Incremental and Semi-Incremental Construction of Pseudo-Minimal Automata

Jan Daciuk[1], Denis Maurel[2], and Agata Savary[2]

[1] Gdańsk University of Technology, Poland
Invited professor at the University of Tours, October through November 2004
jandac@eti.pg.gda.pl
[2] Université François Rabelais, Tours, France
(denis.maurel+agata.savary)@univ-tours.fr

Pseudo-minimal automata ([1],[2]) are minimal acyclic automata that have a proper element (a transition or a state) for each word belonging to the language of the automaton. That proper element is not shared with any other word, and it can be used for implementing a function on words belonging to the language. For instance, dynamic perfect hashing (e.g. a mapping from n unique words to n consecutive numbers, such that addition of new elements does not change the order of the previous elements) can be implemented using a pseudo-minimal automaton ([3]).

The only existing algorithm for the construction of pseudo-minimal automata ([1]) requires the input data to be sorted in an unusual way (in reverse lexicographic order). We propose three other algorithms that can use lexicographically sorted data, unsorted data, or data sorted on decreasing length. All these algorithms result from slight modifications of known algorithms for the incremental and semi-incremental construction of minimal deterministic acyclic automata ([4], [5]). They are based on the following property: in a pseudo-minimal automaton, there is no path on which a divergent state (i.e. a state with more than one outgoing transition) follows a convergent state (i.e. a state with more than one incoming transition).

In a minimal deterministic acyclic automaton, there may be no two different equivalent states, i.e. states having the same right language (set of words spelled out on all paths from the given state to any final state). That is not the case for a pseudo-minimal automaton, in which each state q such that

$$|\overrightarrow{\mathcal{L}}(q)| > 1 \tag{1}$$

may have an equivalent state somewhere in the automaton (where $\overrightarrow{\mathcal{L}}(\mathrm{q})$ is the right language of q). The modifications introduced to the abovementioned algorithms rely on the verification of condition (1).

In the original algorithms for sorted (case 1) and unsorted data (case 2), cf. [4], as well as for data sorted on decreasing length (case 3), cf. [5], each time we add a new word to the automaton, we perform a local minimization of a path corresponding to one or more word suffixes (the suffix of the previously added word in case 1, or the suffix of the current word in case 2, and the suffixes of several previously added words in case 3). This suffix minimization is done by checking, for every suffix' state, if that state has an equivalent state already in the automaton. If an equivalence is discovered both states are merged.

In order for the automaton to be pseudo-minimal instead of minimal, the local minimization has to be modified so that merging of states happens only if states are equivalent and condition (1) is not met. As in cases 1 and 2, the local minimization happens recusively backwards (starting from a final state), and the cardinality of the right language is easy to track on the fly. However, in case 3, the information about condition (1) has to be stored explicitly. After the addition of a new word and before the local minimization, all states from the start state to the newly created divergent state (if any) are marked as not verifying condition (1).

We have performed a correctness proof of the three algorithms we have proposed. The complexity of each of them is linear with regard to the number of symbols in the input data, provided that the operations on the so-called *register* (data structure containing a single representative state of each equivalence class of the automaton under construction) can be carried out in constant time. In incremental algorithms, all through the construction process, the intermediate automaton remains pseudo-minimal, i.e. the computation space usage of the whole algorithm is optimal.

We have perfomed experiments on 3 word lists extracted from English, French and Polish natural language corpora. Lists contained only unique words. For the algorithm for unsorted data, words came in the order they appeared in corpora. For other algorithms, they were sorted appropriately. The comparison of construction speed was done with respect to the first stage of the algorithm by [1] (its first stage produces a pseudo-minimal automaton while the whole algorithm produces a minimal automaton).

The results show that the three new algorithms proposed here are slower (up to 2.4 times for English, up to 3.7 times for French and up to 3 times for Polish) than the original algorithm by [1]. However, that algorithm requires unusual sorting of input data, which can not always be done (e.g. for the sake of disk space, or because of the nature of dynamic perfect hashing).

References

1. Revuz, D.: Dictionnaires et lexiques: méthodes et algorithmes. PhD thesis, Institut Blaise Pascal, Paris, France (1991) LITP 91.44.
2. Maurel, D.: Pseudo-minimal transducer. Theoretical Computer Science (2000) 129–139
3. Daciuk, J., Maurel, D., Savary, A.: Dynamic perfect hashing with finite-state automata. In: Proceedings of Intelligent Information Systems. New Trends in Intelligent Information Processing and Web Mining. Advances in Soft Computing, Springer (2005)
4. Daciuk, J., Mihov, S., Watson, B., Watson, R.: Incremental construction of minimal acyclic finite state automata. Computational Linguistics **26** (2000) 3–16
5. Watson, B.: A fast new (semi-incremental) algorithm for the construction of minimal acyclic DFAs. In: Third Workshop on Implementing Automata, Rouen, France, Lecture Notes in Computer Science, Springer (1998) 91–98