

CS420

Introduction to the Theory of Computation

Lecture 2: An algebra of automata

Tiago Cogumbreiro

Today we will introduce...

- Standard operations on languages (union, concatenation, exponentiation, kleene star)
- The nil automaton
- The empty automaton
- The character automaton
- The union automaton

■ Section 1.1

Formal definition of a Finite Automaton

Definition 1.5

A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

1. Q is a finite set called **states**
2. Σ is a finite set called **alphabet**
3. $\delta: Q \times \Sigma \rightarrow Q$ is the **transition function**
(δ takes a state and an alphabet and produces a state)
4. $q_0 \in Q$ is the **start state**
5. $F \subseteq Q$ is the set of **accepted states**

A formal definition is a precise mathematical language. In this example, item declares a name and possibly some constraint, e.g., $q_0 \in Q$ is saying that q_0 **must** be in set Q . These constraints are visible in the code in the form of assertions.

Formal declaration of our running example

Let the running example be the following finite automaton $M_{turnstile}$

$$(\{\text{Open}, \text{Close}\}, \{\text{Neither}, \text{Front}, \text{Rear}, \text{Both}\}, \delta, \text{Close}, \{\text{Close}\})$$

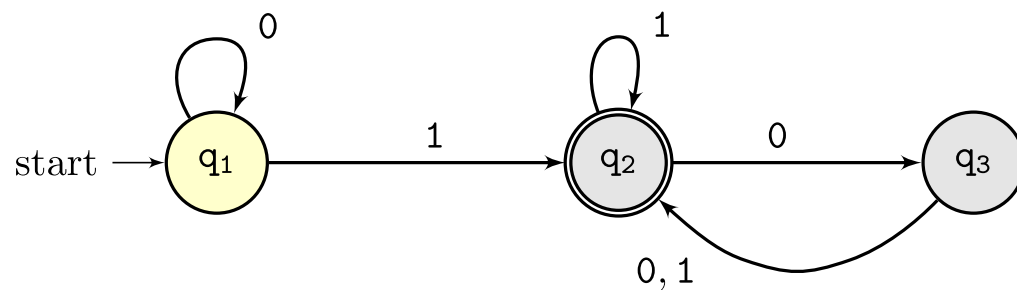
where

$$\begin{aligned}\delta(\text{Close}, \text{Front}) &= \text{Open} \\ \delta(\text{Open}, \text{Neither}) &= \text{Close} \\ \delta(q, i) &= q\end{aligned}$$

Facts

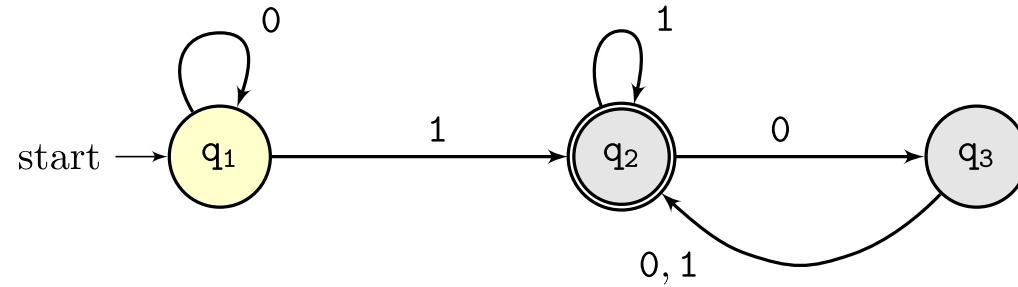
- $M_{turnstile}$ accepts [Front, Neither]
- $M_{turnstile}$ rejects [Rear, Front, Front]
- $M_{turnstile}$ accepts [Rear, Front, Rear, Neither, Rear]

Example



States?

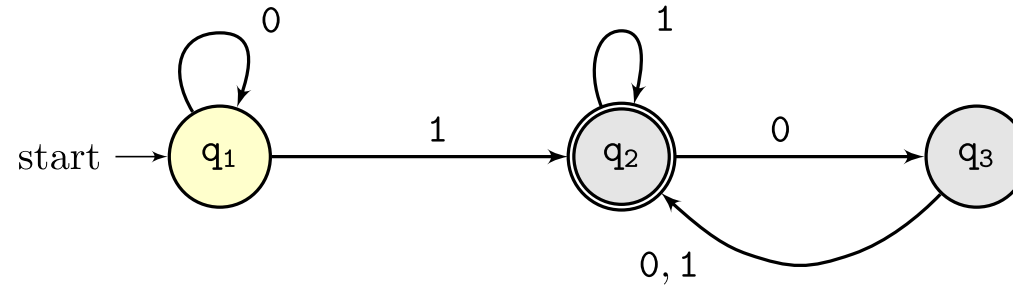
Example



States? $Q = \{q_1, q_2, q_3\}$

Alphabet?

Example

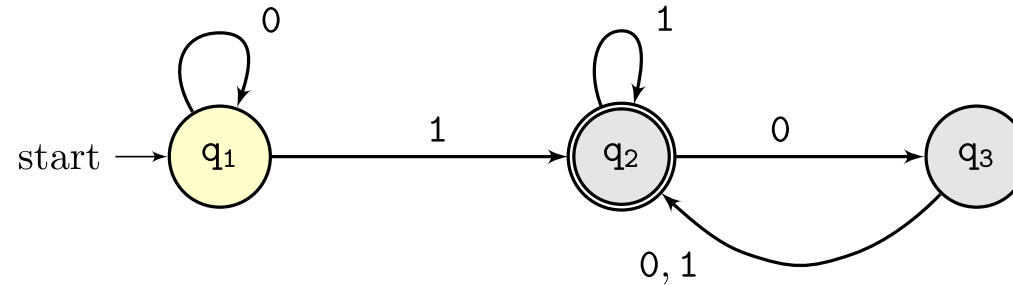


States? $Q = \{q_1, q_2, q_3\}$

Alphabet? $\Sigma = \{0, 1\}$

Transition table δ ?

