COSE312: Compilers

Lecture 3 — Lexical Analysis (2)

Hakjoo Oh
2017 Spring

Specification, Recognition, and Automation

1. **Specification**: how to specify lexical patterns?
   - In C, identifiers are strings like `x`, `xy`, `match0`, and `_abc`.
   - Numbers are strings like 3, 12, `0.012`, and `3.5E4`.

   $\Rightarrow$ *regular expressions*
2. **Recognition**: how to *recognize* the lexical patterns?
   - Recognize `match0` as an identifier.
   - Recognize 512 as a number.

   $\Rightarrow$ *deterministic finite automata*.
3. **Automation**: how to automatically generate string recognizers from specifications?

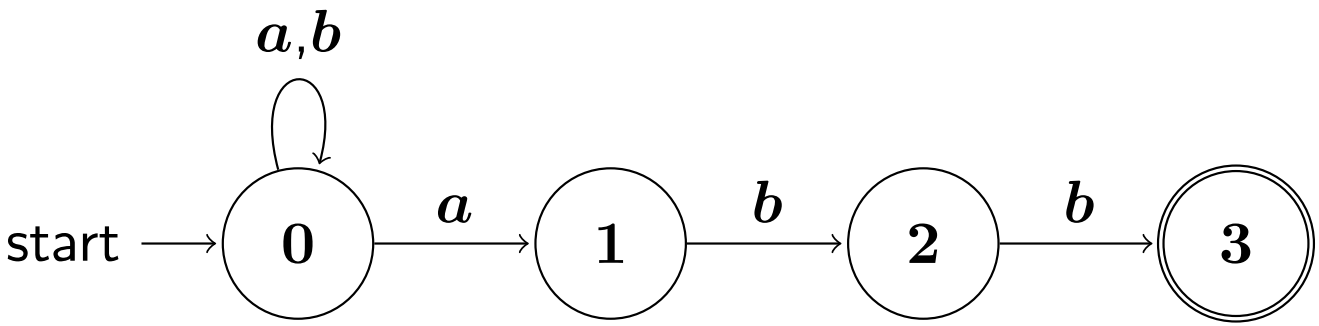
   $\Rightarrow$ *Thompson's construction* and *subset construction*

Part 2: String Recognition by Finite Automata

- Non-deterministic finite automata
- Deterministic finite automata

String Recognizer in NFA

An NFA that recognizes strings $(a|b)^*abb$:



Non-deterministic Finite Automata

Definition (NFA)

A *nondeterministic finite automaton* (or *NFA*) is defined as,

$$M = (Q, \Sigma, \delta, q_0, F)$$

where

- Q: a finite set of *states*
- Σ: a finite set of *input symbols* (or input alphabet). We assume that $\epsilon \notin \Sigma$.
- $q_0 \in Q$: the *initial state*
- $F \subseteq Q$: a set of *final states* (or *accepting states*)
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$: *transition function*

Example

Definition of an NFA:

$$(\{0, 1, 2, 3\}, \{a, b\}, \delta, 0, \{3\})$$

$$\begin{array}{ll} \delta(0, a) = \{0, 1\} & \delta(0, b) = \{0\} \\ \delta(1, a) = \emptyset & \delta(1, b) = \{2\} \\ \delta(2, a) = \emptyset & \delta(2, b) = \{3\} \\ \delta(3, a) = \emptyset & \delta(3, b) = \emptyset \end{array}$$

