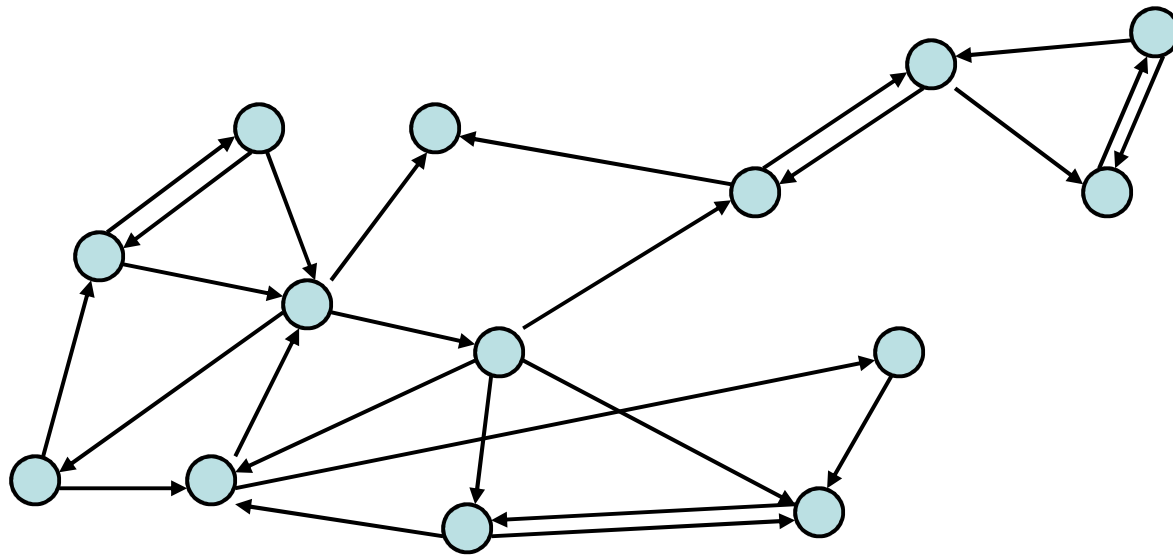


CSE 311: Foundations of Computing

Lecture 19: Context-Free Grammars, Relations and Directed Graphs



Last Class: Context-Free Grammars

- A Context-Free Grammar (CFG) is given by a finite set of substitution rules involving
 - A finite set \mathbf{V} of *variables* that can be replaced
 - Alphabet Σ of *terminal symbols* that can't be replaced
 - One variable, usually \mathbf{S} , is called the *start symbol*
- The rules involving a variable \mathbf{A} are written as

$$\mathbf{A} \rightarrow w_1 \mid w_2 \mid \cdots \mid w_k$$

where each w_i is a string of variables and terminals –
that is $w_i \in (\mathbf{V} \cup \Sigma)^*$

Last Class: How CFGs generate strings

- Begin with start symbol **S**
- If there is some variable **A** in the current string you can replace it by one of the w 's in the rules for **A**
 - $\mathbf{A} \rightarrow w_1 \mid w_2 \mid \cdots \mid w_k$
 - Write this as $\mathbf{xAy} \Rightarrow \mathbf{xwy}$
 - Repeat until no variables left
- The set of strings the CFG generates are all strings produced in this way that have no variables

Last Class: Context-Free Grammars

Example: $S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid \varepsilon$

The set of all binary palindromes

Example: $S \rightarrow 0S \mid S1 \mid \varepsilon$

0^*1^*

Last Class: Context-Free Grammars

Grammar for $\{0^n 1^n : n \geq 0\}$

(all strings with same # of 0's and 1's with all 0's before 1's)

$$S \rightarrow 0S1 \mid \varepsilon$$

Example: $S \rightarrow (S) \mid SS \mid \varepsilon$

The set of all strings of matched parentheses

CFGs and recursively-defined sets of strings

- A CFG with the start symbol **S** as its only variable recursively defines the set of strings of terminals that **S** can generate
- A CFG with more than one variable is a simultaneous recursive definition of the sets of strings generated by *each* of its variables
 - Sometimes necessary to use more than one

Simple Arithmetic Expressions

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

Generate $(2 * x) + y$

Generate $x + y * z$ in two fundamentally different ways

Simple Arithmetic Expressions

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

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 $x + y * z$ in two fundamentally different ways

$E \Rightarrow E + E \Rightarrow x + E \Rightarrow x + E * E \Rightarrow x + y * E \Rightarrow x + y * z$

$E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow x + E * E \Rightarrow x + y * E \Rightarrow x + y * z$

