Büchi Automata

Vansh, Pinakin, Aayush, Bhanu

March 2, 2024

Indian Institute of Science, Bangalore

Regular Expressions and ω-Regular Expressions

Non-Deterministic Büchi Automata

Characterizing Büchi Recognisable Languages

Logic of Sequences and Büchi Automata

Regular Expressions and ω-Regular Expressions

Regular Expressions over Σ

Definition

A regular expression is a sequence of characters that define a search pattern.

$$\alpha := \phi \mid A \mid \alpha_1 + \alpha_2 \mid \alpha_1 \cdot \alpha_2 \mid \alpha^*$$

Where A is an alphabet and ϕ is the empty string.

Some semantics used throughout the presentation:

- $\alpha_1 + \alpha_2$ is the union of α_1 and α_2
- $\alpha_1 \cdot \alpha_2$ is the concatenation of α_1 and α_2
- α^* is the Kleene closure of α
- $\mathcal{L}(\phi) = \phi$, $\mathcal{L}(A) = \{A\}$ and $\mathcal{L}(\epsilon) = \{\epsilon\}$

Regular Expressions over Σ

Definition

A regular expression is a sequence of characters that define a search pattern.

$$\alpha := \phi \mid A \mid \alpha_1 + \alpha_2 \mid \alpha_1 \cdot \alpha_2 \mid \alpha^*$$

Where A is an alphabet and ϕ is the empty string.

Some semantics used throughout the presentation:

- $\alpha_1 + \alpha_2$ is the union of α_1 and α_2
- $\alpha_1 \cdot \alpha_2$ is the concatenation of α_1 and α_2
- α^* is the Kleene closure of α
- $\mathcal{L}(\phi) = \phi$, $\mathcal{L}(A) = \{A\}$ and $\mathcal{L}(\epsilon) = \{\epsilon\}$

Regular Expressions over Σ

Definition

A regular expression is a sequence of characters that define a search pattern.

$$\alpha := \phi \mid A \mid \alpha_1 + \alpha_2 \mid \alpha_1 \cdot \alpha_2 \mid \alpha^*$$

Where A is an alphabet and ϕ is the empty string.

Some semantics used throughout the presentation:

- $\alpha_1 + \alpha_2$ is the union of α_1 and α_2
- $\alpha_1 \cdot \alpha_2$ is the concatenation of α_1 and α_2
- α^* is the Kleene closure of α
- $\mathcal{L}(\phi) = \phi$, $\mathcal{L}(A) = \{A\}$ and $\mathcal{L}(\epsilon) = \{\epsilon\}$

Regular Expressions over Σ

Definition

A regular expression is a sequence of characters that define a search pattern.

$$\alpha := \phi \mid A \mid \alpha_1 + \alpha_2 \mid \alpha_1 \cdot \alpha_2 \mid \alpha^*$$

Where A is an alphabet and ϕ is the empty string.

Some semantics used throughout the presentation:

- $\alpha_1 + \alpha_2$ is the union of α_1 and α_2
- $\alpha_1 \cdot \alpha_2$ is the concatenation of α_1 and α_2
- α^* is the Kleene closure of α
- $\mathcal{L}(\phi) = \phi$, $\mathcal{L}(A) = \{A\}$ and $\mathcal{L}(\epsilon) = \{\epsilon\}$

ω-Regular Expression

Definition

ω-regular expression = regex + ω-operator α^ω

Where we choose any string generated by α and repeat it infinitely many times, denoted by ω-operator.

$$\text{for } \mathcal{L} \subseteq \Sigma^* :$$

$$\mathcal{L}^\omega = \{w_1 w_2 w_3 \cdots : w_i \in \mathcal{L}\ \forall i \geq 1\}$$

$$\mathcal{L}(\omega) \subseteq \Sigma^\omega \text{ if } \epsilon \notin \mathcal{L}$$

- Kleene Star: 'finite repetition'
- ω-operator: 'infinite repetition'

ω-Regular Expression

Definition

ω-regular expression = regex + ω-operator α^ω

Where we choose any string generated by α and repeat it infinitely many times, denoted by ω-operator.

$$\text{for } \mathcal{L} \subseteq \Sigma^* :$$

$$\mathcal{L}^\omega = \{w_1 w_2 w_3 \cdots : w_i \in \mathcal{L}\ \forall i \geq 1\}$$

$$\mathcal{L}(\omega) \subseteq \Sigma^\omega \text{ if } \epsilon \notin \mathcal{L}$$

- Kleene Star: 'finite repetition'
- ω-operator: 'infinite repetition'

ω-Regular Expression

Definition

ω-regular expression = regex + ω-operator α^ω

Where we choose any string generated by α and repeat it infinitely many times, denoted by ω-operator.

$$\text{for } \mathcal{L} \subseteq \Sigma^* :$$

$$\mathcal{L}^\omega = \{w_1 w_2 w_3 \cdots : w_i \in \mathcal{L}\ \forall i \geq 1\}$$

$$\mathcal{L}(\omega) \subseteq \Sigma^\omega \text{ if } \epsilon \notin \mathcal{L}$$

- Kleene Star: 'finite repetition'
- ω-operator: 'infinite repetition'

Syntax and semantics of ω-regular expressions

Definition

Syntax of ω-regular expressions over alphabet Σ:

$$\gamma = \alpha_1 \cdot \beta_1^\omega + \cdots + \alpha_n \cdot \beta_n^\omega$$

α_i, β_i are regular expressions over Σ such that $\epsilon \notin \mathcal{L}(\beta_i)$

Syntax and semantics of ω-regular expressions

Definition

Syntax of ω-regular expressions over alphabet Σ:

$$\gamma = \alpha_1 \cdot \beta_1^\omega + \cdots + \alpha_n \cdot \beta_n^\omega$$

α_i, β_i are regular expressions over Σ such that $\epsilon \notin \mathcal{L}(\beta_i)$

The language generated by γ is:

$$\mathcal{L}_\omega(\gamma) = \bigcup_{1 \leq i \leq n} \mathcal{L}(\alpha_i)\mathcal{L}(\beta_i)^\omega \subseteq \Sigma^\omega$$

Syntax and semantics of ω-regular expressions

Definition

Syntax of ω-regular expressions over alphabet Σ:

$$\gamma = \alpha_1 \cdot \beta_1^\omega + \cdots + \alpha_n \cdot \beta_n^\omega$$

α_i, β_i are regular expressions over Σ such that $\epsilon \notin \mathcal{L}(\beta_i)$

The language generated by γ is:

$$\mathcal{L}_\omega(\gamma) = \bigcup_{1 \leq i \leq n} \mathcal{L}(\alpha_i)\mathcal{L}(\beta_i)^\omega \subseteq \Sigma^\omega$$

Example

The language of $(A^*B)^\omega$ = set of all infinite words containing infinitely many B's

