Errata and Addenda for "Informatics Everywhere" [7]

Tom Verhoeff

June 2013

Errata

p.150 Change 'Verthoeff, T. (2011)' into 'Verhoeff, T. (2011)'

Addenda

Cloud It would have been good to mention 'The Cloud' as a revolution that is separate from 'The Internet'. The internet is about *communication and connectivity*, viz., the ability to send information anywhere anytime. The cloud is about *storage*, viz., the ability to collect information on a large scale (through the internet, possibly involving diverse sensor networks), and store all of it for later processing (harvesting, mining).

Reduction Implicit in the text is the notion of *reduction*, viz., solving one problem by exploiting the solution of another problem. It is an important concept, that works in two directions:

1. To show that a problem is solvable, by reducing it to another solved problem.
2. To show that a problem is not solvable, by reducing another unsolved problem to that problem.

In particular, the latter kind of reductions can be surprising, e.g., when a problem domain (like solving diophantine equations) is shown to be computationally universal.

In a similar vein, it can be surprising how certain phenomena in nature, that were not 'intended' for computing, can in fact be exploited to compute.

Algorithmic Information Theory and Lazy Evaluation Lazy evaluation is related to AIT. E.g., in the programming language Python you have lists and generators. A list consists of all the data 'pregenerated', whereas a generator is a recipe for producing a sequence of items on demand. Such a generator can even 'produce' an infinite number of items. One can iterate over lists and generators in the same way. For lists with low information content, a generator is more storage-efficient than a list. Here is some Python code to illustrate this:

```
def fiblist(n):
    """ Returns the list of Fibonacci numbers < n."""
    result = [ ]
    a, b = 0, 1
    while a < n:
        result.append(a)
        a, b = b, a + b
    return result

def fibgen(n):
    """ Returns a generator for Fibonacci numbers < n."""
    a, b = 0, 1
    while a < n:
        yield a
        a, b = b, a + b

print sum(fiblist(1000))  # first stores all numbers in a list
print sum(fibgen(1000))   # does not store the numbers
```

References : missed or recently published

- Recently published, and highly relevant is [6]. This book uses Ruby to illustrate all technical points in an executable way. If you want to use JavaScript, then these libraries might be useful: [3, 4, 5, 8].
- Also recently published, [1] tells you all about computing and nature, in particular, quantum computing.
- To find out more about how informatics provides insight into the philosophy of science, read [2].

References

[1] Aaronson, Scott. *Quantum Computing since Democritus*. Cambridge University Press, 2013. `http://www.scottaaronson.com/democritus/` (accessed 14 Jun 2013)

[2] Abbott, Russ. "The Reductionist Blind Spot", *Complexity*, **14**(5):10-22, May 2009.

[3] David Ellis. *lambda.js*. `https://npmjs.org/package/lambda-js` (accessed 14 Jun 2013)

[4] Steele, Oliver. *Functional JavaScript*. `http://osteele.com/sources/javascript/functional/` (accessed 14 Jun 2013)

[5] Tao, Dan. *lazy.js*. `http://dtao.github.io/lazy.js/` (accessed 14 Jun 2013)

[6] Stuart, Tom. *Understanding Computation*. O'Reilly, 2013.

[7] Verhoeff, Tom. "Informatics Everywhere: Information and Computation in Society, Science, and Technology", *Olympiads in Informatics*, **7**:140–152, 2013.

[8] Zindros, Dionysis. *stream.js*. `http://streamjs.org` (accessed 14 Jun 2013)