Example



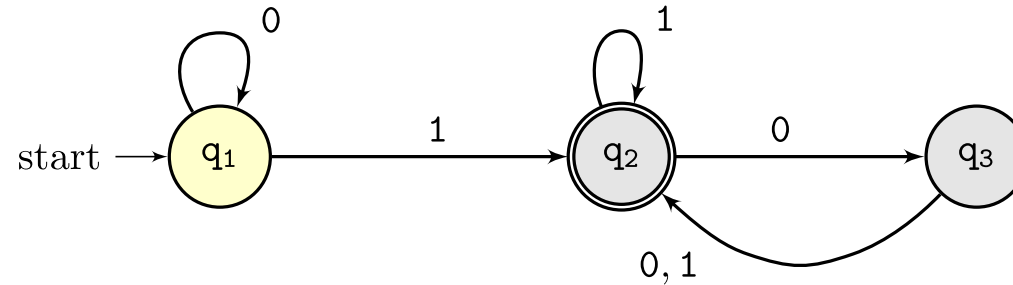
States? $Q = \{q_1, q_2, q_3\}$

Alphabet? $\Sigma = \{0, 1\}$

Transition table δ ?

(prev)	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

Example



States? $Q = \{q_1, q_2, q_3\}$

Alphabet? $\Sigma = \{0, 1\}$

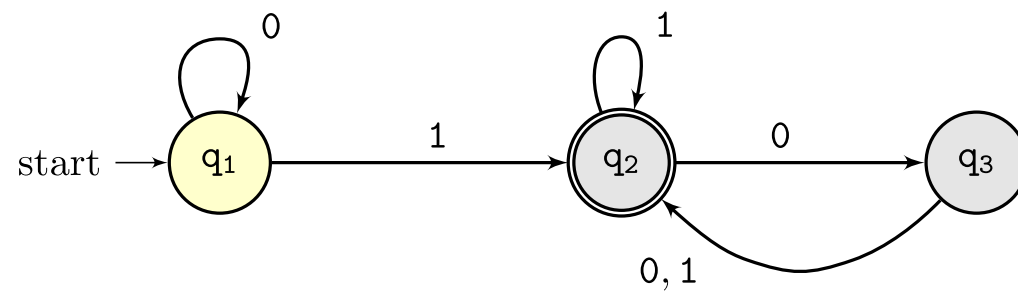
Transition table δ ?

(prev)	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

Finite Automaton:

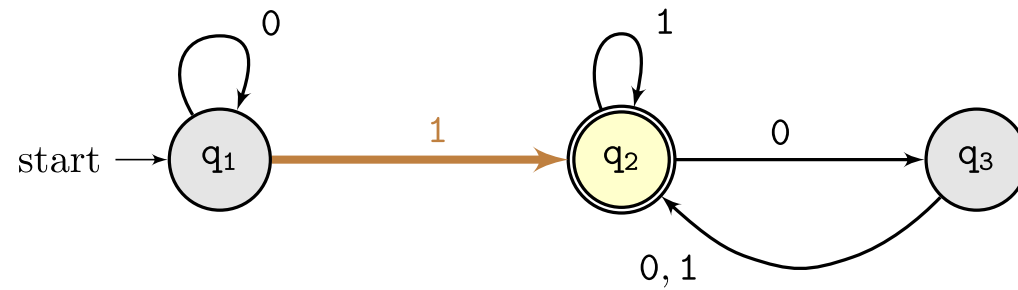
$(\{q_1, q_2, q_3\}, \{0, 1\}, q_1, \{q_2\})$

Example



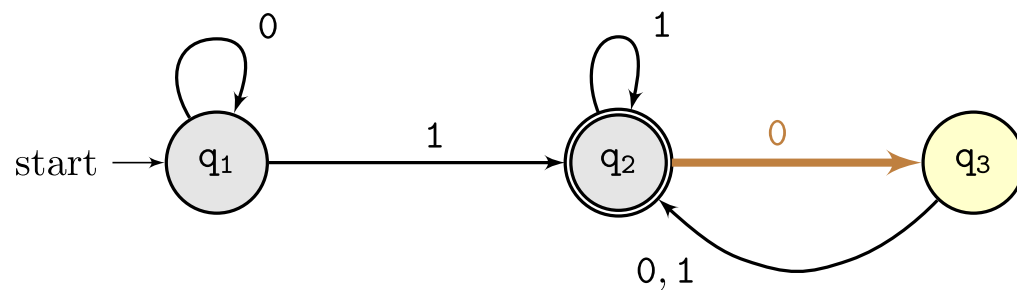
[1, 0, 1, 1]

