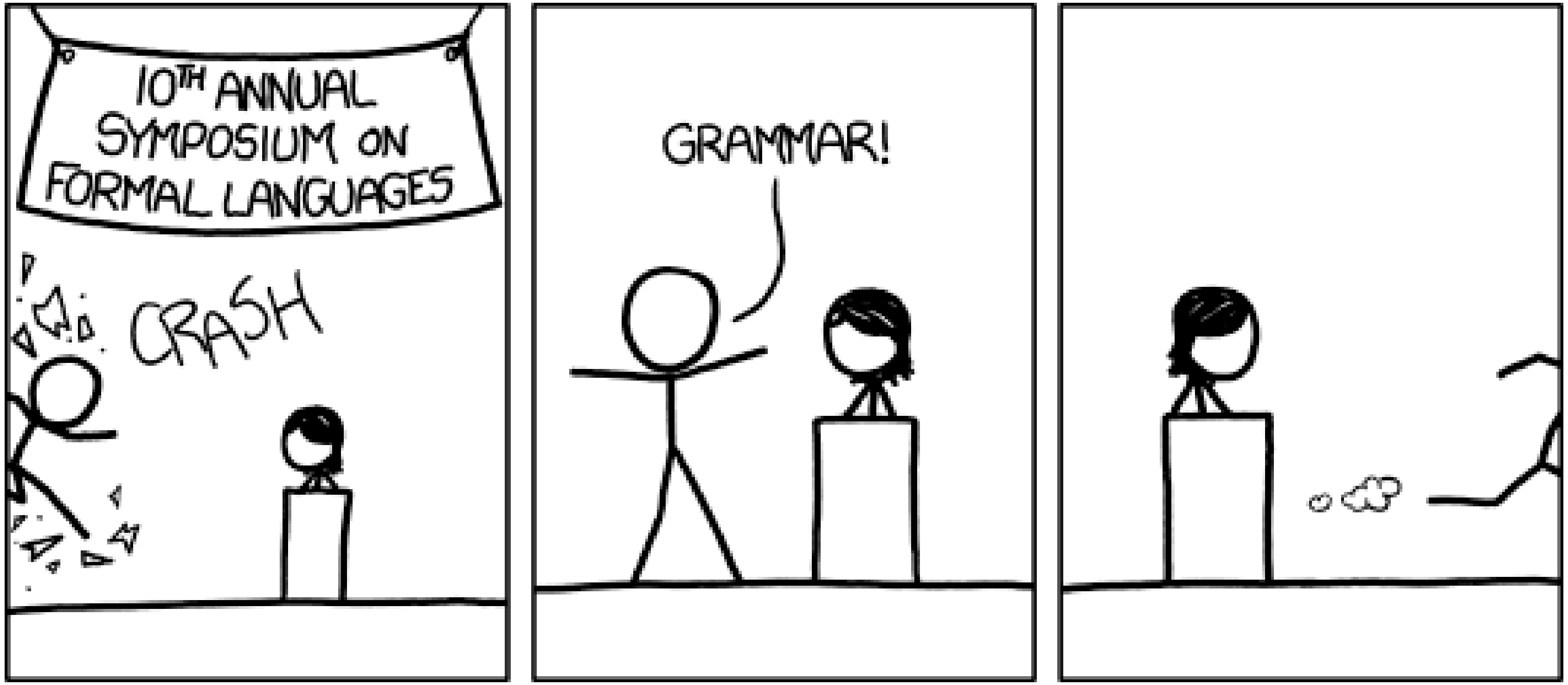


[Audience looks around] “What just happened?” “There must be some context we’re missing.”
xkcd.com/1090

Context Free Grammars

CSE 311 Winter 2023
Lecture 22

Context Free Grammars

What Can't Regular Expressions Do?

Some "easy" things

Where you could say whether a string matches with just a loop

$\{0^k 1^k : k \geq 0\}$

The set of all palindromes.

