you would expect that if team a beats team b and team b beats team c then team a should also beat team c.

This is not the case for Pac 12 football (in 2019)

2019 results by /u/jensyao on reddit

Warm-up

Design a context-free grammar whose language is "all binary strings with exactly two zeros".

$S \to A0A0A$

$A \to \varepsilon \mid 1A$

~~$S \to S0S0S$~~

Relations And Graphs

CSE 311 Autumn 20

Lecture 22

Announcements

Midterm grades are out

- Median was 91% -- you did very well! Even if it did take longer than I intended it to.

Also “grade projections” to interpret your work so far (on Ed).

If you want to talk to me one-on-one about grades, I added a few more slots tomorrow.

Updated HW6 P6. There was a typo in the “find the bug” problem

- If you found the typo, that’s a bug. You can explain (just) that bug to get full credit.
- If you haven’t started yet (or you didn’t see the typo), there’s the bug I intended to put in there still. You can also find that one and explain (just) that bug to get full credit.

Announcements

Thanksgiving is on Thursday!

So there's no class Thursday or Friday.

Wednesday's lecture is wrapping up this slide deck. We'll use the remaining time to talk about common misconceptions from the midterm.

Wednesday's polleverywhere will open **tonight** it's "what question do you most want to talk about?"

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.

Arithmetic

$\Sigma = \{+, *, (,), x, y, z, 0, 1, \ldots, 9\}$

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

Generate $(2 * x) + y$

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

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

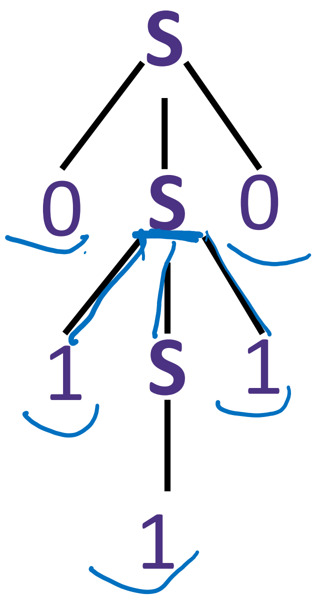
Suppose a context free grammar G generates a string x

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.