Example



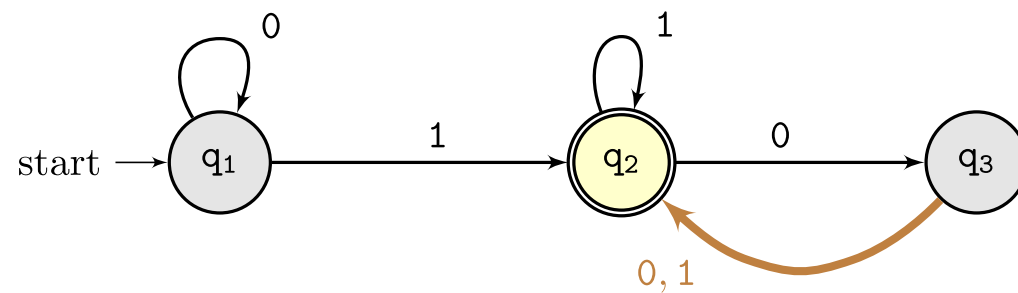
[**1**, 0, 1, 1]

Example



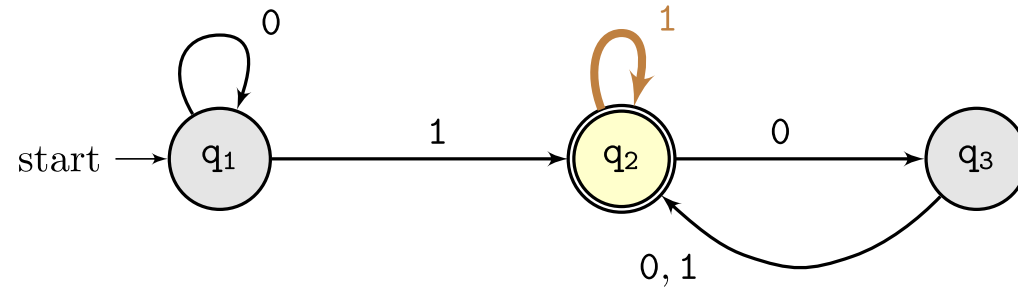
[1, **0**, 1, 1]

Example



[1, 0, **1**, 1]

Example



[1, 0, 1, **1**]

What are the set of inputs
accepted by this automaton?

What are the set of inputs
accepted by this automaton?

Answer: Strings terminating in 1

The language of a machine

Definition: language of a machine

1. We define $L(M)$ to be the set of all strings accepted by finite automaton M .
2. Let $A = L(M)$, we say that the finite automaton M **recognizes** the set of strings A .

The language of a machine

Definition: language of a machine

1. We define $L(M)$ to be the set of all strings accepted by finite automaton M .
2. Let $A = L(M)$, we say that the finite automaton M **recognizes** the set of strings A .

Notes

- The language is the set of all possible alphabet-sequences recognized by a finite automaton
- Since $L(M)$ is a **total** function, then the language recognized by a machine always exists and is unique
- A language may be empty
- We **cannot** write a program that returns the language of an arbitrary finite automaton. Why? **Because the language set may be infinite. How could a program return Σ^* ?**

■ A total function is defined for all inputs.

From a language to an automaton

Define a finite automaton that recognizes $\{[h,i], [h,o]\}$

- What is the alphabet of the finite automaton?

From a language to an automaton

Define a finite automaton that recognizes $\{[h,i], [h,o]\}$

- What is the alphabet of the finite automaton? $\{h, i, o\}$
- What are the states of the finite automaton?

From a language to an automaton

Define a finite automaton that recognizes $\{[h, i], [h, o]\}$

- What is the alphabet of the finite automaton? $\{h, i, o\}$
 - What are the states of the finite automaton?
1. We need an initial state, say q_1 that reads h and moves it to q_2

From a language to an automaton

Define a finite automaton that recognizes $\{[h, i], [h, o]\}$

- What is the alphabet of the finite automaton? $\{h, i, o\}$
 - What are the states of the finite automaton?
1. We need an initial state, say q_1 that reads h and moves it to q_2
 2. We need another state, say q_3 , after reading h it reads i (from q_2)

From a language to an automaton

Define a finite automaton that recognizes $\{[h, i], [h, o]\}$

- What is the alphabet of the finite automaton? $\{h, i, o\}$
 - What are the states of the finite automaton?
1. We need an initial state, say q_1 that reads h and moves it to q_2
 2. We need another state, say q_3 , after reading h it reads i (from q_2)
 3. We need another state, say q_4 , that reads o after having read h (from q_2)

From a language to an automaton

Define a finite automaton that recognizes $\{[h, i], [h, o]\}$

- What is the alphabet of the finite automaton? $\{h, i, o\}$
- What are the states of the finite automaton?
 1. We need an initial state, say q_1 that reads h and moves it to q_2
 2. We need another state, say q_3 , after reading h it reads i (from q_2)
 3. We need another state, say q_4 , that reads o after having read h (from q_2)
- What are the accepted states?

