The Algorithmic Verification of Reactive Systems

Lecture 3:

Automata on Infinite Objects and The Specification of Properties

Motivation

- Infinite behaviors play an important role in the analysis of reactive systems.
- Transition systems generating (Formal Concurrent Systems) or recognizing (observers) infinite behaviors are useful in the analysis of reactive systems.
- There exists a rich theory for such transition systems with a *finite* number of states: the theory of finite-automata on infinite words.
- We will now study this theory as well as its generalization to infinite trees.
- In this theory, the objects recognized by the automata are infinite words (trees) defined over a finite alphabet.

Words and Automata

- A finite word of length n over an alphabet Σ is a mapping $w : \{0, \dots n-1\} \to \Sigma$

- An automaton on finite words is a description of a set of finite words (those that it accepts).

- An infinite word (ω-word) over an alphabet Σ is a mapping $w : \mathbf{N} \to \Sigma$

- An automaton on infinite words is a description of a set of infinite words (those that it accepts).

Note: Automata are often introduced as a model of computation. Here, we will just view them as a description of a set of words.

Automata on Infinite Words

Same structure as automata on finite words:

$A = \{\Sigma, S, \delta, S_0, F\}$ where

- Σ is an alphabet,
- S is a set of states,
- $\delta : S \times \Sigma \rightarrow S$ (deterministic) or
  $\delta : S \times \Sigma \rightarrow 2^S$ (nondeterministic) is a transition function,
- $S_0 \subseteq S$ is a set of initial states, and
- $F \subseteq S$ is a set of accepting states.

What changes is the semantics.

When does an Automaton accept a Word?

From now on, we consider the more general case of nondeterministic automata.

Finite words $w : \{0, \ldots n-1\} \to \Sigma$

The word w is accepted by the automaton if there is a labeling of the word by states

$$\rho : \{0, \ldots n\} \to S$$

such that

- $\rho(0) \in S_0$ (the initial label is an initial state),
- $\forall 0 \leq i \leq n-1,\ \rho(i+1) \in \delta(w(i), \rho(i))$ (the labeling is compatible with the transition relation),
- $\rho(n) \in F$ (the last label is an accepting state).

Don't think of automata as programs, but as a formalism for defining sets of words that has a very simple semantics.

Infinite words $w : \mathbf{N} \to \Sigma$

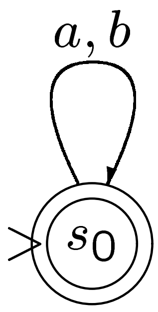
The word w is accepted by the automaton if there is a labeling of the word by states

$$\rho : \mathbf{N} \to S$$

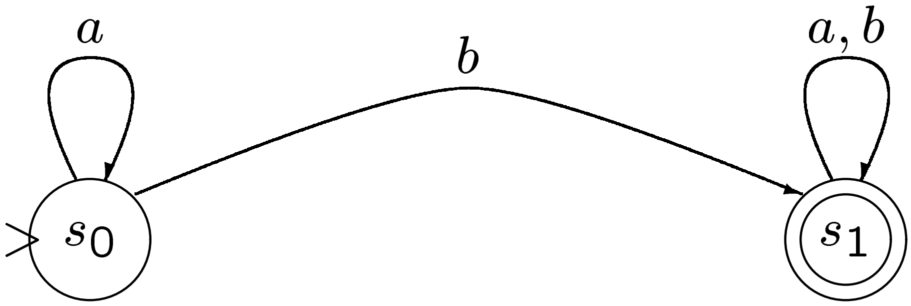
such that

- $\rho(0) \in S_0$ (the initial label is an initial state),
- $\forall 0 \leq i,\ \rho(i+1) \in \delta(w(i), \rho(i))$ (the labeling is compatible with the transition relation),
- $inf(\rho) \cap F \neq \emptyset$ where $inf(\rho)$ is the set of states that appear infinitely often in ρ (the set of repeating states intersects F).

