CSE 311 Section 05

Regular Expressions, CFGs, FSMs, Relations

Administrivia

Announcements & Reminders

- Homework 5 due @ 11:00 pm on Friday (2/27)
- Quiz 5 will be on Tuesday (3/3)
- Check your section participation grade on canvas
  - If it is different than what you expect, let your TA know

Regular Expressions

Regular Expressions

ε matches only the **empty string**

a matches only the one-character string *a*

A ∪ **B** matches all strings that either **A** matches or **B** matches (or both)

AB matches all strings that have a first part that **A** matches followed by a second part that **B** matches

A* matches all strings that have any number of strings (even 0) that **A** matches, one after another (**ε** ∪ **A** ∪ **AA** ∪ **AAA** ∪ **...**)

Definition of the *language* matched by a regular expression

Task 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".

We will do (b) together, then work on c

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

Task 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:

Task 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:

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

Task 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

Task 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)*

⚠️ "<u>0</u>101" or "<u>0</u>91" **are not** Base-10 numbers

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

Task 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)*

Task 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

Task 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

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

Task 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

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

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

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

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

Task 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

Task 1 – Regular Expressions

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

Task 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

Task 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

Task 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

Task 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-<u>10</u> numbers are divisible by <u>10</u> when <u>they end in 0</u> (10, 20, 30, 40…)

Task 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

Task 1 – Regular Expressions

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

Task 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”

Task 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

Task 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

Task 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)*

Task 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”

Task 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)***

Task 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”

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

⚠️ Generates “000” like “00 01 111”

Task 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”

Task 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

Always think back to Regex!

- CFG to match RE $A \cup B$

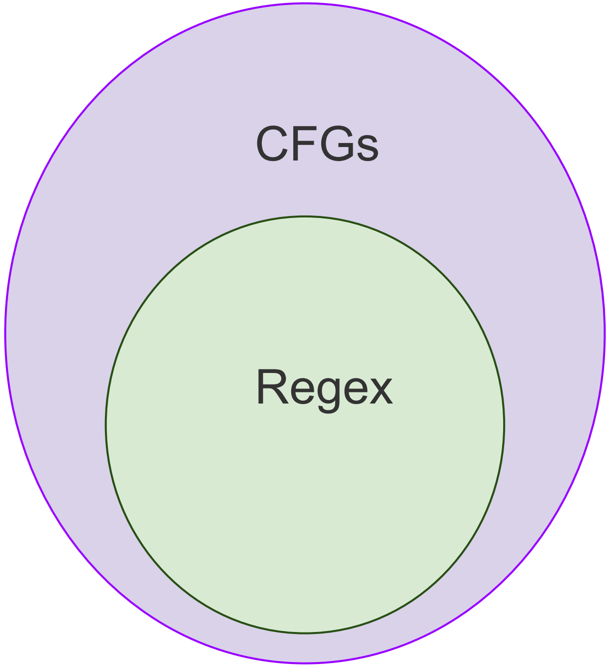
  $S \rightarrow S_1 \mid S_2$ + rules from original CFGs

- CFG to match RE AB

  $S \rightarrow S_1 S_2$ + rules from original CFGs

- CFG to match RE A^* $(= \varepsilon \cup A \cup AA \cup AAA \cup \ldots)$

  $S \rightarrow S_1 S \mid \varepsilon$ + rules from CFG with S_1



Always think back to Regex!

- CFG to match RE **A ∪ B**

  $S \rightarrow S_1 \mid S_2$ + rules from original CFGs

- CFG to match RE **AB**

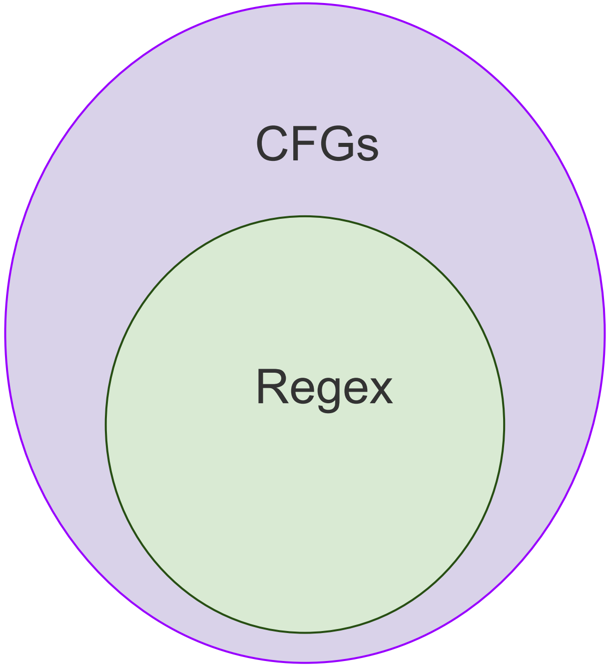
  $S \rightarrow S_1 S_2$ + rules from original CFGs

- CFG to match RE A^* (= $\varepsilon \cup A \cup AA \cup AAA \cup \ldots$)

  $S \rightarrow S_1 S \mid \varepsilon$ + rules from CFG with S_1

CFG or Regex?

