CS3203 #8

6/16/04

Janak J Parekh

Administrivia

- Exam statistics: 76.9 ± 12.3 **out of 90** (85.4%)
  - Go over answers…
- We lost one student ☹
- HW4 will go out today

Relations

- Quick review
  - A *binary relation* from A to B is a **subset** of A x B.
  - We use the notation a R b if (a, b) $\in$ R and a R b (where R is struck out) if they're not. If they are, a is said to be **related to** b by R.
- A relation on the set A is a relation from A to A.
  - Let A be the set {1, 2, 3, 4}; which ordered pairs are in the relation R = { (a, b) | a divides b}
- Several properties of relations… let R be the relation on {a, b, c, d}:
  - R = {(a,a), (a,c), (a,d), (b,a), (b,b), (b,c), (b,d), (c,b), (c,c), (d,b), (d,d))
  - Is it
    - Reflexive? Yes.
    - Irreflexive? No.
    - Symmetric? No (a,c) but no (c,a)
    - Asymmetric? No (b,c) and (c,b)
    - Antisymmetric? No (b,c) and (c,b)
    - Transitive? No (a,c) (c,b) no (a,b)

Combining relations

- Union, intersection, subtraction all work
- *Composite* of R and S, R from A to B and S from B to C, produces a new relation from A to C.   Termed S o R (note backwards)
- Powers R^n, n = 1, 2, 3… are defined by $R^1 = R$ and $R^{n+1} = R^n \circ R$.
  - Example: R = {(a,a), (b,c), (c,a)}… what's R^2?
- R on A is transitive if and only if $R^n \subseteq R$ for n = 1, 2, 3…

n-ary relations

- All the examples we've seen so far are between two sets.
- An n-ary relation on sets A_1, A_2, …, A_n is a subset of A_1 x A_2 x … x A_n. The sets are the *domains* of the relation, and *n* is the *degree*.
  – Let R be the relation on N x N x N consisting of triples (a,b,c) where a < b < c.
  – All the sets don't have to be the same; remember that tuples are ordered
- Basis for modeling databases, and the relational data model in particular.
  – A **record** is an n-tuple, made up of **fields**.
  – For example, (STUDENT NAME, ID NUMBER, MAJOR, GPA)
  – **Tables** can be used to represent these relations; each column represents an *attribute* of the database.
  – A domain of the *n*-ary relation is a **primary key** when the value of the n-tuple from this domain determines the n-tuple (i.e., uniqueness).
  – Should always remain a primary key (i.e., over the **intension** of a database, which contains all possible theoretical n-tuples).
  – Sometimes, need several fields to form a **composite key** to determine uniqueness.

Operations on n-ary relations

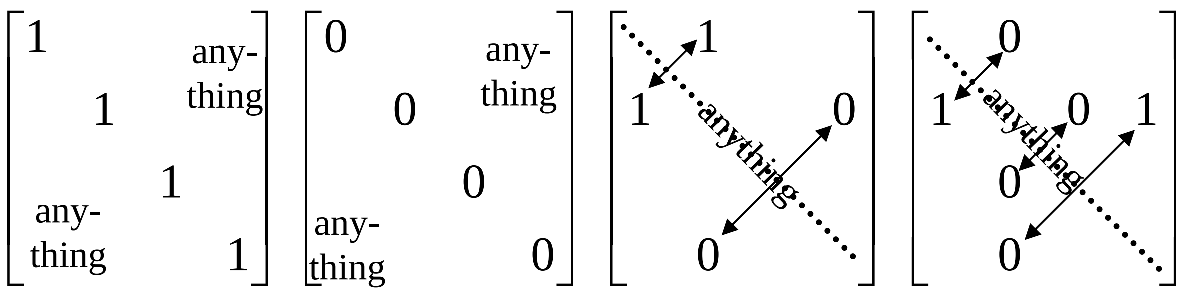
- Let R be an n-ary relation and C a condition that elements in R may satisfy. The *selection operator* s_C maps the n-ary relation R to the n-ary relation of n-tuples from R that satisfy C.
  - Example, s_{C1}, where C_1 is the condition "Major = Computer Science".
- The **projection** $P_{i1i2\ldots im}$ maps the n-tuple $(a_1, a_2, \ldots, a_n)$ to the m-tuple $(a_{i1}, a_{i2}, \ldots, a_{im})$ where $m <= n$.
  - What's $P_{1,4}$ on (49, 21, 45, 6)?
- The two combine to form a "view" on a database.
- A **join** $J_p(R,S)$ where $p < m$ and $p < n$, is a relation of degree m+n-p that consists of the (m+n-p)-tuples where the m-tuple belongs to R and the n-tuple belongs to S and they have some common components $c_1 \ldots c_p$.
- Example: join teaching assignments and class schedule; see page 486.
- SQL SELECT can be used to express these; I'll leave it as a optional reading exercise unless people really want to see…

Representing relations

- List ordered pairs
- Matrices
  - For a binary relation, we define $M_R = [m_{ij}]$, where $m_{ij} = 1$ if $(a_i, b_j) \in R$ and 0 if not.
  - Let $A = \{a_1, a_2, a_3\}$ and $B = \{b_1, b_2, b_3, b_4, b_5\}$. What ordered pairs are in the matrix

$$M_R = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

How to represent properties using matrices?