ω Regular Language

ω Regular Language

A language $\mathcal{L} \subseteq \Sigma^\omega$ is called ω-regular iff there exists an ω regular expression such that $\mathcal{L} = \mathcal{L}_\omega(\gamma)$

ω - regex example

- Alphabet $\Sigma = \{a, b\}$ such that the set of all infinite words contain finitely many a's.

$$(a + b)^* b^\omega$$

- Set of all infinite words where each a is followed immediately by letter b.

$$(b^*.a.b)^*.b^\omega + (b^*.a.b)^\omega$$

ω - regex example

- Alphabet $\Sigma = \{a, b\}$ such that the set of all infinite words contain finitely many a's.

$$(a + b)^* b^\omega$$

- Set of all infinite words where each a is followed immediately by letter b.

$$(b^*.a.b)^*.b^\omega + (b^*.a.b)^\omega$$

Non-Deterministic Büchi Automata

Büchi Automata (NBA)

A Büchi automaton is a method of defining a set of ω words over a finite alphabet Σ

Definition

A (non-deterministic) Büchi automaton is a tuple $A = (Q, \Sigma, \delta, Q_0, F)$ where:

- Q is a finite set of states
- Σ is a finite alphabet
- $Q_0 \subseteq Q$ set of initial states
- $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation.
- F is the set of final states.

Büchi Automata (NBA)

Acceptance

Let $A_0 A_1 A_2 \cdots \in \Sigma^\omega$ be a run of a word, and π be the sequence of states visited:

$$\pi = q_0 q_1 q_2 \cdots \text{ where } q_0 \in Q_0$$

$$q_{i+1} \in \delta(q_i, A_i) \text{ for } i \geq 0$$

We say that run π is accepted if there exist infinitely many i's such that $q_i \in F$.

The language recognised by A, written $\mathcal{L}(A)$, is the set of ω-words accepted by A.

Büchi Automata (NBA)

Acceptance

Let $A_0 A_1 A_2 \cdots \in \Sigma^\omega$ be a run of a word, and π be the sequence of states visited:

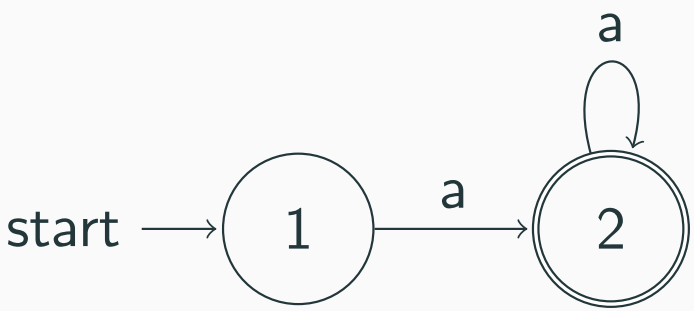
$$\pi = q_0 q_1 q_2 \cdots \text{ where } q_0 \in Q_0$$

$$q_{i+1} \in \delta(q_i, A_i) \text{ for } i \geq 0$$

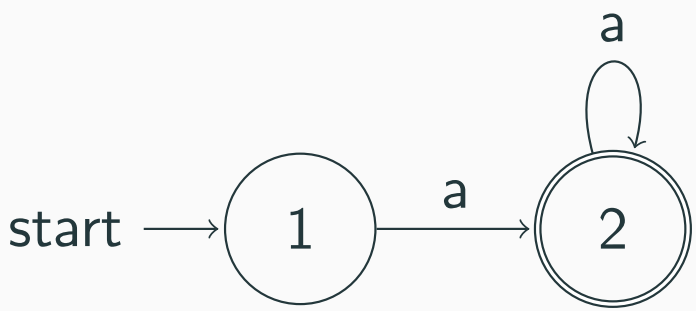
We say that run π is accepted if there exist infinitely many i's such that $q_i \in F$.

The language recognised by A, written $\mathcal{L}(A)$, is the set of ω-words accepted by A.

NBA Example



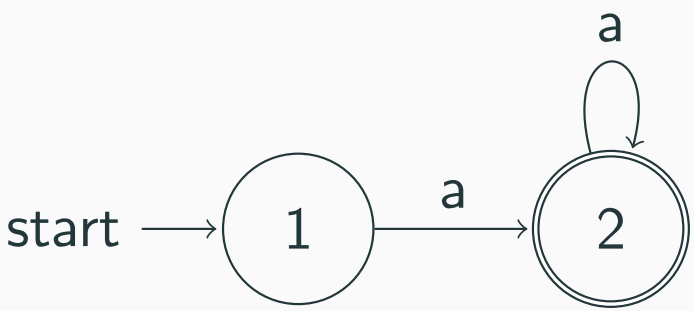
NBA Example

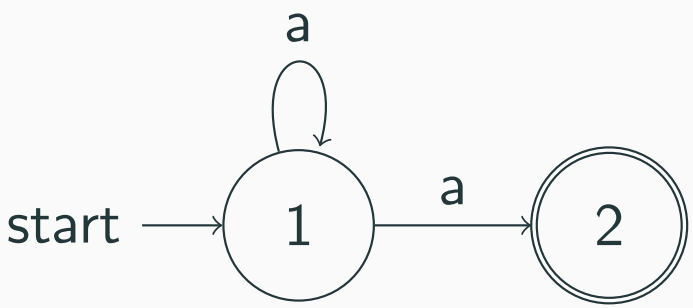


This automaton accepts the language of all ω-words over $\{a\}$ that contain infinitely many a's.

$$\mathcal{L}_\omega(A_1) = \{a\}^\omega$$

NBA Example

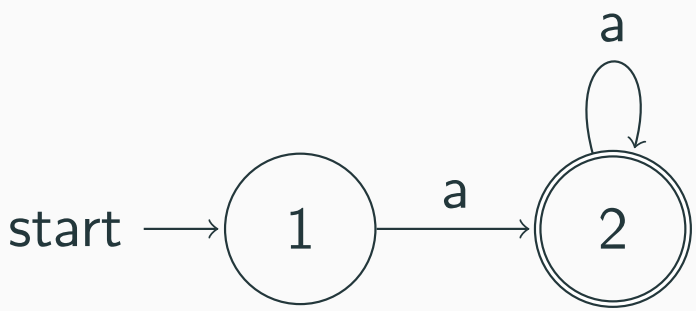




This automaton accepts the language of all ω-words over $\{a\}$ that contain infinitely many a's.

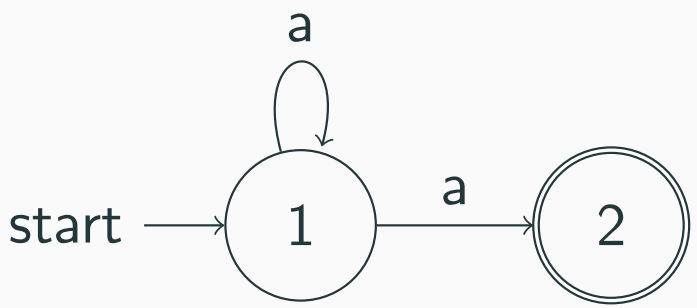
$$\mathcal{L}_\omega(A_1) = \{a\}^\omega$$

NBA Example



This automaton accepts the language of all ω-words over $\{a\}$ that contain infinitely many a's.

