Correctness-by-Construction in Stringology

Bruce W. Watson

FASTAR Research Group
Stellenbosch University
South Africa
bruce@fastar.org

Correctness-by-construction (CbC) is an algorithm derivation technique in which the algorithm is co-developed with its correctness proof. Starting with a specification (most often as a pre- and post-condition), 'derivation steps' are made towards a final algorithm. Critically, each step in the derivation is a *correctness-preserving* one, meaning that the composition of the derivation steps *is* the correctness proof.

In this talk, I will present several stringological derivations to illustrate the usefulness of CbC – with a particular focus on *exploratory algorithmics*[1] (see [10] for an example of a new CbC-derived algorithm) and weak points of other algorithm derivations.

Correctness proofs in stringology algorithms (and in related fields such as compression and arbology) are particularly important for a few reasons:

- Many stringology problems arise in so-called *infrastructure software*, such as network routers, security, operating systems, compilers, computational linguistics, etc. All of these areas, are performance- and correctness-critical, with a low tolerance for bugs – unlike many user-level or web-applications.
- For many stringology algorithms, the *devil is in the details:* correctly defining, using and precomputing various lookup tables often proceeds via case analysis – a technique which is not always convincing or water-tight.
- The broad usefulness of these algorithms makes them ideal and central to many computing science curricula – where convincing correctness proofs are important.
- The multitude of stringology algorithms (take, for example, exact keyword pattern matching) is difficult to oversee (though several works successfully present the breadth of the field [7,1]) and taxonomies play an important role in bringing order.

For showing an algorithm's correctness, CbC has significant advantages over other approaches, namely:

- *Testing:* Edsger Dijkstra famously said "Testing shows the presence, not the absence of bugs". It follows that testing is a poor replacement for proper correctness proofs.
- *A postiori proof:* The majority of new algorithms are *presented* first and followed by a correctness proof. This usually leaves a large gap between the algorithm (how was it arrived at?) and the proof, or the proof remains just a sketch, where the correctness of some parts of the algorithm are left for the reader to work out.

[1] Exploratory algorithmics is the invention of new algorithms by exploring gaps and previously unexplored options amongst the existing algorithms for a field.

Bruce W. Watson: Correctness-by-Construction in Stringology, pp. 1–2.
Proceedings of PSC 2012, Jan Holub and Jan Žďárek (Eds.), ISBN 978-80-01-05095-8

– *Automated proof:* Alongside the algorithm itself, a *model* of the algorithm is created; the model is then verified (using an automated theorem prover or a model-checker). This, of course, depends on a close correspondence between the algorithm and its model. Very few stringology derivations use this technique.

In CbC, at every derivation point there are often several possible derivation steps – which begs the question of how to choose the *right* step? Good derivations have an aspect of beauty and simplicity to them – properties which very often lead to the most efficient algorithms [3] – though writing down a good derivation requires practice and is an iterative process. Additionally, in many cases there are several possible 'next steps', making CbC an ideal technique for deriving entire *families* of algorithms. Such multiple-derivations can function both as a *taxonomy* (useful in teaching, for illustrating the commonalities and differences between related algorithms) and also for *exploratory algorithmics* in which new algorithms are invented [8,9].

CbC was 'invented' by Edsger Dijkstra in the late 1960's [2] with no small amount of input from his contemporaries such as Tony Hoare, Robert Floyd, Niklaus Wirth and Donald Knuth and also Dijkstra's colleagues in Eindhoven and Austin. Several Turing Awards (in particular Dijkstra's) were awarded for CbC-related research. David Gries and Carroll Morgan wrote two of the best follow-on text-books in the 1980's [4,6]. Despite the fact that some of those books are out of print, CbC remains alive and well as a successful and appropriate techniques for inventing and deriving new algorithms. The most recent book on this topic is by Kourie & Watson [5].

Acknowledgement: I would like to thank Nanette Watson-Saes and Derrick Kourie for proof-reading this extended abstract.

References

1. M. A. Crochemore and W. Rytter: *Jewels of Stringology*, World Scientific Publishing Company, 2003.
2. E. W. Dijkstra: *A Discipline of Programming*, Prentice Hall, 1976.
3. W. Feijen, A. van Gasteren, D. Gries, and J. Misra, eds., *Beauty is Our Business*, Springer-Verlag, 1990.
4. D. Gries: *The Science of Computer Programming*, Springer-Verlag, second ed., 1980.
5. D. G. Kourie and B. W. Watson: *The Correctness-by-Construction Approach to Programming*, Springer-Verlag, 2012.
6. C. Morgan: *Programming from Specifications*, Prentice Hall, second ed., 1998.
7. W. F. Smyth: *Computing Patterns in Strings*, Addison-Wesley, 2003.
8. B. W. Watson: *Taxonomies and Toolkits of Regular Language Algorithms*, PhD thesis, Faculty of Computing Science, Eindhoven University of Technology, the Netherlands, Sept. 1995.
9. B. W. Watson: *Algorithms for Constructing Minimal Acyclic Deterministic Finite Automata*, PhD thesis, Department of Computer Science, University of Pretoria, South Africa, 2011.
10. B. W. Watson, D. G. Kourie, and T. Strauss: *A sequential recursive implementation of dead-zone single keyword pattern matching*, in Proceedings of the International Workshop on Combinatorial Algorithms (IWOCA), 2012.