Automata

Matthias Braun

Today's Plan

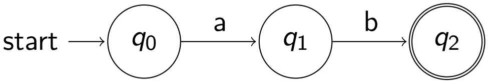
- What's an automaton (plural: "automata")?

Today's Plan

- What's an automaton (plural: "automata")?
- How do automata relate to grammars?

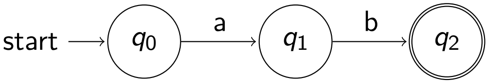
Automata

- An automaton consists of **states**



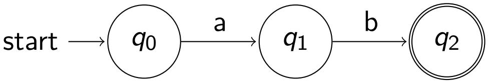
Automata

- An automaton consists of **states**
- An automaton can **transition** from state to state



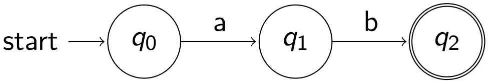
Automata

- An automaton consists of **states**
- An automaton can **transition** from state to state
- An automaton reads a string from left to right, one character at a time



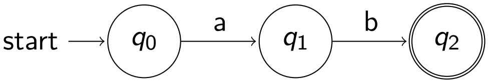
Automata

- An automaton consists of **states**
- An automaton can **transition** from state to state
- An automaton reads a string from left to right, one character at a time
- Each new character can put the automaton into a new state



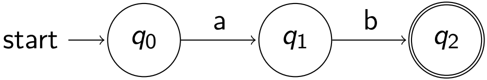
Automata

- An automaton consists of **states**
- An automaton can **transition** from state to state
- An automaton reads a string from left to right, one character at a time
- Each new character can put the automaton into a new state
- If the automaton is in a **final**[1] state after reading the last character, the string is valid



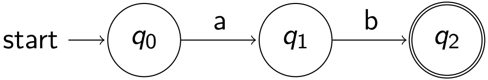
[1] a better word for "final" is probably "accepting"

Automata



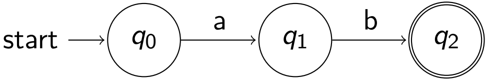
- This automaton has three states: q_0, q_1, q_2

Automata



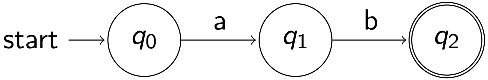
- This automaton has three states: q_0, q_1, q_2
- I has two transitions:

Automata



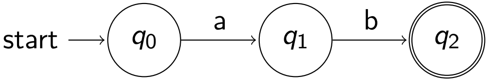
- This automaton has three states: q_0, q_1, q_2
- I has two transitions:
  1. $\delta(q_0, a) = q_1$

Automata



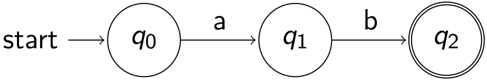
- This automaton has three states: q_0, q_1, q_2
- I has two transitions:
  1. $\delta(q_0, a) = q_1$
  2. $\delta(q_1, b) = q_2$

Automata



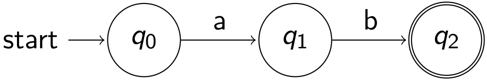
- This automaton has three states: q_0, q_1, q_2
- I has two transitions:
  1. $\delta(q_0, a) = q_1$
  2. $\delta(q_1, b) = q_2$
- That means:

Automata



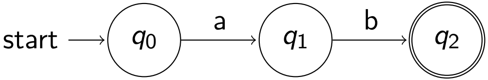
- This automaton has three states: q_0, q_1, q_2
- I has two transitions:
  1. $\delta(q_0, a) = q_1$
  2. $\delta(q_1, b) = q_2$
- That means:
  1. If the automaton is in state $\mathbf{q_0}$ and the current character of the input string is "**a**", the automaton goes into state $\mathbf{q_1}$

Automata



- This automaton has three states: q_0, q_1, q_2
- I has two transitions:
  1. $\delta(q_0, a) = q_1$
  2. $\delta(q_1, b) = q_2$
- That means:
  1. If the automaton is in state $\mathbf{q_0}$ and the current character of the input string is "**a**", the automaton goes into state $\mathbf{q_1}$
  2. If the automaton is in state $\mathbf{q_1}$ and the current character of the input string is "**b**", the automaton goes into state $\mathbf{q_2}$

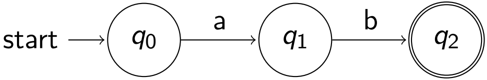
Automata



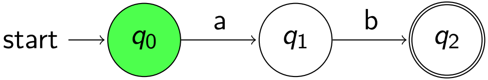
- This automaton has three states: q_0, q_1, q_2
- I has two transitions:
  1. $\delta(q_0, a) = q_1$
  2. $\delta(q_1, b) = q_2$
- That means:
  1. If the automaton is in state $\mathbf{q_0}$ and the current character of the input string is "**a**", the automaton goes into state $\mathbf{q_1}$
  2. If the automaton is in state $\mathbf{q_1}$ and the current character of the input string is "**b**", the automaton goes into state $\mathbf{q_2}$
- δ (lowercase delta) is called the automaton's **transition function**

Automata

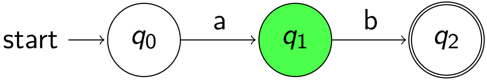
Validating a String



- Does our automaton accept the string “ab”?



