep. *Einführung in die Computerwelt mit Registermaschinen*. Forschungsinstitut für Mathematikdidaktik e.V., Osnabrück, 1995.
104. E.Cohors-Fresenborg, S.Brinkschmidt, and S.Armbrust. Augenbewegungen als Spuren prädikativen oder funktionalen Denkens. *Zentralblatt für Didaktik der Mathematik*, 35:86–93, 2003.
105. E.Cohors-Fresenborg and I. Schwank. On the modelling of learning processes by $\alpha\beta\gamma$ - automata. In *Proc.7th International Congress of Logic, Methodology and Philosophy of Science*, pages 24–27, 1983.
106. E.Cohors-Fresenborg and I. Schwank. Kognitive Aspekte des Business Reengineering. *Gestalt Theory*, 18:233–256, 1996.
107. E.Cohors-Fresenborg and I. Schwank. Individual differences in the managerial mental representation of business processes. In R. P. et al., editor, *Managerial Behaviour and Business Processes: European Research Issues*, pages 93–106, Louvain, 1997.
108. E.Engeler. Algorithmic properties of structures. *Math. Systems Theory*, 1:183–195, 1967.
109. E.G.Wagner. Uniformly reflexive structures: on the nature of Gödelizations and relative computability. *Transac.American Mathematical Society*, 144:1–41, 1969.
110. T. Eichholz. Semantische Untersuchungen zur Entscheidbarkeit im Prädikatenkalkül mit Funktionsvariablen. *Archiv für math. Logik u. Grundlagenforschung*, pages 19–28, 1957.
111. F.-K.Mahn. *Über die Strukturunabhängigkeit des Begriffs der primitiv-rekursiven Funktionen*. PhD thesis, Institut für math. Logik und Grundlagenforschung, Universitä Münster, 1965.
112. F.-K.Mahn. Primitiv-rekursive Funktionen auf Termmengen. *Arch. math. Logik*, 12:54–65, 1969.
113. F. Ferrarotti, K.-D. Schewe, L. Tec, and Q. Wang. A new thesis concerning synchronised parallel computing – simplified parallel ASM thesis. *Theor. Comp. Sci.*, 649:25–53, 2016.

114. F.L.Bauer. The formula-controlled logical computer *Stanislaus*. *Mathematics of Computation*, 14:64–67, 1960.
115. M. Fürer. Alternation and the Ackermann case of the decision problem. *L' Enseignement Mathématique*, II:137–162, 1982.
116. G. Gabriel, H.Hermes, F. Kambartel, C.Thiel, and A.Veraart, editors. *Gottlob Frege. Wissenschaftlicher Briefwechsel*. Felix Meiner Verlag Hamburg, 1976.
117. G.Asser. Das Repräsentantenproblem im Prädikatenkalkül der ersten Stufe mit Identität. *Zeitschr. f. math. Logik u. Grundlagen d. Math.*, 1:252–263, 1955.
118. G.Hasenjaeger. Nachlass. Deutsches Museum München, Archiv, NL 288.
119. G.Hasenjaeger. Register-Maschinen. Deutsches Museum München, Archiv, NL 288/081. See [123].
120. G.Hasenjaeger. Einführung in die Mengenlehre. Lecture Notes (written by Hans-Rüdiger Wiehle) at University of Münster, 1953/4.
121. G.Hasenjaeger. *Einführung in die Grundbegriffe und Probleme der modernen Logik*. Alber, Freiburg and Munich, 1962. Engl.translation Introduction to the basic concepts and problems of modern logic, D. Reidel Publishing Company, Dordrecht, Holland, and Humanities Press, New York, 1972.
122. G.Hasenjaeger. Rekursive Funktionen. Lecture Notes (written by G. Seebach, Tassilo von der Twer and Jutta Klucken) at University of Bonn, 1971/2.
123. G.Hasenjaeger. Registermaschinen. *Contact*, 14 and 15, 1976.
124. G.Hasenjaeger. Zur Vor- und Frühgeschichte des (bis heute so genannten) "Know-How- Computers". pages 1–4, 1984. Apparently unpublished (but most of the material went into [125]). See Hasenjaeger Nachlass, Deutsches Museum München, NL 288 / 089.
125. G.Hasenjaeger. On the early history of register machines. In *Computation Theory and Logic*, volume 270 of *Lecture Notes in Computer Science*, pages 181 – 188. Springer, 1987.