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

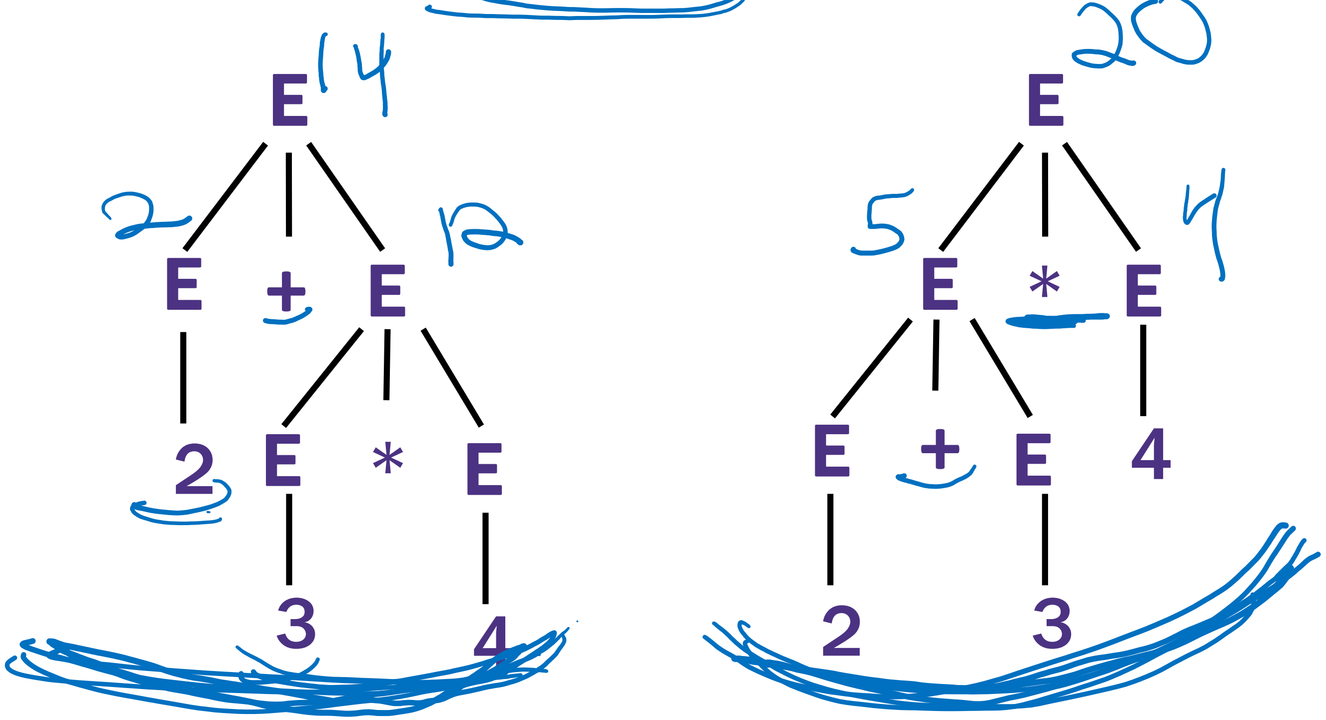
0110



Back to the arithmetic

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

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"

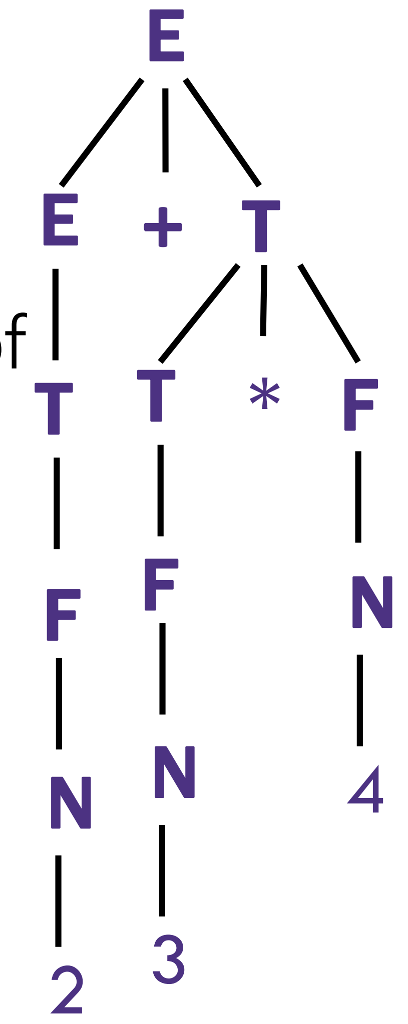
Only introduce a * sign after you've eliminated the possibility of introducing another + sign in that area.

$E \rightarrow T | E + T$

$T \rightarrow F | T * F$

$F \rightarrow (E) | N$

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



CNFs in practice

CFG

Used to define programming languages.

Often written in Backus-Naur Form – just different notation

Variables are <names-in-brackets>

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

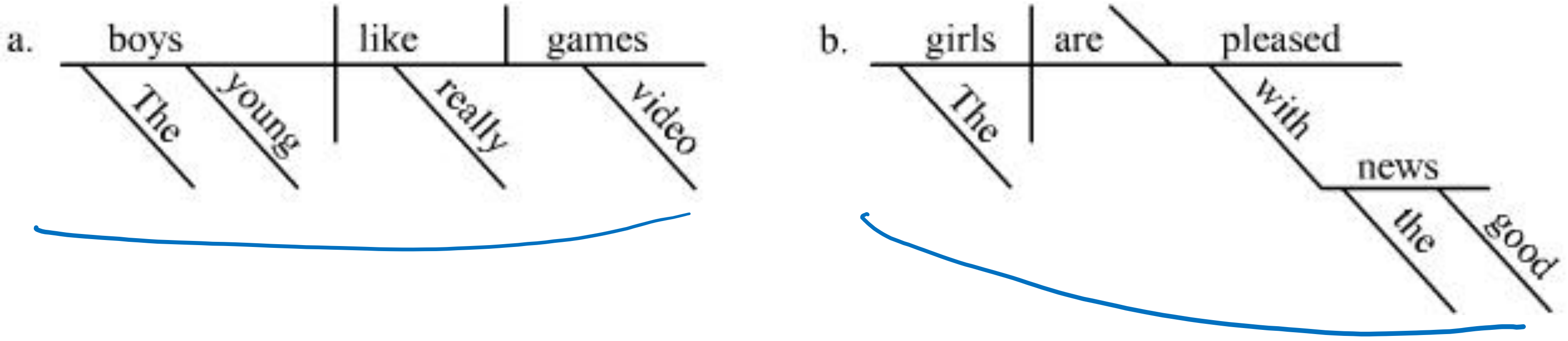
→ is replaced with ::= or :

