41080 Theory of Computing Science
Week 2 Tutorial Class

Chuanqi Zhang

Centre for Quantum Software and Information
University of Technology Sydney

15th August, 2024

Outline

- **Review**: languages and operations on them
- **Keynote**: DFAs and their relation with languages
- **Tutorial**: how to do the product construction of two DFAs

What is a language?

- Σ: an alphabet set;
- Σ^n: the set of all length-n strings over Σ;
- Σ^*: the set of ALL strings over Σ.

Definition (Language)

L is a language if $L \subseteq \Sigma^*$ for some Σ.

Example (Language)

Let $\Sigma = \{0, 1\}$ and $L = \{w \in \{0, 1\}^* \mid w \text{ starts with 0 and ends with 1}\}$.

Let's have a try!

What is a language?

- Σ: an alphabet set;
- Σ^n: the set of all length-n strings over Σ;
- Σ^*: the set of ALL strings over Σ.

Definition (Language)

L is a language if $L \subseteq \Sigma^*$ for some Σ.

Example (Language)

Let $\Sigma = \{0, 1\}$ and $L = \{w \in \{0,1\}^* \mid w \text{ starts with 0 and ends with 1}\}$.

Let's have a try!

What is a language?

- Σ: an alphabet set;
- Σ^n: the set of all length-n strings over Σ;
- Σ^*: the set of ALL strings over Σ.

Definition (Language)

L is a language if $L \subseteq \Sigma^*$ for some Σ.

Example (Language)

Let $\Sigma = \{0, 1\}$ and $L = \{w \in \{0, 1\}^* \mid w \text{ starts with 0 and ends with 1}\}$.

Let's have a try!

What is a language?

- Σ: an alphabet set;
- Σ^n: the set of all length-n strings over Σ;
- Σ^*: the set of ALL strings over Σ. Formally, $\Sigma^* = \cup_{n\in\mathbb{N}}\Sigma^n$.

Definition (Language)

L is a language if $L \subseteq \Sigma^*$ for some Σ.

Example (Language)

Let $\Sigma = \{0,1\}$ and $L = \{w \in \{0,1\}^* \mid w \text{ starts with 0 and ends with 1}\}$.

Let's have a try!

What is a language?

- Σ: an alphabet set;
- Σ^n: the set of all length-n strings over Σ;
- Σ^*: the set of ALL strings over Σ. Formally, $\Sigma^* = \cup_{n\in\mathbb{N}}\Sigma^n$.

Definition (Language)

L is a language if $L \subseteq \Sigma^*$ for some Σ.

Example (Language)

Let $\Sigma = \{0, 1\}$ and $L = \{w \in \{0, 1\}^* \mid w$ starts with 0 and ends with 1$\}$.

Let's have a try!

What is a language?

- Σ: an alphabet set;
- Σ^n: the set of all length-n strings over Σ;
- Σ^*: the set of ALL strings over Σ. Formally, $\Sigma^* = \cup_{n\in\mathbb{N}}\Sigma^n$.

Definition (Language)

L is a language if $L \subseteq \Sigma^*$ for some Σ.

Example (Language)

Let $\Sigma = \{0,1\}$ and $L = \{w \in \{0,1\}^* \mid w \text{ starts with 0 and ends with 1}\}$.

Let's have a try! Is the string '01' in language L?

What is a language?

- Σ: an alphabet set;
- Σ^n: the set of all length-n strings over Σ;
- Σ^*: the set of ALL strings over Σ. Formally, $\Sigma^* = \cup_{n\in\mathbb{N}}\Sigma^n$.

Definition (Language)

L is a language if $L \subseteq \Sigma^*$ for some Σ.

Example (Language)

Let $\Sigma = \{0, 1\}$ and $L = \{w \in \{0,1\}^* \mid w \text{ starts with 0 and ends with 1}\}$.

Let's have a try! Is the string '010' in language L?

UTS:QSI
CENTRE FOR QUANTUM SOFTWARE AND INFORMATION

What is a language?

- Σ: an alphabet set;
- Σ^n: the set of all length-n strings over Σ;
- Σ^*: the set of ALL strings over Σ. Formally, $\Sigma^* = \cup_{n\in\mathbb{N}}\Sigma^n$.

Definition (Language)

L is a language if $L \subseteq \Sigma^*$ for some Σ.

Example (Language)

Let $\Sigma = \{0, 1\}$ and $L = \{w \in \{0,1\}^* \mid w \text{ starts with } 0 \text{ and ends with } 1\}$.

Let's have a try! Is the string '101' in language L?

Basic operations on languages

Given two languages $L_1, L_2 \subseteq \Sigma^*$, we can make the following operations:

- Union: $L_1 \cup L_2 = \{w \in \Sigma^* : w \in L_1 \text{ or } w \in L_2\}$.
- Intersection: $L_1 \cap L_2 = \{w \in \Sigma^* : w \in L_1 \text{ and } w \in L_2\}$.
- Complement: $\neg L_1 = \{w \in \Sigma^* : w \notin L_1\}$.
- Reverse: $L_1^R = \{a_k \dots a_1 \in \Sigma^* : a_1 \dots a_k \in L_1 \text{ for each } a_i \in \Sigma\}$.
- Concatenation: $L_1 \circ L_2 = \{w_1 w_2 \in \Sigma^* : w_1 \in L_1 \text{ and } w_2 \in L_2\}$.
- Kleene star: $L_1^* = \{w_1 \dots w_k \in \Sigma^* : w_i \in L_1\} \cup \{\varepsilon\}$.

  the empty string

- A better way to understand the Kleene star operation:

$$L_1^n = \underbrace{L_1 \circ L_1 \circ \cdots \circ L_1}_{n \text{ times}} \quad \longrightarrow \quad L_1^* = \bigcup_{n \in \mathbb{N}} L_1^n$$

