University of Antwerp

Specification and Verification

Lecture 4: Omega regular-languages and omega automata

Guillermo A. Pérez

October 14, 2024

TL;DR: This lecture in short

What are they? Why study them?

Omega automata recognize infinite-word languages; they allow us to express languages of *infinite* words (e.g., $\mathrm{Words}(\varphi)$) using a *finite* automaton

Main references

- Christel Baier, Joost-Pieter Katoen: **Principles of Model Checking.** MIT Press 2018.
- Mickael Randour: Verification course @ UMONS.

Required and target competences

What tools do we need?

Discrete maths, formal language theory

What skills will we obtain?

- theory: we introduce the automata-theoretic approach to specifying and model checking systems
- practice: the main model checking tool comes from formal language theory (emptiness checks)

How will these skills be useful?

Model checking and synthesis of reactive systems is mostly automata and game based

University of Antwerp

Finite-state automata: reminder

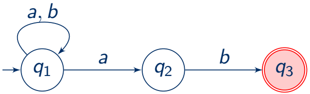
Automata describing languages of *finite* words.

Definition: non-deterministic finite-state automaton (NFA)

Tuple $\mathcal{A} = (Q, A, \delta, I, F)$ with

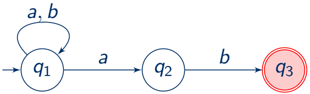
- Q a finite set of states,
- A a finite alphabet,
- $\delta\colon Q \times A \to 2^Q$ a transition function,
- $I \subseteq Q$ a set of initial states,
- $F \subseteq Q$ a set of accepting (or final) states.

Finite-state automata: example



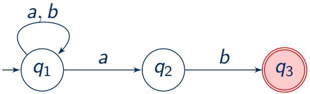
- $Q = \{q_1, q_2, q_3\}$, $A = \{a, b\}$, $I = \{q_1\}$, $F = \{q_3\}$.

Finite-state automata: example



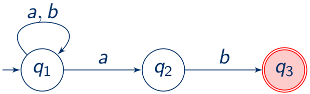
- $Q = \{q_1, q_2, q_3\}$, $A = \{a, b\}$, $I = \{q_1\}$, $F = \{q_3\}$.
- This automaton is non-deterministic: see letter a on state q_1

Finite-state automata: example



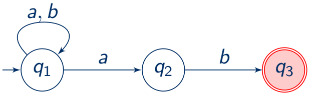
- $Q = \{q_1, q_2, q_3\}$, $A = \{a, b\}$, $I = \{q_1\}$, $F = \{q_3\}$.
- This automaton is non-deterministic: see letter a on state q_1
- Language?

Finite-state automata: example



- $Q = \{q_1, q_2, q_3\}$, $A = \{a, b\}$, $I = \{q_1\}$, $F = \{q_3\}$.
- This automaton is non-deterministic: see letter a on state q_1
- Language?
  - Finite word $w = a_0 a_1 \ldots a_n \in A^*$
  - A run of $\mathcal{A}$ on w is a sequence $q_0 q_1 \ldots q_{n+1}$ such that $q_0 \in I$ and for all $0 \leq i \leq n$, $q_{i+1} \in \delta(q_i, a_i)$
  - $w \in \mathcal{L}(\mathcal{A})$ if there exists a run $q_0 q_1 \ldots q_{n+1}$ for w such that $q_{n+1} \in F$

Finite-state automata: example



- $Q = \{q_1, q_2, q_3\}$, $A = \{a, b\}$, $I = \{q_1\}$, $F = \{q_3\}$.
- This automaton is non-deterministic: see letter a on state q_1
- Language?
  - Finite word $w = a_0 a_1 \ldots a_n \in A^*$
  - A run of $\mathcal{A}$ on w is a sequence $q_0 q_1 \ldots q_{n+1}$ such that $q_0 \in I$ and for all $0 \leq i \leq n$, $q_{i+1} \in \delta(q_i, a_i)$
  - $w \in \mathcal{L}(\mathcal{A})$ if there exists a run $q_0 q_1 \ldots q_{n+1}$ for w such that $q_{n+1} \in F$

$\hookrightarrow$ Here, $\mathcal{L}(\mathcal{A}) = (a \mid b)^* a\, b$, i.e., all words ending by "ab"

NFAs & regular expressions

Recall that NFAs correspond to **regular languages**, which can be described by *regular expressions*.