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>

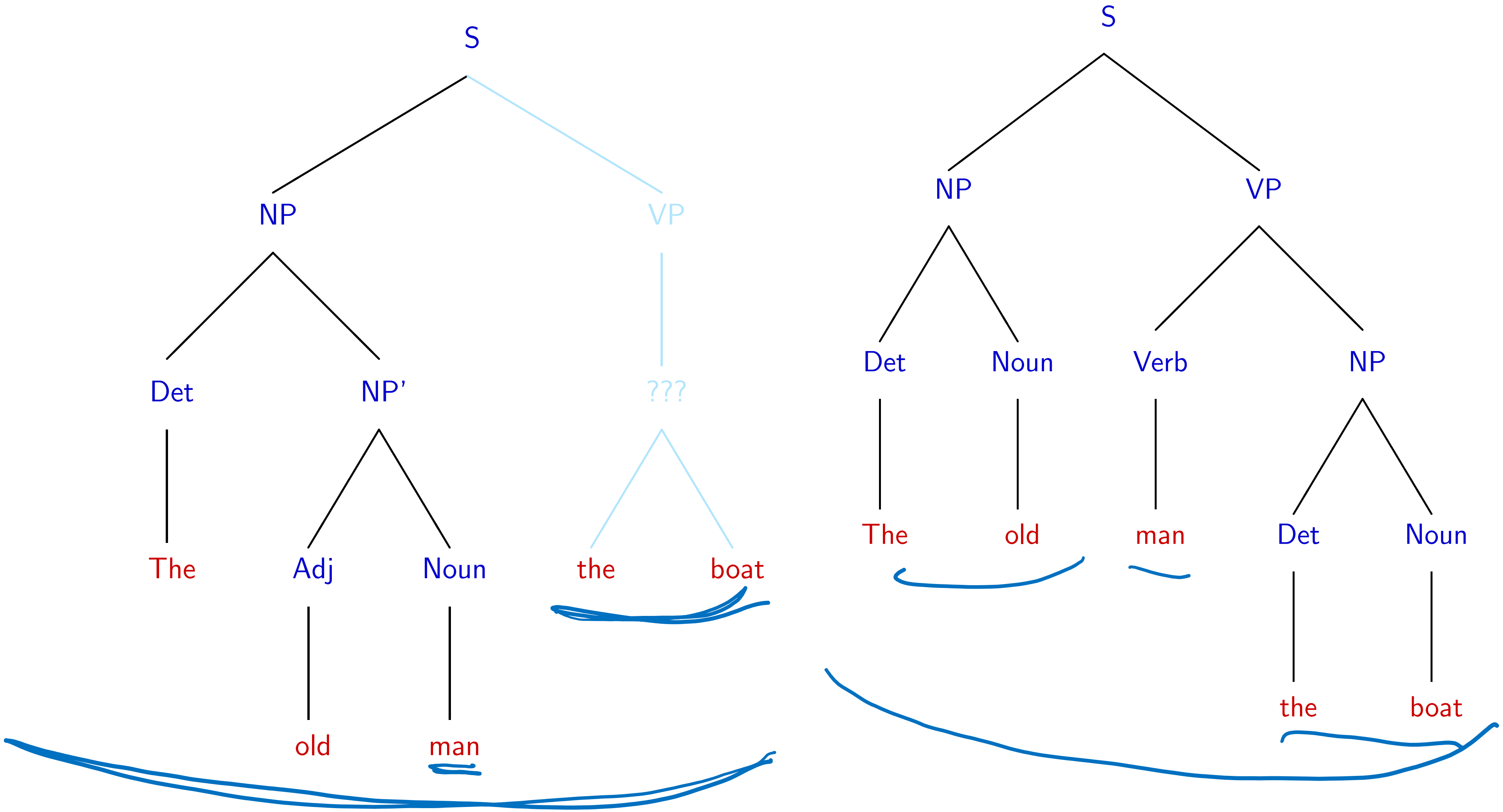
<object>::=<noun phrase>

"The boy hit the girl with the bat."