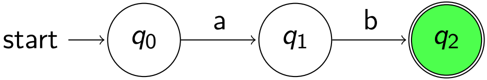
- Does our automaton accept the string “ab”?
- The start state is q_0



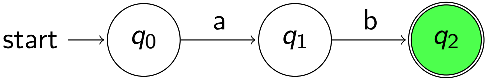
- Does our automaton accept the string "ab"?
- The start state is q_0
- The first character in the string is "a" $\rightarrow$ The automaton transitions from q_0 to q_1

Automata

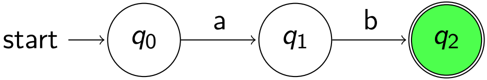
Validating a String



- Does our automaton accept the string “ab”?
- The start state is q_0
- The first character in the string is “a” $\rightarrow$ The automaton transitions from q_0 to q_1
- The next character is “b” $\rightarrow$ The automaton transitions from q_1 to q_2

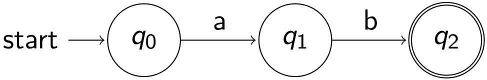


- Does our automaton accept the string “ab”?
- The start state is q_0
- The first character in the string is “a” $\rightarrow$ The automaton transitions from q_0 to q_1
- The next character is “b” $\rightarrow$ The automaton transitions from q_1 to q_2
- We’re done with the string: Is our automaton in a final state?



- Does our automaton accept the string "ab"?
- The start state is q_0
- The first character in the string is "a" $\rightarrow$ The automaton transitions from q_0 to q_1
- The next character is "b" $\rightarrow$ The automaton transitions from q_1 to q_2
- We're done with the string: Is our automaton in a final state?
- Yes. Therefore, this automaton accepts the string "ab"

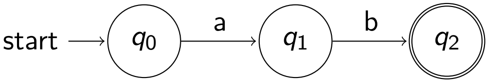
Automata and Grammars



- This automaton corresponds to this grammar in extended Backus-Naur form (EBNF):

```
a = "a";
b = "b";
sentence = a, b;
```

Automata and Grammars

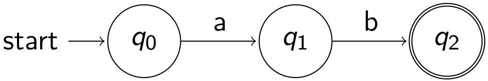


- This automaton corresponds to this grammar in extended Backus-Naur form (EBNF):

```
a = "a";
b = "b";
sentence = a, b;
```

- The language specified by this automaton and grammar is $\{ab\}$

Automata and Grammars



- This automaton corresponds to this grammar in extended Backus-Naur form (EBNF):

```
a = "a";
b = "b";
sentence = a, b;
```

- The language specified by this automaton and grammar is $\{ab\}$
- If there's an automaton like this for a language, that language is called **regular**

Automata and Grammars

- Let's consider this grammar:

```
a = "a";
b = "b";
sentence = a, b, rest;
rest = a, rest | b, rest | "";
```

Automata and Grammars

- Let's consider this grammar:

```
a = "a";
b = "b";
sentence = a, b, rest;
rest = a, rest | b, rest | "";
```

- This grammar creates sentences like “ab”, “abaa”, “abba”

Automata and Grammars

- Let's consider this grammar:

```
a = "a";
b = "b";
sentence = a, b, rest;
rest = a, rest | b, rest | "";
```

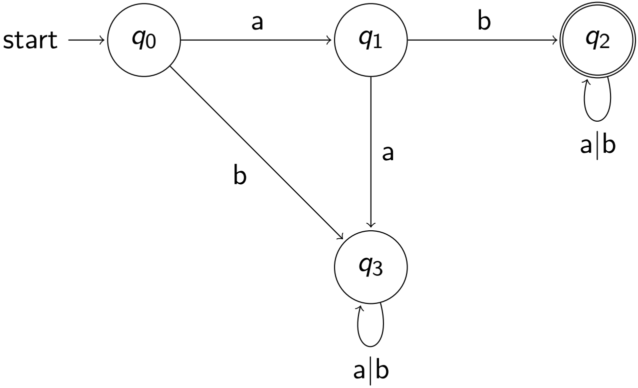
- This grammar creates sentences like "ab", "abaa", "abba"
- They need to start with "ab", the rest can be any letter from the alphabet