From a language to an automaton

Define a finite automaton that recognizes $\{[h, i], [h, o]\}$

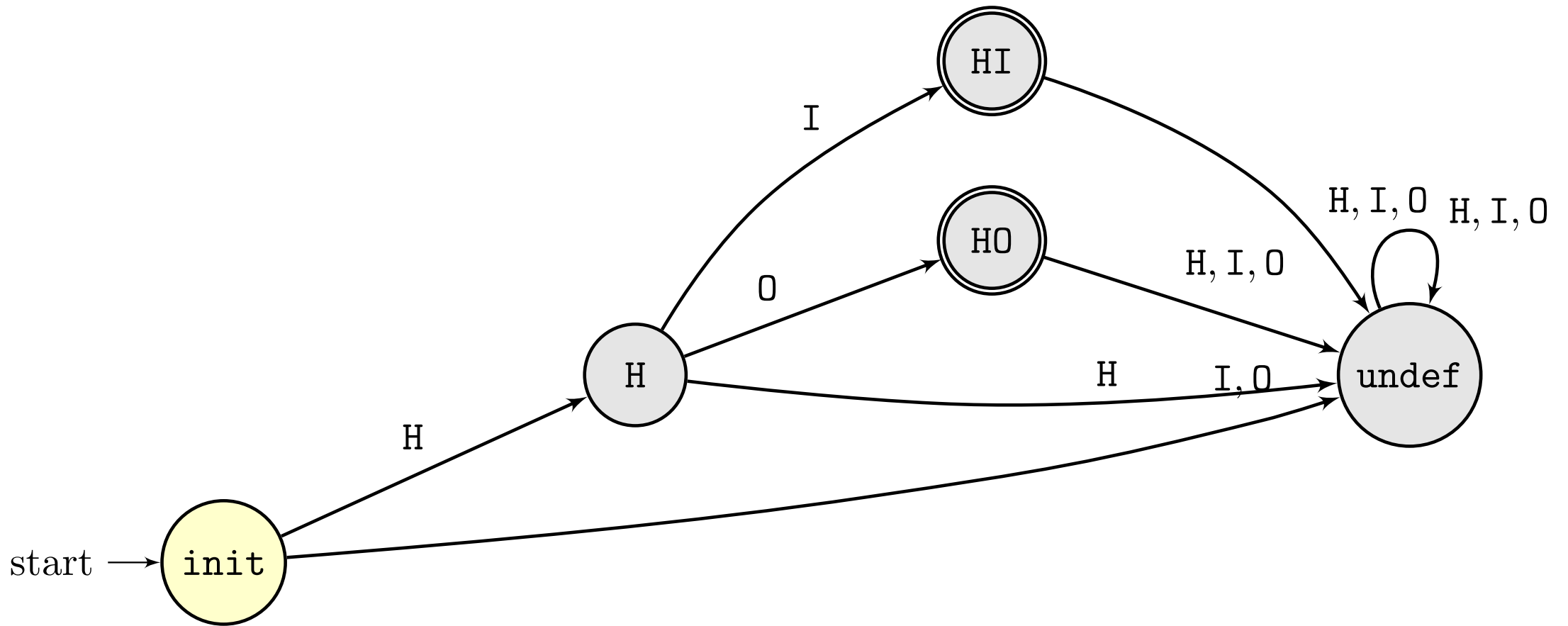
- What is the alphabet of the finite automaton? $\{h, i, o\}$
- What are the states of the finite automaton?
 1. We need an initial state, say q_1 that reads h and moves it to q_2
 2. We need another state, say q_3 , after reading h it reads i (from q_2)
 3. We need another state, say q_4 , that reads o after having read h (from q_2)
- What are the accepted states? $\{q_2, q_3\}$
- What happens if we read h in q_3 ?

From a language to an automaton

Define a finite automaton that recognizes $\{[h, i], [h, o]\}$

- What is the alphabet of the finite automaton? $\{h, i, o\}$
- What are the states of the finite automaton?
 1. We need an initial state, say q_1 that reads h and moves it to q_2
 2. We need another state, say q_3 , after reading h it reads i (from q_2)
 3. We need another state, say q_4 , that reads o after having read h (from q_2)
- What are the accepted states? $\{q_2, q_3\}$
- What happens if we read h in q_3 ? We need a "reject" state, say q_5 , that every unexpected letter takes us to.

Example



Standard operations on languages

Standard operations on languages

1. union (since a language is a set of strings, we can use the union of two languages)
2. concatenation
3. the Kleene star

Concatenation

- $L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1 \wedge w_2 \in L_2\}$

Examples

1. $\{a, aa\} \cdot \{b, bb\} =$

Concatenation

- $L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1 \wedge w_2 \in L_2\}$

Examples

1. $\{a, aa\} \cdot \{b, bb\} = \{ab, aab, abb, aabb\}$

2. $\{a, aa, aaa\} \cdot \{b, bb\} =$

Concatenation

- $L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1 \wedge w_2 \in L_2\}$

Examples

1. $\{a, aa\} \cdot \{b, bb\} = \{ab, aab, abb, aabb\}$
2. $\{a, aa, aaa\} \cdot \{b, bb\} = \{ab, abb, aab, aabb, aaab, aaabb\}$
3. $\{a, aa, aaa\} \cdot \emptyset =$

Concatenation

- $L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1 \wedge w_2 \in L_2\}$

Examples

1. $\{a, aa\} \cdot \{b, bb\} = \{ab, aab, abb, aabb\}$
2. $\{a, aa, aaa\} \cdot \{b, bb\} = \{ab, abb, aab, aabb, aaab, aaabb\}$
3. $\{a, aa, aaa\} \cdot \emptyset = \emptyset$
4. $\{a, aa, aaa\} \cdot \{\epsilon\} =$

Concatenation

- $L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1 \wedge w_2 \in L_2\}$

Examples

1. $\{a, aa\} \cdot \{b, bb\} = \{ab, aab, abb, aabb\}$
2. $\{a, aa, aaa\} \cdot \{b, bb\} = \{ab, abb, aab, aabb, aaab, aaabb\}$
3. $\{a, aa, aaa\} \cdot \emptyset = \emptyset$
4. $\{a, aa, aaa\} \cdot \{\epsilon\} = \{a, aa, aaa\}$

