COMP2111 Week 8
Term 1, 2019
Week 7 recap

Week 7 recap: State machines/Transition systems

Abstractions of step-by-step processes

- Definitions:
  - States and Transitions
  - (Non-)determinism
  - Reachability
  - Safety and Liveness
- The Invariant Principle
- Termination
- Finite automata:
  - DFAs, NFAs
  - Regular languages

Definitions

A **transition system** is a pair $(S, \rightarrow)$ where:

- S is a set (of **states**), and
- $\rightarrow \subseteq S \times S$ is a (**transition**) **relation**.

- S may have a designated **start state**, $s_0 \in S$
- S may have designated **final states**, $F \subseteq S$
- The transitions may be **labelled** by elements of a set Λ:
  - $\rightarrow \subseteq S \times \Lambda \times S$
  - $(s, a, s') \in \rightarrow$ is written as $s \xrightarrow{a} s'$
- If $\rightarrow$ is a function we say the system is **deterministic**, in general it is **non-deterministic**

Runs and reachability

Given a transition system $(S, \rightarrow)$ and states $s, s' \in S$,

- a **run** from s is a (possibly infinite) sequence $s_1, s_2, \ldots$ such that $s = s_1$ and $s_i \rightarrow s_{i+1}$ for all $i \geq 1$.
- we say s' is **reachable** from s, written $s \rightarrow^* s'$, if (s, s') is in the transitive closure of $\rightarrow$.

Safety and Liveness

Common problem (Safety)

Will every run of a transition system avoid a particular state or states? Equivalently, will some run of a transition system reach a particular state or states?

Common problem (Liveness)

Will every run of a transition system reach a particular state or states? Equivalently, will some run of a transition system avoid a particular state or states?

The Invariant Principle (safety)

A **preserved invariant** of a transition system is a unary predicate φ on states such that if $\varphi(s)$ holds and $s \to s'$ then $\varphi(s')$ holds.

Invariant principle

If a preserved invariant holds at a state s, then it holds for all states reachable from s.

Example

Example

- States: $\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$
- Transition:
  - $(x, y, r) \rightarrow (x^2, \frac{y}{2}, r)$ if y is even
  - $(x, y, r) \rightarrow (x^2, \frac{y-1}{2}, rx)$ if y is odd
- Preserved invariant: rx^y is a constant
- $\Rightarrow$ All states reachable from $(m, n, 1)$ will satisfy $rx^y = m^n$
- $\Rightarrow$ if $(x, 0, r)$ is reachable from $(m, n, 1)$ then $r = m^n$.

Example

Example

- States: $\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$
- Transition:
  - $(x, y, r) \to (x^2, \frac{y}{2}, r)$ if y is even
  - $(x, y, r) \to (x^2, \frac{y-1}{2}, rx)$ if y is odd
- Preserved invariant: rx^y is a constant
- $\Rightarrow$ All states reachable from $(m, n, 1)$ will satisfy $rx^y = m^n$
- $\Rightarrow$ if $(x, 0, r)$ is reachable from $(m, n, 1)$ then $r = m^n$.

Example

Example

- States: $\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$
- Transition:
  - $(x, y, r) \to (x^2, \frac{y}{2}, r)$ if y is even
  - $(x, y, r) \to (x^2, \frac{y-1}{2}, rx)$ if y is odd
- Preserved invariant: rx^y is a constant
- $\Rightarrow$ All states reachable from $(m, n, 1)$ will satisfy $rx^y = m^n$
- $\Rightarrow$ if $(x, 0, r)$ is reachable from $(m, n, 1)$ then $r = m^n$.

Example

Example

- States: $\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$
- Transition:
  - $(x, y, r) \to (x^2, \frac{y}{2}, r)$ if y is even
  - $(x, y, r) \to (x^2, \frac{y-1}{2}, rx)$ if y is odd
- Preserved invariant: rx^y is a constant
- $\Rightarrow$ All states reachable from $(m, n, 1)$ will satisfy $rx^y = m^n$
- $\Rightarrow$ if $(x, 0, r)$ is reachable from $(m, n, 1)$ then $r = m^n$.

Termination (liveness)

A transition system $(S, \to)$ **terminates** from a state s if there is an N such that all runs from s have length at most N.

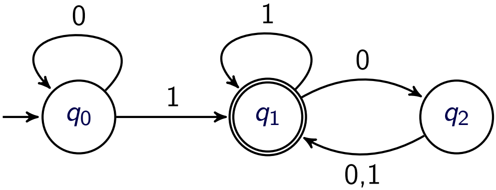
A **derived variable** is a function $f : S \to \mathbb{R}$.

A derived variable is **strictly decreasing** if $s \to s'$ implies $f(s) > f(s')$.