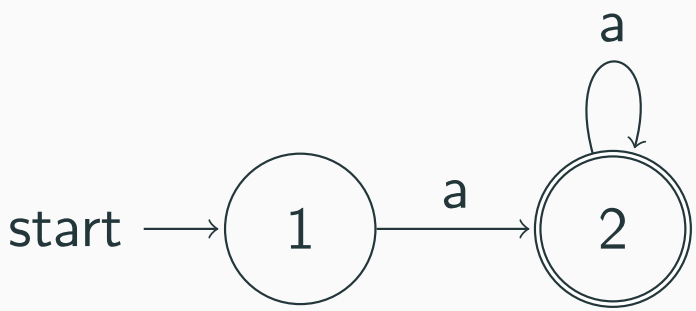
$$\mathcal{L}_\omega(A_1) = \{a\}^\omega$$



This automaton does not accept any ω-words.

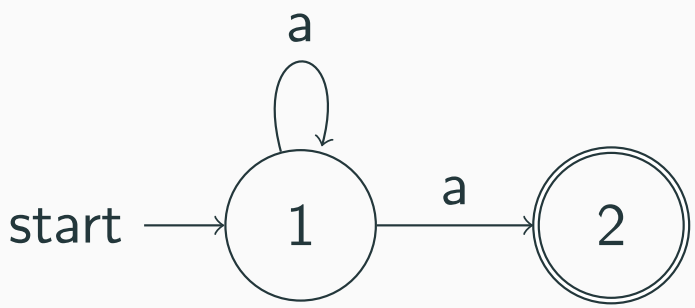
$$\mathcal{L}_\omega(A_2) = \phi$$

NBA Example



This automaton accepts the language of all ω-words over $\{a\}$ that contain infinitely many a's.

$$\mathcal{L}_\omega(A_1) = \{a\}^\omega$$

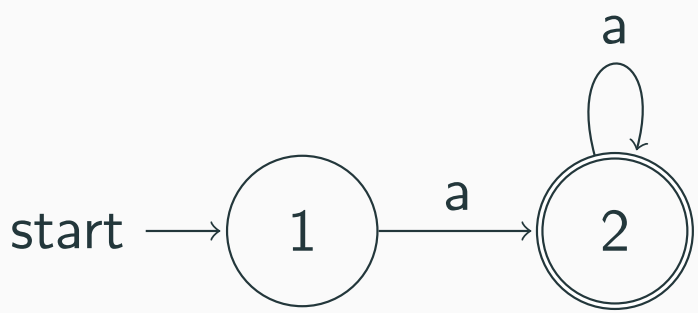


This automaton does not accept any ω-words.

$$\mathcal{L}_\omega(A_2) = \phi$$

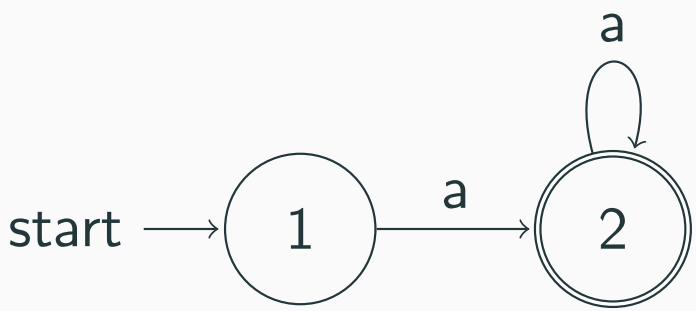
In NBA "perspective", both languages are different.

NBA Example (Language accepted by NFA)

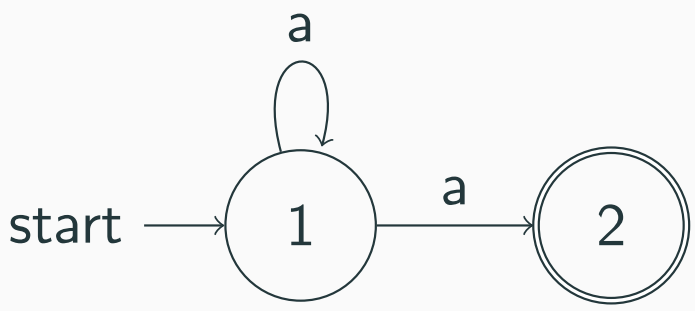


The language accepted by the automaton A_1,
$\mathcal{L}(A_1) = \{a^{n+1} \mid n \in N\}$

NBA Example (Language accepted by NFA)

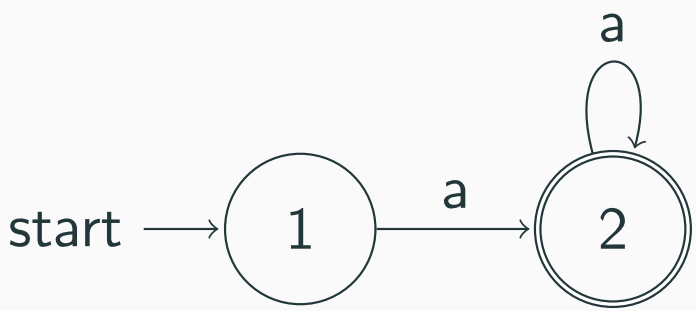


The language accepted by the automaton A_1,
$\mathcal{L}(A_1) = \{a^{n+1} \mid n \in N\}$

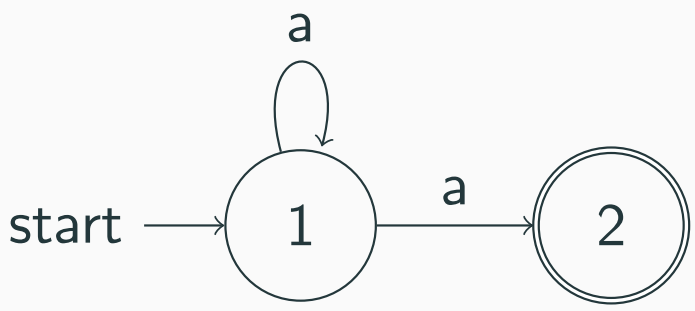


The language accepted by the automaton A_2,
$\mathcal{L}(A_2) = \{a^{n+1} \mid n \in N\}$

NBA Example (Language accepted by NFA)