Reflexive: all 1's on diagonal	*Irreflexive*: all 0's on diagonal	*Symmetric*: all identical across diagonal	*Antisymmetric* all 1's are across from 0's

- Note this is relation of a single set, which has to be square
- Example: Given M_R, what is it?
- More discussion in book

$$M_R = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Using digraphs

- Preview of next chapter, strangely
- "Directed graph" consists of a set V of vertices (nodes) together with a set E of edges (arcs). The vertex a is called the initial vertex on the edge (a,b), and the vertex b is called the terminal vertex.
- An edge (a,a) forms a self-loop.
- Example: Given S = {a,b,c,d} and R on S = {(a,b), (a,d), (b,b), (b,d), (c,a), (c,b), (d,b)}.
- How to determine reflexivity, symmetry, antisymmetry, transitivity?

Closures of relations

- Useful transformation for a large variety of applications
- Suppose we have a relation that documents telephone links between five different data centers
  - Not completely interconnected
  - How can we find out if each node is reachable from each other? *Transitive closure*
- In general, given R over A; if there is a relation S with property P containing R such that S is a subset of ever relation with property P containing R, then S is called the closure of R with respect to P.
- We'll discuss reflexive, symmetric, and transitive closures…

Reflexive closure

- Given R on A, the reflexive closure of R can be formed by adding to R all pairs of the form (a,a) with $a \in A$, not already in R. This process produces a relation that's reflexive in R.
  - Compute reflexive closure of {(1,2),(2,3),(3,4)} over {1,2,3,4}
  - Can be viewed as R U Δ where the latter term is $\{(a,a) \mid a \in A\}$, or the **diagonal relation** on A
  - What's the closure of R = {(a,b) | a < b} over Z?

Symmetric closure

- Symmetric closure can be created by adding all pairs of the form (b,a), where (a,b) exists but where (b,a) doesn't.
- Alternatively, take the union of the R with its inverse R^{-1} where the inverse = $\{(b,a) \mid (a,b) \in R\}$.
- What's the symmetric closure on $\{(a,b) \mid a < b\}$ over Z?

Transitive closure

- Not quite so easy to do
- Given {(1,3),(1,4)(2,1),(3,2)}, if we compute the transitive results of each pair, we do *not* get the transitive closure
  – (1,2), (2,3), (2,4), (3,1), if unioned with the original relation, still doesn't give a closure that holds for transitivity.
  – Simply put, need to add ordered pairs over and over and over again until there are no more needed.
  – How can we do this procedurally? We'll see…

Paths in directed graphs

- By representing relations using digraphs, it'll help us define a transitive closure.
- A path from a to b in a graph G is a sequence of edges (x_0, x_1), (x_1, x_2), (x_2, x_3), …, (x_{n-1}, x_n) in G, where n is a nonnegative integer, and $x_0 = a$ and $x_n = b$. This path is denoted as $x_0, x_1, \ldots, x_{n-1}, x_n$ and has length *n*.
- Construct an example…
- Theorem: let R over A. There is a path of length *n*, where *n* is a positive integer, from *a* to *b* if and only if $(a,b) \in R^n$.

Transitive closure, redux

- Let R be on A. The **connectivity relation** R* consists of the pairs (a,b) such that there is a path of at least one from a to b in R.
  - R* = the union of R^ns, n = 1 to infinity
  - Let R be the relation on the set of all subway stops in NYC that contains (a,b) if it is possible to travel between a and b without changing trains. What is R^n for positive n? What is R*?
- The transitive closure of R equals R*.
  - Given set A with n elements and R on A, if there is a path of length at least one in R from a to b, then there is such a path with length not exceeding *n*. Moreover, if a is not b, there is a path not exceeding *n-1*.
    - What does this mean?
    - Provable by pigeonhole (remove the extra circuit in the path)
- Can also "or" derivative matrices together
  - To be precise, n matrices where n = number of elements in S

Warshall's algorithm

- Reasonably efficient way of computing transitive closure. $O(n^3)$
  - Sounds like a lot, but the matrix multiplication is $O(n^4)$!
- Done over a matrix:
  procedure Warshall(M_R: n x n zero-one matrix)
  W := M_R
  for k := 1 to n; begin
  for i := 1 to n; begin
  for j := 1 to n
  $w_{ij} = w_{ij} \vee (w_{ik} \wedge w_{kj})$
  end
  end {W is M_{R^*}}
- Fundamental idea: for all vertices *k*, compute paths between i and j for which k is an intermediate
- Floyd's algorithm is a variation on this which calculates shortest paths (Wednesday)

Next time

- Finish up equivalence relations, partial orderings
- Start graphs