Informatics Everywhere: Information and Computation in Society, Science, and Technology

Tom Verhoeff
Dept. of Math. and CS, Eindhoven University of Technology
Den Dolech 2, 5612 AZ Eindhoven, The Netherlands
T.Verhoeff@tue.nl

Abstract

Informatics is about information and its processing, also known as computation. Nowadays, children grow up taking smartphones and the internet for granted. Information and computation rule society. Science uses computerized equipment to collect, analyze, and visualize massive amounts of data. Scientific theories more and more involve computational models. Technology incorporates computational mechanisms, even on a nano, molecular, or quantum scale, in every imaginable product. A story.

Keywords. Information, computation, society, science, technology

1 Introduction

In this article, I tell a story of why informatics is highly relevant and extremely exciting as a scientific discipline. Of course, informatics professionals already know all this (I would hope), but do IOI participants know this story? To develop and apply informatics further, we need to mobilize fresh talent and bring them up to speed. IOI participants are good candidates, but they often need some further coaxing and coaching beyond algorithms and programming.

As the title suggests, I will touch on five areas, but do so in a different order.

Society is pervaded with informatics: information and its (automated) processing —known as computation, in informatics lingo— are everywhere.

Information and computation: two key concepts united in informatics. No information without computation, no computation without information.

Science is about understanding 'the world' as it is; information and computation turn out to be indispensable tools for this; not only in analyzing data for scientific research, but also in defining models and theories. Biology, even chemistry, physics, social sciences, psychology, etc. revolve around information and computation.

Technology is about putting 'the world' to our use; most technological products, including medicines, involve ways to exploit nature for dealing with information and computation.

2 Society

Society, and I mean human society in particular, has always been centered around information and its processing. Compared to other species, human beings, as individual organisms, lack many physical characteristics —think of sheer strength, weaponry, armor, speed, camouflage— to survive among other organisms and the violence of nature. What has made us so successful is our capacity to communicate with each other and operate in organized groups. Information connects us and makes it possible to transcend our individual limitations.

In primitives times, information mostly concerned communication about the location of food and shelter, about danger and safety. It also concerned the tracking of human relationships. Group operations require organization, which typically followed blood lines, and a healthy population requires the avoidance of incest. Later, notions such as property, trading, and money got institutionalized. These require administration, which at its core concerns information. Also, farming and hunting benefit from storage and communication of information, no matter how primitive.

Information used to be handled via simple carriers, such as sounds, marks on rocks and trees, knots in ropes, signs on clay tablets, parchment, and papyrus. Information was known only in concrete physical forms. The development of writing systems and subsequently of book printing meant big leaps forward. That way, it became clear that the same language can live in diverse carriers. The information revolution was catapulted by our ability to harness information digitally and decouple it completely from physical carriers. That is, instead of just focusing on the physical information carriers, we are able to view information as something abstract and not necessarily physical. Almost any signal can be digitized and subsequently processed in the abstract digital domain. Such digital signals can then be translated either into physical actions or signals that we can understand. Informatics deals with information and computation in ways that abstract from the physical carriers.

Over time, we have also discovered and developed new ways of observing the world (sensors) and interacting with it (actuators). Information storage and processing used to be done by persons, and was therefore slow and error-prone. Through specialized equipment, the handling of information has now been automated to such an extent that, in numerous cases, it can be delegated to autonomous man-made systems. I will elaborate this point in Section 6.

Information is even more important to be successful nowadays than it was in the past. Informatics can make and break a society. For individuals, the world has become rather more complex, not so much physically, but informationally. You need the ability to tap very diverse sources of information: in order to select proper food, health care, and products, to deal with your finances and property, to plan your travels and navigate the roads, to interact with the government and companies, to connect to relatives and friends. Moreover, organizations and institutions themselves (government, military, education, law, commerce, industry, entertainment) have become more information centered. The internet, though primitive by some standards, has clearly made its mark. Cyberspace started as a of virtual world, but has become an integral part of our everyday reality. This poses new problems and dilemmas, whose understanding requires familiarity with informatics: digital ownership and copyright, privacy, identity, and (in the future) even existence. Gleick (2011) tells more of this story.