126. R. Glaschick. A size index for multi tape Turing machines. Isaac Newton Institute Cambridge preprint, 18.7.2018. https://www.newton.ac.uk/files/preprints/ni12061\0.pdf.
127. R. Glaschick. Alan Turings Wirkung in Münster. *Mitteilungen der Deutschen Mathematiker-Vereinigung*, 20(1):42–48, 2012. https://doi.org/10.1515/dmvm-2012-0019.
128. R. Glaschick. Turing machines in Münster. In S. B. Cooper and J. van Leeuwen, editors, *Alan Turing: His Work and Impact*. Elsevier, 2013. ISBN 9780123869807.
129. E. Grädel. *The Complexity of Subclasses of Logical Theories*. PhD thesis, Universität Basel, 1987. For a summary see Bulletin of the EATCS vol. 34 (1988), 289–291.
130. E. Grädel. Subclasses of Presburger Arithmetic and the Polynomial-Time Hierarchy. *Theoretical Computer Science*, 56:289–301, 1988.
131. E. Grädel. Dominoes and the complexity of subclasses of logical theories. *Annals of Pure and Applied Logic*, 43:1–30, 1989.
132. E. Grädel. Size of models versus length of computations. On inseparability by nondeterministic time complexity classes. In *Proceedings of the Second Workshop on Computer Science Logic CSL 88, Duisburg 1988*, LNCS 385, pages 118–137. Springer, 1989.
133. E. Grädel. On logical descriptions of some concepts in structural complexity theory. In *Proceedings of the Third Workshop on Computer Science Logic CSL 89, Kaiserslautern 1989*, LNCS 440, pages 163–175. Springer, 1990.
134. Y. Gurevich. A new thesis. *Abstracts, American Mathematical Society*, 6(4):317, August 1985. abstract 85T-68-203.

135. Y. Gurevich. Evolving algebras 1993: Lipari Guide. In E. Börger, editor, *Specification and Validation Methods*, pages 9–36. Oxford University Press, 1995.
136. Y. Gurevich. Sequential Abstract State Machines capture sequential algorithms. *ACM Trans. Computational Logic*, 1(1):77–111, July 2000.
137. D. Harel and A. Pnueli. On the development of reactive systems. In K.Apt, editor, *Logics and models of concurrent systems*, pages 477–498. Springer-Verlag New York, 1985.
138. G. Hasenjaeger. Universal Turing machines (UTM) and Jones-Matiyasevich-masking. In E. Börger, G.Hasenjaeger, and D.Rödding, editors, *Logic and Machines: Decision Problems and Complexity*, volume 171 of *Lecture Notes in Computer Science*, pages 248–253. Springer Berlin / Heidelberg, 1984.
139. H.Brämik. *Beweistheoretische Charakterisierung der ω_3-rekursiven Funktionen*. PhD thesis, Inst.math.Logik und Grundlagenforschung, Universität Münster, 1972.
140. H.Bruderer. Wie erfuhr die ETH Zürich von der Zusemaschine Z4? https://doi.org/10.3929/ethz-a-010001419, 2013.
141. H.Friedman. Algorithmic procedures, generalized Turing algorithms, and elementary recursion theory. volume 61 of *Studies in Logic and the Foundations of Mathematics*, pages 361 – 389. 1971.
142. H.Hermes. Definite Begriffe und berechenbare Zahlen. *Semesterberichte zur Pflege des Zusammenhangs von Universität und Schule aus den mathematischen Seminaren*, pages 110–123, 1937.
143. H.Hermes. *Eine Axiomatisierung der allgemeinen Mechanik*. PhD thesis, Universität Münster, 1938. Published in Leipzig as Heft 3 of Forschungen zur Logik und zur Grundlegung der exakten Wissenschaften.
144. H.Hermes. Maschinen zur Entscheidung von mathematischen Problemen. *Mathematisch-Physikalische Semesterberichte (Göttingen)*, pages 179–189, 1952.
145. H.Hermes. Die Universalität programmgesteuerter Rechenmaschinen. *Mathematisch-Physikalische Semesterberichte (Göttingen)*, pages 42–53, 1954.
146. H.Hermes. *Vorlesung über Entscheidungsprobleme in Mathematik und Logik*. Aschendorffsche Verlagsbuchhandlung, 1955. Vol.15 of Ausarbeitungen mathematischer und physikalischer Vorlesungen.