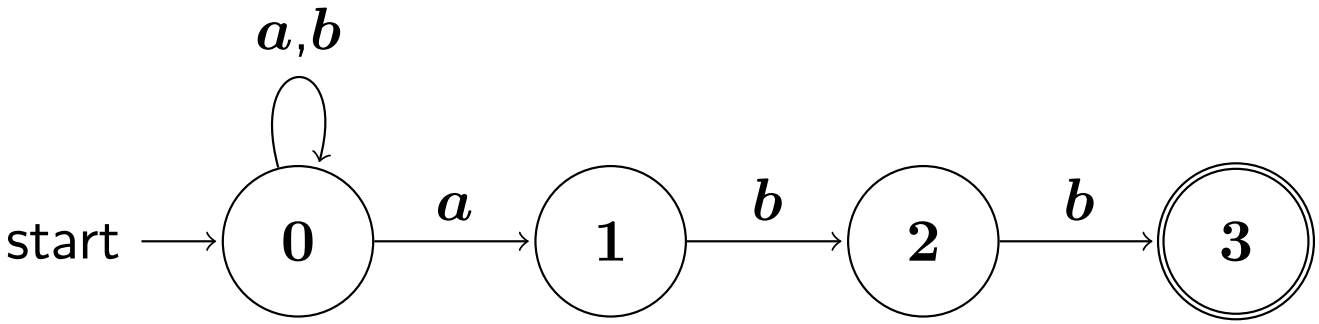
Example

Definition of an NFA:

$$(\{0, 1, 2, 3\}, \{a, b\}, \delta, 0, \{3\})$$

$$\begin{array}{ll} \delta(0, a) = \{0, 1\} & \delta(0, b) = \{0\} \\ \delta(1, a) = \emptyset & \delta(1, b) = \{2\} \\ \delta(2, a) = \emptyset & \delta(2, b) = \{3\} \\ \delta(3, a) = \emptyset & \delta(3, b) = \emptyset \end{array}$$

The *transition graph*:



Example

The *transition table*:

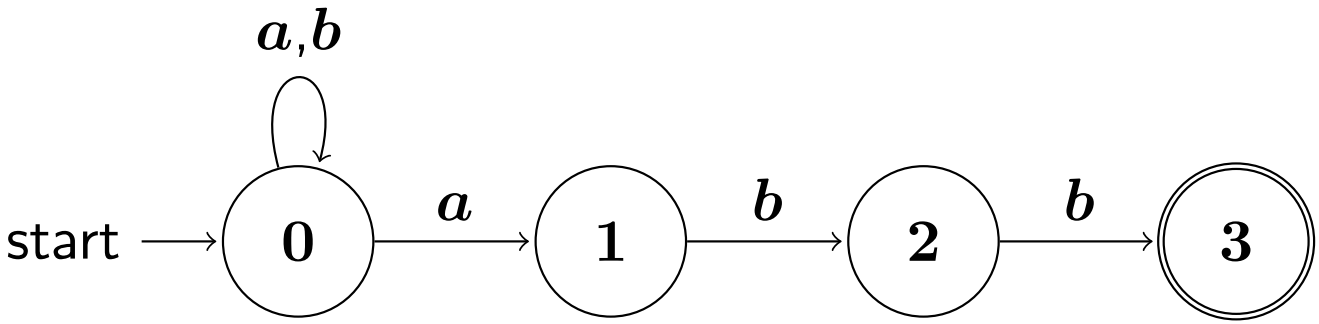
State	a	b	ϵ
$\mathbf{0}$	$\{0, 1\}$	$\{0\}$	$\emptyset$
$\mathbf{1}$	$\emptyset$	$\{2\}$	$\emptyset$
$\mathbf{2}$	$\emptyset$	$\{3\}$	$\emptyset$
$\mathbf{3}$	$\emptyset$	$\emptyset$	$\emptyset$

String Recognition

- An NFA recognizes a string w if there is a path in the transition graph labeled by w.

String Recognition

- An NFA recognizes a string w if there is a path in the transition graph labeled by w.

