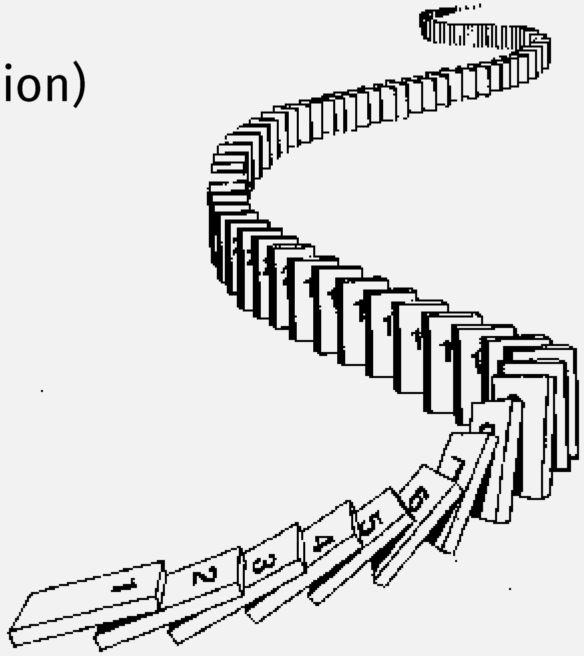
UMB CS 420

Inductive Proofs

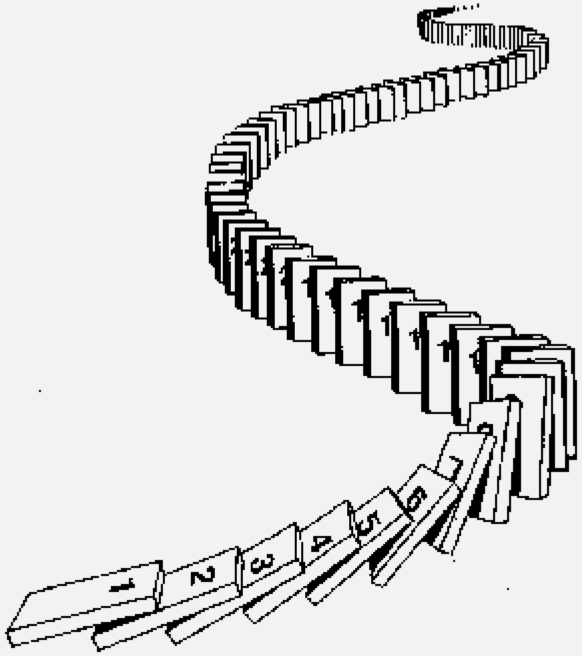
(Proofs involving recursion)

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)



Last Time

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! ☑ ???

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 → equivalent NFA!

GNFA→RegExpr Equivalence

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

Statement to Prove: input output **???**

$$\text{LangOf}(G) = \text{LangOf}(R)$$

This time, let's <u>**really prove**</u> **equivalence!** (we previously "proved" it with some examples)

- <u>where</u>:
  - G = a GNFA
  - R = a Regular Expression = **GNFA→RegExpr**(G)

Language could be infinite set of strings!

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

Recursion!

Kinds of Mathematical Proof

- **Deductive proof** (from before)
  - <u>Start</u> with: **assumptions, axioms,** and **definitions**
  - <u>Prove</u>: **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 <u>recursively</u> defined**

A <u>valid</u> <u>recursive definition</u> has:
- **base case**(s) and
- **recursive case**(s) (with **"smaller" self-reference**)

Proof by Induction

(A proof for **each case** of some recursive definition)

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 $x_{smaller}$
     - 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** $x_{smaller}$ **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:

Base Case
- **0**

Recursive Case
- Or $\boldsymbol{k+1}$, where k is a **Natural Number**

Self-reference

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)

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

<u>Proof</u>: by **induction** on natural number t

A proof by induction follows the <u>cases</u> of the **recursive definition** (here, **natural numbers**) that the **induction** is "**on**"

Base Case, $t = 0$:

A Natural Number is:

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

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

Plug in $t = 0$

Simplify

$P_0 = P$ is a true statement !
(amount owed at start = loan amount)

Proof By Induction Example (Sipser Ch 0)

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

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

A Natural Number is:

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

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

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

IH plugs in "smaller" k

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

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

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

Write $t = k+1$ case in terms of "smaller" k

Plug in IH for P_k

Simplify, to get to goal statement

- Proof of Goal:

$$P_{k+1} = P_k M - Y$$

<u>Definition of Loan</u>:
amt owed in month $k+1$ =
amt owed in month k * interest M – amt paid Y

In-class Exercise: Proof By Induction

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

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: LANGOF (G) = LANGOF (R = **GNFA→RegExpr**(G))

- <u>Where</u>:
  - G = a GNFA
  - R = a Regular Expression **GNFA→RegExpr**(G)

- i.e., **GNFA→RegExpr** must not change the language!