UTS:QSI
CENTRE FOR QUANTUM SOFTWARE AND INFORMATION

Basic operations on languages

Given two languages $L_1, L_2 \subseteq \Sigma^*$, we can make the following operations:

- Union: $L_1 \cup L_2 = \{w \in \Sigma^* : w \in L_1 \text{ or } w \in L_2\}$.
- Intersection: $L_1 \cap L_2 = \{w \in \Sigma^* : w \in L_1 \text{ and } w \in L_2\}$.
- Complement: $\neg L_1 = \{w \in \Sigma^* : w \notin L_1\}$.
- Reverse: $L_1^R = \{a_k \dots a_1 \in \Sigma^* : a_1 \dots a_k \in L_1 \text{ for each } a_i \in \Sigma\}$.
- Concatenation: $L_1 \circ L_2 = \{w_1 w_2 \in \Sigma^* : w_1 \in L_1 \text{ and } w_2 \in L_2\}$.
- Kleene star: $L_1^* = \{w_1 \dots w_k \in \Sigma^* : w_i \in L_1\} \cup \{\underset{\substack{\downarrow \\ \text{the empty string}}}{\varepsilon}\}$.

- A better way to understand the Kleene star operation:

$$L_1^n = \underbrace{L_1 \circ L_1 \circ \cdots \circ L_1}_{n \text{ times}} \quad \longrightarrow \quad L_1^* = \bigcup_{n \in \mathbb{N}} L_1^n$$

Basic operations on languages

Given two languages $L_1, L_2 \subseteq \Sigma^*$, we can make the following operations:

- Union: $L_1 \cup L_2 = \{w \in \Sigma^* : w \in L_1 \text{ or } w \in L_2\}$.
- Intersection: $L_1 \cap L_2 = \{w \in \Sigma^* : w \in L_1 \text{ and } w \in L_2\}$.
- Complement: $\neg L_1 = \{w \in \Sigma^* : w \notin L_1\}$.
- Reverse: $L_1^R = \{a_k \ldots a_1 \in \Sigma^* : a_1 \ldots a_k \in L_1 \text{ for each } a_i \in \Sigma\}$.
- Concatenation: $L_1 \circ L_2 = \{w_1 w_2 \in \Sigma^* : w_1 \in L_1 \text{ and } w_2 \in L_2\}$.
- Kleene star: $L_1^* = \{w_1 \ldots w_k \in \Sigma^* : w_i \in L_1\} \cup \{\underbrace{\varepsilon}_{\text{the empty string}}\}$.

- A better way to understand the Kleene star operation:

$$L_1^n = \underbrace{L_1 \circ L_1 \circ \cdots \circ L_1}_{n \text{ times}} \quad \longrightarrow \quad L_1^* = \bigcup_{n \in \mathbb{N}} L_1^n$$

Basic operations on languages

Given two languages $L_1, L_2 \subseteq \Sigma^*$, we can make the following operations:

- Union: $L_1 \cup L_2 = \{w \in \Sigma^* : w \in L_1 \text{ or } w \in L_2\}$.
- Intersection: $L_1 \cap L_2 = \{w \in \Sigma^* : w \in L_1 \text{ and } w \in L_2\}$.
- Complement: $\neg L_1 = \{w \in \Sigma^* : w \notin L_1\}$.
- Reverse: $L_1^R = \{a_k \ldots a_1 \in \Sigma^* : a_1 \ldots a_k \in L_1 \text{ for each } a_i \in \Sigma\}$.
- Concatenation: $L_1 \circ L_2 = \{w_1 w_2 \in \Sigma^* : w_1 \in L_1 \text{ and } w_2 \in L_2\}$.
- Kleene star: $L_1^* = \{w_1 \ldots w_k \in \Sigma^* : w_i \in L_1\} \cup \{\underbrace{\varepsilon}_{\text{the empty string}}\}$.

- A better way to understand the Kleene star operation:

$$L_1^n = \underbrace{L_1 \circ L_1 \circ \cdots \circ L_1}_{n \text{ times}} \quad \longrightarrow \quad L_1^* = \bigcup_{n \in \mathbb{N}} L_1^n$$

Basic operations on languages

Given two languages $L_1, L_2 \subseteq \Sigma^*$, we can make the following operations:

- Union: $L_1 \cup L_2 = \{w \in \Sigma^* : w \in L_1 \text{ or } w \in L_2\}$.
- Intersection: $L_1 \cap L_2 = \{w \in \Sigma^* : w \in L_1 \text{ and } w \in L_2\}$.
- Complement: $\neg L_1 = \{w \in \Sigma^* : w \notin L_1\}$.
- Reverse: $L_1^R = \{a_k \dots a_1 \in \Sigma^* : a_1 \dots a_k \in L_1 \text{ for each } a_i \in \Sigma\}$.
- Concatenation: $L_1 \circ L_2 = \{w_1 w_2 \in \Sigma^* : w_1 \in L_1 \text{ and } w_2 \in L_2\}$.
- Kleene star: $L_1^* = \{w_1 \dots w_k \in \Sigma^* : w_i \in L_1\} \cup \{\varepsilon\}$.

  the empty string

- A better way to understand the Kleene star operation:

$$L_1^n = \underbrace{L_1 \circ L_1 \circ \cdots \circ L_1}_{n \text{ times}} \longrightarrow L_1^* = \bigcup_{n \in \mathbb{N}} L_1^n$$

Basic operations on languages

Given two languages $L_1, L_2 \subseteq \Sigma^*$, we can make the following operations:

