Regular Expressions and Inductive Proofs

Mon Sept 28, 2020
HW2 questions?
Big Picture Road Map

• We ultimately want to prove:
  • Regular Languages $\Leftrightarrow$ Regular Expressions

• First, we need to show these operations are closed for reglangs:
  • Union (**done**)!
  • Concatenation (**done**)!
  • Kleene star (**done**)!
Thm: A language is regular iff some regular expression describes it

• => If a language is regular, it is described by a regular expression

• <= If a language is described by a regular expression, it is regular
  • Easy!
  • Construct the NFA! (Lemma 1.55)
Regexp-\rightarrow NFA (Lemma 1.55)

**Definition 1.52**

Say that \( R \) is a *regular expression* if \( R \) is

1. \( a \) for some \( a \) in the alphabet \( \Sigma \),
2. \( \varepsilon \),
3. \( \emptyset \),
4. \( (R_1 \cup R_2) \), where \( R_1 \) and \( R_2 \) are regular expressions,
5. \( (R_1 \circ R_2) \),
6. \( (R_1^*) \), where

*Recursively call Regexp-\rightarrow NFA on \( R_1 \) and \( R_2 \), to get \( N_1 \) for \( R_1 \), and \( N_2 \) for \( R_2 \), then combine NFAs!*
**Thm:** A lang is regular iff some regexp describes it

- => If a language is regular, it is described by a regexp
  - Hard!
  - Need something new: a GNFA
- <= If a language is described by a regexp, it is regular
  - Easy!
  - Construct the NFA! (Lemma 1.55)
GNFA = NFA with regexp transitions

- To convert GNFA to regexp, repeatedly "rip out" states until 2 left
GNFA→Regexp(G) fn (where G is GNFA)

• If G has 2 states, return the regular expression

\[ q_i \xrightarrow{(R_1)(R_2)^*(R_3) \cup (R_4)} q_j \]

• Else:
  • “Rip” out one state to get G’
  • Recursively call GNFA→Regexp(G’)
Need to prove \texttt{GNFA→Regexp}(G) correct

• Specifically, need to prove $\text{Lang}(G) = \text{Lang}(\text{GNFA→Regexp}(G))$
• i.e., \texttt{GNFA→Regexp} should not change the language!
Kinds of Mathematical Proof

• Proof by construction

• Proof by contradiction

• Proof by induction
  • Use to prove properties of recursive definitions or functions
Proof by Induction

• To prove property P on all objects of a kind x
  • First, prove base case (usually easy)
  • Then, prove the induction step:
    • Assume the induction hypothesis (IH): P(x) is true, for some x
    • and use it to prove P(x+1)
    • The key is x must be smaller than x+1
Correctness of $\text{GNFA-} \rightarrow \text{Regexp}(G)$

$\text{GNFA-} \rightarrow \text{Regexp}(G)$: (G is an GNFA)

If G has 2 states, return the regexp
Else:
  “Rip” out one state to get $G'$
  Recursively Call $\text{GNFA-} \rightarrow \text{Regexp}(G')$

- **Prove** (by induction): $\text{Lang}(G) = \text{Lang}(\text{GNFA-} \rightarrow \text{Regexp}(G))$
Correctness of $\text{GNFA-\rightarrow Regexp}(G)$

$\text{GNFA-\rightarrow Regexp}(G)$: (G is an GNFA)

If G has 2 states, return the regexp

Else:
  “Rip” out one state to get $G'$
  Recursively Call $\text{GNFA-\rightarrow Regexp}(G')$

*Prove* (by induction): $\text{Lang}(G) = \text{Lang}(\text{GNFA-\rightarrow Regexp}(G))$

- **Base case:** G has 2 states
  - So $\text{Lang}(G) = \text{Lang}(\text{GNFA-\rightarrow Regexp}(G))$
Correctness of $\text{GNFA-}\rightarrow\text{Regexp}(G)$

$\text{GNFA-}\rightarrow\text{Regexp}(G)$: (G is an GNFA)
   If G has 2 states, return the regexp

Else:
   “Rip” out one state to get $G'$
   Recursively Call $\text{GNFA-}\rightarrow\text{Regexp}(G')$

• **Prove** (by induction): $\text{Lang}(G) = \text{Lang}(\text{GNFA-}\rightarrow\text{Regexp}(G))$
  • **Base case:** G has 2 states
    • So $\text{Lang}(G) = \text{Lang}(\text{GNFA-}\rightarrow\text{Regexp}(G))$
  ➢ **IH:** Assume $\text{Lang}(G) = \text{Lang}(\text{GNFA-}\rightarrow\text{Regexp}(G))$, for **any** G with n states
Correctness of $\text{GNFA-\rightarrow Regexp}(G)$

$\text{GNFA-\rightarrow Regexp}(G)$: (G is an GNFA)
   If G has 2 states, return the regexp
   Else:
      “Rip” out one state to get G’
      Recursively Call $\text{GNFA-\rightarrow Regexp}(G’)$

• Prove (by induction): $\text{Lang}(G) = \text{Lang}(\text{GNFA-\rightarrow Regexp}(G))$
  • Base case: G has 2 states
    • So $\text{Lang}(G) = \text{Lang}(\text{GNFA-\rightarrow Regexp}(G))$
  • IH: Assume $\text{Lang}(G) = \text{Lang}(\text{GNFA-\rightarrow Regexp}(G))$, for any G with n states
  • Prove for G with n+1
    ➢ After “rip” step, we have a G’ with n states
Correctness of $\text{GNFA-}\rightarrow\text{Regexp}(G)$