147. H.Hermes. *Aufzählbarkeit - Entscheidbarkeit - Berechenbarkeit. Einführung in die Theorie der rekursiven Funktionen*. Springer-Verlag, 1961. Various editions, english translation 1965, spanish translation INTRODUCCION A LA TEORIA DE LA COMPUTABILIDAD. See also the manuscript [146].
148. H.Hermes. *Einführung in die mathematische Logik - Klassische Prädikatenlogik*. Teubner Verlag, 1963. Second extended edition 1969, english translation Introduction to Mathematical Logic 1973.
149. H.Hermes. *Eine Termlogik mit Auswahloperator*. Springer-Verlag Berlin, 1965. Vol.6 of Lecture Notes in Mathematics. English translation Term Logic with Choice Operator in 1970.
150. H.Hermes. In memoriam WILHELM ACKERMANN 1896-1962. *Notre Dame Journal of Formal Logic*, VIII:1–8, 1967.
151. H.Hermes. Entscheidungsprobleme und Dominospiele. In K. Jakobs, editor, *Selecta Mathematica II*, pages 114–140. Springer, 1970.
152. H.Hermes. A simplified proof for the unsolvability of the decision problem in the case $\forall\exists\forall$. In R. Gandy and C. Yates, editors, *Selecta Mathematica II*, pages 307–310, Amsterdam, 1971. North-Holland.

153. H.Hermes. Logistik in Münster um die Mitte der Dreissiger Jahre. In H.Dollinger, editor, *Logik und Grundlagenforschung. Festkolloquium zum 100. Geburtstag von HEINRICH SCHOLZ*, volume 8 (Neue Folge), pages 41–52. Schriftenreihe der Westfälischen Wilhelms-Universität Münster, 1986.
154. H.Hermes and D.Rödding. A method for producing reduction types in the restricted lower predicate calculus. In *Formal Systems and Recursive Functions*, pages 42–47, Oxford, 1965.
155. H.Hermes and H.Scholz. Mathematische Logik. In *Enzyklopädie der math. Wissenschaften Vol. I, 1.1*, page 82, Leipzig, 1952. Teubner.
156. H.Hermes, F. Kambartel, and F. Kaulbach, editors. *Gottlob Frege. Nachgelassene Schriften*. Felix Meiner Verlag Hamburg, 1969.
157. H.Lewis. Complexity results for classes of quantificational formulas. *Journal of Computer and System Sciences*, 21:317–353, 1980.
158. H.Müller. *Klassifizierungen der primitiv-rekursiven Funktionen*. PhD thesis, Inst.math.Logik und Grundlagenforschung, Universität Münster, 1974.
159. H.Scholz. Ein ungelöstes Problem in der symbolischen Logik. *Journal of Symbolic Logic*, 17:160, 1952.
160. H.Scholz and G.Hasenjaeger. *Grundzüge der Mathematischen Logik*. Springer Verlag, 1961.
161. H.Schwichtenberg. *Eine Klassifikation der mehrfach-rekursiven Funktionen*. PhD thesis, Inst.math.Logik und Grundlagenforschung, Universität Münster, 1968.
162. H.Schwichtenberg. Rekursionszahlen und die Grzegorczyk-Hierarchie. *Archive f. math. Logik u. Grundlagenforschung*, 12:85–97, 1969.
163. H.Schwichtenberg. Eine Klassifikation der ϵ_0-rekursiven Funktionen. *Zeitschrift f. math. Logik u. Grundlagen d. Math.*, 17:61–74, 1971.
164. H.Schwichtenberg and S.S.Wainer. *Proofs and Computations*. Cambridge University Press, 2011. ISBN: 9780521517690.
165. H.Wang. A variant of Turing's theory of computing machines. *J. ACM*, 4:63–92, 1957.
166. I.Schwank. *Präferenzgesteuerte $\alpha\beta\gamma$-Automaten*. PhD thesis, Universität Osnabrück, 1984.
167. J.-R.Abrial and U.Glässer. Rigorous Methods for Software Construction and Analysis. 2006. https://www.dagstuhl.de/06191. See Proceedings [168].
168. J.-R.Abrial and U.Glässer, editors. *Rigorous Methods for Software Construction and Analysis. Essays Dedicated to Egon Börger on the Occasion of his 60th Birthday*. Springer, 2009. Lecture Notes in Computer Science vol. 5115.
