Deterministic Finite-State Automata

Deepak D'Souza

Department of Computer Science and Automation
Indian Institute of Science, Bangalore.

04 January 2024

Outline

Alphabets and Words

- An alphabet is a finite set of set of symbols or "letters". Eg. $A = \{a, b, c\}$ or $\Sigma = \{0, 1\}$.
- A string or word over an alphabet A is a finite sequence of letters from A. Eg. $aaba$ is string over $\{a, b, c\}$.
- Empty string denoted by ϵ.
- Set of all strings over A denoted by A^*.
  - What is the "size" or "cardinality" of A^*?

Alphabets and Words

- An alphabet is a finite set of set of symbols or "letters". Eg. $A = \{a, b, c\}$ or $\Sigma = \{0, 1\}$.
- A string or word over an alphabet A is a finite sequence of letters from A. Eg. $aaba$ is string over $\{a, b, c\}$.
- Empty string denoted by ϵ.
- Set of all strings over A denoted by A^*.
  - What is the "size" or "cardinality" of A^*?
  - Infinite but Countable.
  - Can enumerate in lexicographic order:

$$\epsilon,\ a,\ b,\ c,\ aa,\ ab,\ ba,\ bb,\ aaa,\ aab,\ \ldots$$

.

Concatenation of words

Operation of concatenation on words.

String u concatenated with (or followed by) string v: written

$$u \cdot v$$

or simply

$$uv.$$

Examples:

- $aabb \cdot aaa = aabbaaa$.
- $\epsilon \cdot aba = aba$.

Languages

- A *language* over an alphabet A is a set of strings over A. Eg. for $A = \{a, b, c\}$:
  - $L = \{abc, aaba\}$.
  - $L_1 = \{\epsilon, b, aa, bb, aab, aba, baa, bbb, \ldots\}$.
  - $L_2 = \{\}$.
  - $L_3 = \{\epsilon\}$.
- How many languages are there over a given alphabet A?

Languages

- A *language* over an alphabet A is a set of strings over A. Eg. for $A = \{a, b, c\}$:
  - $L = \{abc, aaba\}$.
  - $L_1 = \{\epsilon, b, aa, bb, aab, aba, baa, bbb, \ldots\}$.
  - $L_2 = \{\}$.
  - $L_3 = \{\epsilon\}$.
- How many languages are there over a given alphabet A?
  - Uncountably infinite
  - Use a diagonalization argument:

	ϵ	a	b	aa	ab	ba	bb	aaa	aab	aba	abb	bbb	$\cdots$
L_0	0	1	0	0	0	1	1	0	0	0	0	0	$\cdots$
L_1	0	0	0	0	0	0	0	0	0	0	0	0	$\cdots$
L_2	1	1	0	1	0	1	1	0	0	1	0	1	$\cdots$
L_3	0	0	0	0	0	0	0	0	0	0	0	0	$\cdots$
L_4	0	1	0	0	0	1	1	0	0	0	0	0	$\cdots$
L_5	1	1	0	1	0	1	1	0	0	1	0	1	$\cdots$
L_6	0	1	0	0	0	1	1	0	0	0	0	0	$\cdots$
L_7	0	0	0	0	0	0	1	0	0	0	1	0	$\cdots$
$\vdots$													

Operations on Languages

- Concatenation of languages:

$$L_1 \cdot L_2 = \{u \cdot v \mid u \in L_1,\ v \in L_2\}.$$

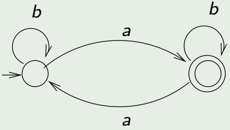
- Example:

$$\{abc, aaba\} \cdot \{\epsilon, a, bb\} = \{abc, aaba, abca, aabaa, abcbb, aababb\}.$$

- Union, Intersection, and Complement.

Example DFA 1

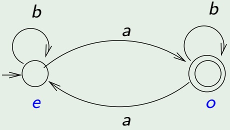
DFA for "Odd number of a's"



- How a DFA works.

Example DFA 1

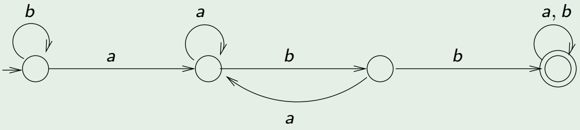
DFA for "Odd number of a's"



- How a DFA works.
- Each state represents a property of the input string read so far:
  - State e: Number of a's seen is even.
  - State o: Number of a's seen is odd.

