CS420

Introduction to the Theory of Computation

Lecture 2: An algebra of automatons

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. \(\Sigma\) is a finite set called alphabet
3. \(\delta : Q \times \Sigma \rightarrow Q\) is the transition function
   \((\delta \text{ takes a state and an alphabet and produces a state})\)
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 $M_{\text{turnstile}}$

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

where

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

**Facts**

- $M_{\text{turnstile}}$ accepts [Front, Neither]
- $M_{\text{turnstile}}$ rejects [Rear, Front, Front]
- $M_{\text{turnstile}}$ 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 \( \delta \)?
Example

States? $Q = \{q_1, q_2, q_3\}$
Alphabet? $\Sigma = \{0, 1\}$
Transition table $\delta$?

<table>
<thead>
<tr>
<th>(prev)</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>$q_1$</td>
<td>$q_2$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$q_3$</td>
<td>$q_2$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$q_3$</td>
<td>$q_2$</td>
</tr>
</tbody>
</table>
Example

States? $Q = \{q_1, q_2, q_3\}$
Alphabet? $\Sigma = \{0, 1\}$
Transition table $\delta$?

<table>
<thead>
<tr>
<th>(prev)</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>$q_1$</td>
<td>$q_2$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$q_3$</td>
<td>$q_2$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$q_3$</td>
<td>$q_2$</td>
</tr>
</tbody>
</table>

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 $\Sigma^*$?

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 \(q_2\))
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?

\{h, i, o\}
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

![Diagram of automaton states and transitions]

- Start state: init
- Transitions:
  - From init to H: H
  - From init to HI: I
  - From H to HI: I
  - From H to HO: O
  - From HO to undef: H, I, O
  - From HI to HI: H, I, O

States:
- init
- H
- HI
- HO
- undef
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

- \( L_1 \cdot L_2 = \{ w_1 \cdot w_2 \mid w_1 \in L_1 \land w_2 \in L_2 \} \)

Examples

1. \( \{a, aa\} \cdot \{b, bb\} = \)
Concatenation

- \( L_1 \cdot L_2 = \{ w_1 \cdot w_2 \mid w_1 \in L_1 \land w_2 \in L_2 \} \)

Examples

1. \( \{a, aa\} \cdot \{b, bb\} = \{ab, aab, abb, aabb\} \)
2. \( \{a, aa, aaa\} \cdot \{b, bb\} = \)
Concatenation

- \( L_1 \cdot L_2 = \{ w_1 \cdot w_2 \mid w_1 \in L_1 \land w_2 \in L_2 \} \)

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

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

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. $\{a, aa, aaa\} \cdot \{\epsilon\} = \ldots$
Concatenation

- \( L_1 \cdot L_2 = \{ w_1 \cdot w_2 \mid w_1 \in L_1 \land w_2 \in L_2 \} \)

Examples

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

- $L^* = \{w^n \mid w \in L \land n \geq 0\}$

Examples

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

\[ L(\text{nil}_\Sigma) = \emptyset \]
The nil automaton

\[ q_1 \]

Start \[ \rightarrow \]

\(0, 1\)
The nil automaton

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

• Note the absence of accepted states
The empty automaton

\[ L(\text{empty}_\Sigma) = \{\epsilon\} \]
The empty automaton

Build an automaton that only accepts the empty string $\epsilon$. You can imagine it to be akin to the zero of finite automatons.
The empty automaton

Build an automaton that only accepts the empty string $\epsilon$. You can imagine it to be akin to the zero of finite automatons.

```python
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")
```

![Diagram of the empty automaton](image.png)
The empty automaton

We define function \texttt{zero} that takes an alphabet \( \Sigma \) as input and outputs an automation that only accepts the empty string whose alphabet is \( \Sigma \).

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

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

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

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 $A_1 \cup A_2$.

![Automaton Diagram]
Implementing the union of DFAs

```python
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
    )
```

Formalizing the union

The union operation is defined as \( \text{union}(M_1, M_2) = (Q_{1,2}, \Gamma_1, \delta_{1,2}, q_{1,2}, F_{1,2}) \) where

- \( M_1 = (Q_1, \Gamma_1, \delta_1, q_1, F_1) \)
- \( M_2 = (Q_2, \Gamma_2, \delta_2, q'_1, F_2) \)
- States: \( Q_{1,2} = Q_1 \times Q_2 \)
- Alphabet: \( \Gamma_1 = \Gamma_2 \)
- Transition: \( \delta_{1,2}(q, a) = (\delta_1(q_1, a), \delta_2(q_2, a)) \)
- Initial: \( q_{1,2} = (q_1, q'_1) \)
- Final: \( F_{1,2} = \{ q \mid q_1 \in F_1 \lor q_2 \in F_2 \} \)

Let notation \( q_1 = x \) be defined when \( q = (x, y) \). Let notation \( q_2 = y \) be defined when \( q = (x, y) \).
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 \( \{ w \mid w \text{ starts with 10 or ends with 01} \} \).
Exercise

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

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 $\{w \mid w$ starts with 10 or ends with 01 $\}$. 

![Automaton Diagram]

- Start state: $q_{1,1}$
- Final states: $q_{3,1}$, $q_{4,3}$