Syntax

Regular expressions over letters $A \in A$ are formed by

$$E ::= \varnothing \mid \varepsilon \mid A \mid E + E' \mid E \cdot E' \mid E^*$$

Semantics

For regular expression E, language $\mathcal{L}(E) \subseteq A^*$ obtained by

$$\mathcal{L}(\varnothing) = \mathcal{L}(E \cdot \varnothing) = \varnothing, \mathcal{L}(\varepsilon) = \{\varepsilon\}, \mathcal{L}(A) = \{A\}, \mathcal{L}(E^*) = \mathcal{L}(E)^*,$$

$$\mathcal{L}(E + E') = \mathcal{L}(E) \cup \mathcal{L}(E'), \mathcal{L}(E \cdot E') = \mathcal{L}(E) \cdot \mathcal{L}(E')$$

Syntactic sugar: we often write $E \mid E'$ for $E + E'$, E^+ for $E \cdot E^*$ and we drop the concatenation operator, i.e., EE' instead of $E \cdot E'$.

Finite-state automata: DFAs vs. NFAs

Expressiveness

Deterministic FAs (DFAs) are *expressively equivalent* to NFAs, i.e., for any NFA, there exists a DFA recognizing the same language.

Finite-state automata: DFAs vs. NFAs

Expressiveness

Deterministic FAs (DFAs) are *expressively equivalent* to NFAs, i.e., for any NFA, there exists a DFA recognizing the same language.

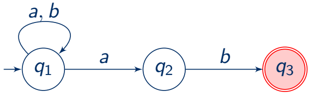
$\Longrightarrow$ **One can determinize any NFA through a (potentially exponential) subset construction**

Finite-state automata: DFAs vs. NFAs

Expressiveness

Deterministic FAs (DFAs) are *expressively equivalent* to NFAs, i.e., for any NFA, there exists a DFA recognizing the same language.

$\Longrightarrow$ **One can determinize any NFA through a (potentially exponential) subset construction**



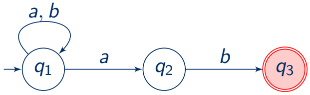
$\Longrightarrow$ **Blackboard illustration**

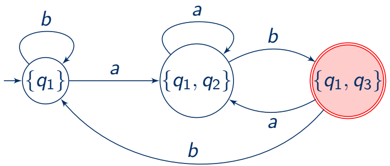
Finite-state automata: DFAs vs. NFAs

Expressiveness

Deterministic FAs (DFAs) are *expressively equivalent* to NFAs, i.e., for any NFA, there exists a DFA recognizing the same language.

$\Longrightarrow$ One can determinize any NFA through a (potentially exponential) subset construction





$\Longrightarrow$ Blackboard illustration

University of Antwerp

ω-regular languages

Intuitively, extension of regular languages to *infinite* words.

Syntax

An ω-regular expression G over A has the form

$$G = E_1 \cdot F_1^{\omega} + \cdots + E_n \cdot F_n^{\omega} \text{ for } n > 0$$

where E_i, F_i are regular expressions over A with $\varepsilon \notin \mathcal{L}(F_i)$.

University of Antwerp

ω-regular languages

Intuitively, extension of regular languages to *infinite* words.

Syntax

An ω-regular expression G over A has the form

$$G = E_1 \cdot F_1^\omega + \cdots + E_n \cdot F_n^\omega \text{ for } n > 0$$

where E_i, F_i are regular expressions over A with $\varepsilon \notin \mathcal{L}(F_i)$.

Semantics

For $\mathcal{L} \subseteq A^*$, let $\mathcal{L}^\omega = \{w_1 w_2 w_3 \cdots \mid \forall\, i \geq 1,\ w_i \in \mathcal{L}\}$

For $G = E_1.F_1^\omega + \cdots + E_n.F_n^\omega$, $\mathcal{L}(G) \subseteq A^\omega$ is given by

$$\mathcal{L}(G) = \mathcal{L}(E_1) \cdot \mathcal{L}(F_1)^\omega \cup \cdots \cup \mathcal{L}(E_n) \cdot \mathcal{L}(F_n)^\omega .$$

ω-regular languages: examples

A language $\mathcal{L}$ is ω-regular if $\mathcal{L} = \mathcal{L}(G)$ for some ω-regular expression G.