And some harder things

Expressions with matched parentheses

Properly formed arithmetic expressions

Context Free Grammars can solve all of these problems!

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

$A \to w_1|w_2|\cdots|w_k$

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

Context Free Grammars

We think of context free grammars as **generating** strings.

1. Start from the start symbol S.

2. Choose a nonterminal in the string, and a production rule $A \to w_1|w_2|\ldots|w_k$ replace that copy of the nonterminal with w_i.

3. If no nonterminals remain, you're done! Otherwise, goto step 2.

A string is in the language of the CFG iff it can be generated starting from S.

Notation: $xAy \Rightarrow xwy$ is rewriting A with w.

Examples

$$S \to 0S0|1S1|0|1|\varepsilon$$

$$S \to 0S|S1|\varepsilon$$

$$S \to (S)|SS|\varepsilon$$

The alphabet here is {(,)} i.e. parentheses are the characters.

$$S \to AB$$

$$A \to 0A1|\varepsilon$$

$$B \to 1B0|\varepsilon$$

Examples

$S \rightarrow 0S0|1S1|0|1|\varepsilon$

The set of all binary palindromes

$S \rightarrow 0S|S1|\varepsilon$

The set of all strings with any 0's coming before any 1's (i.e. 0^*1^*)

$S \rightarrow (S)|SS|\varepsilon$

Balanced parentheses

$S \rightarrow AB$

$A \rightarrow 0A1|\varepsilon$

$B \rightarrow 1B0|\varepsilon$ $\{0^j 1^{j+k} 0^k : j, k \geq 0\}$

Arithmetic

$E \to E + E | E * E | (E) | x | y | z | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$

Generate $(2 * x) + y$

Generate $2 + 3 * 4$ in two different ways

pollev.com/robbie

Arithmetic

$E \rightarrow E + E|E * E|(E)|x|y|z|0|1|2|3|4|5|6|7|8|9$

Generate $(2 * x) + y$

$E \Rightarrow E + E \Rightarrow (E) + E \Rightarrow (E * E) + E \Rightarrow (2 * E) + E \Rightarrow (2 * x) + E \Rightarrow (2 * x) + y$

Generate $2 + 3 * 4$in two different ways

$E \Rightarrow E + E \Rightarrow E + E * E \Rightarrow 2 + E * E \Rightarrow 2 + 3 * E \Rightarrow 2 + 3 * 4$

$E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow 2 + E * E \Rightarrow 2 + 3 * E \Rightarrow 2 + 3 * 4$

Parse Trees

Suppose a context free grammar G generates a string x

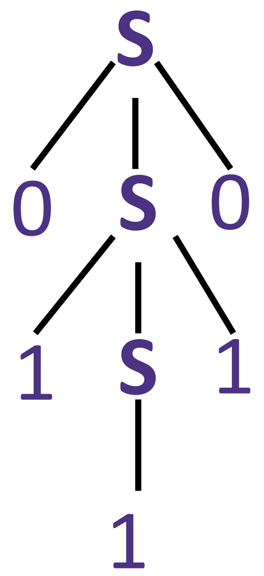
A parse tree of x for G has

Rooted at S (start symbol)

Children of every A node are labeled with the characters of w for some $A \to w$

Reading the leaves from left to right gives x.

