# Thirty Years of Collaboration with Jan van Leeuwen: In Search of Understanding Computation*

Jiří Wiedermann
Institute of Computer Science,
Academy of Sciences of the Czech Republic,
Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic
jiri.wiedermann@cs.cas.cz

*Motto:*

| | |
|---|---|
| *Caminante, son tus huellas* | *Wanderer, your footsteps are* |
| *el camino, y nada más;* | *the road, and nothing more;* |
| *caminante, no hay camino,* | *wanderer, there is no road,* |
| *se hace camino al andar.* | *the road is made by walking.* |
| *Al andar se hace camino,* | *By walking one makes the road,* |
| *y al volver la vista atrás* | *and upon glancing behind* |
| *se ve la senda que nunca* | *one sees the path* |
| *se ha de volver a pisar.* | *that never will be trod again.* |
| *Caminante, no hay camino,* | *Wanderer, there is no road —* |
| *sino estelas en la mar.* | *Only wakes upon the sea.* |

*Antonio Machado: Caminante. In: "Proverbios y cantares XXIX" [Proverbs and Songs 29], Campos de Castilla (1912); trans. Betty Jean Craige in Selected Poems of Antonio Machado (Louisiana State University Press, 1979)*

*This is the accompanying paper of the eponymous talk given at the occasion of Jan van Leeuwen's valedictory symposium "From the world of algorithms to algorithms for the world," held in honour of his retirement as Professor of Informatics at Utrecht University, on 20 December 2011. The work was partially supported by RVO 67985807 and GA ČR grant No. P202/10/1333.

*Bulletin of the EATCS no 108, pp. 148–160, October 2012*

**Abstract**

I have collaborated with Jan van Leeuwen since the beginning of the 1980s. The non-standard models of computations have been the central subject of our joint work. Among these models, we focused especially onto realistic models of supercomputers, interactive computing, hypercomputing, characterization of contemporary computing in terms of extended Turing machine paradigm, and viewing computations as unbounded processes. This note describes the main models designed and investigated during this period, summarizes the most important results and gives references to the respective papers.

# 1 Introduction

For the first time, I met Jan van Leeuwen at the MFCS (Mathematical Foundation of Computer Science) conference in 1981, in the High Tatras, Czechoslovakia (nowadays Slovakia). I remember him giving an invited talk (with Mark Overmars) on dynamization [7]. Dynamization is the process of transforming a static data structure into a dynamic one and in those days dynamization was a very popular subject. In his talk, Jan mentioned examples of so-called decomposable search problems that can be efficiently dynamized. It seemed to me that any common search problem was a decomposable one. After Jan's lecture, before lunch, I asked him for examples of non-decomposable search problems. After some thinking he came with such examples — e.g., computing the median of a set, or the diameter of a graph. After the lunch we both took part in conference excursion — a hiking trip to the mountains — and spent most of the time in conversation. It appeared that when compared to Jan's universal interest in almost every aspect of computer science, my main subject of interest has been much narrower: machine models of computation. Fortunately for me, Jan showed a vivid interest also in this subject. This is how, when and where our collaboration has started.

Rather than in a chronological order, it seems that a better way to describe our collaborative research effort and the respective results would be to group our joint works related to the same subject into clusters and to present each such cluster separately. This will give rise to five thematic areas that are all concerned with various complexity and later also computability aspects of modern computing. A word of caution: when speaking about (our) motivation and the general trends in computing at various times, the references to the related work of other authors is almost entirely missing, because of the nature of the underlying paper. However, in the respective papers of ours a more complete description of status-quo of the research in time of their writing is given.