"equal number of 0's and 1's" (ex. 011010)



Always think back to Regex!

- CFG to match RE **A ∪ B**

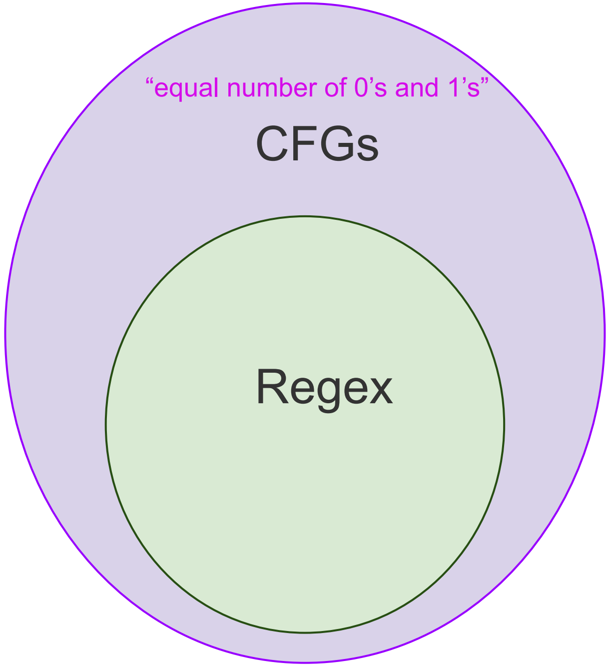
  $S \rightarrow S_1 \mid S_2$ + rules from original CFGs

- CFG to match RE **AB**

  $S \rightarrow S_1 S_2$ + rules from original CFGs

- CFG to match RE **A***   (= ε ∪ **A** ∪ **AA** ∪ **AAA** ∪ …)

  $S \rightarrow S_1 S \mid \varepsilon$ + rules from CFG with S_1



Task 2 – CFGs

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

a) All binary strings that start with 11.

b) All binary strings that contain at most one 1.

Work on this problem with the people around you.

Task 2 – CFGs

a) All binary strings that start with 11.

Task 2 – CFGs

a) All binary strings that start with 11.

Thinking back to regular expressions…

Task 2 – CFGs

a) All binary strings that start with 11.

Thinking back to regular expressions…

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

Task 2 – CFGs

a) All binary strings that start with 11.

Thinking back to regular expressions…

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

Now generate the CFG…

Task 2 – CFGs

a) All binary strings that start with 11.

Thinking back to regular expressions…

11 (0 ∪ 1)*

Now generate the CFG…

S → 11**T**
T → 1**T** | 0**T** | **ε**

Task 2 – CFGs

b) All binary strings that contain at most one 1.

Task 2 – CFGs

b) All binary strings that contain at most one 1.

Thinking back to Regular expressions…

Task 2 – CFGs

b) All binary strings that contain at most one 1.

Thinking back to Regular expressions…

$0^* (1 \cup \boldsymbol{\varepsilon})\ 0^*$

Task 2 – CFGs

b) All binary strings that contain at most one 1.

Thinking back to Regular expressions…

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

Now generate the CFG…

Task 2 – CFGs

b) All binary strings that contain at most one 1.

Thinking back to Regular expressions…

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

Now generate the CFG…

$S \rightarrow ABA$
$A \rightarrow 0A \mid \varepsilon$
$B \rightarrow 1 \mid \varepsilon$

Task 2 – CFGs

b) All binary strings that contain at most one 1.

Thinking back to Regular expressions…

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

Now generate the CFG…

$S \rightarrow ABA$
$A \rightarrow 0A \mid \varepsilon$
$B \rightarrow 1 \mid \varepsilon$