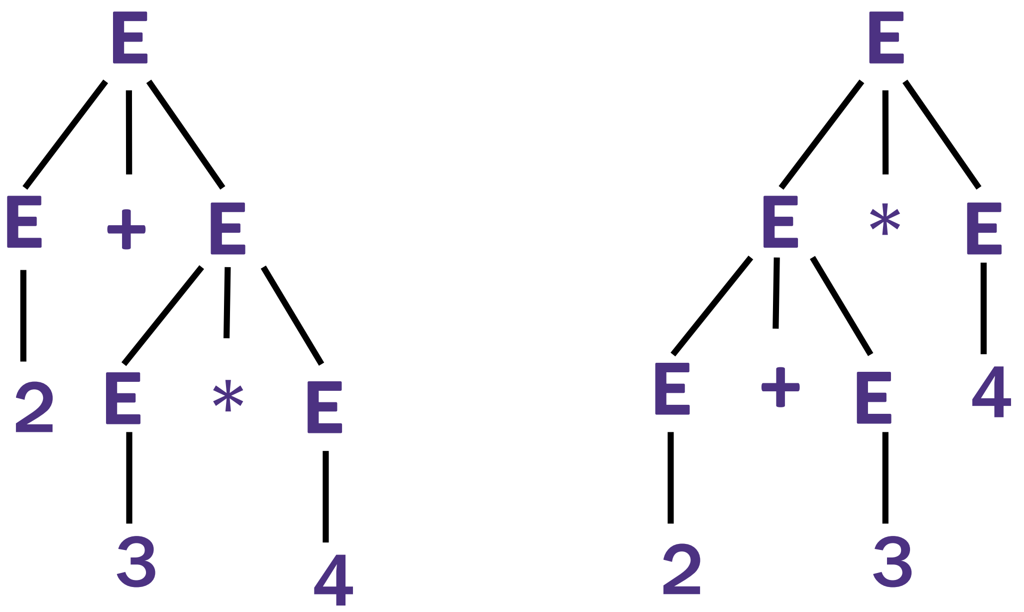
$S \to 0S0|1S1|0|1|\varepsilon$



Back to the arithmetic

$$E \to E + E | E * E | (E) | x | y | z | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$$

Two parse trees for $2 + 3 * 4$



How do we encode order of operations

If we want to keep "in order" we want there to be only one possible parse tree.

Differentiate between "things to add" and "things to multiply"

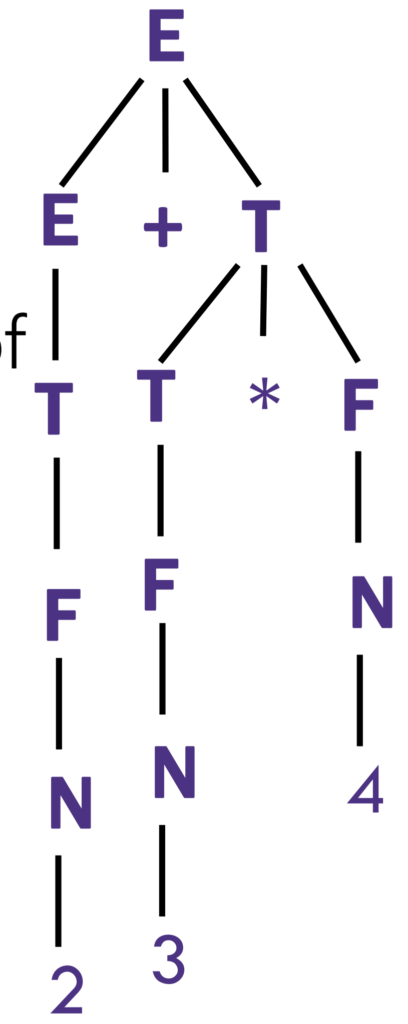
Only introduce a * sign after you've eliminated the possibility of introducing another + sign in that area.

$E \rightarrow T|E + T$

$T \rightarrow F|T * F$

$F \rightarrow (E)|N$

$N \rightarrow x|y|z|0|1|2|3|4|5|6|7|8|9$



CNFs in practice

Used to define programming languages.