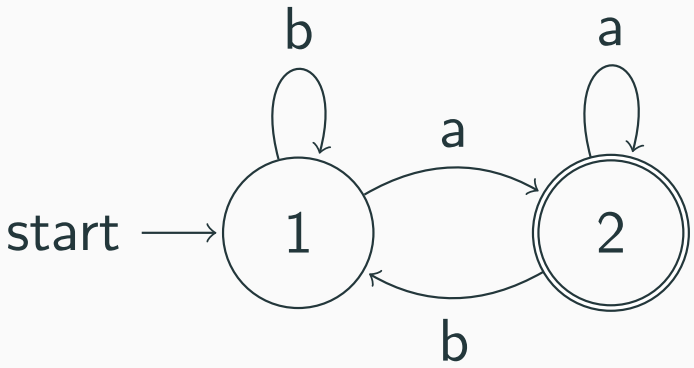
The language accepted by the automaton A_1,
$\mathcal{L}(A_1) = \{a^{n+1} \mid n \in N\}$



The language accepted by the automaton A_2,
$\mathcal{L}(A_2) = \{a^{n+1} \mid n \in N\}$

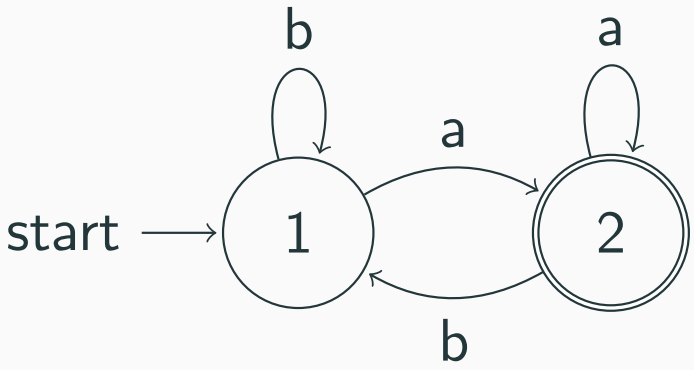
We can see that, the languages accepted by the automata A_1 and A_2 are same in NFA "perspective".
Thus both automaton are equivalent in NFA "perspective", but different in NBA "perspective".

Example of NBA over $\Sigma = \{a, b\}$



Accepted language:
Set of all infinite words that contain infinitely many a's
$\mathcal{L}(M) = (a^* b)^\omega$

Example of NBA over $\Sigma = \{a, b\}$



Accepted language:

Set of all infinite words that contain infinitely many a's

$\mathcal{L}(M) = (a^* b)^\omega$

Characterizing Büchi Recognisable Languages

Converting NBA to ω-regular expression

Theorem

A language is Buchi recognisable iff it is ω regular.

Proof.

($\implies$) Converting NBA to ω regular language.

Let A be an NBA $(Q, \Sigma, \delta, Q_0, F)$ and $q, p \in Q$.

Let $A_{q,p}$ be the NFA $(Q, \Sigma, \delta, q, \{p\})$. Then:

$$\mathcal{L}\omega(A) = \bigcup_{q \in Q_0} \bigcup_{p \in F} \mathcal{L}(A_{q,p})(\mathcal{L}(A_{p,p}) \backslash \{\epsilon\})^{\omega}$$

is ω regular as $\mathcal{L}(A_{q,p})$ and $\mathcal{L}(A_{p,p}) \backslash \{\epsilon\}$ are regular by definition. $\square$

Converting NBA to ω-regular expression

Theorem

A language is Buchi recognisable iff it is ω regular.

Proof.

($\implies$) Converting NBA to ω regular language.

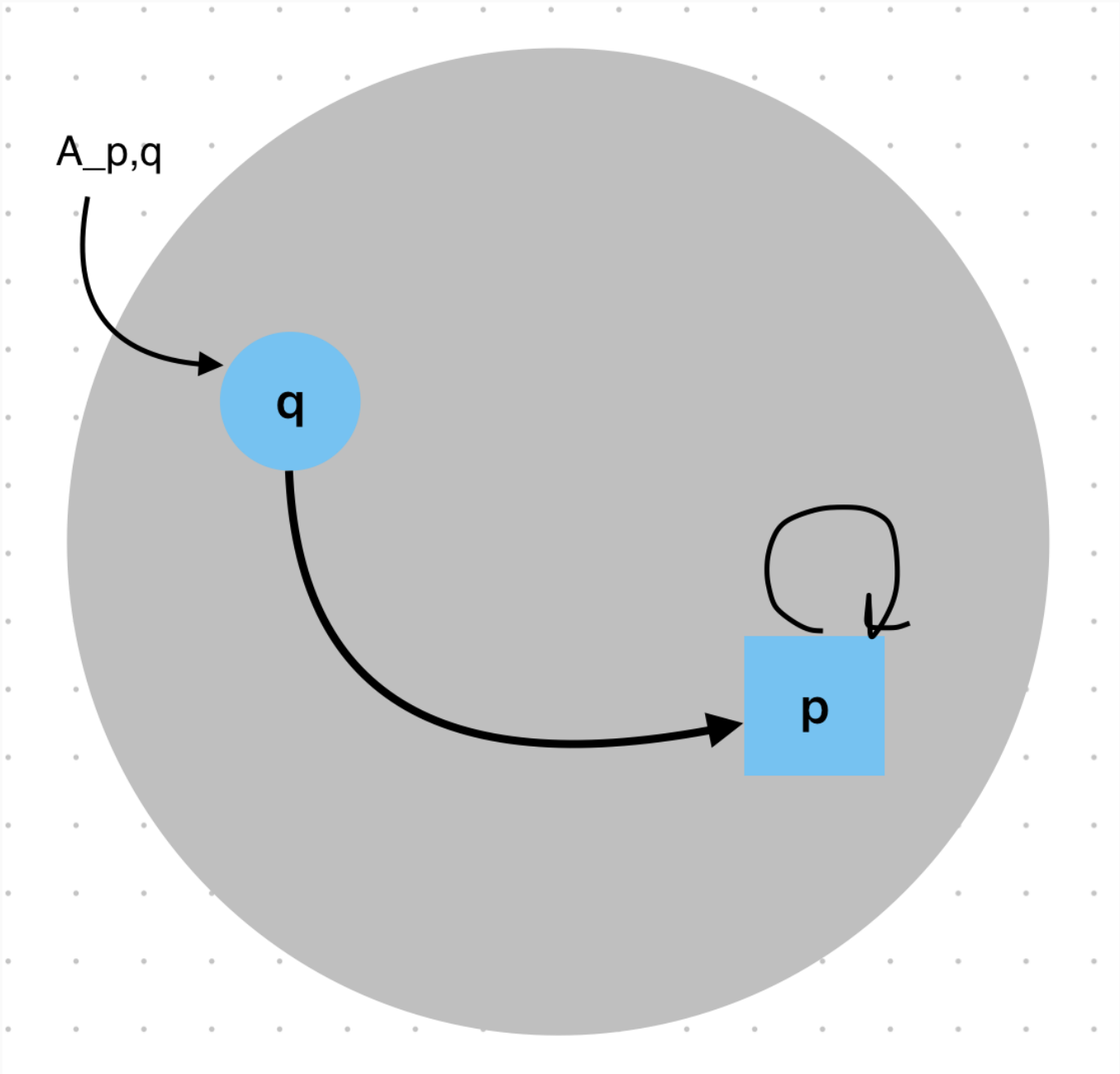
Let A be an NBA $(Q, \Sigma, \delta, Q_0, F)$ and $q, p \in Q$.

