CS21 Decidability and Tractability

Lecture 4
January 11, 2023

1

Outline

- FA, Regular Expression equivalence
- Non-regular languages: Pumping Lemma

** first problem set out today

2

Regular expressions and FA

- **Theorem**: a language L is recognized by a FA if and only if L is described by a regular expression.

Must prove *two* directions:

(⇒) L is recognized by a FA implies L is described by a regular expression

(⇐) L is described by a regular expression implies L is recognized by a FA.

3

Regular expressions and FA

(⇒) L is recognized by a FA implies L is described by a regular expression

Proof: given FA M that recognizes L, we will

1. build an equivalent machine "Generalized Nondeterministic Finite Automaton" (GNFA)
2. convert the GNFA into a regular expression

4

Regular expressions and FA

- GNFA definition:
  - it is a NFA, but may have regular expressions labeling its transitions
  - GNFA accepts string $w \in \Sigma^*$ if can be written

$$w = w_1 w_2 w_3 \ldots w_k$$

where each $w_i \in \Sigma^*$, and there is a path from the start state to an accept state in which the i^{th} transition traversed is labeled with R for which $w_i \in L(R)$

5

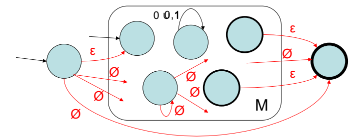
Regular expressions and FA

- Recall step 1: build an equivalent GNFA
- Our FA M *is* a GNFA.
- We will require "normal form" for GNFA to make the proof easier:
  - *single* accept state q_{accept} that has all possible incoming arrows
  - every state has all possible outgoing arrows; exception: start state q_0 has no self-loop

6

Regular expressions and FA

- converting our FA M into GNFA in normal form:



7

Regular expressions and FA

- On to step 2: convert the GNFA into a regular expression

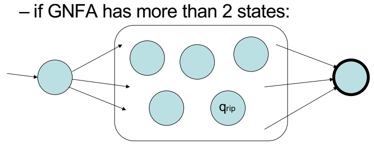
– if normal-form GNFA has two states:

the regular expression R labeling the single transition describes the language recognized by the GNFA

8

Regular expressions and FA

– if GNFA has more than 2 states:



– select one “q_{rip}”; delete it; repair transitions so that machine still recognizes same language.

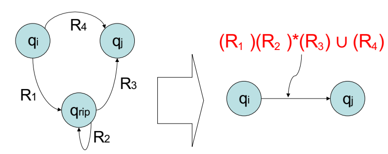
– repeat until only 2 states.

9

Regular expressions and FA

– how to repair the transitions:

– for *every* pair of states q_i and q_j do



10

Regular expressions and FA

– summary:
FA M → k-state GNFA → (k-1)-state GNFA → (k-2)-state GNFA →…→ 2-state GNFA → R

– want to *prove* that this procedure is correct, i.e. L(R) = language recognized by M

- FA M equivalent to k-state GNFA ✓
- i-state GNFA equivalent to (i-1)-state GNFA (we will prove…)
- 2-state GFNA equivalent to R ✓

11

Regular expressions and FA

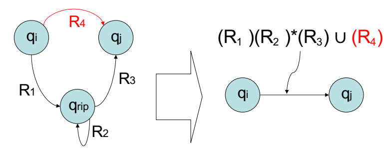
– **Claim**: i-state GNFA G equivalent to (i-1)-state GNFA G’ (obtained by removing q_{rip})

– **Proof**:
- if G accepts string w, then it does so by entering states: $q_0, q_1, q_2, q_3, \ldots, q_{accept}$
- if none are q_{rip} then G’ accepts w (see slide)
- else, break state sequence into runs of q_{rip}:

  $q_0 q_1 \ldots q_i q_{rip} q_{rip} \ldots q_{rip} q_j \ldots q_{accept}$
- transition from q_i to q_j in G’ allows all strings taking G from q_i to q_j using q_{rip} (see slide)
- thus G’ accepts w

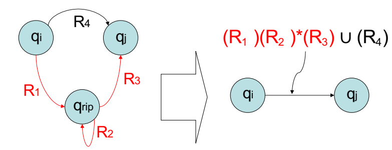
12

Regular expressions and FA



13

Regular expressions and FA



14

Regular expressions and FA

- **Proof** (continued):
  - if G' accepts string w, then every transition from q_i to q_j traversed in G' corresponds to

    either

    a transition from q_i to q_j in G

    or

    transitions from q_i to q_j via q_{rip} in G
  - In both cases G accepts w.
  - Conclude: G and G' recognize the same language.

