

Graphs, Strings, Languages, and Boolean Logic ^{*a*}

Teodor Rus

rus@cs.uiowa.edu

The University of Iowa, Department of Computer Science

^{*a*}These lecture notes have been developed by Teodor Rus. They are copyrighted materials and may not be used in other course settings outside of the University of Iowa in their current form or modified form without the express written permission of one of the copyright holders. During this course, students are prohibited from selling notes to or being paid for taking notes by any person or commercial firm without the express written permission of one of the copyright holders.

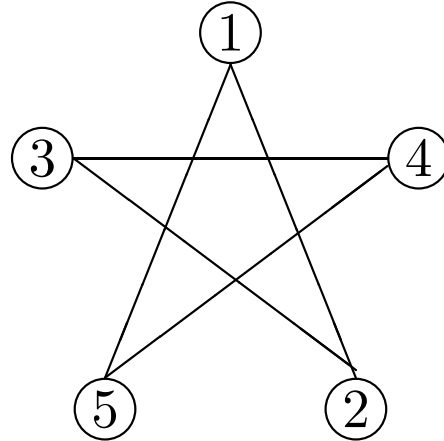
Graphs (informal)

- An *undirected graph*, or simple a *graph*, is a set of points with lines connecting some points;
- The points are called *nodes* or *vertices*, and the lines a called *edges*.

Examples: Figure 1 shows two graphs.

Example graphs

Graph (a)



Graph (b)

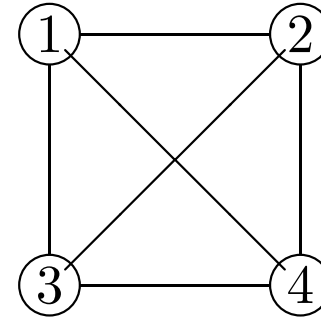


Figure 1: Examples of graphs

Facts

- No more than one edge is allowed between any two nodes;
- The number of edges at a particular node is called the *degree* of that node;
- In Figure 1, Graph (a), each node has the degree 2; in Figure 1, Graph (b), each node has the degree 3.

Edge representation

- In a graph G that contains nodes i and j , the pair (i, j) represents the edge that connects i and j ;
- The order of i and j doesn't matter in an undirected graph, so the pairs (i, j) and (j, i) represent the same edge;
- Because the order of the nodes is unimportant, we can also describe edges by sets such as $\{i, j\}$.

Notation

In a directed graph the edge (i, j) has as the source node i and as target node j .

Formalizing the graph

- If V is the set of nodes of a graph G and E is the set of its edges, we say that $G = (V, E)$;
- Hence, one can specify a graph by a diagram or by specifying the sets V and E .
- **Example:** a formal description of the Graph (a) in Figure 1 is:

$$G = (\{1, 2, 3, 4, 5\}, \{(1, 2), (2, 3), (3, 4), (4, 5), (5, 1)\})$$

Graph usage

- Graphs are frequently used to represent data.
- **Examples:**
 1. nodes might be cities and edges might be the connecting highways;
 2. nodes might be electrical components and edges might be wires between them.
- Sometimes, for convenience, we may label nodes (and edges) of a graph, thus obtaining a *labeled graph*, Figure 2.

