

Theory of computing: Background

1. Logic
2. Sets
3. Relations and functions
4. Proof techniques
5. Graphs and trees

1. Logic

All values in the language of logic are truth values in $\{false, true\}$

Operators in propositional logic:

Negation not \neg

Disjunction or \vee

Conjunction and \wedge

Implication if ... then \Rightarrow

Equivalence iff \Leftrightarrow

Implication and equivalence

$p \Rightarrow q$ is true whenever

- q is true
- Both p and q are false

Assertions p and q are *equivalent*

($p = q, p \Leftrightarrow q$), whenever

- $(p \Rightarrow q) \wedge (q \Rightarrow p)$
- p iff q
- Both p and q are true
- Both p and q are false

Quantifiers in predicate logic

\forall universal for all

\exists existential there exists

- *Example:* $(\forall x \in \mathbb{N}) (\exists x')$ s.t. $x' = x + 1$
(Every natural number has a successor)
- Predicate logic is characterized by the use of quantifiers, predicates, and other functions.
- A *predicate* is a function that returns a truth value

2. Sets

- A set is an aggregation of items, real, imaginary, or abstract, taken as a whole
- Sets may be defined by
 - Enumeration: $A = \{1, 2, 4\}$
 - Membership predicate
 $A = \{x \mid is_odd(x)\}$
- Sets are defined without respect to order:
 $\{1, 2, 4\} = \{2, 1, 4\}$
- A set contains no duplicates:
 $\{1, 2, 4\} = \{1, 2, 4, 2\}$

Union, intersection, complement

- *Membership*: “ $x \in A$ ” means x is a member of A
- *Union* of A and B :
 $(A \cup B) = \{x \mid x \in A \vee x \in B\}$
- *Intersection* of A and B :
 $(A \cap B) = \{x \mid x \in A \wedge x \in B\}$
- *Relative complement* of A w.r.t B :
 $(A - B) = \{x \mid x \in A \wedge x \notin B\}$
- *Generalized union (intersection)*:
 $\cup \{A, B, C\} = A \cup B \cup C$

Inclusion (subsets)

- $A \subseteq B$ (set A is a subset of set B) means that any member of A that is a member of B
- Strict inclusion: $A \subset B$ means $A \subseteq B \wedge A \neq B$
- If $A \subseteq B$ then $B \supseteq A$ (B is a superset of A)
- For every set A , $A \subseteq A$
- The *null set* \emptyset
 - has no members
 - is a subset of all sets
- The *power set* of A , written 2^A , is the set of all subsets of A

3. Relations and functions

- The *Cartesian product* of n sets is the set of all n -tuples chosen from the sets
- *Relation*: a subset of a Cartesian product
- *Function*: a relation of which each left-hand member of a tuple is in not more than one tuple (maps to a unique value)
- *Examples*: *EVEN* is a function,
> is a relation

Relations and databases

- A database table is a relation
- The Cartesian product of *students* and *courses* is the set of all possible pairings of course, student
- If R is a relation and $(x, y) \in R$, we write $x R y$
- A relation R on set A is
 - *Reflexive* iff for all $x \in A$, $x R x$
 - *Symmetric* iff $(\forall x, y \in A) x R y \Rightarrow y R x$
 - *Transitive* iff $(\forall x, y, z \in A) x R y \wedge y R z \Rightarrow x R z$
- A relation that is reflexive, symmetric, and transitive is an *equivalence relation*

Functions

- Computation of a function takes parameter as input, return value as output
- Every function has a
 - *Domain*: set of values mapped from
 - *Range*: set of values mapped to
- *Injective* (one-to-one) function: one for which $(f(x) = f(y)) \Rightarrow (x = y)$
- *Surjection*: one for which all values in range are mapped to by some value in the domain
- *Bijection*: injective and surjective function

4. Proof techniques

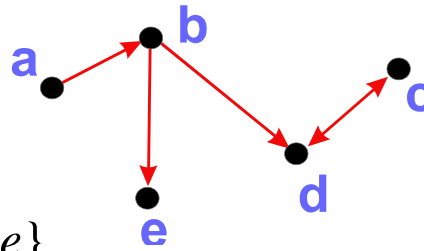
- *Direct*: straightforward step by step inference
- *By contradiction*: Suppose that the claim we wish to disprove is true, then show that this leads to a contradiction
- *Inductive*, using Induction Principle:
 - For a set $A \subseteq \mathbb{N}$, if $0 \in A$
 - and if $x \in A \Rightarrow (x + 1) \in A$, then
 - $A = \mathbb{N}$
- For examples, see handout sheets

5. Graphs and trees

- A graph $G = (V, E)$ is a set V of vertices and a set E of directed or undirected edges that connect pairs of vertices
- Graphs may have weighted edges
- Model communication relations, state transitions, precedence in time
- Algorithms exist to find paths, minimum-weight paths, minimal spanning trees
- *Tree*: a connected acyclic graph

Relations and directed graphs

A set of vertices V and a set of edges E connecting the vertices. (E is a relation on V .)