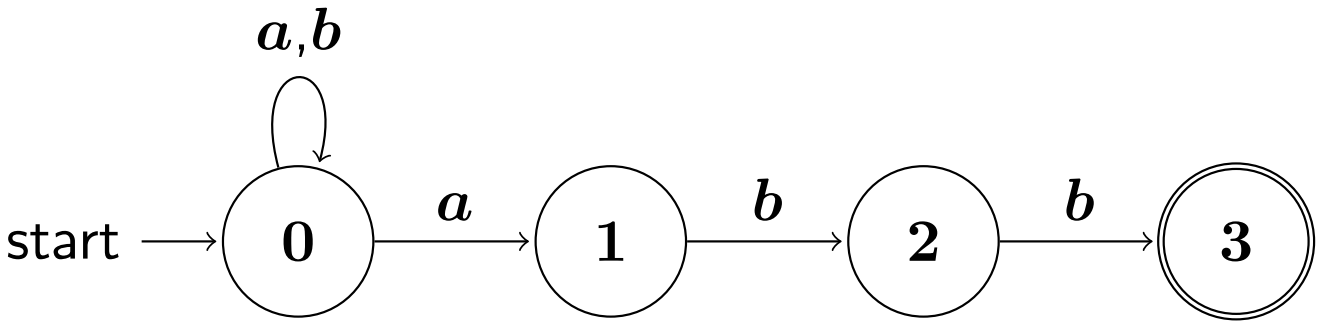


String $aabb$ is accepted because

$$0 \xrightarrow{a} 0 \xrightarrow{a} 1 \xrightarrow{b} 2 \xrightarrow{b} 3$$

String Recognition

- An NFA recognizes a string w if there is a path in the transition graph labeled by w.



String $aabb$ is accepted because

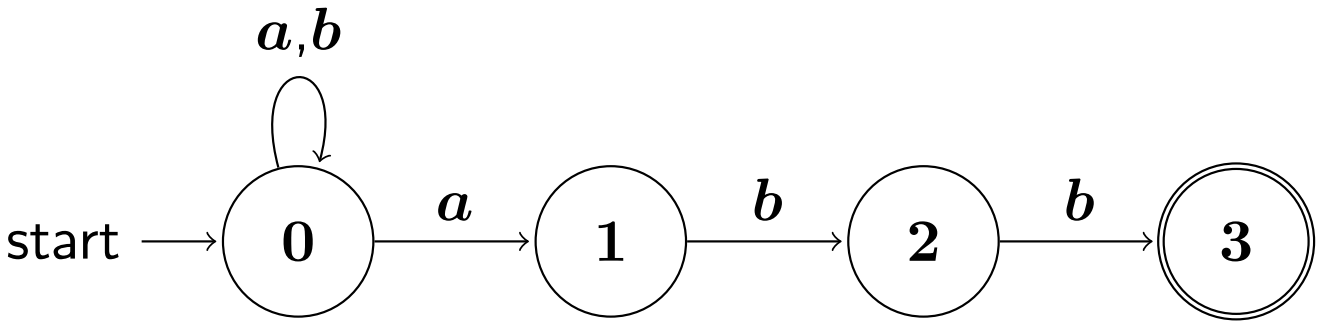
$$0 \xrightarrow{a} 0 \xrightarrow{a} 1 \xrightarrow{b} 2 \xrightarrow{b} 3$$

In general, the automaton recognizes any strings that end with abb:

$$L = \{wabb \mid w \in \{a, b\}^*\}$$

String Recognition

- An NFA recognizes a string w if there is a path in the transition graph labeled by w.



String $aabb$ is accepted because

$$0 \xrightarrow{a} 0 \xrightarrow{a} 1 \xrightarrow{b} 2 \xrightarrow{b} 3$$

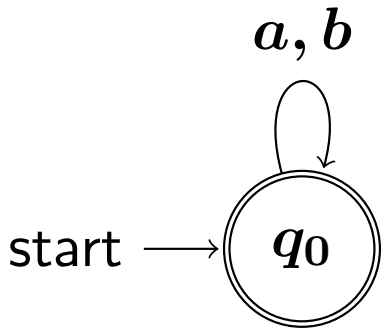
In general, the automaton recognizes any strings that end with abb:

$$L = \{wabb \mid w \in \{a, b\}^*\}$$

- The *language* of an NFA is the set of recognizable strings.