3 Information

So, what is information? Can you deal with information without dealing with some physical carrier of that information? In one sense, information is that which reduces uncertainty or, to put it differently, answers a question. The unit of information is the *bit*, which represents the answer to a question with two *equiprobable* answers. Typically, such an answer is denoted abstractly by the choice between a 0 and a 1. This notion of information does not involve specific physical carriers. Physically, a bit can be represented in many ways, for instance, the direction of magnetization of a small region in a material (such as a hard disk; soon to disappear), or the presence/absence of a package of electrons in a (semi)conductor (in your smartphone). More about this in Section 6.

Claude Shannon developed this *probabilistic information theory*. If you successively answer 100 yes/no questions where the answers are not equiprobable, you can convey the answers in less than 100 bit (data compression). If you have a communication channel that introduces some random errors (noise), then you can throw in a few well-chosen extra bits (redundancy) to protect against accidental loss of information (error correction). And when you want to prevent unauthorized persons from tapping your information, you can jumble up (encrypt) the bits to protect them against eavesdropping (cryptography). Every imaginable piece of information, for instance, numbers, lists of numbers, text, and pictures, can be encoded in appropriate sequences of bits. All of this is abstract, mathematical. In that sense, informatics is a science of the artificial, see Simon (1996), and not a natural science (but also see Section 5). Note that there is more to information than Shannon's probabilistic theory of information, see Adriaans and van Benthem (2008).

Data compression, error correction, cryptography, and encoding all involve (abstract) operations on information. In informatics we refer to such information processing as *computation*, even if the information does not involve numbers but involves text, graphics, etc. Bits become information —get meaning— through computations that operate on these bits and use them to make decisions. Such computations could take place in a brain or a computer. Computation is the topic of the next section. Interestingly, even if you try to confine yourself to the world of information, you will discover that computation can emerge in that world. Let me illustrate that.

Consider the famous *Game of Life* (GoL) defined by Conway (Berlekamp, Conway and Guy, 2004, Ch. 25). It is 'played' on an unbounded 2D grid of square cells, where each cell is in one of two states: dead or alive (or, if you prefer bits, 0 or 1; see Figure 1 left). Each cell is said to have eight neighbors: two horizontal, two vertical, and four diagonal. In the next generation, a dead cell will become alive when it currently has exactly three live neighbors, a live cell dies out when it has fewer than two or more than three live neighbors; otherwise, a cell does not change state.

Figure 1 shows three successive generations of a Game-of-Life configuration that starts with five live cells. Going from generation 2 to 3, one sees that

- three new live cells appear (yellow diamonds);
- one cell dies from starvation, having only one live neighbor (left red cross);
- one cell dies from crowding, having four live neighbors (right red cross).

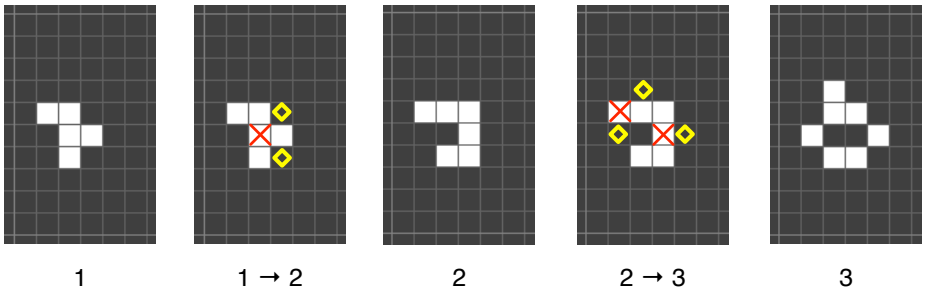

Figure 1: Three successive generations in the Game of Life: a white cell is alive, a black cell dead, a yellow diamond marks a newly born cell, and a red cross marks a dying cell

You might think of this grid of bits as a pure 'information world', that evolves according to a simple rule. You can experiment with the Game of Life (and similar worlds) using *Golly*, see Golly (2013). It turns out to be an interesting world. In fact, it is highly unpredictable, even though the rules are deterministic. That is, given an initial configuration, there is no general way of telling how it will evolve, e.g., whether it will die out completely, stabilize with some live cells, become periodic, or 'explode'.