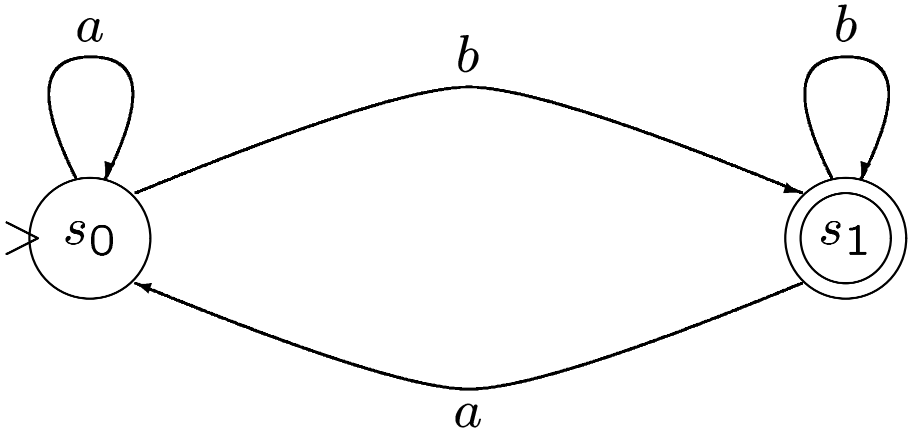
Examples



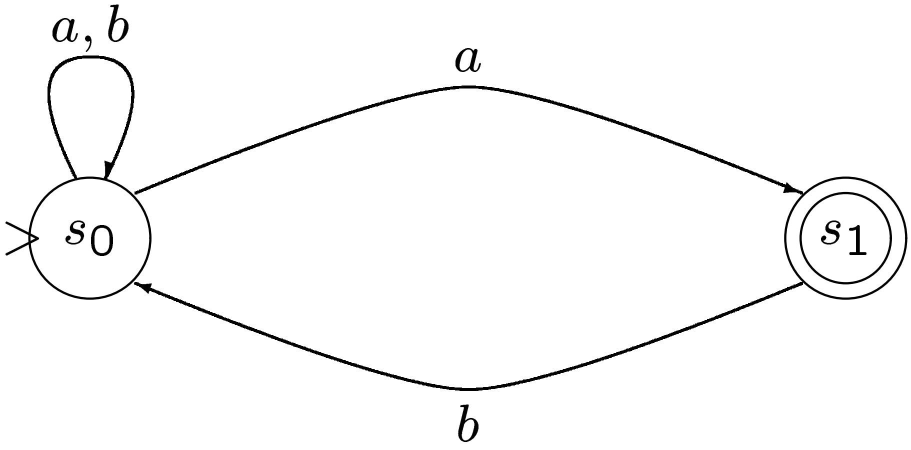
All words over $\Sigma = \{a, b\}$



An automaton accepting $a^\star b(a \cup b)^\omega$



All words containing b infinitely often



Infinitely often ab

Other types of Acceptance conditions

Büchi: $F \subseteq S$,
$\text{inf}(\rho) \cap F \neq \emptyset$.

Generalized Büchi: $\mathcal{F} \subseteq 2^S$,
i.e. $F = \{F_1, \ldots, F_m\}$
For each F_i, $\text{inf}(\rho) \cap F_i \neq \emptyset$.

Rabin: $F \subseteq 2^S \times 2^S$,
i.e. $F = \{(G_1, B_1), (G_2, B_2), \ldots, (G_m, B_m)\}$
For some pair $(G_i, B_i) \in F$, $\text{inf}(\rho) \cap G_i \neq \emptyset$ and $\text{inf}(\rho) \cap B_i = \emptyset$

Streett: $F \subseteq 2^S \times 2^S$,
i.e. $F = \{(G_1, B_1), (G_2, B_2), \ldots, (G_m, B_m)\}$
For all pairs $(G_i, B_i) \in F$, $\text{inf}(\rho) \cap G_i = \emptyset$ or $\text{inf}(\rho) \cap B_i \neq \emptyset$

For nondeterministic automata, all define the ω-regular languages :

$$\bigcup_i \alpha_i \beta_i^\omega$$

where α_i and β_i are finite-word regular languages and ω denotes infinite repetition.

For deterministic automata, Büchi and Generalized Büchi conditions are weaker.

Properties of Büchi Automata

- Closed under union, intersection, projection, and complementation.
- Nonemptiness easy to decide:
  - Check if some accepting state is accessible from an initial state and (nontrivially) from itself.
  - Linear time (compute the strongly connected components).
  - NLOGSPACE.
- For Rabin conditions, nonemptiness can also be decided in polynomial time. The simplest algorithms involve a translation to nondeterministic Büchi automata.
- For Street automata, the translation to nondeterministic Büchi automata is exponential.

