NWO Vidi Grant:

Automata Transforming Streams

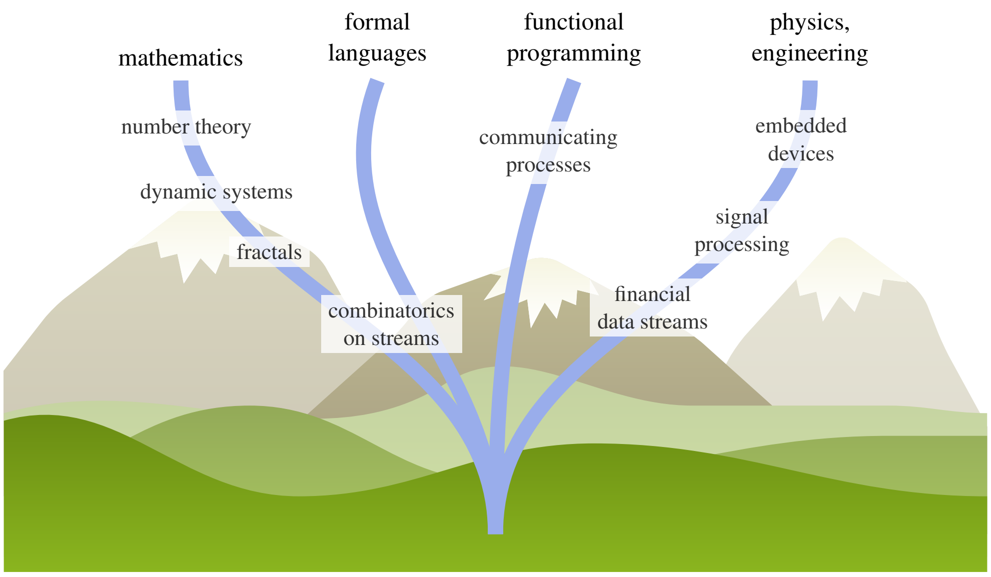
Jörg Endrullis

Vrije Universiteit Amsterdam, Netherlands

Vernieuwingsimpuls 2018
Innovational Research Incentives Scheme
Exacte en Natuurwetenschappen (ENW)

The central objective of this proposal is the study of finite automata for transforming streams.

Streams. In recent times, computer science, logic and mathematics have extended the focus of interest from finite data types to include infinite data types, of which the prototypical example is that of infinite sequences of symbols, or *streams*. Streams are of paramount importance in a wide range of fields, from formal languages to pure mathematics and physics: they appear in functional programming, formal language theory, in the mathematics of dynamical systems, fractals and number theory, in business (financial data streams) and in physics (signal processing). As Democritus in his adagium Panta Rhei already observed, streams are ubiquitous.



Finite automata as transformers. Finite automata play a central role in computer science and mathematics, and have a broad range of applications in industry, ranging from text processing and compiler construction to software and hardware design and verification.

Automata can be used as *acceptors* and *transducers*. As acceptors they simply accept or reject input words, and thereby define a language of accepted words. Transducers, on the other hand, have a much richer output. They transform input words into output words, and thereby realise a function on words.

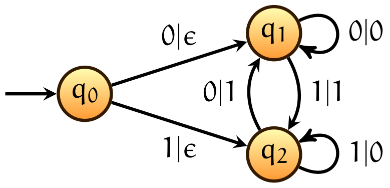


Figure 1 – A finite state transducer realising the difference of consecutive bits modulo 2.

Figure 1 depicts an example of a *finite state transducer* (FST) [5, 112]. A label '$a\,|\,w$' along an edge indicates that the input letter is a and the output word is w. The transducer reads the input stream letter by letter, in each step producing a piece of the output and changing its state. So the total output word is the concatenation of all the output words encountered along the edges.

The transducer given in Figure 1 transforms the input word 011 into the output word 10. When the input is an infinite sequence, a stream, the output of the transducer will be a stream as well. For example, the automaton transforms the Thue–Morse sequence M (see '*Background on streams*' on page 10) to the period doubling sequence PD:

$$\begin{array}{rl} \mathtt{0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ \ldots} & = \mathsf{M} \\ \to\ \mathtt{1\ 0\ 1\ 1\ 1\ 0\ 1\ \ldots} & = \mathsf{PD} \end{array}$$

In a finite state transducer, the output word given by a transition can have arbitrary length (also empty). Thereby FSTs generalise the class of *Mealy machines*; the latter are restricted to output precisely one letter in each step. The transducer shown in Figure 1 is not a Mealy machine, and there exists no Mealy machine implementing this transformation.

State of the art

The state of the art of research on finite automata is summarised in Figure 2.

	automata as acceptors	**automata as transducers**
finite words	well-studied ✓	well-studied ✓
infinite words (streams)	well-studied ✓	hardly studied ✗

Figure 2 – State of the art in automata theory.

Automata on *finite* words have been studied both as acceptors and transducers. The well-known *Chomsky hierarchy* relates classes of languages to different forms of automata and grammars. For instance, *finite state automata*, *pushdown automata* and *Turing machines* accept the classes of *regular languages*, *context-free languages* and *computable languages*, respectively. Also transducers for finite words are well-understood: finite state transducers give rise to the *rational functions*, and Turing machines to the *computable functions*.

For streams, automata have been studied extensively for defining single streams and for defining languages of streams (as acceptors). Concerning the former, the key notion is that of *automatic sequences* [5] presenting deep connections between computer science and number theory. A sequence is k-*automatic* if there exists a finite automaton that computes its n-th element when given the base-k representation of n as input.

Automata for defining languages of streams (acceptors) are known as *stream automata* or ω-*automata*. They play a crucial role in *model checking* and *formal verification* as they allow for describing the (un)acceptable behaviours of non-terminating systems such as operating systems, control systems or hardware. There are various versions of stream automata, in particular *Büchi*, *Muller*, *Rabin* and *parity automata*.

Surprisingly, finite automata for *transforming streams* have hardly been studied, except for Turing machines, for which the ensuing hierarchy of stream degrees is well-known as *Turing degrees* (see below for further information). Beyond Turing machines, there has been almost no research on the power of finite automata, such as *finite state transducers* or *Mealy machines*, for transforming streams. The exceptions are the papers [108, 12, 31]. M. Dekking [31] has shown that every finite state transduct of a morphic stream is again morphic (or finite). G. Rayna [108] and A. Belov [12] have some interesting observations on streams transformations by Mealy machines.

Central objective and methodology

The main goal of this project can be summarised as follows.

Goal The central objective of this project is to develop fundamental theory to reason about the power and limitations of finite automata for transforming streams.

Currently, there is a lack of theory about stream transducers. Even for simple examples of streams, there exist no techniques to determine whether there exists a finite automaton transforming one stream into the other:

Q1 Consider the period doubling sequence PD and drop every third element:

$$\begin{aligned}\mathsf{PD} &= \mathsf{1011\ 1010\ 1011\ 1011\ 1011}\cdots\\ \mathsf{X} &= \mathsf{10\_1\ 1\_10\ \_01\_\ 10\_1\ 1\_11}\cdots\end{aligned}$$

It is easy to find a finite state transducer that transforms PD into X. Is the reverse also possible, or is information irrevocably lost?

Surprisingly, automata theory is not ready to answer such simple questions. Although finite state transducers are very simple and elegant devices, we hardly understand their power for transforming streams. This project aims to develop the necessary theory about finite automata transforming streams. As transformational devices, we consider different well-known automata models:

Goal We envisage to study the power of the following automata models for transforming streams:

(i) finite state transducers (including the variants 'non-erasing' and 'uniform'),

(ii) Mealy machines, and

(iii) pushdown transducers (extension of finite state transducers with a stack).