This time, let's <u>really prove</u> equivalence! (we previously "proved" it with some examples)

Proof by Induction: CS 420 Example

Statement to prove: LANGOF (G) = LANGOF (**GNFA→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 an ok base case (instead of zero)?

(Modified) **Recursive definition:**

A "NatNumber > 1" is:

- 2
- Or k + 1, where k is a "NatNumber > 1"

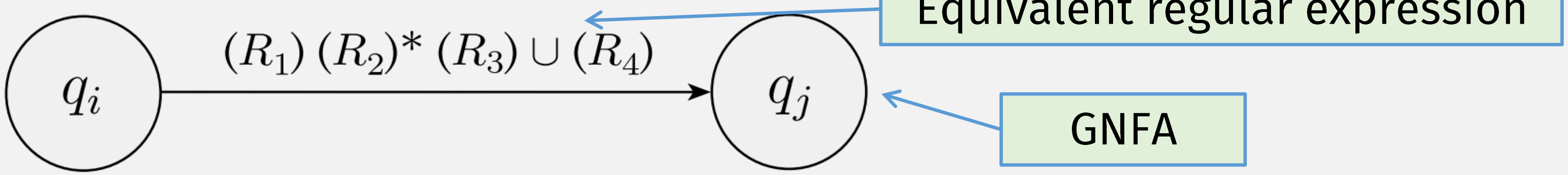
Last Time

GNFA→RegExpr (recursive) function

On GNFA input G:

Base Case

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



Proof by Induction: CS 420 Example

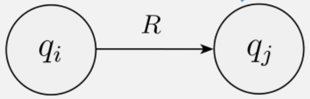
Statement to prove: LANGOF (G) = LANGOF (**GNFA→RegExpr**(G))

Plug in

Proof: by Induction on # of states in G

☑ 1. Prove *Statement* is true for base case

G has 2 states



Statements	Justifications
1. LANGOF ($q_i \xrightarrow{R} q_j$) = LANGOF (R)	1. Definition of GNFA
2. **GNFA→RegExpr**($q_i \xrightarrow{R} q_j$) = R	2. Definition of **GNFA→RegExpr** (base case)
LANGOF ($q_i \xrightarrow{R} q_j$) = LANGOF (**GNFA→RegExpr**($q_i \xrightarrow{R} q_j$))	3. From (1) and (2)

Plug in R

Goal

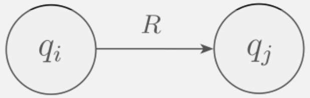
Don't forget the Statements / Justifications !

Proof by Induction: CS 420 Example

Statement to prove: LANGOF (G) = LANGOF (**GNFA→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

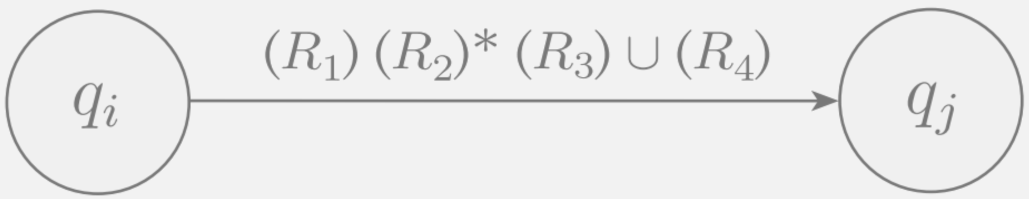
Last Time

GNFA→RegExpr (recursive) function

On GNFA input G:

Base Case

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



- Else:

Recursive Case

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

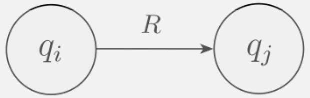
Recursive call (with a "smaller" G')

Proof by Induction: CS 420 Example

Statement to prove: LANGOF (G) = LANGOF (**GNFA→RegExpr**(G))

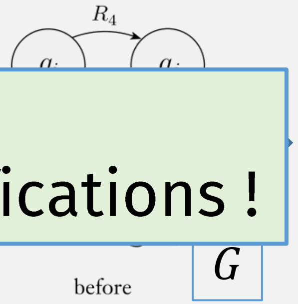
Proof: by Induction on # of states in G

☑ 1. Prove *Statement* is true for base case G has 2 states



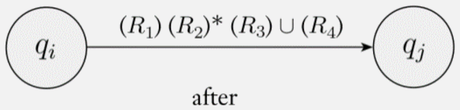
2. Prove *Statement* is true for recursive case: G has > 2 states
   - Assume the **induction hypothesis** (IH):
     - *Statement* is true for smaller G'
   - Use it to prove *Statement* is true for G > 2 states
     - Show that going from G to smaller G' is true!

LANGOF (G')
=
LANGOF (**GNFA→RegExpr**(G'))
(Where G' has less states than G)



G

Don't forget the Statements / Justifications !



smaller G'

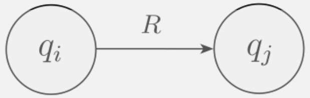
Show that "rip/repair" step ☑ converts G to smaller, equivalent G'

Proof by Induction: CS 420 Example

Statement to prove: **LANGOF (G) = LANGOF (GNFA→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 i
  - *Statement*
- Use it to prov … tates
  - Show that … e!

