UMB CS 420

Inductive Proofs

(Proofs involving recursion)

Monday March 4, 2024
Announcements

• HW 3 in
  • Due Mon 3/4 12pm EST (noon)

• HW 4 out
  • Due Mon 3/18 12pm EST (noon)
  • (After spring break)
Thm: A Lang is Regular \iff Some Reg Expr Describes It

⇒ If a language is regular, then it’s described by a regular expr

  • Use GNFA\rightarrow RegExpr to convert GNFA \rightarrow equiv regular expression!

  ✔️

  ???

  This time, let’s really prove equivalence!
  (we previously “proved” it with some examples)

⇐ If a language is described by a regular expr, then it’s regular

✔️ • Convert regular expression \rightarrow equivalent NFA!
**GNFA→RegExpr Equivalence**

- **Equivalent** = the language does not change (i.e., same set of strings)!

**Statement to Prove:**

\[
\text{LANGOF}\left( G \right) = \text{LANGOF}\left( R \right)
\]

- **where:**
  - \( G = \) a GNFA
  - \( R = \) a Regular Expression = \text{GNFA→RegExpr}( G )

Language could be infinite set of strings!

(how can we show equivalence for a possibly infinite set of strings?)

This time, let’s **really prove equivalence**! (we previously “proved” it with some examples)

Recursion!
Kinds of Mathematical Proof

• **Deductive proof** (from before)
  - **Start** with: assumptions, axioms, and definitions
  - **Prove:** news conclusions by making logical inferences (e.g., modus ponens)

• **Proof by induction** (i.e., “a proof involving recursion”) (now)
  - Same as above …
  - But: use this when proving something that is **recursively** defined

A valid recursive definition has:
- **base case(s)** and
- **recursive case(s)** (with “smaller” self-reference)
Proof by Induction

To Prove: **Statement** for **recursively defined** “thing” \( x \):

1. **Prove**: **Statement** for **base case** of \( x \)
2. **Prove**: **Statement** for **recursive case** of \( x \):
   - **Assume**: induction hypothesis (IH)
     - i.e., **Statement** is true for **some** \( x_{\text{smaller}} \)
     - E.g., if \( x \) is number, then “smaller” = lesser number

   - **Prove**: **Statement** for \( x \), using IH (and known definitions, theorems ...)
     - Typically: show that going from \( x_{\text{smaller}} \) to larger \( x \) is true!

---

A valid recursive definition has:
- **base case(s)** and
- **recursive case(s)** (with “smaller” self-reference)
Natural Numbers Are Recursively Defined

A Natural Number is:
- \(0\)
- Or \(k + 1\), where \(k\) is a Natural Number

Recursive definition is valid because self-reference is “smaller”

So, proving things about: recursive Natural Numbers requires recursive proof, i.e., proof by induction!

A valid recursive definition has:
- base case and
- recursive case (with “smaller” self-reference)
Proof By Induction Example (Sipser Ch 0)

Prove true: \( P_t = PM^t - Y \left( \frac{M^t - 1}{M - 1} \right) \)

- \( P_t \) = loan balance after \( t \) months
- \( t \) = # months
- \( P \) = principal = original amount of loan
- \( M \) = interest (multiplier)
- \( Y \) = monthly payment

(Details of these variables not too important here)
Proof By Induction Example (Sipser Ch 0)

Prove true: \( P_t = PM^t - Y \left( \frac{M^t - 1}{M - 1} \right) \)

Proof: by induction on natural number \( t \)

Base Case, \( t = 0 \):

\[
P_0 = PM^0 - Y \left( \frac{M^0 - 1}{M - 1} \right) = P
\]

A proof by induction follows the cases of the recursive definition (here, natural numbers) that the induction is “on”

A Natural Number is:
- 0
- Or \( k + 1 \), where \( k \) is a natural number

\( P_0 = P \) is a true statement! (amount owed at start = loan amount)
Proof By Induction Example (Sipser Ch 0)

Prove true: \( P_t = P M^t - Y \left( \frac{M^t - 1}{M - 1} \right) \)

Inductive Case: \( t = k + 1 \), for some natural num \( k \)

• Inductive Hypothesis (IH), assume statement is true for some \( t = (\text{smaller}) k \)

\[
P_k = P M^k - Y \left( \frac{M^k - 1}{M - 1} \right)
\]

Goal statement to prove, for \( t = k+1 \):

\[
P_{k+1} = P M^{k+1} - Y \left( \frac{M^{k+1} - 1}{M - 1} \right)
\]

• Proof of Goal:

\[
P_{k+1} = P M^{k+1} - P_k M - Y
\]

A proof by induction follows cases of recursive definition (here, natural numbers) that the induction is “on”

A Natural Number is:

• 0 ✔
• \( k + 1 \), for some nat num \( k \)

• Proof of Goal:

Definition of Loan:

amt owed in month \( k+1 \) = amt owed in month \( k \) * interest \( M \) – amt paid \( Y \)
In-class Exercise: Proof By Induction

Prove: \( z \neq 1 \)

\[
\sum_{i=0}^{m} z^i = \frac{1 - z^{m+1}}{1 - z}
\]

Use Proof by Induction.