ω-regular languages: examples

A language $\mathcal{L}$ is ω-regular if $\mathcal{L} = \mathcal{L}(G)$ for some ω-regular expression G.

Examples for $A = \{a, b\}$

- Words with infinitely many a's: $(b^* \, a)^\omega$

ω-regular languages: examples

A language $\mathcal{L}$ is ω-regular if $\mathcal{L} = \mathcal{L}(G)$ for some ω-regular expression G.

Examples for $A = \{a, b\}$

- Words with infinitely many a's: $(b^* \, a)^\omega$
- Words with finitely many a's: $(a \mid b)^* \, b^\omega$

ω-regular languages: examples

A language $\mathcal{L}$ is ω-regular if $\mathcal{L} = \mathcal{L}(G)$ for some ω-regular expression G.

Examples for $A = \{a, b\}$

- Words with infinitely many a's: $(b^* \, a)^\omega$
- Words with finitely many a's: $(a \mid b)^* \, b^\omega$
- Empty language: $\varnothing^\omega$ (OK because $\varnothing$ is a valid regular expression)

ω-regular languages: examples

A language $\mathcal{L}$ is ω-regular if $\mathcal{L} = \mathcal{L}(G)$ for some ω-regular expression G.

Examples for $A = \{a, b\}$

- Words with infinitely many a's: $(b^* \, a)^\omega$
- Words with finitely many a's: $(a \mid b)^* \, b^\omega$
- Empty language: $\varnothing^\omega$ (OK because $\varnothing$ is a valid regular expression)

Properties of ω-regular languages

They are *closed* under union, intersection and complementation.

ω-regular languages: non-example

Not all languages on infinite words are ω-regular.

E.g., $\mathcal{L} = \{$words on $A = \{a, b\}$ such that a appears infinitely often with increasingly many b's between occurrences of $a\}$ is not.

Link with LTL?

We know that every LTL formula φ describes a language of infinite words $\mathrm{Words}(\varphi) \subseteq (2^{P})^{\omega}$.

Link with LTL?

We know that every LTL formula φ describes a language of infinite words $\mathrm{Words}(\varphi) \subseteq (2^P)^\omega$.

$\implies$ **For every LTL formula φ, $\mathrm{Words}(\varphi)$ is an ω-regular language.**

Link with LTL?

We know that every LTL formula φ describes a language of infinite words $\mathrm{Words}(\varphi) \subseteq (2^P)^\omega$.

$\Longrightarrow$ **For every LTL formula φ, $\mathrm{Words}(\varphi)$ is an ω-regular language.**

The converse is false!

There are ω-regular languages that cannot be expressed in LTL, e.g.

$$\mathcal{L} = \left\{ a_0 a_1 a_2 \cdots \in (2^{\{a\}})^\omega \;\middle|\; \forall\, i \geq 0,\ a \in a_{2i} \right\},$$

all words where a must hold in all even positions.

- ω-regular expression $G = (\{a\} \, (\{a\} \mid \varnothing))^\omega$.
- Intuitively, LTL *cannot count modulo k* (e.g., words with "a" every k steps)

Büchi automata: definition

Automata describing languages of **infinite** words

- ω-regular languages.

Definition: non-deterministic Büchi automaton (NBA)

Tuple $\mathcal{A} = (Q, A, \delta, I, F)$ with

- Q a finite set of states,
- A a finite alphabet,
- $\delta \colon Q \times A \to 2^Q$ a transition function,
- $I \subseteq Q$ a set of initial states,
- $F \subseteq Q$ a set of accepting (or final) states.

Büchi automata: definition

Automata describing languages of **infinite** words

- ω-regular languages.

Definition: non-deterministic Büchi automaton (NBA)

Tuple $\mathcal{A} = (Q, A, \delta, I, F)$ with

- Q a finite set of states,
- A a finite alphabet,
- $\delta \colon Q \times A \to 2^Q$ a transition function,
- $I \subseteq Q$ a set of initial states,
- $F \subseteq Q$ a set of accepting (or final) states.

Same as before?