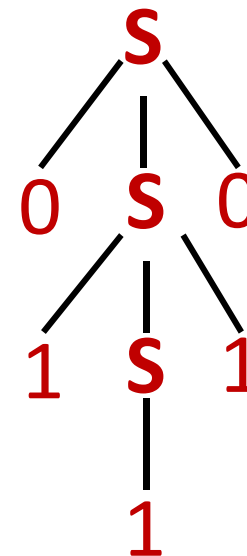
Parse Trees

Suppose that grammar G generates a string x

- A *parse tree* of x for G has
 - Root labeled S (start symbol of G)
 - The children of any node labeled A are labeled by symbols of w left-to-right for some rule $A \rightarrow w$
 - The symbols of x label the leaves ordered left-to-right

$$S \rightarrow OSO \mid 1S1 \mid 0 \mid 1 \mid \varepsilon$$

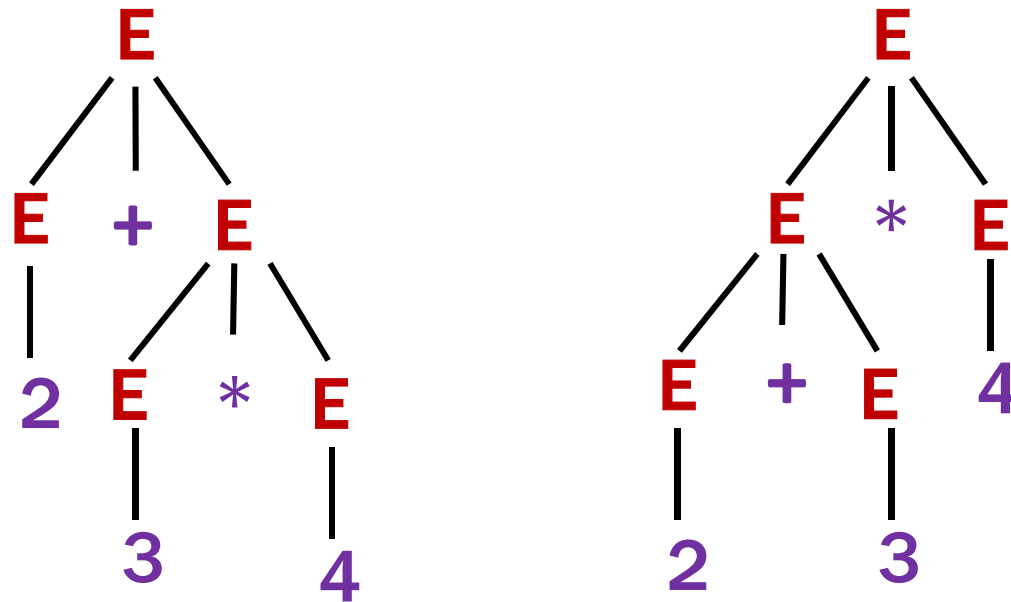
Parse tree of 01110



Simple Arithmetic Expressions

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

Two parse trees for $2+3*4$



Building precedence in simple arithmetic expressions

- **E** – expression (start symbol)
- **T** – term **F** – factor **I** – identifier **N** - number

E → **T** | **E+T**

T → **F** | **T*F**

F → (**E**) | **I** | **N**

I → **x** | **y** | **z**

N → **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

Building precedence in simple arithmetic expressions

- **E** – expression (start symbol)
- **T** – term **F** – factor **I** – identifier **N** - number

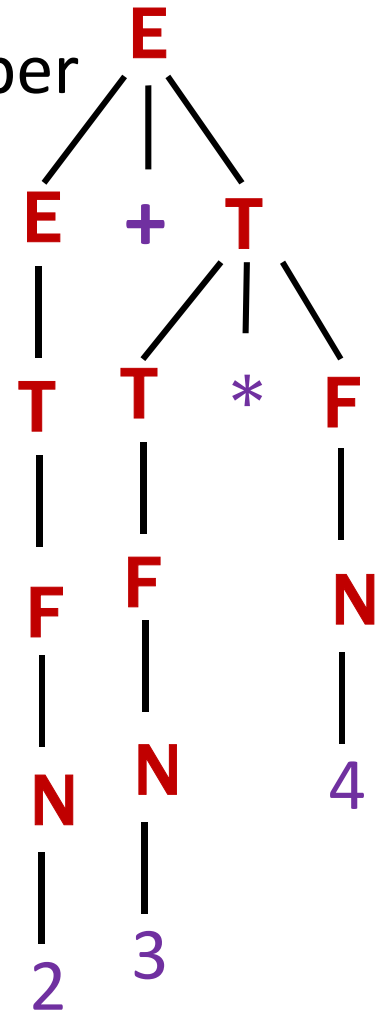
E → **T** | **E+T**

T → **F** | **T*F**

F → (**E**) | **I** | **N**

I → **x** | **y** | **z**

N → 0 | 1 | 2 | 3 | 4 |
5 | 6 | 7 | 8 | 9



Backus-Naur Form (The same thing...)