- Union: $L_1 \cup L_2 = \{w \in \Sigma^* : w \in L_1 \text{ or } w \in L_2\}$.
- Intersection: $L_1 \cap L_2 = \{w \in \Sigma^* : w \in L_1 \text{ and } w \in L_2\}$.
- Complement: $\neg L_1 = \{w \in \Sigma^* : w \notin L_1\}$.
- Reverse: $L_1^R = \{a_k \ldots a_1 \in \Sigma^* : a_1 \ldots a_k \in L_1 \text{ for each } a_i \in \Sigma\}$.
- Concatenation: $L_1 \circ L_2 = \{w_1 w_2 \in \Sigma^* : w_1 \in L_1 \text{ and } w_2 \in L_2\}$.
- Kleene star: $L_1^* = \{w_1 \ldots w_k \in \Sigma^* : w_i \in L_1\} \cup \{\underbrace{\varepsilon}_{\text{the empty string}}\}$.
- A better way to understand the Kleene star operation:

$$L_1^n = \underbrace{L_1 \circ L_1 \circ \cdots \circ L_1}_{n \text{ times}} \quad \longrightarrow \quad L_1^* = \bigcup_{n \in \mathbb{N}} L_1^n$$

Basic operations on languages

Given two languages $L_1, L_2 \subseteq \Sigma^*$, we can make the following operations:

- Union: $L_1 \cup L_2 = \{w \in \Sigma^* : w \in L_1 \text{ or } w \in L_2\}$.
- Intersection: $L_1 \cap L_2 = \{w \in \Sigma^* : w \in L_1 \text{ and } w \in L_2\}$.
- Complement: $\neg L_1 = \{w \in \Sigma^* : w \notin L_1\}$.
- Reverse: $L_1^R = \{a_k \dots a_1 \in \Sigma^* : a_1 \dots a_k \in L_1 \text{ for each } a_i \in \Sigma\}$.
- Concatenation: $L_1 \circ L_2 = \{w_1 w_2 \in \Sigma^* : w_1 \in L_1 \text{ and } w_2 \in L_2\}$.
- Kleene star: $L_1^* = \{w_1 \dots w_k \in \Sigma^* : w_i \in L_1\} \cup \{\underbrace{\varepsilon}_{\text{the empty string}}\}$.

- A better way to understand the Kleene star operation:

$$L_1^n = \underbrace{L_1 \circ L_1 \circ \cdots \circ L_1}_{n \text{ times}} \longrightarrow L_1^* = \bigcup_{n \in \mathbb{N}} L_1^n$$

Basic operations on languages

Given two languages $L_1, L_2 \subseteq \Sigma^*$, we can make the following operations:

- Union: $L_1 \cup L_2 = \{w \in \Sigma^* : w \in L_1 \text{ or } w \in L_2\}$.
- Intersection: $L_1 \cap L_2 = \{w \in \Sigma^* : w \in L_1 \text{ and } w \in L_2\}$.
- Complement: $\neg L_1 = \{w \in \Sigma^* : w \notin L_1\}$.
- Reverse: $L_1^R = \{a_k \ldots a_1 \in \Sigma^* : a_1 \ldots a_k \in L_1 \text{ for each } a_i \in \Sigma\}$.
- Concatenation: $L_1 \circ L_2 = \{w_1 w_2 \in \Sigma^* : w_1 \in L_1 \text{ and } w_2 \in L_2\}$.
- Kleene star: $L_1^* = \{w_1 \ldots w_k \in \Sigma^* : w_i \in L_1\} \cup \{\underset{\substack{\downarrow \\ \text{the empty string}}}{\varepsilon}\}$.

- A better way to understand the Kleene star operation:

$$L_1^n = \underbrace{L_1 \circ L_1 \circ \cdots \circ L_1}_{n \text{ times}} \longrightarrow L_1^* = \bigcup_{n \in \mathbb{N}} L_1^n$$

Basic operations on languages

Given two languages $L_1, L_2 \subseteq \Sigma^*$, we can make the following operations:

- Union: $L_1 \cup L_2 = \{w \in \Sigma^* : w \in L_1 \text{ or } w \in L_2\}$.
- Intersection: $L_1 \cap L_2 = \{w \in \Sigma^* : w \in L_1 \text{ and } w \in L_2\}$.
- Complement: $\neg L_1 = \{w \in \Sigma^* : w \notin L_1\}$.
- Reverse: $L_1^R = \{a_k \dots a_1 \in \Sigma^* : a_1 \dots a_k \in L_1 \text{ for each } a_i \in \Sigma\}$.
- Concatenation: $L_1 \circ L_2 = \{w_1 w_2 \in \Sigma^* : w_1 \in L_1 \text{ and } w_2 \in L_2\}$.
- Kleene star: $L_1^* = \{w_1 \dots w_k \in \Sigma^* : w_i \in L_1\} \cup \{\underset{\substack{\downarrow \\ \text{the empty string}}}{\varepsilon}\}$.

- A better way to understand the Kleene star operation:

$$L_1^n = \underbrace{L_1 \circ L_1 \circ \cdots \circ L_1}_{n \text{ times}} \quad \longrightarrow \quad L_1^* = \bigcup_{n \in \mathbb{N}} L_1^n$$

What is a deterministic finite automaton?

Definition (DFA)

A deterministic finite automaton (DFA) is a five tuple $(Q, \Sigma, q_0, F, \delta)$:

1. Q: a set of states;
2. Σ: an alphabet set;
3. $q_0 \in Q$: the start state;
4. $F \subseteq Q$: a set of accept states;
5. $\delta : Q \times \Sigma \to Q$: a transition function.

What is a deterministic finite automaton?

Definition (DFA)

A deterministic finite automaton (DFA) is a five tuple $(Q, \Sigma, q_0, F, \delta)$:

1. Q: a set of states;
2. Σ: an alphabet set;
3. $q_0 \in Q$: the start state;
4. $F \subseteq Q$: a set of accept states;
5. $\delta : Q \times \Sigma \to Q$: a transition function.

What is a deterministic finite automaton?

Definition (DFA)

A deterministic finite automaton (DFA) is a five tuple $(Q, \Sigma, q_0, F, \delta)$:

1. Q: a set of states;
2. Σ: an alphabet set;
3. $q_0 \in Q$: the start state;
4. $F \subseteq Q$: a set of accept states;
5. $\delta : Q \times \Sigma \to Q$: a transition function.

