

EECS 343 E-Handout: Adding Clocks to Automata

Prof. Branicky

In class, we introduced homomorphisms. Herein, we will use them to analyze a way that some people add a simple version of time to automata models.

Automata are *untimed* or *asynchronous* models of computation. Only events, not the timing of events, dictates transitions; only the ordering of events, not the relative or absolute times at which they occur, effect the outcome of a computation. (Think of typing at your keyboard.) However, when two or more systems are interacting concurrently, the timing of events may start to play a role. (Think of typing keys at the ATM or on your phone.)

A simple way to start addressing such problems is to add a *global clock*. Indeed, computer scientists and engineers have studied such things as “timed automata,” which associate real values of time with event occurrences. However, we will look at assuming that there is a fundamental period, or “clock tick,” underlying all computations. The basic idea is that events may occur or not occur for a particular machine in each clock period.

To add concreteness, consider a setup with N interacting finite state machines, M_1, M_2, \dots, M_N . Suppose that each performs computation in the form of accepting or rejecting (the necessarily regular) languages, B_1, B_2, \dots, B_N , each of which are subsets of their respective alphabets, $\Gamma_1, \Gamma_2, \dots, \Gamma_N$.

For example, suppose $N = 2$ and M_1 has $\Sigma_1 = \{a, b\}$ and $B_1 = (ab)^*(\epsilon + a)$, while M_2 has $\Sigma_2 = \{c\}$ and $B_2 = (cc)^*$. Then, they might each see the following strings, with the noted results:

$$\frac{M_1 : \text{ ababa } \text{ (accept)}}{M_2 : \text{ ccc } \text{ (reject)}}$$

Now, imagine that these machines interact in the sense that “c” to M_2 is an event that is generated as an (eventual) consequence of M_1 ’s seeing an “a”. We now have to introduce an ordering of events *across* machines, and we will do this by noting whether or not a symbol has occurred during a global clock tick. Continuing the above example, we might have

$$\frac{M_1 : \begin{array}{|c|c|c|c|c|c|c|c|} \hline a & b & & & a & & b & & a & \\ \hline \end{array}}{M_2 : \begin{array}{|c|c|c|c|c|c|c|c|} \hline c & & & & & & c & & & c \\ \hline \end{array}}$$

We can turn models like these into string-acceptance problems for new machines by introducing a new symbol, \perp (pronounced “bottom”), which indicates the event that “no input symbol has been seen” in that period:

$$\frac{M'_1 : \text{ a b } \perp \perp \perp \text{ a } \perp \text{ b } \perp \perp \text{ a } \perp}{M'_2 : \perp \text{ c } \perp \perp \perp \perp \perp \perp \text{ c } \perp \perp \perp \text{ c}}$$

Thus, we have introduced a process, consisting of interleaving arbitrary numbers of \perp ’s between ordinary input symbols, to model a simple form of concurrency. Denote this process on string w as $\text{CLOCK}(w)$ and on language B_i as $\text{CLOCK}(B_i)$. Above, we have shown particular members of $\text{CLOCK}(ababa)$ and $\text{CLOCK}(ccc)$. There are many more of each. In fact, $\text{CLOCK}(ccc) = L(\perp^*c\perp^*c\perp^*c\perp^*)$.

To be precise, $\text{CLOCK}(\epsilon) = L(\perp^*)$ and for $w \in \Gamma^*$,

$$\text{CLOCK}(w\gamma) = L(\text{CLOCK}(w)\gamma\perp^*), \quad \gamma \in \Gamma.$$

That is,

$$\text{CLOCK}(w) = L(\perp^*w_1\perp^*w_2\perp^*\cdots\perp^*w_n\perp^*), \quad w = w_1w_2\cdots w_n, \quad w_i \in \Gamma.$$

Extending to languages $B \subseteq \Gamma^*$,

$$\text{CLOCK}(B) = \bigcup_{w \in B} \text{CLOCK}(w).$$

The question is, are the resulting set of strings (i.e., the languages, $\text{CLOCK}(B_i)$) still regular? More pointedly, can we build M'_1 and M'_2 accepting them that are still finite state machines?

The answer is yes, and we may prove it by introducing homomorphisms, $h_i : \Gamma_i \cup \{\perp\} \rightarrow \Gamma_i$, defined recursively from

$$h_i(\sigma) = \begin{cases} \sigma, & \sigma \in \Gamma_i, \\ \epsilon, & \sigma = \perp. \end{cases}$$

The result follows from Theorem 10.2 since $\text{CLOCK}(B_i) = h_i^{-1}(B_i)$.