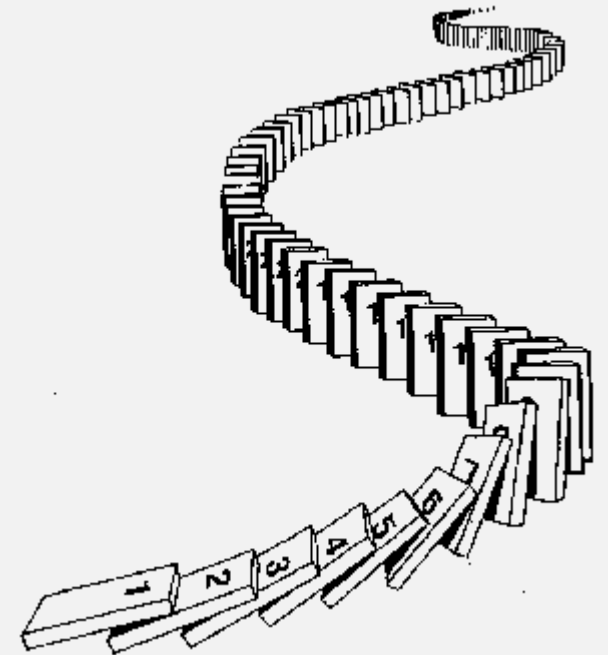


UMB CS 420
Inductive Proofs

Thursday, October 6, 2022



Announcements

- HW 1
 - Grades returned
 - Use Gradescope “Regrade Request” feature for questions / complaints

- HW 3
 - Due Sun 10/9 11:59pm EST

Kinds of Mathematical Proof

- **Deductive proof** (from before)
 - Starting from assumptions and known definitions,
 - Reach conclusion by making logical inferences
- **Inductive proof** (now)
 - ...
 - Use this when working with recursive definitions

Proof by Induction

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

1. Prove: *Statement* for base case of x (usually easy)
2. Prove: *Statement* for recursive case of x :
 - Assume: **induction hypothesis (IH)**
i.e., *Statement* is true for some x_{smaller}
 - E.g., if x is number, then “smaller” = lesser number
 - Prove: *Statement* for x_{larger} - using IH (and known definitions, theorems ...)
 - Usually, must show that going from x_{smaller} to x_{larger} preserves *Statement*

Natural Numbers Are Recursively Defined


A Natural Number is:

- **0**
- Or **$k + 1$** , where **k** is a Natural Number

Self-reference



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



So proving things about Natural Numbers requires induction!

Last Time: 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)

Last Time: Proof By Induction Example (Sipser Ch 0)

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

An inductive proof exactly follows the recursive definition (here, natural numbers) that the induction is “on”

Proof: by **induction** on natural number t

Base Case, $t = 0$:

- Goal: Show $P_0 = P$
- Proof of Goal:

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

Plug in $t = 0$

Simplify, to get to goal statement

A Natural Number is:

- 0
- Or $k + 1$, where k is a natural number

Last Time: Proof By Induction Example (Sipser Ch 0)

An inductive proof exactly follows the recursive definition (here, natural numbers) that the induction is "on"

A Natural Number is:

- 0
- $k + 1$, for some nat num k

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

Inductive Case: $t = k + 1$, for some nat num k

- Inductive Hypothesis (IH), assume statement true for some $t =$ (smaller) k

"Connect together" known definitions and statements

- Goal statement to prove, for $t = k+1$:

$$P_k = PM^k - Y \left(\frac{M^k - 1}{M - 1} \right)$$

Plug in IH

- Proof of Goal:

$$P_{k+1} = PM^{k+1} - Y \left(\frac{M^{k+1} - 1}{M - 1} \right)$$

Simplify, to derive goal statement

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

Definition of P_{k+1}

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 the induction is “on”

Proof by Induction: CS 420 Example

Statement to prove:

$$\text{LANGOF} (G) = \text{LANGOF} (\text{GNFA} \rightarrow \text{RegExpr} (G))$$

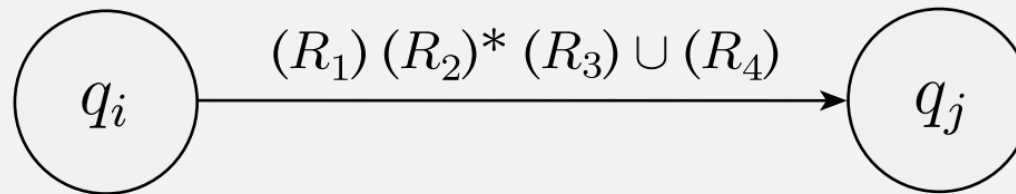
Condition for $\text{GNFA} \rightarrow \text{RegExpr}$ function to be “correct”,
i.e., the languages must be equivalent

Last Time: **GNFA→RegExpr** (recursive) function

On GNFA input G :

Base
Case

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



Recursive definitions have:
- base case and
- recursive case
(with a “smaller” object)

- Else:

Recursive
Case

- “Rip out” one state
- “Repair” the machine to get an equivalent GNFA G'
- Recursively call **GNFA→RegExpr**(G')

Proof by Induction: CS 420 Example

Statement to prove:

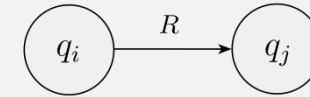
$$\text{LANGOF} (G) = \text{LANGOF} (\text{GNFA} \rightarrow \text{RegExpr} (G))$$

Recursively defined "thing"

Proof: by Induction on # of states in G

✓ 1. Prove Statement is true for base case

G has 2 states



Why is this ok base case?