What is a deterministic finite automaton?

Definition (DFA)

A deterministic finite automaton (DFA) is a five tuple $(Q, \Sigma, q_0, F, \delta)$:

1. Q: a set of states;
2. Σ: an alphabet set;
3. $q_0 \in Q$: the start state;
4. $F \subseteq Q$: a set of accept states;
5. $\delta : Q \times \Sigma \to Q$: a transition function.

What is a deterministic finite automaton?

Definition (DFA)

A deterministic finite automaton (DFA) is a five tuple $(Q, \Sigma, q_0, F, \delta)$:

1. Q: a set of states;
2. Σ: an alphabet set;
3. $q_0 \in Q$: the start state;
4. $F \subseteq Q$: a set of accept states;
5. $\delta : Q \times \Sigma \to Q$: a transition function.

What is a deterministic finite automaton?

Definition (DFA)

A deterministic finite automaton (DFA) is a five tuple $(Q, \Sigma, q_0, F, \delta)$:

1. Q: a set of states;
2. Σ: an alphabet set;
3. $q_0 \in Q$: the start state;
4. $F \subseteq Q$: a set of accept states;
5. $\delta : Q \times \Sigma \to Q$: a transition function.

What is a deterministic finite automaton?

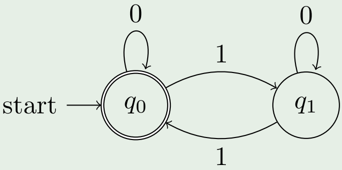
Definition (DFA)

A deterministic finite automaton (DFA) is a five tuple $(Q, \Sigma, q_0, F, \delta)$:

1. Q: a set of states;
2. Σ: an alphabet set;
3. $q_0 \in Q$: the start state;
4. $F \subseteq Q$: a set of accept states;
5. $\delta : Q \times \Sigma \to Q$: a transition function, e.g., $\delta(q_0, 1) = q_1$.

From DFA to language

Example (DFA)



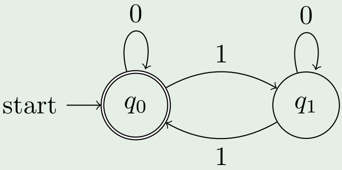
Exercise

Write down the language that the above DFA recognises.

Solution: $L = \{w \in \{0,1\}^* \mid w \text{ contains even number of 1s}\}$.

From DFA to language

Example (DFA)



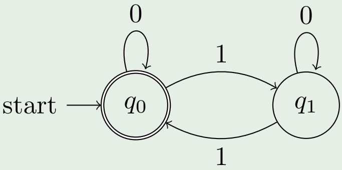
Exercise

Write down the language that the above DFA recognises.

Solution: $L = \{w \in \{0,1\}^* \mid w \text{ contains even number of 1s}\}$.

From DFA to language

Example (DFA)



Exercise

Write down the language that the above DFA recognises.

Solution: $L = \{w \in \{0,1\}^* \mid w \text{ contains even number of 1s}\}$.

From language to DFA

Example

Let $\Sigma = \{0, 1\}$ and $L = \{w \in \{0,1\}^* \mid w \text{ starts with } 0 \text{ and ends with } 1\}$.

Exercise

Design a DFA that recognises the above language.

Solution:

From language to DFA

Example

Let $\Sigma = \{0, 1\}$ and $L = \{w \in \{0,1\}^* \mid w \text{ starts with 0 and ends with 1}\}$.

Exercise

Design a DFA that recognises the above language.

Solution:

From language to DFA

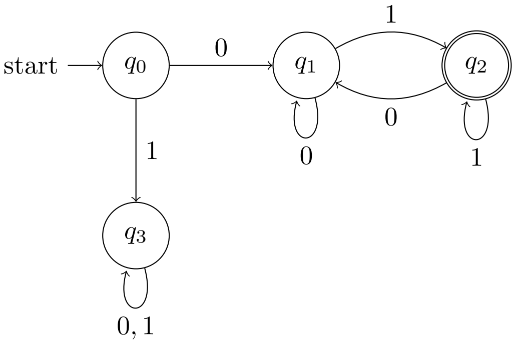
Example

Let $\Sigma = \{0, 1\}$ and $L = \{w \in \{0,1\}^* \mid w$ starts with 0 and ends with $1\}$.

Exercise

Design a DFA that recognises the above language.

Solution:



From language to DFA

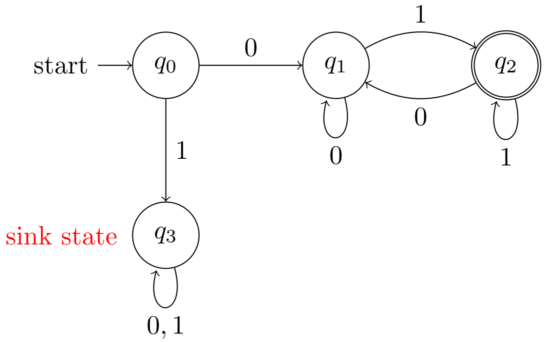
Example

Let $\Sigma = \{0, 1\}$ and $L = \{w \in \{0,1\}^* \mid w \text{ starts with 0 and ends with 1}\}$.

Exercise

Design a DFA that recognises the above language.

Solution:



What is a non-deterministic finite automaton?

Definition (NFA)

A non-deterministic finite automaton (NFA) is a five tuple $(Q, \Sigma, Q_0, F, \delta)$:

1. Q: a set of states;
2. Σ: an alphabet set;
3. $Q_0 \subseteq Q$: a set of start states;
4. $F \subseteq Q$: a set of accept states;
5. $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$: a transition function.

UTS:QSI
CENTRE FOR QUANTUM SOFTWARE AND INFORMATION

What is a non-deterministic finite automaton?