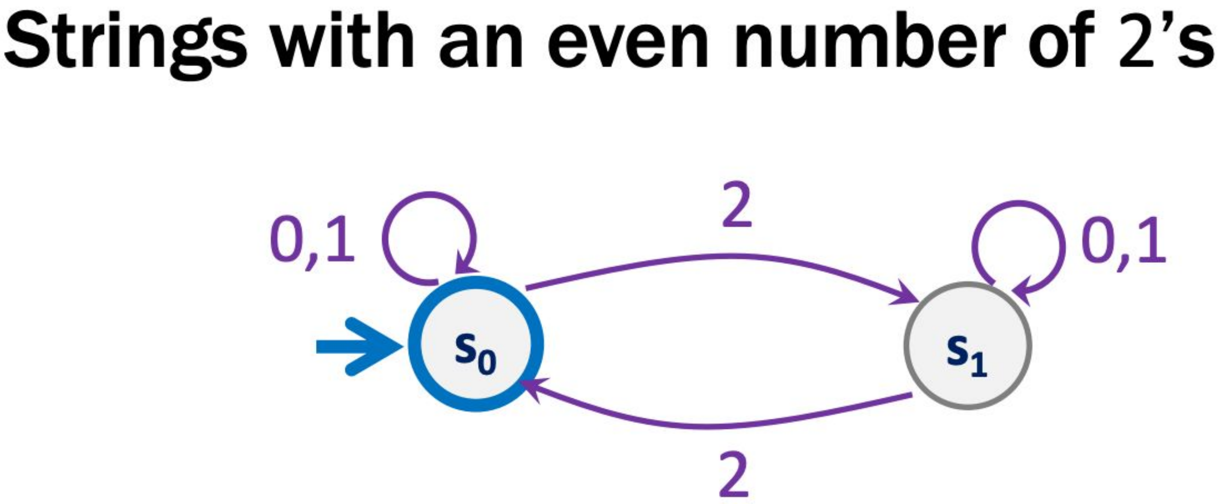
Alternative solution:

$S \rightarrow 0S \mid S0 \mid 1 \mid 0 \mid \varepsilon$

Finite State Machines

Finite State Machines

- An FSA is a finite-state machine that accepts or rejects a given string of symbols, by running through a state sequence uniquely determined by the string.



- An edge for every symbol in the language
- Every reject and accept state is handled

Task 3 – FSM Design

Construct FSMs to recognize each of the following languages.

Let $\Sigma = \{0, 1, 2, 3\}$.
b) All strings whose digits sum to an even number.

Let $\Sigma = \{0, 1\}$.
c) All strings that do not contain the substring 101.

Work on this problem with the people around you.

Task 3 – FSM Design

Let Σ = {0, 1, 2, 3}.

b) All strings whose digits sum to an even number.

Hint: start with just the language of 1’s

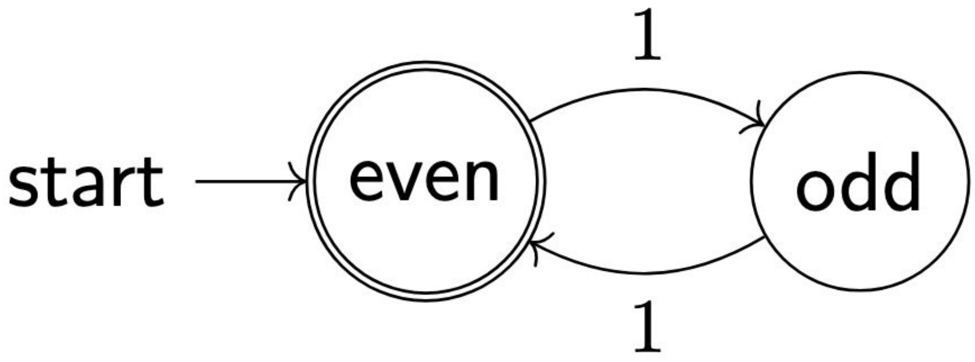
Task 3 – FSM Design

Let Σ = {0, 1, 2, 3}.

b) All strings whose digits sum to an even number.

Hint: start with just the language of 1’s

If we had Σ = {1}

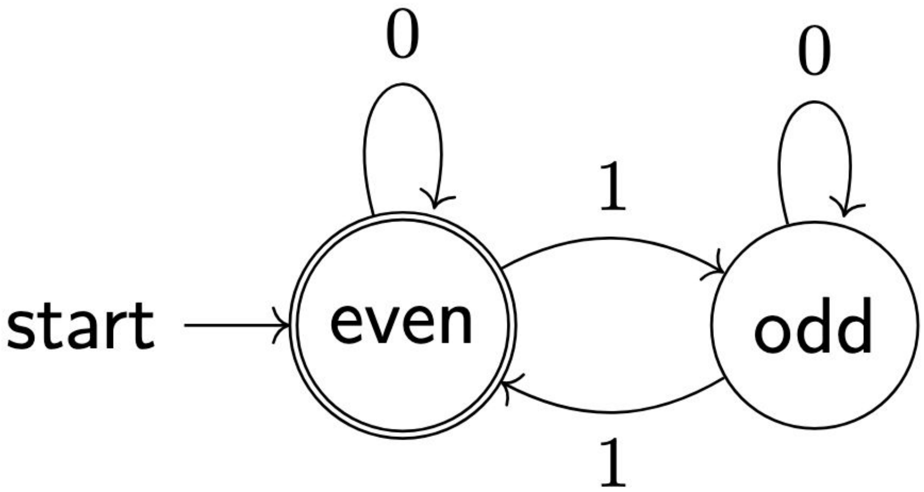


Task 3 – FSM Design

Let Σ = {0, 1, 2, 3}.

b) All strings whose digits sum to an even number.

