Summary on Lecture 11, April 18th, 2016

Finite state machines: the minimization process

Humans are fairly good at constructing small Finite State Machines that are already minimal: one naturally tends to avoid "useless" states. Unfortunately, this little reassuring fact does not help much. In fact, humans fail spectacularly when the machines get large, even a few dozen states are tricky, thousands are not manageable.

In our course we will describe only first "baby steps" toward Automata Theory. If you are interested to undertand much more, you are encouraged to take the course CIS 420/520 Automata Theory.

Let $M = (S, I, O, \nu, \omega)$ be a finite state machine, where $I = \{x_1, \dots, x_t, \dots\}$ is an input alphabet, and $O = \{v_1, \dots, v_q, \dots\}$ is an input alphabet.

Definition 1. We say that two states $s, s' \in S$ are 1-equivalent if $\omega(s, x_t) = \omega(s', x_t)$ for each input letter $x_t \in I$. We use the notation $s \sim_1 s'$. Furthermore, if $k > 1$, we say that two states $s, s' \in S$ are k-equivalent if $\omega(s, x_{t_1} \cdots x_{t_k}) = \omega(s', x_{t_1} \cdots x_{t_k})$ for each input word $x_{t_1} \cdots x_{t_k} \in I^k$ of length k. We use the notation $s \sim_k s'$ for the k-equivalence.

It is easy to check that the k-equivalence of states is equivalence relation. With a little effort, one can prove the following statements:

Lemma 1. Let $k \geq 2$. Then if $s \sim_k s'$, then $s \sim_{k-1} s'$.

Lemma 2. Let $k \geq 2$. Then $s \sim_{k+1} s'$ iff $s \sim_k s'$ and $\nu(s, x) \sim_k \nu(s', x)$ for any input letter $x \in I$.

Exercise. Prove Lemmas 1, 2.

Example. Let $M = (S, I, O, \nu, \omega)$ be a machine described at Fig.1. It is easy to show that the $s_1 \sim_1 s_2 \sim_1 s_3$, and $s_4 \sim_1 s_5 \sim_1 s_6 \sim_1 s_7$. Check it!

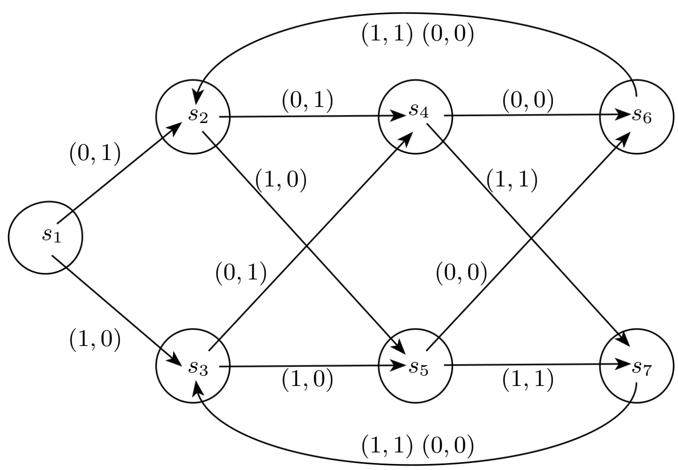

Fig. 1. The machine M

At the same time, we will see in few moments that $s_3 \sim_2 s_4$, and $s_5 \sim_2 s_6$, however, the states s_3 and s_5 are not 2-equivalent.

We will analyze this example further, and the goal is to find a finite state machine M' which will perform exactly the same task as M, but with fewer states. In other words, we would like to minimize the machine M.

Here is the idea: we partition the set of states S into the sets of 1-equivalent states, then each set of 1-equivalent states is partitioned to the set of 2-equivalent states and so on. In order to do that, we will use the state tables describing the functions ν and ω.