## 2 A realistic model of supercomputing

Machine models of parallel computing have become a popular subject of investigations in theoretical computer science in the 1980s. Those years were a golden era of parallel computers. In real life, there existed supercomputers produced by traditional companies such as Cray, IBM and Hewlett-Packard. In theory, the literature abounded with models of parallel computers hidden under the acronyms and terms like PRAM, SIMDAG, MRAM, EDITRAM, recursive Turing machine, alternating Turing machine, k-PRAM, LPRAM, MIMD-RAM, etc. The nice computational property of the respective models has been that all “reasonable" models of parallel computers were polynomially (parallel) time related thanks to the fact that they all fulfilled the so-called *Parallel Computation Thesis: whatever can be solved in polynomially bounded space on a reasonable sequential machine model can be solved in polynomially bounded time on a reasonable parallel machine, and vice versa* [3]. Unfortunately, in practice the real instances of parallel computers behaved differently, being plagued by features that have been abstracted from in the “idealized" models of parallel computers (e.g., the cost of data transfers among the processors has been neglected). The practitioners even complained that computer scientists were not sufficiently prepared for the advent of parallel computers and are not able to offer reasonable remedies for the (then) critical situation with the supercomputing (cf. [26]). Facing this crisis, people started to investigate and design the so-called realistic models of parallel computers. Our model of Array Processing Machine (APM), which we designed in the mid-1980s [8] seems to be the first one in the still on-going series of realistic models of parallel computing. The APM was designed to closely model the architecture of existing vector- and array processors, and to provide a suitable unifying framework for the complexity theory of parallel combinatorial and numerical algorithms. Basically, the model can be seen as a random access machine with the ability to process arrays, i.e., blocks of memory of arbitrary length, in parallel (and hence, using a unit cost criterion, in a constant time).

In paper [8] we have shown that the APM can efficiently simulate a variety of extant models of parallel computation and vector processing. With a unit cost of its parallel operations, the model has satisfied the above mentioned Parallel Computation Thesis. However, with suitable “realistic" complexity measures taking into account the length of the manipulated sub-arrays (an analogue of the logarithmic cost measures of RAMs) this model adequately reflected the complexity of parallel computations observed in practice on the corresponding vector and array processors [9, 10]. The model has been used in the well-known textbook on structural complexity [1] by Balcázar, Díaz, and Gabarró as one of the main representatives of parallel machine models.

# 3 Interactive computing

Interactive computing denotes computations reacting to the inputs from the environment — be it from humans or other data sources (like sensors or other computers). In a sense, interactive computing can be seen as a computational "dialog" among the participating entities. In the typical case, an interactive computation processes potentially infinite streams of inputs and produces potentially infinite stream of outputs (reactions, behaviors). The feeling that interactive computing presents a computation which is qualitatively different from the classical one (embodied by the notion of a standard Turing machine computing a finite function) has been with computer science at least since the advent of personal computers whose modus operandi typically involved a lot of interaction with their environment. By the end of the 1990s, the most visible proponent of the idea that *"interaction is more powerful than algorithms"* was probably P. Wegner [24]. In his works (cf. [24] [25]) he called for a more computational view of interactive systems, claiming that they have a richer behavior than 'algorithms' as we know them. He even challenged the validity of Church's Thesis by proclaiming that Turing machines cannot adequately model the interactive computing behavior of typical reactive systems in practice. Wegner [25] (p. 318) writes:

*"The intuition that computing corresponds to formal computability by Turing machines ... breaks down when the notion of what is computation is broadened to include interaction. Though Church's thesis is valid in the narrow sense that Turing machines express the behavior of algorithms, the broader assertion that algorithms precisely capture what can be computed is invalid."*

Irrespective of whether this claim had been valid or not, we looked at the implications of interactiveness, from a computational view point, in [14] and [15]. The cornerstone in these studies has been the notion of a generic interactive machine interacting with an environment using single streams of input and output signals over a finite alphabet. The model used ingredients from the theory of $\omega$-automata. Viewing the interactive machines as transducers of infinite streams of signals, we showed that their interactive recognition and generation capabilities are identical. From the viewpoint of computability theory we showed that interactive computing does not lead to super-Turing computing power. One cannot say that *"interaction is more powerful than algorithms".* Our conclusion was that interactive computing merely extends our view of classically computable functions over finite domains to computable functions (translations) defined over infinite domains. Interactive computers simply compute something different from non-interactive ones because they follow a different computational scenario.

# 4 Extending Turing machine paradigm

The previously described investigations of interactive computing and the respective results were, at least for us, not yet quite satisfactory. We still felt that there is something more in the contemporary computing — a computational resource that is not captured by our models. What it could be?