169. J.A.Makowsky. Some thoughts on computational models: from massive human computing to Abstract State Machines. In *Logic, Computation and Rigorous Methods. Essays Dedicated to Egon Börger on the Occasion of His 75th Birthday*, Springer Lecture Notes in Computer Science vol.12750, pages 173–186, 2021.
170. J.Bartnick. *Eine algebraisch-kombinatorische Darstellung der Prädikatenlogik*. PhD thesis, Institut für math. Logik und Grundlagenforschung, Universitä Münster, 1971. The scientific advisor of this dissertation has been D.Rödding, but the thesis is not listed in the Mathematics Genealogy Project [221].
171. J.Bennett. *On spectra*. PhD thesis, Princeton University, 1962.
172. J.Büchi. Turing machines and the Entscheidungsproblem. *Mathematische Annalen*, 148:201–213, 1962.
173. J.Derrick, J.Fitzgerald, S.Gnesi, S.Khurshid, M.Leuschel, S.Reeves, and E.Riccobene, editors. *Abstract State Machines, Alloy, B, VDM, and Z - Third International Conference ABZ 2012*, volume 7316 of *Lecture Notes in Computer Science*, Pisa (Italy), 2012. Springer.

174. J.Elstrodt and N. Schmitz. Geschichte der Mathematik an der Universität Münster. Teil II:1945–1969, Kap.7: Ehemalige Professoren 1945 - 1969. http://www.math.uni-muenster.de/historie/. Consulted July 11, 2021.
175. R. Jensen. Ein neuer Beweis für die Entscheidbarkeit des einstelligen Prädikatenkalküls mit Identität. *Archiv math. Logik u. Grundlagenforschung*, 7:128–138, 1962.
176. J.Genenz. Reduktionstheorie des Entscheidungsproblems im Prädikatenkalkül der ersten Stufe nach der Methode von Kahr-Moore-Wang, 1965. Diplomarbeit, Universität Münster.
177. J.Genenz. *Untersuchungen zum Entscheidungsproblem im Prädikatenkalkül der ersten Stufe*. PhD thesis, Inst.math.Logik und Grundlagenforschung, Universität Münster, 1965.
178. J.Müller. *Die mathematische Behandlung von Präferenz und Tausch unter Zugrundelegung des Automatenbegriffs*. PhD thesis, Inst.math.Logik und Grundlagenforschung, Universität Münster, 1982. The scientific advisor of this dissertation has been D.Rödding, but the thesis is not listed in the Mathematics Genealogy Project [221].
179. J. Jones and Yu.V.Matijasevich. Register machine proof of the theorem on exponential diophantine representation of enumerable sets. *Journal of Symbolic Logic*, 49:818–829, 1984.
180. N. Jones and A. Selman. Turing machines and the spectra of first-order formulas. *Journal of Symbolic Logic*, 39:139–150, 1974.
181. J.Suranyi. *Reduktionstheorie des Entscheidungsproblems im Prädikatenkalkül der ersten Stufe*. Verlag der Ungarischen Akademie der Wissenschaften (Budapest), 1959.
182. K.-D.Schewe and F.Ferrarotti. Behavioural theory of reflective algorithms I: Reflective sequential algorithms. arXiv:2001.01873. submitted 2020.
183. K.-P.Kniza. *Automaten und rekursive Funktionale endlichen Typs*. PhD thesis, Inst.math.Logik und Grundlagenforschung, Universität Münster, 1980.
184. K.Erk and L.Priese. *Theoretische Informatik. Eine umfassende Einführung*. Springer, 2000.
185. K.Heidler. *Untersuchungen zur Reduktionstheorie des Entscheidungsproblems in der Prädikaten- und Termlogik*. PhD thesis, Universität Freiburg, 1973.
186. K.Heidler, H.Hermes, and K.Mahn. *Rekursive Funktionen*. Bibliographisches Institut-Wissenschaftsverlag, Mannheim, Wien, Zürich, 1977.
187. H. Kleine Büning. *Über Probleme bei homogener Parkettierung von Z Z durch Mealy-Automaten bei normierter Verwendung*. PhD thesis, Inst. für math. Logik und Grundlagenforschung, Universität Münster, 1977.