Closure under Union

Given $A_1 = (\Sigma, S_1, \delta_1, S_{01}, F_1)$ and
$A_2 = (\Sigma, S_2, \delta_2, S_{02}, F_2)$,

the automaton $A = \{\Sigma, S, \delta, S_0, F\}$ where

- $S = S_1 \cup S_2, \quad S_0 = S_{01} \cup S_{02}, \quad F = F_1 \cup F_2,$

- $t \in \delta(s, a)$ if
$$\begin{cases} \text{either } t \in \delta_1(s,a) & \text{and } s \in S_1, \\ \text{or } t \in \delta_2(s,a) & \text{and } s \in S_2. \end{cases}$$

accepts $L(A_1) \cup L(A_2)$.

Closure under Intersection

Given $A_1 = (\Sigma, S_1, \delta_1, S_{01}, F_1)$
and $A_2 = (\Sigma, S_2, \delta_2, S_{02}, F_2)$,

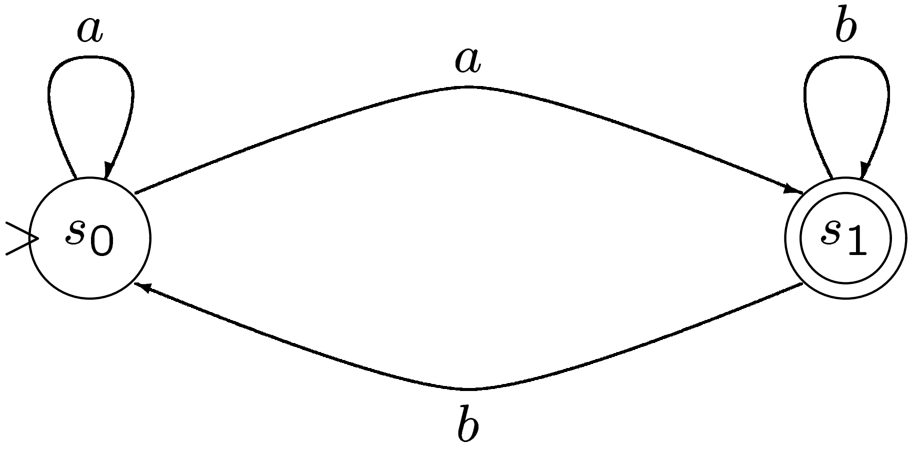
The generalized Büchi automaton and
$A = \{\Sigma, S, \delta, S_0, \mathcal{F}\}$ where

- $S = S_1 \times S_2, \quad S_0 = S_{01} \times S_{02}$,
- $\mathcal{F} = \{F_1 \times S_2, F_2 \times S_1\}$,
- $(u, v) \in \delta((s, t), a)$ if $u \in \delta_1(s, a)$ and $v \in \delta_2(t, a)$.

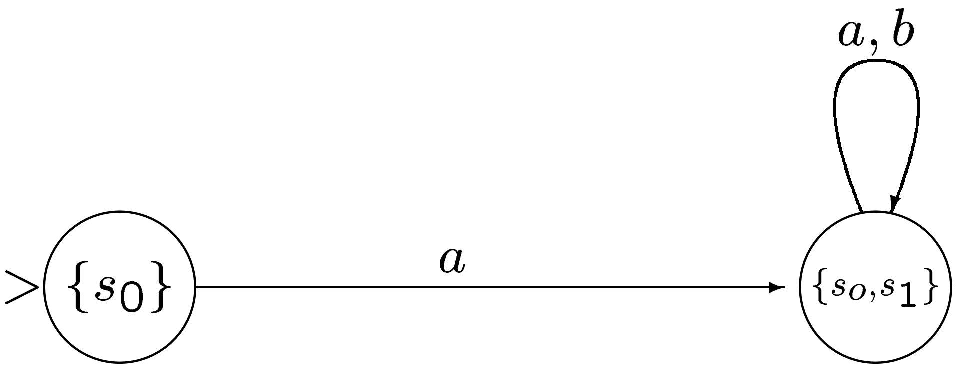
accepts the language $L(A_1) \cap L(A_2)$.

Projection is also simple, but *complementation* is completely nontrivial (the Rabin-Scott subset construction is not sufficient!).