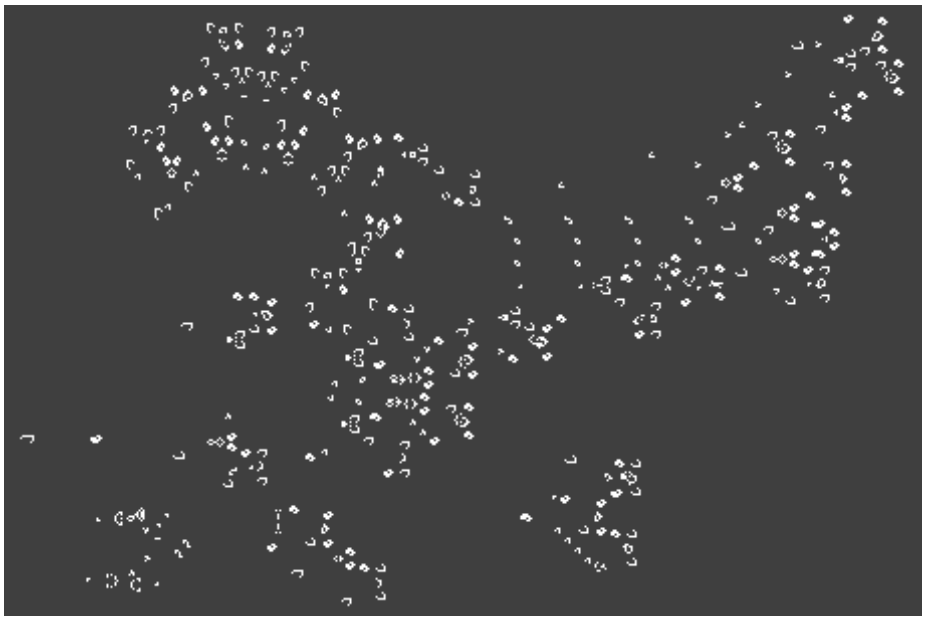

Figure 2: Game-of-Life configuration to report twin primes; it starts with 3062 live cells (in white)

How do we know this? Because we have found out that we can exploit this 'information world' to do arbitrary computations for us, see (Berlekamp, Conway and Guy, 2004, Ch. 25). Given any computation, you can construct a Game-of-Life configuration that will simulate that computation. Figure 2 shows

a configuration that 'generates' twin primes; that is, it 'searches' for twin primes, and whenever it 'finds' a pair it will 'report' that in a way that is recognizable from the configuration. The question whether there are infinitely many twin primes is then equivalent to whether there are infinitely many reportings during the evolution of that Game-of-Life configuration. Since we do not know whether there are infinitely many twin primes, the behavior of this configuration is not (yet) predictable.

Anything that can be computed, can be 'computed' by an appropriate Game-of-Life configuration (the Game of Life is *universal*). Thus, even in a pure 'information world' that was not designed for doing general information processing, the possibility of general computation can emerge, provided the 'information world' is 'sufficiently rich'. You can 'see' the computational aspects when you put on the right kind of interpretational 'glasses'.

4 Computation

The counterpart of a pure 'information world' is a pure 'computation world'. Can you have computation without information? The *lambda calculus* and, even more intriguing, *combinatory logic* can be viewed as pure computation worlds. In lambda calculus and combinatory logic, there are only functions, in the mathematical sense. These functions take one argument and return one result, and both the argument and the result are a function. A function of two parameters, such as $add(x, y) = x + y$ must be treated as a function of one parameter that returns a function of one parameter: $add(x)$ returns a function, say with parameter y, that adds x to its argument y. In the notation of lambda calculus:

$$add \quad = \quad (\lambda\, x\, .\, (\lambda\, y\, .\, x + y)) \tag{1}$$

A term of the form $(\lambda\, x\, .\, t)$ is called a λ-*abstraction*; in it, all free occurrences of x in t are said to be bound by the $\lambda\, x$.