Below on the left is the state table of the machine M:

	ν		ω	
	0	1	0	1
s_1	s_2	s_3	1	0
s_2	s_4	s_5	1	0
s_3	s_4	s_5	1	0
s_4	s_6	s_7	0	1
s_5	s_6	s_7	0	1
s_6	s_2	s_2	0	1
s_7	s_3	s_3	0	1

	ν		ω	
	0	1	0	1
s_1	s_2	s_3	1	0
s_2	s_4	s_5	1	0
s_3	s_4	s_5	1	0
s_4	s_6	s_7	0	1
s_5	s_6	s_7	0	1
s_6	s_2	s_2	0	1
s_7	s_3	s_3	0	1

We examine the output values of the states, and we see that two groups of states $\{s_1, s_2, s_3\}$ (red) and $\{s_4, s_5, s_6, s_7\}$ (blue) are 1-equivalent. Thus we have the partition into the equivalence classes under 1-equivalence:

$$S = \{s_1, s_2, s_3\} \cup \{s_4, s_5, s_6, s_7\}$$

Now we notice that

$$\omega(s_1, 11) = 00, \quad \omega(s_2, 11) = 01, \quad \omega(s_3, 11) = 01,$$

i.e. s_1 is not 2-equivalent to s_2 and s_3. Then the values $\omega(s_2, xy) = \omega(s_3, xy)$ for all $x, y \in \{0, 1\}$. Thus $s_2 \sim_2 s_3$, and the set $\{s_1, s_2, s_3\}$ brakes into two subsets $\{s_1, s_2, s_3\} = \{s_1\} \cup \{s_2, s_3\}$ of the equivalence classes under 2-equivalence.

We examine 2-equivalence on the set $\{s_4, s_5, s_6, s_7\}$. Then we see that

$$\omega(s_4, 11) = 11, \quad \omega(s_5, 11) = 11, \quad \omega(s_6, 11) = 10, \quad \omega(s_7, 11) = 10,$$

i.e. the states s_4, s_5 are not 2-equivalent to s_6, s_7. We check further to find that $s_4 \sim_2 s_5$ and $s_6 \sim_2 s_7$:

	ν		ω	
	0	1	0	1
s_1	s_2	s_3	1	0
s_2	s_4	s_5	1	0
s_3	s_4	s_5	1	0
s_4	s_6	s_7	0	1
s_5	s_6	s_7	0	1
s_6	s_2	s_2	0	1
s_7	s_3	s_3	0	1

	ν		ω	
	0	1	0	1
s_1	s_2	s_3	1	0
s_2	s_4	s_5	1	0
s_3	s_4	s_5	1	0
s_4	s_6	s_7	0	1
s_5	s_6	s_7	0	1
s_6	s_2	s_2	0	1
s_7	s_3	s_3	0	1

We obtain the decomposition:

$$S = \{s_1\} \cup \{s_2, s_3\} \cup \{s_4, s_5\} \cup \{s_6, s_7\}.$$

Now it is easy to check that $s_2 \sim_k s_3$, $s_4 \sim_k s_5$, and $s_6 \sim_k s_7$. We form new machine with $s'_1 = \{s_1\}$ $s'_{2,3} = \{s_2, s_3\}$, $s'_{4,5} = \{s_4, s_5\}$, $s'_{6,7} = \{s_6, s_7\}$. Here is new machine M' with just four states:

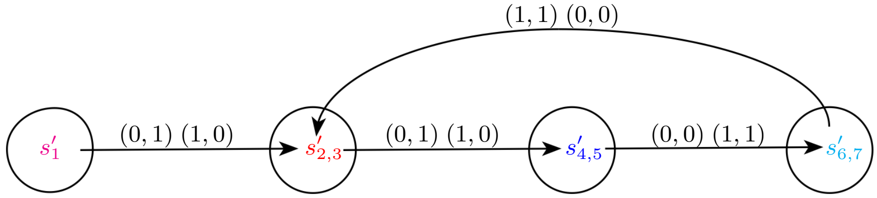

Fig. 2. The machine M'

The above example gives an idea how to minimize a finite state machine. Please read the textbook: section 7.5 provides details on the minimization process for one more example. There several examples will be given for you to analyze in the next homework.