Make sure to clearly state what (number) the induction is “on”
Proof by Induction: CS 420 Example

Statement to prove: \[ \text{LANGOF}(G) = \text{LANGOF}(R = \text{GNFA} \rightarrow \text{RegExpr}(G)) \]

• Where:
  • \(G\) is a GNFA
  • \(R\) is a Regular Expression \(\text{GNFA} \rightarrow \text{RegExpr}(G)\)

• i.e., \(\text{GNFA} \rightarrow \text{RegExpr}\) must not change the language!

This time, let’s really prove equivalence!
(we previously “proved” it with some examples)
Proof by Induction: CS 420 Example

Statement to prove: \( \text{LANG}_G(G) = \text{LANG}_G(\text{GNFA}\rightarrow\text{RegExpr}(G)) \)

Proof: by Induction on \# of states in \( G \)

1. Prove **Statement** is true for base case \( G \) has 2 states

(Modified) Recursive definition:

A “NatNumber > 1” is:

- 2
- Or \( k + 1 \), where \( k \) is a “NatNumber > 1”
Last Time

**GNFA→RegExp** (recursive) function

On GNFA input $G$:
- If $G$ has 2 states, return the regular expression (from the transition), e.g.:

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

Equivalent regular expression

GNFA

Base Case
Proof by Induction: CS 420 Example

Statement to prove: \[ \text{LANGOF} (G) = \text{LANGOF} ( \text{GNFA→RegExpr}(G)) \]

Proof: by Induction on \# of states in \(G\)

✅ 1. Prove Statement is true for base case \(G\) has 2 states

\( G \):

- \( q_i \) \( \xrightarrow{R} \) \( q_j \)

**Statements**

1. \[ \text{LANGOF} ( \begin{array}{c} q_i \\ \xrightarrow{R} \\ q_j \end{array} ) = \text{LANGOF} ( R ) \]
2. \[ \text{LANGOF} ( \begin{array}{c} q_i \\ \xrightarrow{R} \\ q_j \end{array} ) = \text{LANGOF} ( \text{GNFA→RegExpr}(G) ) \]

**Justifications**

1. Definition of GNFA
2. Definition of GNFA→RegExpr (base case)
3. From (1) and (2)

---

Don’t forget the Statements / Justifications!
Proof by Induction: CS 420 Example

Statement to prove: \( \text{LANGOF}(G) = \text{LANGOF}(\text{GNFA}\Rightarrow\text{RegExp}(G)) \)

Proof: by Induction on # of states in \( G \)

1. Prove Statement is true for base case: \( G \) has 2 states

2. Prove Statement is true for recursive case: \( G \) has > 2 states
GNFA$\rightarrow$RegExpr (recursive) function

On GNFA input $G$:

- If $G$ has 2 states, return the regular expression (from the transition), e.g.:

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

- Else:
  - “Rip out” one state
  - “Repair” the machine to get an equivalent GNFA $G'$
  - Recursively call GNFA$\rightarrow$RegExpr($G'$) (with a “smaller” $G'$)
Proof by Induction: CS 420 Example

Statement to prove: \( \text{LANGOF}(G) = \text{LANGOF}(\text{GNFA}\rightarrow\text{RegExpr}(G)) \)

Proof: by Induction on \# of states in \( G \)

1. Prove Statement is true for base case
   - \( G \) has 2 states

2. Prove Statement is true for recursive case:
   - Assume the induction hypothesis (IH):
     - Statement is true for smaller \( G' \)
   - Use it to prove Statement is true for \( G > 2 \) states
     - Show that going from \( G \) to smaller \( G' \) is true!

Don’t forget the Statements / Justifications!

Show that “rip/repar” step ✓ converts \( G \) to smaller, equivalent \( G' \)
Proof by Induction: CS 420 Example

Statement to prove: \[ \text{LANGOF}(G) = \text{LANGOF}(\text{GNFA} \triangleright \text{RegExpr}(G)) \]

Proof: by Induction on \# of states in \( G \)

1. **Prove Statement is true for base case**
   - \( G \) has 2 states
   - \( G = G' \)
   - Known “facts” available to use:
     - IH
     - Equiv of Rip/Repair step
     - Def of GNFA \( \triangleright \) RegExpr

2. **Prove Statement is true for recursive case**: \( G \) has more than 2 states
   - \( G \to G' \)
   - Use it to prove if \( G' \) is inductive
   - Show that \( \text{LANGOF}(G) = \text{LANGOF}(\text{GNFA} \triangleright \text{RegExpr}(G')) \)
   - Plug in

Statements
1. \( \text{LANGOF}(G') = \text{LANGOF}(\text{GNFA} \triangleright \text{RegExpr}(G')) \)
2. \( \text{LANGOF}(G) = \text{LANGOF}(G') \)
3. \( \text{GNFA} \triangleright \text{RegExpr}(G) = \text{GNFA} \triangleright \text{RegExpr}(G') \)
4. \( \text{LANGOF}(G) = \text{LANGOF}(\text{GNFA} \triangleright \text{RegExpr}(G)) \)

Justifications
1. IH
2. Equivalence of Rip/Repair step (prev)
3. Def of GNFA \( \triangleright \) RegExpr (recursive call)
4. From (1), (2), and (3)
Thm: A Lang is Regular iff Some Reg Expr Describes It

⇒ If a language is regular, then it’s described by a regular expr

✓ • Use GNFA→RegExpr to convert GNFA → equiv regular expression!