Definition (NFA)

A non-deterministic finite automaton (NFA) is a five tuple $(Q, \Sigma, Q_0, F, \delta)$:

1. Q: a set of states;
2. Σ: an alphabet set;
3. $Q_0 \subseteq Q$: a set of start states;
4. $F \subseteq Q$: a set of accept states;
5. $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$: a transition function.

UTS:QSI

CENTRE FOR QUANTUM SOFTWARE AND INFORMATION

What is a non-deterministic finite automaton?

Definition (NFA)

A non-deterministic finite automaton (NFA) is a five tuple $(Q, \Sigma, Q_0, F, \delta)$:

1. Q: a set of states;
2. Σ: an alphabet set;
3. $Q_0 \subseteq Q$: a set of start states;
4. $F \subseteq Q$: a set of accept states;
5. $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$: a transition function.

What is a non-deterministic finite automaton?

Definition (NFA)

A non-deterministic finite automaton (NFA) is a five tuple $(Q, \Sigma, Q_0, F, \delta)$:

1. Q: a set of states;
2. Σ: an alphabet set;
3. $Q_0 \subseteq Q$: a set of start states;
4. $F \subseteq Q$: a set of accept states;
5. $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$: a transition function.

What is a non-deterministic finite automaton?

Definition (NFA)

A non-deterministic finite automaton (NFA) is a five tuple $(Q, \Sigma, Q_0, F, \delta)$:

1. Q: a set of states;
2. Σ: an alphabet set;
3. $Q_0 \subseteq Q$: a set of start states;
4. $F \subseteq Q$: a set of accept states;
5. $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$: a transition function.

What is a non-deterministic finite automaton?

Definition (NFA)

A non-deterministic finite automaton (NFA) is a five tuple $(Q, \Sigma, Q_0, F, \delta)$:

1. Q: a set of states;
2. Σ: an alphabet set;
3. $Q_0 \subseteq Q$: a set of start states;
4. $F \subseteq Q$: a set of accept states;
5. $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$: a transition function.

What is a non-deterministic finite automaton?

Definition (NFA)

A non-deterministic finite automaton (NFA) is a five tuple $(Q, \Sigma, Q_0, F, \delta)$:

1. Q: a set of states;
2. Σ: an alphabet set;
3. $Q_0 \subseteq Q$: a set of start states;
4. $F \subseteq Q$: a set of accept states;
5. $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$: a transition function, e.g., $\delta(q_0, 1) = \{q_1, q_2\}$.

What is a non-deterministic finite automaton?

Definition (NFA)

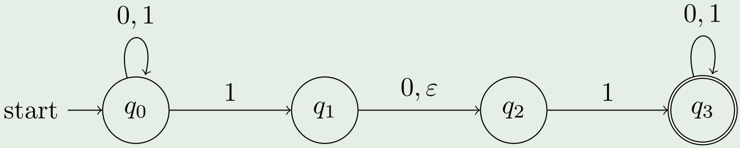
A non-deterministic finite automaton (NFA) is a five tuple $(Q, \Sigma, Q_0, F, \delta)$:

1. Q: a set of states;
2. Σ: an alphabet set;
3. $Q_0 \subseteq Q$: a set of start states;
4. $F \subseteq Q$: a set of accept states;
5. $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$: a transition function.

Note that 2^Q refers to the set consisting of all subsets of Q.

From NFA to language

Example (NFA)



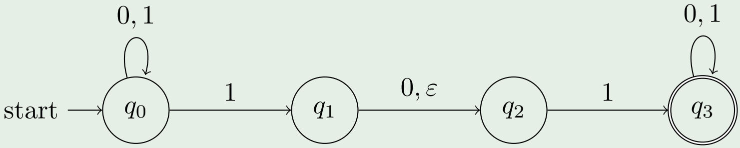
Exercise

Write down the language that the above NFA recognises.

Solution: $L = \{w \in \{0,1\}^* \mid w$ contains 11 or 101 as substrings.$\}$.

From NFA to language

Example (NFA)



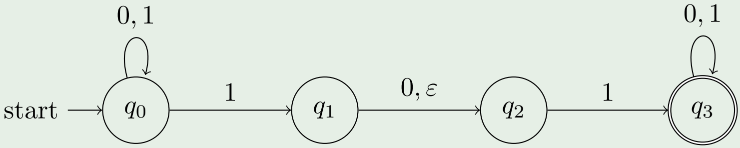
Exercise

Write down the language that the above NFA recognises.

Solution: $L = \{w \in \{0,1\}^* \mid w$ contains 11 or 101 as substrings.$\}$.

From NFA to language

Example (NFA)



Exercise

Write down the language that the above NFA recognises.

Solution: $L = \{w \in \{0, 1\}^* \mid w \text{ contains 11 or 101 as substrings.}\}$.

From language to NFA

Example

Let $\Sigma = \{0, 1\}$ and $L = \{w \in \{0,1\}^* \mid w \text{ starts with 0 and ends with 1}\}$.

Problem

Design an NFA that recognises the above language.

Solution:

From language to NFA

Example

Let $\Sigma = \{0, 1\}$ and $L = \{w \in \{0,1\}^* \mid w \text{ starts with } 0 \text{ and ends with } 1\}$.

Problem

Design an NFA that recognises the above language.

Solution:

From language to NFA

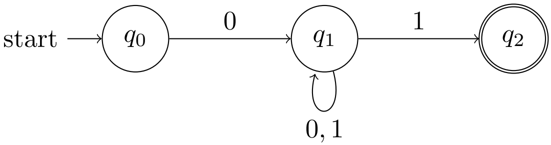
Example

Let $\Sigma = \{0, 1\}$ and $L = \{w \in \{0,1\}^* \mid w \text{ starts with } 0 \text{ and ends with } 1\}$.

Problem

Design an NFA that recognises the above language.

Solution:



What is a regular language?

Definition (Regular languages)

L is a regular language if there exists a DFA that recognises L.

