Tianjin International Engineering Institute

Formal Languages and Automata

Lesson 7: Properties of regular languages

Marc Gaetano

Edition 2018

Operations that preserve regularity

- In the last lecture we saw three operations that preserve regularity:
  - Union: If L, L' are regular languages, so is $L \cup L$'
  - Concatenation: If L, L' are regular languages, so is LL'
  - Star: If L is a regular language, so is $L*$

- Exercise: If L is regular, is L^4 also regular?
- Answer: Yes, because

$$L^4 = ((LL)L)L$$

Example

- The language L of strings that end in 101 is regular

$$(0+1)^*101$$

- How about the language $\overline{L}$ of strings that do not end in 101?

Example

- **Hint:** A string does not end in 101 if and only if it ends in one of the following patterns:

$$000,\ 001,\ 010,\ 011,\ 100,\ 110,\ 111$$

  (or it has length 0, 1, or 2)

- So $\overline{L}$ can be described by the regular expression

$$(0+1)^*(000+001+010+010+100+110+111) \\ + \varepsilon + (0+1) + (0+1)(0+1)$$

Complement

- The complement $\overline{L}$ of a language L is the set of all strings (over Σ) that are not in L

- Examples ($\Sigma = \{0, 1\}$)
  - $L_1 =$ all strings that end in 101
  - $\overline{L_1} =$ all strings that do not end in 101
    $=$ all strings end in $000, \ldots, 111$ or have length 0, 1, or 2
  - $L_2 = 1^* = \{\varepsilon, 1, 11, 111, \ldots\}$
  - $\overline{L_2} =$ all strings that contain at least one 0
    $= (0 + 1)^*0(0 + 1)^*$

Closure under complement

- If L is a regular language, is $\overline{L}$ also regular?
- Previous examples indicate answer should be yes

- Theorem

> If L is a regular language, so is $\overline{L}$.

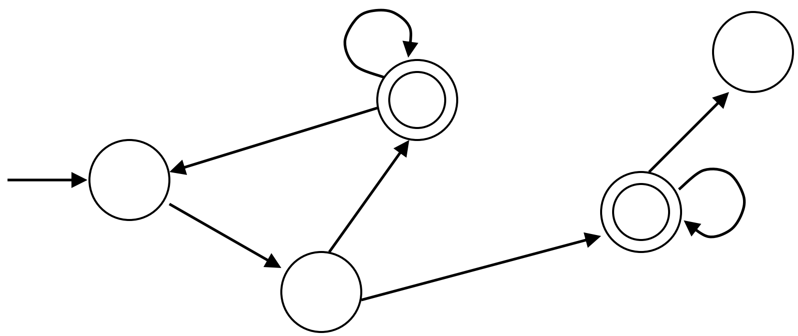
Proof of closure under complement

- To prove this in general, we can use any of the equivalent definitions for regular languages:

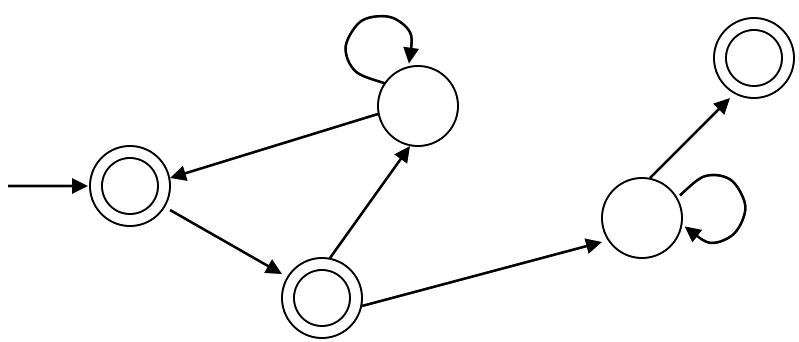
- In this proof DFA definition will be most convenient
  - We will assume L is accepted by a DFA, and show the same for $\overline{L}$

Proof of closure under complement

- Suppose L is regular, then it is accepted by a DFA M



- Now consider the DFA M' with the accepting and rejecting states of M reversed



Proof of closure under complement

- Now for every input $x \in \Sigma^*$:

M accepts x

$\Updownarrow$

After processing x, M ends in an accepting state

$\Updownarrow$

After processing x, M' ends in an rejecting state

$\Updownarrow$

M' rejects x

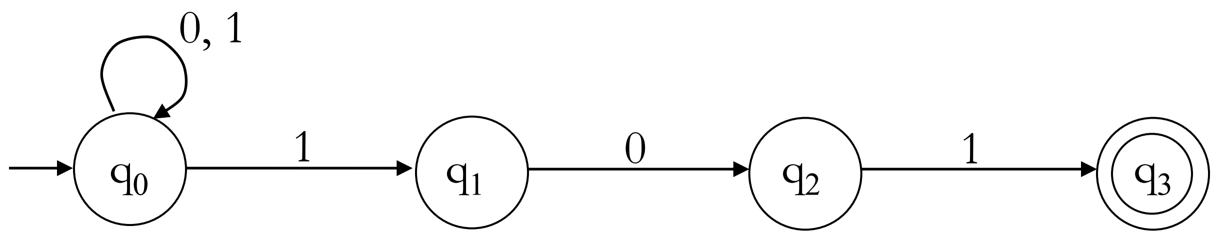
Language of M' is $\overline{L}$

$\Downarrow$

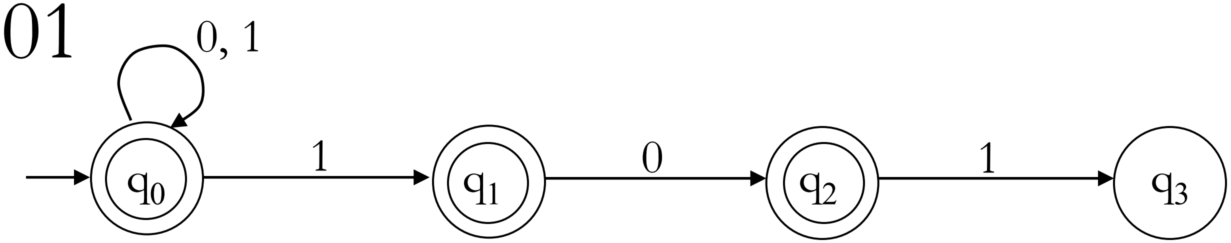
$\overline{L}$ is regular

A warning

- NFA for language of strings ending in 101



- Give NFA that accepts strings that do not end in 101



wrong!

Intersection

- The intersection $L \cap L'$ is the set of strings that are in both L and L'

- Examples:

  $L = (0 + 1)^*111 \qquad L' = 1^* \qquad L \cap L' = ?$

  $L = (0 + 1)^*101 \qquad L' = 1^* \qquad L \cap L' = ?$

- If L, L' are regular, is $L \cap L'$ also regular?

Closure under intersection

- Theorem

If L and L' are regular languages, so is $L \cap L'$.

- Proof: L regular $\Downarrow$ $\overline{L}$ regular; L' regular $\Downarrow$ $\overline{L'}$ regular

$$\underbrace{\overline{L} \text{ regular} \qquad \overline{L'} \text{ regular}}_{\overline{L} \cup \overline{L'} \text{ regular}}$$

But $\overline{L} \cup \overline{L'} = \overline{L \cap L'}$

$\overline{L \cap L'}$ regular

$\Downarrow$

$L \cap L'$ regular.

Reversal

- The reversal w^R of a string w is w written backwards

$$w = \text{cave} \qquad\qquad w^R = \text{evac}$$

- The reversal L^R of a language L is the language obtained by reversing all its strings

$$L = \{\text{push}, \text{pop}\} \qquad\qquad L^R = \{\text{hsup}, \text{pop}\}$$

Reversal of regular languages

- $L =$ all strings that end in 101 is regular

$$(0+1)^*101$$

- How about L^R?
- This is the language of all strings beginning in 101
- Yes, because it is represented by

$$101(0+1)^*$$

Closure under reversal

- Theorem

If L is a regular language, so is L^R.

- Proof

  – We will use the representation of regular languages by regular expressions

Proof of closure under reversal

- If L is regular, then there is a regular expression E that describes it
- We will give a systematic way of reversing E
- Recall that a regular expression can be of the following types:
  - Special expressions $\emptyset$ and ε
  - Alphabet symbols a, b, …
  - The union, concatenation, or star of simpler expressions
- In each of these cases we show how to do a reversal

Proof of closure under reversal

regular expression E	reversal E^R
$\emptyset$	$\emptyset$
ε	ε
a (alphabet symbol)	a
$E_1 + E_2$	$E_1{}^R + E_2{}^R$
$E_1 E_2$	$E_2{}^R E_1{}^R$
$E_1 *$	$(E_1{}^R)*$

Applications: text search

- Text search: `grep`
  - Looks for occurrences of a regular expression in a file
- Syntax of regular expressions in `grep`
  - `[atr]` means the set {`a`, `t`, `r`}
  - `[b-e]` means {`b`, `c`, `d`, `e`} (in ASCII/Unicode ordering)
  - | means + (union), * means star
  - `?` means "zero or one": `R?` is ε + R
  - + means "one or more": `R+` is RR*
  - {n} means "n copies of": `R{5}` is RRRRR

Regular expressions in grep

- Say we have a file `w.txt`

```
1/1/08   14C   rain\n
1/2/08   17C   sunny\n
1/3/08   18C   sunny\n
               …
```

- Want to know all sunny days

```
> grep 'sunny' w.txt
```

- Any cloudy days in April '08?

```
> grep '4/[0-9]*/08 ([0-9]|C| )* cloudy' w.txt
```

- Any consecutive string of 7 sunny days?

```
> grep '(([0-9]|/|C| )* sunny\n){7}' w.txt
```

Implementation of grep

- One way to implement `grep` is to convert the regular expression into a DFA and then "run" the DFA on this text file

- Two issues that arise:
  - grep looks for patterns inside text, while the DFA processes the input as a whole
  - DFA only produces "yes/no" answer, while `grep` outputs lines that contain given pattern
- How would you resolve these issues?