Example:
Failure of the Rabin-Scott Construction



A simple Büchi Automaton



The Rabin-Scott construction applied to the automaton above

From Generalized Büchi to Büchi

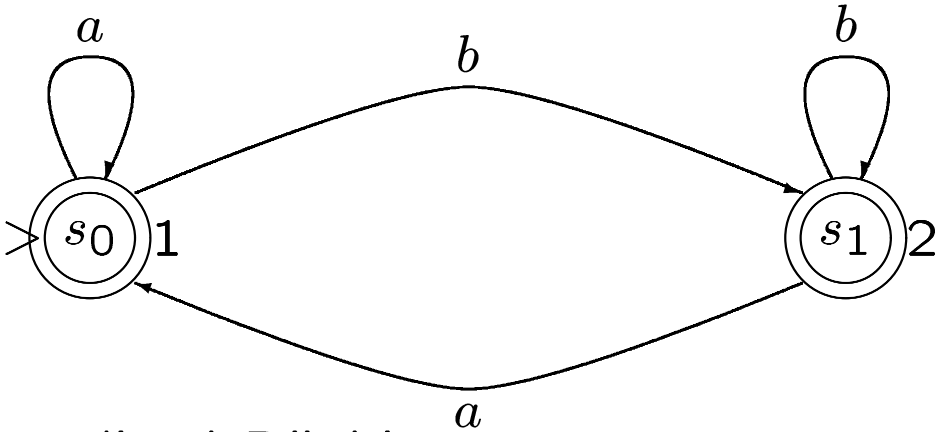
Given $A = (\Sigma, S, \delta,\ S_0, \mathcal{F})$,
where $\mathcal{F} = \{F_1, \ldots, F_k\}$,

$A' = (\Sigma, S', \delta', S'_0, F')$ where

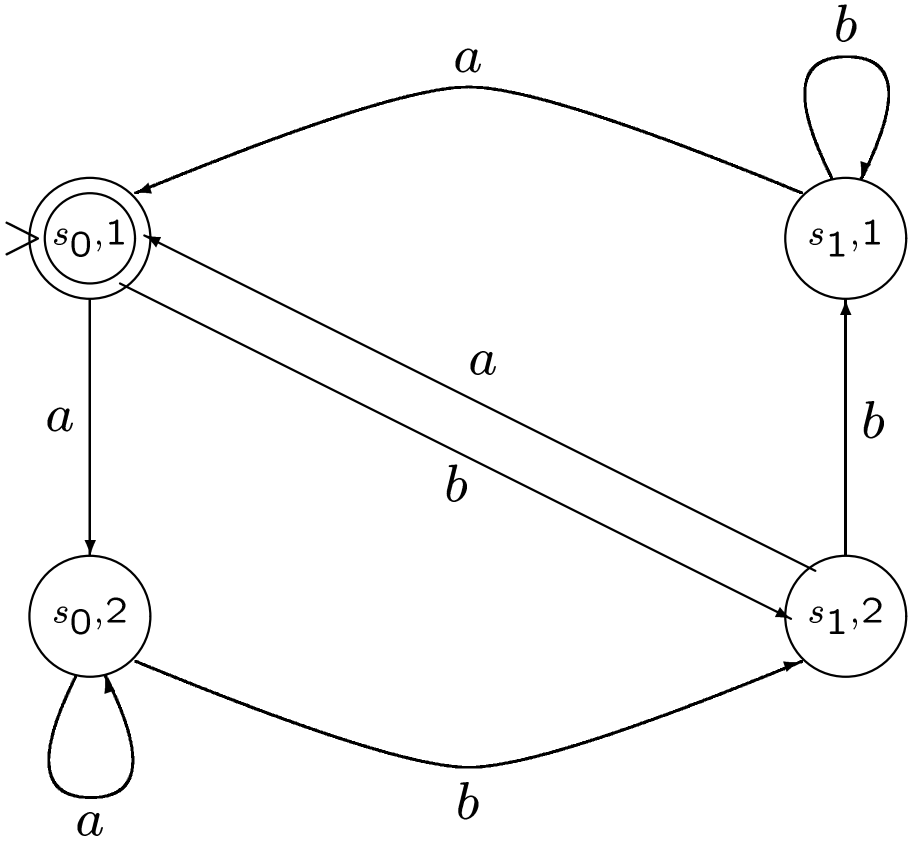
- $S' = S \times \{1, \ldots, k\}$.
- $S'_0 = S_0 \times \{1\}$.
- δ' is defined by $(t, i) \in \delta'((s, j), a)$ if

  $t \in \delta(s, a)$ and $\begin{cases} i = j & \text{if } s \notin F_j, \\ i = (j \bmod k) + 1 & \text{if } s \in F_j. \end{cases}$
- $F' = F_1 \times \{1\}$.

accepts $L(A)$.