188. H. Kleine Büning. Some undecidable theories with monadic predicates and without equality. *Archiv math. Logik u. Grundlagenforschung*, 21:137–148, 1981.
189. H. Kleine Büning. Complexity of LOOP-problems in normed networks. In *Logic and Machines: Decision Problems and Complexity*, Springer LNCS 171, pages 254–269, 1984.
190. H. Kleine Büning and T. Lettmann. Classes of first order formulas under various satisfiability definitions. In *8th International Conference on Automated Deduction (CADE 1986)*, Springer LNCS 230, pages 553–563, 1968.
191. H. Kleine Büning and T. Lettmann. *Aussagenlogik: Deduktion und Algorithmen*. Teubner, 1994.
192. H. Kleine Büning and L.Priese. Universal asynchronous iterative arrays of Mealy automata. *Acta Informatica*, 13, 1980.

193. H. Kleine Büning and Th.Ottmann. Kleine universelle mehrdimensionale Turingmaschinen. *J. Inf. Process. Cybern.*, 13:179–201, 1977.
194. K.Rödding. *Zur Klasseneinteilung der rekursiven Funktionen nach Kleene*. PhD thesis, Inst. math. Logik und Grundlagenforschung, Universität Münster, 1967.
195. K.Schröter. *Ein allgemeiner Kalkülbegriff*. PhD thesis, Universität Münster, 1941.
196. K.Schröter. *Axiomatisierung der Fregeschen Aussagenkalküle*. PhD thesis, Universität Münster, 1943. Habilitationsschrift.
197. L.Blum, M.Shub, and S.Smale. On a theory of computation and complexity over the real numbers. *Bulletin American Math.Society*, 21:1–46, 1989.
198. H. Lewis. *Unsolvable Classes of Quantificational Formulas*. Addison-Wesley, 1979.
199. L.Priese. *Über einfache unentscheidbare Probleme: Computational- und constructional universelle asynchrone cellulare Räume*. PhD thesis, Inst. math. Logik und Grundlagenforschung, Universität Münster, 1974. The scientific advisor of this dissertation has been D.Rödding, but the thesis is not listed in the Mathematics Genealogy Project [221].
200. L.Priese. On the minimal complexity of component-machines for self-correcting networks. *Journal of Cybernetics*, 5:97–118, 1975.
201. L.Priese. On a simple combinatorial structure sufficient for sublying non-trivial self-reproduction. *Journal of Cybernetics*, 6:101–137, 1976.
202. L.Priese. On stable organization of normed networks. In *Proceedings of the third European meeting on cybernetics and systems research*, pages 381–394, 1976.
203. L.Priese. Reversible Automaten und einfache universelle 2-dimensionale Thuesysteme. *Zeitschr.f.math.Logik und Grundlagen der Math.*, 22:353–384, 1976.
204. L.Priese. Normed networks: Their mathematical theory and applicability. In *Applied General Systems Research*, NATO Conference Series vol.5, pages 381–394, 1978.
205. L.Priese. Towards a precise characterization of the complexity of universal and nonuniversal Turing machines. *SIAM J. Computing*, 8:508–523, 1979.
206. L.Priese. Modular implementation of concurrency. *International Journal of Theoretical Physics*, 21:993–1005, 1982.
207. L.Priese. On the concept of simulation in asynchronous, concurrent systems. *Progress in Cybernetics and Systems Research*, II, 1982. Proceedings of European Meeting on Cybernetics and Systems Research (Linz 1978).
208. L.Priese. Automata and concurrency. *Theor. Comp. Sci.*, 25:221–265, 1983.
209. L.Priese and D.Rödding. A combinatorial approach to self-correction. *J.Cybernetics*, 4:7–24, 1974.
210. M.Butler, A.Raschke, T.S.Hoang, and K.Reichel, editors. *Abstract State Machines, Alloy, B, TLA, VDM, and Z*, volume 10817 of *Lecture Notes in Computer Science*. Springer, Southampton (UK), 2018. 6th International Conference ABZ 2018.
211. M.Davis. *Computability and Unsolvability*. New York, 1958.
212. M.Davis, R.Sigal, and E.Weyuker. *Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science (2nd ed.)*. Elsevier Science and Technology, San Francisco (US), 1994. ISBN10 0122063821,ISBN13 9780122063824.
213. M.Deutsch. *Normalformen aufzählbarer Prädikate*. PhD thesis, Inst.math.Logik und Grundlagenforschung, Universität Münster, 1968.