$\text{GNFA-}\rightarrow\text{Regexp}(G)$: ($G$ is an GNFA)
- If $G$ has 2 states, return the regexp
- Else:
  - “Rip” out one state to get $G'$
  - Recursively Call $\text{GNFA-}\rightarrow\text{Regexp}(G')$

- **Prove** (by induction): $\text{Lang}(G) = \text{Lang}(\text{GNFA-}\rightarrow\text{Regexp}(G))$
  - **Base case**: $G$ has 2 states
    - So $\text{Lang}(G) = \text{Lang}(\text{GNFA-}\rightarrow\text{Regexp}(G))$
  - **IH**: Assume $\text{Lang}(G) = \text{Lang}(\text{GNFA-}\rightarrow\text{Regexp}(G))$, for any $G$ with $n$ states
  - **Prove for $G$ with $n+1$**
    - After “rip” step, we have a $G'$ with $n$ states
    - $\text{Lang}(G') = \text{Lang}(\text{GNFA-}\rightarrow\text{Regexp}(G'))$ (by assumption)
Correctness of $\text{GNFA-}\rightarrow\text{Regexp}(G)$

$\text{GNFA-}\rightarrow\text{Regexp}(G)$: (G is an GNFA)

If G has 2 states, return the regexp

Else:

“Rip” out one state to get $G'$

Recursively Call $\text{GNFA-}\rightarrow\text{Regexp}(G')$

- **Prove** (by induction): $\text{Lang}(G) = \text{Lang}(\text{GNFA-}\rightarrow\text{Regexp}(G))$
  - **Base case**: G has 2 states
    - So $\text{Lang}(G) = \text{Lang}(\text{GNFA-}\rightarrow\text{Regexp}(G))$
  - **IH**: Assume $\text{Lang}(G) = \text{Lang}(\text{GNFA-}\rightarrow\text{Regexp}(G))$, for **any** G with $n$ states
  - **Prove for G with $n+1$**
    - After “rip” step, we have a G’ with $n$ states
    - $\text{Lang}(G') = \text{Lang}(\text{GNFA-}\rightarrow\text{Regexp}(G'))$ (by assumption)
    - Now just need correctness of “rip” step
GNFA→Regexp: “rip” step correctness

- Must prove:
  - Every string accepted before is accepted after
  - 2 cases
    - String does not go through qrip
      - Acceptance unchanged
    - String goes through qrip
      - Acceptance unchanged?
Thm: A lang is regular iff some regexp describes it

• => If a language is regular, it is described by a regexp
  • Hard!
  • Use GNFA→Regexp(G) to convert GNFA to regexp!

• <= If a language is described by a regexp, it is regular
  • Easy!
  • Construct the NFA!

DONE!
Now we may use regular expressions to to represent regular langs.
Regexp makes some closure operations easier to prove, via induction!
Regexp is inductive definition; constructed from smaller regexps

**Definition 1.52**

Say that $R$ is a *regular expression* if $R$ is

1. $a$ for some $a$ in the alphabet $\Sigma$,
2. $\varepsilon$,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, where $R_1$ and $R_2$ are regular expressions,
5. $(R_1 R_2)$, where $R_1$ and $R_2$ are regular expressions,
6. $R^*$

So any inductive proof of regular languages can just follow this definition!
Homomorphisms: closed under reg langs

A homomorphism is a function $f : \Sigma \rightarrow \Gamma$ from one alphabet to another.

- extend $f$ to operate on strings by defining
  $$f(w) = f(w_1)f(w_2) \cdots f(w_n),$$
  where $w = w_1w_2 \cdots w_n$ and each $w_i \in \Sigma$.

- extend $f$ to operate on languages by defining $f(A) = \{f(w) | w \in A\}$

Think like a secret decoder!
- E.g., if $f(x) \rightarrow c$, $f(y) \rightarrow a$, $f(z) \rightarrow t$, then “xyz” $\rightarrow$ “cat”

Prove: homomorphisms are closed under regular langs
- E.g., if $A$ is regular, then $f(A)$ is regular
Homomorphisms closed for reg langs

• Proof by construction
  • If lang L is regular, then DFA M recognizes it.
  • Create M’ from M such that all transitions use new alphabet
  • (Details left to you to work out)

• Proof by induction:
  • If lang L is regular, then some regexp R describes it.
Proof by Induction

• To prove property P on all objects of a kind x
  • First, prove base case (usually easy)
  • Then, prove the induction step:
    • Assume the induction hypothesis (IH) P(x) is true, for some x
    • and use it to prove P(x+1)
    • The key is x must be smaller than x+1
Homomorphisms closed: inductive proof

**Definition 1.52**

Say that \( R \) is a *regular expression* if \( R \) is

1. \( a \) for some \( a \) in the alphabet \( \Sigma \), 3 base cases
2. \( \varepsilon \),
3. \( \emptyset \), I.H: assume true for smaller \( R_1 \) (and \( R_2 \)), i.e., applying homomorphism produces regular lang
4. \( (R_1 \cup R_2) \), where \( R_1 \) and \( R_2 \) are regular expressions,
5. \( (R_1 \circ R_2) \), where \( R_1 \) and \( R_2 \) are regular expressions,
6. \( (R_1^*) \), where \( R_1 \) is a regular expression.

Now we just need to show closure of union, concat, and star operations for reg langs 😊
Next Time: Non-regular languages

• In general, we have many ways to show a language is regular
  • Construct DFA or NFA (or GNFA)
  • Create a regular expression

• But how to show a language is not regular?

• E.g., how do we know that XML is non-regular???

• Hint: The Pumping Lemma!
Check-in Quiz 9/28
On gradescope

End of Class Survey 9/28
See course website