15

Regular expressions and FA

- **<u>Theorem</u>**: a language L is recognized by a FA iff L is described by a regular expr.
- Languages recognized by a FA are called regular languages.
- Rephrasing what we know so far:
  - regular languages closed under 3 operations
  - NFA recognize exactly the regular languages
  - regular expressions describe exactly the regular languages

16

Limits on the power of FA

- Is *every* language describable by a sufficiently complex regular expression?
- If someone asks you to design a FA for a language that seems hard, how do you know when to give up?
- Is this language regular?

  {w : w has an equal # of "01" and "10" substrings}

17

Limits on the power of FA

- Intuition:
  - FA can only remember finite amount of information. They cannot count
  - languages that "entail counting" should be non-regular…
- Intuition not enough:

  {w : w has an equal # of "01" and "10" substrings}

  $= 0\Sigma^*0 \cup 1\Sigma^*1$

  but {w: w has an equal # of "0" and "1" substrings} is not regular!

18

Limits on the power of FA

How do you *prove* that there is *no* Finite Automaton recognizing a given language?

19

Non-regular languages

Pumping Lemma: Let L be a regular language. There exists an integer p ("pumping length") for which every $w \in L$ with $|w| \geq p$ can be written as

$w = xyz$ such that

1. for every $i \geq 0$, $xy^iz \in L$, and
2. $|y| > 0$, and
3. $|xy| \leq p$.

20

Non-regular languages

- Using the Pumping Lemma to prove L is not regular:
  - assume L is regular
  - then there exists a pumping length p
  - select a string $w \in L$ of length at least p
  - argue that for every way of writing $w = xyz$ that satisfies (2) and (3) of the Lemma, pumping on y yields a string not in L.
  - contradiction.

21

Pumping Lemma Examples

- Theorem: $L = \{0^n1^n : n \geq 0\}$ is not regular.
- Proof:
  - let p be the pumping length for L
  - choose $w = 0^p1^p$

$$w = \underbrace{000000000\ldots0}_{p}\underbrace{111111111\ldots1}_{p}$$

  - $w = xyz$, with $|y| > 0$ and $|xy| \leq p$.

22

Pumping Lemma Examples

- 3 possibilities:

$$w = \underbrace{00000}_{x}\underbrace{000}_{y}\underbrace{0\ldots0111111111\ldots1}_{z}$$

$$w = \underbrace{000000000\ldots011}_{x}\underbrace{11111}_{y}\underbrace{11\ldots1}_{z}$$

$$w = \underbrace{0000000}_{x}\underbrace{00\ldots0111111}_{y}\underbrace{1111\ldots1}_{z}$$

- in each case, pumping on y gives a string not in language L.

23

Pumping Lemma Examples

- Theorem: L = {w: w has an equal # of 0s and 1s} is not regular.
- Proof:
  - let p be the pumping length for L
  - choose $w = 0^p1^p$

$$w = \underbrace{000000000\ldots0}_{p}\underbrace{111111111\ldots1}_{p}$$

  - $w = xyz$, with $|y| > 0$ and $|xy| \leq p$.

24

Pumping Lemma Examples

– 3 possibilities:

$$w = \underbrace{00000}_{x}\underbrace{000}_{y}\underbrace{0\ldots0111111111\ldots1}_{z}$$

$$w = \underbrace{000000000\ldots011}_{x}\underbrace{11111}_{y}\underbrace{11\ldots1}_{z}$$

$$w = \underbrace{000000000\ldots}_{x}\underbrace{01}_{y}\underbrace{11111111\ldots1}_{z}$$

– first 2 cases, pumping on y gives a string not in language L; 3rd case a problem!

Pumping Lemma Examples

– recall condition 3: $|xy| \leq p$

– since $w = 0^p1^p$ we know more about how it can be divided, and this case cannot arise:

$$w = \underbrace{000000000\ldots}_{x}\underbrace{01}_{y}\underbrace{11111111\ldots1}_{z}$$

– so we do get a contradiction.

– conclude that L is not regular.