The answer eventually came after a closed inspection of the most common interactive system in general use — the Internet. It has been the evolution. For a casual user of the Internet, its evolution presents an almost invisible activity, pursued behind the scene: new sites are added to the Internet, some sites vanish, the hardware and the software of the machines keep changing. The next important observation was that these evolutionary changes were, in fact, external, and hence, non-computable interventions. This insight lead straightforwardly to the definition of the so-called *interactive Turing machine with advice*, cf. [17], [30]. This is a Turing machine whose architecture is changed in two ways:

- instead of an input and output tape the interactive Turing machine has an *input port* and an *output port* allowing reading or writing potentially infinite streams of symbols;

- the machine is enhanced by a special, so-called *advice tape* that, upon a request, allows insertion of a possibly non-computable external information that takes a form of a finite string of symbols. This string must not depend on the concrete stream of symbols read by the machine until that time; it can only depend on the number of those symbols.

An advice is different from an oracle also considered in the computability theory: an oracle value can depend on the current input (cf. [6]), whereas an advice value must not. Such machines capture the following three properties of the modern computing systems (and especially of the Internet): (i) *interactiveness*, enabling communication with the environment, to reflect its changes, to get feedback, etc.; (ii) *evolution*, enabling development over generations, and (iii) potential *time-unboundedness*, allowing their open-ended development. Hence, the interactive Turing machines with advice represent a *non-uniform model of interactive, evolving, and time-unbounded computation*. (Non-uniformness is provided by the possibly uncomputable information inserted into running computations via the mechanism of the advice — this models the external interventions.)

A further support to this view of modern computing systems came from another model — evolving automata — that we have developed in [17], [23].

The *evolving automaton* with a schedule is an infinite sequence of finite automata sharing the following property: each automaton in the sequence contains a subset of states of the previous automaton in that sequence. This requirement

captures the persistence of data in the evolving automaton over time: some information available to the current automaton is also available to its successor. In this way, passing of information over generations is ensured.

For an on–line delivered potentially infinite sequence of the inputs symbols the schedule of an evolving automaton determines the *switching times* when the inputs to an automaton must be redirected to the next automaton. This feature models the (hardware) evolution. Note that at each time a computation of an evolving automaton is performed by exactly one of its elements (one automaton), which is a finite object.

An evolving automaton is an infinite object given by an explicit enumeration of all its elements. There may not exist an algorithm enumerating the individual automata. Similarly, the schedule may also be non-computable. Therefore, also evolving automata represent a non-uniform, interactive evolutionary computational model.

In [17] we have proved the computational equivalence of the evolving automata with the interactive Turing machine with advice. Based on these two models, we have formulated the following thesis [17]:

**Extended Turing Machine Paradigm** *A computational process is any process whose evolution over time can be captured by evolving automata or, equivalently, by interactive Turing machines with advice.*

Interactive Turing machine with advice is known to possess super-Turing computing power. Namely, in [17] we showed that such machines can solve the halting problem. In order to do so they need an advice that for each input of size $n$ allows to stop their computation once it runs beyond a certain maximum time. This time is defined as the maximum, over computations over all inputs of size $n$ and over all machines of size $n$ that halt on such inputs. Obviously (and unfortunately), such an advice is uncomputable. Thus, the super-Turing computing power of evolutionary interactive systems cannot be harnessed for practical purposes — it is only needed to precisely capture their computational potential, where the elements of uncomputability enter computing via unpredictable evolution of the underlying hardware and software.

We believe that the new paradigm represents a new understanding of computing. It innovates the classical view of computing in three ways: a shift from finite computations to potentially infinite interactive ones, a shift from rigid computing systems towards systems whose architecture and functionality evolve over time and, last but not least, an understanding that in general the latter process of evolution happens in an unpredictable, non-uniform, non-computable way.