Theorem

If f is an $\mathbb{N}$-valued, strictly decreasing derived variable, then the length of any run from s is at most $f(s)$.

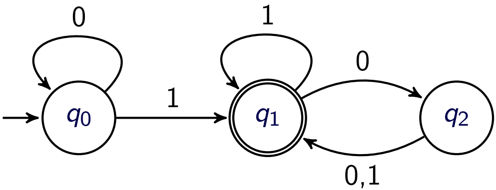
Deterministic Finite Automata



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

- Q is a finite set of states
- Σ is the input alphabet
- $\delta : Q \times \Sigma \to Q$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of final/accepting states

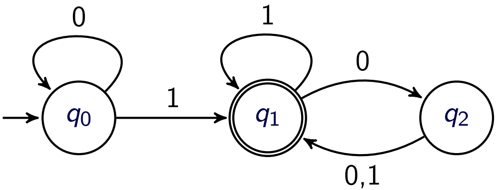
Language of a DFA



A DFA accepts a sequence of symbols from Σ – i.e. elements of Σ^*

Informally: A word defines a run in the DFA and the word is accepted if the run ends in a final state.

Language of a DFA

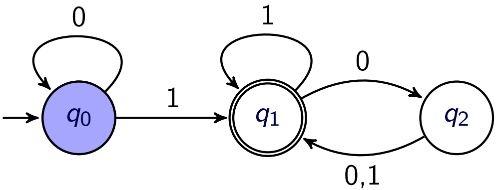


w: 1001

A DFA accepts a sequence of symbols from Σ – i.e. elements of Σ^*

- Start in state q_0
- Take the first symbol of w
- Repeat the following until there are no symbols left:
  - Based on the current state and current input symbol, transition to the appropriate state determined by δ
  - Move to the next symbol in w
- Accept if the process ends in a final state, otherwise reject.

Language of a DFA

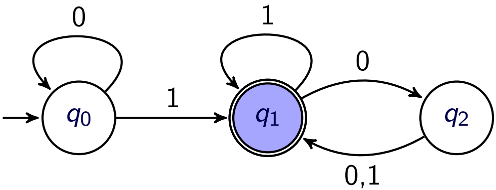


w: 1001

A DFA accepts a sequence of symbols from Σ – i.e. elements of Σ^*

- Start in state q_0
- Take the first symbol of w
- Repeat the following until there are no symbols left:
  - Based on the current state and current input symbol, transition to the appropriate state determined by δ
  - Move to the next symbol in w
- Accept if the process ends in a final state, otherwise reject.

Language of a DFA

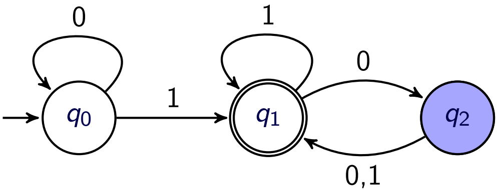


w: 1001

A DFA accepts a sequence of symbols from Σ – i.e. elements of Σ^*

- Start in state q_0
- Take the first symbol of w
- Repeat the following until there are no symbols left:
  - Based on the current state and current input symbol, transition to the appropriate state determined by δ
  - Move to the next symbol in w
- Accept if the process ends in a final state, otherwise reject.

Language of a DFA

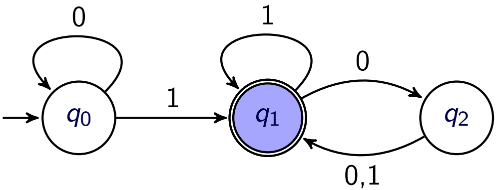


w: 1001

A DFA accepts a sequence of symbols from Σ – i.e. elements of Σ^*

- Start in state q_0
- Take the first symbol of w
- Repeat the following until there are no symbols left:
  - Based on the current state and current input symbol, transition to the appropriate state determined by δ
  - Move to the next symbol in w
- Accept if the process ends in a final state, otherwise reject.

Language of a DFA

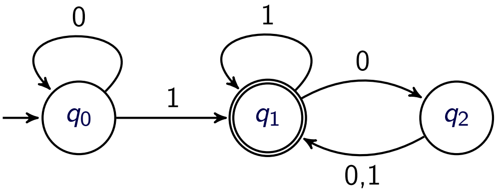


w: 1001

A DFA accepts a sequence of symbols from Σ – i.e. elements of Σ^*

- Start in state q_0
- Take the first symbol of w
- Repeat the following until there are no symbols left:
  - Based on the current state and current input symbol, transition to the appropriate state determined by δ
  - Move to the next symbol in w
- Accept if the process ends in a final state, otherwise reject.