Often written in Backus-Naur Form – just different notation

Variables are <names-in-brackets> or technical terms

like <if-then-else-statement>, <condition>, <identifier>

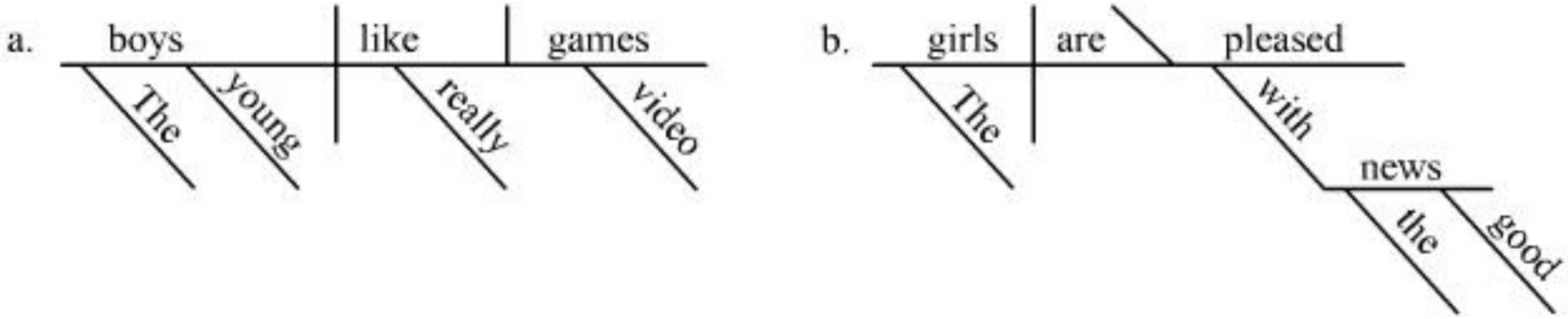
→ is replaced with ::= or :

BNF for C (no <…> and uses : instead of ::=)

```
statement:
  ((identifier | "case" constant-expression | "default") ":")*
  (expression? ";" |
   block |
   "if" "(" expression ")" statement |
   "if" "(" expression ")" statement "else" statement |
   "switch" "(" expression ")" statement |
   "while" "(" expression ")" statement |
   "do" statement "while" "(" expression ")" ";" |
   "for" "(" expression? ";" expression? ";" expression? ")" statement |
   "goto" identifier ";" |
   "continue" ";" |
   "break" ";" |
   "return" expression? ";"
  )

block: "{" declaration* statement* "}"

expression:
  assignment-expression%

assignment-expression: (
    unary-expression (
      "=" | "*=" | "/=" | "%=" | "+=" | "-=" | "<<=" | ">>=" | "&=" |
      "^=" | "|="
    )
  )* conditional-expression

conditional-expression:
  logical-OR-expression ( "?" expression ":" conditional-expression )?
```

Parse Trees

Remember diagramming sentences in middle school?