Example labeled graph

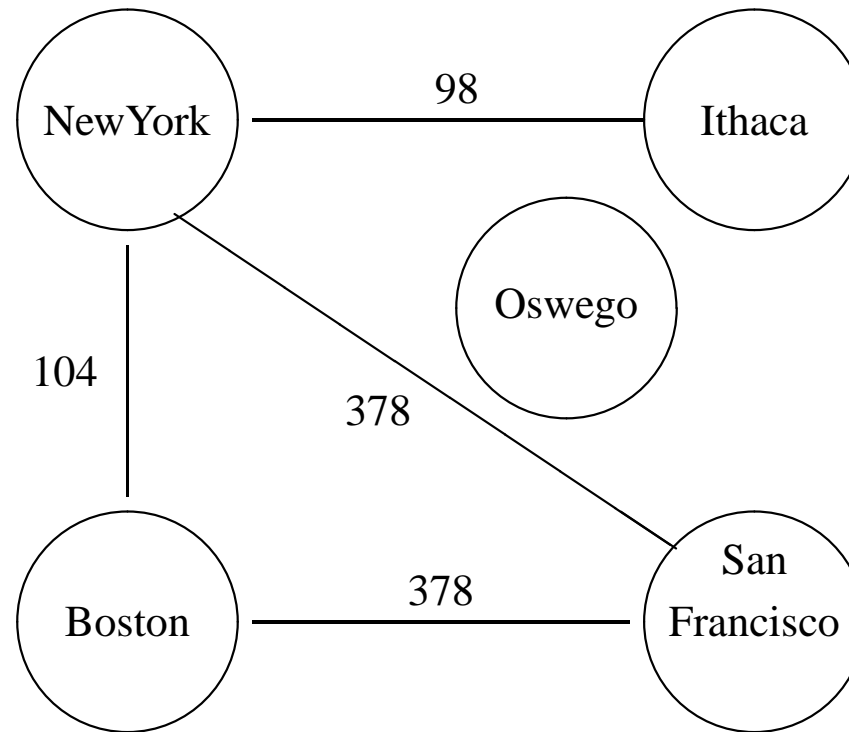


Figure 2: Cheapest air fares between cities

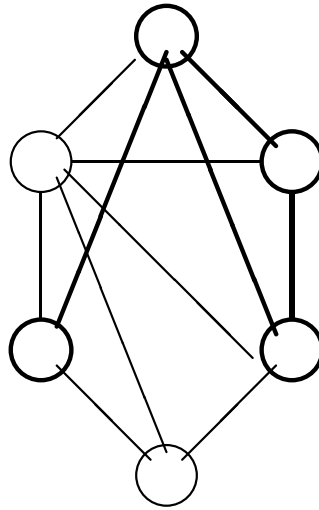
Subgraph

A graph $G = (V_1, E_1)$ is a subgraph of a graph $H = (V_2, E_2)$ if $V_1 \subseteq V_2$.

Note: the edges of G are the edges of H on the corresponding nodes, Figure 3.

Example subgraph

Graph H



Subgraph G , shown darker

Figure 3: Graph G (darker), a subgraph of H

Graph paths

- A *path* in a graph is a sequence of nodes connected by edges.
- A *simple path* is a path that does not repeat any node.
- A graph is *connected* if every two nodes have a path between them.

Graph cycles

- A path is a *cycle* if it starts and ends in the same node.
- A *simple cycle* is a cycle that doesn't repeat any node except the first and the last.

Trees

- A graph is a *tree* if it is connected and has no simple cycles, Figure 4.
- The nodes of degree 1 in a tree are called *leaves*.
- Sometimes there is a specially designated node of a tree called the *root*.

Example graphs

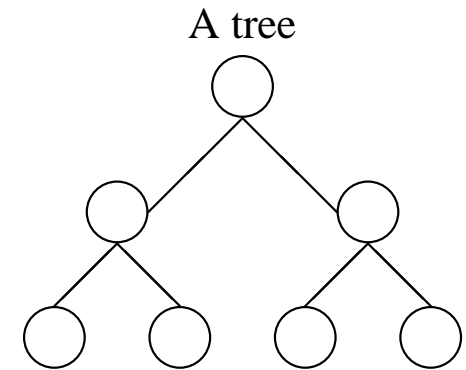
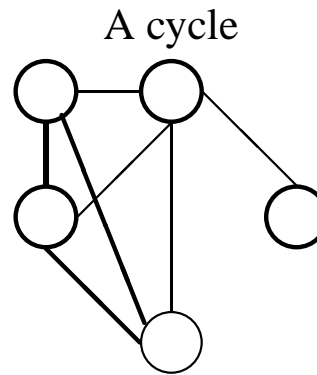
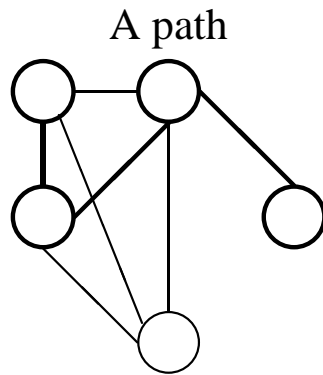


Figure 4: Path, cycle, and tree

Directed graphs

- If the edges of a graph are arrows instead of lines the graph is a *directed graph*.
- The number of arrows pointing from a particular node is the *outdegree* of that node.
- The number of arrows pointing to a particular node is the *indegree* of that node.

Example directed graph

Figure 5 shows a directed graph.