# 5 Hypercomputing

The transition from the classical computing (embodied by classical Turing machines and captured by the Church-Turing Thesis) to the interactive one did not bring a qualitative leap in computing: not even evolutionary interactive computing could solve the undecidable problems. Yet, especially among the theorists, there was an increasingly more urgent wish to find a model of computing compatible with the sound scientific principles that could provably qualitatively outperform the classical Turing machine. Of course, by that time, by the the beginning of the twenty first century we (i.e., Jan and myself) knew that in a certain sense modern computing represented, e.g., by the Internet, has had this power which, unfortunately, could not be used for solving undecidable problems (because of the uncomputability of the necessary advice).

However, within relativistic theoretical black-hole physics, in the so-called Malament-Hogarth spacetime, the physicists have shown the existence of two different trajectories, $T_1$ and $T_2$, respectively, coming out from the same point, having the following property. The journey along $T_1$ takes an infinite time, whereas for a traveller along $T_2$ only finite time will elapse. Now, let $\mathcal{M}$ be a classical Turing machine performing a possibly non-halting computation. Then, we let $\mathcal{M}$ follow trajectory $T_1$ while a person waiting for the result of $\mathcal{M}$'s computation (called the observer) will follow trajectory $T_2$. In our Universe, rotating black holes are the candidates for a Malament-Hogarth spacetime. The above schema of computation could be implemented as follows. Trajectory $T_1$ will be the orbit around the black hole (outside the event horizon) while $T_2$ will be the trajectory of the observer falling towards the black hole. The physical property of the black holes and that of their environment allows the falling observer to receive signals from the orbiting computer. As soon as the observer reaches the black hole inner horizon, the so-called *Malement-Hogarth event* will occur, i.e., for $\mathcal{M}$ infinite time will elapse. If no signal is obtained by the observer until that time, the computation of $\mathcal{M}$ did not halt in finite time. The schema of computation is in Fig 1.

The existence of the Malament-Hogarth spacetime follows from the theory, but has not been confirmed empirically.

In 2002 we learned about the possibility of black hole computing from the paper by Etesi and Németi [2] and soon afterwards we explored the computational limits and features of their relativistic computers. Consequently, in [28] we defined the so-called *relativistic Turing machine* $\mathcal{R}$ as a specific deterministic multitape Turing machine with a separate input tape. Among its states tree is a non-empty subset or *relativistic* and *signal* states, respectively. There are two kinds of signal states: the positive, and the negative ones. Upon entering a relativistic state, a relativistic phase of the computation will start. A specific computation is launched answering the question *"continuing the computation from the*

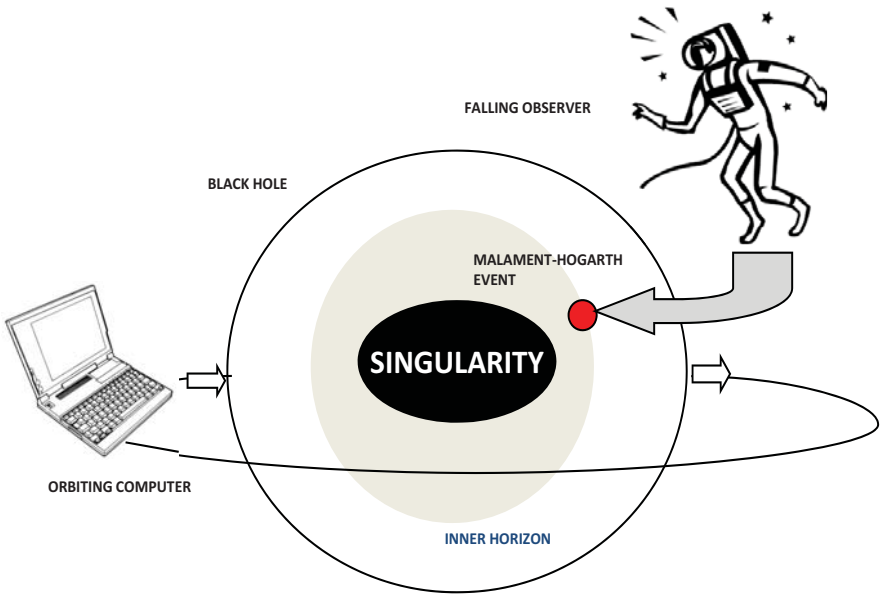


Figure 1: The schema of the black hole computing

*current configuration c, will $\mathcal{R}$ halt in a finite time?"* If so, then in the next step $\mathcal{R}$ enters a positive signal state; otherwise, it enters a negative state. Afterwards, in both cases the computation can proceed classically, starting from configuration $c$. A computation of $\mathcal{R}$ can contain but a finite number of alternations between classical and relativistic phases of computation.