<sentence>::=<noun phrase><verb phrase>

<noun phrase>::=<determiner><adjective><noun>

<verb phrase>::=<verb><adverb>|<verb><object>

<object>::=<noun phrase>

Parse Trees

<sentence>::=<noun phrase><verb phrase>

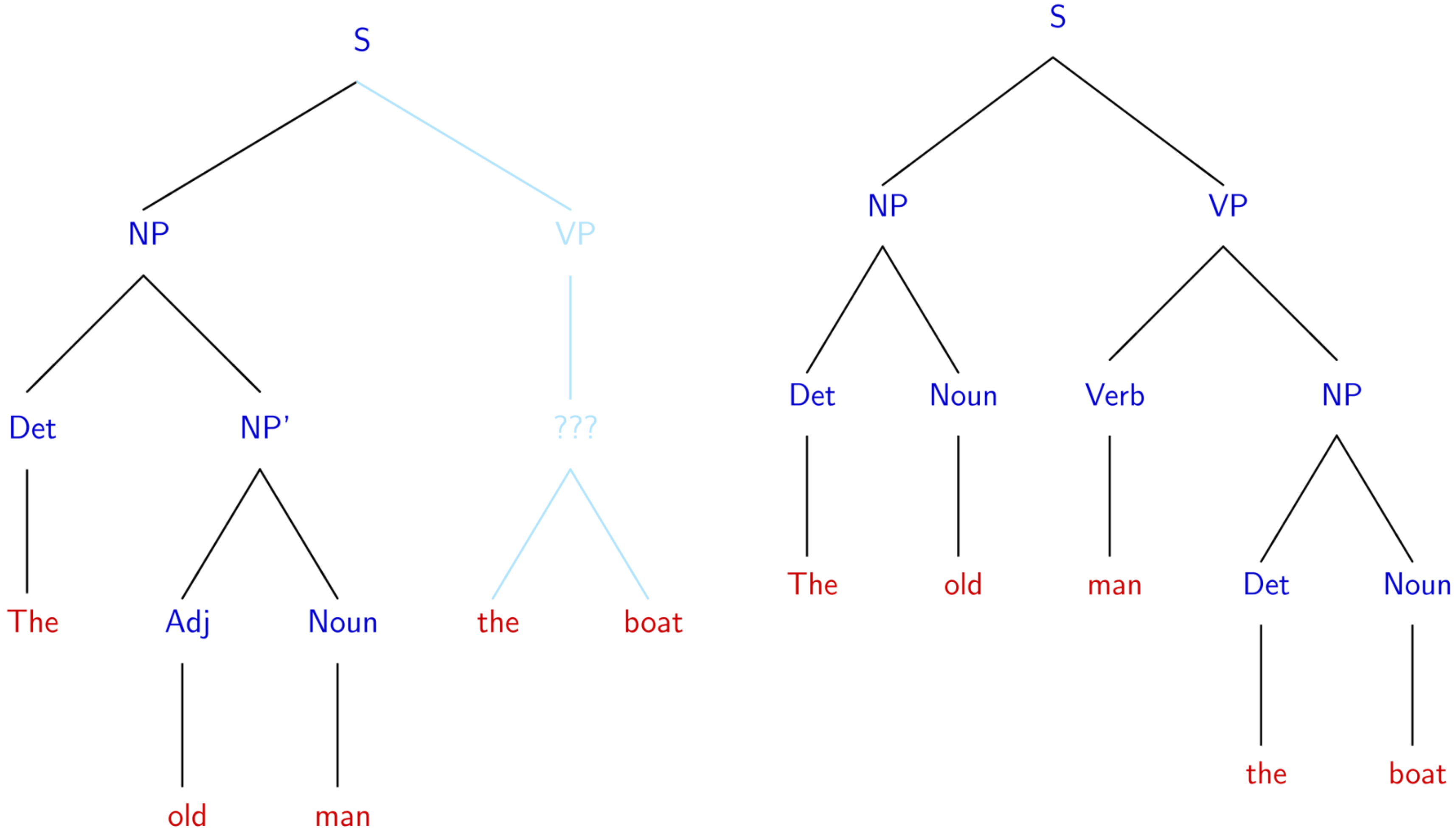
<noun phrase>::=<determiner><adjective><noun>

<verb phrase>::=<verb><adverb>|<verb><object>

<object>::=<noun phrase>

The old man the boat.

The old man the boat



Power of Context Free Languages

There are languages CFGs can express that regular expressions can't

e.g. palindromes

What about vice versa – is there a language that a regular expression can represent that a CFG can't?

No!

Are there languages even CFGs cannot represent?

Yes!

$\{0^k 1^j 2^k 3^j | j, k \geq 0\}$ cannot be written with a context free grammar.

Takeaways

CFGs and regular expressions gave us ways of succinctly representing sets of strings

- Regular expressions super useful for representing things you need to search for
- CFGs represent complicated languages like "java code with valid syntax"

Next Week, we'll talk about how each of these are "equivalent to weaker computers."

This week: Two more tools for our toolbox.

Relations and Graphs

Relations

Relations

A (binary) relation from A to B is a subset of $A \times B$
A (binary) relation on A is a subset of $A \times A$

Wait what?