Example DFA 2

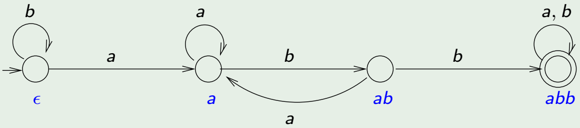
DFA for “Contains the substring *abb*”



Each state represents a property of the input string read so far:

Example DFA 2

DFA for "Contains the substring *abb*"



Each state represents a property of the input string read so far:

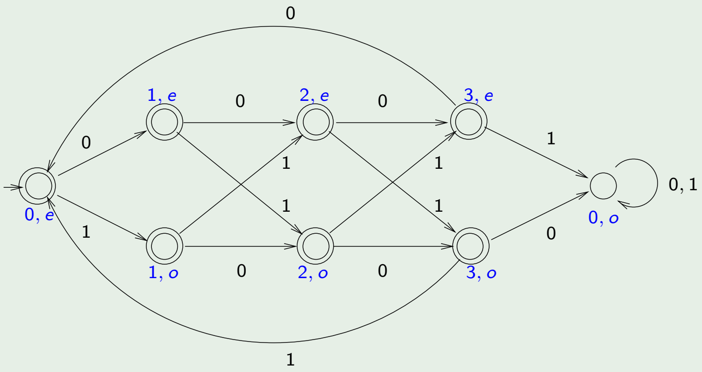
- State ϵ: Not seen *abb* and no suffix in *a* or *ab*.
- State *a*: Not seen *abb* and has suffix *a*.
- State *ab*: Not seen *abb* and has suffix *ab*.
- State *abb*: Seen *abb*.

Example DFA 3

Accept strings over $\{0, 1\}$ which have even parity in each length 4 block.

- Accept "0101 · 1010"
- Reject "0101 · 1011"

DFA for "Even parity checker"



Example DFA 4

Accept strings over $\{a, b, /, *\}$ which don't end inside a C-style comment.

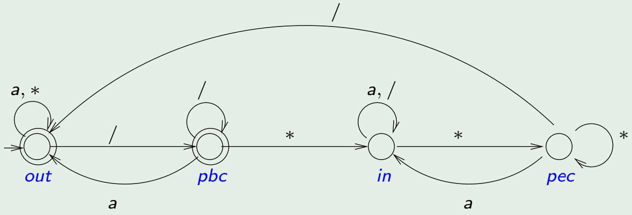
- Scan from left to right till first "/*" is encountered; from there to next "*/" is first comment; and so on.
- Accept "$ab/*aaa*/abba$" and "$ab/*aa/*aa*/bb*/$".
- Reject "$ab/*aaa*$" and "$ab/*aa/*aa*/bb/*a$".

Example DFA 4

Accept strings over $\{a, b, /, *\}$ which don't end inside a C-style comment.

- Scan from left to right till first "/*" is encountered; from there to next "*/" is first comment; and so on.
- Accept "$ab/*aaa*/abba$" and "$ab/*aa/*aa*/bb*/$".
- Reject "$ab/*aaa*$" and "$ab/*aa/*aa*/bb/*a$".

DFA for "C-comment tracker"



Good way to construct DFAs

Suppose we have to construct a DFA for a language L over an alphabet A.

1. Think of some properties of strings that you might want to keep track of. For example "number of a's in the string is even". Properties should be finite in number, say $p_1, \ldots, p_k$.
2. Identify an initial property (should hold on the empty string ϵ), say p_0.
3. Make sure you have a rule to update the property you are keeping track of for a string wa, based purely on the property of w and the last input a.
4. The properties should be such that either they imply membership in L or they imply non-membership in L.

Good way to construct DFAs

Use a slip of paper to keep track of the property of strings as you read the characters given to you.

1. Initially, write down p_o on the paper.
2. Ask for the letters of the string to be given to you one at a time.
3. For each letter you read, update the property written on your slip of paper (based only on what was written last and the current input symbol)
4. When the string ends, if the paper has a property which implies it should belong to L, accept; else reject.

