

**Problem 1: Forward State Merging** (25 pts.)**Background**

Below is the transition matrix for a (somewhat random) 13-state DFA  $M$  over the alphabet  $\{a, b\}$ . Initial state is 1 and the final states are  $\{4, 6, 8, 9, 10, 11\}$ .

	1	2	3	4	5	6	7	8	9	10	11	12	13
$a$	2	4	6	8	12	9	12	13	12	13	12	12	13
$b$	3	5	7	12	10	13	11	12	12	13	12	13	12

This machine turns out to be non-minimal.

**Task**

- A. Use the forward state merging algorithm to compute the minimal DFA  $M_0$  for this machine.
- B. Describe the language  $L$  accepted by the machine.
- C. Construct the minimal DFA  $M_1$  for  $L$  directly by hand, using whatever method you prefer.
- D. Show that  $M_0$  and  $M_1$  are isomorphic.

**Comment** Make sure to build a nice table for the state merging process, don't just write down the final result. If you like, you can write a program to do this (or use the code on the web).

Part C. will be obvious once you have done part B.

**Problem 2: More Forward State Merging** (25 pts.)**Background**

We have seen that the standard forward state merging algorithm can be implemented in expected time  $O(n^2)$  (assuming the alphabet has fixed size). However, often the running time is better than quadratic since the number of refinement steps

$$\eta_{k+1} = \eta_k \sqcap \eta_k^{a_1} \sqcap \eta_k^{a_2} \sqcap \dots \sqcap \eta_k^{a_n}$$

is far less than  $n$ .

**Task**

- A. Give an example of a machine with  $n$  states where  $n - 2$  refinement steps are required before the approximation process stops.
- B. The standard implementation uses a hash table in the refinement step, so in principle the running time could be worse than quadratic (however unlikely such a failure may be). Give a modified version of the algorithm that is guaranteed quadratic time.

Hint: get rid of the hash tables and use a lookup table instead.

### Problem 3: Primitivity and Minimality (50 pts.)

#### Background

Let  $w$  be a non-empty word. Then  $z$  is the *root* of  $w$  if  $w \in z^*$  but there is no shorter word with this property. A word is *primitive* if it is its own root. For example,  $ab$  is the root of  $abababab$ . The primitive words of length 3 over  $\{a, b\}$  are  $aab, aba, abb, baa, bab, bba$ .

Given a binary word  $u = u_0u_1 \dots u_{n-1}$  define the corresponding subset of  $S(u) \subseteq \{0, 1, \dots, n-1\}$  by  $i \in S(u) \iff u_i = 1$  (in other words, think of  $u$  as a bitvector for membership). Define a DFA  $M(u)$  over the one-letter alphabet  $\{a\}$  as follows:

$$M(u) = \langle \{0, 1, \dots, n-1\}, \{a\}, \delta; 0, S(u) \rangle$$

where  $n = |u|$  and  $\delta(p, a) = p + 1 \pmod n$ .

#### Task

- A. Show that  $M(u)$  is minimal if, and only if,  $u$  is primitive by reasoning about the behavior of the states in  $M(u)$ .
- B. Give an alternative proof by “running” a minimization algorithm on  $M(u)$ .
- C. **Extra Credit (50):**  
Figure out how to characterize arbitrary minimal DFAs over a one-letter alphabet.

#### Comment

For the extra credit come up with an elegant description of the automata and make sure to prove that your description is correct. For even more extra credit figure out how to count minimal DFAs of size  $n$  over a one-letter alphabet. BTW, this is just about hopeless for larger alphabets.