$\leq$ is a relation on $\mathbb{Z}$.

"$3 \leq 4$" is a way of saying "3 relates to 4" (for the $\leq$ relation)

$(3,4)$ is an element of the set that defines the relation.

Relations, Examples

It turns out, they've been here the whole time

$<$ on $\mathbb{R}$ is a relation

I.e. $\{(x, y) : x < y \text{ and } x, y \in \mathbb{R}\}$.

$=$ on Σ^* is a relation

i.e. $\{(x, y) : x = y \text{ and } x, y \in \Sigma^*\}$

For your favorite function f, you can define a relation from its domain to its co-domain

i.e. $\{(x, y) : f(x) = y\}$

"x when squared gives y" is a relation

i.e. $\{(x, y): x^2 = y, x, y \in \mathbb{R}\}$

Relations, Examples

Fix a universal set $\mathcal{U}$.

$\subseteq$ is a relation. What's it on?

$\mathcal{P}(\mathcal{U})$
The set of all subsets of $\mathcal{U}$

More Relations

$R_1 = \{(a,1),(a,2),(b,1),(b,3),(c,3)\}$

Is a relation (you can define one just by listing what relates to what)

Equivalence mod 5 is a relation.

$\{(x,y) : x \equiv y (mod\ 5)\}$

We'll also say "x relates to y if and only if they're congruent mod 5"

Properties of relations

What do we do with relations? Usually we prove properties about them.

Symmetry

A binary relation R on a set S is "symmetric" iff
for all $a, b \in S$, $[(a, b) \in R \rightarrow (\mathbf{b}, \mathbf{a}) \in R]$

$=$ on Σ^* is symmetric, for all $a, b \in \Sigma^*$ if $a = b$ then $b = a$.
$\subseteq$ is not symmetric on $\mathcal{P}(\mathcal{U})$ – $\{1,2,3\} \subseteq \{1,2,3,4\}$ but $\{1,2,3,4\} \nsubseteq \{1,2,3\}$

Transitivity

A binary relation R on a set S is "transitive" iff
for all $a, b, c \in S$, $[(a, b) \in R \wedge (b, c) \in R \rightarrow (\mathbf{a}, \mathbf{c}) \in R]$

$=$ on Σ^* is transitive, for all $a, b, c \in \Sigma^*$ if $a = b$ and $b =$ c then $a = c$.
$\subseteq$ is transitive on $\mathcal{P}(\mathcal{U})$ – for any sets A, B, C if $A \subseteq B$ and $B \subseteq C$ then $A \subseteq C$.
$\in$ is not a transitive relation – $1 \in \{1,2,3\}$, $\{1,2,3\} \in \mathcal{P}(\{1,2,3\})$ but $1 \notin \mathcal{P}(\{1,2,3\})$.

Warm up

Show that $a \equiv b\ (mod\ n)$ if and only if $b \equiv a(mod\ n)$

$a \equiv b(mod\ n) \leftrightarrow n|(b-a) \leftrightarrow nk = b - a(\text{for } k \in \mathbb{Z}) \leftrightarrow$

$n(-k) = a - b(\text{for } -\mathrm{k} \in \mathbb{Z}) \leftrightarrow n|(a-b) \leftrightarrow b \equiv a(mod\ n)$

This was a proof that the relation $\{(a,b) : a \equiv b(mod\ n)\}$ is symmetric!

It was actually overkill to show if and only if. Showing just one direction turns out to be enough!

$a - n = (q-1)n + (a\%n)$. Observe that $q-1$ is an integer, and that this is the form of the division theorem for $(a-n)\%n$. Since the division theorem guarantees a unique integer, $(a-n)\%n = (a\%n)$

What about transitivity?

Some quarters there's a homework problem…we didn't have one this time.

Divides is a transitive relation!

If $p|q$ and $q|r$ then $p|r$.

More Properties of relations

What do we do with relations? Usually we prove properties about them.