214. M.Frappier, U.Glässer, S.Khurshid, R.Laleau, and S.Reeves, editors. *Abstract State Machines, Alloy, B, and Z - Second International Conference ABZ 2010*, volume 5977 of *Lecture Notes in Computer Science*, Orford,QC (Canada), 2010. Springer.

215. M. Minsky. Recursive unsolvability of Post's problem of 'tag' and other topics in the theory of Turing machines. *Annals of Mathematics*, 74:437–455, 1961.
216. M.Mölle and I.Schwank. Dimensional complexity and power spectral measures of the eeg during functional versus predicative problem solving. *Brain and Cognition*, 44:547 –563, 2000.
217. E. Moore. A simplified universal Turing machine. *ACM National Meeting (Toronto)*, pages 50–54, 1952. http://doi.acm.org/10.1145/800259.808993.
218. T. Neary, D. Woods, N.Murphy, and R.Glaschick. Wangs B machines are efficiently universal, as is Hasenjaegers small universal electromechanical toy. *J.of Complexity*, 30:634–646, 2014. https://doi.org/10.1016/j.jco.2014.02.003, ISSN 0885-064X.
219. N.Immerman. Descriptive and computational complexity. In J. Hartmanis, editor, *Computational Complexity Theory*, pages 75–91. American Math.Society, 1989.
220. NN. Heinrich Scholz. Biography. https://mathshistory.st-andrews.ac.uk/Biographies/Scholz/. Published at School of Mathematics and Statistics, University of St Andrews, Scotland.
221. NN. Mathematics Genealogy Project. https://genealogy.math.ndsu.nodak.edu/. Consulted July 14, 2021.
222. B. Pehrson and I. Simon, editors. *Technology and Foundations. Information Processing'94*, volume I, Track 4, Stream C: Evolving Algebras, Hamburg (Germany), 1994. Elsevier. Contains Proceedings of First ASM Workshop.
223. P.Koerber. *Untersuchung an sequentiellen, durch normierte Konstruktionen gewonnenen Netzwerken endlicher Automaten.* PhD thesis, Inst.math.Logik und Grundlagenforschung, Universität Münster, 1976.
224. P.Koerber and Th.Ottmann. Simulation endlicher Automaten durch Ketten aus einfachen Bausteinen. *EIK*, 10:133–148, 1974.
225. A. Raschke and D. Méry, editors. *Rigorous State-Based Methods*, volume 12709 of *Lecture Notes in Computer Science*, Ulm (Germany), 2021. Springer. 8th International Conference ABZ 2021.
226. A. Raschke, D. Méry, and F. Houdek, editors. *Rigorous State-Based Methods*, volume 1271 of *Lecture Notes in Computer Science*, Ulm (Germany), 2020. Springer. 7th International Conference ABZ 2020.
227. R.Fagin. *Contributions to the model theory of finite structures.* PhD thesis, University of California, Berkeley, 1973.
228. R.Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R.Karp, editor, *Complexity of Computation*, pages 43–73. SIAM-AMS Proceedings vol.7, 1974.
229. R.Moreno-Diaz and A.Quesada-Arencibia, editors. *Formal Methods and Tools for Computer Science. Eurocast 2001*, Las Palmas (Spain), 2001. IUCTC Universida de Las Palmas de Gran Canaria. Contains Extended Abstracts of 8th ASM Workshop. For selected full workshop papers see [45].
230. D. Rödding. Über die Eliminierbarkeit von Definitionsschemata in der Theorie der rekursiven Funktionen. *Zeitschr. Math. Logik Grundlagen der Mathematik*, 10:315–330, 1964.
231. D. Rödding. Einige äquivalente Präzisierungen des intuitiven Berechenbarbeitsbegriffs. *Math. Unterricht*, 11:21–38, 1965.
232. D. Rödding. Über Darstellungen der elementaren Funktionen II. *Arch. Math. Logik Grundlagenforsch.*, 9:36–48, 1966.
233. D. Rödding. Primitiv-rekursive Funktionen über einem Bereich endlicher Mengen. *Arch. Math. Logik Grundlagenforsch.*, 10:13–29, 1967.