Another term in the lambda calculus is *application of function f to argument x*, written as $(f\, x)$. Parentheses can be omitted under the assumption that function application is left-associative. That is, $f\, x\, y$ means $((f\, x)\, y)$, and thus $add\, x\, y = x + y$.

In lambda calculus, you express which functions are applied to what arguments. It abstracts completely from the nature of the 'data' passed in the arguments, like passing around envelopes without ever opening them. Only a few rules are needed to define the meaning of lambda terms. In particular, there is β-*reduction*, that defines the effect of applying a λ-abstraction $(\lambda\, x\, .\, t)$ to a term u:

$$(\lambda\, x\, .\, t)\, u \quad = \quad t[x := u] \tag{2}$$

where $t[x := u]$ denotes a *capture-avoiding substitution*: it is obtained from t by replacing every free occurrence of x in t by u, 'avoiding capture' of free variables in u by λ-abstractions occurring inside t (if needed, bound formal parameters inside t are systematically renamed through so-called α-conversion). The technical details are easy to find, so we will skip them here, see e.g. Moore and Mertens (2011), or Chu-Caroll (2013).

To illustrate this shuffling of unopened envelopes, consider the following function f defined by

$$f \quad = \quad (\lambda\, x\, .\, (\lambda\, y\, .\, (\lambda\, z\, .\, x\, z\, y))) \tag{3}$$

Thus, $f\, x\, y\, z = x\, z\, y$; that is, f applies its first argument to its third and then its second argument; it switches the roles of the two arguments of x.

The notation of lambda calculus involves formal parameters to help define how arguments are used by the function. However, such explicit parameters can be avoided, by starting with three special functions, called *combinators*, defined as follows.

$$\begin{aligned} \text{Identity function:} \quad I &= (\lambda\, x\, .\, x) && (4)\\ \text{Constant function:} \quad K &= (\lambda\, x\, .\, (\lambda\, y\, .\, x)) && (5)\\ \text{Substitution function:} \quad S &= (\lambda\, x\, .\, (\lambda\, y\, .\, (\lambda\, z\, .\, x\, z\, (y\, z)))) && (6) \end{aligned}$$

These functions are uniquely characterized by the following properties:

$$\begin{aligned} I\, x &= x && (7)\\ K\, x\, y &= x && (8)\\ S\, x\, y\, z &= x\, z\, (y\, z) && (9) \end{aligned}$$

The beautiful thing is that every lambda term can be expressed by an appropriate combination of SKI combinators. For instance, function f defined above in (3) can also be defined by

$$f \quad = \quad S(S(KS)(S(KK)S))(KK) \tag{10}$$

If you carefully carry out ten reductions, then you obtain (see Appendix A):

$$S(S(KS)(S(KK)S))(KK)\, x\, y\, z \quad = \quad x\, z\, y \tag{11}$$

The SKI combinators are basic forms of passing around empty envelopes; they suffice to express any other form. In fact, with a bit more juggling, you can even come up with a single combinator that can express all lambda terms. Unfortunately, I do not know of a Golly-like program for lambda calculus.

To define this pure 'computation world' takes a bit more effort than defining the pure 'information world' Game of Life. The surprising thing is that in this 'computation world' without data, it is possible to 'discover' data. In a way similar to how certain Game-of-Life configurations can be interpreted as doing information processing, you can also interpret certain pure functions as data values. For instance, the natural number n can be represented by the function

$$\widehat{n} \quad = \quad (\lambda\, s\, .\, (\lambda\, z\, .\, s^n\, z)) \tag{12}$$

where s^n consists of n applications of function s. For example, the number two is represented by

$$\widehat{2} \quad = \quad (\lambda\, s\, .\, (\lambda\, z\, .\, s\, (s\, z))) \tag{13}$$

To understand this representation, it may help to think of parameter s as a successor function that adds one, and parameter z as a zero function. The

number n is represented by a function that applies its first argument s precisely n times to its second argument z. In a way, this definition is very practical, since the function that represents the number n captures what it means to do something n times. These are known as *Church numerals*. Other functions can be defined to operate on these 'numbers', such as addition:

$$add\, x\, y \quad = \quad (\lambda\, s\, .\, (\lambda\, z\, .\, x\, s\, (y\, s\, z))) \tag{14}$$