If we had Σ = **{1,0}**

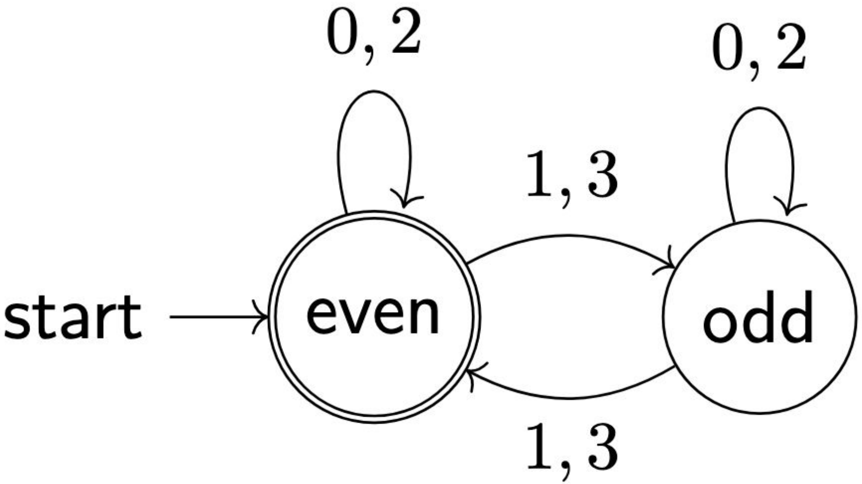


Task 3 – FSM Design

Let Σ = {0, 1, 2, 3}.

b) All strings whose digits sum to an even number.

If we had Σ = **{0, 1, 2, 3}**



Task 3 - FSM Design

Let Σ = {0, 1}.

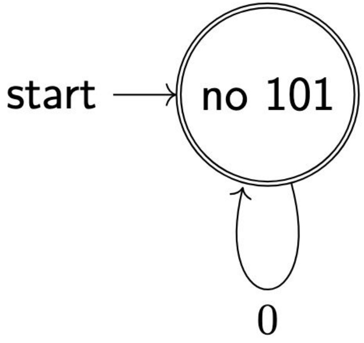
c) All strings that do not contain the substring 101.

Task 3 - FSM Design

Let Σ = {0, 1}.

c) All strings that do not contain the substring 101.

Let's start somewhere. The initial state should be valid because we couldn't have seen 101 yet. Another 0 does not even begin to start the substring 101.

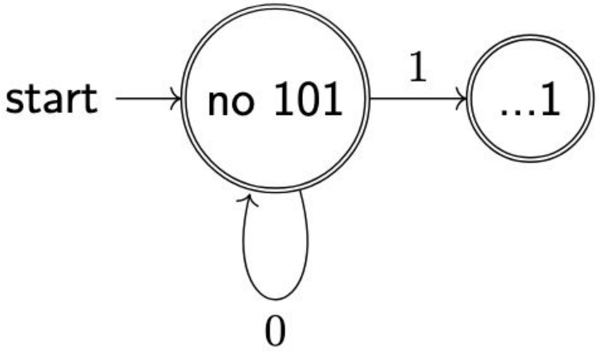


Task 3 - FSM Design

Let Σ = {0, 1}.

c) All strings that do not contain the substring 101.

A 1 might be the start of the substring 101, but still a valid state at that point. Let's create a new state for it.

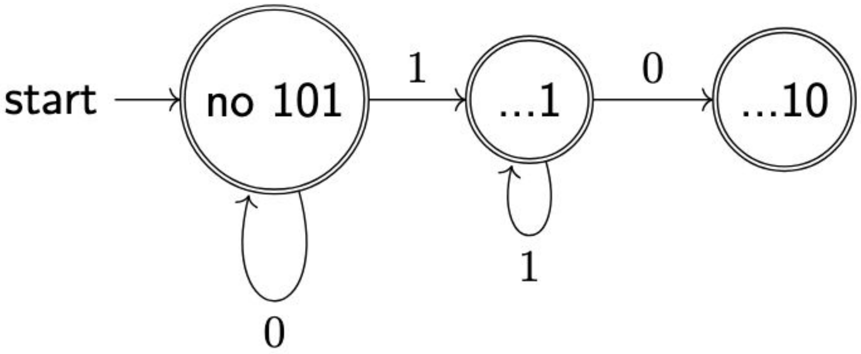


Task 3 - FSM Design

Let Σ = {0, 1}.

c) All strings that do not contain the substring 101.

