CS21
Decidability
and
Tractability
Lecture 4
January 10,
2024

Regular expressions
• R is a regular expression if R is
  – a, for some a ∈ Σ
  – ε, the empty string
  – Ø, the empty set
  – (R₁ ∪ R₂), where R₁ and R₂ are reg. exprs.
  – (R₁ ∘ R₂), where R₁ and R₂ are reg. exprs.
  – (R₁*), where R₁ is a regular expression
A reg. expression R describes the language L(R).

January 10, 2024
CS21 Lecture 4

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.

January 10, 2024
CS21 Lecture 4

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

Proof: given regular expression R we will
        build a NFA that recognizes L(R).
        then NFA, FA equivalence implies a FA for
        L(R).

January 10, 2024
CS21 Lecture 4

Regular expressions and FA
• R is a regular expression if R is
  – a, for some a ∈ Σ
  – ε, the empty string
  – Ø, the empty set

January 10, 2024
CS21 Lecture 4

Regular expressions and FA
• R is a regular expression if R is
  – (R₁ ∪ R₂), where R₁ and R₂ are reg. exprs.
  – (R₁ ∘ R₂), where R₁ and R₂ are reg. exprs.
  – (R₁*), where R₁ is a regular expression

January 10, 2024
CS21 Lecture 4
Theorem: a language \( L \) is recognized by a FA if and only if \( L \) is described by a regular expression.

Must prove two directions:

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

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

Proof: given FA \( M \) that recognizes \( L \), we will

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

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 \( w_i \in L(R) \)

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 with all possible incoming arrows
  - every state has all possible outgoing arrows; exception: start state has no self-loop

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.
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.

January 10, 2024
CS21 Lecture 4

Regular expressions and FA

- how to repair the transitions:
  - for every pair of states $q_i$ and $q_j$ do
    
    $$(R_1)(R_2)^*(R_3) \cup (R_4)$$

January 10, 2024
CS21 Lecture 4

Regular expressions and FA

- summary:
  - FA $M \rightarrow k$-state GNFA $\rightarrow (k-1)$-state GNFA $\rightarrow (k-2)$-state GNFA $\rightarrow \ldots \rightarrow 2$-state GNFA $\rightarrow R$
  - want to prove that this procedure is correct, i.e. $L(R) =$ language recognized by $M$
    
    - $M$ equivalent to $k$-state GNFA
    - $i$-state GNFA equivalent to $(i-1)$-state GNFA (we will prove...)
    - $2$-state GFNA equivalent to $R$

January 10, 2024
CS21 Lecture 4

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: $q_0, q_1, q_2, \ldots, q_{accept}$
  - if none are $q_{rip}$ then $G'$ accepts $w$ (see slide)
  - else, break state sequence into runs of $q_{rip}$:
    
    $$(q_0) (q_1) (q_{rip}) (q_2, \ldots, q_{accept})$$
    
    - 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 $w$

January 10, 2024
CS21 Lecture 4

Regular expressions and FA

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.

January 10, 2024
CS21 Lecture 4

Regular expressions and FA

- how to repair the transitions:
  - for every pair of states $q_i$ and $q_j$ do
    
    $$(R_1)(R_2)^*(R_3) \cup (R_4)$$

January 10, 2024
CS21 Lecture 4
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 \( q_{ri} \) in \( G \).
  - In both cases \( G \) accepts \( w \).
  - Conclude: \( G \) and \( G' \) recognize the same language.

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?
  \( \{w : w \text{ has an equal # of "01" and "10" substrings}\} \)

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 \( i \geq 0 \), \( xyz \in L \), and
2. \( |y| > 0 \), and
3. \( |xy| \leq p \).
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.

Pumping Lemma Examples

- Theorem: $L = \{0^n1^n : n \geq 0\}$ is not regular.
  - Proof:
    - let $p$ be the pumping length for $L$
    - choose $w = 0^p1^p$
      $$w = 00000000...01111111...1$$
    - $w = xyz$, with $|y| > 0$ and $|xy| \leq p$.

Pumping Lemma Examples

- 3 possibilities:
  - $w = 00000000...01111111...1$
  - $w = 00000000...01111111...1$
  - $w = 00000000...01111111...1$
  - in each case, pumping on $y$ gives a string not in language $L$.

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 $w = 0^p1^p$
      $$w = 00000000...01111111...1$$
    - $w = xyz$, with $|y| > 0$ and $|xy| \leq p$.

Pumping Lemma Examples

- 3 possibilities:
  - $w = 00000000...01111111...1$
  - $w = 00000000...01111111...1$
  - $w = 00000000...01111111...1$
  - 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:
    $$w = 00000000...01111111...1$$
  - so we do get a contradiction.
  - conclude that $L$ is not regular.