Exponentiation

- $L^0 = \{\epsilon\}$
- $L^{n+1} = L \cdot (L^n)$

Alternatively:

- $L^n = \{w^n \mid w \in L\}$

Examples

- $\{a, b\}^0 = \{\epsilon\}$
- $\{a, b\}^1 = \{a, b\}$
- $\{a, b\}^2 = \{aa, ab, ba, bb\}$

The Kleene star

- $L^* = \{w^* \mid w \in L\}$

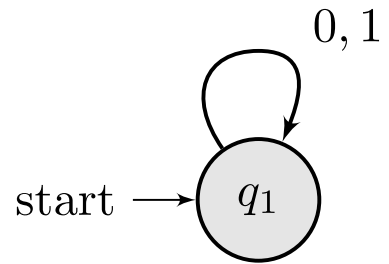
Examples

- $\{a\}^* = \{w \mid \text{words that only contain } a\}$
- $\{a, b\}^* = \{a, b\}^0 \cup \{a, b\}^1 \cup \{a, b\}^2 \cup \dots \cup \{a, b\}^n$

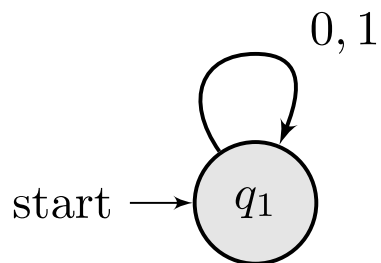
The nil automaton

$$L(\text{nil}_\Sigma) = \emptyset$$

The nil automaton



The nil automaton



- Note the absence of accepted states

```

def make_nil(alphabet):
    Q1 = "q_1"
    def transition(q, a):
        return q
    return DFA([Q1], alphabet, transition, Q1, lambda x: False)
  
```

The empty automaton

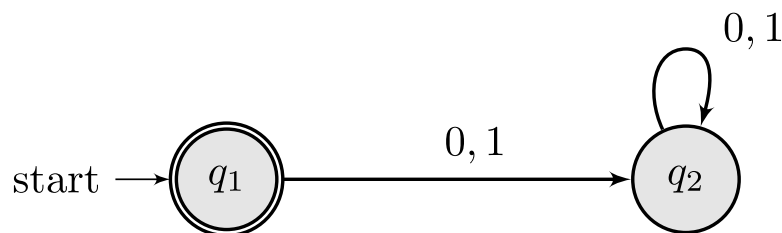
$$L(\text{empty}_\Sigma) = \{\epsilon\}$$

The empty automaton

Build an automaton that *only* accepts the empty string ϵ . You can imagine it to be akin to the zero of finite automata.

The empty automaton

Build an automaton that *only* accepts the empty string ϵ . You can imagine it to be akin to the zero of finite automata.

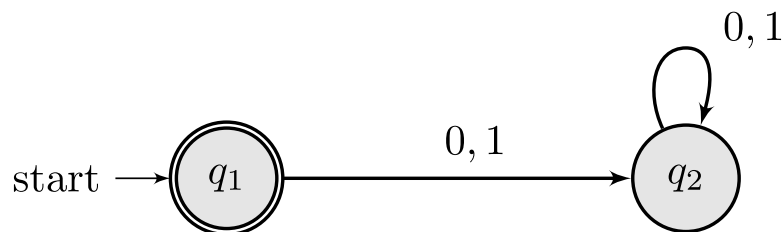


```

def make_empty(alphabet):
    return DFA(
        states = ["q_1", "q_2"],
        alphabet = alphabet,
        transition_func = lambda q, a: "q_2",
        start_state = "q_1",
        accepted_states = lambda x: x == "q_1")
  
```

The empty automaton

We define function `zero` that takes an alphabet Σ as input and outputs an automaton that only accepts the empty string whose alphabet is Σ .



$$\text{empty}_{\Sigma} = (\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_1\})$$