To achieve these goals, it is crucial to go beyond existing methodologies, in particular by finding connections between mathematics and computer science.

Goal Develop new techniques to reason about finite automata. To this end, we envisage to build bridges between automata theory and different fields of mathematics and computer science.

We have already taken the first steps towards this goal. Although streams and finite state transducers are *discrete* objects, we demonstrated in [53, 54] that techniques from *continuous mathematics* can be used to reason about the power and limitations of finite state transducers. To be precise, we have used the following *methods from linear algebra and analysis*:

(a) continuity,
(b) Vandermonde matrices,

(c) invertibility of matrices, and

(d) the generalised mean inequality.

We have used (a)–(c) to establish that there exist transducers that realise certain transformations on streams. This allowed us to conclude that there is an infinite number of atoms (non-trivial, minimal degrees) in the Transducer degrees (see below). We have employed (d) to reason about limitations of finite state transducers, to show that certain transductions are impossible. For instance, we have shown that the stream $\langle n^3 + n^2 + n \rangle$ cannot be transformed into $\langle n^3 \rangle$. See '*Transducer degrees*' for the definition of streams $\langle \cdots \rangle$.

To our best knowledge, our papers [53, 54] constitute the first application of techniques from continuous mathematics to reason about the power and limitations of finite automata, and thereby pioneers a fruitful bridge between these fields of mathematics and computer science.

Jeffrey Shallit, one of the world authorities in automata theory, highlighted two of our intriguing questions about Transducer degrees during his invited talk at the *British Colloquium for Theoretical Computer Science* in 2014 (see [117] for the slides):

(1) Is the degree of the Thue-Morse sequence an atom?

(2) Are there atom degrees other than that of $10^0 10^1 10^2 10^3 \cdots$?

Our approach (described above) enabled us to answer question (2) by showing that there is an infinite number of atoms. Question (1) remains open and is one of the questions we intend to investigate in this project.

Our general methodology in this project is based on mathematical logic, formal languages and combinatorics on finite and infinite words [19, 14, 15, 85, 69, 105]. We use techniques from the vast culture of automata theory, the backbone of computer science [36, 112, 84, 104].

We frequently define streams equationally. Thus, more operationally, our background is that of term rewriting systems, also infinitary. We have successfully used term rewriting to reason about properties of streams in [41, 38, 47, 45, 46, 39, 142, 43, 49], including the solution of open problems on streams in [48, 62]. In [48] we have solved a long-standing open problem (from 1993) on the subword complexity of words generated by periodically iterating morphisms.

Degrees of Transducibility

Our goal is to understand the power of finite state transducers, Mealy machines and pushdown transducers (and Turing machines) for transforming streams. For each of these machine models we obtain a *hierarchy of stream degrees*, as we explain below. For simplicity, we will speak of

(i) Transducer degrees,
(ii) Mealy degrees,
(iii) Pushdown degrees, and
(iv) Turing degrees, respectively.

Let $\mathcal{M}$ be one of the above machine models. The hierarchy of stream degrees arises as follows:

We obtain a *transducibility relation* $\geq_{\mathcal{M}}$ on the set of streams as follows: for streams u and w we define

$$u \geq_{\mathcal{M}} w \quad\iff\quad u \text{ can be transformed into } w \text{ by a machine from } \mathcal{M}$$

This relation $\geq_{\mathcal{M}}$ is a pre-order, which induces equivalence classes of streams, called *degrees*, and a partial order on these degrees. So a degree is an equivalence class with respect to the equivalence relation $\geq_{\mathcal{M}} \cap \leq_{\mathcal{M}}$.

For instance, the Thue–Morse sequence M and the period doubling sequence PD are equivalent with respect to finite state transduction. Both streams belong to the same degree in the Transducer degrees. We have already seen a FST transforming M into PD, and it is not difficult to find a FST for the reverse transformation.

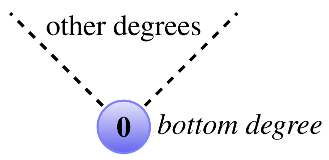
The transducibility relation can be interpreted as a complexity comparison for streams: if the stream u can be transformed into w, then u is *at least as complex as* w, denoted $u \geq_{\mathcal{M}} w$. A degree consists of streams that have the same complexity, meaning that they are transformable into each other.

Goal We envisage to study the transducibility relation $\geq_{\mathcal{M}}$ and the ensuing hierarchies of stream degrees for the machine models mentioned above. Turing degrees have been studied extensively, and will provide us with many guiding analogies.

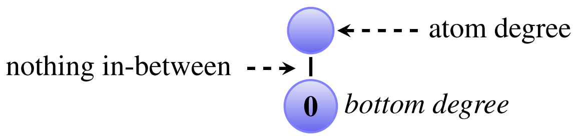
On the one hand, we are interested in the structural properties of these hierarchies. On the other hand, we are aiming at decision methods (or at least heuristic criteria) for determining whether given streams can or cannot be transformed into each other.

To give a better impression of the hierarchies we discuss a few basic properties: bottom degrees and atoms.

The *bottom degree* $\mathbf{0}$, illustrated on the right, is a degree that is less or equal to all other degrees: for all degrees $\mathbf{x}$ we have $\mathbf{0} \leq \mathbf{x}$. For the Turing degrees the bottom degree consists of all computable streams. For the Transducer and Mealy degrees, it consists of all *ultimately periodic streams*, streams of the form $uvvvv\cdots$ for finite words u, v.



An interesting concept is that of an *atom degree*, that is, a degree that is directly above the bottom degree with nothing in-between:



Thus the atom degrees reduce only to **0** or themselves:

(i) For the Transducer degrees, it was an open problem for a while whether there exists more than one atom. In [44, 55], we have proven the existence of a countably infinite number of atom degrees, but it remains an *open* problem whether there are continuum many.

(ii) In the Mealy degrees, there exist no atom degrees, see further [12].

(iii) In the Turing degrees, atoms are usually referred to as *minimal degrees*. A famous result about Turing degrees, obtained by Spector [125], is the existence of an atom degree strictly below the first Turing jump (the degree of the halting problem). Lacombe [119] has extended the construction of Spector to show that there are continuum many atoms in the Turing degrees.