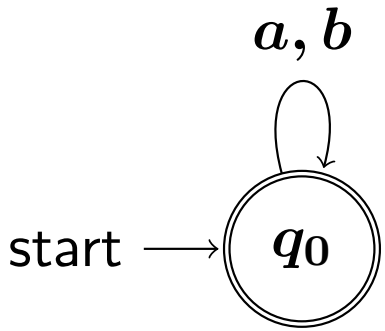
Exercises

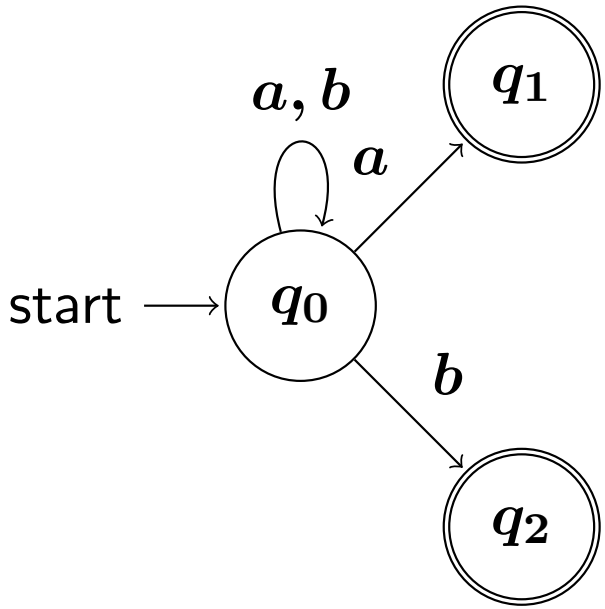
Find the languages of the NFAs:

- 

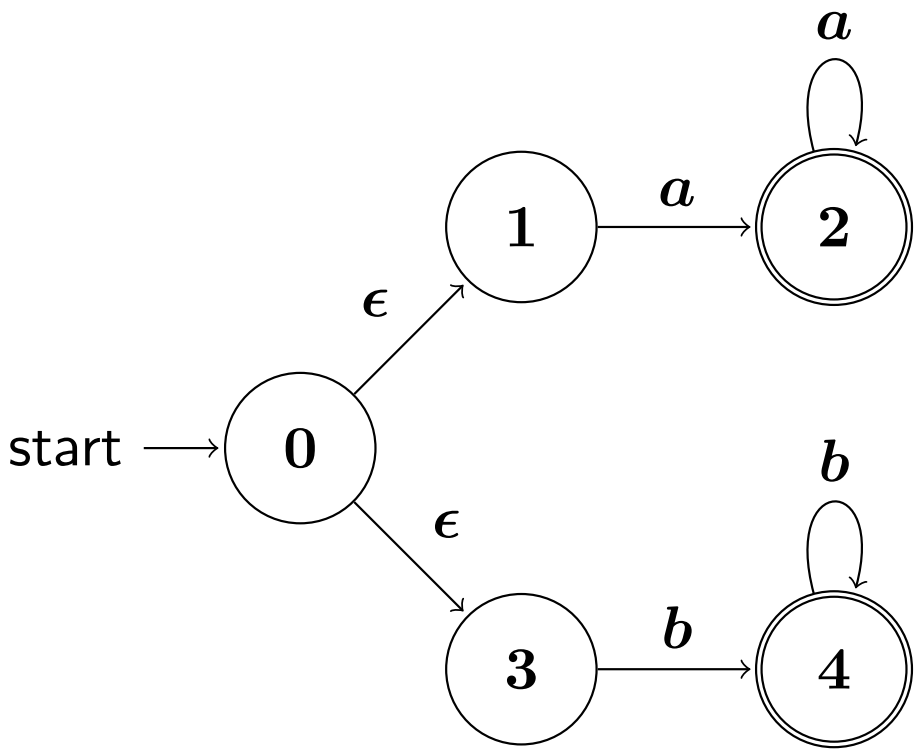
Exercises

Find the languages of the NFAs:

