Bernd Finkbeiner
Sven Schewe

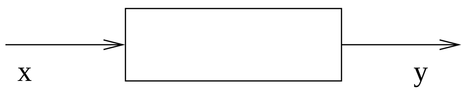
Date: April 17, 2008

Automata, Games and Verification: Lecture 1

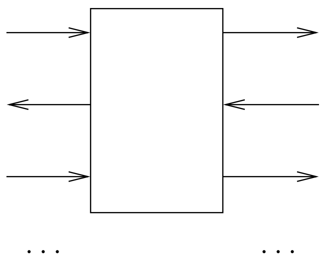
1 Motivation

We distinguish

- Transformational programs



- Reactive systems



  - nonterminating behavior
  - interaction (program vs. environment)

1.1 Problem 1: Verification

Example: Mutual execution with program TURN

$$\text{local } t\text{: boolean where initially } t = 0$$

$$P_0 :: \left[\begin{array}{l} \text{loop forever do} \\ \qquad \left[\begin{array}{ll} 00: & \text{noncritical;} \\ 01: & \text{await } t = 0; \\ 10: & \text{critical;} \\ 11: & t := 1; \end{array}\right] \end{array}\right] \;||\; P_1 :: \left[\begin{array}{l} \text{loop forever do} \\ \qquad \left[\begin{array}{ll} 00: & \text{noncritical;} \\ 01: & \text{await } t = 1; \\ 10: & \text{critical;} \\ 11: & t := 0; \end{array}\right] \end{array}\right]$$

TURN is a finite-state program with 32 states, which can be encoded as bit vectors $(b_1, b_2, b_3, b_4, b_5)$, with (b_1, b_2) for the location of P_0, (b_3, b_4) for the location of P_1, and b_5 for t.

Behavior: infinite sequence of states
Specification: set of correct behaviors

Example: specifications:

- Mutual execution: it is never the case that P_0 and P_1 are in their critical sections, i.e. the states 10100 and 10101 do not occur
- Accessibility: whenever P_i is in location 01 it will eventually reach location 10

The Verification Problem: Given a program P and a specification φ, decide whether P satisfies φ.

Underlying concept: Automata over infinite words (more generally: objects)

Solution:

1. Construct automaton that accepts all sequences that are
   - possible in P and
   - violate φ.
2. Check automaton for emptiness.

1.2 Problem 2: Synthesis

Example: Mutual execution by arbiter

$$\text{local } t, r_1, r_2\text{: boolean where initially } t = r_1 = r_2 = 0$$

$$P_0 :: \left[\begin{array}{l} \text{loop forever do} \\ \quad \left[\begin{array}{ll} 00: & r_0 := 1; \\ 01: & \text{await } t = 0; \\ 10: & \text{critical;} \\ 11: & r_0 := 0; \end{array}\right] \end{array}\right] \; || \; P_1 :: \left[\begin{array}{l} \text{loop forever do} \\ \quad \left[\begin{array}{ll} 00: & r_1 := 1; \\ 01: & \text{await } t = 1; \\ 10: & \text{critical;} \\ 11: & r_1 := 0; \end{array}\right] \end{array}\right] \; || \; \text{Arbiter:: } \mathbf{?}$$

The Synthesis Problem: Given a specification φ, decide if *there exists* a program P that satisfies φ. If yes: construct such a program.

Underlying concept: Infinite games.
Play of the game = infinite sequence of states.
Player "system" wins the game if sequence satisfies φ for all possible behaviors of player "environment".

Solution:

1. Decide whether player "system" has a winning strategy.
2. If yes, construct a program that implements that strategy.

1.3 History

1960 – 1970 Fundamental results about ω-automata and games. Motivation: Logical decision problems, circuit design.

