Reactive Systems: Automata on Infinite Words

Timo Latvala

February 25, 2004

Automata on Infinite Words

To handle model checking of the 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.

Büchi Automata

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

Definition 1 A (nondeterministic) Büchi automaton $\mathcal{A}$ *is a tuple* $(\Sigma, S, S^0, \Delta, F)$, *where*

- Σ *is a finite* alphabet*,*
- S *is a finite set of* states*,*
- $S^0 \subseteq S$ *is set of* initial states*,*
- $\Delta \subseteq S \times \Sigma \times S$ *is the* transition relation*, and*
- $F \subseteq S$ *is the set of* accepting states*.*

The meaning of the transition relation $\Delta \subseteq S \times \Sigma \times S$ is the following: $(s, a, s') \in \Delta$ means that there is a move from state s to state s' with symbol a.

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

A Büchi automaton $\mathcal{A}$ accepts a set of infinite 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 $(s_i, a_i, s_{i+1}) \in \Delta$ for all $i \geq 0$.

Let $\mathit{inf}(r)$ denote the set of states appearing infinitely often in the run 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 $s \in S^0$ an accepting state s' can be reached, such that s' can reach itself by a non-empty sequence of transitions. (I.e., there should be a path from s' back to itself in Δ which contains at least one edge.)

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

Operations for Büchi Automata

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

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

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

Definition 2 *Let* $\mathcal{A}_1 = (\Sigma, S_1, S_1^0, \Delta_1, F_1)$ *and* $\mathcal{A}_2 = (\Sigma, S_2, S_2^0, \Delta_2, F_2)$ *be two Büchi automata.*

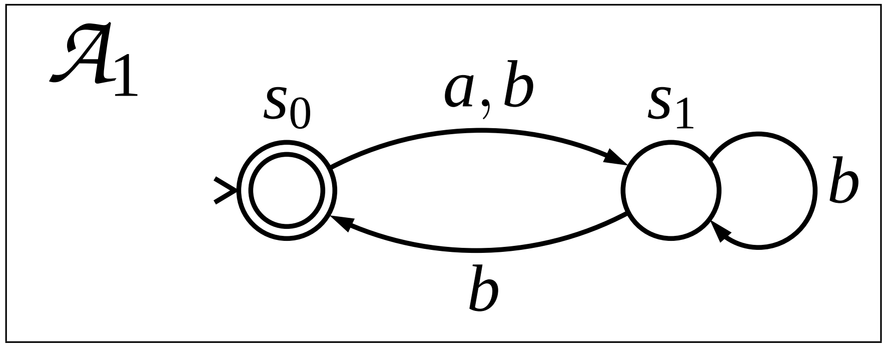
We define the union *Büchi automaton to be* $\mathcal{A} = (\Sigma, S, S^0, \Delta, F)$*, where:*

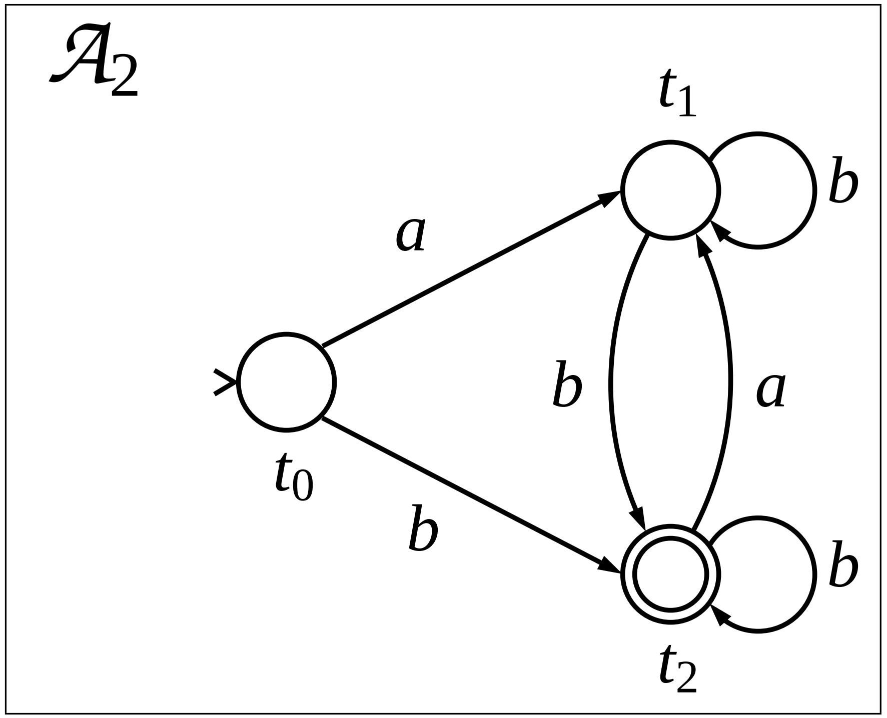
- $S = S_1 \cup S_2$,
- $S^0 = S_1^0 \cup S_2^0$,
- $\Delta = \Delta_1 \cup \Delta_2$, *and*
- $F = F_1 \cup F_2$.