234. D. Rödding. Registermaschinen. *Math.Unterricht*, 18:32–41, 1972.
235. D. Rödding. Modular decomposition of automata. In M.Karpinski, editor, *Proc. FCT-1983 Conference*, Lecture Notes in Computer Science vol.158, pages 394–412. Springer, 1983.
236. D. Rödding. Some logical problems connected with a modular decomposition theory of automata. In M. Richter, E. Börger, W. Oberschelp, B. Schinzel, and W. Thomas, editors, *Computation and proof theory*, Lecture Notes in Mathematics vol.1104, pages 365–388. Springer, 1984.
237. D. Rödding and E. Börger. The undecidability of $\forall\exists\forall(0,4)$ – formulae with binary disjunctions. *Journal of Symbolic Logic*, 39:412–413, 1974.
238. D. Rödding and H.Schwichtenberg. Bemerkungen zum Spektralproblem. *Zeitschr. f. math. Logik u. Grundlagen d. Math.*, 18:1–12, 1972.
239. D. Rödding and W. Rödding. Networks of finite automata. In *Proceedings of the third European meeting on cybernetics and systems research*, Progress in Cybernetics and Systems Research 1979. Hemisphere, Washington D.C., 1976.
240. W. Rödding. Netzwerke abstrakter Automaten als Modelle wirtschaftlicher und sozialer Systeme. *Schriftenreihe der Österreichischen Studiengesellschaft für Kybernetik*, 1975.
241. W. Rödding and H.Nachtkamp. On the aggregation of preferences to form a preference of a system. *Naval Research Logistic Quarterly*, 1978.
242. R.Vobl. Komplexitätsuntersuchungen an Basisdarstellungen endlicher Automaten. Diplomarbeit am Inst. für math. Logik und Grundlagenforschung in Münster, 1980.
243. R.W.Ritchie. Classes of predicatably computable functions. *Transactions American Math.Society*, 106:139–173, 1963.
244. K.-D. Schewe. Computation on structures: Behavioural theory, logic, complexity. In A.Raschke, E. Riccobene, and K.-D. Schewe, editors, *Logic, Computation and Rigorous Methods. Essays Dedicated to Egon Börger on the Occasion of His 75th Birthday*, volume 1275 of *Lecture Notes in Computer Science*, pages 266–282. Springer, 2021.
245. I. Schwank. Cognitive structures of algorithmic thinking. In *Proceedings of the 10th International Conference for the Psychology of Mathematics Education*, pages 404 – 409, 1986.
246. I. Schwank. $\alpha\beta\gamma$-automata realizing preferences. In E.Börger, editor, *Computation Theory and Logic*, volume 270 of *LNCS*, pages 320–333. Springer, 1987.
247. I. Schwank. Maschinenintelligenz: ein Ergebnis der Mathematisierung von Vorgängen Zur Idee und Geschichte der Dynamischen Labyrinthe. In C. Kaune, I. Schwank, and J. Sjuts, editors, *Mathematikdidaktik im Wissenschaftsgefüge: Zum Verstehen und Unterrichten mathematischen Denkens*, pages 30–72. Forschungsinstitut für Mathematikdidaktik in Osnabrück, 2005.
248. S.C.Kleene. General recursive functions of natural numbers. *Math.Ann.*, 112:727–742, 1936.
249. S.C.Kleene. *Introduction to Metamathematics*. North-Holland, 1952.
250. S.Cook. The complexity of theorem-proving procedures. In *Proc. 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.
251. J. Shepherdson and H. Sturgis. Computability of recursive functions. *J. ACM*, 10:217–255, 1963.
252. S.Stein and A.Wegener. Bericht über die Dienstreise nach Münster/W zur Durchsuchung des Korrespondenz-Nachlasses von Prof. Scholz. HNF - Heinz Nixdorf MuseumsForum, 2011.

253. R. F. Stärk, J. Schmid, and E. Börger. *Java and the Java Virtual Machine: Definition, Verification, Validation*. Springer-Verlag, 2001.
254. Th.Ottmann. Einfache universelle mehrdimensionale Turingmaschinen. Habilitationsschrift (Universität Karlsruhe).
255. Th.Ottmann. *Eine Theorie sequentieller Netzwerke*. PhD thesis, Inst.math.Logik und Grundlagenforschung, Universität Münster, 1971.
256. Th.Ottmann. über Möglichkeiten zur Simulation endlicher Automaten durch eine Art sequentieller Netzwerke aus einfachen Bausteinen. *Zeitschrift f.math.Logik und Grundlagen der Mathematik*, 19:223–238, 1973.
