Today’s topics

• Relations
  – Kinds of relations
  – n-ary relations
  – Representations of relations

• Reading: Sections 7.1-7.3

• Upcoming
  – Minesweeper
Binary Relations

- Let $A$, $B$ be any sets. A *binary relation* $R$ from $A$ to $B$, written (with signature) $R:A \times B$, or $R:A,B$, is (can be identified with) a subset of the set $A \times B$.
  - *E.g.*, let $\langle \colon \mathbb{N} \leftrightarrow \mathbb{N} \rangle := \{(n,m) \mid n < m\}$
- The notation $a R b$ or $aRb$ means that $(a,b) \in R$.
  - *E.g.*, $a < b$ means $(a,b) \in \langle$.
- If $aRb$ we may say “$a$ is related to $b$ (by relation $R$)”,
  - or just “$a$ relates to $b$ (under relation $R$)”.
- A binary relation $R$ corresponds to a predicate function $P_R:A \times B \to \{T,F\}$ defined over the 2 sets $A,B$;
  - *e.g.*, predicate “eats” $:= \{(a,b)\mid$ organism $a$ eats food $b\}$
Complementary Relations

• Let $R:A,B$ be any binary relation.
• Then, $\bar{R}:A\times B$, the complement of $R$, is the binary relation defined by
  $$\bar{R} : \{(a,b) \mid (a,b) \notin R\} = (A\times B) - R$$
• Note this is just $\bar{R}$ if the universe of discourse is $U = A\times B$; thus the name complement.
• Note the complement of $\bar{R}$ is $R$.

Example: $\prec = \{(a,b) \mid (a,b) \notin \prec\} = \{(a,b) \mid \neg a < b\} = \succeq$
Inverse Relations

• Any binary relation $R:A \times B$ has an inverse relation $R^{-1}:B \times A$, defined by
  
  \[ R^{-1} \equiv \{(b,a) \mid (a,b)\in R\}. \]

  E.g., $\prec^{-1} = \{(b,a) \mid a< b\} = \{(b,a) \mid b>a\} = >$.

• E.g., if $R$:People→Foods is defined by $a \mathrel{R} b \iff a$ eats $b$, then:
  
  \[ b \mathrel{R^{-1}} a \iff b$ is eaten by $a. \] (Passive voice.)
Relations on a Set

• A (binary) relation from a set $A$ to itself is called a relation on the set $A$.

• *E.g.*, the “$<$” relation from earlier was defined as a relation on the set $\mathbb{N}$ of natural numbers.

• The (binary) identity relation $I_A$ on a set $A$ is the set $\{(a,a) | a \in A\}$. 
Reflexivity

- A relation $R$ on $A$ is *reflexive* if $\forall a \in A, aRa$.
  - *E.g.*, the relation $\geq := \{(a,b) \mid a \geq b\}$ is reflexive.

- A relation $R$ is *irreflexive* iff its *complementary* relation $R^c$ is reflexive.
  - Example: $<$ is irreflexive, because $\geq$ is reflexive.
  - Note “irreflexive” does NOT mean “not reflexive”!
    - For example: the relation “likes” between people is not reflexive, but it is not irreflexive either.
      - Since not everyone likes themselves, but not everyone dislikes themselves either!
Symmetry & Antisymmetry

• A binary relation $R$ on $A$ is *symmetric* iff $R = R^{-1}$, that is, if $(a,b) \in R \iff (b,a) \in R$.
  - *E.g.*, $=$ (equality) is symmetric. $<$ is not.
  - “is married to” is symmetric, “likes” is not.

• A binary relation $R$ is *antisymmetric* if $orall a \neq b, (a,b) \in R \rightarrow (b,a) \notin R$.
  - *Examples*: $<$ is antisymmetric, “likes” is not.
  - *Exercise*: prove this definition of antisymmetric is equivalent to the statement that $R \cap R^{-1} \subseteq I_A$.  


Transitivity

- A relation $R$ is transitive iff (for all $a,b,c$)
  $$(a,b)\in R \land (b,c)\in R \rightarrow (a,c)\in R.$$  
- A relation is intransitive iff it is not transitive.
- Some examples:
  - “is an ancestor of” is transitive.
  - “likes” between people is intransitive.
  - “is located within 1 mile of” is…?
Totality

- A relation $R: A \times B$ is *total* if for every $a \in A$, there is at least one $b \in B$ such that $(a,b) \in R$.
- If $R$ is not total, then it is called *strictly partial*.
- A *partial relation* is a relation that might be strictly partial. (Or, it might be total.)
  - In other words, all relations are considered “partial.”
Functionality

• A relation \( R:A \times B \) is \textit{functional} if, for any \( a \in A \), there is \textit{at most 1} \( b \in B \) such that \((a,b) \in R\).
  
  – “\( R \) is functional” \( \iff \) \( \forall a \in A : \neg \exists b_1 \neq b_2 \in B : aRb_1 \land aRb_2 \).
  