Our paper [28] has complemented the study in [2] where only some consequences for computability theory are discussed. We showed that relativistic computing has precisely the power of recognizing the $\Delta_2-$sets of the Arithmetical Hierarchy [4]. If the underlying physical theory is accepted, this would lift the barrier of recursiveness "in nature" to the $\Delta_2-$level, i.e. without violating any feasible thought experiments in the general relativity theory. We also gave a complexity-theoretic characterization of relativistic computing in terms of Turing machines with advice mentioned in Section 4.

Our result has proved a kind of duality between infinite relativistic and non-uniform finite computations. This can be seen as a further evidence for the emerging central role of non-uniform computation models in capturing the information processing capabilities in natural systems, as formulated in [15]. The result complements the existing set of examples of such kinds of models, including evolutionary interactive systems, the Internet, artificial living systems, social systems, and amorphous computing systems (cf. [17],[30]), by systems operating by the principles of the general relativity theory. The result is also interesting in the context of (theoretical) physics or philosophy.

# 6 Computations as unbounded processes

In our forthcoming effort to further characterize the nature of contemporary computations, by the end of 2009 we realized that our quite general characterizations via the extended Turing paradigm abstracted from another important property of current computational systems. This is the fact that majority of these systems are multi-processor systems. As a rule, such a typical system runs uninterruptedly and consists of a (not necessarily a static) set of alternatively running communicating processes. Control goes from process to process and, whenever a process has its turn, the process computes until it executes an instruction that explicitly transfers control to another process.

Abstracting further, the work of the multi-process systems can be seen as in interplay between red and green set of processes, where the control occasionally passes from red to green processes, or vice versa, as dictated by the processes themselves. Using the Turing machine model as the basic underlying architecture, this approach has lead straightforwardly to the notion of *red-green Turing machines.* A red-green Turing machine is formally almost identical with the classical model of Turing machines. The only difference is that in red-green Turing machines the set of states is decomposed into two disjoint subsets: the set of green states, and the set of red states, respectively. There are no halting states. A computation of a red-green Turing machine proceeds as in the classical case, changing between green and red states in accordance with the transition function. The moment of state color changing is called mind change. A formal language is said to be recognized just in case when on the inputs from that language the machine computations “stabilize" in green states, i.e., from a certain time on, the machine keeps entering only green states. Similarly, a language is said to be accepted if and only if the inputs from that language are recognized, and the computations on the inputs outside that language eventually stabilize in red states.

In [21] we have investigated various aspects of red-green computations from the viewpoint of the computability theory. E.g., we showed that the computational power of red-green Turing machines increases with the number of mind changes allowed (their computational power increases along the so-called Ershov hierarchy, cf. [4]) and for any finite number of mind changes red-green Turing machines recognize languages in $\Sigma_2$ and accept languages from $\Delta_2$. In fact, computations of red-green Turing machines exactly characterize the latter two classes. This, together with the similar results achieved with the help of other machine or logical models of unbounded computation, suggests that, thanks to their simplicity and mathematical elegance, the red-green Turing machines can serve as a bridging model among the various alternative models of potential infinite computations. For instance, in [21] we proved that red-green Turing machines can elegantly and straightforwardly be simulated by relativistic Turing machines mentioned in Sec-

tion 5. This indicates relation to hypercomputing. A non-deterministic model of red-green Turing achiness can be defined as well. Interestingly, it appears that w.r.t. the number of mind changes, nondeterministic model is demonstrably more powerful than the deterministic one. The first results have been very promising and encouraging.