- **J. Richard Büchi** (1924-1984)
  Swiss logician and mathematician; Ph.D. at ETH, then Purdue University, Lafayette, Indiana. Inventor of Büchi automata. Great influence on theoretical computer science, combinatorics, grapth theory.
- **Robert McNaughton**
  taught philosophy; then switched to computer science in 1950s; emeritus at Harvard; McNaughton's theorem: each recognizable set of infinite words can be recognized by a deterministic ω-automaton.
- **Michael Rabin** (*1931, Breslau)
  won Turing award together with Dana Scott for inventing nondeterministic machines; proved that second order theory of n successors is decidable; determinacy of parity games.

Since 1980: Revival of the theory in the setting of temporal logics

Motivation today:

- industrial use (especially finite-state verification "model checking")
- decidability of many problems with infinite structures
- bridge between logic and computer science

2 Büchi Automata

2.1 Basic Definitions

- The *set of natural numbers* $\{0, 1, 2, 3, \ldots\}$ is denoted by ω.
- An *alphabet* Σ is a finite set of symbols.
- An *infinite sequence/string/word* is a function from natural numbers to an alphabet:
  $\alpha : \omega \to \Sigma$
  An infinite word is composed of its letters, so that in particular $\alpha = \alpha(0)\alpha(1)\alpha(2)\ldots$
- The *set of infinite words over alphabet* Σ is denoted Σ^ω (finite words: Σ^*).
- An *ω-language* L is a subset of Σ^ω.

Example:

- $\emptyset$ is the *empty* ω-language.

- $\{a^\omega\} = \{aaaa\ldots\}$;
- $\{ba^\omega, aba^\omega, aaba^\omega, \ldots\}$.

✚

Definition 1 *A* nondeterministic Büchi automaton $\mathcal{A}$ *over alphabet* Σ *is a tuple* (S, I, T, F)*:*

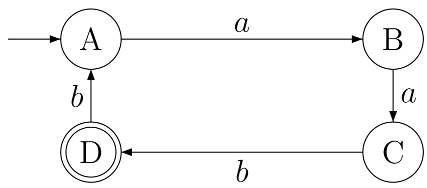
- S *: a finite set of* states
- $I \subseteq S$ *: a subset of* initial states
- $T \subseteq S \times \Sigma \times S$ *: a set of* transitions
- $F \subseteq S$ *: a subset of* accepting/final states

Now we define how a Büchi automaton uses an infinite word as input. Notice that we do not refer to acceptance in this definition.

Definition 2 *A* run *of a nondeterministic Büchi automaton* $\mathcal{A}$ *on an infinite input word* $\alpha = \sigma_0\sigma_1\sigma_2\ldots$ *is an infinite sequence of states* $s_0, s_1, s_2, \ldots$ *such that the following hold:*

- $s_0 \in I$
- *for all* $i \in \omega$, $(s_i, \sigma_i, s_{i+1}) \in T$

Example:



In the automaton shown the set of states are $S = \{A, B, C, D\}$, the initial set of states are $I = \{A\}$ (indicated with pointing arrow with no source), the transitions $T = \{(A, a, B), (B, a, C), (C, b, D), (D, b, A)\}$ are the remaining arrows in the diagram, and the set of accepting states is $F = \{D\}$ (double-lined state circle).

On input $aabbaabb\ldots$ the Büchi automaton shown has only the run:

$ABCDABCDABCD\ldots$ ✚

Determinism is a property of machines that can only react in a unique way to their input. The following definition makes this clear for Büchi automata.

Definition 3 *A Büchi automaton $\mathcal{A}$ is* deterministic *when T is a partial function (with respect to the next input letter and the current state):*
$\forall \sigma \in \Sigma, \forall s, s_0, s_1 \in S \,.\, (s, \sigma, s_0) \in T$ *and* $(s, \sigma, s_1) \in T \;\Rightarrow\; s_0 = s_1$
and I is a singleton.

(By Büchi automaton we usually mean nondeterministic Büchi automaton.)