University of Antwerp

Büchi automata: acceptance condition

$\Longrightarrow$ The automaton is identical, but **the acceptance condition is different!**

Büchi automata: acceptance condition

$\Longrightarrow$ The automaton is identical, but **the acceptance condition is different!**

Run

A run on an *infinite* word $w = a_0 a_1 \dots \in A^\omega$ is a sequence $q_0 q_1 \dots$ of states such that $q_0 \in I$ and for all $i \geq 0$, $q_{i+1} \in \delta(q_i, a_i)$.

Büchi automata: acceptance condition

$\Longrightarrow$ The automaton is identical, but **the acceptance condition is different!**

Run

A run on an *infinite* word $w = a_0 a_1 \cdots \in A^\omega$ is a sequence $q_0 q_1 \ldots$ of states such that $q_0 \in I$ and for all $i \geq 0$, $q_{i+1} \in \delta(q_i, a_i)$.

Accepting run

A run is accepting if $q_i \in F$ for **infinitely many** indices $i \in \mathbb{N}$.

Büchi automata: acceptance condition

$\Longrightarrow$ The automaton is identical, but **the acceptance condition is different!**

Run

A run on an *infinite* word $w = a_0 a_1 \cdots \in A^\omega$ is a sequence $q_0 q_1 \ldots$ of states such that $q_0 \in I$ and for all $i \geq 0$, $q_{i+1} \in \delta(q_i, a_i)$.

Accepting run

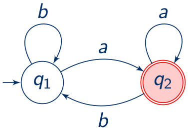
A run is accepting if $q_i \in F$ for **infinitely many** indices $i \in \mathbb{N}$.

Accepted language of $\mathcal{A}$

$$\mathcal{L}(\mathcal{A}) = \{w \in A^\omega \mid \text{ there is an accepting run for } w \text{ in } \mathcal{A}\}.$$

Büchi automata: examples

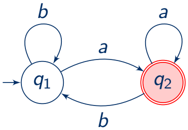
- Words with infinitely many a's: $(b^* \, a)^\omega$.



Deterministic Büchi automaton (DBA)

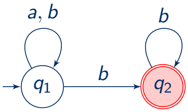
Büchi automata: examples

- Words with infinitely many a's: $(b^* \, a)^\omega$.



Deterministic Büchi automaton (DBA)

- Words with finitely many a's: $(a \mid b)^* \, b^\omega$

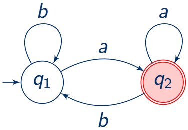


Non-deterministic Büchi automaton (NBA)

Is there an equivalent DBA?

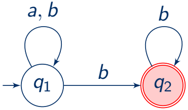
Büchi automata: examples

- Words with infinitely many a's: $(b^* a)^\omega$.



Deterministic Büchi automaton (DBA)

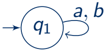
- Words with finitely many a's: $(a \mid b)^* b^\omega$



Non-deterministic Büchi automaton (NBA)

Is there an equivalent DBA?

- Empty language: $\varnothing^\omega$



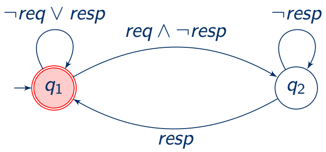
Büchi automata: as specifications

Liveness property: "once a request is provided, eventually a response shall occur"

- $\{req, resp\} \subseteq P$ for the TS
- NBA $\mathcal{A}$ uses alphabet 2^P

  $\hookrightarrow$ Succinct representation of multiple transitions using propositional logic. E.g., for $P = \{a, b\}$,

$$q \xrightarrow{a \vee b} q' \text{ stands for } q \xrightarrow{\{a\}} q',\ q \xrightarrow{\{b\}} q',\ \text{and } q \xrightarrow{\{a,b\}} q'.$$



Büchi automata vs. ω-regular languages

Theorem

A language is ω-regular if and only if it is recognized by an NBA.

Büchi automata vs. ω-regular languages

Theorem

A language is ω-regular if and only if it is recognized by an NBA.

$\Longrightarrow$ **For any ω-regular property, we can build a corresponding NBA.**

$\Longrightarrow$ **For any NBA $\mathcal{A}$, the language $\mathcal{L}(\mathcal{A})$ is ω-regular.**

From ω-regular expressions to NBAs

Reminder

An ω-regular expression G over A has the form

$$G = E_1 \cdot F_1^\omega + \cdots + E_n \cdot F_n^\omega \text{ for } n > 0$$

where E_i, F_i are regular expressions over A with $\varepsilon \notin \mathcal{L}(F_i)$.