The work on this subject is in its beginning. The preliminary results were announced by Jan in February 2010 at the Workshop 'Philosophy of the Information and Computing Sciences' in Leiden [19]. The present state of the research and the results achieved up to now are in [21]. Finally, the red-green Turing machines have also been in the center of our contribution to the book dedicated to a centenary celebration of the life and work of Alan Turing. Namely, in 2011 we have "discovered" that the notion of computation as unbounded process as we distinguished it can be traced back to the discussion of so-called automatic machines (or: a-machines) in Turing's fundamental 1936 paper [5]. Whereas the vocabulary and notational style were different, we have argued that the non-terminating version of, what Turing called, *circular a-machines,* coincide quite precisely with our notion of red-green computation. Thus, our latest study [22] seems to fill a gap that has existed since then.

# 7 Conclusion

Usually, people do not distinguish cooperation from collaboration, and indeed, Webster dictionary defines these two words almost identically. But for the management specialists these two words represent different ways of joint work. When cooperating, people take coordinated decisions in order to achieve common goals (or benefits), whereas possible ways for achieving the goals are usually known in advance. Cooperation is based on a mutual trust and typically takes the form of a relatively short-term project, with little risk. When collaborating, people work together (co-labor) on solving the problems that are dynamically stated by the participants. The ways of solving the problems are not known beforehand. Collaboration is based on a mutual respect and typically is a long-term, on-going, open ended creative project with high risks as far as progress is concerned. The joint work of Jan and myself has definitively been the case of collaboration (albeit a minimal one in terms of participants and maximal one in terms of its duration).

Like the "caminante" from Machado's poem, at the beginning of our collaborative endeavor, little had we known how it would work, what problems we would address, and where it would lead us. Retrospectively, we see that our long joint journey has been a quest for understanding computation. Do we understand computation today? Well, certainly more than at the beginning of our collaboration. Our current understanding of computing is expressed by the extended Turing ma-

chine paradigm which represents an important shift in our thinking about computations.

Nevertheless, our recent findings around the red-green Turing machines indicate that there is still a long way ahead of us.

# References

[1] Balcázar, J. L., Díaz, J., Gabarró, J.: Structural Complexity II, Springer-Verlag, Heidelberg, 1990, 283 p.

[2] G. Etesi, I. Németi: Non-Turing computations via Malament-Hogarth spacetimes, *Int. J. Theor. Phys.* 41 (2002) 341-370, see also: http://arXiv.org/abs/gr-qc/0104023.

[3] van Emde-Boas, P.: Machine Models and Simulations. In: Handbook of Theoretical Computer Science, Vol A: Algorithms and Complexity, Jan van Leeuwen, editor, Elsevier, 1990, pp.3-61

[4] H. Rogers Jr, *Theory of recursive functions and effective computability*, McGraw-Hill, New York, 1967

[5] Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem, in: Proc. London Mathematical Society 42 1936-37, pp. 230-265.

[6] Turing, A. M.: Systems of logic based on ordinals, Proc. London Math. Soc. Series 2, vol. 45, 1939, pp. 161-228

[7] van Leeuwen, J., Overmars, M. H.: The Art of Dynamizing. Jozef Gruska, Michal Chytil (Eds.): Mathematical Foundations of Computer Science 1981, Štrbské Pleso, Czechoslovakia, August 31 - September 4, 1981, Proceedings. LNCS Vol. 118, Springer 1981, pp. 121-131

[8] van Leeuwen, J., Wiedermann, J.: Array Processing Machines. Technical Report RUU-CS-84-13, 1984

[9] van Leeuwen, J., Wiedermann, J.: Array Processing Machines. In: Proceedings FCT'85. Berlin, Springer Verlag 1985, pp. 257-268

[10] van Leeuwen, J., Wiedermann, J.: Array Processing Machines: An Abstract Model. BIT, Vol. 27, 1987, pp. 25-43

[11] van Leeuwen, J., Wiedermann J.: The Turing Machine Paradigm in Contemporary Computing. Utrecht, Universiteit Utrecht 2000, 19p. (Research Report UU-CS-2000-33)

[12] van Leeuwen, J. - Wiedermann, J.: On algorithms and interaction, in: M. Nielsen and B. Rovan (Eds), Mathematical Foundations of Computer Science 2000, 25th Int. Symposium (MFCS'2000), LNCS Vol. 1893, Springer-Verlag, Berlin, 2000, pp. 99-112.