The old man the boat.

verb

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!

$S \to SS \mid \epsilon$

Are there languages even CFGs cannot represent?

Yes!

$\{0^k 1^j 2^k 3^j \mid j, k \geq 0\}$ 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”

After Thanksgiving, we’ll talk about how each of these are “equivalent to weaker computers.”

Next time: 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 $A \times A$

$(a,b) \in R$

$a \leq b$

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

divides on $\{1,2,3,4\}$

$\{(1,1),(1,2),(1,3),(1,4),(2,4)\}$

"y when squared gives x" is a relation on $\mathbb{R}$

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

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

$p \mid q$ $a = b$

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

$(a,b) \in R$ aRb

Symmetry

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

$=$ 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 $a, b, c \in S$, $[(a,b) \in R \wedge (b,c) \in R \rightarrow (a,c) \in R]$

$=$ 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!

this is the form of the division theorem for $(a-n)\%n$. Since the division theorem guarantees a unique integer, $(a-n)\%n = (a\%n)$

You've also done a proof of transitivity!

5. Divide[s] we fall [14 points]

(a) Write an English proof showing that for any **positive** integers p, q, r if $p \mid q$ and $q \mid r$ then $p \mid r$. [8 points]

You did this proof on HW4. You were showing:
$\mid$ is a transitive relation on $\mathbb{Z}^+$.

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 $a, b \in S$, $[(a, b) \in R \wedge a \neq b \rightarrow (\mathbf{b}, \mathbf{a}) \notin R]$

$\leq$ 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!

(b) Write an English proof showing that for any **positive** integers p, q if $p \mid q$ and $q \mid p$, then $p = q$. For this problem, you may not use the result of Section 4's problem 5a as a fact, but you may find that proof useful to model yours after. [6 points]

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

You showed | is antisymmetric on $\mathbb{Z}^+$

for all $a, b \in S$, $[(a, b) \in R \wedge (b, a) \in R \rightarrow a = b]$ 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

Fill out the poll everywhere for Activity Credit!

Go to pollev.com/cse311 and login with your UW identity
Or text cse311 to 22333

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 $a, b \in S$, $[(a, b) \in R \wedge a \neq b \to (\mathrm{b}, \mathrm{a}) \notin R]$

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

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

Two Prototype Relations

A lot of fundamental relations follow one of two prototypes:

Equivalence Relation

A relation that is reflexive, symmetric, and transitive is called an "equivalence relation"

Partial Order Relation

A relation that is reflexive, antisymmetric, and transitive is called a "partial order"

Equivalence Relations

Equivalence relations "act kinda like equals"

$\equiv \pmod{n}$ is an equivalence relation.

$\equiv$ on compound propositions is an equivalence relation.

Fun fact: Equivalence relations "partition" their elements.

An equivalence relation R on S divides S into sets $S_1, \ldots S_k$ such that.

$\forall s\ (s \in S_i \text{ for some } i)$

$\forall s, s'$ $(s, s' \in S_i$ for some i if and only if $(s, s') \in R)$

$S_i \cap S_j = \emptyset$ for all $i \neq j$

Partial Orders

Partial Orders "behave kinda like less than or equal to"

In the sense that they put things in order

But it's only kinda like less than – it's possible that some elements can't be compared.

$|$ on $\mathbb{Z}^+$ is a partial order

$\subseteq$ on $\mathcal{P}(\mathcal{U})$ is a partial order

x is a prerequisite of (or-equal-to) y is a partial order on CSE courses

Why Bother?

If you prove facts about all equivalence relations or all partial orders, you instantly get facts in lots of different contexts.

If you learn to recognize partial orders or equivalence relations, you can get **a lot** of intuition for new concepts in a short amount of time.

Combining Relations

Given a relation R from A to B

And a relation S from B to C,

The relation $S \circ R$ from A to C is

$\{(a, c) : \exists b[(a, b) \in R \wedge (b, c) \in S]\}$

Yes, I promise it's $S \circ R$ not $R \circ S$ – it makes more sense if you think about relations $(x, f(x))$ and $(x, g(x))$

But also don't spend a ton of energy worrying about the order, we almost always care about $R \circ R$, where order doesn't matter.

Combining Relations

To combine relations, it's a lot easier if we can see what's happening.