Construction scheme

Use operators on NBAs mimicking operators on ω-regular expressions:

- union of NBAs ($E_1 \cdot F_1^\omega + E_2 \cdot F_2^\omega$)
- ω-operator for NFA (F^ω)
- concatenation of an NFA and an NBA ($E \cdot F^\omega$)

From expressions to a union of NBAs

Goal

Mimic $E_1 \cdot F_1^\omega + E_2 \cdot F_2^\omega$.

Let $\mathcal{A}^1 = (Q^1, A, \delta^1, I^1, F^1)$ and $\mathcal{A}^2 = (Q^2, A, \delta^2, I^2, F^2)$ be two NBAs over the same alphabet with disjoint state spaces.

From expressions to a union of NBAs

Goal

Mimic $E_1 \cdot F_1^\omega + E_2 \cdot F_2^\omega$.

Let $\mathcal{A}^1 = (Q^1, A, \delta^1, I^1, F^1)$ and $\mathcal{A}^2 = (Q^2, A, \delta^2, I^2, F^2)$ be two NBAs over the same alphabet with disjoint state spaces.

Union

$\mathcal{A}^1 + \mathcal{A}^2 = (Q^1 \cup Q^2, A, \delta, I^1 \cup I^2, F^1 \cup F^2)$ with $\delta(q, a) = \delta^i(q, a)$ if $q \in Q^i$.

A word is accepted by $\mathcal{A}^1 + \mathcal{A}^2$ iff it is accepted by (at least) one of the automata.

From expressions to a union of NBAs

Goal

Mimic $E_1 \cdot F_1^\omega + E_2 \cdot F_2^\omega$.

Let $\mathcal{A}^1 = (Q^1, A, \delta^1, I^1, F^1)$ and $\mathcal{A}^2 = (Q^2, A, \delta^2, I^2, F^2)$ be two NBAs over the same alphabet with disjoint state spaces.

Union

$\mathcal{A}^1 + \mathcal{A}^2 = (Q^1 \cup Q^2, A, \delta, I^1 \cup I^2, F^1 \cup F^2)$ with $\delta(q, a) = \delta^i(q, a)$ if $q \in Q^i$.

A word is accepted by $\mathcal{A}^1 + \mathcal{A}^2$ iff it is accepted by (at least) one of the automata.

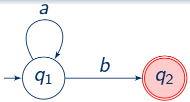
$$\Longrightarrow \mathcal{L}(\mathcal{A}^1 + \mathcal{A}^2) = \mathcal{L}(\mathcal{A}^1) \cup \mathcal{L}(\mathcal{A}^2)$$

ω-operator (1/2)

Goal

Mimic F^ω (from an automaton recognizing F)

Example:

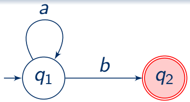


ω-operator (1/2)

Goal

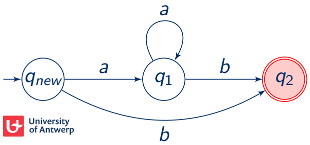
Mimic F^ω (from an automaton recognizing F)

Example:

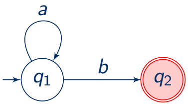


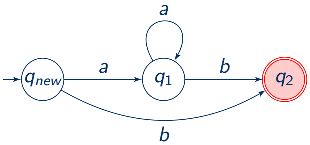
Step 1. If states in I have incoming transitions or $I \cap F \neq \emptyset$:

- Introduce new initial state $q_{new} \notin F$
- Add $q_{new} \xrightarrow{a} q$ iff $q_0 \xrightarrow{a} q$ for some $q_0 \in I$
- Keep all other transitions of $\mathcal{A}$
- New $I = \{q_{new}\}$



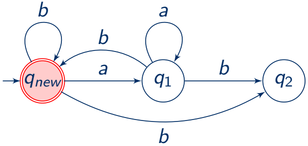
ω-operator (2/2)



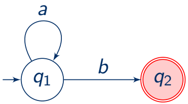


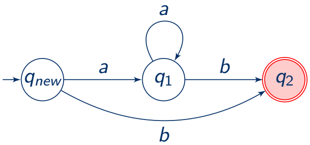
Step 2. Build the NBA $\mathcal{A}'$ as follows.