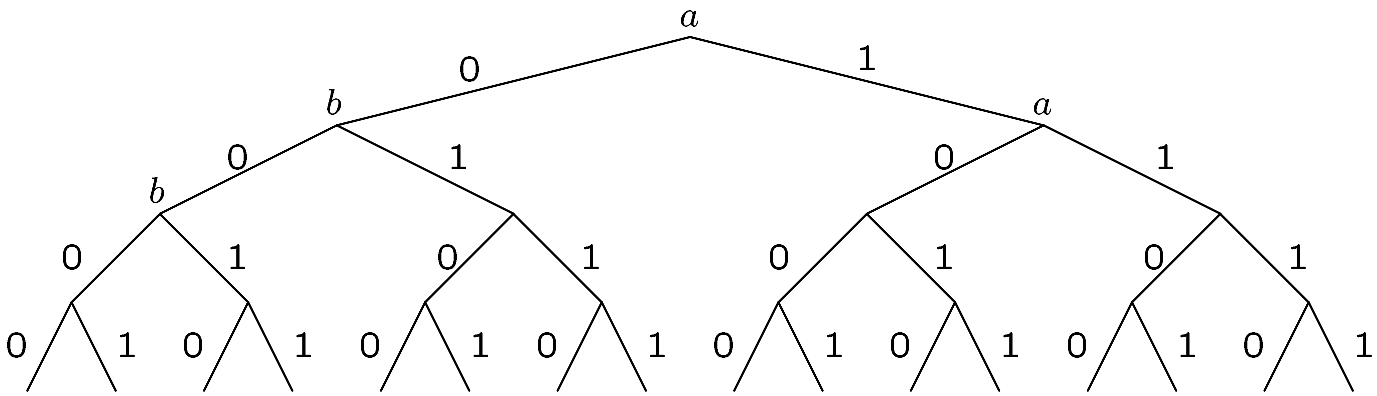
A generalized Büchi automaton



From generalized Büchi to Büchi

Automata on infinite trees

For simplicity we consider only binary labeled trees : $T = \{0,1\}^* \to \Sigma$



$\vdots$

An automaton $A = (\Sigma, S, \delta, S_0, F)$ is a description of a set of trees that has the form of a labeling rule (if the tree can be successfully labeled, it is in the set):

- Label the root of the tree with a state in S_0.
- Propagate the labeling to the other nodes according to the transition relation.
- The labeling is successful if it satisfies the acceptance condition F.

Types of transition relations

Deterministic: $\delta : S \times \Sigma \to S \times S$.

$$\delta(s, a) = (s_1, s_2)$$

If a node x labeled by a is assigned state s, the nodes $x0$ and $x1$ have to be labeled by s_1 and s_2 respectively.

Nondeterministic: $\delta : S \times \Sigma \to 2^{S \times S}$.

$$\delta(s, a) = \{(s_1, s_2), (s'_1, s'_2), (s''_1, s''_2)\}$$

If a node x labeled by a is assigned state s, the nodes $x0$ and $x1$ have to be respectively labeled by s_1 and s_2, s'_1 and s'_2, or s''_1 and s''_2.

Runs (Labelings) by Automata

A run assigns a state to each note of the tree: $\rho : \{0, 1\}^* \to S$.

- A state $s \in S_0$ assigned to ε.
- Compatible with the transition relation:
  - $\forall x \in \{0, 1\}^*$,
    $(\rho(x0), \rho(x1)) = \delta(\rho(x), T(x))$
    (deterministic)
  - $\forall x \in \{0, 1\}^*$,
    $(\rho(x0), \rho(x1)) \in \delta(\rho(x), T(x))$ (non deterministic)
- Acceptance condition: Büchi, Rabin, . . . *on all paths*

Properties of Büchi Tree Automata

- Closed under union, intersection and projection.
- Nonemptiness can be decided:
  - Repeatedly eliminate states from which there is no run that reaches an accepting state on all paths.
  - Can be done in quadratic time.
  - PTIME-complete.
- For Rabin and Street conditions, nonemptiness can also be decided, but the algorithms are substantially more complicated and of higher complexity (NP-complete for Rabin conditions, worse for Street conditions).
- One has closure under complementation for Rabin and Street tree automata, but not for Büchi tree automata.