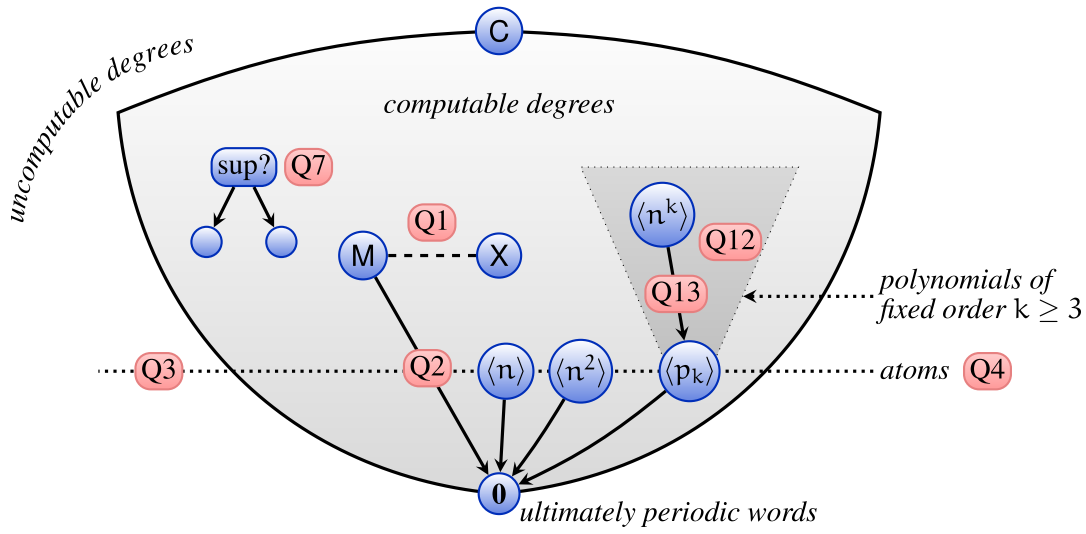


Figure 3 – The partial order of Transducer degrees. Some open problems are indicated in red. The blue nodes are degrees. Note that $\langle n \rangle$ and $\langle n^2 \rangle$ are atoms, while $\langle n^k \rangle$ is not an atom for $k \geq 3$. Here p_k is a particular polynomial of order k. The degree of $\langle p_k \rangle$ is an atom and all other polynomials of order k can be transduced to $\langle p_k \rangle$. The degree of C is the top degree of the computable streams (there are only uncomputable degrees above C).

Due to the space limitations in this proposal, we will focus on *Transducer degrees* in the remainder. Figure 3 sketches some open problems together with initial results about the hierarchy of Transducer degrees that we have already obtained [50, 44, 55, 53, 54]. We use the following notation: for $f : \mathbb{N} \to \mathbb{N}$ we define the stream

$$\langle f \rangle = \prod_{i=0}^{\infty} 0^{f(i)} 1 = 0^{f(0)}\, 1 0^{f(1)}\, 1 0^{f(2)} \cdots .$$

In the sequel we often write $\langle f(n) \rangle$ to denote the sequence $\langle n \mapsto f(n) \rangle$. For instance:

$$\langle n \rangle = 1\ 10\ 100\ 1000\ 10000\ 100000\ \cdots \qquad \langle n^2 \rangle = 1\ 10\ 10000\ 1000000000\ \cdots$$

An initial study of this partial order of degrees has been carried out in [50]. The bottom degree **0** of the hierarchy is formed by the ultimately periodic streams. The hierarchy is not dense, not well-founded (there exist infinite decreasing chains), there exist no maximal degrees, and a set of degrees has an upper bound if and only if the set is countable. Beyond these initial observations, the structure of the Transducer degrees is largely unexplored territory. There is a plethora of interesting open questions:

Q2 Is the degree of Thue-Morse an atom?

Q3 Does there exist a non-computable atom in the Transducer degrees?

Q4 Are there continuum many atoms in the Transducer degrees?

Q5 Are the degrees of Thue-Morse M and Mephisto Waltz W incomparable? Here W is the morphic stream obtained as the limit of iterating the morphism $0 \mapsto 001$, $1 \mapsto 110$ on the starting word 0.

Q6 Does every degree have a minimal cover, that is, a degree directly above with nothing in-between?

Q7 When does a pair of degrees have a least upper (greatest lower) bound?

Q8 Are there infinite ascending (descending) sequences having a least upper (greatest lower) bound?

Q9 Are there interesting dense substructures? Are there dense intervals? That is degrees **a** and **e** with $\mathbf{a} < \mathbf{e}$ such that for all $\mathbf{b}, \mathbf{d}$ with $\mathbf{a} \leq \mathbf{b} < \mathbf{d} \leq \mathbf{e}$ there exists **c** with $\mathbf{b} < \mathbf{c} < \mathbf{d}$.

Q10 Can every finite partial order be embedded in the hierarchy?

Can every finite distributive lattice be embedded in the hierarchy?

Q11 How complex is the first-order theory in the language $\langle \geq, = \rangle$?

Answering these questions will require the development of novel techniques to reason about transducers. In [50, 44, 55] we have carried out the first steps in this direction and have obtained some results on atoms in the Transducer degrees:

The degrees of $\langle n \rangle$ and $\langle n^2 \rangle$ are atoms. But the degree of $\langle n^k \rangle$ is **not** an atom, for any $k \geq 3$.

While the degree of $\langle n^k \rangle$ is not an atom for $k \geq 3$, we found the following. For every $k \geq 1$, there is a unique atom degree among the degrees of polynomials of order k (with non-negative integer coefficients), namely the degree of $\langle p_k(n) \rangle$ where

$$p_k(n) = \sum_{i=0}^{k-1} a_i (kn + i)^k$$

for some $a_0, \ldots, a_{k-1} > 0$. To avoid confusion between two meanings of *degrees*, namely *degrees of streams* and *degrees of polynomials*, we speak of the *order* of a polynomial.

The degree of $\langle p_k \rangle$ is an atom for every $k \geq 1$. Moreover, for every polynomial q of order k we have $\langle q \rangle \geq \langle p_k \rangle$, that is, the stream $\langle q \rangle$ can be transduced to $\langle p_k \rangle$.

These results indicate a rich structure among the degrees of polynomials that we intend to investigate:

Q12 How many degrees exist among polynomials of order k?

Q13 What is the structure of the degrees of polynomials?

Dekking [31] has proven the following important theorem about finite state transduction:

The finite state transduct of a morphic stream is morphic (or finite).

As a consequence, the morphic streams form a subhierarchy of the Transducer degrees. As a matter of fact, many of the sequences, that we are interested in, are morphic. Unfortunately, Dekking's proof gives no insights on the structure of the transducts (beyond being morphic). Such an insight could be very useful for proving non-transducibility between morphic sequences and thereby to answer questions such as Q1 , above. This leads to the following research questions:

Q14 Can Dekking's result be strengthened to characterise the possible shapes of the resulting morphisms?

Q15 Is transducibility ($\geq$) decidable for morphic (or automatic) sequences?

For *non-erasing* (every transition has a non-empty output) finite state transducers, we have shown in [126] that α-substitutivity can be used as a criterion for non-transducibility between morphic sequences. However, this criterion does not generalise to general finite state transduction.

We mention a few more research questions that are interesting to investigate:

Q16 Is every degree **a** the greatest lower bound of a pair of degrees ($\neq$ **a**)?

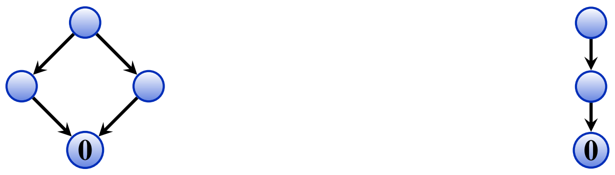


Figure 4 – Are these structures possible in the Transducer degrees?

Q17 Is there a degree that has precisely *two* degrees below itself? This is displayed in Figure 4 on the right.

Q18 Is there a degree that has precisely *three* degrees below itself: two incomparable degrees and the bottom degree? This is displayed in Figure 4 on the left.

Q19 Find suitable notions of Kolmogorov complexity that have relations to the Transducer degrees.

Q20 Establish the relations between the well-known families of streams (automatic sequences, morphic sequences, sturmian sequences, paperfolding sequences, Toeplitz sequences, ...) and the hierarchy of stream degrees.