  – Iff \( R \) is functional, then it corresponds to a partial function \( R:A \to B \)
    
    • where \( R(a) = b \iff aRb; \ \text{e.g.} \)
      
      – E.g., The relation \( aRb : = “a + b = 0” \) yields the function \( -(a) = b \).

• \( R \) is \textit{antifunctional} if its inverse relation \( R^{-1} \) is functional.
  
  – Note: A functional relation (partial function) that is also antifunctional is an \textit{invertible} partial function.

• \( R \) is a \textit{total function} \( R:A \to B \) if it is both functional and total, that is, for any \( a \in A \), there is \textit{exactly 1} \( b \) such that \((a,b) \in R\).

I.e., \( \forall a \in A : \neg \exists ! b : aRb \).

  – If \( R \) is functional but not total, then it is a \textit{strictly partial function}.
  
  – \textbf{Exercise:} Show that iff \( R \) is functional and antifunctional, and both it and its inverse are total, then it is a bijective function.
Lambda Notation

• The lambda calculus is a formal mathematical language for defining and operating on functions.
  – It is the mathematical foundation of a number of functional (recursive function-based) programming languages, such as LISP and ML.
• It is based on lambda notation, in which “\( \lambda a: f(a) \)” is a way to denote the function \( f \) without ever assigning it a specific symbol.
  – E.g., \( (\lambda x. 3x^2+2x) \) is a name for the function \( f: \mathbb{R} \rightarrow \mathbb{R} \) where \( f(x) = 3x^2+2x \).
• Lambda notation and the “such that” operator “\( \exists \)” can also be used to compose an expression for the function that corresponds to any given functional relation.
  – Let \( R: A \times B \) be any functional relation on \( A, B \).
  – Then the expression \( (\lambda a: b \in aRb) \) denotes the function \( f: A \rightarrow B \) where \( f(a) = b \) iff \( aRb \).
    • E.g., If I write: \( f : (\lambda a: b \in a+b = 0) \),
      this is one way of defining the function \( f(a) = -a \).
Composite Relations

- Let $R: A \times B$, and $S: B \times C$. Then the composite $S \circ R$ of $R$ and $S$ is defined as:
  \[ S \circ R = \{(a,c) \mid \exists b: aRb \land bSc\} \]
- Note that function composition $f \circ g$ is an example.
- Exer.: Prove that $R: A \times A$ is transitive iff $R \circ R = R$.
- The $n^{th}$ power $R^n$ of a relation $R$ on a set $A$ can be defined recursively by:
  \[ R^0 := I_A ; \quad R^{n+1} := R^n \circ R \quad \text{for all } n \geq 0. \]
  - Negative powers of $R$ can also be defined if desired, by $R^{-n} := (R^{-1})^n$. 
§7.2: \( n \)-ary Relations

- An \( n \)-ary relation \( R \) on sets \( A_1, \ldots, A_n \), written (with signature) \( R : A_1 \times \ldots \times A_n \) or \( R : A_1, \ldots, A_n \), is simply a subset \( R \subseteq A_1 \times \ldots \times A_n \).

- The sets \( A_i \) are called the \textit{domains} of \( R \).

- The \textit{degree} of \( R \) is \( n \).

- \( R \) is \textit{functional in the domain} \( A_i \) if it contains at most one \( n \)-tuple \((\ldots, a_i, \ldots)\) for any value \( a_i \) within domain \( A_i \).
Relational Databases

• A *relational database* is essentially just an $n$-ary relation $R$.
• A domain $A_i$ is a *primary key* for the database if the relation $R$ is functional in $A_i$.
• A *composite key* for the database is a set of domains $\{A_i, A_j, \ldots\}$ such that $R$ contains at most 1 $n$-tuple $(\ldots,a_i,\ldots,a_j,\ldots)$ for each composite value $(a_i, a_j,\ldots) \in A_i \times A_j \times \ldots$
Selection Operators

• Let $A$ be any $n$-ary domain $A = A_1 \times \ldots \times A_n$, and let $C:A \rightarrow \{T,F\}$ be any condition (predicate) on elements ($n$-tuples) of $A$.

• Then, the selection operator $s_C$ is the operator that maps any ($n$-ary) relation $R$ on $A$ to the $n$-ary relation of all $n$-tuples from $R$ that satisfy $C$.