Language of a DFA

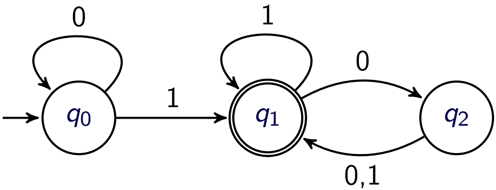


w: 1001 ✓

A DFA accepts a sequence of symbols from Σ – i.e. elements of Σ^*

- Start in state q_0
- Take the first symbol of w
- Repeat the following until there are no symbols left:
  - Based on the current state and current input symbol, transition to the appropriate state determined by δ
  - Move to the next symbol in w
- Accept if the process ends in a final state, otherwise reject.

Language of a DFA



$$L(\mathcal{A}) = \{1, 01, 11, 101, \ldots\}$$

A DFA accepts a sequence of symbols from Σ – i.e. elements of Σ^*

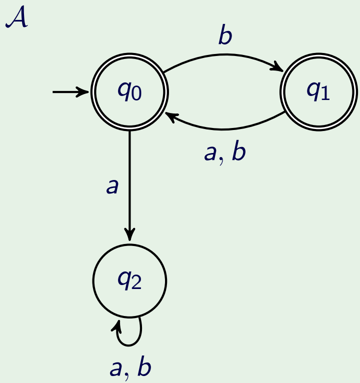
For a DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, the **language of** $\mathcal{A}$, $L(\mathcal{A})$, is the set of words from Σ^* which are accepted by $\mathcal{A}$

A language $L \subseteq \Sigma^*$ is **regular** if there is some DFA $\mathcal{A}$ such that $L = L(\mathcal{A})$

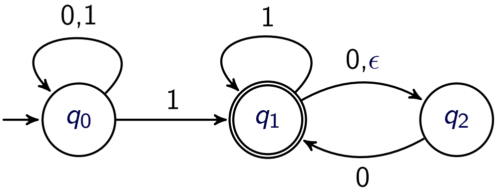
Example

Example

$\mathcal{A}$ such that $L(\mathcal{A}) = \{w \in \{a, b\}^* \ : \text{ every odd symbol is } b\}$



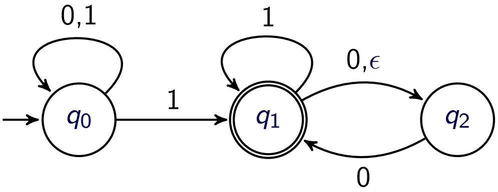
Non-deterministic Finite Automata



A **non-deterministic finite automaton (NFA)** is a non-deterministic, finite state acceptor.

More general than DFAs: A DFA is an NFA

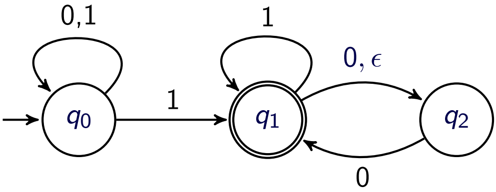
Non-deterministic Finite Automata



Formally, a **non-deterministic finite automaton (NFA)** is a tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of states
- Σ is the input alphabet
- $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q$ is the transition relation
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of final/accepting states

Language of an NFA



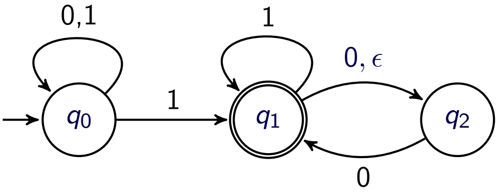
An NFA accepts a sequence of symbols from Σ – i.e. elements of Σ^*

Informally: A word defines several runs in the NFA and the word is accepted if **at least one run** ends in a final state.

Note 1: Runs can end prematurely (these don't count)

Note 2: An NFA will always "choose wisely"

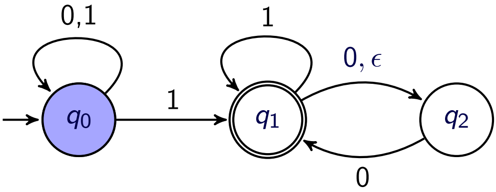
Language of an NFA



w: 1000

- Start in state q_0
- Take the first symbol of w
- Repeat until there are no symbols left or no transitions available:
  - Based on the current state and current input symbol or ϵ, transition to any state determined by δ
  - If not an ϵ-transition, move to the next symbol in w
- Accept if there are no symbols left and the process ends in a final state, otherwise reject.