Motivation

Motivation: degrees of unsolvability

The Transducer degrees are analogous to – but much more fine-grained than – the recursion-theoretic *degrees of unsolvability* or *Turing degrees*. Turing degrees have been the subject of extensive research in the last century with many fascinating results and techniques (Shoenfield [119]). In contrast, very little is known about the power of the finite automata models when it comes to transforming streams.

In the Turing degrees, sets of natural numbers are compared by means of transducibility using Turing machines. The ensuing hierarchy of degrees of unsolvability has been widely studied in the 60's and 70's of the last century and later. These degrees can be seen as a measure of complexity of a set of natural numbers. Note that a set of natural numbers (as the subject of degrees of unsolvability) is also a stream over the alphabet $\{0, 1\}$ via its characteristic function. Thus the degrees of unsolvability can equivalently be considered as a hierarchy of stream degrees. Then we have Turing machines transforming streams into each other.

For a complexity comparison, *Turing machines are too strong*. We are typically interested in computable streams, but they are all identified by transducibility via Turing machines. In the hierarchy of Turing degrees, all computable streams are trivialised in the bottom degree. We are therefore interested in studying transducibility of streams with respect to less powerful devices, such as the concepts of finite automata mentioned above. A reduction of the computational power results in a finer structure of degrees. A finer structure of degrees gives rise to a novel way to compare/measure the *complexity of streams*.

Motivation: constant-space algorithms

Analysing and processing the enormous amount of data generated by various real-world applications (for instance *Internet of Things*) is one of the major challenges of computer science today. This concerns streams of sensor data from continual measurements, streams of financial transactions (stock markets), streams of messages in social media (Twitter or Facebook), and so forth. When it comes to data sets that are massive in size, even linear algorithms may be too complex for processing the data. For instance, think of an algorithm with linear space complexity applied to petabytes of input data. This has led to the rapidly developing research field of *sublinear algorithms* that is concerned with the development of algorithms having sublinear-space and/or sublinear-time complexity.

The most strict form of sublinear-space complexity is *constant-space*. It is important to note that a constant-space algorithm is nothing else than a finite state transducer transforming some input stream into an output stream while using a fixed amount of memory. A constant space-complexity is indispensable for programs that are intended to run indefinitely, to continually transform an endless input stream into an endless output stream. Any algorithm *not* having constant space-complexity will eventually run out of memory when transforming an *infinite* stream. This motivates the study of constant-space algorithms for the transformation of streams.

Background on Streams

Streams are infinite sequences of symbols, also called infinite words, or ω-words. A landmark was the discovery in 1906 by Axel Thue, founding father of formal language theory, of the famous Thue–Morse sequence

$$\mathsf{M} = \mathsf{0110\ 1001\ 1001\ 0110\ 1001\ 0110\ 0110\ 1001} \ldots$$

Thue was interested in avoiding patterns, like squares ww or cubes www. Indeed, the Thue–Morse sequence is cube-free. Interpreting the stream as drawing instructions, this stream gives rise to (via scaling back the approximations and convergence in the Hausdorff metric) a well-known fractal curve, the Koch snowflake. This famous stream, rediscovered in the theory of dynamical systems by mathematician Marston Morse in 1942, is a member of two important families of streams: automatic sequences (defined before) and morphic sequences. The Thue–Morse sequence is a *(purely) morphic* sequence that arises starting from the word $\mathsf{0}$, iterating the morphism $\mathsf{0} \mapsto \mathsf{01}$, $\mathsf{1} \mapsto \mathsf{10}$, obtaining $\mathsf{0} \mapsto \mathsf{01} \mapsto \mathsf{0110} \mapsto \mathsf{01101001} \mapsto \mathsf{0110100110010110} \mapsto \cdots$.

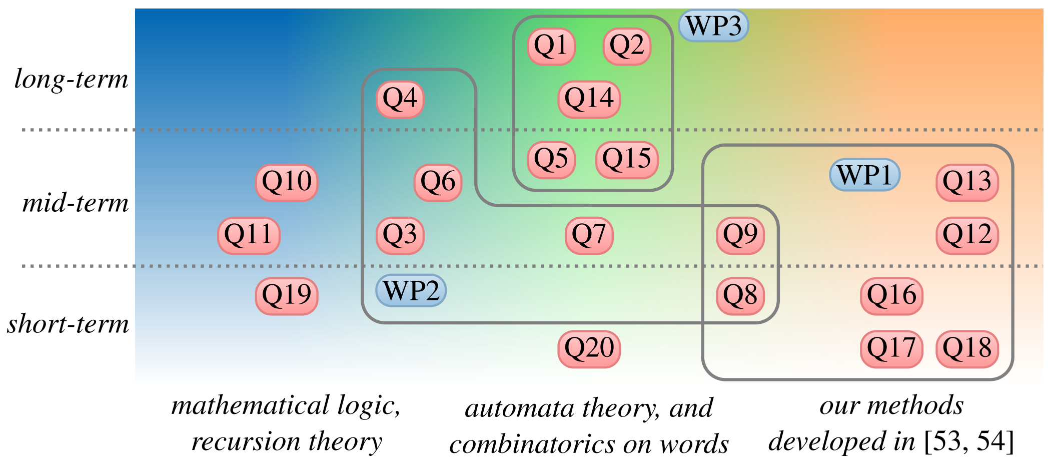
Research team and workplan

The core research team will consist of 2 PhD students, 1 postdoc and the principal investigator (PI). One of the project goals is to stimulate the collaboration of different branches of computer science and mathematics. To this end, we embed the project in the following extended research team:

Team member	**University**	**Expertise**
Dr. Helle Hvid Hansen	TU Delft	coalgebra and coinductive reasoning
Dr. Robbert Fokkink	TU Delft	topology, automata theory and combinatorics
Dr. Femke van Raamsdonk	VU Amsterdam	logic and higher-order term rewriting

To facilitate a close collaboration, the second PhD student will work for 2 days per week at the *TU Delft* with Helle Hansen and Robbert Fokkink; the principal investigator will join for one of the two days. Beyond this national collaboration, we have reserved travel budget for research visits abroad and to invite international researchers to the Vrije Universiteit Amsterdam. In particular, the postdoc is intended to visit the group of Jeffrey Shallit at the University of Waterloo for 2 or 3 months.

The following diagram presents a rough classification of the questions (for finite state transducers) with respect to difficulty (short-term, mid-term, long-term) and with respect to methodology:

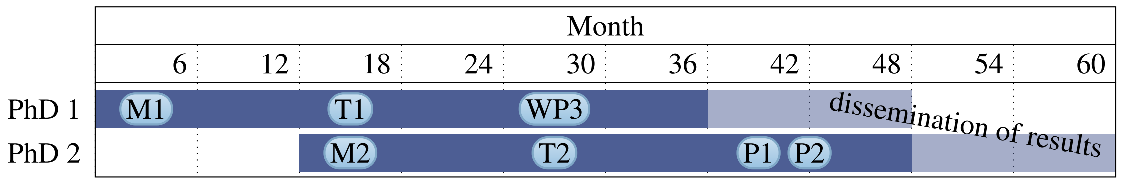

long-term
mid-term
short-term
WP3
Q1
Q2
Q4
Q14
Q5
Q15
WP1
Q13
Q10
Q6
Q11
Q3
Q7
Q9
Q12
Q19
WP2
Q8
Q16
Q20
Q17
Q18
mathematical logic, recursion theory
automata theory, and combinatorics on words
our methods developed in [53, 54]

The diagrams also depicts the main work packages WP1, WP2 and WP3.