⇐ If a language is described by a regular expr, then it’s regular

✓ • Convert regular expression → equiv NFA!

Now: we can use regular expressions to represent regular langs!

So a regular language has these equivalent representations:
- DFA
- NFA
- Regular Expression
So Far: How to Prove A Language Is Regular?

Key step, either:

• Construct DFA

• Construct NFA

• Create Regular Expression

\[ R \text{ is a regular expression if } R \text{ is} \]
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. \( R_1 \circ R_2 \), where \( R_1 \) and \( R_2 \) are regular expressions, or
6. \( R_1^+ \), where \( R_1 \) is a regular expression.
Proof by Induction

To Prove: a **Statement** about a recursively defined “thing” \( x \):

1. **Prove**: Statement for **base case** of \( x \)

2. **Prove**: Statement for **recursive case** of \( x \):
   - Assume: induction hypothesis (IH)
     - I.e., Statement is true for some \( x_{\text{smaller}} \)
     - E.g., if \( x \) is number, then “smaller” = lesser number
     - E.g., if \( x \) is regular expression, then “smaller” = ...
   - Prove: Statement for \( x \), using IH (and known definitions, theorems ...)
     - Usually, must show that going from \( x_{\text{smaller}} \) to larger \( x \) is true!
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. $(R_1 \circ R_2)$, where $R_1$ and $R_2$ are regular expressions, or 
6. $(R_1^*)$, where $R_1$ is a regular expression.
**Thm:** Reverse is Closed for Regular Langs

**Example string:** $abc^R = cba$

For any string $w = w_1w_2 \cdots w_n$, the *reverse* of $w$, written $w^R$, is the string $w$ in reverse order, $w_n \cdots w_2w_1$.

For any language $A$, let $A^R = \{ w^R \mid w \in A \}$

**Example language:**

$\{ a, ab, abc \}^R = \{ a, ba, cba \}$

**Theorem:** if $A$ is regular, so is $A^R$

**Proof:** by induction on the regular expression of $A$
Thm: Reverse is Closed for Regular Langs

if $A$ is regular, so is $A^R$

Proof: by Induction on regular expression of $A$: (6 cases)

1. $a$ for some $a$ in the alphabet $\Sigma$,
   same reg. expr. represents $A^R$ so it is regular

2. $\varepsilon$,
   same reg. expr. represents $A^R$ so it is regular

3. $\emptyset$,
   same reg. expr. represents $A^R$ so it is regular

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, or

6. $(R_1^*)$, where $R_1$ is a regular expression.

Need to Prove: if $A$ is a regular language, described by reg expr $R_1 \cup R_2$, then $A^R$ is regular

IH1: if $A_1$ is a regular language, described by reg expr $R_1$, then $A_1^R$ is regular

IH2: if $A_2$ is a regular language, described by reg expr $R_2$, then $A_2^R$ is regular

“smaller”
Thm: Reverse is Closed for Regular Langs

Proof: by Induction on regular expression of $A$: (Case # 4)

**Statements**

1. Language $A$ is regular, with reg expr $R_1 \cup R_2$
2. $R_1$ and $R_2$ are regular expressions
3. $R_1$ and $R_2$ describe regular langs $A_1$ and $A_2$
4. If $A_1$ is a regular language, then $A_1^R$ is regular
5. If $A_2$ is a regular language, then $A_2^R$ is regular
6. $A_1^R$ and $A_2^R$ are regular
7. $A_1^R \cup A_2^R$ is regular
8. $A_1^R \cup A_2^R = (A_1 \cup A_2)^R$
9. $A = A_1 \cup A_2$
10. $A^R$ is regular

**Justifications**

1. Assumption of IF in IF-THEN
2. Def of Regular Expression
3. Reg Expr $\Leftrightarrow$ Reg Lang (Prev Thm)
4. IH
5. IH
6. By (3), (4), and (5)
7. Union Closed for Reg Langs
8. Reverse and Union Ops Commute
9. By (1), (2), and (3)
10. By (7), (8), (9)
**Thm:** Reverse is Closed for Regular Langs

if $A$ is regular, so is $A^R$

**Proof:** by Induction on regular expression of $A$: (6 cases)

- **Base cases**
  - 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. $(R_1 \circ R_2)$, where $R_1$ and $R_2$ are regular expressions, or
  - 6. $(R_1^*)$, where $R_1$ is a regular expression.

Remaining cases will use similar reasoning
Non-Regular Languages?

• Are there languages that are not regular languages?

• How can we prove that a language is not a regular language?
Submit in-class work 3/4

See gradescope