257. Th.Ottmann. Arithmetische Prädikate über einem Bereich endlicher Automaten. *Archiv f.math.Logik*, 16, 1974.
258. Th.Ottmann. Eine universelle Turingmaschine mit zweidimensionalem Band. *Elektronische Informationsverarbeitung und Kybernetik*, 11:27–38, 1975.
259. Th.Ottmann. Eine einfache universelle Menge endlicher Automaten. *Zeitschrift f.math.Logik und Grundlagen der Mathematik*, 24, 1978.
260. B. Trakhtenbrot. The impossibility of an algorithm for the decision problem for finite models. *Dokl. Akad. Nauk SSSR*, 70:596–572, 1950. English translation in: AMS Transl. Ser. 2, vol.23 (1963), p.1-6.
261. U.Glässer and P.Schmitt, editors. *Fifth International Workshop on Abstract State Machines*, Magdeburg (Germany), 1998. Otto-von-Guericke-Universität. Contains Proceedings of Fifth International ASM Workshop at Informatik'98.
262. U.Rohde. Computer für Anfänger.Teil 1. *mc*, 5:40–45, 1983.
263. B. van der Waerden. Denken ohne Sprache. In G.Révész, editor, *Thinking and Speaking*, pages 165–174. North-Holland, 1954.
264. W.Heinermann. *Untersuchungen über die Rekursionszahlen rekursiver Funktionen*. PhD thesis, Institut für math. Logik und Grundlagenforschung, Universität Münster, 1961.
265. C.-P. Wirth. A most interesting draft for Hilbert and Bernays Grundlagen der Mathematik that never found its way into any publication, and 2 cv of Gisbert Hasenjaeger. SEKI Working-Paper SWP201701 https://arxiv.org/pdf/1803.01386.pdf, 2017.
266. W.Oberschelp. Hans Hermes 12.2.1912 bis 10.11.2003. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 109:99–109, 2007.
267. D. Woods and T. Neary. The complexity of small universal Turing machines: A survey. *Theoretical Computer Science*, 410:443–450, 2009.
268. W.Rödding. Geschichte des Turingraums. Personal Communication to Egon Börger (November 8, 2021) and Letter of 27.4.2012 to Norbert Ryska from the Heinz Nixdorf MuseumsForum in Paderborn (Germany). Unpublished.
269. W.Schwabhäuser. *Entscheidbarkeit und Vollständigkeit der elementaren hyperbolischen Geometrie*. PhD thesis, Humboldt-Universität Berlin, 1960.
270. W.Schwabhäuser, W.Szmielew, and A.Tarski. *Metamathematische Methoden in der Geometrie*. Springer Verlag, 1983.
271. W.Zimmermann and B.Thalheim, editors. *Abstract State Machines 2004. Advances in Theory and Practice*, volume 3052 of *LNCS*. Springer, 2004. Contains Proceedings of 11th ASM Workshop (Lutherstadt Wittenberg).
272. Y.Ait-Ameur and K.-D.Schewe, editors. *Abstract State Machines, Alloy, B, TLA, VDM, and Z - 4th International Conference ABZ 2014*, volume 8477 of *Lecture Notes in Computer Science*, Toulouse (France), 2014. Springer.
273. Y.Gurevich, P.W.Kutter, M.Odersky, and L.Thiele, editors. *Abstract State Machines. Theory and Applications*, volume 1912 of *LNCS*, Monte Verità (Switzerland), 2000. Springer. Proceedings of 7th International ASM Workshop.

274. K. Zuse. Ansätze einer Theorie des allgemeinen Rechnens unter besonderer Berücksichtigung des Aussagenkalküls und dessen Anwendung auf Relaisschaltungen. https://digital.deutsches-museum.de/item/NL-207-0281/. Proposal for a doctoral dissertation submitted to H.Scholz. Unpublished.
275. K. Zuse. Mathematische Logik und Informatik. In *Proc. GI-5.Jahrestagung*, volume 34 of *Springer Lecture Notes in Computer Science*, pages 57–70, 1975.

Appeared in **Newsletter of the Formal Aspects of Computing Science**
ISSN 0950-1231, published by British Computer Society-FACS Specialist Group.
volume 2022, number 1 (2022), pages 69-129.