– WP1 concerns *global structural properties* of the hierarchy of stream degrees. For these questions we expect that (extensions of) our techniques from [53, 54] are applicable.
– WP2 is about *local structural properties*, that is, what degrees have certain properties (e.g. a minimal cover, or being the greatest lower bound of two other degrees). We intend to tackle these questions using general methodology of mathematical logic, recursion theory and combinatorics on words.
– WP3 concerns the development of criteria (e.g. invariants) for transducibility and non-transducibility (with a particular focus on morphic and automatic streams).

Each work package can profit from insights obtained in the other packages, but they are sufficiently independent to prevent stagnation. Each of the work packages comes in 3 flavours, for instance, the work package WP1 consists of T1 for finite state transducers, M1 for Mealy machines, and P1 for Pushdown transducers.

We envisage the following work plan for the PhD students:



The principal investigator and the postdoc (from year 2 to 3) will be active in all tasks and all activities.

Knowledge utilisation

($\times$) Yes, this proposal has the potential of knowledge utilization.

( ) No, this proposal has no direct knowledge utilization.

This project is fundamental, curiosity driven research. We do not claim direct impact on industrial applications or society. This being said, we wish to point out the following. Most of the groundbreaking results in automata theory have been found by purely curiosity driven research. Kleene invented *regular expressions* in 1951, and Chomsky established the *Chomsky-hierarchy* in 1956, long before regular and context-free languages became indispensable tools in numerous industrial applications (for instance, parsing and text processing). Büchi has studied Büchi automata in 1962, decades before they paved the way to *model checking* for *software and hardware verification*.

The goal of this project is to develop fundamental theory about finite automata for transforming streams. The applications of this research reside mainly inside the research community. The interest of the research community is witnessed by three invited talks on the subject of this proposal given by the principal investigator:

– at the conference *Automatic Sequences, Number Theory, Aperiodic Order 2015* in Delft,
– at the conference *WORDS 2015* in Kiel,
– at the *Fields Workshop - Challenges in Combinatorics on Words 2013* in Toronto.

The importance of the subject has also been recognised by some of the leading researchers in the field of automata theory. As mentioned before, Jeffrey Shallit highlighted some open problems on finite state transducers in his invited talk at the British Colloquium for Theoretical Computer Science 2014.

Building bridges between different areas in computer science and mathematics is an important goal of this project. We have pioneered the use of techniques from continuous mathematics for reasoning about the capabilities and limitations of finite state transducers (see Section '*Central objective and methodology*'). We also strive to establish results in the reverse direction, enabling the use of theory about finite state transducers to derive results in continuous mathematics. Similar connections have been fruitfully exploited in the past: finite automata, giving rise to *automatic sequences*, have been shown to have deep connections to number theory.

Shallit, Goc and Henshall [60, 61] develop impressive tools for automated reasoning about streams properties (for automatic sequences and paperfolding sequences). Among other results, they obtain automatic proofs of challenging open problems, and derive optimal avoidance bounds that improve on the best known bounds in the literature. Similarly, the current research proposal has the potential to contribute to automated reasoning about stream transduction.

In a recent paper [77], Kozen and Soloviev employ a (coalgebraic model) of finite state transducers as stream transformers for constructing and reasoning about efficient reductions between random processes. Their framework allows for analysing the trade-off between latency and entropy in a compositional way. They envisage a research program casting Shannon's information theory in terms of stream transducers. This is an intriguing application of stream transducers, highlighting the importance of understanding these devices. We think that our project will contribute in this direction, and we strive to collaborate with Kozen and Soloviev.

Beyond the research community, teaching automata theory to students can of course profit from illustrative, elegant and deep results about automata as the ones we strive for.

If this project succeeds in gaining new fundamental insights in finite automata, then these results may very well have industrial impact in the long term. In modern computing and information processing applications, the datatype of infinite streams of symbols is of paramount importance since information is often conveyed interactively, in a continual way without termination, by communication with environment stimuli. Applications are ubiquitous in embedded systems, signal processing, financial data streams, and nowadays even in social media analysis where patterns in data streams have to be identified and recognized leading for instance to directed advertising. Indirect applications may even be found in chemistry, where the analysis of quasicrystals turned out to rest ultimately on the understanding of infinite sequences such as sturmian sequences [13].

References

[1] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri. Openfst: A general and efficient weighted finite-state transducer library. In *12th International Conference on Implementation and Application of Automata (CIAA)*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23. Springer, 2007.

[2] J.-P. Allouche. Sur la Complexité des Suites Infinies. *Journées Montoises*, 1(2):133–143, 1994.

[3] J.-P. Allouche and R. Bacher. Toeplitz Sequences, Paperfolding, Towers of Hanoi, and Progression-Free Sequences of Integers. *L'Enseignement Mathématique*, 38:315–327, 1992.

[4] J.-P. Allouche and J. Shallit. The Ubiquitous Prouhet–Thue–Morse Sequence. In *Sequences and Their Applications: Proceedings of SETA '98*, pages 1–16. Springer, 1999.

[5] J.-P. Allouche and J. Shallit. *Automatic Sequences: Theory, Applications, Generalizations*. Cambridge University Press, New York, 2003.

[6] J.-P. Allouche and G. Skordev. Von Koch and Thue–Morse Revisited, 2006.

[7] R. Alur and P. Černý. Streaming transducers for algorithmic verification of single-pass list-processing programs. *SIGPLAN Not.*, 46(1):599–610, January 2011.

[8] R. Alur and P. Černý. Streaming transducers for algorithmic verification of single-pass list-processing programs. In *Proceedings of the 38th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '11, pages 599–610, New York, NY, USA, 2011. ACM.

[9] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[10] H. P. Barendregt. The type free lambda calculus. In Jon Barwise, editor, *Handbook of mathematical Logic*, pages 1091–1132. Nort-Holland Publishing Company, Amsterdam, 1977.

[11] H. P. Barendregt and J. W. Klop. Applications of Infinitary Lambda Calculus. *Information of Computation*, 207(5):559–582, 2009.

[12] A. Belov. Some Algebraic Properties of Machine Poset of Infinite Words. *ITA*, 42(3):451–466, 2008.

[13] J. Berstel. Recent results on Sturmian words. In *Proc. Conf. on Developments in Language Theory (DLT 1996)*, pages 13–24. World Scientific, 1996.

[14] J. Berstel and J. Karhumäki. Combinatorics on Words - A Tutorial. *Bull. of the EATCS*, 79:178–228, 2003.

[15] V. Berthé. *Sequences Of Low Complexity: Automatic And Sturmian Sequences*. Course notes.

[16] A. Carpi. Repetitions in the Kolakovski Sequence. *Bull. of the EATCS*, 50:194–196, 1993.

[17] J. Cassaigne and J. Karhumäki. Toeplitz words, Generalized Periodicity and Periodically Iterated Morphisms. *European Journal of Combinatorics*, 18(5):497–510, 1997.

[18] S. Cautis, F. Mignosi, J. Shallit, M. Wang, and S. Yazdani. Periodicity, morphisms, and matrices. *Theoretical Computer Science*, 295:107–121, 2003.

[19] C. Choffrut and J. Karhumäki. Combinatorics of Words. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 1, pages 329–438. Springer, 1997.

[20] V. Chvátal. Notes on the Kolakoski Sequence. Technical Report 93-84, DIMACS, 1994.

[21] A. Cobham. On the Base-Dependence of Sets of Numbers Recognizable by Finite Automata. *Mathematical Systems Theory*, 3(2):186–192, 1969.