where

$$\delta(q, i) = q_2$$

The character-automaton

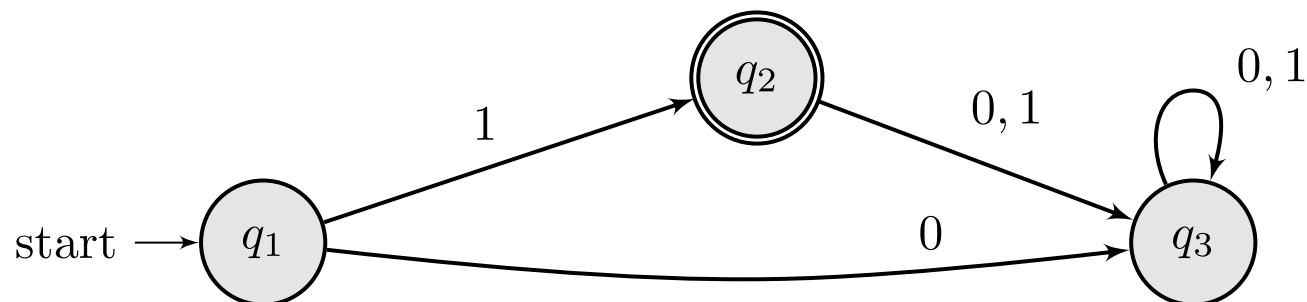
$$L(\text{char}(a)) = \{a\}$$

The character automaton

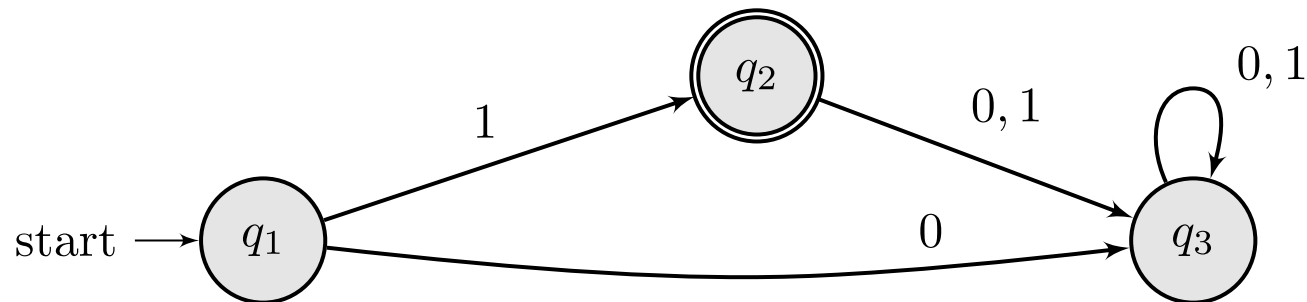
Given some character c , build an automaton that only accepts string $[c]$. This is akin to the numeral 1.

The character automaton

Given some character c , build an automaton that only accepts string $[c]$. This is akin to the numeral 1.



The character automaton



Implementation

```

def make_char(alphabet, char):
    return DFA(
        states=["q_1", "q_2", "q_3"],
        alphabet=alphabet,
        transition_func=lambda q, a: "q_2" if q == "q_1" and a == char else "q_3",
        start_state="q_1",
        accepted_states = lambda x: x == "q_2")
  
```

The character automaton

We define a function `char` that takes an alphabet Σ , a function `eq` that tests if two elements of Σ are equal, and a character $c \in \Sigma$ as input and outputs an automation that only accepts the string $[c]$ and whose alphabet is Σ .

$$\text{char}_{\Sigma}(c) = (\{q_1, q_2, q_3\}, \Sigma, \delta, q_1, \{q_2\})$$

where

2. $c \in \Sigma$
3. $\delta(q_1, c) = q_2$ (Note: This says that the arguments must be exactly q_1 and c .)
 $\delta(q, i) = q_3$ (otherwise)