Proposition (Closure properties)

If L_1 and L_2 are both regular languages, then

1. $L_1 \cup L_2$ *is a regular language;*
2. $L_1 \cap L_2$ *is a regular language.*

What is a regular language?

Definition (Regular languages)

L is a regular language if there exists a DFA that recognises L.

Proposition (Closure properties)

If L_1 and L_2 are both regular languages, then

1. $L_1 \cup L_2$ *is a regular language;*
2. $L_1 \cap L_2$ *is a regular language.*

What is a product construction?

Definition (Product construction of two DFAs)

Let $M = (P, \Sigma, p_0, E, \alpha)$ and $N = (Q, \Sigma, q_0, F, \beta)$ be two DFAs. The product construction for recognising $L(M) \cup L(N)$ is to construct $O = (R, \Sigma, r_0, G, \gamma)$ where

1. the state set $R = P \times Q$;
2. the start state $r_0 = (p_0, q_0)$;
3. the accept state set $G = \{(p, q) \mid p \in E \text{ or } q \in F\}$.
4. the transition function $\gamma : (P \times Q) \times \Sigma \to (P \times Q)$ given by $\gamma((p, q), a)) = (\alpha(p, a), \beta(q, a))$.

What is a product construction?

Definition (Product construction of two DFAs)

Let $M = (P, \Sigma, p_0, E, \alpha)$ and $N = (Q, \Sigma, q_0, F, \beta)$ be two DFAs. The product construction for recognising $L(M) \cup L(N)$ is to construct $O = (R, \Sigma, r_0, G, \gamma)$ where

1. the state set $R = P \times Q$;
2. the start state $r_0 = (p_0, q_0)$;
3. the accept state set $G = \{(p, q) \mid p \in E \text{ or } q \in F\}$.
4. the transition function $\gamma : (P \times Q) \times \Sigma \to (P \times Q)$ given by $\gamma((p, q), a)) = (\alpha(p, a), \beta(q, a))$.

What is a product construction?

Definition (Product construction of two DFAs)

Let $M = (P, \Sigma, p_0, E, \alpha)$ and $N = (Q, \Sigma, q_0, F, \beta)$ be two DFAs. The product construction for recognising $L(M) \cup L(N)$ is to construct $O = (R, \Sigma, r_0, G, \gamma)$ where

1. the state set $R = P \times Q$;
2. the start state $r_0 = (p_0, q_0)$;
3. the accept state set $G = \{(p, q) \mid p \in E \text{ or } q \in F\}$.
4. the transition function $\gamma : (P \times Q) \times \Sigma \to (P \times Q)$ given by $\gamma((p, q), a)) = (\alpha(p, a), \beta(q, a))$.

What is a product construction?

Definition (Product construction of two DFAs)

Let $M = (P, \Sigma, p_0, E, \alpha)$ and $N = (Q, \Sigma, q_0, F, \beta)$ be two DFAs. The product construction for recognising $L(M) \cup L(N)$ is to construct $O = (R, \Sigma, r_0, G, \gamma)$ where

1. the state set $R = P \times Q$;
2. the start state $r_0 = (p_0, q_0)$;
3. the accept state set $G = \{(p, q) \mid p \in E \text{ or } q \in F\}$.
4. the transition function $\gamma : (P \times Q) \times \Sigma \to (P \times Q)$ given by $\gamma((p, q), a)) = (\alpha(p, a), \beta(q, a))$.

What is a product construction?

Definition (Product construction of two DFAs)

Let $M = (P, \Sigma, p_0, E, \alpha)$ and $N = (Q, \Sigma, q_0, F, \beta)$ be two DFAs. The product construction for recognising $L(M) \cup L(N)$ is to construct $O = (R, \Sigma, r_0, G, \gamma)$ where

1. the state set $R = P \times Q$;
2. the start state $r_0 = (p_0, q_0)$;
3. the accept state set $G = \{(p, q) \mid p \in E \text{ or } q \in F\}$.
4. the transition function $\gamma : (P \times Q) \times \Sigma \to (P \times Q)$ given by $\gamma((p, q), a)) = (\alpha(p, a), \beta(q, a))$.

UTS:QSI
CENTRE FOR QUANTUM SOFTWARE AND INFORMATION

What is a product construction?

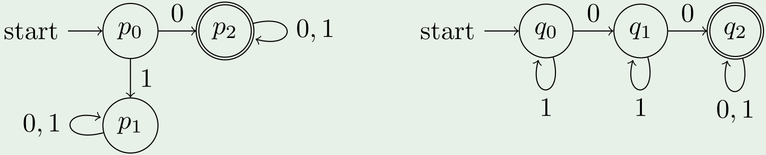
Definition (Product construction of two DFAs)

Let $M = (P, \Sigma, p_0, E, \alpha)$ and $N = (Q, \Sigma, q_0, F, \beta)$ be two DFAs. The product construction for recognising $L(M) \cap L(N)$ is to construct $O = (R, \Sigma, r_0, G, \gamma)$ where

1. the state set $R = P \times Q$;
2. the start state $r_0 = (p_0, q_0)$;
3. the accept state set $G = \{(p, q) \mid p \in E \text{ and } q \in F\}$.
4. the transition function $\gamma : (P \times Q) \times \Sigma \to (P \times Q)$ given by $\gamma((p, q), a)) = (\alpha(p, a), \beta(q, a))$.

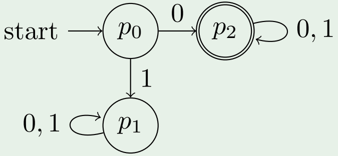
Tutorial: product construction

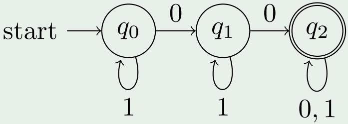
Construct a DFA for $L_1 \cup L_2$ (recognised by the given two DFAs, respectively).