If we see another 1, that resets the substring 101. If we see a 0, it means we're closer to seeing the substring 101, so let's create a new state for it as well.

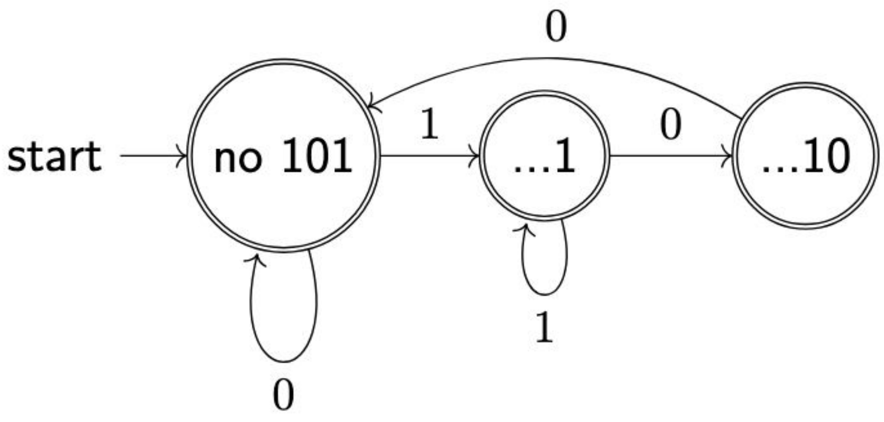


Task 3 - FSM Design

Let Σ = {0, 1}.

c) All strings that do not contain the substring 101.

If we see a 0, the substring matching is completely reset, let’s return back to the initial state.

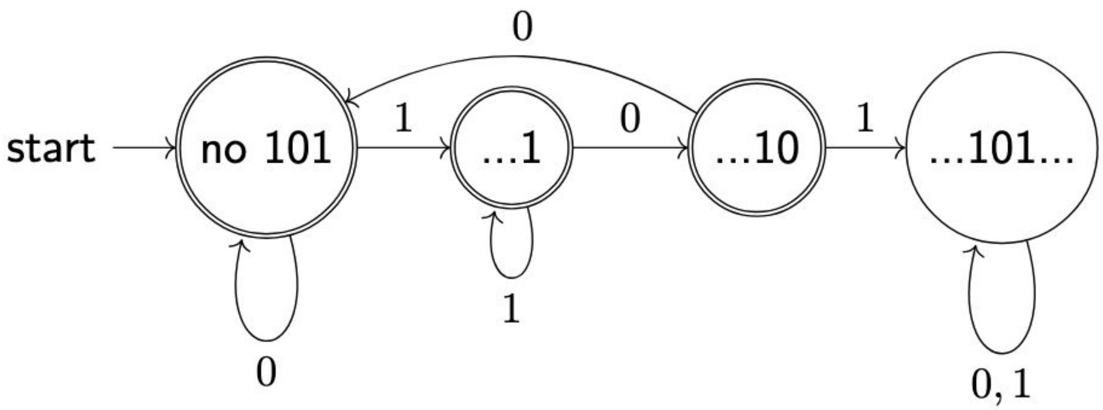


Task 3 - FSM Design

Let Σ = {0, 1}.

c) All strings that do not contain the substring 101.

If we see a 1, the string contains the substring 101. This is a trap state since no matter what comes after, the string will still be invalid.

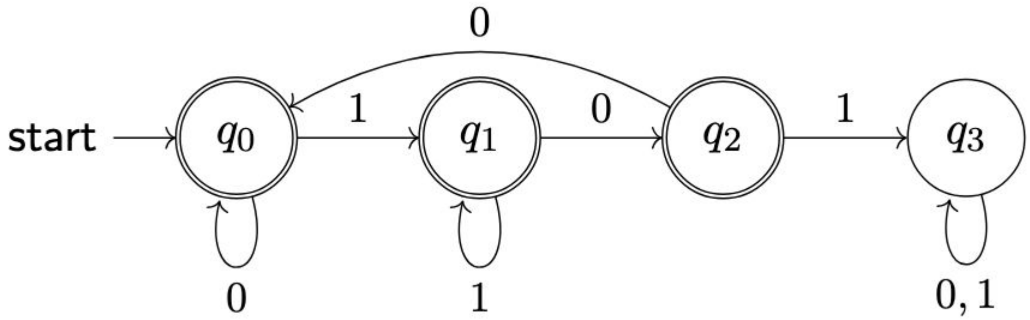


Task 3 - FSM Design

Let Σ = {0, 1}.

c) All strings that do not contain the substring 101.