BNF (Backus-Naur Form) grammars

- Originally used to define programming languages
- Variables denoted by long names in angle brackets, e.g.
 - <identifier>, <if-then-else-statement>,
<assignment-statement>, <condition>
 - ::= used instead of \rightarrow

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

Back to middle school:

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

<noun phrase> ::= <article> <adjective> <noun>

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

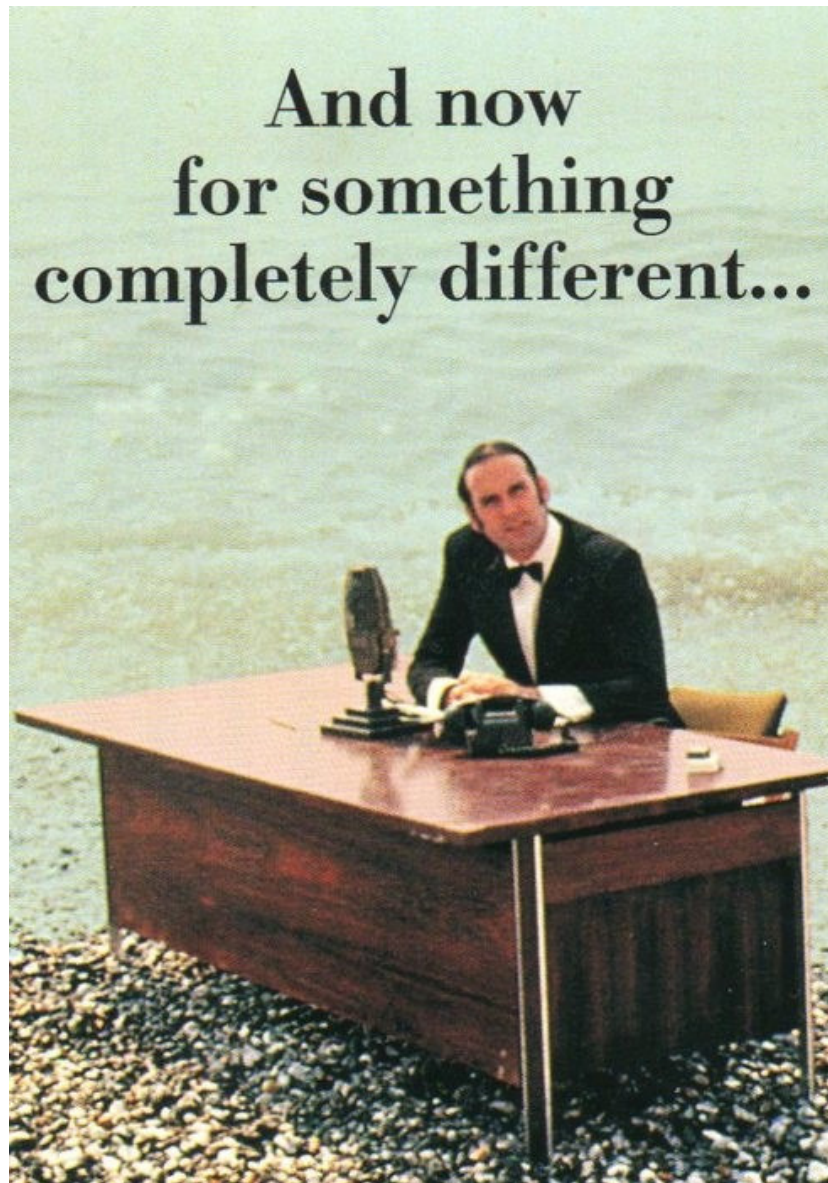
<object> ::= <noun phrase>

Parse:

The yellow duck squeaked loudly

The red truck hit a parked car

Relations and Directed Graphs



Relations

Let A and B be sets,

A **binary relation from A to B** is a subset of $A \times B$

Let A be a set,

A **binary relation on A** is a subset of $A \times A$

Relations You Already Know!

\geq on \mathbb{N}

That is: $\{(x,y) : x \geq y \text{ and } x, y \in \mathbb{N}\}$

$<$ on \mathbb{R}

That is: $\{(x,y) : x < y \text{ and } x, y \in \mathbb{R}\}$

$=$ on Σ^*

That is: $\{(x,y) : x = y \text{ and } x, y \in \Sigma^*\}$

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

That is: $\{(A,B) : A \subseteq B \text{ and } A, B \in \mathcal{P}(U)\}$

More Relation Examples

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

$$R_2 = \{(x, y) \mid x \equiv y \pmod{5}\}$$

$$R_3 = \{(c_1, c_2) \mid c_1 \text{ is a prerequisite of } c_2\}$$

$$R_4 = \{(s, c) \mid \text{student } s \text{ has taken course } c\}$$

Properties of Relations

Let R be a relation on A .

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 $(a,c) \in R$

Which relations have which properties?

\geq on \mathbb{N} :

$<$ on \mathbb{R} :

$=$ on Σ^* :

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

$R_2 = \{(x, y) \mid x \equiv y \pmod{5}\}$:

$R_3 = \{(c_1, c_2) \mid c_1 \text{ is a prerequisite of } c_2\}$:

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 $(a, c) \in R$

Which relations have which properties?

\geq on \mathbb{N} : Reflexive, Antisymmetric, Transitive

$<$ on \mathbb{R} : Antisymmetric, Transitive

$=$ on Σ^* : Reflexive, Symmetric, Antisymmetric, Transitive

\subseteq on $\mathcal{P}(U)$: Reflexive, Antisymmetric, Transitive

$R_2 = \{(x, y) \mid x \equiv y \pmod{5}\}$: Reflexive, Symmetric, Transitive

$R_3 = \{(c_1, c_2) \mid c_1 \text{ is a prerequisite of } c_2\}$: Antisymmetric

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 $(a, c) \in R$

Combining Relations

Let R be a relation from A to B .