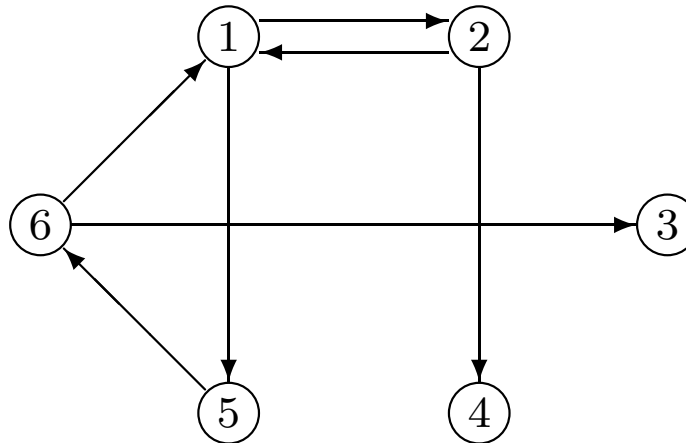


Figure 5: A directed graph

Formal description

- The formal representation of a directed graph G is:

G is a tuple (V, E) where V is the set of nodes and E is the set of directed edges.

- **Example:** formal description of the graph in Figure 5 is:

$G = (\{1, 2, 3, 4, 5, 6\}, \{(1, 2), (1, 5), (2, 1), (2, 4), (5, 6), (6, 1), (6, 3)\})$

Facts

- A path in which all arrows point in the same direction is called a *direct path*.
- A directed graph is *strongly connected* if a directed path connects every two nodes.

Example directed graph

The directed graph in Figure 6 represents the relation that characterizes the game *scissors, paper, stone*:

A game representation

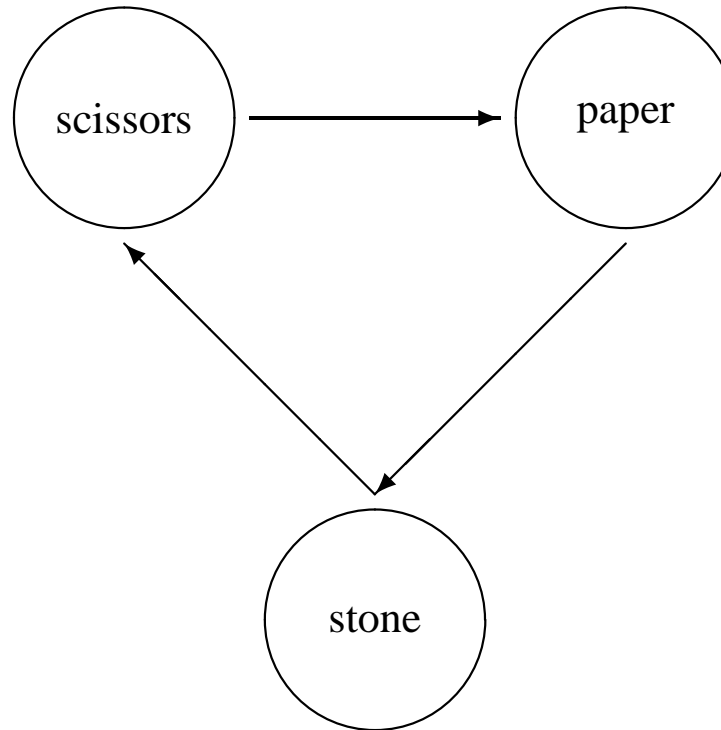


Figure 6: The graph of a relation

Applications

- Directed graphs are a handy way of depicting binary relations.
- If R is a binary relation whose domain and range is D , i.e., $R \subseteq D \times D$, a labeled graph $G = (D, E)$ represents R with $E = \{(x, y) \mid xRy\}$.
- Graph in Figure 6 illustrate this fact.

Strings (informal)

- Strings of characters are fundamental building blocks in CS;
- The alphabet over which strings are defined may vary with application;
- **Alphabet:** is a finite sets;
- Members of the alphabet are the *symbols*.

Notation

- We use Greek letters Σ and Γ to designate alphabets;
- We also use typewriter fonts to denote symbols of an alphabet.

Examples:

$$\Sigma_1 = \{0, 1\};$$

$$\Sigma_2 = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\};$$

$$\Gamma = \{0, 1, x, y, z\}.$$

Strings over an alphabet

- A *string over an alphabet* is a finite sequence of symbols from that alphabet, usually written next to one another.