- If $q \xrightarrow{a} q' \in F$, then add $q \xrightarrow{a} q_0$ for all $q_0 \in I$
- Keep all other transitions of $\mathcal{A}$
- $I' = I$ and $F' = I$



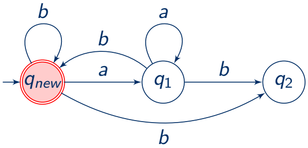
ω-operator (2/2)





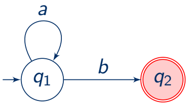
Step 2. Build the NBA $\mathcal{A}'$ as follows.

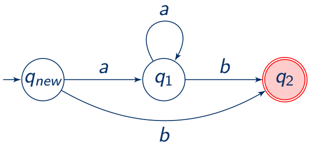
- If $q \xrightarrow{a} q' \in F$, then add $q \xrightarrow{a} q_0$ for all $q_0 \in I$
- Keep all other transitions of $\mathcal{A}$
- $I' = I$ and $F' = I$



$\hookrightarrow$ *In practice, state q_2 is now useless and can be removed.*

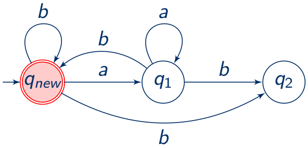
ω-operator (2/2)





Step 2. Build the NBA $\mathcal{A}'$ as follows.

- If $q \xrightarrow{a} q' \in F$, then add $q \xrightarrow{a} q_0$ for all $q_0 \in I$
- Keep all other transitions of $\mathcal{A}$
- $I' = I$ and $F' = I$



$\hookrightarrow$ *In practice, state q_2 is now useless and can be removed.*

$\Longrightarrow \mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})^\omega$**, i.e., this NBA recognizes** $(a^*b)^\omega$**.**

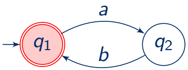
NFA-NBA concatenation (1/2)

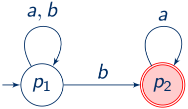
Goal

Mimic $E \cdot F^\omega$ (from automata for E and F)

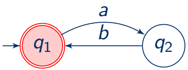
Let $\mathcal{A}^1 = (Q^1, A, \delta^1, I^1, F^1)$ be an NFA and $\mathcal{A}^2 = (Q^2, A, \delta^2, I^2, F^2)$ be an NBA, both over the same alphabet and with disjoint state spaces.

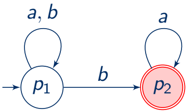
Example: NFA $\mathcal{A}^1$ with $\mathcal{L}(\mathcal{A}^1) = (a\,b)^*$ and NBA $\mathcal{A}^2$ with $\mathcal{L}(\mathcal{A}^2) = (a \mid b)^* b\, a^\omega$.





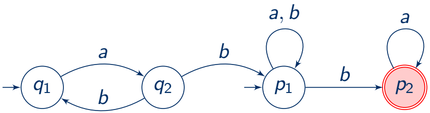
NFA-NBA concatenation (2/2)



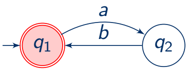


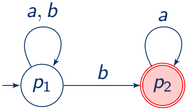
Construction of NBA. $\mathcal{A} = (Q = Q^1 \cup Q^2, A, \delta, I, F = F^2)$

- $I = \begin{cases} I^1 & \text{if } I^1 \cap F^1 = \varnothing \\ I^1 \cup I^2 & \text{otherwise} \end{cases}$
- $\delta(q,a) = \begin{cases} \delta^1(q,a) & \text{if } q \in Q^1 \text{ and } \delta^1(q,a) \cap F^1 = \varnothing \\ \delta^1(q,a) \cup I^2 & \text{if } q \in Q^1 \text{ and } \delta^1(q,a) \cap F^1 \neq \varnothing \\ \delta^2(q,a) & \text{if } q \in Q^2 \end{cases}$



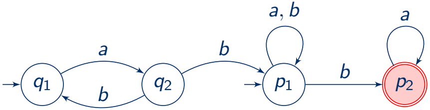
NFA-NBA concatenation (2/2)