In this definition of *add*, 'number' $y = \widehat{n}$ is used as the 'zero' of 'number' $x = \widehat{m}$, and since x applies s precisely m times to that 'zero', it yields the function that applies s precisely $m + n$ times to z, hence, representing the sum $m + n$: $add\, \widehat{m}\, \widehat{n} = \widehat{m + n}$. Here is $2+2$ in lambda calculus (see Appendix A for details):

$$\begin{array}{ll} & add\, \widehat{2}\, \widehat{2} \\ = & \quad \{ \text{ definitions of } add \text{ and } \widehat{2} \text{ } \} \\ & (\lambda\, s\, .\, (\lambda\, z\, .\, (\lambda\, s\, .\, (\lambda\, z\, .\, s\, (s\, z)))\, s\, ((\lambda\, s\, .\, (\lambda\, z\, .\, s\, (s\, z)))\, s\, z))) \\ = & \quad \{ \text{ four } \beta\text{-reductions } \} \\ & (\lambda\, s\, .\, (\lambda\, z\, .\, s\, (s\, (s\, (s\, z))))) \\ = & \quad \{ \text{ definition of } \widehat{4} \text{ } \} \\ & \widehat{4} \end{array}$$

In a similar way, we can define boolean values and boolean operators. With some more effort, we can define an *if*-construct, and also a *loop*-construct (fix-point combinator). That way, we have a full-blown (*universal*) programming language.

Any data that can be described, can be described by appropriate pure functions. Thus, even in a pure 'computation world' that was not designed for general data representation, the possibility of describing arbitrary data values can emerge, provided the 'computation world' is sufficiently rich. You can 'see' the data values (information) when you put on the right kind of interpretational 'glasses'.

5 Science

Science aims to understand the world around (and inside) us. Ultimately, such understanding could give us the ability to find out what will happen in any given situation, that is, to predict (some aspects of) the future. For instance, whether in the next year, a specific asteroid will hit the Earth or pass by at a safe distance. Or, whether a specific molecular compound will be toxic or work as a medicine. Typically, scientific theories have a mathematical basis, and predictions are obtained through computations. The development, fine tuning, and testing of theories involves huge amounts of data. At CERN, the Large Hadron Collider generates petabytes of data, see CERN (2013).

However, it is not this kind of informatics involvement in science that I consider fundamental. More interesting is that scientific theories and models themselves have become more and more computational in nature. Such computational models involve system *states* described by relevant information, and *state changes* described by computational steps on that state information. Constructing, analyzing, and applying such models requires informatics skills. Also,

the development of tools to assist in such modeling activities belongs to the domain of informatics.

In biology, this became obvious with the discovery of the genetic code and the mechanisms of self-reproduction. DNA is a carrier of genetic information, which is manipulated (transcribed, repaired, and copied) by protein structures in cells (see Verhoeff (2010) for an interactive challenge to write a self-reproducing program). These protein structures implement information processors, that is, computations. But also other sciences became more computational. Think of catalytic surface reactions in chemistry modeled by cellular automata like the Game of Life. In *digital physics* ('it from bit', as Wheeler phrased it; see Gleick (2011); Wikipedia (2013)), the universe is modeled as one big computation, with discrete information 'particles' as fundamental building blocks. This theme is also pursued in Wolfram (2002), see Rucker (2006) for a readable summary. Investigating nature is very much like investigating the (artificial) Game of Life or lambda calculus. Information and computation turn out to be inherent in nature: *nature is universal*, in a computational sense. Thus, informatics is not just a science of the artificial, but a natural science as well; also see Rosenbloom (2013).