  – I.e., $\forall R \subseteq A$, $s_C(R) = \{ a \in R \mid s_C(a) = T \}$
Selection Operator Example

- Suppose we have a domain
  \[ A = \text{StudentName} \times \text{Standing} \times \text{SocSecNos} \]
- Suppose we define a certain condition on \( A \),
  \[ \text{UpperLevel}(\text{name}, \text{standing}, \text{ssn}) \equiv \]
  \[ [(\text{standing} = \text{junior}) \lor (\text{standing} = \text{senior})] \]
- Then, \( s_{\text{UpperLevel}} \) is the selection operator that takes any relation \( R \) on \( A \) (database of students) and produces a relation consisting of just the upper-level classes (juniors and seniors).
Projection Operators

- Let $A = A_1 \times \ldots \times A_n$ be any $n$-ary domain, and let $\{i_k\} = (i_1, \ldots, i_m)$ be a sequence of indices all falling in the range 1 to $n$,
  - That is, where $1 \leq i_k \leq n$ for all $1 \leq k \leq m$.
- Then the projection operator on $n$-tuples

$$P_{\{i_k\}} : A \rightarrow A_{i_1} \times K \times A_{i_m}$$

is defined by:

$$P_{\{i_k\}}(a_1, \ldots, a_n) = (a_{i_1}, \ldots, a_{i_m})$$
Projection Example

• Suppose we have a ternary (3-ary) domain $\text{Cars}=\text{Model} \times \text{Year} \times \text{Color}$. (note $n=3$).

• Consider the index sequence $\{i_k\} = 1,3$. ($m=2$)

• Then the projection $P_{\{i_k\}}$ simply maps each tuple $(a_1,a_2,a_3) = (\text{model}, \text{year}, \text{color})$ to its image:

$$ (a_{i_1}, a_{i_2}) = (a_1, a_3) = (\text{model}, \text{color}) $$

• This operator can be usefully applied to a whole relation $R \subseteq \text{Cars}$ (a database of cars) to obtain a list of the model/color combinations available.
Join Operator

- Puts two relations together to form a sort of combined relation.
- If the tuple \((A,B)\) appears in \(R_1\), and the tuple \((B,C)\) appears in \(R_2\), then the tuple \((A,B,C)\) appears in the join \(J(R_1,R_2)\).
  - \(A\), \(B\), and \(C\) here can also be sequences of elements (across multiple fields), not just single elements.
Join Example

- Suppose $R_1$ is a teaching assignment table, relating Professors to Courses.
- Suppose $R_2$ is a room assignment table relating Courses to Rooms, Times.
- Then $J(R_1,R_2)$ is like your class schedule, listing (professor, course, room, time).
§7.3: Representing Relations

• Some ways to represent $n$-ary relations:
  – With an explicit list or table of its tuples.
  – With a function from the domain to $\{T,F\}$.
    • Or with an algorithm for computing this function.

• Some special ways to represent binary relations:
  – With a zero-one matrix.
  – With a directed graph.
Using Zero-One Matrices

• To represent a binary relation \( R : A \times B \) by an \(|A| \times |B|\) 0-1 matrix \( M_R = [m_{ij}] \), let \( m_{ij} = 1 \) iff \((a_i,b_j) \in R\). 

• *E.g.*, Suppose Joe likes Susan and Mary, Fred likes Mary, and Mark likes Sally.

• Then the 0-1 matrix representation of the relation \( \text{Likes: Boys} \times \text{Girls} \) relation is:

\[
\begin{array}{ccc}
\text{Susan} & \text{Mary} & \text{Sally} \\
\text{Joe} & 1 & 1 & 0 \\
\text{Fred} & 0 & 1 & 0 \\
\text{Mark} & 0 & 0 & 1 \\
\end{array}
\]
Zero-One Reflexive, Symmetric

- **Terms:** Reflexive, non-reflexive, irreflexive, symmetric, asymmetric, and antisymmetric.
  - These relation characteristics are very easy to recognize by inspection of the zero-one matrix.

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 0 & 0 \\
1 & 0 & 1 \\
\end{bmatrix}
\]

- **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
Using Directed Graphs

- A directed graph or digraph $G=(V_G,E_G)$ is a set $V_G$ of vertices (nodes) with a set $E_G \subseteq V_G \times V_G$ of edges (arcs, links). Visually represented using dots for nodes, and arrows for edges. Notice that a relation $R:A \times B$ can be represented as a graph $G_R=(V_G=A \cup B, E_G=R)$.

Matrix representation $M_R$:

\[
\begin{bmatrix}
\text{Susan} & \text{Mary} & \text{Sally} \\
\hline
\text{Joe} & 1 & 1 & 0 \\
\text{Fred} & 0 & 1 & 0 \\
\text{Mark} & 0 & 0 & 1
\end{bmatrix}
\]

Graph rep. $G_R$:

- Edge set $E_G$ (blue arrows)
- Node set $V_G$ (black dots)
Digraph Reflexive, Symmetric

It is extremely easy to recognize the reflexive/irreflexive/symmetric/antisymmetric properties by graph inspection.

Reflexive: Every node has a self-loop
Irreflexive: No node links to itself
Symmetric: Every link is bidirectional
Antisymmetric: No link is bidirectional

These are asymmetric & non-antisymmetric
These are non-reflexive & non-irreflexive