Let $A_{q,p}$ be the NFA $(Q, \Sigma, \delta, q, \{p\})$. Then:

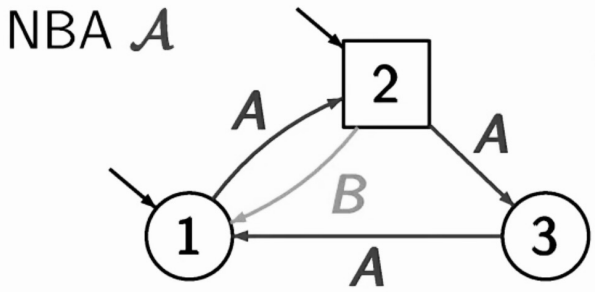
$$\mathcal{L}\omega(A) = \bigcup_{q \in Q_0} \bigcup_{p \in F} \mathcal{L}(A_{q,p})(\mathcal{L}(A_{p,p}) \backslash \{\epsilon\})^{\omega}$$

is ω regular as $\mathcal{L}(A_{q,p})$ and $\mathcal{L}(A_{p,p}) \backslash \{\epsilon\}$ are regular by definition. $\square$

Converting NBA to ω-regular expression



Example: Converting NBA to ω-regular expression

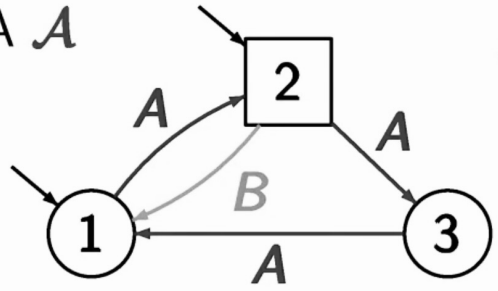


$$\mathcal{L}_\omega(\mathcal{A}) = L_{12}(L'_{22})^\omega \cup L_{22}(L'_{22})^\omega$$

$$L_{12} = \mathcal{L}(\mathcal{A}_{12})$$
$$L_{22} = \mathcal{L}(\mathcal{A}_{22})$$
$$L'_{22} = L_{22} \setminus \{\varepsilon\}$$

Example: Converting NBA to ω-regular expression

NBA $\mathcal{A}$

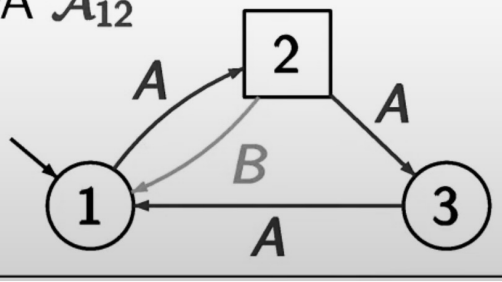


$$\mathcal{L}_\omega(\mathcal{A}) = L_{12}(L'_{22})^\omega \cup L_{22}(L'_{22})^\omega$$

$$L_{12} = \mathcal{L}(\mathcal{A}_{12})$$
$$L_{22} = \mathcal{L}(\mathcal{A}_{22})$$
$$L'_{22} = L_{22} \setminus \{\varepsilon\}$$

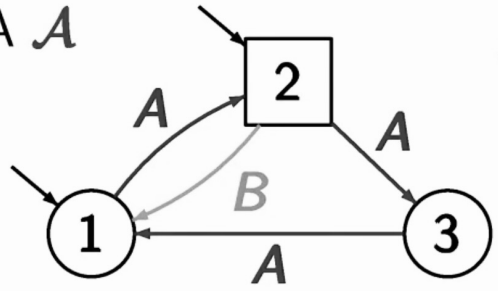
$$L_{12} \mathrel{\widehat{=}} A.(B.A + A.A.A)^*$$

NFA $\mathcal{A}_{12}$



Example: Converting NBA to ω-regular expression

NBA $\mathcal{A}$



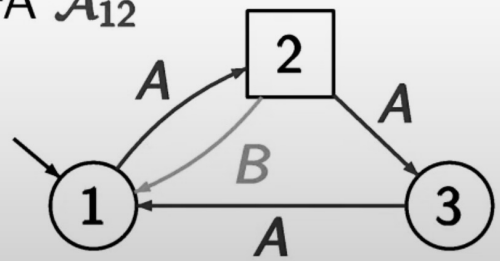
$$\mathcal{L}_\omega(\mathcal{A}) = L_{12}(L'_{22})^\omega \cup L_{22}(L'_{22})^\omega$$

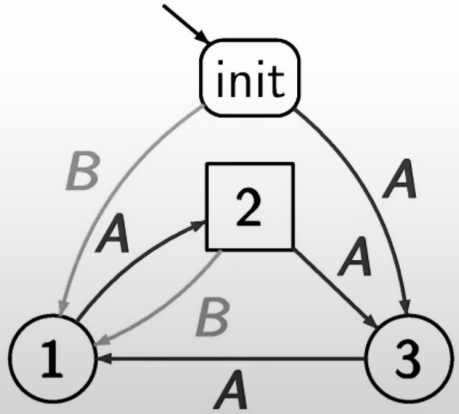
$$L_{12} = \mathcal{L}(\mathcal{A}_{12})$$
$$L_{22} = \mathcal{L}(\mathcal{A}_{22})$$
$$L'_{22} = L_{22} \setminus \{\varepsilon\}$$

$L_{12} \mathrel{\widehat{=}} A.(B.A + A.A.A)^*$

$L'_{22} \mathrel{\widehat{=}} (B.A + A.A.A)^+$

NFA $\mathcal{A}_{12}$





From ω-regular expression to NBA

Proof.

($\Longleftarrow$) Converting ω regular expression to NBA.

Let γ be an ω regular expression over Σ.

$$\gamma = \alpha_1 \cdot \beta_1^\omega + \cdots + \alpha_n \cdot \beta_n^\omega$$

there exists an NBA A, with $\mathcal{L}\omega(A) = \mathcal{L}\omega(\gamma)$

Proof: consider NFA $\mathcal{A}_i$ for α_i and $\mathcal{B}_i$ for β_i

- construct NBA $\mathcal{B}_i^\omega$ for β_i^ω
- construct NBA $\mathcal{C}_i = \mathcal{A}_i \cdot \mathcal{B}_i^\omega$ for $\alpha_i \cdot \beta_i^\omega$
- construct NBA for $\bigcup_{1 \leq i \leq n} \mathcal{L}_\omega(\mathcal{C}_i)$

□

We proceed the proof by showing that the set of Buchi recognisable languages satisy the following closure properties:

Closure under Finite Union

Finite Union

Let $A_1 = (Q_1, \Sigma, \delta_1, Q_{01}, F_1)$ and $A_2 = (Q_2, \Sigma, \delta_2, Q_{02}, F_2)$ be two NBA's. Then the language $\mathcal{L}(A_1) \cup \mathcal{L}(A_2)$ is Buchi recognisable.