Exercise

Exercise

Give a DFA that accepts strings over the alphabet $\{a, b\}$ containing an even number of a's and odd number of b's.

Definitions and notation: DFA

A *Deterministic Finite-State Automaton* $\mathcal{A}$ over an alphabet A is a structure of the form

$$(Q, s, \delta, F)$$

where

- Q is a finite set of states
- $s \in Q$ is the start state
- $\delta : (Q \times A) \to Q$ is the transition function.
- $F \subseteq Q$ is the set of final states.

Definitions and notation: DFA

A *Deterministic Finite-State Automaton* $\mathcal{A}$ over an alphabet A is a structure of the form

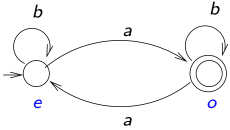
$$(Q, s, \delta, F)$$

where

- Q is a finite set of states
- $s \in Q$ is the start state
- $\delta : (Q \times A) \to Q$ is the transition function.
- $F \subseteq Q$ is the set of final states.

Example of "Odd a's" DFA:
Here: $A = \{a, b\}$, $Q = \{e, o\}$, $s = e$,
$F = \{o\}$, and δ is given by:

$$\begin{aligned} \delta(e, a) &= o, \\ \delta(e, b) &= e, \\ \delta(o, a) &= e, \\ \delta(o, b) &= o. \end{aligned}$$



Definitions and notation: Language accepted by a DFA

- $\widehat{\delta}$ tells us how the DFA $\mathcal{A}$ behaves on a given word u.
- Define $\widehat{\delta} : Q \times A^* \to Q$ as
  - $\widehat{\delta}(q, \epsilon) = q$
  - $\widehat{\delta}(q, w \cdot a) = \delta(\widehat{\delta}(q, w), a)$.
- Language *accepted* by $\mathcal{A}$, denoted $L(\mathcal{A})$, is defined as:

$$L(\mathcal{A}) = \{w \in A^* \mid \widehat{\delta}(s, w) \in F\}.$$

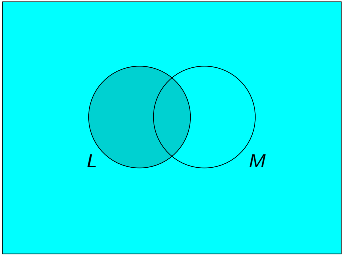
- Eg. For $\mathcal{A}$ = DFA for "Odd a's",

$$L(\mathcal{A}) = \{a, ab, ba, aaa, abb, bab, bba, \ldots\}.$$

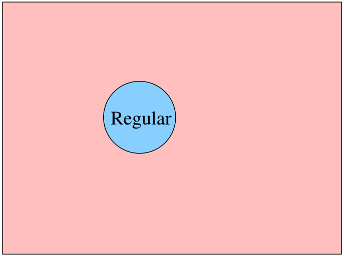
Regular Languages

- A language $L \subseteq A^*$ is called *regular* if there is a DFA $\mathcal{A}$ over A such that $L(\mathcal{A}) = L$.
- Examples of regular languages: "Odd a's", "strings that don't end inside a C-style comment", $\{\}$, any finite language.

All strings over A



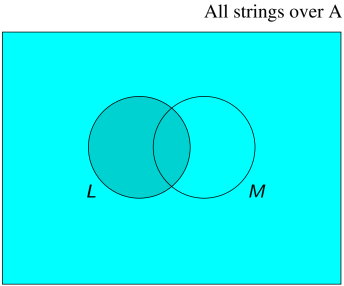
All languages over A

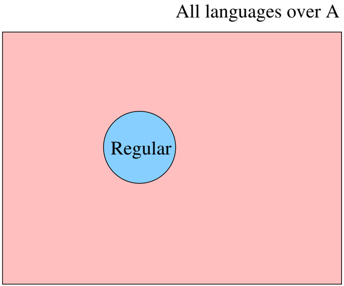


- Are there non-regular languages?

Regular Languages

- A language $L \subseteq A^*$ is called *regular* if there is a DFA $\mathcal{A}$ over A such that $L(\mathcal{A}) = L$.
- Examples of regular languages: "Odd a's", "strings that don't end inside a C-style comment", $\{\}$, any finite language.





- Are there non-regular languages?
  - Yes, uncountably many, since Reg is only countable while class of all languages is uncountable.