A More General Type of Automata: Alternating Automata

The transition relation of a nondeterministic automaton is of the form

$$\delta : S \times \Sigma \to 2^{S \times S}.$$

For example,

$$\delta(s,a) = \{(s_1, s_2), (s'_1, s'_2), (s''_1, s''_2)\},$$

which can also be written

$$\delta(s,a) = ((0,s_1) \wedge (1,s_2)) \vee ((0,s'_1) \wedge (1,s'_2)) \vee ((0,s''_1) \wedge (1,s''_2))$$

This suggest the idea of having arbitrary positive Boolean formulas as transitions, which is the definition of **Alternating** automata:

$$\delta : S \times \Sigma \to \mathcal{B}^+(\{0,1\} \times S).$$

For example,

$$\delta(s,a) = ((0,s_1) \wedge (1,s_2)) \wedge ((1,s_3) \wedge (1,s_4)) \vee ((1,s_1) \wedge (0,s_2))$$

If a node x labeled by a is assigned state s, the nodes $x0$ and $x1$ have to be labeled in a way that makes the positive Boolean formula true.

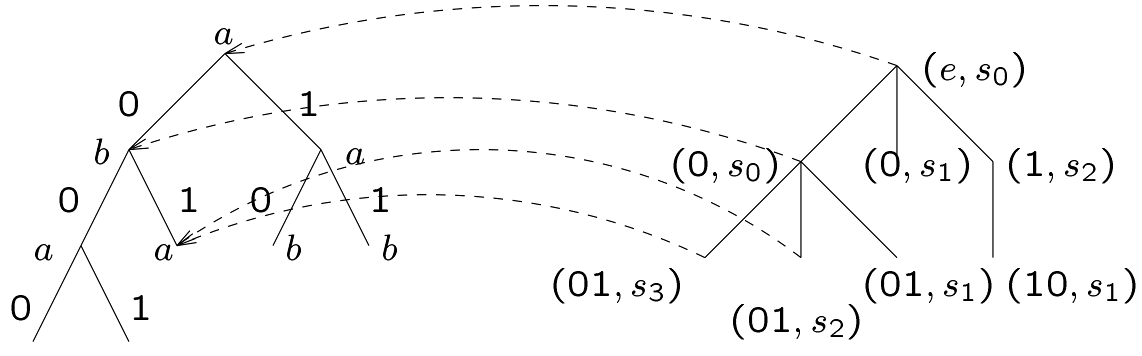
Runs of Alternating Automata

Given a tree $T : \{0, 1\}^* \to \Sigma$, a run of an alternating automaton can assign several states to each node (the run has to accept from all of these)..

In order to check the acceptance conditions, one needs to separate the conjunctively related states labeling a given node of the tree. This is done with the help a *run tree* $\langle T_r, r\rangle$, with

$$r : T_r \to \{0, 1\}^* \times S$$

in which multiply labeled nodes are duplicated.



A tree is then accepted is there is a run tree such that:

- Its root is labeled by a state $s \in S_0$,
- At every node the transition relation is satisfied.
- On every infinite path the acceptance condition is satisfied.

Properties of Alternating Automata

- Closure under union and intersection is immediate.
- Closure under complementation is direct, except with respect to the acceptance condition.
- Projection is a hard operation.
- Checking for nonemptiness is hard (exponential even for the simplest acceptance conditions).

Projection and Alternation: An Example

Consider the alternating automaton on finite words (finite unary trees) $A = (\Sigma, S, S_0, \delta, F)$ with

$$\Sigma = \{a, b, c\}$$

$$S = \{s_0, s_1, s_2, s_3, s_4\}$$

$$S_0 = \{s_0\}$$

$$F = \{s_3\}$$

$$\begin{array}{rcl}
\delta(s_0, a) & = & s_1 \wedge s_2 \\
\delta(s_1, b) & = & s_3 \\
\delta(s_1, c) & = & s_4 \\
\delta(s_2, b) & = & s_4 \\
\delta(s_2, c) & = & s_3
\end{array}$$

This automaton does not accept any word, but its naive projection $(b, c \to a)$ does. For the same reason, nonemptiness cannot be reduced to one letter nonemptiness.

Conclusions

- Automata on infinite objects are an expressive formalism for representing properties of reactive systems.
- There is a rich algorithmic theory of these automata.
- The expression of properties by automata on infinite objects is often not intuitive.
- Fortunately, what they can express can also be expressed by more intuitively readable logical languages.
- Such languages and in particular temporal logics will be the topic of next week's lecture.