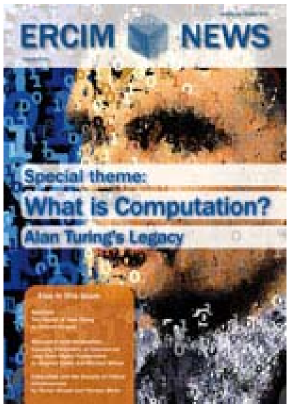


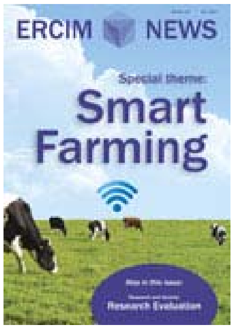
(/images/stories/EN91/EN91-web.pdf)

October 2012

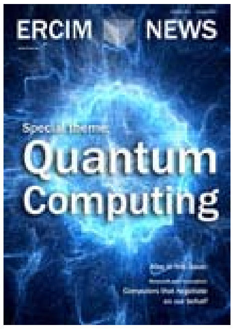
Special theme: What is Computation - Alain Turing’s Legacy

Guest editors: Gilles Dowek (Inria) and Samson Abramsky (University of Oxford)

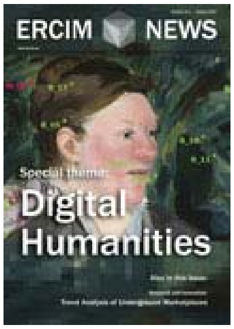
This issue in pdf (/images/stories/EN91/EN91-web.pdf)(40 pages; 6 Mb)



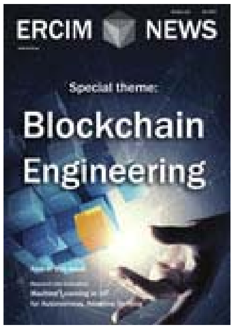
(/)ERCIM News 113



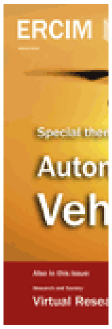
(/en112)ERCIM News 112



(/en111)ERCIM News 111



(/en110)ERCIM News 110



(/en109)

1 / 10

CONTENTS

In Brief (/en91/ib)

Keynote (/en91/keynote)

When Turing Meets Milner (/en91/special/when-turing-meets-milner)

Special Theme (/En91/special) 08 October 2012

Hits: 7982

by Jos Baeten, Bas Luttik and Paul van Tilburg

At CWI and Eindhoven University of Technology in the Netherlands, we enhanced the notion of a computation in the classical theory of computing with the notion of interaction from concurrency theory. In this way, we adapted a Turing machine as a model of computation to a Reactive Turing Machine that is an abstract model of a computer as it is used nowadays, always interacting with the user and the world.

What is a computation? This is a central question in the theory of computing, dating back to the work of Alan Turing in 1936. The classical answer is that a computation is given by a Turing machine, with the input given on its tape at the beginning, after which a deterministic sequence of steps takes place, leaving the output on the tape at the end. A computable function is a function of which the transformation of input to output can be computed by a Turing machine.

A Turing machine can serve thus as a basic model of a computation and, until the advent of the terminal in the 1970s, could also serve as a basic model of a computer. The terminal made direct interaction with the computer possible. Nowadays, a computer interacts continuously with the user at the click of a mouse or with many other computers all over the world through the Internet.

An execution of a computer is thus not just a series of steps of a computation, but also involves interaction. It cannot be modelled as a function, and is inherently nondeterministic. In our research, we made the notion of an execution precise, and compared this to the notion of a computation. To illustrate the difference between a computation and an execution, we can say that a Turing machine cannot fly an airplane, but a computer can. An automatic pilot cannot know all weather conditions en

Next issue: **July 2018**
Special theme:
Human-Robot Interaction
Call for the next issue (/call)

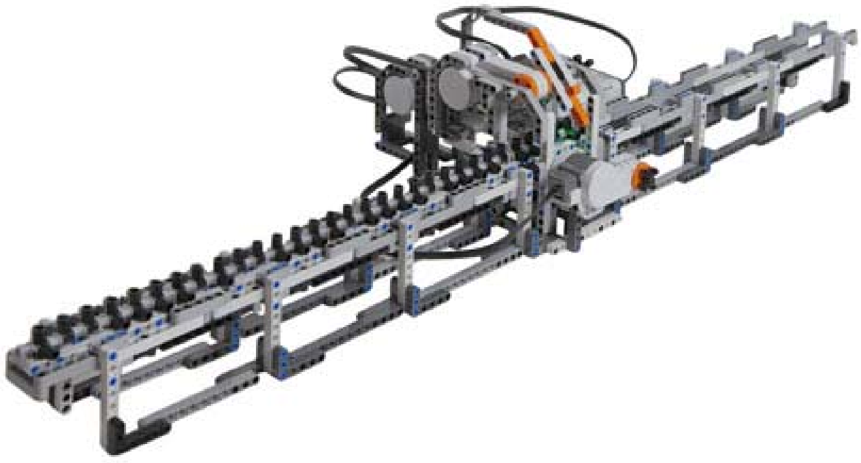
route beforehand, but can react to changing conditions run-time.