Definition 4 *The* infinity set *of an infinite word $\alpha \in \Sigma^\omega$ is the set* $In(\alpha) = \{\sigma \in \Sigma \mid \forall i \exists j \,.\, j \geq i$ *and* $\alpha(j) = \sigma\}$

Definition 5

- *A Büchi automaton $\mathcal{A}$* accepts *an infinite word α if:*
  - *there is a run $r = s_0 s_1 s_2 \ldots$ of α on $\mathcal{A}$*
  - *r is* accepting*: $In(r) \cap F \neq \emptyset$*
- *The* language recognized by Büchi automaton $\mathcal{A}$ *is defined as follows:*
  $\mathcal{L}(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \mathcal{A}$ *accepts* $\alpha\}$

Example: Automaton $\mathcal{A}$ from previous example. $\mathcal{L}(\mathcal{A}) = \{aabbaabbaabb\ldots\}$. ✚

Comment: A deterministic Büchi automaton $\mathcal{A} = (S, I, T, F)$ defines a partial function[1] from Σ^ω to a set of runs $R \subseteq S^\omega$. **End Comment**

Definition 6 *An ω-language L is Büchi recognizable if there is a Büchi automaton $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = L$.*

Example: The singleton ω-language $L = \{\sigma\}$ with $\sigma = abaabaaabaaaab\ldots$ is not Büchi recognizable. (Note that all finite languages of finite words are NFA-recognizable. Analog result does not hold for Büchi-automata)

Proof:

- Suppose there is a Büchi automaton $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) = L$.
- Let $r = s_0 s_1 \ldots$ be an accepting run on σ.
- Since F is finite, there exists $k, k' \in \omega$ with $k < k'$ and $s_k = s_{k'} \in F$.
- $r' = r_0 \ldots r_{k'-1}(r_k \ldots r_{k'-1})$ is an accepting run on $\sigma' = \sigma(0) \ldots \sigma(k'-1)(\sigma(k) \ldots \sigma(k'-1))^\omega$.
- Hence, $\sigma' \in \mathcal{L}(\mathcal{A})$. Contradiction.

■

✚

Definition 7 *A Büchi automaton is* complete *if its transition relation contains a function:*
$\forall s \in S \sigma \in \Sigma \exists s' \in S \,.\, (s, \sigma, s') \in T$

[1] A *partial function* is a function that is not defined on all of the elements of its domain.

Theorem 1 *For every Büchi automaton $\mathcal{A}$, there is a complete Büchi automaton $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

Proof:

We define $\mathcal{A}'$ in terms of the components S, I, T, F of $\mathcal{A}$:

$$S' = S \cup \{f\} \qquad f \text{ new}$$

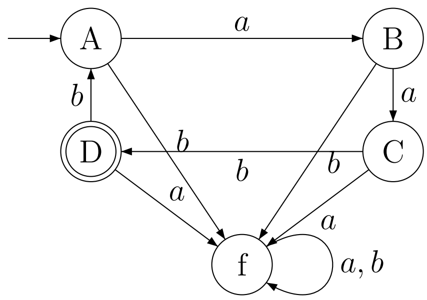
$$I' = I$$

$$T' = T \cup \{(s, \sigma, f) \mid \nexists s' . (s, \sigma, s') \in T\} \cup \{(f, \sigma, f) \mid \sigma \in \Sigma\}$$

$$F' = F$$

The runs of $\mathcal{A}'$ are a superset of those of $\mathcal{A}$ since we have added states and transistions. Furthermore, on any infinite input word α the accepting runs of $\mathcal{A}$ and $\mathcal{A}'$ correspond, because any run that reaches f stays in f, and since $f \notin F'$, such a run is not accepting. ■

Example: Completing the Büchi automaton from a previous example we obtain the following automaton:



Unless we specify otherwise, we will only consider complete automata when we prove results.

Comment: A complete deterministic Büchi automaton $\mathcal{A} = (S, I, T, F)$ may be viewed as a total function[2] from Σ^ω to S^ω. A complete (possibly nondeterministic) Büchi automaton can produce at least one run for every Σ^ω input word.
End Comment

[2]A total function, in contrast to a partial one, is defined on its entire domain.