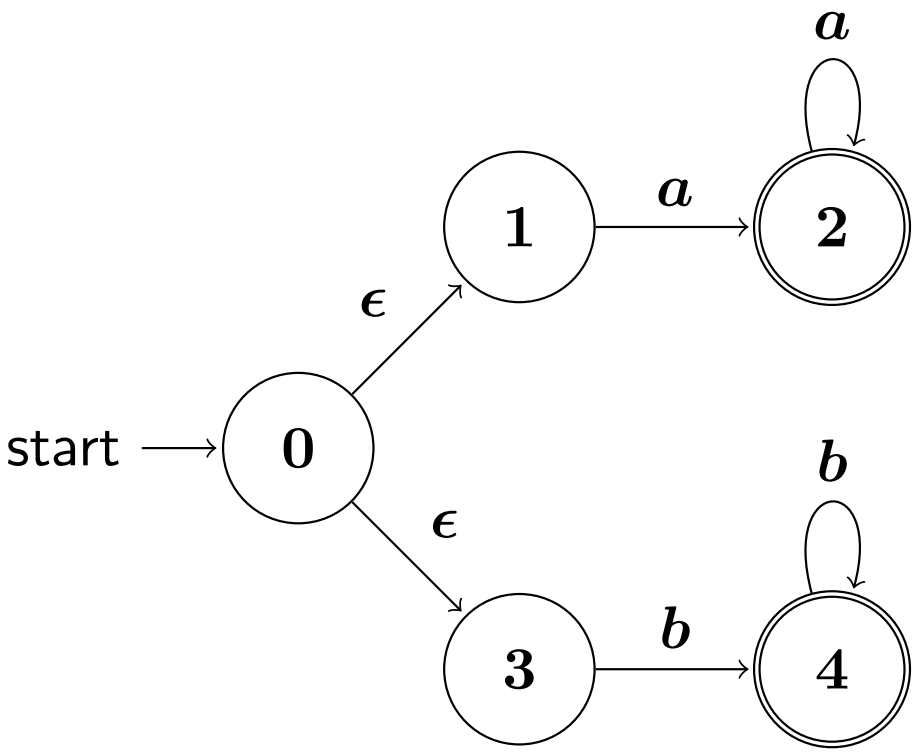
- 

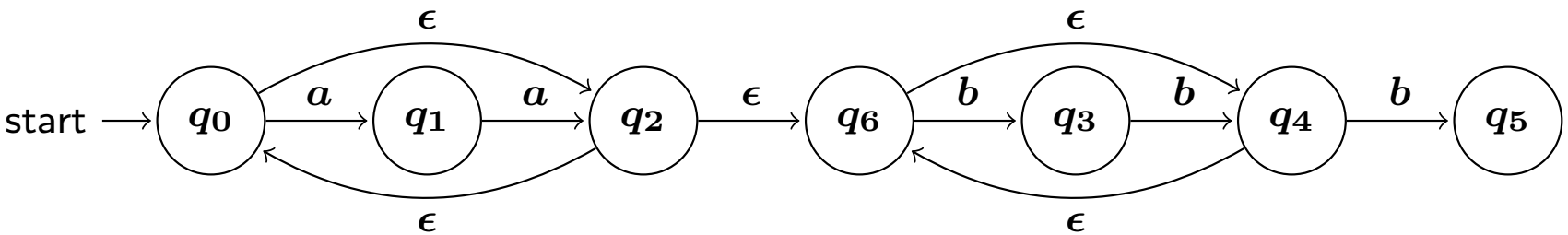
- 

Exercises



Exercises

- 

- 

Deterministic Finite Automata (DFA)

A DFA is a special case of an NFA, where

1. there are no moves on ϵ, and
2. for each state and input symbol, the next state is unique.