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

Example: Operations on Büchi Automata

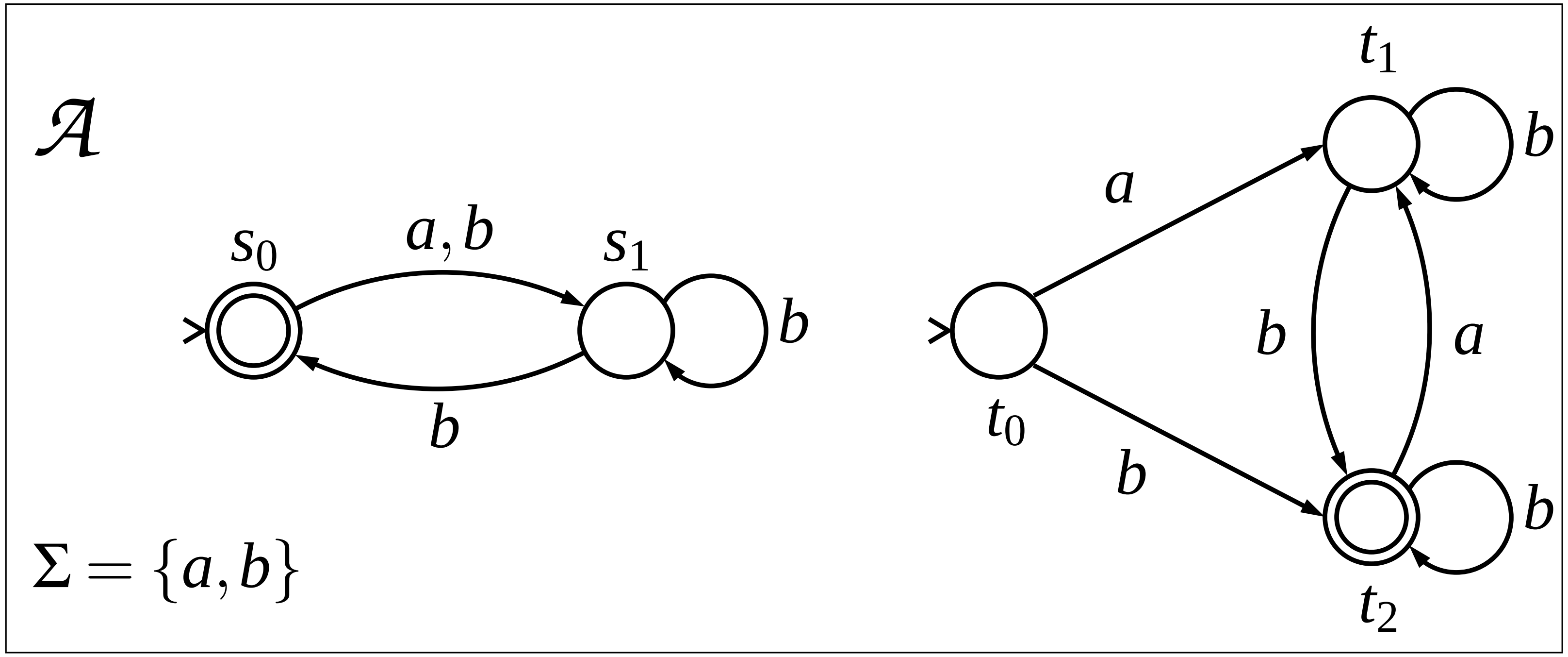
Consider the following Büchi automata $\mathcal{A}_1$ and $\mathcal{A}_2$, both over the alphabet $\Sigma = \{a, b\}$.