Statements

1. $\text{LANGOF} (\text{state } q_i \xrightarrow{R} \text{state } q_j) = \text{LANGOF} (R)$
2. $\text{LANGOF} (\text{GNFA} \rightarrow \text{RegExpr} (\text{state } q_i \xrightarrow{R} \text{state } q_j)) = \text{LANGOF} (R)$
3. $\text{LANGOF} (\text{state } q_i \xrightarrow{R} \text{state } q_j) = \text{LANGOF} (\text{GNFA} \rightarrow \text{RegExpr} (\text{state } q_i \xrightarrow{R} \text{state } q_j))$

Goal

Justifications

1. Definition of GNFA
2. Definition of $\text{GNFA} \rightarrow \text{RegExpr}$
3. From (1) and (2)

Don't forget to write out Statements / Justifications !

Proof by Induction: CS 420 Example

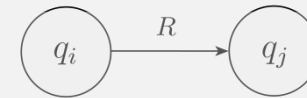
Statement to prove:

$$\text{LANGOF} (G) = \text{LANGOF} (\text{GNFA} \rightarrow \text{RegExpr} (G))$$

Proof: by Induction on # of states in G

✓ 1. Prove Statement is true for base case

G has 2 states

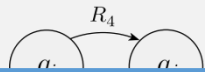


2. Prove Statement is true for recursive case:

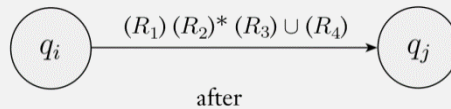
G has > 2 states

- Assume the induction hypothesis (IH):
 - *Statement* is true for smaller G'
- Use it to prove *Statement* is true for larger G
 - Show that going from G to G' preserves *Statement*

$$\begin{aligned} \text{LANGOF} (G') \\ = \\ \text{LANGOF} (\text{GNFA} \rightarrow \text{RegExpr} (G')) \\ \text{(Where } G' \text{ has less states than } G \text{)} \end{aligned}$$



Don't forget to write out Statements / Justifications !



Show that "rip/repair" step converts G to smaller, equivalent G'

Proof by Induction: CS 420 Example

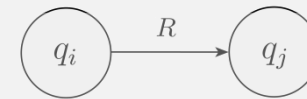
Statement to prove:

$$\text{LANGOF} (G) = \text{LANGOF} (\text{GNFA} \rightarrow \text{RegExpr} (G))$$

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:

G has > 2 states

- Assume the induction hypothesis (IH):
 - *Statement* is true for smaller G'
- Use it to prove *Statement* is true for larger G
 - Show that going from G to G' preserves *Statement*

$$\begin{aligned} & \text{LANGOF} (G') \\ & = \\ & \text{LANGOF} (\text{GNFA} \rightarrow \text{RegExpr} (G')) \end{aligned}$$

(Where G' has less states than G)

Statements

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

Justifications

1. IH
2. Correctness of Rip/Repair step (prev)
3. Def of $\text{GNFA} \rightarrow \text{RegExpr}$
4. From (1), (2), and (3)

Goal

So Far: How to Prove A Language Is Regular?

- Construct DFA
- Construct NFA
- Create Regular Expression

Slightly different because
of recursive definition

R is a **regular expression** if R is

1. a for some a in the alphabet Σ ,
2. ϵ ,
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 (usually easy)
2. Prove: *Statement* for recursive case of x :
 - Assume: **induction hypothesis (IH)**
 - l.e., *Statement* is true for some x_{smaller}
 - E.g., if x is number, then “smaller” = lesser number
 - ➔• E.g., if x is regular expression, then “smaller” = ...
 - Prove: *Statement* for x_{larger} - using IH (and known definitions, theorems ...)
 - Usually, must show that going from x_{smaller} to x_{larger} preserves *Statement*

1. a for some a in the alphabet Σ ,
 2. ϵ ,
 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.
-
- Whole reg expr
- "smaller"

Thm: Reverse is Closed for Regular Langs

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

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

Theorem: if A is regular, so is $A^{\mathcal{R}}$

Proof: by induction on the regular expression of A

Thm: Reverse is Closed for Regular Langs

if A is regular, so is $A^{\mathcal{R}}$

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

Base cases

1. a for some a in the alphabet Σ , same reg. expr. represents $A^{\mathcal{R}}$ so it is regular
2. ϵ , same reg. expr. represents $A^{\mathcal{R}}$ so it is regular
3. \emptyset , same reg. expr. represents $A^{\mathcal{R}}$ so it is regular

Inductive cases

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 $A^{\mathcal{R}}$ is regular

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

IH1: if A_2 is a regular language, described by reg expr R_2 , then $A_2^{\mathcal{R}}$ is regular

“smaller”

Thm: Reverse is Closed for Regular Langs

if A is regular, so is $A^{\mathcal{R}}$

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 $A_1^{\mathcal{R}}$ is regular
5. If A_2 is a regular language, then $A_2^{\mathcal{R}}$ is regular
6. $A_1^{\mathcal{R}}$ and $A_2^{\mathcal{R}}$ are regular
7. $A_1^{\mathcal{R}} \cup A_2^{\mathcal{R}}$ is regular
8. $A_1^{\mathcal{R}} \cup A_2^{\mathcal{R}} = (A_1 \cup A_2)^{\mathcal{R}}$
9. $A = A_1 \cup A_2$
10. $A^{\mathcal{R}}$ is regular

Goal

Justifications

1. Given
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 $A^{\mathcal{R}}$

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

Base cases

- ✓ 1. a for some a in the alphabet Σ ,
- ✓ 2. ϵ ,
- ✓ 3. \emptyset ,

Inductive cases

- ✓ 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

In-Class quiz 10/6

See gradescope