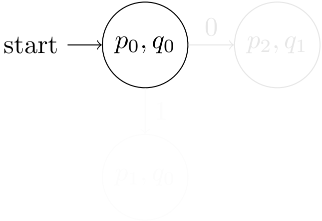


Tutorial: product construction

Construct a DFA for $L_1 \cup L_2$ (recognised by the given two DFAs, respectively).

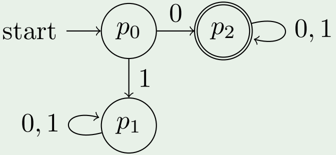


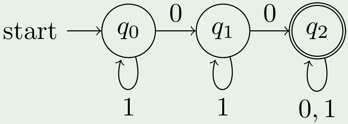


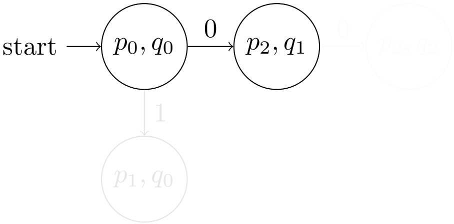


Tutorial: product construction

Construct a DFA for $L_1 \cup L_2$ (recognised by the given two DFAs, respectively).

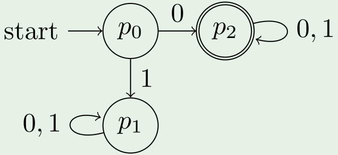


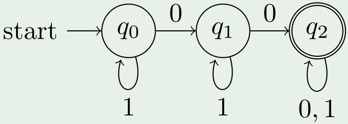


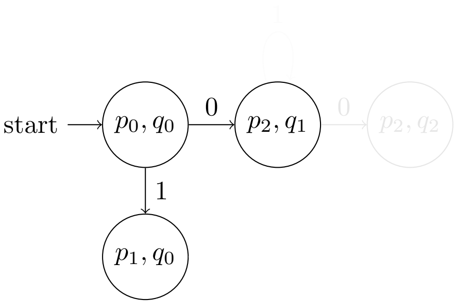


Tutorial: product construction

Construct a DFA for $L_1 \cup L_2$ (recognised by the given two DFAs, respectively).

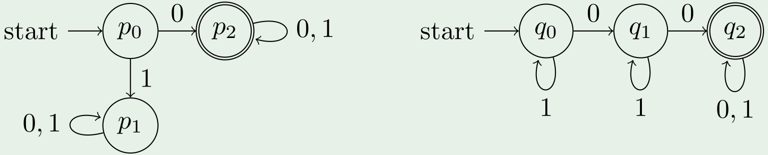


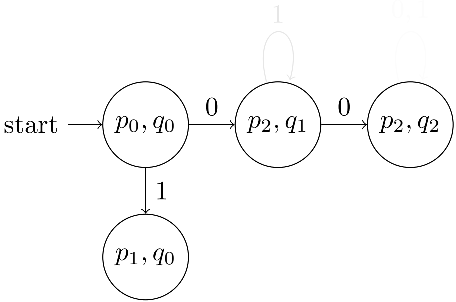




Tutorial: product construction

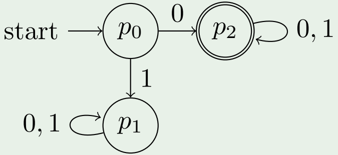
Construct a DFA for $L_1 \cup L_2$ (recognised by the given two DFAs, respectively).

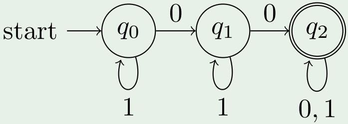


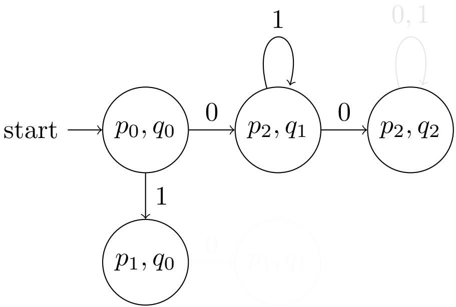


Tutorial: product construction

Construct a DFA for $L_1 \cup L_2$ (recognised by the given two DFAs, respectively).

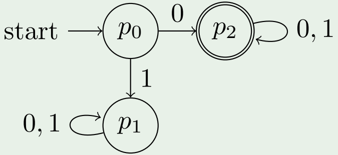


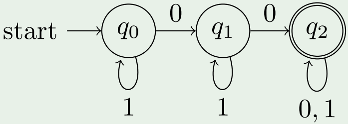


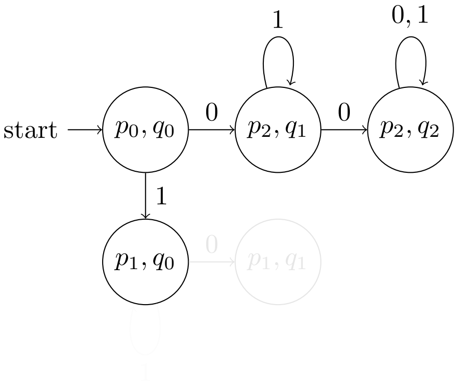


Tutorial: product construction

Construct a DFA for $L_1 \cup L_2$ (recognised by the given two DFAs, respectively).

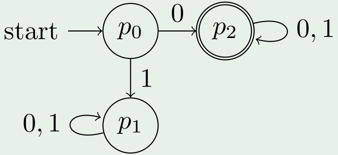


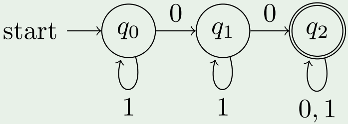


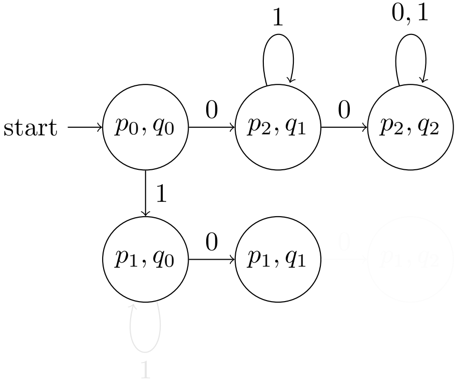


Tutorial: product construction

Construct a DFA for $L_1 \cup L_2$ (recognised by the given two DFAs, respectively).