Examples:

if $\Sigma_1 = \{0, 1\}$ then 01001 is a string over Σ_1 ;

if $\Sigma_2 = \{a, b, c, \dots, z\}$ then abracadabra is a string over Σ_2 .

String properties

- If w is a string over Σ , the *length* of w , written $|w|$, is the number of symbols contained in w ;
- The string of length zero is called the *empty string*, written ϵ ;
- The empty string plays the role of 0 in a number system;
- If $|w| = n$, we can write
 $w = w_1w_2 \dots w_n, w_i \in \Sigma, i = 1, 2, \dots, n.$

More properties

- The reverse of $w = w_1w_2 \dots w_n$, written $w^{\mathcal{R}}$, is $w^{\mathcal{R}} = w_n \dots w_2w_1$;
- A string z is a substring of w if $w = xzy$ for x, y not necessarily the empty strings.

Example:

cad is a substring of abracadabra and $x = abra, y=abra$.

String operations

- **Concatenation:** two strings $x = x_1x_2 \dots x_m$ and $y = y_1y_2 \dots y_n$, by concatenation define a new string $xy = x_1x_2 \dots x_my_1y_2 \dots y_n$;
- The concatenation $xx \dots x$, k -times is written x^k ;
- **Lexicographic ordering:** is the familiar dictionary ordering of strings, where shorter strings precede longer strings.

Example:

Lexicographic ordering of all strings over $\Sigma = \{0, 1\}$ is:

$\{\epsilon, 0, 1, 00, 01, 10, 11, 000, \dots\}$

Strings (formally)

Let Σ be an alphabet. Σ^* denotes the set of all strings over Σ , and is formally defined by the following rules:

1. $\epsilon \in \Sigma^*$
2. For each $x \in \Sigma$, $x \in \Sigma^*$.
3. If $x, y \in \Sigma^*$ then $x \circ y = xy \in \Sigma^*$.

Language (informal)

A language is a set of strings over a given alphabet.

Language (formal)

Let Σ be an alphabet and Σ^* be the set of all strings over Σ .

A language L over Σ is a set $L \subseteq \Sigma^*$.

Fundamental problems

For Σ an alphabet and $L \subseteq \Sigma^*$ a language the following are fundamental problems:

- **Language specification:** devise a specification mechanism, (SM_L) , that discriminates strings $x \in L$ from strings in $y \in \Sigma^*$ and $y \notin L$.
Examples language specification mechanisms?
- **Language recognition:** devise a recognition mechanism, (RM_L) , that for any string $x \in \Sigma^*$ decides whether $x \in L$ or $x \notin L$.
Examples language recognition mechanism?
- **Language translation:** let L_1 and L_2 be languages over the alphabets Σ_1 and Σ_2 , $f : \Sigma_1 \rightarrow \Sigma_2$ a function and $F : \Sigma_1^* \rightarrow \Sigma_2^*$ the semigroup homomorphism induced by f .
Example: Device a translation mechanism $T : L_1 \rightarrow L_2$ that preserves the semigroup structures of Σ_1^* and Σ_2^* on L_1 and L_2 , i.e., if $x, y, xy \in L_1$ then $T(x), T(y), T(x)T(y) \in L_2$.

Questions

- When is a language also:
 - a programming language or
 - a mathematical language or
 - a natural language
- When is a programming language also a natural language?
Example:
- When is a natural language also a programming language?
Examples:

Boolean logic

- Boolean logic is a mathematical system built around two values, `true` and `false`, called boolean values and often represented by 1 and 0, respectively;
- Though originally conceived as pure mathematics, now this system is considered to be the foundation of digital electronics and computer design;
- Boolean values are used in situations with two possibilities such as:
high or low voltage, true or false proposition, yes or no answer.

Boolean operations

Boolean values are manipulated by boolean operations:

1. *Negation* or *NOT*, \neg : $\neg 0 = 1$; $\neg 1 = 0$
2. *Conjunction* or *AND*, \wedge : $0 \wedge 0 = 0$; $0 \wedge 1 = 0$; $1 \wedge 0 = 0$; $1 \wedge 1 = 1$
3. *Disjunction* or *OR*, \vee : $0 \vee 0 = 0$; $0 \vee 1 = 1$; $1 \vee 0 = 1$; $1 \vee 1 = 1$.

