ON THE PERMUTATIONAL POWER OF TOKEN PASSING NETWORKS.

M. H. ALBERT, N. RUŠKUC, AND S. LINTON

Abstract. A token passing network is a directed graph with one or more specified input vertices and one or more specified output vertices. A vertex of the graph may be occupied by at most one token, and tokens are passed through the graph. The reorderings of tokens that can arise as a result of this process are called the language of the token passing network. It was known that these languages correspond through a natural encoding to certain regular languages. We show that the collection of such languages is relatively restricted, in particular that only finitely many occur over each fixed alphabet.

1. Introduction

The study of graphs whose vertices can be occupied by tokens, or pebbles, which are moved along the edges has ranged from recreational mathematics [8, 10] to motion planning and related topics [3, 4, 6]. In most of these papers the problem is restricted to moving a fixed set of pebbles within a given graph, generally aiming to obtain a specific configuration. On the other hand, early works such as [5, 7, 9] dealt in a similar way with moving tokens, now thought of as items of data, within a network (represented as a directed graph) with the aim of producing specified outputs from a fixed input, or sorted output from a variable input.

In the latter area the problem of identifying permutations which could be produced when the network was restricted to a fixed size was considered in [2]. They showed that, under a natural encoding scheme, the collection of permutations generated by a token passing network is always a regular language.

2. Definitions and major results

A *token passing network*, $\mathcal{T}(G, I, O)$, consists of a directed graph G together with non-empty subsets I and O of the vertices of G. Of course I represents the set of input vertices, and O the set of output vertices.

In order to identify a permutation output from a token passing network, it suffices to know the *rank* of each output token, among all the remaining elements of the permutation. This rank cannot exceed the number of vertices in the underlying graph G of $\mathcal{T}$. The *rank encoding* of a permutation is obtained by replacing each symbol with its rank among the remaining symbols. By concentrating on the rank of each token, rather than its actual value, we can describe the operation of $\mathcal{T}$ quite simply.

A *state* of $\mathcal{T}$ consists of a sequence (possibly empty) of vertices of the network, not containing any duplicates. This represents the situation where the vertices in the sequence are occupied by numbered tokens, whose relative ordering agrees with the ordering of the sequence. That is, the smallest token occupies the first vertex of the sequence, the second smallest token the second vertex, and so on.

The *primitive transitions* in a token passing network in state s are of the following types:

Input: If $i \in I$ and i does not occur in s, then there is a transition $s \to si$.

Movement: If $s = avb$ and $v \to w$ is an edge of G and w does not occur in s then there is a transition $s \to awb$.

Output: If $s = aob$ with $o \in O$ then there is a transition $s \to ab$

The *output class*, $\mathrm{Out}(\mathcal{T})$ consists of all the permutations π that can be produced by some run of $\mathcal{T}$. It is easy to see that the output class of a token passing network is a pattern class.

The language $B(k)$ consists of the rank encodings of all permutations where the rank of any element is bounded by k. These permutations are referred to as *k-bounded*. Many token passing networks produce this language, the simplest being a buffer of k vertices, each both an input and output vertex. The language accepted by a token passing network, $\mathcal{T}(G, I, O)$, is a sublanguage of $B(|G|)$ since at most $|G|$ tokens can occupy the graph. We define the *boundedness* of $\mathcal{T}$ to be the minimum k such that $L(\mathcal{T}) \subseteq B(k)$.

We will also consider token passing networks restricted to operate with a total of at most c tokens in the network at any one time. In terms of the states introduced above, we restrict the operation of $\mathcal{T}$ to states represented by sequences of length at most c. We refer to such networks as *capacity restricted token passing networks* and denote the token passing network $\mathcal{T}$ restricted in this way by $\mathcal{T}_c$. Obviously $L(\mathcal{T}_c) \subseteq B(c) \cap L(\mathcal{T})$.

Our main theoretical results are:

Theorem 1. *Let c be a fixed positive integer. Then:*

$$\{L(\mathcal{T}) \,:\, \mathcal{T} \text{ a token passing network of boundedness } c\}$$

is finite.

Theorem 2. *Let c be a fixed positive integer. Then:*

$$\{L(\mathcal{T}_c) : \mathcal{T} \text{ a token passing network}\}$$

is finite.

These two results indicate that the permutational power of token passing networks is relatively restricted, in that there are infinitely many pattern classes represented by regular sublanguages of $B(C)$. Alternatively they can be viewed as compactness results for token passing networks. They say that, for a fixed boundedness or capacity bound, there is a finite test set, T, of permutations such that, if two token passing networks satisfying the boundedness conditions generate the same subset of T then they generate the same language. In Section 3 we will see that for boundedness (or capacity bound) 2, the set $T = \{21\}$ suffices, while for boundedness 3 we may take:

$$T = \{21, 321, 312, 31542, 324651\}.$$

3. Examples

The proofs of Theorems 1 and 2 show that a token passing network which produces a 3-bounded class cannot be very complex. There are in fact precisely five 3-bounded classes that can be produced by token passing networks (and there is no distinction between the inherently bounded, or capacity bounded framework in this case). The five classes are:

- **(A)** All 3-bounded permutations.
- **(B)** The 3-bounded permutations avoiding the pattern 321. These can be produced by two queues in parallel.
- **(C)** The 3-bounded permutations avoiding the pattern 312. These can be produced by a stack.
- **(D)** The 3-bounded permutations avoiding both the pattern 31542 and 32541.
- **(E)** A class whose set of minimal avoided patterns is infinite, given in the language of the rank encoding by:

$$322321, 3213(31)^*321.$$

Networks producing the latter four classes are shown in Figure 1.