Let S be a relation from B to C .

The **composition** of R and S , $S \circ R$ is the relation from A to C defined by:

$$S \circ R = \{ (a, c) \mid \exists b \text{ such that } (a,b) \in R \text{ and } (b,c) \in S \}$$

Intuitively, a pair is in the composition if there is a “connection” from the first to the second.

Examples

$(a,b) \in \text{Parent}$ iff b is a parent of a

$(a,b) \in \text{Sister}$ iff b is a sister of a

When is $(x,y) \in \text{Sister} \circ \text{Parent}$?

When is $(x,y) \in \text{Parent} \circ \text{Sister}$?

$$S \circ R = \{(a, c) \mid \exists b \text{ such that } (a,b) \in R \text{ and } (b,c) \in S\}$$

Examples

Using the relations: Parent, Child, Brother, Sister, Sibling, Father, Mother, Husband, Wife express:

Uncle: b is an uncle of a

Cousin: b is a cousin of a

Powers of a Relation

$$\begin{aligned} R^2 &= R \circ R \\ &= \{(a, c) \mid \exists b \text{ such that } (a, b) \in R \text{ and } (b, c) \in R\} \end{aligned}$$

$$R^0 = \{(a, a) \mid a \in A\} \quad \text{“the equality relation on } A\text{”}$$

$$R^1 = R = R^0 \circ R$$

$$R^{n+1} = R^n \circ R \quad \text{for } n \geq 0$$

Matrix Representation

Relation R on $A = \{a_1, \dots, a_p\}$

$$m_{ij} = \begin{cases} 1 & \text{if } (a_i, a_j) \in R \\ 0 & \text{if } (a_i, a_j) \notin R \end{cases}$$