We'll use a representation of a directed graph

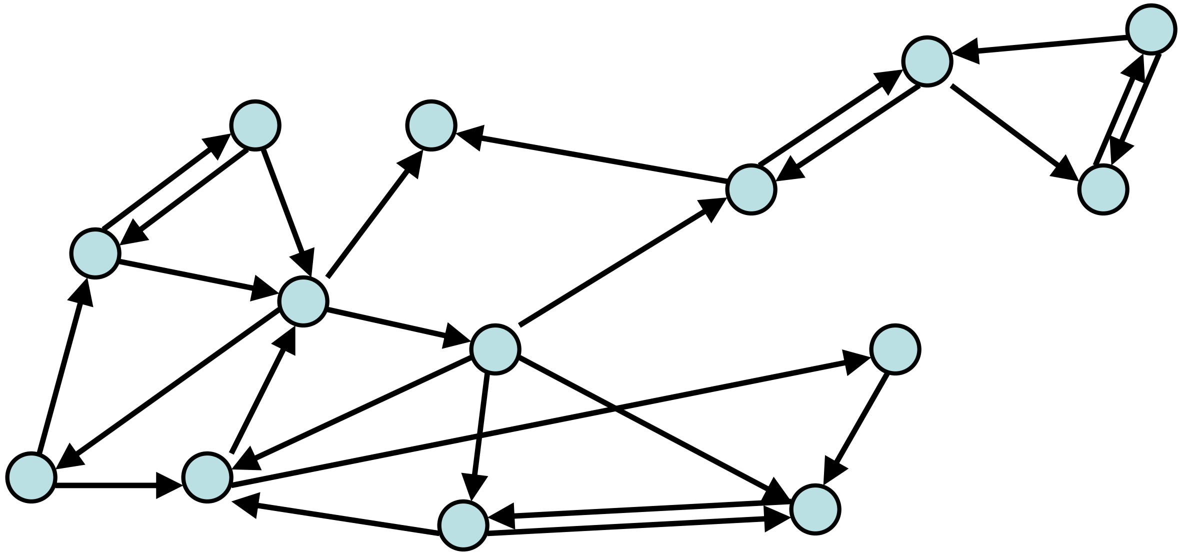
Directed Graphs

$G = (V, E)$

V is a set of vertices (an underlying set of elements)

E is a set of edges (ordered pairs of vertices; i.e. connections from one to the next).

Path $v_0, v_1, \ldots, v_k$ such that $(v_i, v_{i+1}) \in E$
Simple Path: path with all v_i distinct
Cycle: path with $v_0 = v_k$ (and $k > 0$)
Simple Cycle: simple path plus edge (v_k, v_0) with $k > 0$



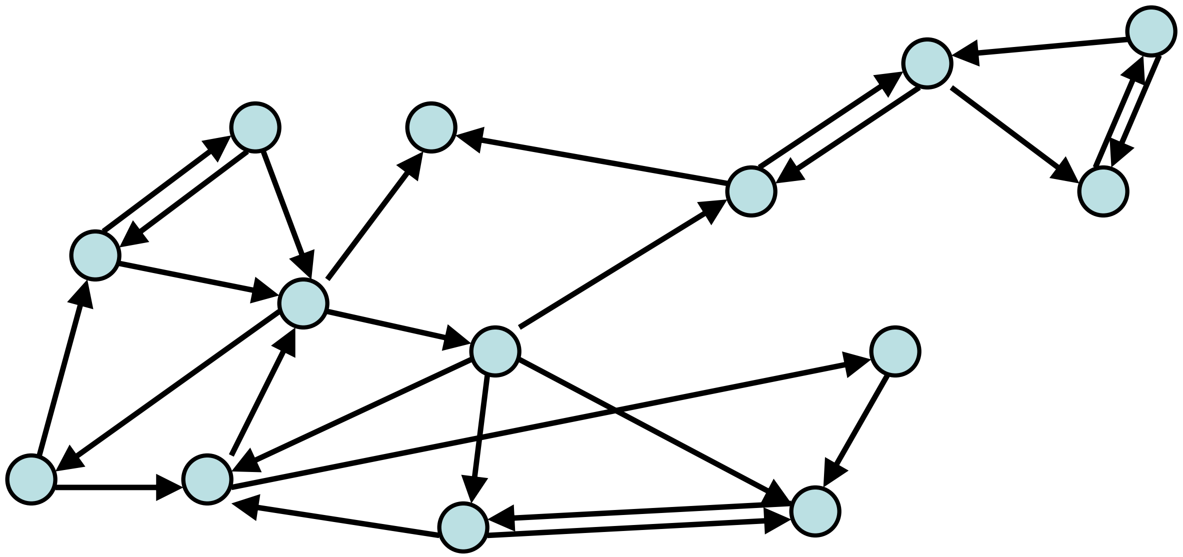
Directed Graphs

$G = (V, E)$

V is a set of vertices (an underlying set of elements)