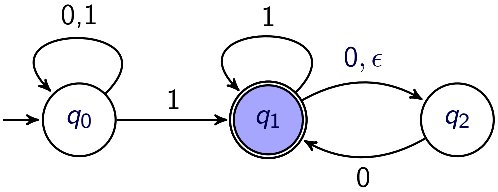
Language of an NFA



w: 1000

- Start in state q_0
- Take the first symbol of w
- Repeat until there are no symbols left or no transitions available:
  - Based on the current state and current input symbol or ϵ, transition to any state determined by δ
  - If not an ϵ-transition, move to the next symbol in w
- Accept if there are no symbols left and the process ends in a final state, otherwise reject.

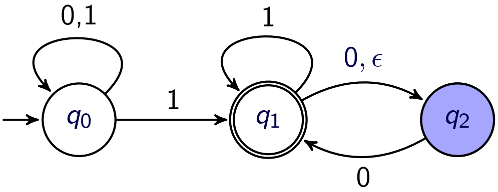
Language of an NFA



w: 1000

- Start in state q_0
- Take the first symbol of w
- Repeat until there are no symbols left or no transitions available:
  - Based on the current state and current input symbol or ϵ, transition to any state determined by δ
  - If not an ϵ-transition, move to the next symbol in w
- Accept if there are no symbols left and the process ends in a final state, otherwise reject.

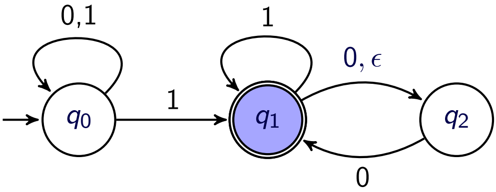
Language of an NFA



w: 1000

- Start in state q_0
- Take the first symbol of w
- Repeat until there are no symbols left or no transitions available:
  - Based on the current state and current input symbol or ϵ, transition to any state determined by δ
  - If not an ϵ-transition, move to the next symbol in w
- Accept if there are no symbols left and the process ends in a final state, otherwise reject.

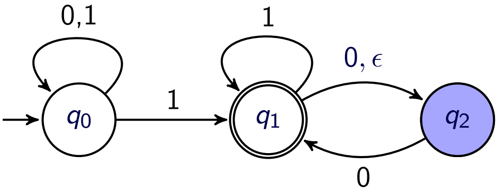
Language of an NFA



w: 1000

- Start in state q_0
- Take the first symbol of w
- Repeat until there are no symbols left or no transitions available:
  - Based on the current state and current input symbol or ϵ, transition to any state determined by δ
  - If not an ϵ-transition, move to the next symbol in w
- Accept if there are no symbols left and the process ends in a final state, otherwise reject.

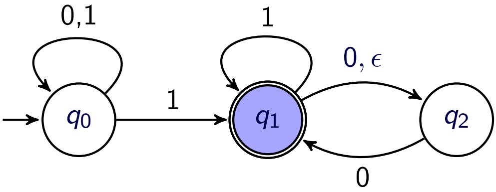
Language of an NFA



w: 1000

- Start in state q_0
- Take the first symbol of w
- Repeat until there are no symbols left or no transitions available:
    - Based on the current state and current input symbol or ϵ, transition to any state determined by δ
    - If not an ϵ-transition, move to the next symbol in w
- Accept if there are no symbols left and the process ends in a final state, otherwise reject.

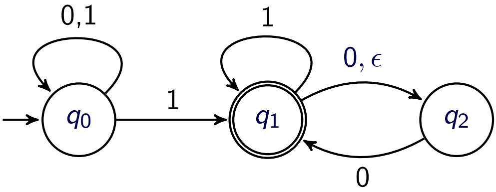
Language of an NFA



w: 1000

- Start in state q_0
- Take the first symbol of w
- Repeat until there are no symbols left or no transitions available:
  - Based on the current state and current input symbol or ϵ, transition to any state determined by δ
  - If not an ϵ-transition, move to the next symbol in w
- Accept if there are no symbols left and the process ends in a final state, otherwise reject.

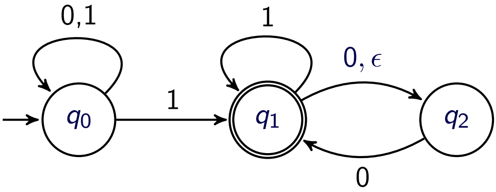
Language of an NFA



w: 1000 ✓

- Start in state q_0
- Take the first symbol of w
- Repeat until there are no symbols left or no transitions available:
  - Based on the current state and current input symbol or ϵ, transition to any state determined by δ
  - If not an ϵ-transition, move to the next symbol in w
- Accept if there are no symbols left and the process ends in a final state, otherwise reject.

Language of an NFA



$$L(\mathcal{A}) = \{1, 01, 11, 10, \ldots\}$$

For an NFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, the **language of** $\mathcal{A}$, $L(\mathcal{A})$, is the set of words from Σ^* which are accepted by $\mathcal{A}$