Computability theory, which is firmly grounded in automata theory and formal language theory, is the study of languages and the sets of strings, induced by them. Language can be viewed as an equivalence class of automata (under language equivalence).

The notion of interaction has been studied extensively in concurrency theory and process theory exemplified by the work of Robin Milner, who played a central role in the development of concurrency theory. He proposed a powerful parallel composition operator that is used to compose systems in parallel, including their interaction.

The semantics of concurrency theory are largely given in terms of transition systems which, in some respects, are similar to automata. There are, however, important differences. Firstly, a notion of final state, of termination, is often missing in concurrency theory. Secondly, transition systems need not be finite. The third and main difference between automata theory and concurrency theory is that language equivalence in concurrency theory is too coarse to capture a notion of interaction. Therefore, other notions of equivalence are studied in concurrency theory, capturing more of the branching structure of an automaton. Prominent among these is bisimulation equivalence.

We studied the notion of a computation, taking interaction into account. We defined, next to the notion of a computable function, the notion of an executable process, a behaviour that can be exhibited by a computer (interacting with its environment). An executable process is a divergence-preserving branching bisimulation equivalence class of transition system defined by a Reactive Turing Machine, an adaptation of the classical Turing Machine that can properly deal with ubiquitous interaction.



Lego Turing Machine built at CWI. Picture: CWI.

There have been other attempts to add a notion of interaction to computability theory but none have taken full advantage of the results of concurrency theory. Studies tend not to give interaction the status it deserves; in all the formalizations of interaction machines we could find, the notion of interaction itself is still very implicit.

Get the latest issue to your desktop
RSS 2.0 (/?format=feed&type=rss)

In our research, we used process theory to consider finite-state processes and pushdown processes. We also defined executable processes, highlighted the relationship of computable functions and executable processes, laying the foundations of executability theory alongside computability theory, and defined a new grammar for executable processes, based on the universality of the queue process.

In conclusion, we discussed the notion of an execution that enhances a computation by taking interaction into account. This was achieved by marrying computability theory, moving up from finite automata through pushdown automata to Turing machines, with concurrency theory, not using language equivalence but divergence-preserving branching bisimilarity on automata. Although every undergraduate curriculum in computer science contains a course on automata theory, an introduction to concurrency theory is rarely included. Both theories as basic models of computation are part of the foundations of computer science, and can be taught in an integrated manner, with the results of our research [2].

Finally, it is unlikely that Turing and Milner ever did meet: by the time Milner entered King's College in Cambridge as a young student, Turing had left some years earlier to take up a job in Manchester.

With the help of Annette Kik, CWI

References:

1. J.C.M. Baeten, T. Basten, and M.A. Reniers: "Process Algebra (Equational Theories of Communicating Processes)". Number 50 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2009.
2. J.C.M. Baeten: "Models of Computation: Automata, Formal Languages and Communicating Processes". Technische Universiteit Eindhoven, 2011. Syllabus 2IT15.
3. J.C.M. Baeten, B. Luttik, and P. van Tilburg: "Turing meets Mliner" in proc. of CONCUR 2012, Springer LNCS volume 7454, pp. 1-20. 2012.

Please contact:

Jos Baeten, CWI, The Netherlands
E-mail Jos.Baeten@cwi.nl (mailto:Jos.Baeten@cwi.nl)

{jcomments on}

Fraunhofer (https://www.ercim.eu/fhg) FNR (https://www.ercim.eu/fnr) FORTH (https://www.ercim.eu/forth)
INESC (https://www.ercim.eu/inesc) INRIA (https://www.ercim.eu/inria) ISI (https://www.ercim.eu/isi)
NTNU (https://www.ercim.eu/ntnu) SBA (https://www.ercim.eu/sba) SICS (https://www.ercim.eu/sics)
SZTAKI (https://www.ercim.eu/sztaki) UCY (https://www.ercim.eu/ucy) UWAW (https://www.ercim.eu/uwaw)
VTT (https://www.ercim.eu/vtt)