[22] A. Cobham. Uniform tag sequences. *Theory of Computing Systems*, 6:164–192, 1972.

[23] J. H. Conway. Unpredictable Iterations. In *Proceedings of the 1972 Number Theory Conference*, pages 49–52, 1972.

[24] J. H. Conway. Fractran: A Simple Universal Programming Language for Arithmetic. In T. M. Cover and B. Gopinath, editors, *Open Problems in Communication and Computation*, pages 4–26. Springer, 1987.

[25] K. Culik II and T. Harju. The omega-Sequence Problem for DOL Systems Is Decidable. *J. ACM*, 31(2):282–298, 1984.

[26] K. Culik II and J. Karhumäki. Iterative Devices Generating Infinite Words. In *Proc. 9th Ann. Symp. on Theoretical Aspects of Computer Science (STACS 1992)*, volume 577 of *Lecture Notes in Computer Science*, pages 531–543. Springer, 1992.

[27] K. Culik II, J. Karhumäki, and A. Lepistö. Alternating Iteration of Morphisms and Kolakovski [*sic*] Sequence. In G. Rozenberg and A. Salomaa, editors, *Lindermayer Systems, Impacts on Theoretical Computer Science, Computer Graphics and Developmental Biology*, pages 93–106. Springer, 1992.

[28] K. Culik II and J. Kari. Digital images and formal languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of formal languages, vol. 3*, pages 599–616. Springer, 1997.

[29] F. M. Dekking. Constructies voor 0-1-Rijen met Strikt Ergodische Afgesloten Baan. Master's thesis, Universiteit van Amsterdam, 1974.

[30] F. M. Dekking. On the Distribution of Digits in Arithmetic Sequences. In *Séminaire de Théorie des Nombres de Bordeaux*, volume 12, pages 3201–3212, 1983.

[31] F. M. Dekking. Iteration of Maps by an Automaton. *Discrete Mathematics*, 126(1-3):81–86, 1994.

[32] F. M. Dekking. What Is the Long Range Order in the Kolakoski Sequence? In R. V. Moody and J. Patera, editors, *The Mathematics of Long-Range Aperiodic Order (Waterloo, ON, 1995)*, volume 489 of *NATO ASI Series, Series C: Mathematical and Physical Sciences*, pages 115–125, Dordrecht, The Netherlands, 1997. Kluwer Academic Publishers.

[33] G. Edgar. *Measure, Topology, and Fractal Geometry*. Springer, 2nd edition, 2008.

[34] A. Ehrenfeucht, K. P. Lee, and G. Rozenberg. Subword Complexities of Various Classes of Deterministic Developmental Languages Without Interaction. *Theoretical Computer Science*, 1:59–75, 1975.

[35] A. Ehrenfeucht and G. Rozenberg. Subword Complexity of Square-Free D0L Languages. *Theoret. Comp. Sci.*, 16(1):25–32, 1981.

[36] S. Eilenberg. *Automata, Languages and Machines (Vol. A)*. Academic Press, 1974.

[37] J. Endrullis. Jambox: Automated Termination Proofs For String and Term Rewriting. The tool is available via http://joerg.endrullis.de/.

[38] J. Endrullis, C. Grabmayer, and D. Hendriks. Data-Oblivious Stream Productivity. In *Proc. Conf. Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2008)*, pages 79–96, 2008.

[39] J. Endrullis, C. Grabmayer, and D. Hendriks. Complexity of Fractran and Productivity. In *Proc. Conf. on Automated Deduction (CADE 2009)*, pages 371–387, 2009.

[40] J. Endrullis, C. Grabmayer, D. Hendriks, A. Isihara, and J. W. Klop. Productivity of Stream Definitions. In *Fundamentals of Computation Theory, 16th International Symposium, FCT 2007, Budapest, Hungary, August 27-30, 2007, Proceedings*, number 4639 in LNCS, pages 274–287. Springer, 2007.

[41] J. Endrullis, C. Grabmayer, D. Hendriks, A. Isihara, and J. W. Klop. Productivity of Stream Definitions. *Theoretical Computer Science*, 411:765–782, 2010.

[42] J. Endrullis, C. Grabmayer, D. Hendriks, and J. W. Klop. Infinite Streams. In *Nieuwsbrief van de NVTI (Newsletter of the Dutch Association for Theoretical Computer Science)*, pages 39–48. 2009. `http://www.nvti.nl/Newsletter/Nieuwsbrief2009.pdf`.

[43] J. Endrullis, C. Grabmayer, D. Hendriks, J. W. Klop, and R. C. de Vrijer. Proving Infinitary Normalization. In S. Berardi, F. Damiani, and U. de'Liguoro, editors, *TYPES 2008*, volume 5497 of *Lecture Notes in Computer Science*, pages 64–82. Springer, 2009.

[44] J. Endrullis, C. Grabmayer, D. Hendriks, and H. Zantema. The Degree of Squares is an Atom. In *Proc. Conf. on Combinatorics on Words (WORDS 2015)*, volume 9304 of *LNCS*, pages 1–13. Springer, 2015.

[45] J. Endrullis and D. Hendriks. From Outermost to Context-Sensitive Rewriting. In *Proc. Conf. on Rewriting Techniques and Applications (RTA 2009)*, volume 5595 of *LNCS*, pages 305–319. Springer, 2009.

[46] J. Endrullis and D. Hendriks. Transforming Outermost into Context-Sensitive Rewriting. *Logical Methods in Computer Science*, 6(2), 2010.

[47] J. Endrullis and D. Hendriks. Lazy Productivity via Termination. *Theoretical Computer Science*, 412(28):3203–3225, 2011.

[48] J. Endrullis and D. Hendriks. On Periodically Iterated Morphisms. In *Proc. Joint Meeting of Conference on Computer Science Logic and Symposium on Logic in Computer Science (CSL-LICS 2014)*, pages 39:1–39:10. ACM, 2014.

[49] J. Endrullis, D. Hendriks, and M. Bodin. Circular Coinduction in Coq Using Bisimulation-Up-To Techniques. In *Proc. Conf. on Interactive Theorem Proving (ITP)*, volume 7998 of *LNCS*, pages 354–369. Springer, 2013.

[50] J. Endrullis, D. Hendriks, and J. W. Klop. Degrees of Streams. *Journal of Integers*, 11B(A6):1–40, 2011. Proceedings of the Leiden Numeration Conference 2010.

[51] J. Endrullis, D. Hendriks, and J. W. Klop. Highlights in Infinitary Rewriting and Lambda Calculus. *Theoretical Computer Science*, 464:48–71, 2012.

[52] J. Endrullis, D. Hendriks, and J. W. Klop. Streams are Forever. *Bulletin of the EATCS*, 109:70–106, 2013.

[53] J. Endrullis, J. Karhumäki, J. W. Klop, and A. Saarela. Degrees of Infinite Words, Polynomials and Atoms. In *Proc. Conf. on Developments in Language Theory (DLT 2016)*, volume 9840 of *LNCS*, pages 164–176. Springer, 2016.

[54] J. Endrullis, J. Karhumäki, J. W. Klop, and A. Saarela. Degrees of infinite words, polynomials and atoms. *International Journal of Foundations of Computer Science*, 29(5), 2018.

[55] J. Endrullis, J. W. Klop, A. Saarela, and M. Whiteland. Degrees of Transducibility. In *Proc. Conf. on Combinatorics on Words (WORDS 2015)*, volume 9304 of *LNCS*, pages 109–121. Springer, 2015.