Deterministic Finite Automata (DFA)

A DFA is a special case of an NFA, where

1. there are no moves on ϵ, and
2. for each state and input symbol, the next state is unique.

Definition (DFA)

A *deterministic finite automaton* (or *DFA*) is defined by a tuple of five components:

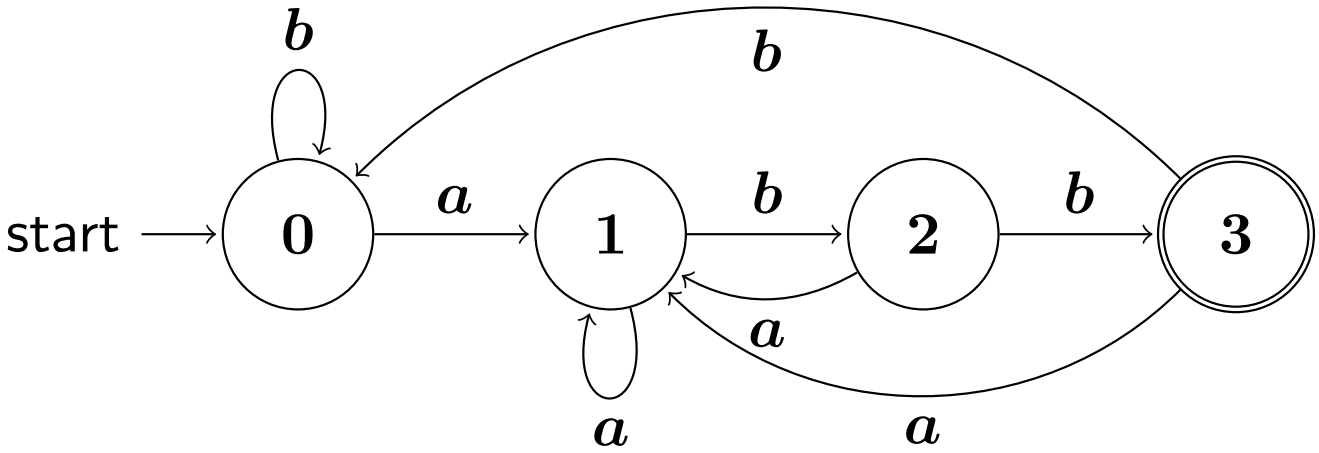
$$M = (Q, \Sigma, \delta, q_0, F)$$

where

- Q: a finite set of *states*
- Σ: a finite set of *input symbols* (or input alphabet)
- $\delta : Q \times \Sigma \to Q$: a *total* function called *transition function*
- $q_0 \in Q$: the *initial state*
- $F \subseteq Q$: a set of *final states*

