Derrick G. Kourie
Department of Information Science
Stellenbosch University

Derrick G. Kourie, Department of Information Science, Stellenbosch University

Prof Ian Sanders
Subeditor
SACJ

January 25, 2017

"An Analysis of Algorithms for Deriving Failure Deterministic Finite Automata"

Dear Ian,

Thank you for the various referee reports in respect of the above submission to SACJ.

We have carefully considered the reviewer concerns and made a number of revisions accordingly.

We note that two of the reviewers commented that not all their original suggestions had been taken up. We draw your attention to the fact that the original paper of 9.5 pages now stands at 12.25 pages—thus almost 30% longer. The changes we have made are therefore quite significant. It is not possible to fully align changes with the views of each of 5 reviewers without significantly expanding the paper beyond its initial scope.

Nevertheless, we offer the responses below in respect of reviewer concerns. We trust that the current version of the paper will now be favourably considered.

Regards

Derrick G Kourie

Responses to Reviewers

1. Reviewer A: In the first round of reviewing, we contended with 5 different reviews. Reviewer A complains that "Most suggestions were ignored". Clear errors and typos were hopefully not ignored, but certain suggestions might have been. We tried to address the intersection of reviewer suggestions, as well as anything that was clearly erroneous. Non-standard views or suggestions that seemed of only marginal relevance were not taken up. For example, a suggestion that we use a Python-like pseudo-code instead of the classical GCL was ignored. Similarly, an annotation indicating the start state of a DFA was not removed, as it appears classically in several texts. We are unsure to what the reviewer refers in saying that "Some new erratic and inconsistent punctuation has been added to newer versions of algorithms." Again, since this has not been noted by any other reviewer, we ignore the comment.

2. Reviewer B: This reviewer expresses concern about the fact that two earlier publications have appeared covering aspects of this work presented here. To allay the reviewer's concerns, we firstly point out that earlier publications are mentioned upfront in a footnote in the text as follows:

   > Initial results have been published in [references supplied], the former being the proceedings of Formal Concept Analysis conference and the latter being an output of a private workshop between two research groups. Since these were both highly specialised forums, it seems appropriate to repeat the results here, together with subsequent results, so that all relevant findings are readily available to a wider community in one place.

   Moreover, we believe that there are several features about this text (some of which have been introduced or more clearly highlighted in the current revision) that the reviewer might not have fully noted.

   - The theme of FDFAs is not well developed in the literature. This is affirmed by the fact that, after several years of working on the topic, we discovered the Kumar et al. algorithm in a corner of the literature that is not routinely accessed by the Stringology community. They had not been using the term "Failure" DFAs, but had invented a new term "delayed input" DFAs. We think it would be useful for the research community if all our work was brought together in a single place.
   - The role of failure cycles in FDFAs is particularly underdeveloped. To our knowledge, the fact that FDFA optimality might entail a non-divergent cycle has not been noted in any previous publication. We were not aware of this until recently and it is, as far as we know, a new result. It is proven for the first time, by the example given in Figures 6 & 7.
   - Nowhere has it been previously explicitly noted that, because the Kumar et al. algorithm completely avoids cycles, the algorithm is guaranteed to miss optimal solutions that contain non-divergent cycles.

- We have slightly revised some of the previous text and moved it into subsections 5.3 and 5.5 respectively. This material illuminates the intuition behind the heuristics chosen for the DHA (subsection 5.3) in a way that was not given before. Subsection 5.5 also highlights in a more substantial way the core difference between DHA and the Kumar et al. algorithm: namely, that the latter is based on a pair-wise assessment of states, whereas the former collects together larger sets of states.
- The last two paragraphs of Section 7 give a correctness argument that the new algorithm for generating random FDFAs never produces a divergent cycle. This argument has not appeared before.
- Although we used Figure 2 to illustrate the idea of an FDFA in previous work, we make explicit here, for the first time, the fact that the failure cycle in Figure 2 is non-divergent.

The foregoing shows that the material in this submission is *not* simply a cut & paste from the two previous publications with new material tagged on at the end. Instead, most of the earlier material has been entirely re-written and integrated into a unified whole, reflecting the fact that ideas have solidified and matured over time.

3. Reviewer D: This reviewer prefaces his/her critique with the following statement:

> "...a paper is worth publishing when a reader can profit from reading it."

This reflects the reviewer's general concern that the DHA variants do not show a decisive performance (compaction) advantages over Kumer et al. with respect to the data sample studied. Moreover, it is also true that DHA is space-wise costly, because it relies on the construction of a concept lattice.