Construction of NBA. $\mathcal{A} = (Q = Q^1 \cup Q^2, A, \delta, I, F = F^2)$

- $I = \begin{cases} I^1 & \text{if } I^1 \cap F^1 = \varnothing \\ I^1 \cup I^2 & \text{otherwise} \end{cases}$
- $\delta(q, a) = \begin{cases} \delta^1(q, a) & \text{if } q \in Q^1 \text{ and } \delta^1(q, a) \cap F^1 = \varnothing \\ \delta^1(q, a) \cup I^2 & \text{if } q \in Q^1 \text{ and } \delta^1(q, a) \cap F^1 \neq \varnothing \\ \delta^2(q, a) & \text{if } q \in Q^2 \end{cases}$



$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}^1) \cdot \mathcal{L}(\mathcal{A}^2)$, **this NBA recognizes** $(a\, b)^*(a \mid b)^*\, b\, a^\omega$

University of Antwerp

Büchi automata: checking non-emptiness

Criterion for non-emptiness

Let $\mathcal{A}$ be an NBA. Then,

$$\mathcal{L}(\mathcal{A}) \neq \varnothing$$
$$\Updownarrow$$
$$\exists\, q_0 \in I, \exists\, q \in F, \exists\, w \in A^*, \exists\, v \in A^+, q \in \delta^*(q_0, w) \wedge q \in \delta^*(q, v),$$

i.e., **there is an accepting state on a reachable cycle**

Büchi automata: checking non-emptiness

Criterion for non-emptiness

Let $\mathcal{A}$ be an NBA. Then,

$$\mathcal{L}(\mathcal{A}) \neq \varnothing$$
$$\Updownarrow$$
$$\exists\, q_0 \in I,\ \exists\, q \in F,\ \exists\, w \in A^*,\ \exists\, v \in A^+,\ q \in \delta^*(q_0, w) \wedge q \in \delta^*(q, v),$$

i.e., **there is an accepting state on a reachable cycle**

$\Longrightarrow$ Can be checked in *linear time* by computing reachable strongly connected components (SCCs)

$\Longrightarrow$ **Important tool for LTL model checking**

NBAs vs. DBAs

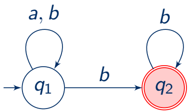
Recall that **DFAs are as expressive as NFAs**. What about DBAs w.r.t. NBAs?

NBAs vs. DBAs

Recall that **DFAs are as expressive as NFAs**. What about DBAs w.r.t. NBAs?

NBAs are strictly more expressive than DBAs

There exists no DBA $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}((a \mid b)^* a^\omega)$.



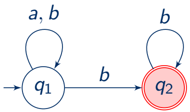
Words with finitely many a's

NBAs vs. DBAs

Recall that **DFAs are as expressive as NFAs**. What about DBAs w.r.t. NBAs?

NBAs are strictly more expressive than DBAs

There exists no DBA $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}((a \mid b)^* a^\omega)$.



Words with finitely many a's

$\Longrightarrow$ Intuition: by contradiction, if a DBA would exist, then we show that it would accept some words with infinitely many a's by exploiting determinism to construct corresponding accepting runs

Generalized Büchi automata

- NBAs describe ω-regular languages.

Generalized Büchi automata

- NBAs describe ω-regular languages.
- Several **equally expressive** variants exist, with different acceptance conditions: Muller, Rabin, Streett, parity and **generalized Büchi** automata (**GNBAs**)

$\Longrightarrow$ **Will help us for LTL model checking**

Generalized Büchi automata: definition

Definition: non-det. generalized Büchi automaton (GNBA)

Tuple $\mathcal{G} = (Q, A, \delta, I, \mathcal{F})$ with

- Q a finite set of states,
- A a finite alphabet,
- $\delta \colon Q \times A \to 2^Q$ a transition function,
- $I \subseteq Q$ a set of initial states,
- $\mathcal{F} = \{F_1, \ldots, F_k\} \subseteq 2^Q$ ($k \geq 0$ and $\forall 0 \leq i \leq k$, $F_i \subseteq Q$)

Intuition: a GNBA requires to visits **each set** F_i infinitely often

University of Antwerp

GNBAs: acceptance condition

Accepting run

A run $q_0 q_1 \ldots$ is accepting if for all $F \in \mathcal{F}$, $q_i \in F$ for infinitely many indices $i \in \mathbb{N}$.

GNBAs: acceptance condition

Accepting run

A run $q_0 q_1 \ldots$ is accepting if for all $F \in \mathcal{F}$, $q_i \in F$ for infinitely many indices $i \in \mathbb{N}$.