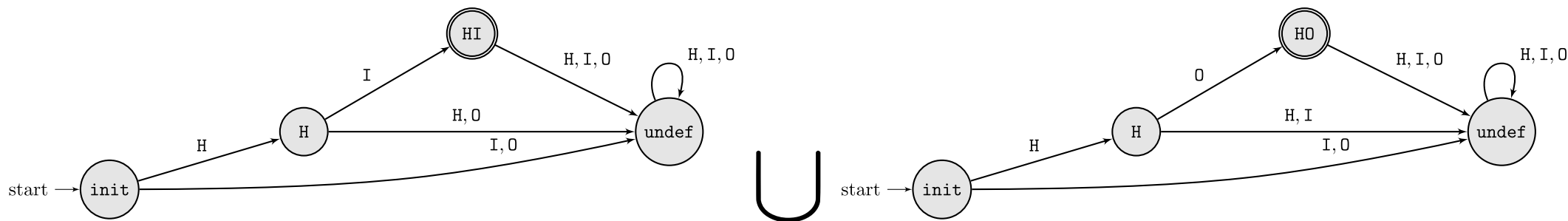
The union automaton

$$L(\text{union}(M_1, M_2)) = L(M_1) \cup L(M_2)$$

The union automaton

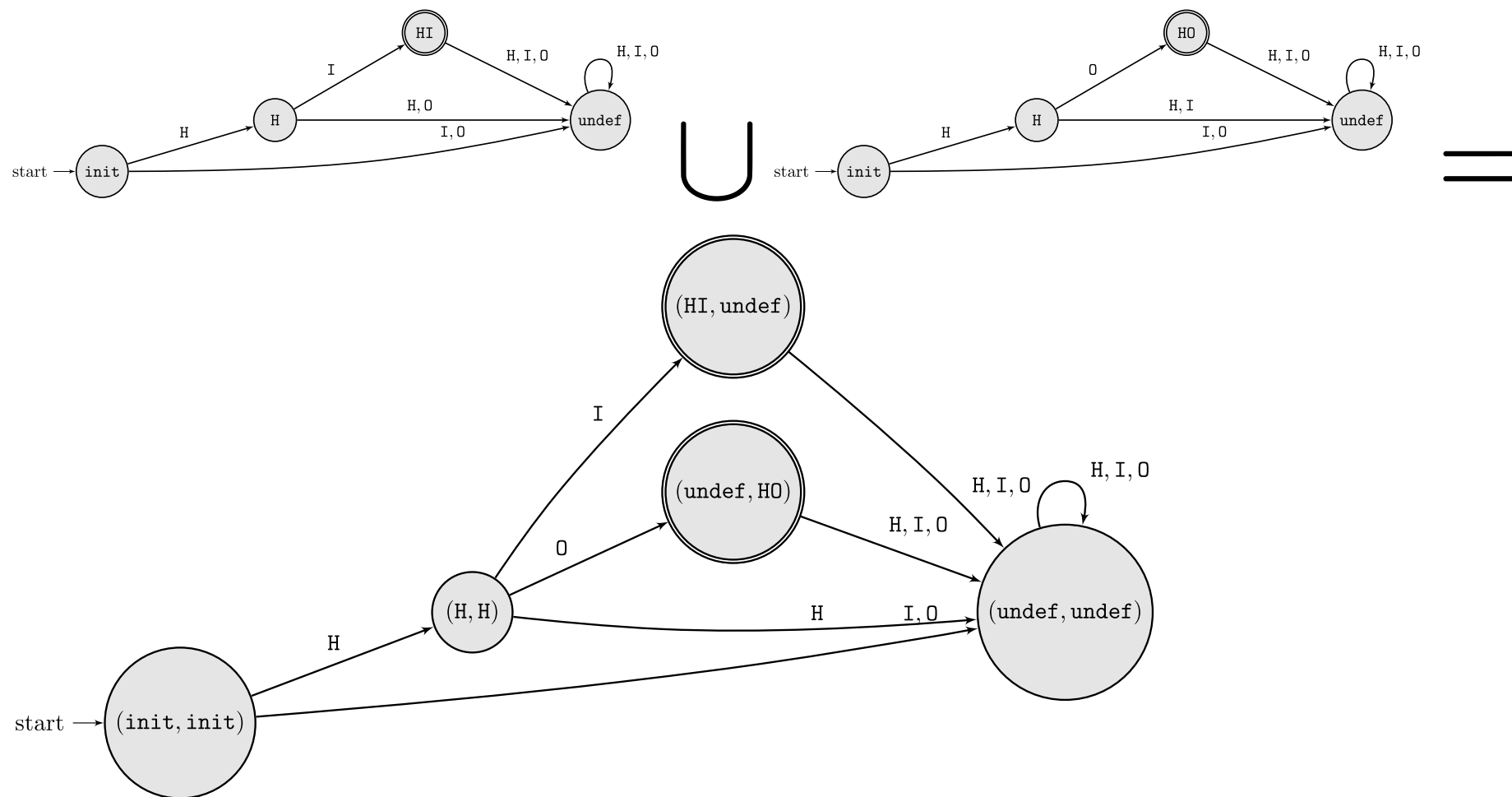
Formally, the union of two automata is defined as the union of the recognized languages.

Definition. Say M_1 recognizes A_1 and M_2 recognizes A_2 , then $M_1 \cup M_2$ accepts $A_1 \cup A_2$.



=?

Visually



Implementing the union of DFAs

```
def union(dfa1, dfa2):
    def transition(q, a):
        return (dfa1.transition_func(q[0], a), dfa2.transition_func(q[1], a))

    def is_final(q):
        return dfa1.accepted_states(q[0]) or dfa2.accepted_states(q[1])

    return DFA(
        states = set(product(dfa1.states, dfa2.states)),
        alphabet = set(dfa1.alphabet).union(dfa2.alphabet),
        transition_func = transition,
        start_state = (dfa1.start_state, dfa2.start_state),
        accepted_states = is_final
    )
```

The concatenation automaton

$$L(\text{concat}(M_1, M_2)) = L(M_1) \cdot L(M_2)$$

Building a concatenation
automation is non-trivial!

Building a concatenation automation is non-trivial!

Idea: new formalism!

(To be continued...)

Exercise

Draw an automaton that **recognizes** $\{w \mid w \text{ starts with } 10 \text{ or ends with } 01\}$.

Exercise

Draw an automaton that **recognizes** $\{w \mid w \text{ starts with } 10 \text{ or ends with } 01\}$.

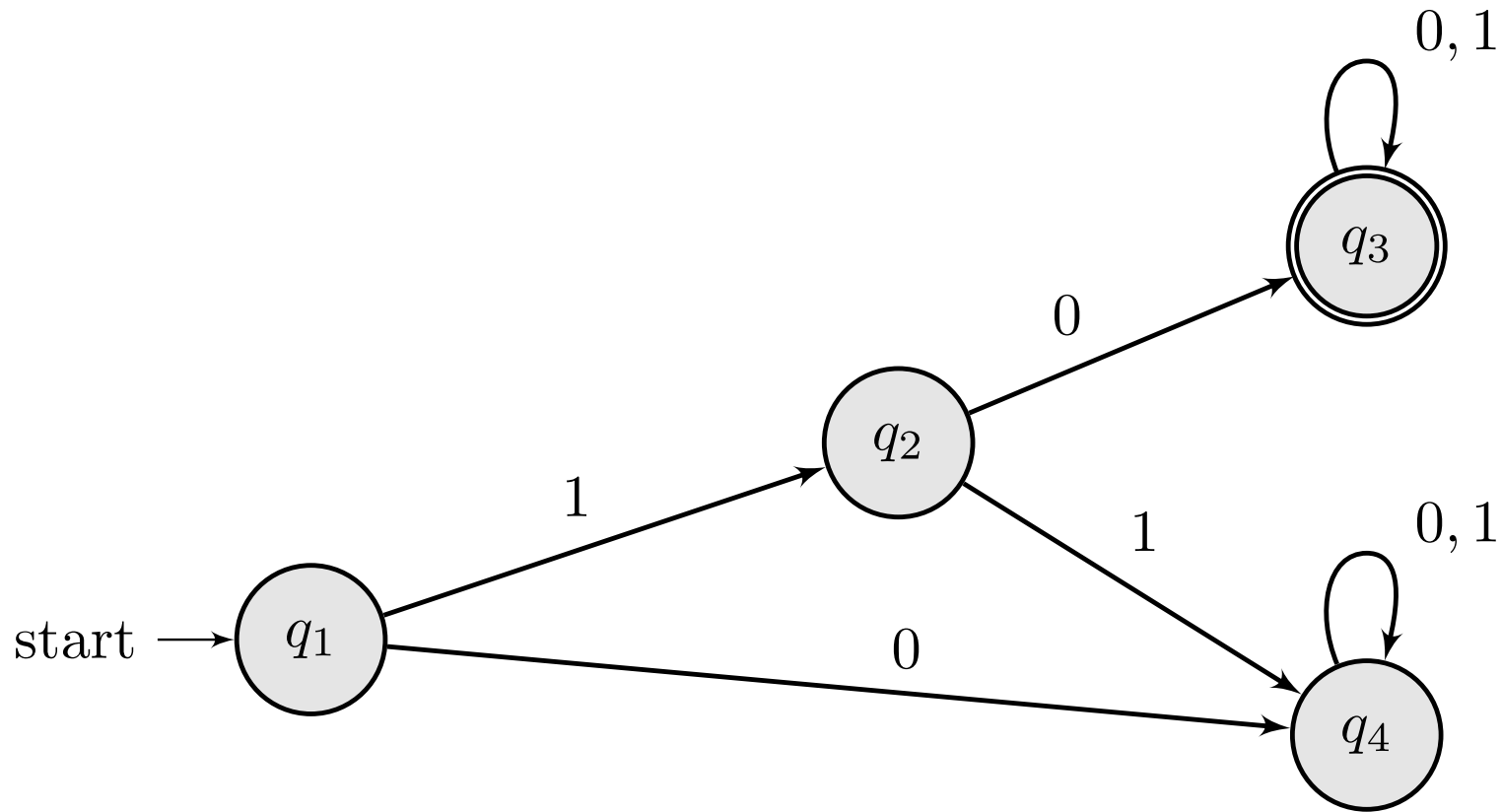
Idea: separate into two languages and then apply the union automaton