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

	1	2	3	4
1	1	1	0	1
2	1	0	1	0
3	0	1	1	0
4	0	1	1	0

Directed Graphs

$G = (V, E)$

V – vertices

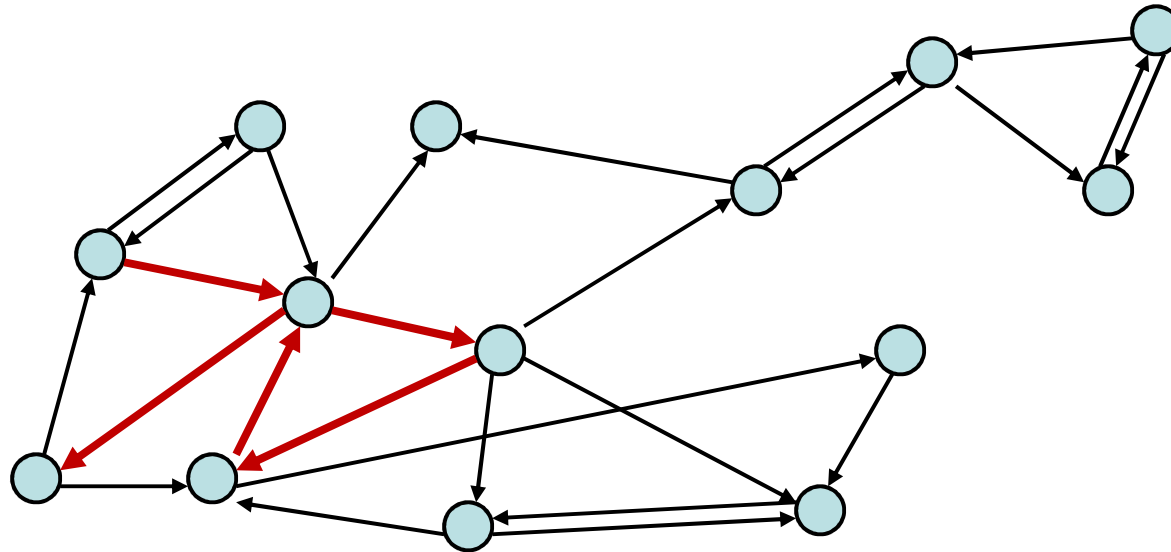
E – edges, ordered pairs of vertices

Path: v_0, v_1, \dots, v_k with each (v_i, v_{i+1}) in E

Simple Path: none of v_0, \dots, v_k repeated

Cycle: $v_0 = v_k$

Simple Cycle: $v_0 = v_k$, none of v_1, \dots, v_k repeated



Directed Graphs

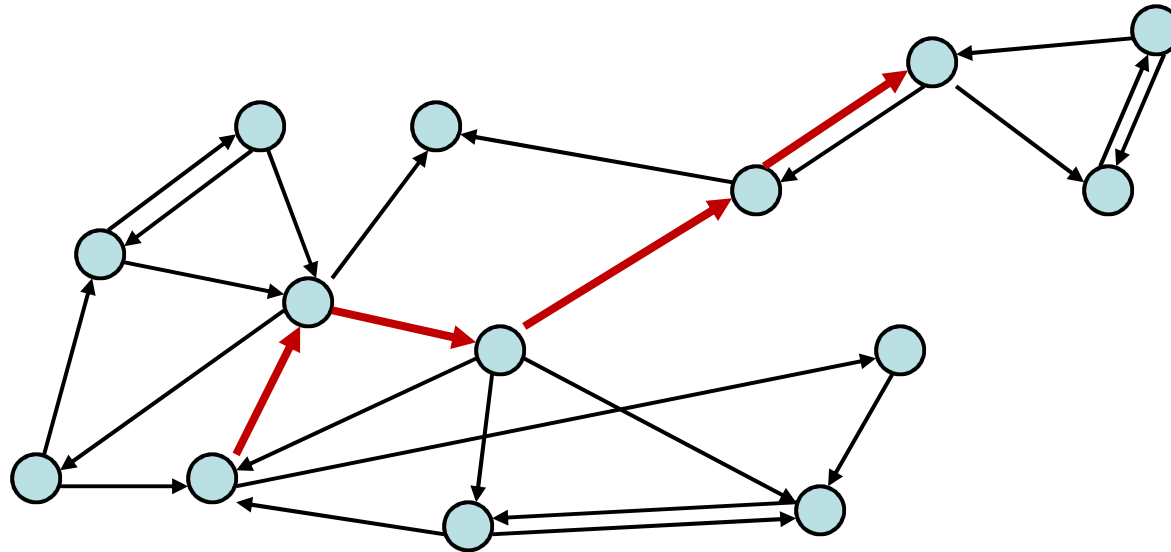
$G = (V, E)$ V – vertices
 E – edges, ordered pairs of vertices

Path: v_0, v_1, \dots, v_k with each (v_i, v_{i+1}) in E

Simple Path: none of v_0, \dots, v_k repeated

Cycle: $v_0 = v_k$

Simple Cycle: $v_0 = v_k$, none of v_1, \dots, v_k repeated



Directed Graphs

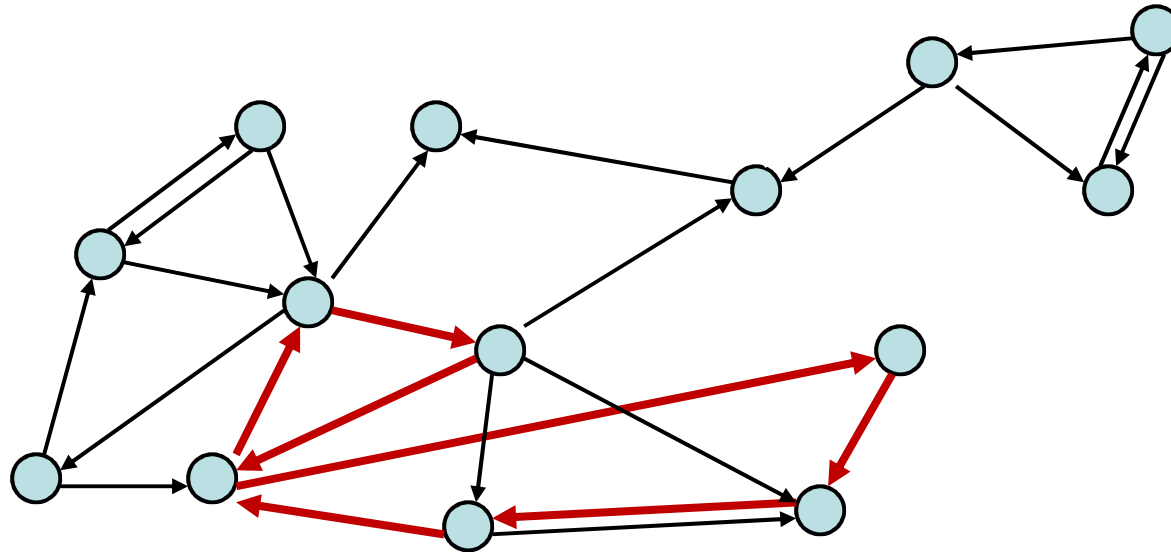
$G = (V, E)$ V – vertices
 E – edges, ordered pairs of vertices