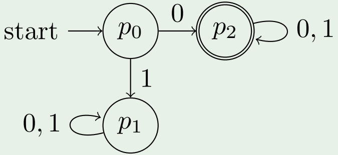


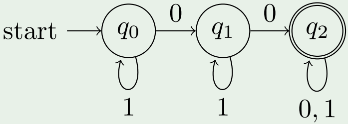


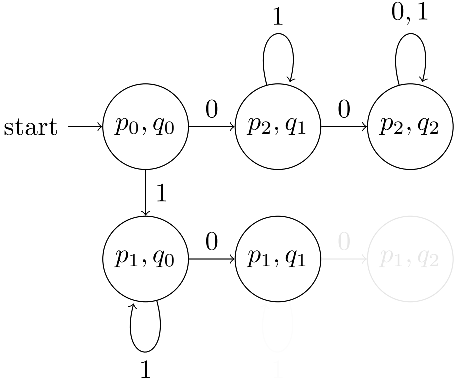


Tutorial: product construction

Construct a DFA for $L_1 \cup L_2$ (recognised by the given two DFAs, respectively).

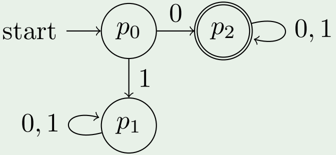


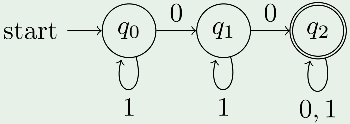


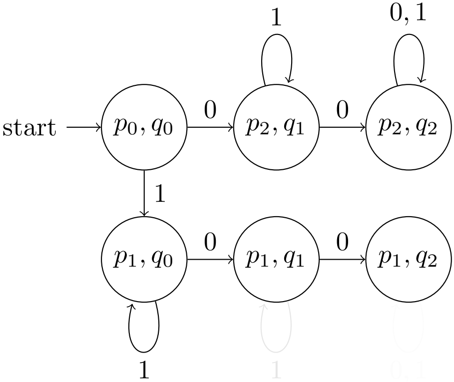


Tutorial: product construction

Construct a DFA for $L_1 \cup L_2$ (recognised by the given two DFAs, respectively).

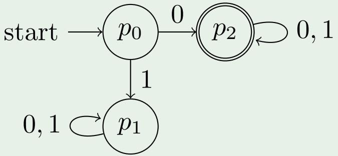


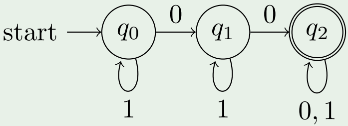


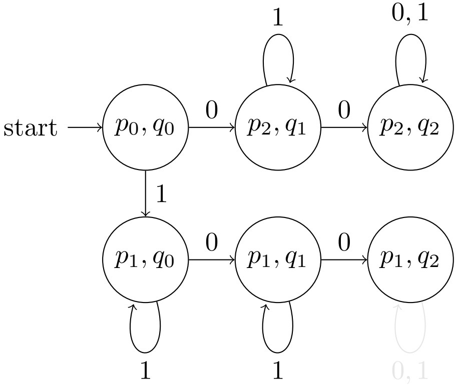


Tutorial: product construction

Construct a DFA for $L_1 \cup L_2$ (recognised by the given two DFAs, respectively).

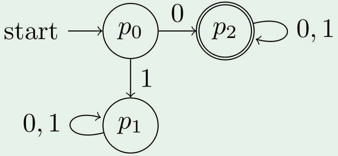


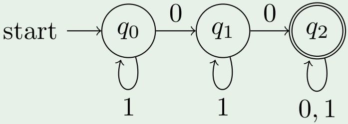


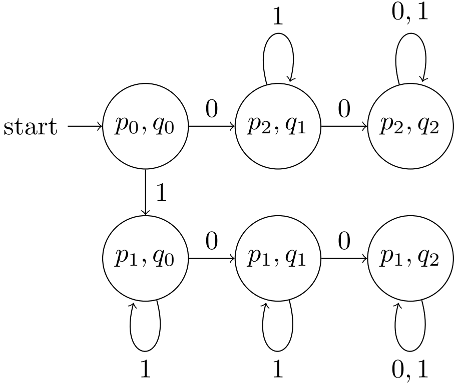


Tutorial: product construction

Construct a DFA for $L_1 \cup L_2$ (recognised by the given two DFAs, respectively).

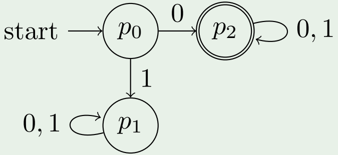


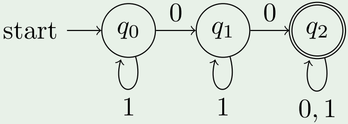


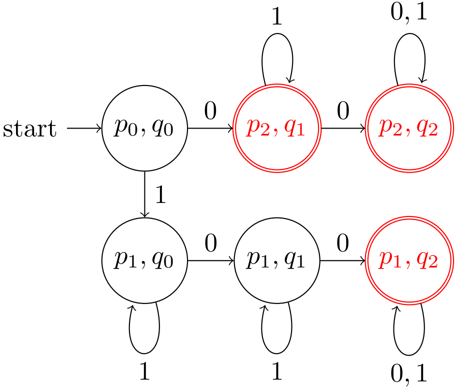


Tutorial: product construction

Construct a DFA for $L_1 \cup L_2$ (recognised by the given two DFAs, respectively).







Tutorial: product construction

Construct a DFA for $L_1 \cap L_2$ (recognised by the given two DFAs, respectively).