A notable feature of the bottom right network in Figure 1 is that it produces the permutation 32541, but in doing so, the element 2 cannot be output before the element 4 is added to the network. This establishes that one type of greedy protocol for operating token passing networks is not effective.

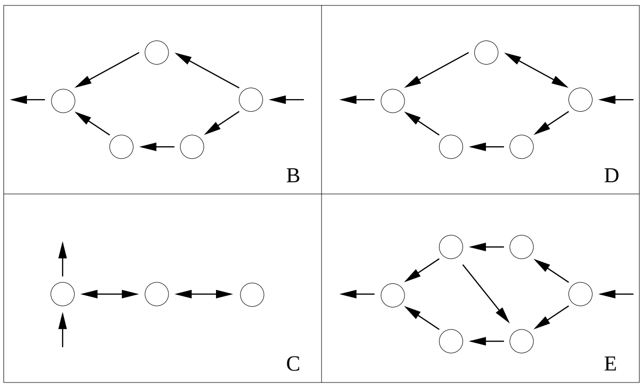

FIGURE 1. Examples of networks producing each possible non-universal 3-bounded class.

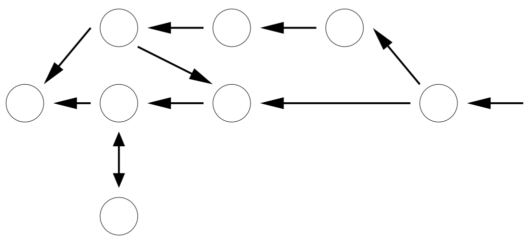

FIGURE 2. A network which produces the 4-bounded permutation with encoding 4222434111 but not without holding at least five tokens at some point.

Our final example is a token passing network $\mathcal{T}$ which has the property that:

$$L(\mathcal{T}_4) \neq L(\mathcal{T}) \bigcap B(4).$$

The network shown in Figure 2 can produce the permutation

$$4\,2\,3\,5\,8\,7\,10\,1\,6\,9$$

whose rank encoding is:

$$4\,2\,2\,2\,4\,3\,4\,1\,1\,1.$$

However, it cannot do so without at some point having five tokens in the network, namely token 10 must be added before token 7 is output. As this network has an inherent capacity of 5 this still leaves open the question of whether it is ever necessary to add more tokens to the network than its inherent capacity.

4. Summary and conclusions

The permutational power of token passing networks is relatively limited, at least in the variety of classes of permutations that they can produce. However, token passing networks are by no means trivial –

determining precisely the class produced by a token passing network, or its basis, is not a simple problem.

The effective procedures for doing so, found in [1], are generally of (at least) exponential complexity. However, even in the small cases to which they apply, the results include a number of novel antichains, of significant growth rates.

References

[1] M. H. Albert, M. D. Atkinson, and N. Ruškuc. Regular closed sets of permutations. *Theoret. Comput. Sci.*, 306(1-3):85–100, 2003.

[2] M. D. Atkinson, M. J. Livesey, and D. Tulley. Permutations generated by token passing in graphs. *Theoret. Comput. Sci.*, 178(1-2):103–118, 1997.

[3] V. Auletta, A. Monti, M. Parente, and P. Persiano. A linear-time algorithm for the feasibility of pebble motion on trees. *Algorithmica*, 23(3):223–245, 1999.

[4] Vincenzo Auletta and Pino Persiano. Optimal pebble motion on a tree. *Inform. and Comput.*, 165(1):42–68, 2001.

[5] Donald E. Knuth. *The art of computer programming.* Addison-Wesley Publishing Co., Reading, Mass.-London-Amsterdam, second edition, 1975. Volume 1: Fundamental algorithms, Addison-Wesley Series in Computer Science and Information Processing.

[6] C. H. Papadimitriou, P. Raghavan, M. Sudan, and H. Tamaki. Motion planning on a graph. In *Proc. of 35-th IEEE Symp. on Found. of Comp. Sc.*, pages 511–520, 1994.

[7] Vaughan R. Pratt. Computing permutations with double-ended queues. Parallel stacks and parallel queues. In *Fifth Annual ACM Symposium on Theory of Computing (Austin, Tex., 1973)*, pages 268–277. Assoc. Comput. Mach., New York, 1973.

[8] Daniel Ratner and Manfred Warmuth. The (n^2-1)-puzzle and related relocation problems. *J. Symbolic Comput.*, 10(2):111–137, 1990.

[9] Robert Tarjan. Sorting using networks of queues and stacks. *J. Assoc. Comput. Mach.*, 19:341–346, 1972.

[10] Richard M. Wilson. Graph puzzles, homotopy, and the alternating group. *J. Combinatorial Theory Ser. B*, 16:86–96, 1974.

Department of Computer Science, University of Otago, Dunedin, New Zealand

E-mail address: `malbert@cs.otago.ac.nz`

School of Mathematics and Statistics, University of St. Andrews, St. Andrews, United Kingdom

E-mail address: `nik@mcs.st-and.ac.uk`

School of Computer Science, University of St. Andrews, St. Andrews, United Kingdom

E-mail address: `sal@dcs.st-and.ac.uk`