More philosophically, when it comes to contemplating the predictability of the future, again, informatics is highly relevant. What does it mean that something is predictable? That we have a model that enables us to deduce information about a future state, given sufficient information about the current state. Note that it is not necessarily the case that we can deduce the future state in full detail. To be practically useful, the method for deducing information about that future state must be *effective*, that is, the method must be unambiguously executable in finite time. And that is precisely what, in informatics, we call an *algorithm*. A model is capable of predicting the future, when there is an algorithm that, given a current state as input, outputs information about a future state. Interestingly, from informatics, we know that for certain questions about the future of certain models, there exist no algorithms to decide those questions. Actually, any model that is sufficiently 'rich' (*universal*, in informatics terminology) is inherently unpredictable. Furthermore, some models are such that any algorithm to predict its future necessarily runs longer than just letting reality unfold (and bring about its own future).

This brings me to *algorithmic information theory* (AIT), as opposed to the probabilistic information theory of Section 3. In AIT, the information content of an object (such as a bit string), is defined as the length of a shortest program that generates the object. If a shortest program to generate a given bit string has 'nearly' the same size as the object itself, then that object is said to be *random*. A bit string consisting of one million zeros is clearly not random. There is no shortcut to storing or communicating a random bit string: you might as well store or communicate (a copy of) it. When observing the bits of a random bit string one by one, each next bit cannot be predicted efficiently. Discovery of patterns in the structure or behavior of nature is, in this sense, proof that nature is not completely random. However, there is no inherent reason why nature would not be random, at least in some respects. Indeed, 'most' bit strings turn out to be random, and indeed many processes in nature turn out to be 'chaotic', that is, their behavior is highly unpredictable.

6 Technology

Technology aims to provide artifacts that can help us live better in this world. Through technology, we twist nature to our intent. The artifacts vary over a wide range of products, including tools to assist in design and production. They interact with the world and with us. Some artifacts mostly have a physical purpose, like a hammer, a bow and arrow, a bridge, or a steam engine. Other artifacts, although physical in their construction, (also) concern information and computation, like a planetarium, a clock, a lock-with-key, or an abacus.

Technological artifacts somehow involve control, sensors, and actuators. Control and sensing used to be exclusively in human hands. But this turns out to be tedious, time consuming, costly, and error prone. Hence, there is a long history of automating the control. Initially, mechanical control mechanisms were invented and discovered, such as the centrifugal governor on a steam engine to stabilize its operating speed, the camshaft in a combustion engine to time the ignition, and the punched cards that controlled Jacquard's loom. Then, control became electro-mechanical, e.g., the rotating drum sequencers in old telephone exchanges and dish washers.

In hindsight, control centers around information and its processing. However, early controllers involved particular physical information carriers, and they were designed with a focus on the physical carriers, rather than on information as an abstract concept. This changed with the advent of electronics. Controllers became computers based on binary logic.

We have used (and still use) all kinds of natural phenomena as information carriers and processors in automated controllers (computers): electromagnetism (solenoid relays, core memory, hard disks), electrons (vacuum tubes, radio valves, transistors), photons (fiber optics), organic molecules (Adleman's DNA computer), quantum states of matter (quantum cryptography, experimental quantum gates and computers). The possibility for nature to compute is an emergent property. Like the Game of Life and lambda calculus, nature appears to be *universal*, when it comes to computing. For any imaginable computation, there is a way to 'hardwire' nature to do this computation. Moreover, this universality also implies that we can have *programmable* machines, where we can change the computation by changing a program rather than changing the 'wiring'. Informatics has contributed powerful programming techniques that enable us to solve complex computational problems, thereby paling the primitive controllers of the past.

Modern technology would be impossible without an abstract treatment of information and its processing. A key informatics concept is the *algorithm*. It needs no explanation in the IOI community, but for the general public it remains somewhat mystical. Fortunately, informatics is slowly becoming accepted as a serious topic in general education. I can recommend a book like Cormen (2013).

An overwhelming number of products nowadays contain computing devices and software. We have fully automated factories, autonomous robots, chemistry labs on a chip, and 3D-printers. The scale at which technology operates is shrinking: macro, micro, nano, molecular, quantum level. Who knows what is still to come.

7 Conclusion

The basic building blocks of informatics are surprisingly simple. To describe any kind of data, all you need is a bunch of bits, and to describe any kind of computation, all you need is an SKI combination. Information can be studied without emphasis on computation, and computation can be studied without emphasis on information. But, in the end, information and computation always go together.