Proof.

Just take the union of the two automata !

Closure under Finite Union

Finite Union

Let $A_1 = (Q_1, \Sigma, \delta_1, Q_{01}, F_1)$ and $A_2 = (Q_2, \Sigma, \delta_2, Q_{02}, F_2)$ be two NBA's. Then the language $\mathcal{L}(A_1) \cup \mathcal{L}(A_2)$ is Buchi recognisable.

Proof.

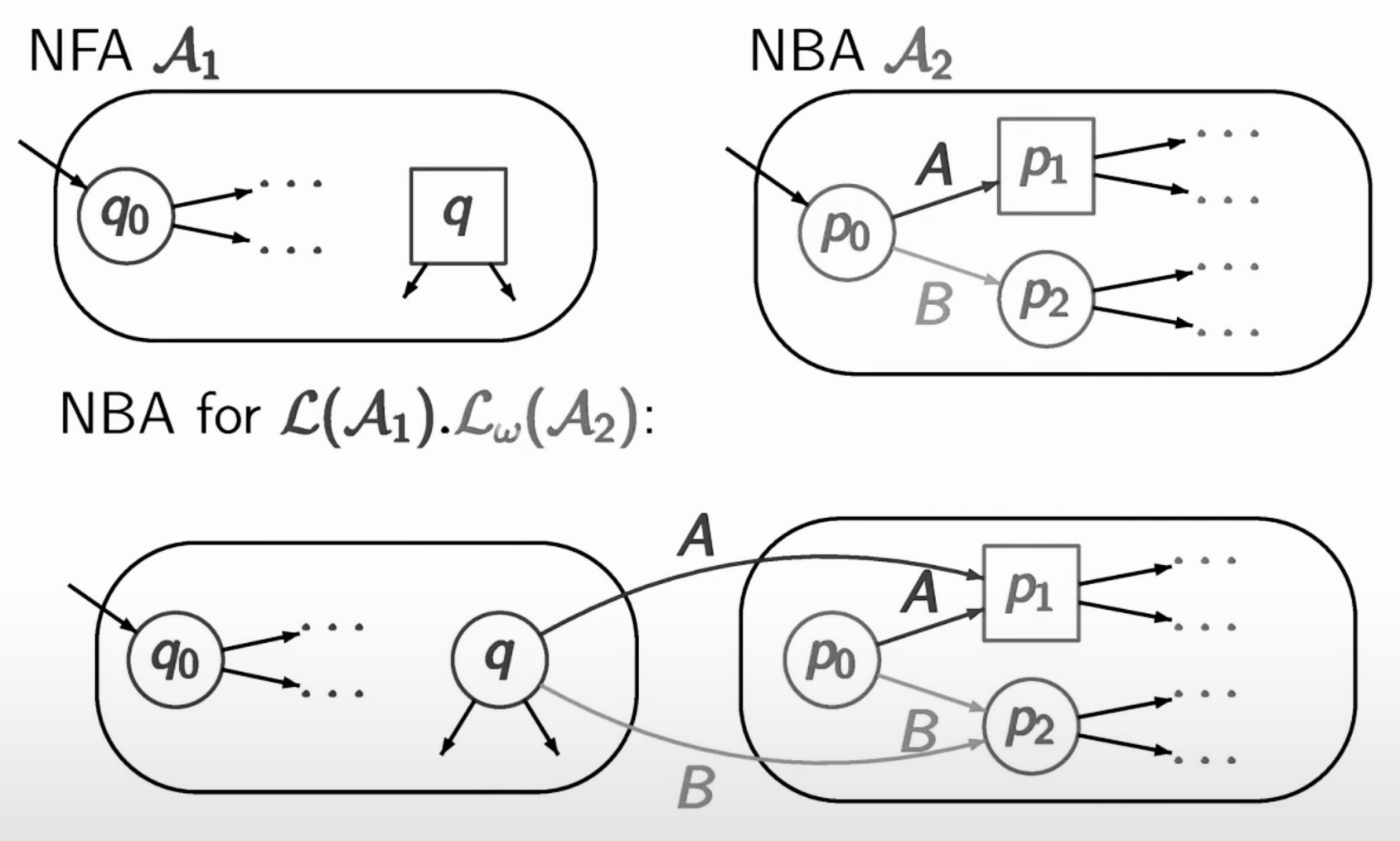
Just take the union of the two automata !