Path: v_0, v_1, \dots, v_k with each (v_i, v_{i+1}) in E

Simple Path: none of v_0, \dots, v_k repeated

Cycle: $v_0 = v_k$

Simple Cycle: $v_0 = v_k$, none of v_1, \dots, v_k repeated



Directed Graphs

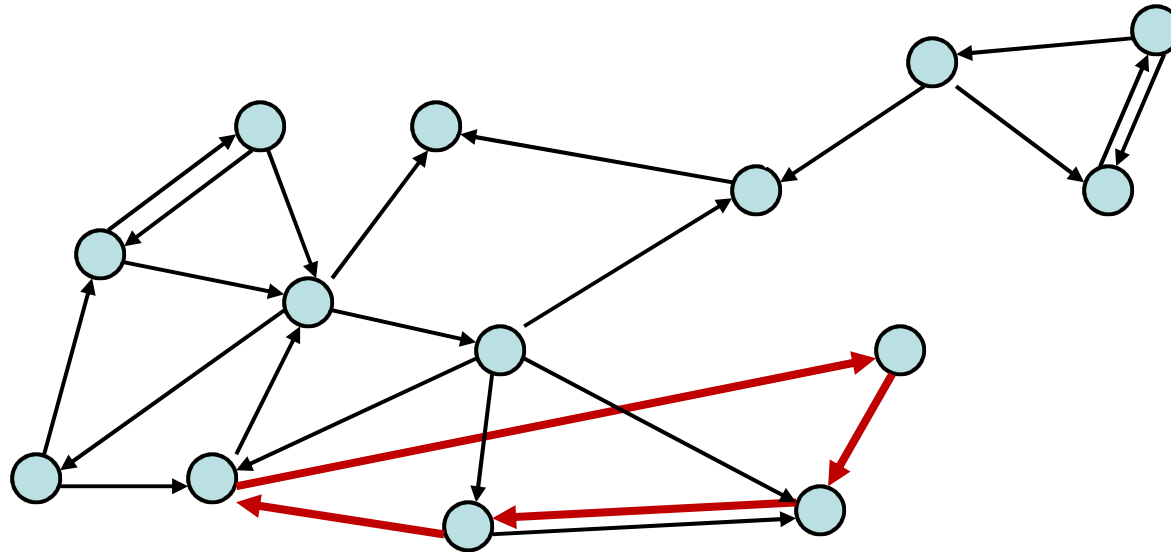
$G = (V, E)$ V – vertices
 E – edges, ordered pairs of vertices

Path: v_0, v_1, \dots, v_k with each (v_i, v_{i+1}) in E

Simple Path: none of v_0, \dots, v_k repeated

Cycle: $v_0 = v_k$

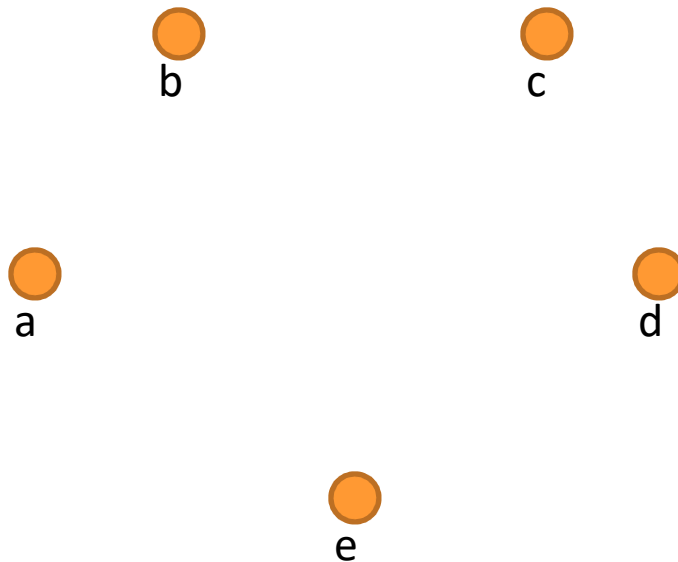
Simple Cycle: $v_0 = v_k$, none of v_1, \dots, v_k repeated



Representation of Relations

Directed Graph Representation (Digraph)

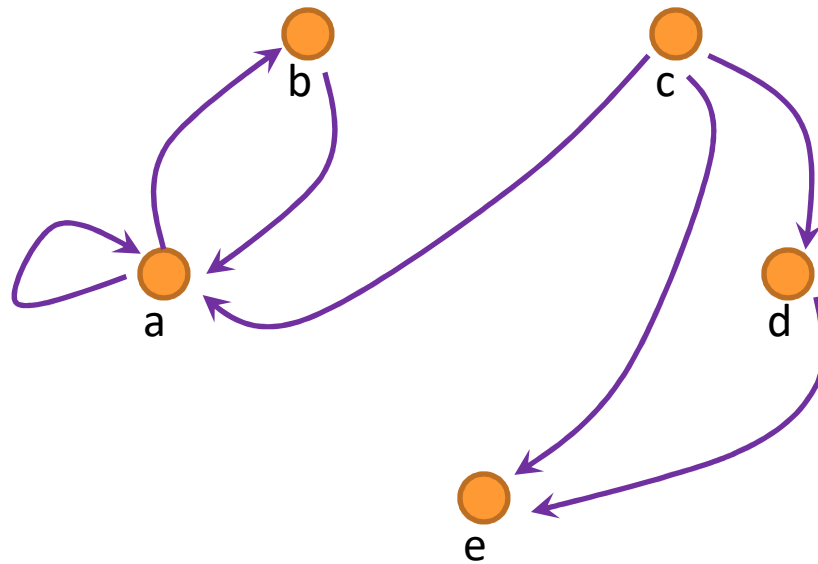
$\{(a, b), (a, a), (b, a), (c, a), (c, d), (c, e), (d, e)\}$



Representation of Relations

Directed Graph Representation (Digraph)