Antisymmetry
A binary relation R on a set S is "antisymmetric" iff
for all $a, b \in S$, $[(a, b) \in R \land a \neq b \to (\mathbf{b}, \mathbf{a}) \notin R]$

$\leq$ is antisymmetric on $\mathbb{Z}$

Reflexivity
A binary relation R on a set S is "reflexive" iff
for all $a \in S$, $[(a, a) \in R]$

$\leq$ is reflexive on $\mathbb{Z}$

$\leq$

You've proven antisymmetry too!

(a) Prove that if $a \mid b$ and $b \mid a$, where a and b are integers, then $a = b$ or $a = -b$.

Solution:

Suppose that $a \mid b$ and $b \mid a$, where a, b are integers. By the definition of divides, we have $a \neq 0$, $b \neq 0$ and $b = ka, a = jb$ for some integers k, j. Combining these equations, we see that $a = j(ka)$.

Then, dividing both sides by a, we get $1 = jk$. So, $\frac{1}{j} = k$. Note that j and k are integers, which is only possible if $j, k \in \{1, -1\}$. It follows that $b = -a$ or $b = a$.

Antisymmetry

A binary relation R on a set S is "antisymmetric" iff for all $a, b \in S$, $[(a, b) \in R \wedge a \neq b \rightarrow (\mathrm{b}, \mathrm{a}) \notin R]$

You showed | is antisymmetric on $\mathbb{Z}^+$ in section 5.

for all $a, b \in S$, $[(a, b) \in R \wedge (\mathrm{b}, \mathrm{a}) \in R \rightarrow a = b]$ is equivalent to the definition in the box above

The box version is easier to understand, the other version is usually easier to prove.

Try a few of your own

Pollev.com/robbie

Decide whether each of these relations are

Reflexive, symmetric, antisymmetric, and transitive.

$\subseteq$ on $\mathcal{P}(\mathcal{U})$

$\geq$ on $\mathbb{Z}$

$>$ on $\mathbb{R}$

$|$ on $\mathbb{Z}^+$

$|$ on $\mathbb{Z}$

$\equiv (mod\ 3)$ on $\mathbb{Z}$

Symmetry: for all $a, b \in S$, $[(a, b) \in R \rightarrow (\mathrm{b}, \mathrm{a}) \in R]$

Antisymmetry: for all $a, b \in S$, $[(a, b) \in R \wedge a \neq b \rightarrow (\mathrm{b}, \mathrm{a}) \notin R]$

Transitivity: for all $a, b, c \in S$, $[(a, b) \in R \wedge (b, c) \in R \rightarrow (\mathrm{a}, \mathrm{c}) \in R]$

Reflexivity: for all $a \in S$, $[(a, a) \in R]$

Try a few of your own

Symmetry: for all $a, b \in S$, $[(a, b) \in R \to (\mathrm{b}, \mathrm{a}) \in R]$

Antisymmetry: for all $a, b \in S$, $[(a, b) \in R \land a \neq b \to (\mathrm{b}, \mathrm{a}) \notin R]$

Transitivity: for all $a, b, c \in S$, $[(a, b) \in R \land (b, c) \in R \to (\mathrm{a}, \mathrm{c}) \in R]$

Reflexivity: for all $a \in S$, $[(a, a) \in R]$

Decide whether each of these relations are

Reflexive, symmetric, antisymmetric, and transitive.

$\subseteq$ on $\mathcal{P}(\mathcal{U})$ reflexive, antisymmetric, transitive

$\geq$ on $\mathbb{Z}$ reflexive, antisymmetric, transitive

$>$ on $\mathbb{R}$ antisymmetric, transitive

$|$ on $\mathbb{Z}^+$ reflexive, antisymmetric, transitive

$|$ on $\mathbb{Z}$ reflexive, transitive

$\equiv (mod\ 3)$ on $\mathbb{Z}$ reflexive, symmetric, transitive