Exercise

Draw an automaton that **recognizes** $\{w \mid w \text{ starts with } 10\}$.

Exercise

Draw an automaton that **recognizes** $\{w \mid w \text{ starts with } 10\}$.

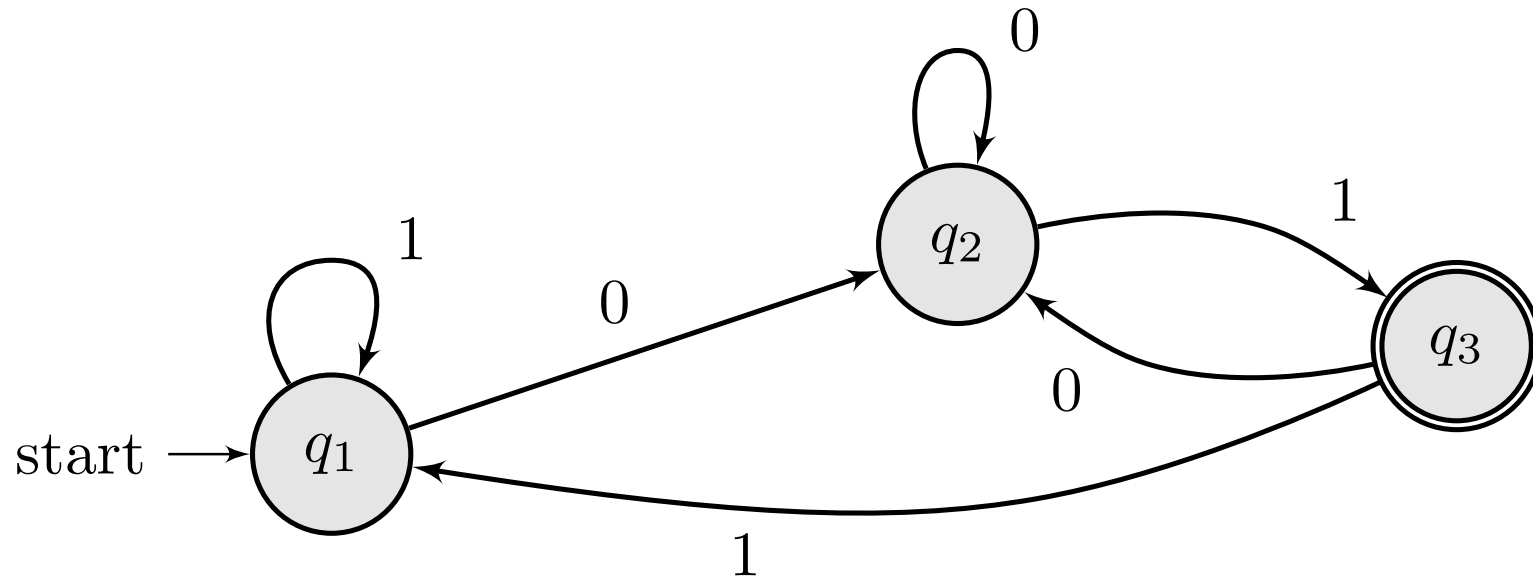


Exercise

Draw an automaton that **recognizes** $\{w \mid w \text{ ends with } 01\}$.

Exercise

Draw an automaton that **recognizes** $\{w \mid w \text{ ends with } 01\}$.



Exercise

Draw an automaton that **recognizes** $\{w \mid w \text{ starts with } 10 \text{ or ends with } 01\}$.