Nevertheless, in our view it would be premature to conclude from these experiments that DHA should be abandoned in favour of the Kumar et al. for the following reasons:

- As pointed out above, Kumar et al. cannot handle cycles, whereas DHA can.
- Because of its pair-wise approach, Kumar et al. tends to produce long failure chains while DHA tends to avoid them.
- The respective DHA performances of certain variants are only slightly worse than Kumar et al. and/or AC-opt.
- The set of concepts in the underlying lattice produced by DHA fully encapsulates the solution space in the following sense: Each concept represents the maximum sized set of states that have common exiting symbol transitions. Failure transitions can therefore connect all of these states (i.e. in the concept's extent), relying on a single one of the states to express the symbol transitions. Intuitively, an optimal solution will always be represented by a selection of some sequence of these concepts. It seems, therefore, that future

research to approximate an optimal solution (recall that an (exact) optimal solution has been proven to be NP-hard) should somehow take cognisance of this set of concepts, either explicitly or implicitly.

Reviewer D then repeats a sequence of assertions / questions that summarise an earlier review. We apologise for not fully responding to those questions in the earlier round[1]. We attempt to do so now.

- "The authors present three variants/specializations (MaxAR, MaxInt, MinExt) of a more general DHA algorithm, and compare them to two other, third party algorithms (AC-fail and D^2FA). In one test suite —the Aho-Corasick experiment (building an FDFA for simulated text search)— they compare it to AC-fail, and the results are that MaxInt and MinExt are very close to AC-fail in terms of reducing the number of transitions. They are better in that respect than D^2FA. What is their advantage over AC-fail? Do they have a better time complexity or execution time? They seem to require more memory. "

  This question misconstrues the nature of the AC-fail based experiments. AC-fail FDFAs *are* optimal (in terms of number of symbol transitions eliminated), but they only apply to a subset of DFAs. The experiment proves that DHA variants perform well against this standard, even though these variants are generic — i.e. in contrast to AC-fail, DHA variants can derive FDFAs from *any* complete DFA.

- "The other test suite is randomly generated automata with certain properties that result from the algorithm that generates them. In that setup, it is D^2FA that achieves the best reduction of the number of transitions. What is the advantage of the DHA variants? Are they faster? Are the generated FDFAs faster to use because of their shorter failure paths? "

  To some extent, these questions have already been answered, by indicating that the DHA variants are not limited to being cycle-free. Additionally, it is also true that shorter failure paths (obtained by DHA) would lead to faster processing times. However, the extent of such gains have not been verified in our experiments. Indeed, because the actual results are fairly close, the gains are likely to be marginal.

- "I don't think FCA should be discussed in this paper. It is sufficient to mention that DHA is inspired by FCA, but refer to states and outgoing transitions in the algorithm."

  We respectfully disagree with this view. Much of the discussion is reliant on the FCA-related definitions such as extent, intent, arc redundancy, etc. It

[1]It was somewhat daunting and confusing to know how to respond to 5 reviews, when their assessments and opinions did not necessarily converge.

is difficult to see how the reader would be able to make sense of the DHA variants without some FCA background, and that which we have provided is truly minimal.

- "It is not clear to me what the reduction rate would be in incomplete automata — is the sink state the most popular target of failure transitions?"

Our stated starting assumption is that we are dealing exclusively with complete DFAs. It would be trivial to convert any incomplete DFA to a complete DFA by introducing a sink state. In that case, whether or not the sink state would be the most popular target state of failure transitions, would be context dependent.

- "Do DHA variants offer any advantage over 'tail sharing' compression method by Kowaltowski et al?"

As far as we could see, the Kowaltoski et al article of 1993 deals with a compression strategy for representing binary automata. It indicates that a DFA state with multiple out-transitions can be represented as node list. Each node has a pointer on a relevant symbol to the appropriate destination state, as well as a pointer to the next node (a "failure transition") if the symbol is not encountered. The structure is called a binary automaton, and the article shows how such a binary automaton can be represented in memory in a more compact format. It is pointed out that the KMP algorithm implements such a binary automata. The thrust of the article therefore appears to be about a memory compression for binary automata, rather than about minimising the number of transitions in a DFA like structure.

However, we never cover the question of compression strategies. Instead, we have provided the reader with a reference to the 2014 book of J Daciuk (Optimization of Automata, published in 2014 by Gdansk University of Technology) whose chapter 8 gives a good summary of compression strategies.

Additionally, in our text giving an historical account FDFAs, we make that point FDFAs "are implicit in the data structure used in the Knuth-Morris-Pratt (KMP) algorithm". We then add the following footnote:

> "In this case, so-called binary automata are built from an originating DFA in which the successor states of a state are represented as a linked list. A binary automaton is essentially an FDFA that has a single symbol transition and a failure transition at every state. Its language (but not its states) corresponds to that of the originating DFA. Daciuk (2014) references sources that discuss compaction strategies for binary automata."