T-79.186 Reactive Systems

Spring 2003, Lecture 5

Keijo Heljanko

February 11, 2003

8 Automata on Infinite Words

To handle model checking of linear time temporal logic LTL it is natural to use automata. However, these automata will accept infinite words (strings/sequences) instead of finite words. We will later show how LTL formulas can be translated into automata on infinite words.

These automata are very closely related to finite state automata (on finite words). Note, however, that several of the used definitions for them are subtly different than for finite state automata.

The most widely used class of automata on infinite strings is called Büchi automata. We will introduce them next.

8.1 Büchi Automata

The definition of Büchi automata is identical to the definition of a finite state automata (modulo name of the automaton class).

Definition 8.1 A (nondeterministic) Büchi automaton $\mathcal{A}$ is a tuple $(\Sigma, S, S^0, \rho, F)$, where

- Σ is a finite **alphabet**,
- S is a finite set of **states**,
- $S^0 \subseteq S$ is set of **initial states**,
- $\rho : S \times \Sigma \to 2^S$ is the **transition function**, and
- $F \subseteq S$ is the set of **accepting states**.

The meaning of the transition function ρ is as follows: $t \in \rho(s, a)$ means that there is a move from state s to state t with symbol a.

The definition of the language accepted by the automaton differs from FSAs.

A Büchi automaton $\mathcal{A}$ accepts a set of words $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^\omega$ called the **language** accepted by $\mathcal{A}$, defined as follows:

A **run** r of $\mathcal{A}$ on an infinite word $a_0, a_1, \ldots \in \Sigma^\omega$ is an infinite sequence $s_0, s_1, \ldots$ of states in S, such that $s_0 \in S^0$, and $s_{i+1} \in \rho(s_i, a_i)$ for all $i \geq 0$.

Let $\mathit{inf}(r)$ denote the set of states appearing infinitely often in r. The run r is **accepting** iff $\mathit{inf}(r) \cap F \neq \emptyset$. A word $w \in \Sigma^\omega$ is accepted by $\mathcal{A}$ iff $\mathcal{A}$ has an accepting run on w.

The language of $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^\omega$ is the set of infinite words accepted by the Büchi automaton $\mathcal{A}$.

A language of automaton $\mathcal{A}$ is said to be **empty** when $\mathcal{L}(\mathcal{A}) = \emptyset$.

It is easy to check whether $\mathcal{L}(\mathcal{A}) \neq \emptyset$ by using the following observation: The language of the Büchi automaton is non-empty iff from some initial state $s \in S_0$ a final state s' can be reached, such that s' can reach itself by a non-empty sequence of transitions.

The check above can be easily made by a linear time algorithm. (We'll come back to that later.)

We define a strongly connected component C of a graph to be a set of nodes $C \subseteq S$, in which for all pairs of distinct states $s, s' \in C$ it holds that: s' can be reached from s and s can be reached from s'.

A strongly connected component C is called maximal, if no strongly connected component $C' \subseteq S$ exists, such that $C \subset C'$.

A maximal strongly connected component is called non-trivial iff: (i) $|C| > 1$, or (ii) there exists $s \in C$ such that there is an edge in the graph from s back to s.

Another way of checking the non-emptiness of $\mathcal{L}(\mathcal{A})$ is to compute the maximal strongly connected components (MSCCs) of the Büchi automaton, and check whether some non-trivial maximal strongly connected component C reachable from some initial state $s \in S_0$ contains an accepting state ($C \cap F \neq \emptyset$).

Also this emptiness checking approach can be implemented with a linear time algorithm. (Compute the reachable MSCCs and check whether any non-trivial MSCC contains an accepting state.)

We will now start defining the Boolean operators on Büchi automata:
$\mathcal{A}_1 \cup \mathcal{A}_2$ and $\mathcal{A}_1 \cap \mathcal{A}_2$.

We will not be able to show $\overline{\mathcal{A}}$ in this course due to its technical complexity, but it also exists. Thus also Büchi automata are closed under Boolean operations.

The $\cup$ definition for Büchi automata is identical with the FSA definition.

Definition 8.2 Let $\mathcal{A}_1 = (\Sigma, S_1, S_1^0, \rho_1, F_1)$ and $\mathcal{A}_2 = (\Sigma, S_2, S_2^0, \rho_2, F_2)$ be two Büchi automata.

We define the **union** automaton to be $\mathcal{A} = (\Sigma, S, S^0, \rho, F)$, where:

- $S = S_1 \cup S_2$,
- $S^0 = S_1^0 \cup S_2^0$,
- $F = F_1 \cup F_2$, and
- $\rho(s, a) = \rho_1(s, a)$ if $s \in S_1$, and $\rho_2(s, a)$ otherwise.

Now for the union Büchi automaton $\mathcal{A}$ (also denoted by $\mathcal{A}_1 \cup \mathcal{A}_2$) it holds that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$.

The $\cap$ definition for Büchi automaton **differs**! (If you use either FSA or Büchi definition in the wrong context, you will get **incorrect results**!)

Definition 8.3 Let $\mathcal{A}_1 = (\Sigma, S_1, S_1^0, \rho_1, F_1)$ and $\mathcal{A}_2 = (\Sigma, S_2, S_2^0, \rho_2, F_2)$ be Büchi automata.