E is a set of edges (ordered pairs of vertices; i.e. connections from one to the next).

Path $v_0, v_1, \dots, v_k$ such that $(v_i, v_{i+1}) \in E$
Simple Path: path with all v_i distinct
Cycle: path with $v_0 = v_k$ (and $k > 0$)
Simple Cycle: simple path plus edge (v_k, v_0) with $k > 0$



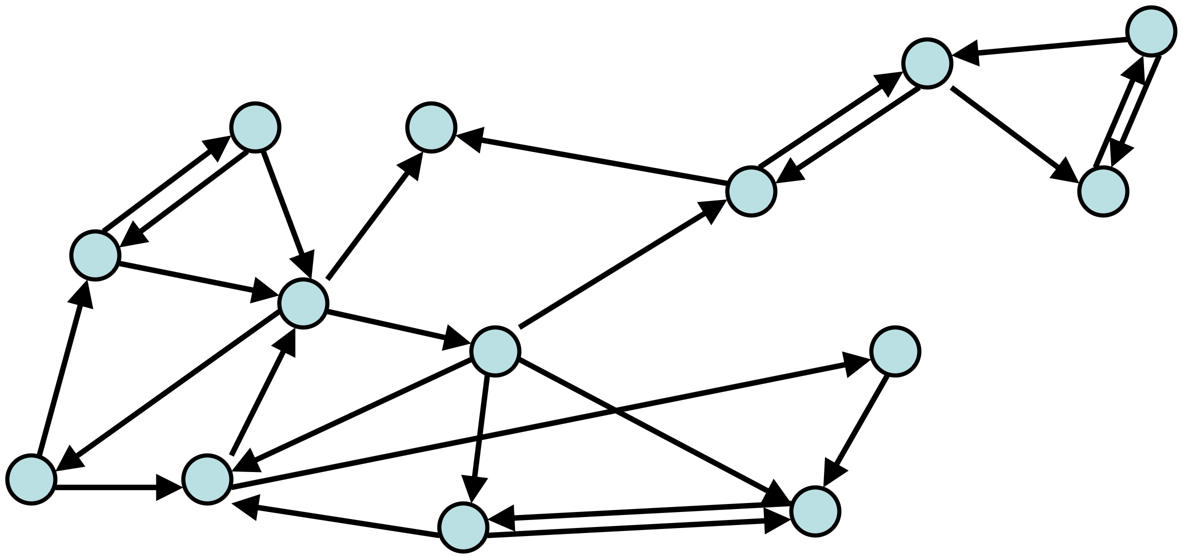
Directed Graphs

$G = (V, E)$

V is a set of vertices (an underlying set of elements)

E is a set of edges (ordered pairs of vertices; i.e. connections from one to the next).

Path $v_0, v_1, \ldots, v_k$ such that $(v_i, v_{i+1}) \in E$
Simple Path: path with all v_i distinct
Cycle: path with $v_0 = v_k$ (and $k > 0$)
Simple Cycle: simple path plus edge (v_k, v_0) with $k > 0$