Automata and Grammars

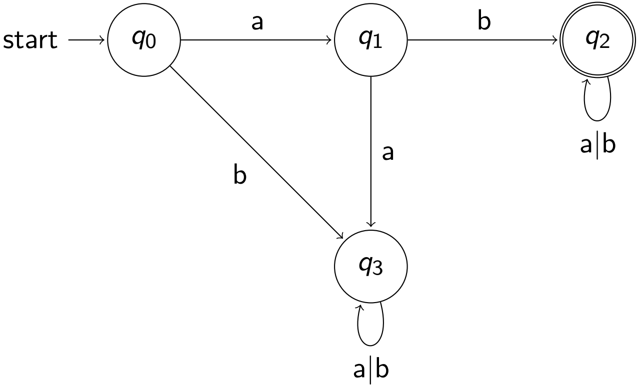
- Let's consider this grammar:

```
a = "a";
b = "b";
sentence = a, b, rest;
rest = a, rest | b, rest | "";
```

- This grammar creates sentences like "ab", "abaa", "abba"
- They need to start with "ab", the rest can be any letter from the alphabet
- What's the automaton for this grammar?



- q_2 is a final trap state: Once we're in it, the rest of the input string doesn't matter, the string will be accepted anyway



- q_2 is a final trap state: Once we're in it, the rest of the input string doesn't matter, the string will be accepted anyway
- q_3 is a non-final trap state: Once we're in it, the rest of the input string doesn't matter, the string will never be accepted

Run the Automaton

- Note the state for each character and determine if the automaton is in a final or non-final state after the last character:

Run the Automaton

- Note the state for each character and determine if the automaton is in a final or non-final state after the last character:
    1. “ababb”

Run the Automaton

- Note the state for each character and determine if the automaton is in a final or non-final state after the last character:
  1. “ababb”
  2. “aab”

Run the Automaton

- Note the state for each character and determine if the automaton is in a final or non-final state after the last character:
  1. "ababb"
  2. "aab"
  3. "bbab"

Run the Automaton

- Note the state for each character and determine if the automaton is in a final or non-final state after the last character:
  1. "ababb"
  2. "aab"
  3. "bbab"
  4. "aba"

Your Turn

1. Write down the transitions of the previous automaton in δ notation

Your Turn

1. Write down the transitions of the previous automaton in δ notation
2. Create an automaton from this grammar written in EBNF:

```
rest = "a" | "b" | "";
sentence = "X", rest;
```

Your Turn

1. Write down the transitions of the previous automaton in δ notation
2. Create an automaton from this grammar written in EBNF:
   ```
   rest = "a" | "b" | "";
   sentence = "X", rest;
   ```
3. Create an automaton from this grammar written in EBNF:
   ```
   rest = "a" | "b" | "";
   sentence = "X", rest | "Y", rest;
   ```

Your Turn

1. Write down the transitions of the previous automaton in δ notation
2. Create an automaton from this grammar written in EBNF:

```
rest = "a" | "b" | "";
sentence = "X", rest;
```

3. Create an automaton from this grammar written in EBNF:

```
rest = "a" | "b" | "";
sentence = "X", rest | "Y", rest;
```

4. Write down the alphabets of the previous two grammars/automata

Still Your Turn

Using the alphabet $\{0, 1\}$, construct different automata that accept:

1. strings of even length (think about whether the empty string has even length)
2. strings with a length of at least five
3. strings with an even number of 0s, for example "", "1", "010", "00"
4. strings with an even number of 0s and an odd number of 1s, for example "1", "010", "11001"

Using the alphabet $\{a, b\}$, construct different automata that accept:

1. strings containing exactly one a. For example: “a”, “abb”, “ba”.
2. strings with at least two a’s. For example “aa”, “abaa”, “baba”
3. strings with no more than two a’s
4. strings with exactly two a’s and at least one b. This automaton requires seven states. Hint: Name the states “oneA”, “twoAs”, “atLeastOneB”, etc.
5. Write down the transitions of the first three automata in δ notation

Create an automaton for each of these three EBNF grammars.
First, create a few example strings from the grammar.

1.
```
sentence = as, "b", as;
as = "" | "a", as;
```
2.
```
sentence = "a" | "a", rest, "a";
rest = "a", rest | "b", rest | "";
```
3.
```
s = "a" | "b" | "a", r, "a" | "b", r, "b";
r = "a", r | "b", r | "";
```

Using the alphabet $\{a, b\}$, construct different automata that accept the following strings and create grammars in EBNF that generate those strings:

1. all strings with exactly two a's.
2. all strings with at least two a's.
3. all strings with no more than three a's.
4. all strings with at least three a's.
5. all strings that start with a and end with b.
6. all strings with an even number of b's.