$\{(a, b), (a, a), (b, a), (c, a), (c, d), (c, e), (d, e)\}$



Relational Composition using Digraphs

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

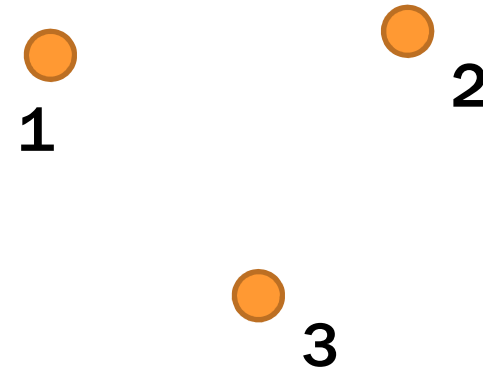
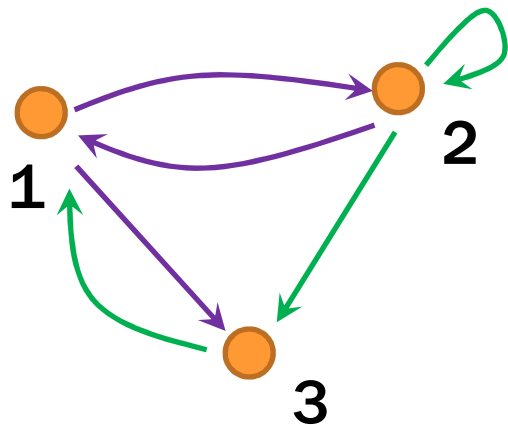
Compute $S \circ R$



Relational Composition using Digraphs

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

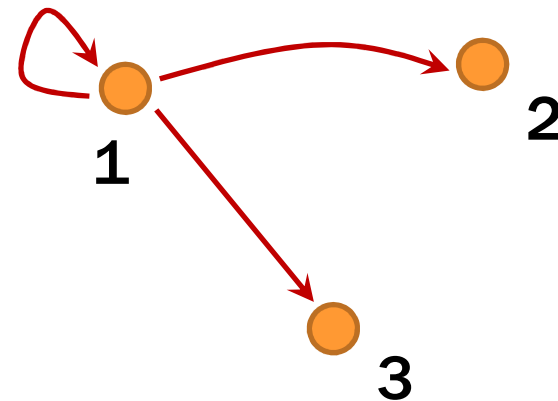
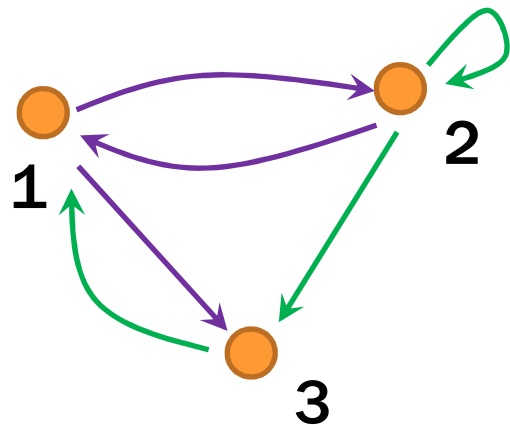
Compute $S \circ R$



Relational Composition using Digraphs

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

Compute $S \circ R$



Paths in Relations and Graphs

Defn: The **length** of a path in a graph is the number of edges in it (counting repetitions if edge used $>$ once).

Let R be a relation on a set A . There is a path of length n from a to b if and only if $(a,b) \in R^n$

Connectivity In Graphs

Defn: Two vertices in a graph are **connected** iff there is a path between them.

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

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

Note: The text uses the wrong definition of this quantity. What the text defines (ignoring $k=0$) is usually called R^+

How Properties of Relations show up in Graphs

Let R be a relation on A .

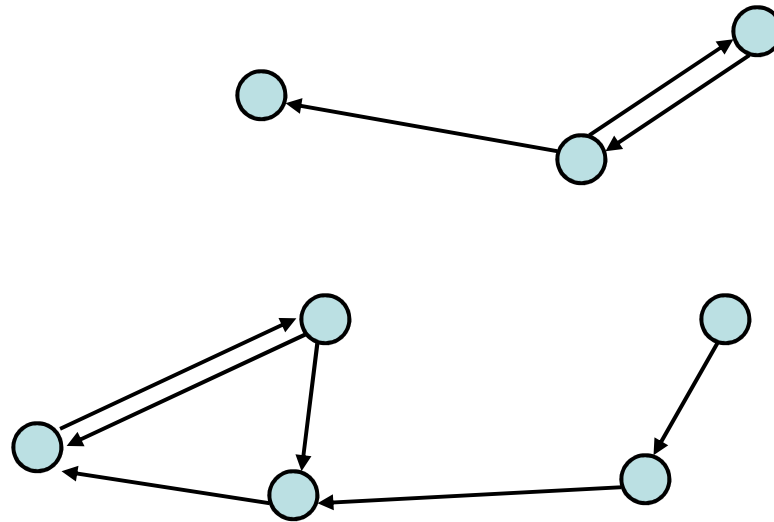
R is **reflexive** iff $(a,a) \in R$ for every $a \in A$

