CSE 311 Section 08

Regular Expressions, CFGs, & Relations

Administrivia

Announcements & Reminders

- Homework 6 was due Wednesday (5/15)
- Midterm grades have been released? (Maybe)
    - Regrade requests are open
- Check your section participation grade on canvas
    - If it different than what you expect, let your TA know

Regular Expressions

Regular Expressions

Basis:

- ε is a regular expression. The empty string itself matches the pattern (and nothing else does).
- $\varnothing$ is a regular expression. No strings match this pattern.
- a is a regular expression, for any $a \in \Sigma$ (i.e. any character). The character itself matching this pattern.

Recursive:

- If A, B are regular expressions then $(A \cup B)$ is a regular expression. matched by any string that matches A or that matches B [or both]).
- If A, B are regular expressions then AB is a regular expression. matched by any string x such that $x = yz$, y matches A and z matches B.
- If A is a regular expression, then $A*$ is a regular expression. matched by any string that can be divided into 0 or more strings that match A.

Regular Expressions

A regular expression is a recursively defined set of strings that form a language.

A regular expression will generate all strings in a language, and won't generate any strings that ARE NOT in the language

Hints:

- Come up with a few examples of strings that ARE and ARE NOT in your language
- Then, after you write your regex, check to make sure that it CAN generate all of your examples that are in the language, and it CAN'T generate those that are not

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

c) Write a regular expression that matches all binary strings that contain the substring "111", but not the substring "000".

Work on this problem with the people around you.

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

base-10 numbers:

Our everyday numbers! Notice we have 10 symbols (0-9) to represent numbers.

256: $(\mathbf{2} * 10^2) + (\mathbf{5} * 10^1) + (\mathbf{6} * 10^0)$

base-2 numbers: (binary)

10: $(\mathbf{1} * 2^1) + (\mathbf{0} * 2^0)$

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)^*$

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

(0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9)*

⚠️ "0101" or "091" **are not** Base-10 numbers

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)^*$

⚠️ "0101" or "091" **are not** Base-10 numbers

All possible *strings* using numbers 0-9 that never start with 0

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

(0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9)*

⚠️ "0101" or "091" **are not** Base-10 numbers

All possible *strings* using numbers 0-9 that never start with 0

(1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9)(0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9)*

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

(0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9)*

⚠️ "0101" or "091" **are not** Base-10 numbers

All possible *strings* using numbers 0-9 that never start with 0

(1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9)(0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9)*

⚠️ "0" **is** a Base-10 number not considered

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

(0 U 1 U 2 U 3 U 4 U 5 U 6 U 7 U 8 U 9)*

⚠️ "0101" or "091" **are not** Base-10 numbers

All possible *strings* using numbers 0-9 that never start with 0

(1 U 2 U 3 U 4 U 5 U 6 U 7 U 8 U 9)(0 U 1 U 2 U 3 U 4 U 5 U 6 U 7 U 8 U 9)*

⚠️ "0" **is** a Base-10 number not considered

All possible *strings* using numbers 0-9 that never start with 0 *or* is 0

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

(0 U 1 U 2 U 3 U 4 U 5 U 6 U 7 U 8 U 9)*

⚠️ “0101” or “091” **are not** Base-10 numbers

All possible *strings* using numbers 0-9 that never start with 0

(1 U 2 U 3 U 4 U 5 U 6 U 7 U 8 U 9)(0 U 1 U 2 U 3 U 4 U 5 U 6 U 7 U 8 U 9)*

⚠️ “0” **is** a Base-10 number not considered

All possible *strings* using numbers 0-9 that never start with 0 *or* is 0

0 U ((1 U 2 U 3 U 4 U 5 U 6 U 7 U 8 U 9)(0 U 1 U 2 U 3 U 4 U 5 U 6 U 7 U 8 U 9)*)

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

(0 U 1 U 2 U 3 U 4 U 5 U 6 U 7 U 8 U 9)*

⚠️ "0101" or "091" **are not** Base-10 numbers

All possible *strings* using numbers 0-9 that never start with 0

(1 U 2 U 3 U 4 U 5 U 6 U 7 U 8 U 9)(0 U 1 U 2 U 3 U 4 U 5 U 6 U 7 U 8 U 9)*

⚠️ "0" **is** a Base-10 number not considered

All possible *strings* using numbers 0-9 that never start with 0 *or* is 0

0 U ((1 U 2 U 3 U 4 U 5 U 6 U 7 U 8 U 9)(0 U 1 U 2 U 3 U 4 U 5 U 6 U 7 U 8 U 9)*)

✅Generates only all possible Base-10 numbers

Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Write a regular expression that matches all base-3 numbers

Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Write a regular expression that matches all base-3 numbers

$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*)$

✅Generates only all possible Base-3 numbers

Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Write a regular expression that matches all base-3 numbers

$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*)$

Generates only all possible Base-3 numbers

…divisible by 3

Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Write a regular expression that matches all base-3 numbers

$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)^*)$

Generates only all possible Base-3 numbers

…divisible by 3

Hint: you know that Base-10 numbers are divisible by 10 when they end in 0 (10, 20, 30, 40…)

Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Write a regular expression that matches all base-3 numbers

$$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*)$$

Generates only all possible Base-3 numbers

…divisible by 3

Hint: you know that Base-10 numbers are divisible by 10 when they end in 0 (10, 20, 30, 40…)

$$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*\mathbf{0})$$

✅all possible Base-3 numbers divisible by 3

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

$(0 \cup 1)^* \, 111 \, (0 \cup 1)^*$

⚠️ The Kleene-star has us generating any number of 0’s

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

$(0 \cup 1)^* \, 111 \, (0 \cup 1)^*$

⚠️ The Kleene-star has us generating any number of 0’s

…without the substring “000”

Use careful case-work to modify this and produce only 0,1,or two 0’s

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring "111", but not the substring "000".

all binary strings that contain the substring "111"

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0's

…without the substring "000"

Use careful case-work to modify this and produce only 0,1,or two 0's

(0 ∪ 00 ∪ ε) (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

$(0 \cup 1)^* \, 111 \, (0 \cup 1)^*$

⚠️ The Kleene-star has us generating any number of 0’s

…without the substring “000”

Use careful case-work to modify this and produce only 0,1,or two 0’s

$(0 \cup 00 \cup \varepsilon)\,(1)^* \, 111 \, (0 \cup 00 \cup \varepsilon)\,(1)^*$

⚠️ Cannot produce 1’s with “0” or “00” like “<u>1</u>011101<u>1</u>”

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

$(0 \cup 1)^* 111 (0 \cup 1)^*$

⚠️ The Kleene-star has us generating any number of 0’s

…without the substring “000”

Use careful case-work to modify this and produce only 0,1,or two 0’s

$(0 \cup 00 \cup \varepsilon) (1)^* 111 (0 \cup 00 \cup \varepsilon) (1)^*$

⚠️ Cannot produce 1’s with “0” or “00” like “1011101”

$(0 \cup 00 \cup \varepsilon) (01 \cup 001 \cup 1)^* 111 (0 \cup 00 \cup \varepsilon) (01 \cup 001 \cup 1)^*$

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring "111", but not the substring "000".

all binary strings that contain the substring "111"

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0's

…without the substring "000"

Use careful case-work to modify this and produce only 0,1,or two 0's

(0 ∪ 00 ∪ ε) (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

⚠️ Cannot produce 1's with "0" or "00" like "1011101"

(0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)*** 111 (0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)***

⚠️ Generates "000" like "00 01 111"

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring "111", but not the substring "000".

all binary strings that contain the substring "111"

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0's

…without the substring "000"

Use careful case-work to modify this and produce only 0,1,or two 0's

(0 ∪ 00 ∪ ε) (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

⚠️ Cannot produce 1's with "0" or "00" like "1011101"

(0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)*** 111 (0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)***

⚠️ Generates "000" like "00 01 111"

(01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε) 111 (01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε)

✅ all binary strings with "111" and without "000"

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0’s

…without the substring “000”

Use careful case-work to modify this and produce only 0,1,or two 0’s

(0 ∪ 00 ∪ ε) (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

⚠️ Cannot produce 1’s with “0” or “00” like “1011101”

(0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)*** 111 (0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)***

⚠️ Generates “000” like “00 01 111”

(01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε) 111 (01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε)

✅ all binary strings with “111” and without “000”

(01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε) 111 (01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε)

Context-Free Grammars

Context-Free Grammars

A context free grammar (CFG) is a finite set of production rules over:

- An alphabet Σ of "terminal symbols"
- A finite set V of "nonterminal symbols"
- A start symbol (one of the elements of V) usually denoted S

A production rule for a nonterminal $A \in V$ takes the form

- $A \rightarrow w1 \mid w2 \mid \ldots \mid wk$

Where each $wi \in V \cup \Sigma^*$ is a string of nonterminals and terminals.

Problem 2 – CFGs

Write a context-free grammar to match each of these languages.

a) All binary strings that start with 11

d) All binary strings that contain at least three 1's

e) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.
(^bonus)

Work on this problem with the people around you.

Problem 2 – CFGs

a) All binary strings that start with 11.

Problem 2 – CFGs

a) All binary strings that start with 11.