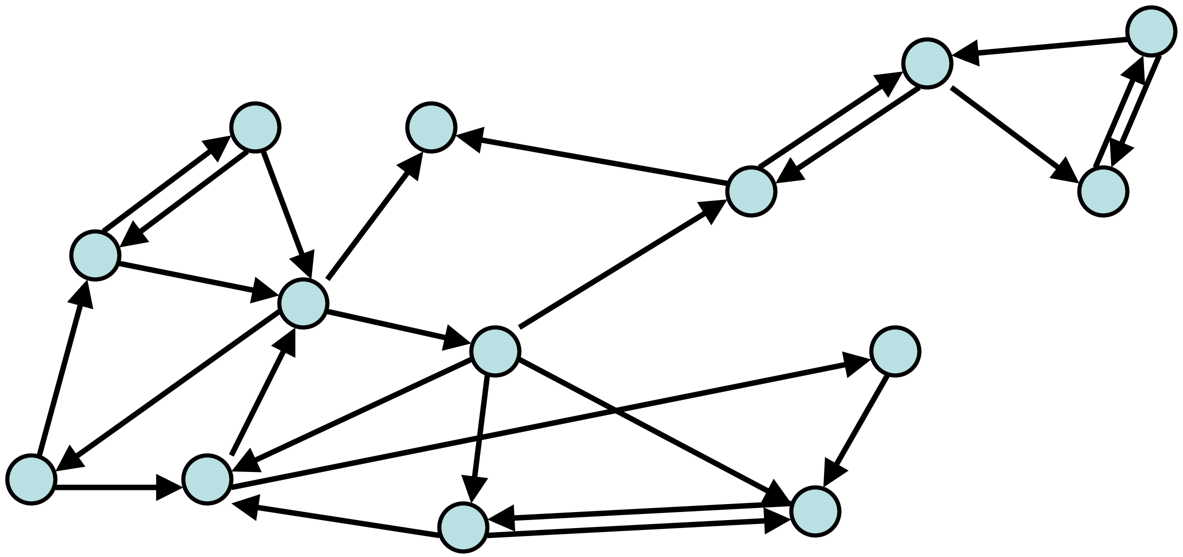
Directed Graphs

$G = (V, E)$

V is a set of vertices (an underlying set of elements)

E is a set of edges (ordered pairs of vertices; i.e. connections from one to the next).

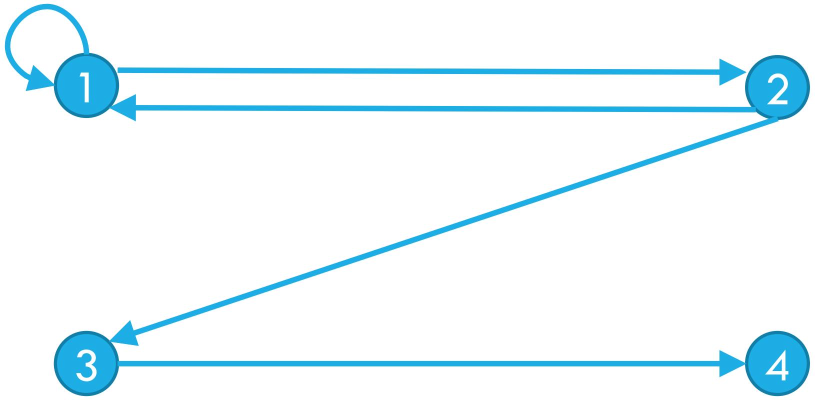
Path $v_0, v_1, \ldots, v_k$ such that $(v_i, v_{i+1}) \in E$
Simple Path: path with all v_i distinct
Cycle: path with $v_0 = v_k$ (and $k > 0$)
Simple Cycle: simple path plus edge (v_k, v_0) with $k > 0$



Representing Relations

To represent a relation R on a set A, have a vertex for each element of A and have an edge (a, b) for every pair in R.

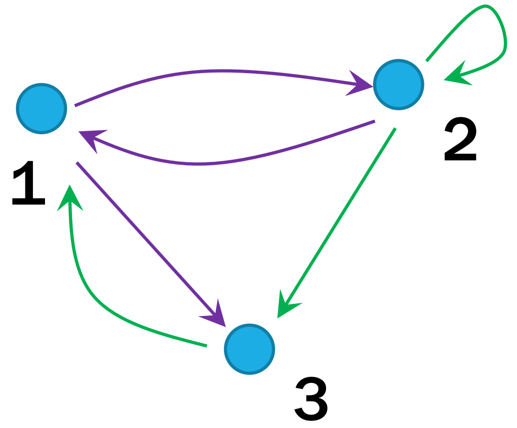
Let A be $\{1,2,3,4\}$ and R be $\{(1,1), (1,2), (2,1), (2,3), (3,4)\}$