Concatenation of NFA and NBA

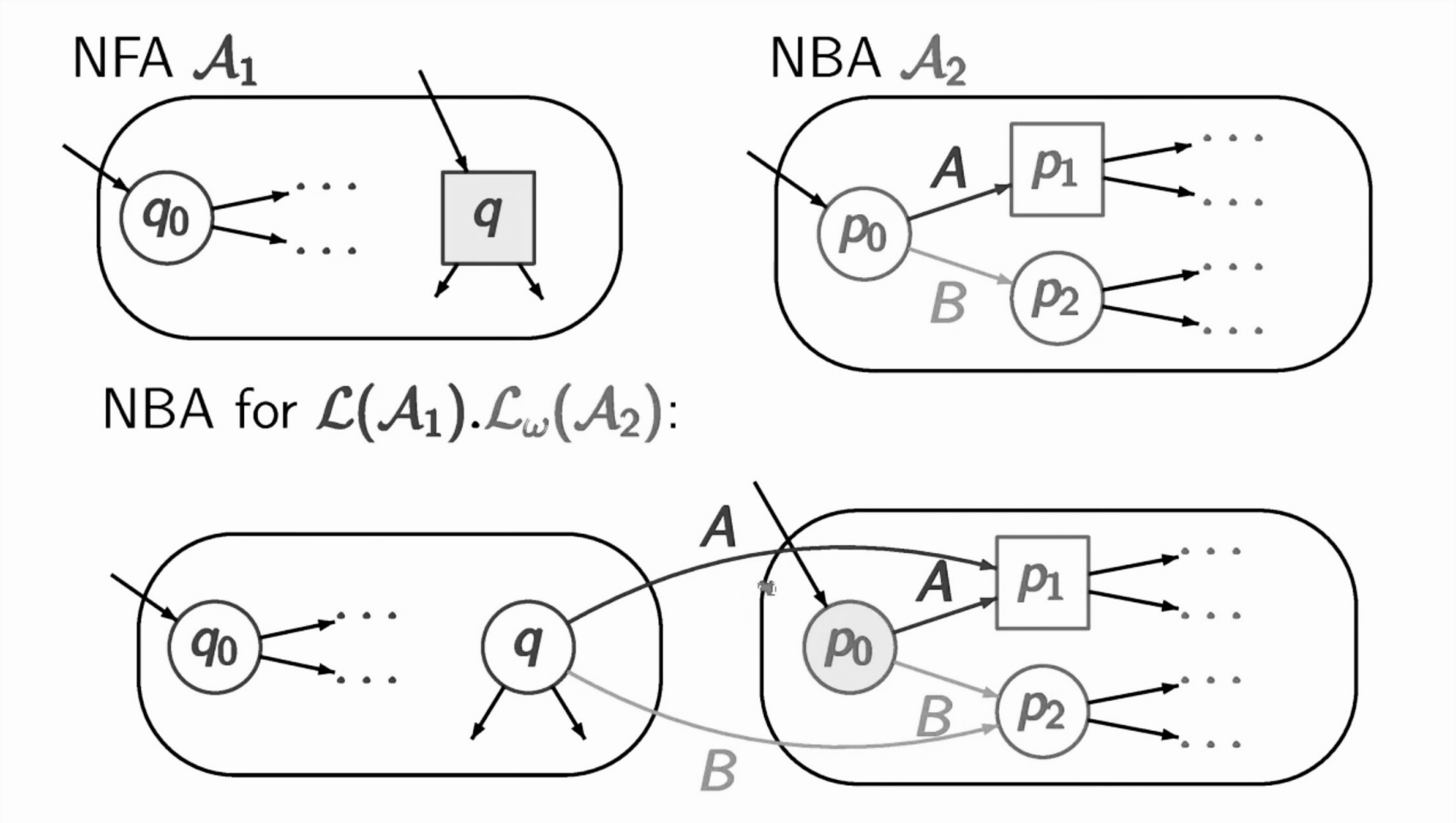
Concatenation

If $L(A_1)$ is regular and $L_\omega(A_2)$ is Buchi recognisable then $L(A_1)L_\omega(A_2)$ is Buchi recognisable.

Concatenation of NFA and NBA

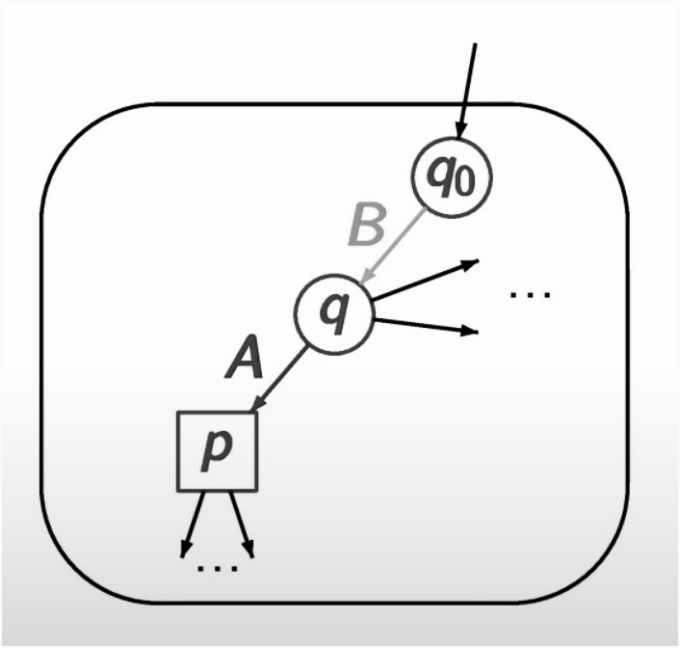


Concatenation of NFA and NBA

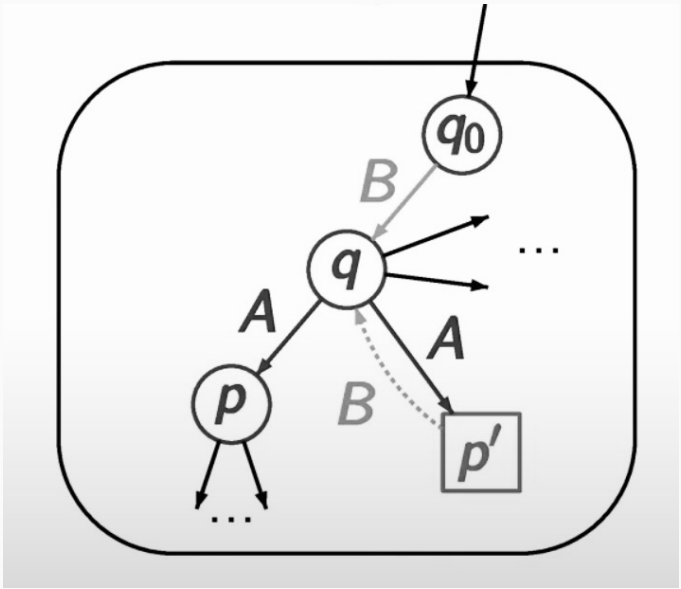


Construct NBA for NFA

NFA A for a language $\Longrightarrow$ $L \subseteq \Sigma^+$



NFA $\mathcal{B}$ for L such that all final states are terminal.



$$\mathcal{L}(A)^\omega = \mathcal{L}_w(\mathcal{B}^\omega)$$

Construct NBA for NFA

The final states (e.g. p) must be terminal states. If not make them.