Known "facts" available to use:
- ☑IH
- ☑Equiv of Rip/Repair step
- ☑Def of GNFA->RegExpr

LANGOF (G')
=
LANGOF (GNFA→RegExpr(G'))
(Where G' has less states than G)

Statements	Justifications
1. LANGOF (G') = LANGOF (GNFA→RegExpr(G'))	1. IH
2. LANGOF (G) = LANGOF (G')	2. Equivalence of Rip/Repair step (prev)
3. GNFA→RegExpr(G) = GNFA→RegExpr(G') — Plug in	3. Def of GNFA→RegExpr (recursive call)
Goal — 4. LANGOF (G) = LANGOF (GNFA→RegExpr(G))	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 we also have another way to prove things about regular languages!

So Far: How to Prove A Language Is Regular?

Key step, either:

- 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
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
     - 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_{smaller} to larger x is true!

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

"smaller"

Whole reg expr

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: $\texttt{abc}^{\mathcal{R}} = \texttt{cba}$

For any string $w = w_1 w_2 \cdots w_n$, the ***reverse*** of w, written $w^{\mathcal{R}}$, is the string w in reverse order, $w_n \cdots w_2 w_1$.

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

Example language:

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

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	Justifications
1. Language A is regular, with reg expr $R_1 \cup R_2$	1. Assumption of IF in IF-THEN
2. R_1 and R_2 are regular expressions	2. Def of Regular Expression
3. R_1 and R_2 describe regular langs A_1 and A_2	3. Reg Expr ⇔ Reg Lang (Prev Thm)
4. If A_1 is a regular language, then $A_1{}^{\mathcal{R}}$ is regular	4. IH
5. If A_2 is a regular language, then $A_2{}^{\mathcal{R}}$ is regular	5. IH
6. $A_1{}^{\mathcal{R}}$ and $A_2{}^{\mathcal{R}}$ are regular	6. By (3), (4), and (5)
7. $A_1{}^{\mathcal{R}} \cup A_2{}^{\mathcal{R}}$ is regular	7. Union Closed for Reg Langs
8. $A_1{}^{\mathcal{R}} \cup A_2{}^{\mathcal{R}} = (A_1 \cup A_2)^{\mathcal{R}}$	8. Reverse and Union Ops Commute
9. $A = A_1 \cup A_2$	9. By (1), (2), and (3)
Goal: 10. $A^{\mathcal{R}}$ is regular	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

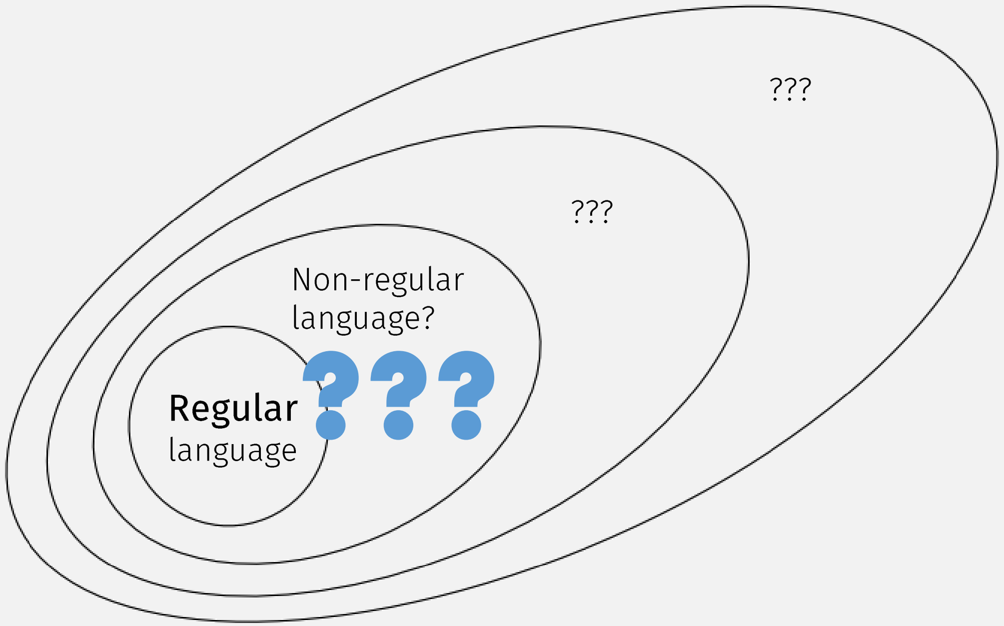
- ☑ **4.** $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,
- **5.** $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, or
- **6.** (R_1^*), where R_1 is a regular expression.

Remaining cases will use similar reasoning

Next Time

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