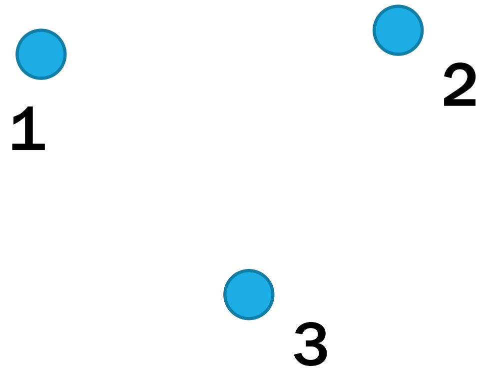


Combining Relations

If $S = \{(2,2),(2,3),(3,1)\}$ and $R = \{(1,2),(2,1),(1,3)\}$

Compute $S \circ R$ i.e. every pair (a, c) with a b with $(a, b) \in R$ and $(b, c) \in S$

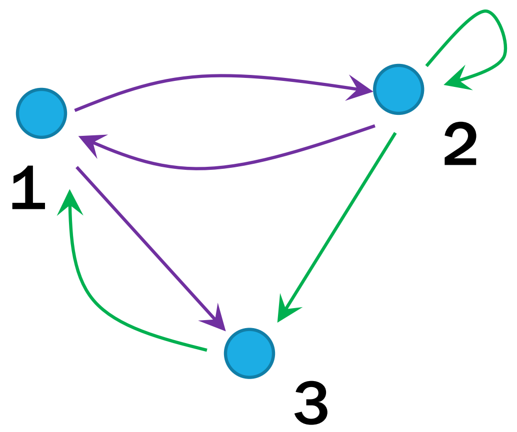


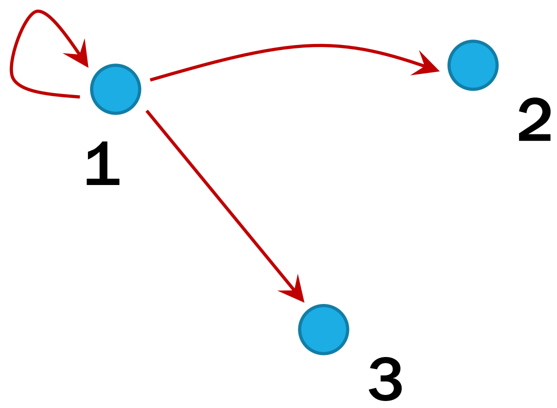


Combining Relations

If $\boldsymbol{S} = \{(\mathbf{2},\mathbf{2}),(\mathbf{2},\mathbf{3}),(\mathbf{3},\mathbf{1})\}$ and $\boldsymbol{R} = \{(\mathbf{1},\mathbf{2}),(\mathbf{2},\mathbf{1}),(\mathbf{1},\mathbf{3})\}$

Compute $\boldsymbol{S} \circ \boldsymbol{R}$ i.e. every pair (a, c) with a b with $(a, b) \in R$ and $(b, c) \in S$





Combining Relations

Let R be a relation on A.

Define R^0 as $\{(a, a) : a \in A\}$

$R^k = R^{k-1} \circ R$

$(a, b) \in R^k$ if and only if there is a path of length k from a to b in R.

We can find that on the graph!

More Powers of R.

For two vertices in a graph, a can reach b if there is a path from a to b.

Let R be a relation on the set A. The connectivity relation R^* consists of all pairs (a, b) such that a can reach b (i.e. there is a path from a to b in R)

$R^* = \bigcup_{k=0}^{\infty} R^k$

Note we're starting from 0 (the textbook makes the unusual choice of starting from $k = 1$).

What's the point of R^*

R^* is also the "reflexive-transitive closure of R.

It answers the question "what's the minimum amount of edges I would need to add to R to make it reflexive and transitive.

Why care about that? The transitive-reflexive closure can be a summary of data – you might want to precompute it so you can easily check if a can reach b instead of recomputing it every time.

Relations and Graphs

Describe how each property will show up in the graph of a relation.

Reflexive

Symmetric

Antisymmetric

Transitive

Relations and Graphs

Describe how each property will show up in the graph of a relation.

Reflexive

Every vertex has a "self-loop" (an edge from the vertex to itself)

Symmetric

Every edge has its "reverse edge" (going the other way) also in the graph.

Antisymmetric

No edge has its "reverse edge" (going the other way) also in the graph.

Transitive

If there's a length-2 path from a to b then there's a direct edge from a to b