[13] van Leeuwen, J., Wiedermann, J.: On the Power of Interactive Computing. In: Theoretical Computer Science. Exploring New Frontiers of Theoretical Informatics. IFIP TCS 2000, LNCS Vol. 1872, Berlin, Springer 2000, pp. 619-623

[14] van Leeuwen, J., Wiedermann, J.: A computational model of interaction, Technical Report, Dept of Computer Science, Utrecht University, Utrecht, 2000.

[15] van Leeuwen, J., Wiedermann, J.: Beyond the Turing Limit: Evolving Interactive Systems. In: SOFSEM'2001: Theory and Practice of Informatics. LNCS Vol. 2234, Berlin, Springer, 2001, pp. 90-109

[16] van Leeuwen, J., Wiedermann, J.: A Computational Model of Interaction in Embedded Systems. Utrecht, Universiteit Utrecht 2001, 28p. (Research Report UU-CS-2001-02)

[17] van Leeuwen, J. - Wiedermann, J.: The Turing machine paradigm in contemporary computing. In: B. Enquist and W. Schmidt (Eds), Mathematics Unlimited - 2001 and Beyond, Springer-Verlag, 2001, pp. 1139-1155.

[18] van Leeuwen, J., Wiedermann, J.: A Theory of Interactive Computation. A chapter in: Interactive Computation: The New Paradigm. Goldin, D.; Smolka, S. A.; Wegner, P. (Eds.) Springer Verlag, XV, 487 p., 84 illus., Hardcover, 2006

[19] van Leeuwen, J.: Computation as Unbounded Process. Slides from the presentation at the Workshop 'Philosophy of the Information and Computing Sciences', Lorentz Center, Leiden 2010, cf. http://www.cs.uu.nl/people/jan/LC-Philosophy-2010.ppt

[20] van Leeuwen, J., Wiedermann, J.: Name resolution by rewriting in dynamic networks of mobile entities. In: Rainbow of Computer Science, LNCS Vol. 6570, Springer, Heidelberg, 2011

[21] van Leeuwen, J., Wiedermann, J.: Computation as Unbounded Process. Theoretical Computer Science, accepted, 2011

[22] van Leeuwen, J., Wiedermann, J.: The computational power of TuringŠs nonterminating circular a-machines. In: S.B. Cooper, J. van Leeuwen (Eds), Alan Turing - His Work and Impact, Elsevier Publ., to appear.

[23] Verbaan, P.R.A., Leeuwen, J. van, Wiedermann, J. Complexity of Evolving Interactive Systems. In: Theory Is Forever, Essays Dedicated to Arto Salomaa on the Occasion of His 70th Birthday, Springer-Verlag, Berlin, p. 261-281, 2004.

[24] Wegner, P.: Why Interaction Is More Powerful Than Algorithms, Commun. ACM Vol. 40, No. 5, pp. 80-91, 1997

[25] Wegner, P.: Interactive foundations of computing, Theor. Comp. Sci. 192, pp. 315-351, 1998

[26] Wiedermann, J.: Quo vadetis, parallel machine models? In: Computer Science Today, J. van Leeuwen, editor. LNCS 1000, Springer, pp. 101-114, 1995

[27] Wiedermann, J., van Leeuwen J.: Emergence of a Super-Turing Computational Potential in Artificial living Systems. In: Advances in Artificial Life, ECAL 2001, LNCS 2159, Springer, pp. 55-65

[28] Wiedermann, J., van Leeuwen J.: Relativistic Computers and Non-Uniform Complexity Theory. In: Unconventional Models of Computation (UMC'2002), Berlin, Springer, LNCS 2509, 2001, pp. 287-299

[29] Wiedermann J., van Leeuwen, J.: The Emergent Computational Potential of Evolving Artificial Living Systems. Ai Communications, IOS Press, Vol. 15, No. 4, 2002, pp. 205-216

[30] Wiedermann, J., van Leeuwen, J.: How We Think of Computing Today. (Invited Talk) Proc. CiE 2008, LNCS 5028, Springer, Berlin, 2008, pp. 579-593