Those basic building blocks are not convenient for practical application. Over the years, we have acquired an awesome arsenal of algorithms and data structures to solve many computational problems efficiently. This is well known to IOI participants. Some limits of computability have been mapped out clearly, see for instance the challenging and inspiring book by Hofstadter (1979), or Harel (2000); Moore and Mertens (2011); Cormen (2013).

The power of informatics is in abstraction, also see Verhoeff (2011). Informatics treats information and its processing without reference to concrete physical carriers (and certainly not to concrete computers). In that sense, it is like mathematics, where numbers are treated without reference to concrete physical objects. Guo (2010) defines informatics as "efficiently implementing automated abstractions". You might call informatics a branch of mathematics, but informatics goes beyond mathematics: *informatics automates mathematics*.

Informatics has earned the status of a separate scientific discipline, as eloquently argued by Rosenbloom (2013). It has changed the way we look at society, science, and technology. This concludes my story of informatics. However, the story of informatics is unfinished, with numerous exciting open problems; see, for instance, Adriaans (2013).

I hope this article

- inspires IOI coaches to broaden the view of IOI contestants on informatics,
- encourages discussion on informatics as a true scientific discipline, and
- persuades you to read some interesting literature on informatics topics.

Acknowledgments I would like to thank Ruurd Kuiper for critically reading a draft version of this article, and trying to keep me honest.

References

Adriaans, P. and J. van Benthem (2008). *Philosophy of Information*, Vol. 8 of the Handbook of the Philosophy of Science. Elsevier.

Adriaans, P. (2013). *Fundamental Problems in the Study of Information and Computation*. `http://www.pieter-adriaans.com/information/fundamental-problems-in-the-study-of-information-and-computation.html` (accessed Apr. 2013)

Berlekamp, E. R. and J. H. Conway and R. K. Guy (2004). *Winning Ways for Your Mathematical Plays*, Volume 4 (2nd Ed.). A. K. Peters.

CERN. *Computing*. `http://home.web.cern.ch/about/computing` (accessed Apr. 2013)

Chu-Caroll, M. C. (2013). *Good Math: A Geek's Guide to the Beauty of Numbers, Logic, and Computation*. The Pragmatic Bookshelf.

Cormen, Th. H. (2013). *Algorithms Unlocked*. The MIT Press.

Gleick, J. (2011). *The Information: A History, A Theory, A Flood*. Pantheon.

Golly (2013), An open source, cross-platform application for exploring Conway's Game of Life and other cellular automata. `http://golly.sourceforge.net` (accessed Mar. 2013)

Guo, P. J. (Feb. 2010). *What is Computer Science? Efficiently Implementing Automated Abstractions*. `http://www.pgbovine.net/what-is-computer-science.htm` (accessed Apr. 2013)

Harel, D. (2000). *Computers Ltd: What They Really Can't Do*. Oxford Univ. Press.

Hofstadter, D. R. (1979). *Gödel, Escher, Bach: An Eternal Golden Braid*. Basic Books.

Moore, C. and S. Mertens (2011). *The Nature of Computation*. Oxford University Press. Authors' book page: `http://www.nature-of-computation.org/` (accessed Mar. 2013)

Rosenbloom, P. (2013). *On Computing: The Fourth Great Scientific Domain*. The MIT Press.

Rucker, R. (2006). *The Lifebox, the Seashell, and the Soul: What Gnarly Computation Taught Me About Ultimate Reality, the Meaning of Life, and How to Be Happy*. Basic Books. Author's book page: `http://www.rudyrucker.com/lifebox/` (accessed Mar. 2013)

Simon, H. A. (1996). *The Sciences of the Artificial*. (3rd Ed.) The MIT Press.

Verhoeff, T. (2010). "An Enticing Environment for Programming". *Olympiads in Informatics*, **4**:134-141.

Verhoeff, T. (2011). "On Abstraction in Informatics", *ISSEP 2011: Proceedings of Selected Papers*, on CD-ROM. `http://pubshop.bmukk.gv.at/detail.aspx?id=444` Download: `http://www.win.tue.nl/~wstomv/publications/issep-2011-on-abstraction.pdf` (accessed Mar. 2013)

Wikipedia. *Digital Physics*. `http://en.wikipedia.org/wiki/Digital_physics` (accessed Apr. 2013)

Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media.