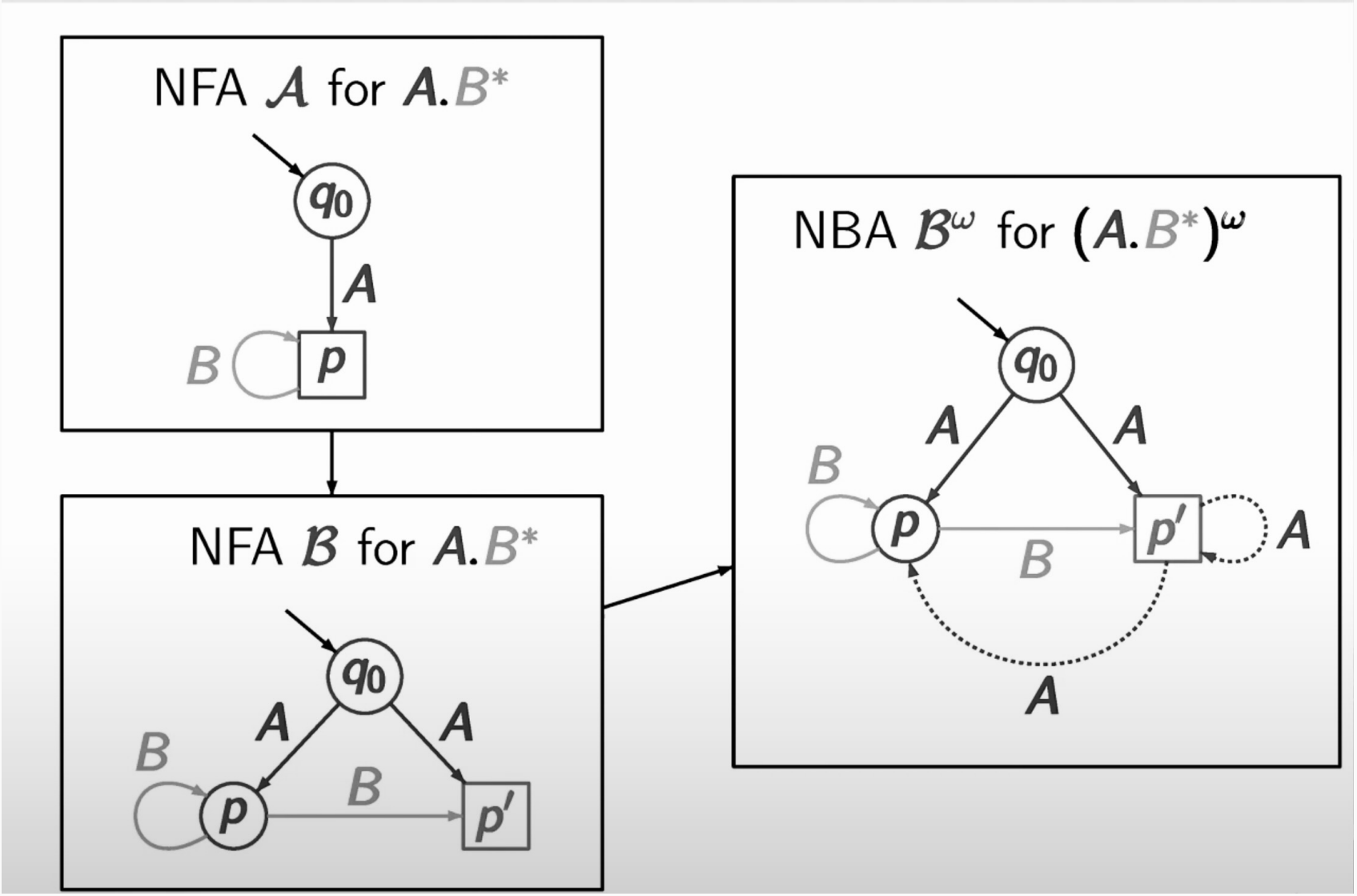
WHY ? Because, if not, then there might be other possible paths introduced when we make the final state an "initial state" in the NBA.

Construct NBA for NFA

The final states (e.g. p) must be terminal states. If not make them.

WHY ? Because, if not, then there might be other possible paths introduced when we make the final state an "initial state" in the NBA.

Example: Construct NBA for NFA



Logic of Sequences and Büchi Automata

Logic of Sequences

Buchi's original motivation for studying automata on infinite inputs was to solve a decision problem from logic. He discovered a deep and beautiful connection between ω-regular languages and sets of models of formulas in certain logics.

S1S

The logic that Buchi considered was the monadic second-order theory of one successor, abbreviated as S1S. This logic is interpreted over the set N0 of natural numbers. In general, second-order logic permits quantification over relations and functions, unlike first-order logic, which permits quantification over just individual elements.

- Second-order means that we allow quantifications over relations.
- Monadic means that quantifications is restricted to monadic relations, namely sets.

S1S (contd.)

Fix a logical structure $(\omega, s, \in)$

- s is the successor function $x \to x + 1$
- $\in$ is the standard memebrship relation between elements and sets.

Variables

- First-order variables (x, y, z, etc.) range over natural numbers.
- Second-order variables (X, Y, Z, etc.) range over sets of natural numbers.

Terms

- First-order variables are terms.
- If t is a term, then so is s(t)

S1S (contd.)

Formulas

- Atomic formulas are of the form $t \in X$ where t is a term and X is a second-order variable.
- S1S formulas are built up from atomic formulas using standard boolean connectives, with $\forall$ and $\exists$ quantifications over first-order and second-order variables.

Satisfiability and Buchi's Theorem

An S1S formula is said to be satisfiable if we can choose M such that $M \vDash \phi$

Buchi's Theorem

Buchi showed how to associate an ω-regular language $L_{(\phi)}$ with each S1S formula ϕ, such that every word in L_ϕ represents an interpretation for the free variables in ϕ under which the formula ϕ evaluates to true. Moreover, every interpretation that makes ϕ true is represented by some word in L_ϕ. Thus, ϕ is satisfiable iff there is some interpretation that makes it true iff L_ϕ is non-empty.

Example

Let ϕ be a invariant with invariant condition, $a \vee \neg b$. Then the ω- regular language associated with the invariant condition is $(\phi + \{a\} + \{a, b\})^\omega$ where ϕ is the empty set.

Motivation

- Büchi automata's primary application is in formal verification of hardware and software systems.
- They are used to verify the correctness of systems that are infinite-state, such as communication protocols, distributed systems, and embedded systems.
- By infinite state, we mean the system runs indefinitely.
- Büchi automata are used to verify the correctness of such systems by checking if the system satisfies a given property.
- The property is usually specified as a formula in a temporal logic, such as LTL or CTL.

Motivation

- Büchi automata's primary application is in formal verification of hardware and software systems.
- They are used to verify the correctness of systems that are infinite-state, such as communication protocols, distributed systems, and embedded systems.
- By infinite state, we mean the system runs indefinitely.
- Büchi automata are used to verify the correctness of such systems by checking if the system satisfies a given property.
- The property is usually specified as a formula in a temporal logic, such as LTL or CTL.

Motivation

- Büchi automata's primary application is in formal verification of hardware and software systems.
- They are used to verify the correctness of systems that are infinite-state, such as communication protocols, distributed systems, and embedded systems.
- By infinite state, we mean the system runs indefinitely.
- Büchi automata are used to verify the correctness of such systems by checking if the system satisfies a given property.
- The property is usually specified as a formula in a temporal logic, such as LTL or CTL.

Motivation

- Büchi automata's primary application is in formal verification of hardware and software systems.
- They are used to verify the correctness of systems that are infinite-state, such as communication protocols, distributed systems, and embedded systems.
- By infinite state, we mean the system runs indefinitely.
- Büchi automata are used to verify the correctness of such systems by checking if the system satisfies a given property.
- The property is usually specified as a formula in a temporal logic, such as LTL or CTL.

Motivation

- Büchi automata's primary application is in formal verification of hardware and software systems.
- They are used to verify the correctness of systems that are infinite-state, such as communication protocols, distributed systems, and embedded systems.
- By infinite state, we mean the system runs indefinitely.
- Büchi automata are used to verify the correctness of such systems by checking if the system satisfies a given property.
- The property is usually specified as a formula in a temporal logic, such as LTL or CTL.

The end

References:

- https://www.youtube.com/watch?v=0Vnth-1X-OE&t=2290s&ab_channel=songsong
- https://www.cmi.ac.in/~madhavan/papers/pdf/iisc2011-buchi.pdf
- https://www.cs.ox.ac.uk/people/luke.ong/personal/talks/s1s.pdf
- https://medium.com/@sakshimahesh20/introduction-to-buchi-automata-c9e598935ab5

The end

Thank You!