Finished! Let's come up with better state descriptions.



q_0: ε, 0, and strings that don't contain 101 and end in 00

q_1: strings that don't contain 101 and end in 1

q_2: strings that don't contain 101 and end in 10

q_3: strings that contain 101

Relations

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 $(b,a) \notin R$

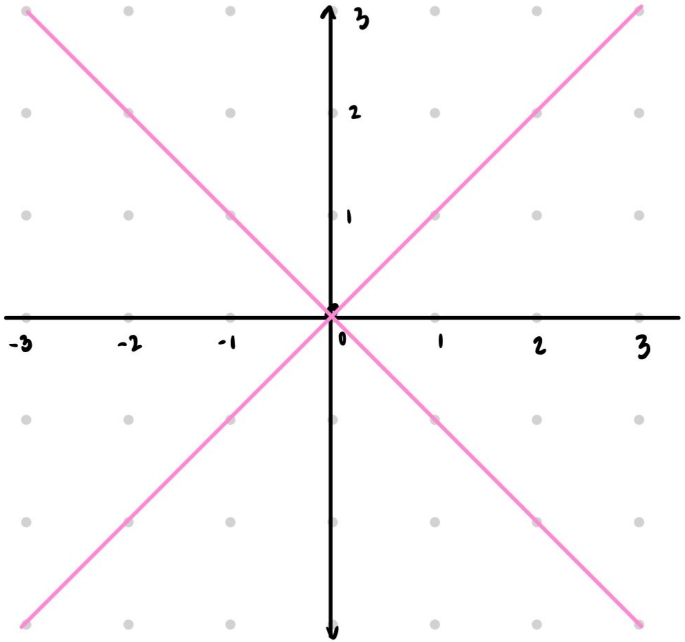
R is **transitive** iff $(a,b) \in R$ and $(b, c) \in R$ implies $(a, c) \in R$

Task 4b

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

Task 4b

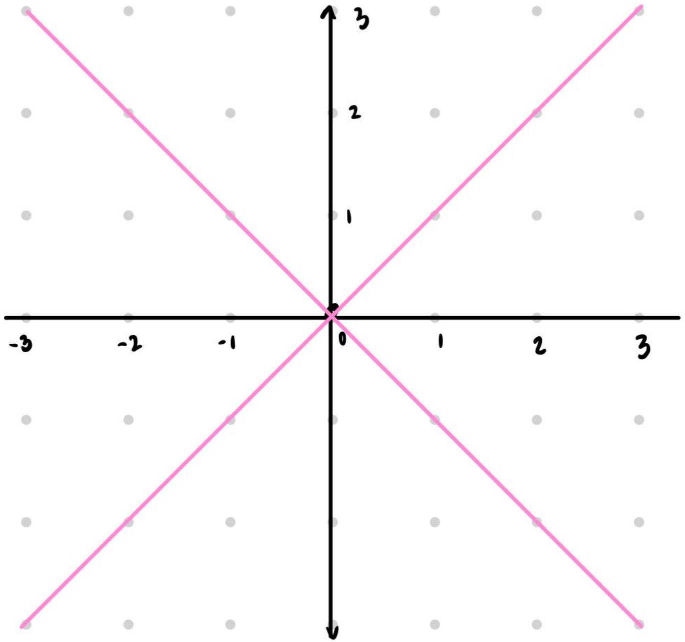
Let $R = \{(x, y) \; : \; x^2 = y^2\}$ on $\mathbb{R}$.



We can graph the points of R

Task 4b

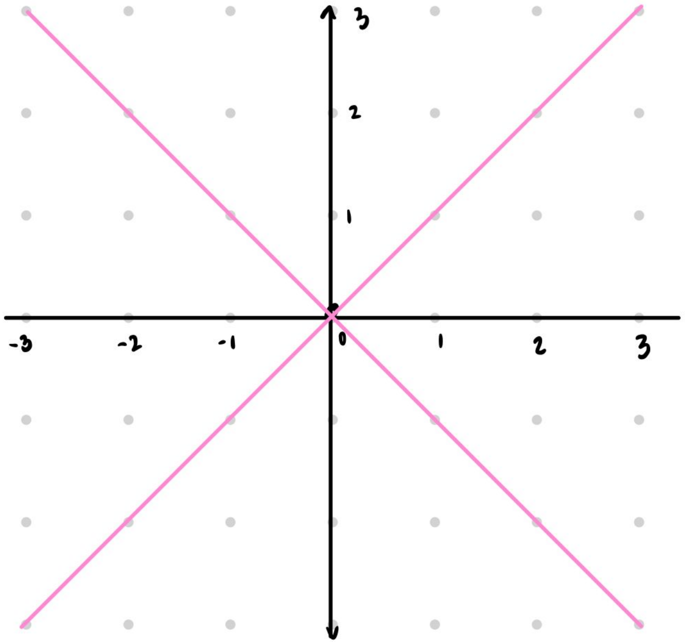
Let $R = \{(x, y) \ : \ x^2 = y^2\}$ on $\mathbb{R}$.