A Some Details

Here are the reductions to derive (11). Note that different reduction orders are possible.

$$
\begin{aligned}
& \underline{S(\underline{S(KS)(S(KK)S)})(\underline{KK})\,\underline{x}\,y\,z} \\
= \quad & \{ \text{ property (9) of } S \text{ } \} \\
& S(KS)(S(KK)S)\,x\,(\underline{KK\,x})\,y\,z \\
= \quad & \{ \text{ property (8) of } K \text{ } \} \\
& \underline{S(\underline{KS})(\underline{S(KK)S})\,\underline{x}}\,K\,y\,z \\
= \quad & \{ \text{ property (9) of } S \text{ } \} \\
& \underline{KS\,x}(S(KK)S\,x)K\,y\,z \\
= \quad & \{ \text{ property (8) of } K \text{ } \} \\
& S(\underline{S(\underline{KK})\underline{S}\,\underline{x}})K\,y\,z \\
= \quad & \{ \text{ property (9) of } S \text{ } \} \\
& S(\underline{KK\,x}(S\,x))K\,y\,z \\
= \quad & \{ \text{ property (8) of } K \text{ } \} \\
& \underline{S(\underline{K(S\,x)})\underline{K}\,\underline{y}}\,z \\
= \quad & \{ \text{ property (9) of } S \text{ } \} \\
& \underline{K(S\,x)\,y}\,(K\,y)\,z \\
= \quad & \{ \text{ property (8) of } K \text{ } \} \\
& \underline{S\,\underline{x}(\underline{K\,y})\,\underline{z}} \\
= \quad & \{ \text{ property (9) of } S \text{ } \} \\
& x\,z\,(\underline{K\,y\,z}) \\
= \quad & \{ \text{ property (8) of } K \text{ } \} \\
& x\,z\,y
\end{aligned}
$$

Here is $2 + 2$ in lambda calculus with all the details:

$$
\begin{aligned}
& add\,\widehat{2}\,\widehat{2} \\
= \quad & \{ \text{ definition of } add \text{ } \} \\
& (\lambda\,s\,.\,(\lambda\,z\,.\,\widehat{2}\,s\,(\widehat{2}\,s\,z))) \\
= \quad & \{ \text{ definition of } \widehat{2} \text{ } \} \\
& (\lambda\,s\,.\,(\lambda\,z\,.\,\underline{(\lambda\,s\,.\,(\lambda\,z\,.\,s\,(s\,z)))\,s}\,(\underline{(\lambda\,s\,.\,(\lambda\,z\,.\,s\,(s\,z)))\,s}\,z))) \\
= \quad & \{ \ \beta\text{-reduction on underlined applications } \} \\
& (\lambda\,s\,.\,(\lambda\,z\,.\,(\lambda\,z\,.\,s\,(s\,z))\,\underline{((\lambda\,z\,.\,s\,(s\,z))\,z)})) \\
= \quad & \{ \ \beta\text{-reduction on underlined application } \} \\
& (\lambda\,s\,.\,(\lambda\,z\,.\,\underline{(\lambda\,z\,.\,s\,(s\,z))\,(s\,(s\,z))})) \\
= \quad & \{ \ \beta\text{-reduction on underlined application } \} \\
& (\lambda\,s\,.\,(\lambda\,z\,.\,s\,(s\,(s\,(s\,z))))) \\
= \quad & \{ \text{ definition of } \widehat{4} \text{ } \} \\
& \widehat{4}
\end{aligned}
$$

Tom Verhoeff is Assistant Professor in Computer Science at Eindhoven University of Technology, where he works in the group Software Engineering & Technology. His research interests are support tools for verified software development and model driven engineering. He received the IOI Distinguished Service Award at IOI 2007 in Zagreb, Croatia, in particular for his role in setting up and maintaining a web archive of IOI-related material and facilities for communication in the IOI community, and in establishing, developing, chairing, and contributing to the IOI Scientific Committee from 1999 until 2007.