Accepted language of $\mathcal{G}$

$$\mathcal{L}(\mathcal{G}) = \{w \in A^\omega \mid \text{ there is an accepting run for } w \text{ in } \mathcal{G}\}.$$

GNBAs: acceptance condition

Accepting run

A run $q_0 q_1 \dots$ is accepting if for all $F \in \mathcal{F}$, $q_i \in F$ for infinitely many indices $i \in \mathbb{N}$.

Accepted language of $\mathcal{G}$

$$\mathcal{L}(\mathcal{G}) = \{w \in A^\omega \mid \text{ there is an accepting run for } w \text{ in } \mathcal{G}\}.$$

For $k = 0$, all runs are accepting. For $k = 1$, $\mathcal{G}$ is a simple NBA.

GNBAs: acceptance condition

Accepting run

A run $q_0 q_1 \ldots$ is accepting if for all $F \in \mathcal{F}$, $q_i \in F$ for infinitely many indices $i \in \mathbb{N}$.

Accepted language of $\mathcal{G}$

$$\mathcal{L}(\mathcal{G}) = \{w \in A^\omega \mid \text{ there is an accepting run for } w \text{ in } \mathcal{G}\}.$$

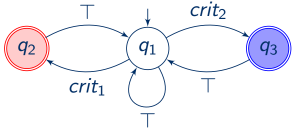
For $k = 0$, all runs are accepting. For $k = 1$, $\mathcal{G}$ is a simple NBA.

Observe the difference between $F = \varnothing$ for an NBA (i.e., no run is accepting) and $\mathcal{F} = \varnothing$ for a GNBA (i.e., all runs are accepting). In fact, $\mathcal{F} = \varnothing$ is equivalent to having $F = \{Q\}$.

GNBAs: as specifications

Liveness property: "both processes are infinitely often in their critical section"

- $\{crit_1, crit_2\} \subseteq P$ for the TS.



- $\mathcal{F} = \{\{q_2\}, \{q_3\}\}$. **Both must be visited infinitely often!**

GNBAs vs. NBAs

From a GNBA to an NBA

For any GNBA $\mathcal{G}$, there exists a language-equivalent NBA.

GNBAs vs. NBAs

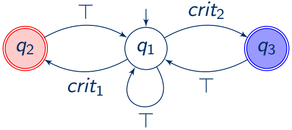
From a GNBA to an NBA

For any GNBA $\mathcal{G}$, there exists a language-equivalent NBA.

1. Make k copies of Q arranged in k levels.
2. At level $i \in \{1, \ldots, k\}$, keep all transitions leaving states $q \notin F_i$ at the same level.
3. At level $i \in \{1, \ldots, k\}$, redirect transitions leaving states $q \in F_i$ to level $i + 1$ (level $k + 1$ to level 1).
4. $I' = \{\langle q_0, 1\rangle \mid q_0 \in I\}$, i.e., initial states in level 1; and $F' = \{\langle q, 1\rangle \mid q \in F_1\}$, i.e., final states in level 1.

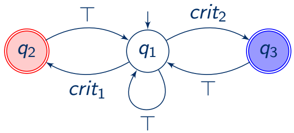
$\Longrightarrow$ **F' can only be visited infinitely often if the accepting states (F_i) at every level i are visited infinitely often.**

GNBAs vs. NBAs: example

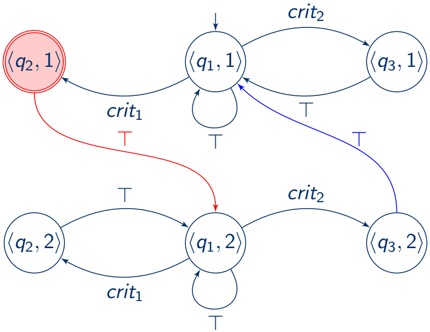


$\Longrightarrow$ **Blackboard illustration**

GNBAs vs. NBAs: example



$\Longrightarrow$ **Blackboard illustration**



Summary and conclusions

Parity automata

- Max even priority version on board...
- Deterministic parity automata recognize all ω-regular languages.

Omega automata

- Büchi automata recognize all ω-regular languages
- Deterministic automata are less powerful than non-deterministic ones
- Automata-based tools will be a recurring topic in the sequel!