If all points on the line of y = x are in the relation then the relation is reflexive

Task 4b

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

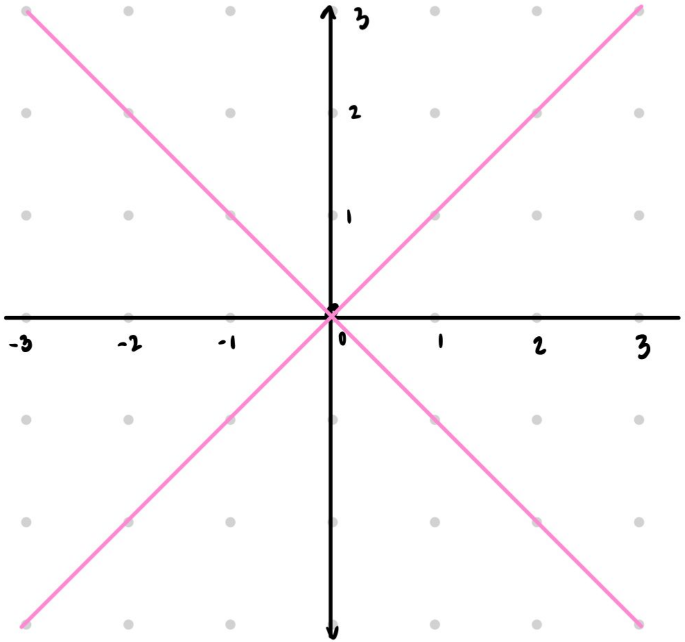


If all points on the line of y = x are in the relation then the relation is reflexive

The relation is reflexive!

Task 4b

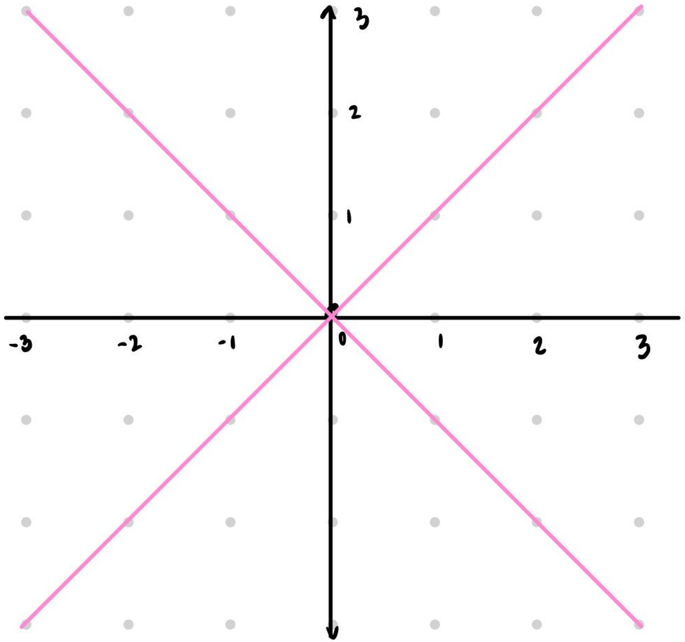
Let $R = \{(x, y) \ : \ x^2 = y^2\}$ on $\mathbb{R}$.



If all points that are reflected across y = x are also in the relation, then the relation is symmetric