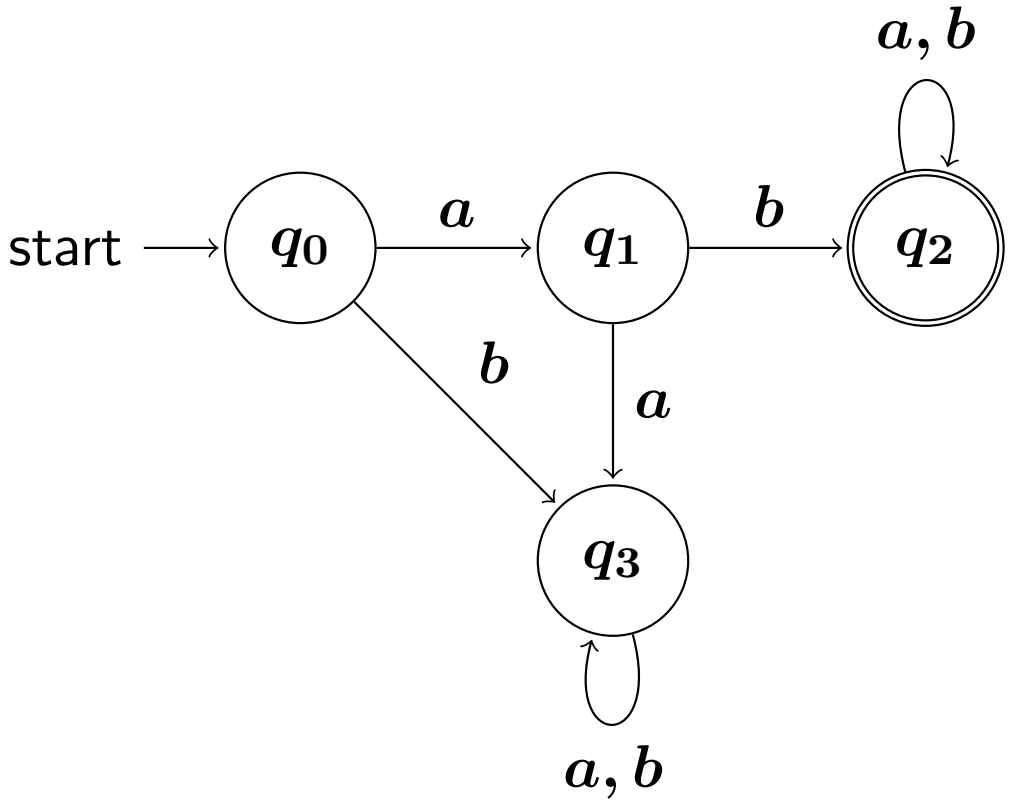
Example

A DFA that accepts $(a \mid b)^* abb$:



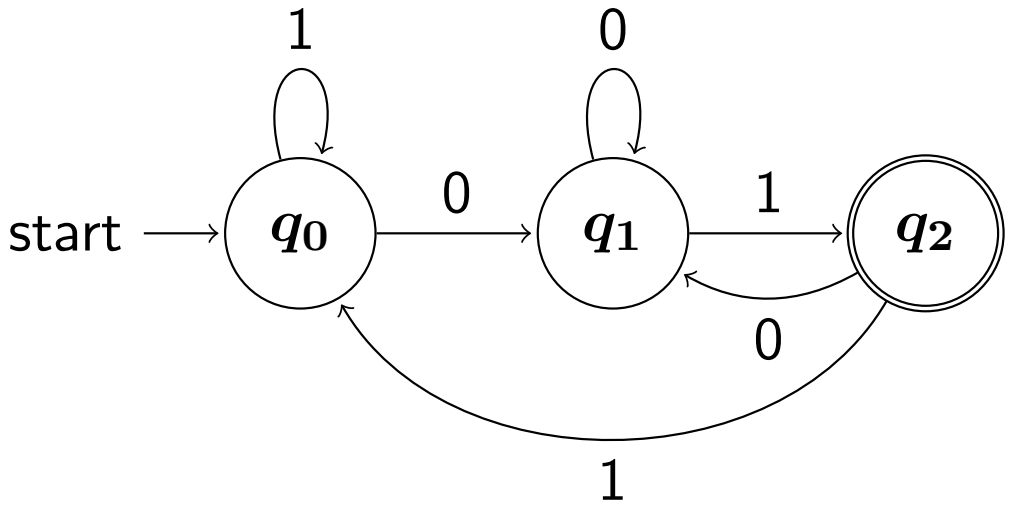
Exercise 1

What is the language of the DFA?



Exercise 2

What is the language of the DFA?



Summary

NFAs and DFAs are string recognizers.

- DFAs provide a concrete algorithm for recognizing strings.
- NFAs bridge the gap between REs and DFAs:
  - REs are descriptive but not executable.
  - DFAs are executable but not descriptive.
  - NFAs are in-between the REs and DFAs.