[56] S. Ferenczi. Tiling the Morse Sequence. *Theor. Comput. Sci.*, 94(2):215–221, 1992.

[57] S. Ferenczi. Complexity of Sequences and Dynamical Systems. *Discrete Mathematics*, 206(1-3):145–154, 1999.

[58] A. E. Frid. Sequences of Linear Arithmetical Complexity. *Theoretical Computer Science*, 339(1):68–87, 2005.

[59] J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In *Proc. Conf. on Automated Reasoning (IJCAR '06)*, volume 4130 of *LNCS*, pages 281–286. Springer, 2006.

[60] D. Goč, D. Henshall, and J. Shallit. Automatic theorem-proving in combinatorics on words. In Nelma Moreira and Rogério Reis, editors, *Implementation and Application of Automata*, pages 180–191, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[61] D. Goč, D. Henshall, and J. Shallit. Automatic theorem-proving in combinatorics on words. *International Journal of Foundations of Computer Science*, 24(6), 2013.

[62] C. Grabmayer, J. Endrullis, D. Hendriks, J. W. Klop, and L. S. Moss. Automatic Sequences and Zip-Specifications. In *Proc. Symp. on Logic in Computer Science (LICS 2012)*, pages 335–344. IEEE Computer Society, 2012.

[63] N. Hirokawa and A. Middeldorp. Tyrolean Termination Tool. In *Proc. Conf. on Rewriting Techniques and Applications (RTA '05)*, volume 3467 of *LNCS*, pages 175–184. Springer, 2005.

[64] R. A. Holmgren. *A First Course in Discrete Dynamical Systems*. Springer Universitext, 2nd edition, 1996.

[65] J. Hromkovič, J. Karhumäki, and A. Lepistö. Comparing Descriptional and Computational Complexity of Infinite Words. In J. Karhumäki, H. Maurer, and G. Rozenberg, editors, *Results and Trends in Theoretical Computer Science*, volume 812 of *LNCS*, pages 169–182. Springer, 1994.

[66] J. E. Hutchinson. Fractals and Self-Similarity. *Indiana Univ. Math. J.*, 30:713–747, 1981.

[67] K. Jacobs. *Invitation to mathematics*. Princeton University Press, 1992.

[68] K. Jacobs and M. Keane. 0-1-Sequences of Toeplitz Type. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 13(2):123–131, 1969.

[69] J. Karhumäki and A. Lepistö. Combinatorics on Infinite Words. Technical Report 578, Turku Centre for Computer Science, 2003.

[70] M. Keane. Generalized Morse Sequences. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 10(4):335–353, 1968.

[71] S. C. Kleene and E. L. Post. The upper semi-lattice of degrees of recursive unsolvability. *Annals of Mathematics*, 59(3):379–407, 1954.

[72] J. W. Klop. Term Rewriting Systems. In *Handbook of Logic in Computer Science*, volume II, pages 1–116. Oxford University Press, 1992.

[73] J. W. Klop and R. C. de Vrijer. Infinitary Normalization. In S. Artemov, H. Barringer, A. S. d'Avila Garcez, L. C. Lamb, and J. Woods, editors, *We Will Show Them: Essays in Honour of Dov Gabbay*, volume 2, pages 169–192. College Publications, 2005.

[74] H. von Koch. Une Méthode Géométrique Élementaire pour l'Étude de Certaines Questions de la Theorie des Courbes Planes. In *Acta Math.*, volume 30, pages 145–174, 1906.

[75] W. Kolakovski. Self Generating Runs. *Amer. Math. Monthly*, 72, 1965. Problem 5304.

[76] A. Koprowski. TPA: Termination Proved Automatically. In *Proc. Conf. on Rewriting Techniques and Applications (RTA '06)*, volume 4098 of *LNCS*, pages 257–266. Springer, 2006.

[77] D. Kozen and M. Soloviev. Coalgebraic tools for randomness-conserving protocols. In *Coalgebra, Now FLoC 2018*, 2018.

[78] C. Kupke and J. J. M. M. Rutten. Complete Sets of Cooperations. *Information and Computation*, 208(12):1398–1420, 2010.

[79] C. Kupke and J. J. M. M. Rutten. On the Final Coalgebra of Automatic Sequences. CWI Technical Report SEN-1112, CWI, 2011.

[80] C. Kupke and J. J. M. M. Rutten. On the Final Coalgebra of Automatic Sequences. In *Logic and Program Semantics - Essays Dedicated to Dexter Kozen on the Occasion of his 60th Birthday*, volume 7230 of *Lecture Notes in Computer Science*, pages 149–164. Springer, 2012.

[81] C. Kupke, J. J. M. M. Rutten, and M. Niqui. Stream Differential Equations: Concrete Formats for Coinductive Definitions. Technical Report RR-11-10, Oxford University, 2011.

[82] A. Lepistö. On the Power of Periodic Iteration of Morphisms. In *Proc. Colloquium on Automata, Languages and Programming (ICALP)*, volume 700, pages 496–506. Springer, 1993.

[83] M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 2nd edition, 2008.

[84] M. Lothaire. *Combinatorics on Words*. Cambridge University Press, 2nd edition, 1997.

[85] M. Lothaire. *Algebraic Combinatorics on Words*. Cambridge University Press, 2002.

[86] B. Löwe. Complexity Hierarchies Derived from Reduction Functions. In *Classical and New Paradigms of Computation and their Complexity Hierarchies*, volume 23 of *Trends in Logic*, pages 1–14. Springer, 2004.

[87] S. Lucas. mu-term: A Tool for Proving Termination of Context-Sensitive Rewriting. In *Proc. Conf. on Rewriting Techniques and Applications (RTA '04)*, volume 3091 of *LNCS*, pages 200–209. Springer, 2004.

[88] J. Ma and J. A. Holdener. When Thue–Morse meets Koch. In *Fractals: Complex Geometry, Patterns, and Scaling in Nature and Society*, volume 13, pages 191–206, 2005.

[89] A. Middeldorp, V. van Oostrom, F. van Raamsdonk, and R. C. de Vrijer, editors. *Processes, Terms and Cycles: Steps on the Road to Infinity, Essays Dedicated to Jan Willem Klop, on the Occasion of his 60th Birthday*, volume 3838 of *Lecture Notes in Computer Science*. Springer, 2005.

[90] F. Mignosi, A. Restivo, and S. Salemi. Periodicity and the Golden Ratio. *Theor. Comput. Sci.*, 204(1–2):153–167, 1998.

[91] R. Milner. The Spectra of Words. In [89], pages 1–5, 2005.

[92] M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

[93] M. Mohri, F. Pereira, and M. Riley. Weighted Automata in Text and Speech Processing. *CoRR*, abs/cs/0503077, 2005.

[94] M. Morse. Recurrent Geodesics on a Surface of Negative Curvature. In *Trans. Amer. Math. Soc.*, volume 22, pages 84–100, 1921.

[95] M. Morse and G. A. Hedlund. Symbolic Dynamics. *American Journal of Mathematics*, 60:815–866, 1938.

[96] M. Morse and G. A. Hedlund. Symbolic dynamics II: Sturmian trajectories. *Amer. J. Math.*, 62:1–42, 1940.

[97] A. A. Muchnik, Y. L. Pritykin, and A. L. Semenov. Sequences Close to Periodic. *Russian Mathematical Surveys*, 64(5):805–871, 2009.

[98] T. Nagell, editor. *Selected mathematical papers of Axel Thue*. Universitetsforlaget, Oslo, 1977.

