A Formalized Checker for Size-Optimal Sorting Networks

Luís Cruz-Filipe and Peter Schneider-Kamp*

Department of Mathematics and Computer Science, University of Southern Denmark
{lcf,petersk}@imada.sdu.dk

Sorting networks are hardware-oriented algorithms to sort a fixed number of inputs using a predetermined sequence of comparisons between them. They are built from a primitive operator – the *comparator* –, which reads the values on two channels, and interchanges them if necessary to guarantee that the smallest one is always on a predetermined channel. Comparisons between independent pairs of values can be performed in parallel, and the two main optimization problems one wants to address are: how many comparators do we need to sort n inputs (the *optimal size* problem); and how many computation steps do we need to sort n inputs (the *optimal depth* problem).

So far, most results on these two problems were obtained by special-purpose computer programs that performed exhaustive analysis of huge space states, using only a handful of mathematical results. In particular, in previous work [1] we proposed a generate-and-prune algorithm to show size optimality of sorting networks, and used it to confirm all known bounds for up to 8 inputs and to establish the previously unknown optimality of 25-comparator sorting networks on 9 inputs.

Our program includes a pruning step that takes two sequences of comparators and decides whether one of them can be ignored, by searching among all 9! permutations of 9 elements for one with a particular property. This expensive step, which often fails, has to be repeated billions of times throughout the whole run; therefore, the total execution required more than 10 years of computational time, or around one month on a state-of-the-art parallel cluster able to run 288 threads simultaneously. However, during execution we recorded the comparator sequences and permutations that allowed *successful* pruning steps, so that the whole run could later be independently validated in just a few hours of sequential computation by shortcutting the expensive search process. In this work we go one step further, and use this information to produce a formal proof of these results. Our proof is divided in three parts: (1) a formalization of the theory of size-optimal sorting networks, closely following the classical reference on the topic [4]; (2) a formalization of the generate-and-prune algorithm from [1], further optimized for performance, taking advantage of the information previously recorded; and (3) an extracted program, correct by design, that successfully replicated the previous executions of generate-and-prune.

The formalization of the theory. The formalization closely follows the presentation of sorting networks in [4], in particular when defining the concept of optimal size. It presents a number of problems regarding the handling of finite sets and combinatorial arguments involving permutations and cardinality, which we attempt to solve in a systematic way. As key results, we obtain operational characterizations of sorting networks.

However, it is not feasible to verify that a sequence of comparators on 9 inputs is a sorting network (the Coq process eventually crashes after a few hours of computation), which makes the execution of any generate-and-test algorithm in Coq unrealistic. This motivates the approach of using program extraction to confirm the optimality results we aim for.

*The authors were supported by the Danish Council for Independent Research, Natural Sciences. Computational resources were provided by the Danish Center for Scientific Computing.

The formalization of the checker. The original formalization of the checker closely followed the pseudo-code for generate-and-prune presented in [1], with one main difference: the expensive search step at the core of the pruning phase was replaced by information obtained from an oracle (a parameter in the formalization). This oracle is untrusted, so any data it provides is checked before it is used, and we formally prove soundness theorems stating that applying k steps of generate-and-prune on n channels will either: return a sorting network of the smallest possible size, if this is smaller than or equal to k; or return a negative answer, guaranteeing that no sorting network of size k or smaller exists.

However, the resulting extracted program had unfeasible execution times on 9 inputs, due to the huge size of the search space. Therefore, we optimized the underlying algorithm taking advantage of the fact that the oracle is *offline*: the data is freely available for preprocessing. That allowed us to reformulate the pruning step in linear, rather than quadratic, time (in the size of the search space). Simultaneously, we improved the data structures used by the checker in order to obtain performance gains in steps that are performed in constant time. After reproving the soundness properties, we were able to extract a program that ran in under a week on a modest processor.

The extracted program. The formalization uses an oracle as a parameter, which had to be implemented outside of Coq. This was achieved by means of a simple program that reads the (suitably preprocessed) files generated by the original proof and feeds them as arguments to the extracted checker. Since the checker does not trust the oracle, this information is verified, so we do not have to worry about showing that the implementation of the oracle is itself correct.

The major bottleneck in executing the checker was memory consumption, and we were forced to extract natural numbers into native types in order to bring the requirements down to a manageable level. However, as natural numbers are used only as identifiers for channels in comparator networks, this does not raise any issues regarding the correctness-by-design of the extracted program.

Results and future work. Running the extracted program, we formally confirmed all the values for size-optimal sorting networks up to 9 inputs, as described in [2, 3]. The formalization is available at `http://imada.sdu.dk/~petersk/sn/`. During this process, we identified a few minor mistakes in previously published proofs, the fixing of which required adjusting some definitions. We would like to extend this work with the theory of depth-optimal sorting networks, which is substantially different from the one currently formalized, as well as with a formal proof of a theoretical result regarding size-optimality [5].

References

[1] M. Codish, L. Cruz-Filipe, M. Frank, and P. Schneider-Kamp. Twenty-five comparators is optimal when sorting nine inputs (and twenty-nine for ten). In *ICTAI 2014*, pages 186–193. IEEE, 2014.

[2] L. Cruz-Filipe and P. Schneider-Kamp. Formalizing size-optimal sorting networks: Extracting a certified proof checker. *CoRR*, abs/1502.05209, 2015. Submitted for publication.

[3] L. Cruz-Filipe and P. Schneider-Kamp. Optimizing a certified proof checker for a large-scale computer-generated proof. In *CICM 2015*, LNCS. Springer, 2015. Accepted for publication.

[4] D.E. Knuth. *The Art of Computer Programming, Volume III*. Addison-Wesley, 1973.

[5] D.C. van Voorhis. Toward a lower bound for sorting networks. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 119–129. Plenum Press, New York, 1972.