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 $(a,c) \in R$

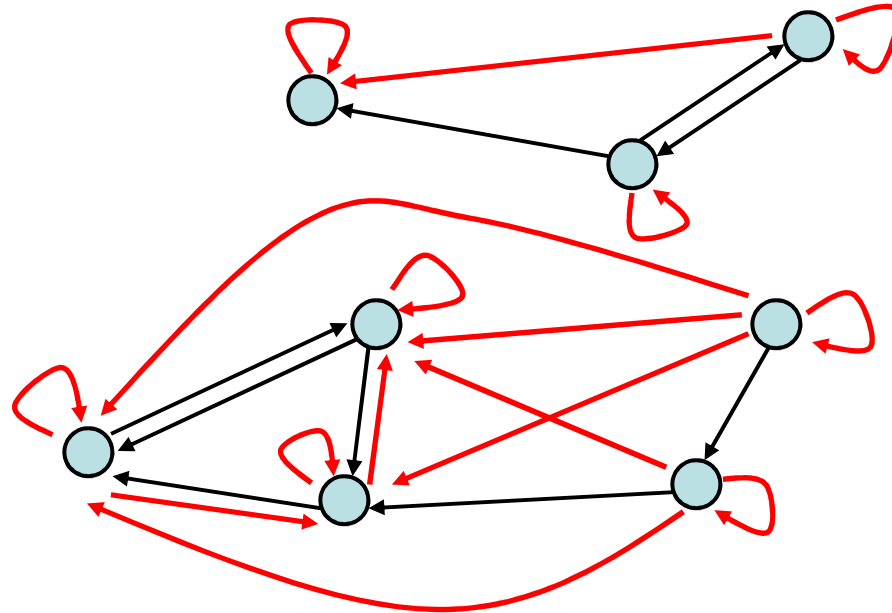
Transitive-Reflexive Closure



Add the **minimum possible** number of edges to make the relation transitive and reflexive.

The **transitive-reflexive closure** of a relation R is the connectivity relation R^*

Transitive-Reflexive Closure



Relation with the **minimum possible** number of **extra edges** to make the relation both transitive and reflexive.

The **transitive-reflexive closure** of a relation R is the connectivity relation R^*

n-ary Relations

Let A_1, A_2, \dots, A_n be sets. An ***n*-ary** relation on these sets is a subset of $A_1 \times A_2 \times \dots \times A_n$.

Relational Databases

STUDENT

Student_Name	ID_Number	Office	GPA
Knuth	328012098	022	4.00
Von Neuman	481080220	555	3.78
Russell	238082388	022	3.85
Einstein	238001920	022	2.11
Newton	1727017	333	3.61
Karp	348882811	022	3.98
Bernoulli	2921938	022	3.21

Relational Databases

STUDENT

Student_Name	ID_Number	Office	GPA	Course
Knuth	328012098	022	4.00	CSE311
Knuth	328012098	022	4.00	CSE351
Von Neuman	481080220	555	3.78	CSE311
Russell	238082388	022	3.85	CSE312
Russell	238082388	022	3.85	CSE344
Russell	238082388	022	3.85	CSE351
Newton	1727017	333	3.61	CSE312
Karp	348882811	022	3.98	CSE311
Karp	348882811	022	3.98	CSE312
Karp	348882811	022	3.98	CSE344
Karp	348882811	022	3.98	CSE351
Bernoulli	2921938	022	3.21	CSE351

What's not so nice?

Relational Databases

STUDENT

Student_Name	ID_Number	Office	GPA
Knuth	328012098	022	4.00
Von Neuman	481080220	555	3.78
Russell	238082388	022	3.85
Einstein	238001920	022	2.11
Newton	1727017	333	3.61
Karp	348882811	022	3.98
Bernoulli	2921938	022	3.21

TAKES

ID_Number	Course
328012098	CSE311
328012098	CSE351
481080220	CSE311
238082388	CSE312
238082388	CSE344
238082388	CSE351
1727017	CSE312
348882811	CSE311
348882811	CSE312
348882811	CSE344
348882811	CSE351
2921938	CSE351

Better

Database Operations: Projection

Find all offices: $\Pi_{\text{Office}}(\text{STUDENT})$

Office
022
555
333

Find offices and GPAs: $\Pi_{\text{Office,GPA}}(\text{STUDENT})$

Office	GPA
022	4.00
555	3.78
022	3.85
022	2.11
333	3.61
022	3.98
022	3.21

Database Operations: Selection

Find students with GPA > 3.9 : $\sigma_{\text{GPA}>3.9}(\text{STUDENT})$

Student_Name	ID_Number	Office	GPA
Knuth	328012098	022	4.00
Karp	348882811	022	3.98

Retrieve the name and GPA for students with GPA > 3.9:

$\Pi_{\text{Student_Name,GPA}}(\sigma_{\text{GPA}>3.9}(\text{STUDENT}))$

Student_Name	GPA
Knuth	4.00
Karp	3.98

Database Operations: Natural Join

Student ⋈ Takes

Student_Name	ID_Number	Office	GPA	Course
Knuth	328012098	022	4.00	CSE311
Knuth	328012098	022	4.00	CSE351
Von Neuman	481080220	555	3.78	CSE311
Russell	238082388	022	3.85	CSE312
Russell	238082388	022	3.85	CSE344
Russell	238082388	022	3.85	CSE351
Newton	1727017	333	3.61	CSE312
Karp	348882811	022	3.98	CSE311
Karp	348882811	022	3.98	CSE312
Karp	348882811	022	3.98	CSE344
Karp	348882811	022	3.98	CSE351
Bernoulli	2921938	022	3.21	CSE351