[99] P. Odifreddi. *Classical recursion theory*. Studies in logic and the foundations of mathematics. North-Holland, Amsterdam, 1999.

[100] G. Paun. How much Thue is Kolakovski? *Bull. of the EATCS*, 49:183–185, 1993.

[101] H. Peitgen, H. Jürgens, and D. Saupe. *Chaos and Fractals*. New Frontiers of Science. Springer, 2nd edition, 2004.

[102] F. Pereira, M. Riley, and R. Sproat. Weighted rational transductions and their application to human language processing. In *Proceedings of the workshop on Human Language Technology*, HLT '94, pages 262–267. Association for Computational Linguistics, 1994.

[103] S. Peyton Jones. *Haskell 98 Language and Libraries, The Revised Report*. Cambridge University Press, 2003.

[104] J.-É. Pin. Profinite Methods in Automata Theory. In *Proc. of the 26th Symposium on Theoretical Aspects of Computer Science (STACS 2009)*, pages 31–50. IBFI Schloss Dagstuhl, 2009.

[105] J.-É. Pin and P. V. Silva. A topological approach to transductions. *Theoretical Computer Science*, 340(2):443–456, 2005.

[106] R. Plasmeijer and M. Eekelen. The Concurrent Clean Language Report (version 2.0). Technical report, University of Nijmegen, 2001.

[107] N. Pytheas Fogg. *Substitutions in Dynamics, Arithmetics and Combinatorics*, volume 1794 of *Lecture Notes in Mathematics*. Springer, 2002.

[108] G. Rayna. Degrees of finite-state transformability. *Information and Control*, 24(2):144–154, 1974.

[109] E. Roche. Text disambiguation by finite state automata, an algorithm and experiments on corpora. In *COLING*, pages 993–997, 1992.

[110] E. Roche. Compact factorization of finite-state transducers and finite-state automata. *Nord. J. Comput.*, 4(2):187–216, 1997.

[111] G. Roşu. Equality of Streams is a Π^0_2-complete Problem. In *ICFP*, pages 184–191, 2006.

[112] J. Sakarovitch. *Elements Of Automata Theory*. Cambridge, 2003.

[113] A. Salomaa. *Jewels of Formal Language Theory*. Computer Science Press, Rockville, Maryland, 1981.

[114] Helmut Schmid. A programming language for finite state transducers. In *Finite-State Methods and Natural Language Processing*, volume 4002 of *Lecture Notes in Computer Science*, pages 308–309. Springer Berlin / Heidelberg, 2006.

[115] M. Schroeder. *Fractals, Chaos, Power Laws, Minutes from an Infinite Paradise*. W. H. Freeman and Company, 1991.

[116] J. I. Seiferas and R. McNaughton. Regularity-preserving relations. *Theoretical Computer Science*, 2(2):147–154, 1976.

[117] J. Shallit. Open problems in automata theory: an idiosyncratic view. `https://cs.uwaterloo.ca/~shallit/Talks/bc4.pdf`, 2014. LMS Keynote Talk in Discrete Mathematics, BCTCS.

[118] J. Shallit and J. Stolfi. Two methods for generating fractals. *Computer & Graphics*, 13(2):185–191, 1989.

[119] J. R. Shoenfield. *Degrees of Unsolvability*. North-Holland, Elsevier, 1971.

[120] R. A. Shore. Conjectures and questions from Gerald Sacks's Degrees of Unsolvability. *Archive for Mathematical Logic*, 36(4-5):233–253, 1997.

[121] D. Siefkes. Undecidable extensions of monadic second order successor arithmetic. *Mathematical Logic Quarterly*, 17(1):385–394, 1971.

[122] N. J. A. Sloane. Online Encyclopedia of Integer Sequences. `http://oeis.org/`.

[123] N. J. A. Sloane. Online Encyclopedia of Integer Sequences, 2009. `http://www.research.att.com/~njas/sequences/`.

[124] R. I. Soare. Recursively enumerable sets and degrees. *Bulletin of the American Mathematical Society*, 84(6):1149–1181, 1978.

[125] C. Spector. On degrees of recursive unsolvability. *Annals of mathematics*, pages 581–592, 1956.

[126] D. Sprunger, W. Tune, J. Endrullis, and L. S. Moss. Eigenvalues and Transduction of Morphic Sequences. In *Proc. Conf. Developments in Language Theory (DLT 2014)*, volume 8633 of *Lecture Notes in Computer Science*, pages 239–251. Springer, 2014.

[127] R. Steacy. Structure in the Kolakoski Sequence. *Bull. of the EATCS*, 59:173–181, 1996.

[128] S. Stern. The Thue–Morse Sequence. Master's thesis, Vrije Universiteit Amsterdam, 2008.

[129] A. Telford and D. Turner. Ensuring Streams Flow. In *AMAST*, pages 509–523, 1997.

[130] A. Telford and D. Turner. Ensuring the Productivity of Infinite Structures. Technical Report 14-97, The Computing Laboratory, Univ. of Kent at Canterbury, 1997.

[131] Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003. Author's acronym includes J. W. Klop, F. van Raamsdonk and R. C. de Vrijer.

[132] Termination Competition. `http://www.termination-portal.org/`.

[133] W. Thomas. Automata on Infinite Objects. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 133–192. MIT Press, 1990.

[134] A. Thue. Über unendliche Zeichenreihen. In *Norske vid. Selsk. Skr. Mat. Nat. Kl.*, volume 7, pages 1–22, 1906. Reprinted in [98, 139–158].

[135] A. Thue. Über die gegenseitige Lage gleicher Teile gewisser Zeichenreihen. In *Norske vid. Selsk. Skr. Mat. Nat. Kl.*, volume 1, pages 1–67, 1912. Reprinted in [98, 413–478].

[136] D. Turner. An overview of miranda. *SIGPLAN Not.*, 21(12):158–166, 1986.

[137] M. Veanes, P. Hooimeijer, B. Livshits, D. Molnar, and N. Bjorner. Symbolic finite state transducers: algorithms and applications. In *Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '12, pages 137–150. ACM, 2012.

[138] M. Veanes, P. Hooimeijer, B. Livshits, D. Molnar, and N. Bjorner. Symbolic finite state transducers: algorithms and applications. *SIGPLAN Not.*, 47(1):137–150, January 2012.

[139] J. Waldmann. Matchbox: A Tool for Match-Bounded String Rewriting. In *Proc. Conf. on Rewriting Techniques and Applications (RTA '04)*, volume 3091 of *LNCS*, pages 85–94. Springer, 2004.

[140] H. Zantema. Termination of String Rewriting Proved Automatically. *J. Autom. Reason.*, 34(2):105–139, 2005.

[141] H. Zantema. Well-Definedness of Streams by Termination. In *Proc. Conf. on Rewriting Techniques and Applications (RTA '09)*, pages 164–178. Springer, 2009.

[142] H. Zantema and J. Endrullis. Proving Equality of Streams Automatically. In *RTA*, pages 393–408, 2011.

[143] H. Zantema and M. Raffelsieper. Stream Productivity by Outermost Termination. In *Proc. Workshop on Reduction Strategies in Rewriting and Programming (WRS 2009)*, volume 15 of *Electronic Proceedings in Theoreatical Computer Science*, pages 83–95, 2009.

[144] H. Zantema and M. Raffelsieper. Proving Productivity in Infinite Data Structures. In *Proc. 21st Int. Conf. on Rewriting Techniques and Applications (RTA 2010)*, volume 6, pages 401–416. Schloss Dagstuhl, 2010.