We define the **product** Büchi automaton to be $\mathcal{A} = (\Sigma, S, S^0, \rho, F)$, where:

- $S = S_1 \times S_2 \times \{1, 2\}$,
- $S^0 = S_1^0 \times S_2^0 \times \{1\}$,
- $F = F_1 \times S_2 \times \{1\}$, and
- For all $s \in S_1, t \in S_2, i \in \{1, 2\}$:
  (a) if $s \in F_1 \wedge i = 1 : \rho((s, t, 1), a) = \{(s', t', 2) \mid s' \in \rho_1(s, a) \text{ and } t' \in \rho_2(t, a)\}$
  (b) if $t \in F_2 \wedge i = 2 : \rho((s, t, 2), a) = \{(s', t', 1) \mid s' \in \rho_1(s, a) \text{ and } t' \in \rho_2(t, a)\}$
  (c) if neither (a) or (b) applies:
  $\rho((s, t, i), a) = \{(s', t', j) \mid s' \in \rho_1(s, a), t' \in \rho_2(t, a), \text{ and } j = i\}$

Now for the product Büchi automaton $\mathcal{A}$ (also denoted by $\mathcal{A}_1 \cap \mathcal{A}_2$ or $\mathcal{A}_1 \times \mathcal{A}_2$) it holds that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$.

Actually, we were careful to define the mapping from Kripke structures to finite state automata so that it will work also with Büchi automata:

Definition 8.4 Let $M = (S, s^0, R, L)$ be a Kripke structure over a set of atomic propositions AP. Define a Büchi automaton $\mathcal{A}_M = (\Sigma, S_M, S_M^0, \rho_M, F_M)$, where

- $\Sigma = 2^{AP}$,
- $S_M = S$,
- $S_M^0 = \{s^0\}$,
- for all $s \in S, a \in \Sigma$:
  - if $L(s) \neq a$: $\rho_M(s, a) = \emptyset$,
  - if $L(s) = a$: $\rho_M(s, a) = T$, where $T \subseteq S$ is the largest set such that for all $t \in T$ it holds that $(s, t) \in R$,

  , and
- $F_M = S_M$.

The Büchi automaton $\mathcal{A}_M$ accepts exactly those infinite sequences of labellings which correspond to infinite paths of the Kripke structure starting from the initial state s^0.

We will later show how given an LTL formula f, we can create a Büchi automaton $\mathcal{A}_f$ which accepts exactly all the infinite sequences of valuations which satisfy f.

In model checking we actually negate the property f first, and then create a Büchi automaton $\mathcal{A}_{\neg f}$. This automaton accepts all violations of the property f.

If we have been given $\mathcal{A}_{\neg f}$, it holds that $M \models f$ iff $\mathcal{L}(\mathcal{A}_M \times \mathcal{A}_{\neg f}) = \emptyset$.

In other words: if no path of the Kripke structure is a model of the complement of the specification, then all paths of the Kripke structure are models of the specification.

We'll now show a small trick, using which $\mathcal{A}_M \times \mathcal{A}_{\neg f}$ can be replaced by a slightly smaller automaton, which we call $\mathcal{A}_M \otimes \mathcal{A}_{\neg f}$.

In the special case $F_1 = S_1$ we can actually use a simpler product construction, call it $\mathcal{A}_1 \otimes \mathcal{A}_2$:

Definition 8.5 Let $\mathcal{A}_1 = (\Sigma, S_1, S_1^0, \rho_1, F_1)$ and $\mathcal{A}_2 = (\Sigma, S_2, S_2^0, \rho_2, F_2)$ be two Büchi automata, such that for automaton $\mathcal{A}_1$ it holds that $F_1 = S_1$.

We define the **Kripke product** automaton to be $\mathcal{A} = (\Sigma, S, S^0, \rho, F)$, where:

- $S = S_1 \times S_2$,
- $S^0 = S_1^0 \times S_2^0$,
- $F = S_1 \times F_2$, and
- $\rho((s,t),a) = \rho_1(s,a) \times \rho_2(t,a)$.

Now for the product Büchi automaton $\mathcal{A}$ (also denoted by $\mathcal{A}_1 \otimes \mathcal{A}_2$) it holds that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$.

Now if $\mathcal{A}_1$ is a Kripke structure automaton $\mathcal{A}_M$, it fulfills the property above, and thus this **Kripke product** construction can be used instead. (It has half as many states.)

A Büchi automaton $\mathcal{A}$ is **deterministic** if $|S^0| = 1$ and $|\rho(s,a)| \leq 1$ for all states $s \in S$ and symbols $a \in \Sigma$.

Unlike finite state automata, Büchi automata are not expressively complete when deterministic. In other words, there are languages accepted by non-deterministic Büchi automata, which no deterministic Büchi automaton accepts. An example of such a language is $(a+b)^* b^\omega$.

Also note that LTL requires non-deterministic automata to be expressed, the language above is effectively the same as the requirement expressed by the LTL formula $\Diamond\Box b$.

The complementation procedure for Büchi automata is thus very different from finite state automata, as a normal determinization construction cannot be used. In fact, we have the following result:

Theorem 8.6 Let $\mathcal{A}$ be any (non-deterministic) Büchi automaton with n states. Then in the worst case the smallest Büchi automaton $\mathcal{A}'$, such that $\mathcal{L}(\mathcal{A}') = \Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$ will have $2^{\mathcal{O}(n \log n)}$ states.

This blow-up is much worse than the blow-up for finite state automata complementation. Note that $\mathcal{O}(n!) = \mathcal{O}(2^{\mathcal{O}(n \log n)})$.
(For example, $10! = 3628800$, while $2^{10} = 1024$.)

There are several different ways to complement Büchi automata matching the lower bound. However, we do not know of a publicly available implementation of those algorithms.

Thankfully in (basic) model checking complementation of Büchi automata is not needed.