Example: Union of Büchi Automata

The following Büchi automaton $\mathcal{A}$ is their union, in other words $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$.



Intersection of Büchi Automata

The $\cap$ definition for Büchi automaton *differs* from the FSA version!

If you use either FSA or Büchi definition in the wrong context, you will get *incorrect results*!

Definition 3 *Let* $\mathcal{A}_1 = (\Sigma, S_1, S_1^0, \Delta_1, F_1)$ *and* $\mathcal{A}_2 = (\Sigma, S_2, S_2^0, \Delta_2, F_2)$ *be Büchi automata.*

We define the product *Büchi automaton to be* $\mathcal{A} = (\Sigma, S, S^0, \Delta, F)$*, 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 $s, s' \in S_1, t, t' \in S_2, a \in \Sigma, i, j \in \{1,2\}$:
  $((s,t,i), a, (s',t',j)) \in \Delta$ iff $(s,a,s') \in \Delta_1$, $(t,a,t') \in \Delta_2$, and:

  a) ($i = 1$, $s \in F_1$, and $j = 2$), or

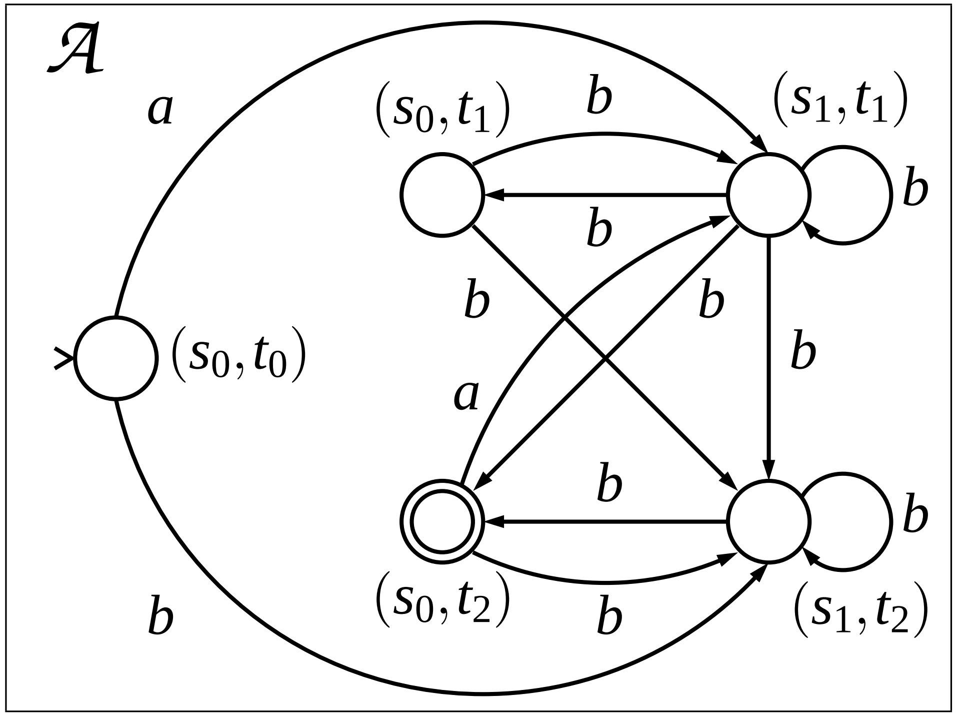
  b) ($i = 2$, $t \in F_2$, and $j = 1$), or

  c) (neither a) or b) above applies 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 $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$.