$$V = \{a, b, c, d, e\}$$

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

Applications of graphs

- Network topologies: star, linear bus, ring
- Network routing algorithms
- Scheduling (e.g., a prerequisites diagram)
- Finite automata (vertices are states; edges are transitions between states)
- Trees to represent hierarchies of all kinds

Edges in a digraph

- An edge in a directed graph relates a vertex to another vertex
- *Example:* Who has sent email to whom? What intersections are connected by streets?
- A *relation* associates elements of sets, such as senders and recipients of email
- [pic]

Cartesian product of two sets:

- The set of ordered pairs defined by selecting one member of each
- *Example:*

$$\{1,2\} \times \{a,b\} = \{(1,a), (1,b), (2,a), (2,b)\}$$

	a	b
1	(1,a)	(1,b)
2	(2,a)	(2,b)

Relation:

- A subset of a Cartesian product

- *Example:*

$R = \{(1, a), (2, b)\}$ is a relation

$R \subseteq \{1,2\} \times \{a,b\}$

R	a	b
1	T	F
2	F	T

Example: $>$ is a relation

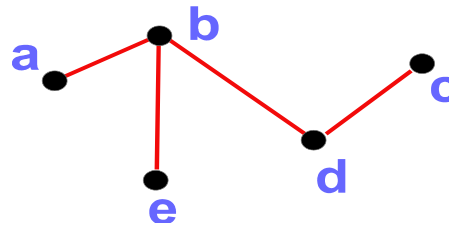
>	1	2	3	4
1	F	F	F	F
2	T	F	F	F
3	T	T	F	F
4	T	T	T	F

- Because $2 > 1$; $3 > 1$; $3 > 2$, etc.
- A relation is a set of ordered pairs:

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

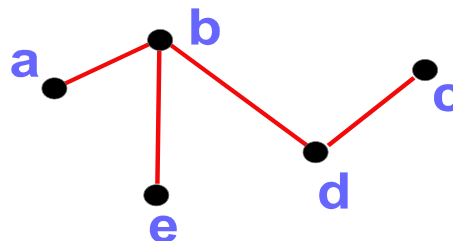
Undirected graphs

- Edges are segments, not arrows
- Undirected graphs are equivalent to directed ones in which all edges are double-arrowed



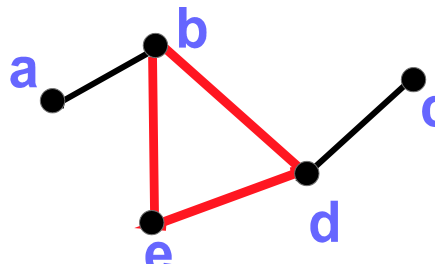
Paths

- *Path*: A sequence of pairwise adjacent vertices whose edges connect two vertices in a graph
- Path from a to d : (a,b,d)



Cycles

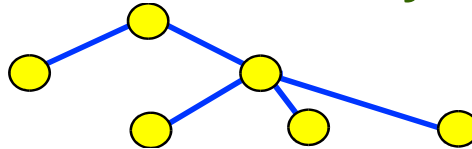
- *Cycle*: a path from one vertex to itself that does not repeat an edge
- Cycle below: (b, e, d)
- A graph with no cycles is *acyclic*



A tree is hierarchic

- Examples:
 - Family tree
 - Organizational hierarchy
 - Module hierarchy chart
 - Taxonomy
 - Parse tree
 - Disk directory
- A tree has a *root* at top, *leaves* at bottom
- A nonroot node has one *parent* node
- A nonleaf has one or more *child* nodes
- Every tree node is the root of a subtree

Tree: a connected acyclic graph



- Exactly one path joins any pair of vertices
- Removing an edge disconnects the tree
- Adding any edge creates a cycle
- Cardinalities of edges and vertices:
 $|E| = |V| - 1$
- *Degree* of a vertex is the number of edges that include it

M-ary and binary trees

- An *m*-ary tree is one in which no node has more than *m* children
- A *binary* tree is one in which no node has more than two children
- A *complete* binary tree is one in which every node but the leaves has two children
- A *full* binary tree is a complete binary tree in which the leaves are all at the same level

Parse trees

```

graph TD
    expression[expression] --- term1[term]
    expression --- op[addition operator]
    expression --- term2[term]
    term1 --- factor1[factor]
    factor1 --- 2[2]
    op --- plus[+]
    term2 --- factor2[factor]
    term2 --- mul[multiplication operator]
    term2 --- factor3[factor]
    factor2 --- 5[5]
    mul --- asterisk[*]
    factor3 --- 3[3]
    
```

- Compilers build a program structure from tokens
- Root may be *program*
- Lexemes are leaves of tree
- Nonterminal syntax elements (e.g., *expression*, *factor*) are internal tree nodes or the root

David Keil Theory of Computing 4/11 25

Height of a complete binary tree with n nodes is $O(\log_2 n)$

$O(\log_2 n)$ levels ← $(n + 1) / 2$ nodes

- ...because size n doubles with each level added
- Or, $2^{\text{height}-1} \leq n \leq 2^{\text{height}} - 1$

David Keil Theory of Computing 4/11 26

References