Thinking back to regular expressions…

Problem 2 – CFGs

a) All binary strings that start with 11.

Thinking back to regular expressions…

$11\ (0 \cup 1)^*$

Problem 2 – CFGs

a) All binary strings that start with 11.

Thinking back to regular expressions…

$11\ (0 \cup 1)^*$

Now generate the CFG…

Problem 2 – CFGs

a) All binary strings that start with 11.

Thinking back to regular expressions…

$11\ (0 \cup 1)^*$

Now generate the CFG…

$S \rightarrow 11T$
$T \rightarrow 1T \mid 0T \mid \varepsilon$

Problem 2 – CFGs

d) All binary strings that contain at least three 1's

Problem 2 – CFGs

d) All binary strings that contain at least three 1's

Thinking back to Regular expressions…

$(1 \cup \mathbf{0})^* \, 1 \, (1 \cup \mathbf{0})^* \, 1 \, (1 \cup \mathbf{0})^* \, 1 \, (1 \cup \mathbf{0})^*$

Now generate the CFG…

S → TTT

T → 0T | T0 | 1T | 1

Problem 2 – CFGs

d) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

Problem 2 – CFGs

e) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

Strings to Consider:

000111**2** ← **beware!**
20101
01**2**10
2

Problem 2 – CFGs

e) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

$$\mathbf{S} \rightarrow 01\mathbf{S} \mid 10\mathbf{S} \mid 0\mathbf{S}1 \mid 1\mathbf{S}0 \mid \mathbf{S}01 \mid \mathbf{S}10 \mid 2$$

Problem 2 – CFGs

e) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

~~**S** → 01**S** | 10**S** | 0**S**1 | 1**S**0 | **S**01 | **S**10 | 2~~

Counter example: 001121100

Problem 2 – CFGs

e) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

~~**S** → 01**S** | 10**S** | 0**S**1 | 1**S**0 | **S**01 | **S**10 | 2~~

Counter example: 001121100

Correct Answer:

S → 2 | **ST** | **TS** | 0**S**1 | 1**S**0

T → **TT** | 0**T**1 | 1**T**0 | **ε**

Relations

R is **reflexive** iff $(a,a) \in R$ for every $a \in A$

R is **symmetric** iff $(a,b) \in R$ implies $(b, a) \in R$

R is **antisymmetric** iff $(a,b) \in R$ and $a \neq b$ implies $(b,a) \notin R$

R is **transitive** iff $(a,b) \in R$ and $(b, c) \in R$ implies $(a, c) \in R$

Problem 3b

Let $R = \{(x, y) \ : \ x^2 = y^2\}$ on $\mathbb{R}$.

Problem 3b

Let $R = \{(x, y) \ : \ x^2 = y^2\}$ on $\mathbb{R}$.

reflexive, symmetric, not antisymmetric (counterexample: $(-2, 2) \in R$ and $(2, -2) \in R$ but $2 \neq -2$), transitive

Problem 4b

Prove that $R \subseteq R^2$. (Remember that R^2 is defined to be $R \circ R$.)

Problem 4b

Prove that $R \subseteq R^2$. (Remember that R^2 is defined to be $R \circ R$.)

Let x and y be arbitrary. Suppose $(x, y) \in R$.

Since x and y were arbitrary, by definition of subset $R \subseteq R^2$.

Problem 4b

Prove that $R \subseteq R^2$. (Remember that R^2 is defined to be $R \circ R$.)

Let x and y be arbitrary. Suppose $(x, y) \in R$.

So by definition of relation composition, it follows that $(x, y) \in R \circ R = R^2$.

Since x and y were arbitrary, by definition of subset $R \subseteq R^2$.

Problem 4b

Prove that $R \subseteq R^2$. (Remember that R^2 is defined to be $R \circ R$.)

Let x and y be arbitrary. Suppose $(x, y) \in R$.

Since R is reflexive, we know $(y, y) \in R$ as well.

So by definition of relation composition, it follows that $(x, y) \in R \circ R = R^2$.

Since x and y were arbitrary, by definition of subset $R \subseteq R^2$.

Problem 4b

Prove that $R \subseteq R^2$. (Remember that R^2 is defined to be $R \circ R$.)

Let x and y be arbitrary. Suppose $(x, y) \in R$.

Since R is reflexive, we know $(y, y) \in R$ as well.

In other words, there is a z (namely y) such that $(x, z) \in R$ and $(z, y) \in R$.

So by definition of relation composition, it follows that $(x, y) \in R \circ R = R^2$.

Since x and y were arbitrary, by definition of subset $R \subseteq R^2$.

That's All, Folks!

Thanks for coming to section this week!
Any questions?

By: Aruna Srivastava