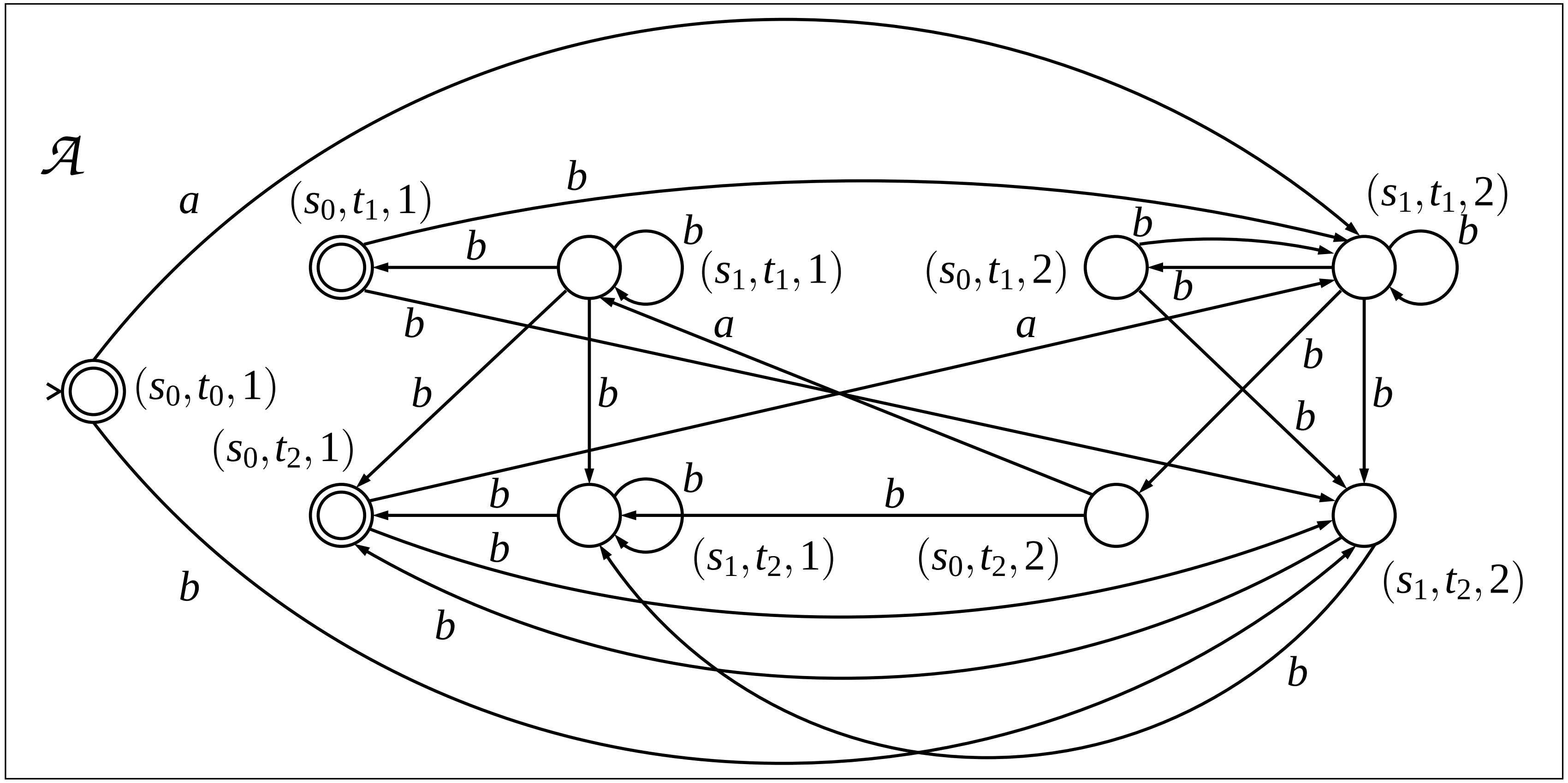
Example: Intersection as Finite State Automata

Recall from previous lectures how the FSA are intersected. (Note: This is NOT the way to deal with Büchi automata, just a reminder of the FSA intersection!) The following FSA automaton $\mathcal{A}$ is the intersection of $\mathcal{A}_1$ and $\mathcal{A}_2$, when they are seen as FSAs.



Example: Intersection of Büchi Automata

The following Büchi automaton $\mathcal{A}$ is the intersection $\mathcal{A} = \mathcal{A}_1 \cap \mathcal{A}_2$.



From Kripke Structure to Büchi Automaton

Actually, we were careful to define the mapping from Kripke structures to finite state automata in such a way that it will work also without modifications with Büchi automata.

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

- $\Sigma = 2^{AP}$,
- $S_M = S \cup \{s^i\}$,
- $S_M^0 = \{s^i\}$,
- For all $s, s' \in S_M, a \in \Sigma : (s, a, s') \in \Delta_M$ iff
  $L(s') = a$ and $(((s, s') \in R)$ or $(s = s^i$ and $s' = s^0))$; 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 some initial state.

We will later show given an LTL formula f, how 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 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.