Task 4b

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

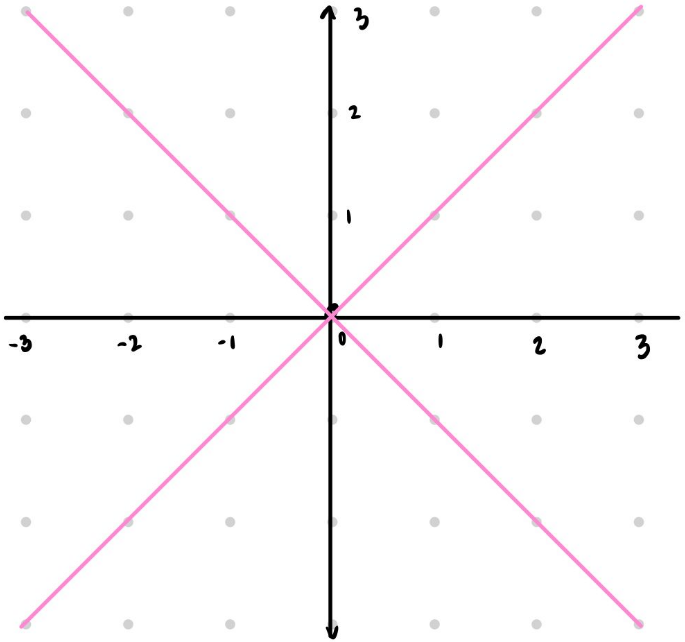


If all points that are reflected across y = x are also in the relation, then the relation is symmetric

The relation is symmetric!

Task 4b

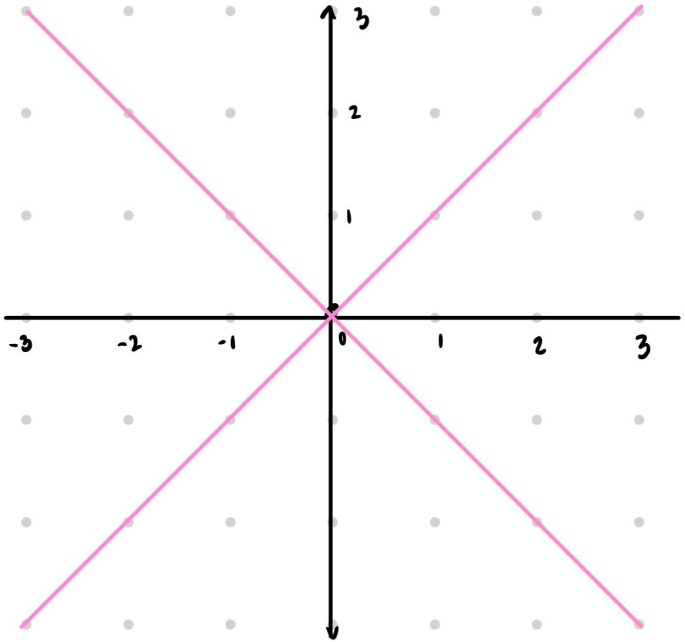
Let $R = \{(x, y) \ : \ x^2 = y^2\}$ on $\mathbb{R}$.



If all points that are reflected across y = x are not in the relation, then the relation is antisymmetric

Task 4b

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



If all points that are reflected across y = x are not in the relation, then the relation is antisymmetric

The relation is not antisymmetric!

Task 4b

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

Task 4a

Let $R = \{(x, y) \ : \ x = y + 1\}$ on $\mathbb{N}$.

Task 4a

Let $R = \{(x, y) \ : \ x = y + 1\}$ on $\mathbb{N}$.

Is it reflexive?

Task 4a

Let $R = \{(x, y) \ : \ x = y + 1\}$ on $\mathbb{N}$.

Is it reflexive?

No, not all points on the line of y = x are in the relation. For example, (1,1) is not in the relation.

Task 4a

Let $R = \{(x, y) \ : \ x = y + 1\}$ on $\mathbb{N}$.

Is it symmetric?

Task 4a

Let $R = \{(x, y) \ : \ x = y + 1\}$ on $\mathbb{N}$.

Is it symmetric?

No, not all points that are reflected across y = x are in the relation. For example, (2,1) is in the relation (2=1+1), but (1,2) isn't (1≠2+1)

Task 4a

Let $R = \{(x, y) \ : \ x = y + 1\}$ on $\mathbb{N}$.

Is it antisymmetric?

Task 4a

Let $R = \{(x, y) \ : \ x = y + 1\}$ on $\mathbb{N}$.

Is it antisymmetric?

Yes, all points that are reflected across y = x are not in the relation.

Task 4a

Let $R = \{(x, y) \ : \ x = y + 1\}$ on $\mathbb{N}$.

Is it transitive?

Task 4a

Let $R = \{(x, y) \ : \ x = y + 1\}$ on $\mathbb{N}$.

Is it transitive?

No,for example, (3,2) and (2,1) are both in the relation (3=2+1 and 2=1+1), but (3,1) is not in the relation (3≠1+1)

Task 4a

Let $R = \{(x, y) \ : \ x = y + 1\}$ on $\mathbb{N}$.

not reflexive (counterexample: $(1,1) \notin R$), not symmetric (counterexample: $(2,1) \in R$ but $(1,2) \notin R$), antisymmetric, not transitive (counterexample: $(3,2) \in R$ and $(2,1) \in R$, but $(3,1) \notin R$

That’s All, Folks!

Thanks for coming to section this week!
Any questions?