Boolean expressions

- Boolean operations are used to combine simple statements into more complex *Boolean expressions* just as the arithmetic operations $+$ and \times are used to construct arithmetic expressions
- **Examples:** Let P and Q be Boolean values representing the truth of statements “the sun is shining” and “today is Monday”:
 - $P \wedge Q$ represent the truth value of statement: “the sun is shining and today is Monday”;
 - $P \vee Q$ represents the truth value of statement: “the sun is shining or today is Monday”;

Other Boolean operations

- *Exclusive or or XOR*, \oplus : $0 \oplus 0 = 0$; $0 \oplus 1 = 1$; $1 \oplus 0 = 1$; $1 \oplus 1 = 0$.
- *Equality*, \leftrightarrow : $0 \leftrightarrow 0 = 1$; $0 \leftrightarrow 1 = 0$; $1 \leftrightarrow 0 = 0$; $1 \leftrightarrow 1 = 1$.
- *Implication*, \rightarrow : $0 \rightarrow 0 = 1$; $0 \rightarrow 1 = 1$; $1 \rightarrow 0 = 0$; $1 \rightarrow 1 = 1$.

Properties

- One can establish various relationship among Boolean operations
- All Boolean operations can be expressed in terms of *AND* and *NOT* by the following identities:

$$P \vee Q = \neg(\neg P \wedge \neg Q)$$

$$P \rightarrow Q = \neg P \vee Q$$

$$P \leftrightarrow Q = (P \rightarrow Q) \wedge (Q \rightarrow P)$$

$$P \oplus Q = \neg(P \leftrightarrow Q)$$

Distribution law

- Distribution law for *AND* and *OR* comes in handy while manipulating Boolean expressions;
- This law is similar to distribution law for addition and multiplication in arithmetic:
$$a \times (b + c) = (a \times b) + (a \times c)$$
- Boolean version: two dual laws:
$$P \wedge (Q \vee R) \text{ equals } (P \wedge Q) \vee (P \wedge R)$$
$$P \vee (Q \wedge R) \text{ equals } (P \vee Q) \wedge (P \vee R)$$

Fact

The dual of the distribution law for addition and multiplication does not hold in general, i.e.

$a + (b * c) \neq (a + b) * (a + c)$ while the dual of a boolean distribution law always hold, i.e., $P \vee (Q \wedge R)$ always equals $(P \vee Q) \wedge (P \vee R)$

Summary

Concept	Meaning
Alphabet	A finite set of objects called symbols
Argument	An input to a function
Binary relation	A relation whose domain is a set of pairs
Boolean operation	An operation on Boolean values
Boolean value	<i>TRUE</i> , <i>FALSE</i> represented by 1, 0
Cartesian product	Operation on sets forming set of all tuples
Complement	Operation on a set forming set of elements not present
Concatenation	Operation that sticks together two strings
Conjunction	Boolean <i>AND</i> operation
Connected graph	Graph with paths connecting every two nodes
Cycle	Path that starts and ends with the same node
Directed graph	Graph whose edges are arrows

Summary, continuation

Concept	Meaning
Disjunction	Boolean <i>OR</i> operation
Domain	The set of possible inputs to a function
Edge	A line connecting two nodes in a graph
Element	An object in a set
Empty set	A set with no elements
Equivalence relation	reflexive, symmetric, and transitive binary relation
Function	A mapping from inputs to outputs
Graph	A collection of nodes and lines connecting some points
Intersection	Operation of forming common elements of two sets
<i>k</i> -tuple	A list of <i>k</i> objects
Language	A set of strings
Member	An object in a set

Summary, continuation

Concept	Meaning
Node	A point in a graph
Pair	A list of two elements, also called a 2-tuple
Path	A sequence of nodes in a graph connected by edges
Predicate	A function whose range is $\{TRUE, FALSE\}$
Property	A predicate
Range	The set from which the outputs of a function are drawn
Relation	Predicate, typically when domain is a set of k -tuples
Sequence	A list of objects
Set	A collection of objects
Simple path	A path without repetition
String	A finite list of symbols from an alphabet
Symbol	A member of an alphabet

Summary, continuation

Concept	Meaning
Tree	A graph without cycles
Union	Operation combining elements of two sets into a single set
Vertex	A point in a graph