Relations

- We have seen several types of mathematical objects over the course of the semester: propositions, sets, ordered pairs, and functions.
- **Relations** use ordered tuples to represent relationships among objects.
- **Examples:**
  - "x is a parent of y " $-$ $(Morris, Steve)$, $(Ria, Steve)$
  - "x is a number less than y" $-$ (3, 42), (42, 43)
  - "Student number x is named y and majors in z" $-$ (124324443, $Mary, CSE$), (563565426, $Mary, PSY$)
  - "x is an even number" ... (2)
- Essentially, a relation is the set of assignments which makes a predicate true.
- **Examples:**
  - $IsParent = \{(Morris, Steve),\ (Ria, Steve)\}$
  - $LessThan = \{(3, 42), (42, 43)\}$
  - $MajorIn = \{(124324443, Mary, CSE),$ $(563565426, Mary, PSY)\}$
  - $IsEven = \{n \mid n = 2k\}$

Binary Relations

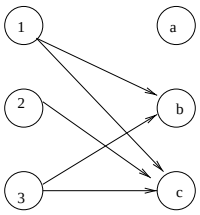
- Binary relations have two blanks, relating two objects.
- More formally, suppose A and B are sets.
  A **binary relation** from A to B is a set $R \subseteq A \times B$.
- Thus R is a set of ordered pairs (a, b) where $a \in A$ and $b \in B$.
- **Notation:** If $(a, b) \in R$ then we sometimes write aRb.
- **Example:**
  $A = \{2, 6, 7\}$, $B = \{1, 2, 5\}$.
  R_1 is "x in A is an integer multiple of y in B."
  so $R_1 = \{(2, 1), (2, 2), (6, 1), (6, 2), (7, 1)\}$

Presenting Binary Relations

- Binary relations are particularly useful because they have two kinds of compact visual representation, **tables** and **graphs**.
- **Tables:**

R	a	b	c
1		*	*
2			*
3		*	*

- **Graphs** are composed of **vertices** or **nodes** connected by **edges** or **arcs**.
  There is an arc from a to b iff $(a, b) \in R$

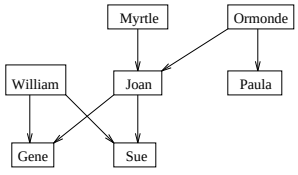


The Parent-Of Relation

- The parent of relations, "x is a parent of y", is a binary relation between pairs of people.
- Table:

R	Gene	Joan	William	Sue	Myrtle	Ormonde	Paula
Gene							
Joan	*			*			
William	*			*			
Sue							
Myrtle		*					
Ormonde		*					*
Paula							

- Graph:



- Which representation is better for testing whether the pair (x, y) is in the relation?
- Which representation is better for capturing the overall structure?

Reflexive Relations

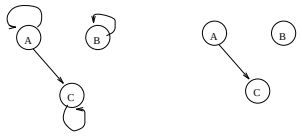
- A relation R on A is **reflexive** if a is related to a (aRa) for every $a \in A$.
- A relation R on A is **irreflexive** if a is not related to any $a \in A$.
- **(Ir)Reflexivity and Table representation:**

R	1	2	3
1	*	?	?
2	?	*	?
3	?	?	*

R	1	2	3
1		?	?
2	?		?
3	?	?	

The difference is what happens on the main diagonal of the matrix.

- **(Ir)Reflexivity and Graph representation:**



- **Examples:**

  Is the **parent** relation reflexive, irreflexive, or neither?

  Is the $\geq$ relation reflexive, irreflexive, or neither?

  Is the **boss-of** relation reflexive, irreflexive, or neither?

Symmetric Relations

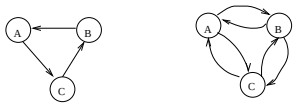
- A relation R on A is **symmetric** if for all $a, b \in A$, whenever $(a, b) \in R$ then $(b, a) \in R$.
- A relation R on A is **antisymmetric** if for all $a, b \in A$, if $(a, b) \in R$ and $(b, a) \in R$, then $a = b$.
- **(Anti)Symmetry and Table representation:**

R	1	2	3
1	?	*	*
2	*	?	*
3	*	*	?

R	1	2	3
1	?		*
2	*	?	
3		*	?

In a symmetric relation, the matrix is symmetric around the main diagonal.

- **(Anti)Symmetry and Graph representation:**



In an antisymmetric relation, the only symmetric entries are one the diagonal, i.e. the only back arcs are self-loops.

- **Examples:**

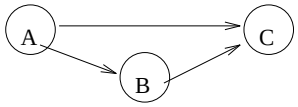
  Is the **cousin-of** relation symmetric, antisymmetric, or neither?

  Is the **brother-of** relation symmetric, antisymmetric, or neither?

  Is the $\geq$ relation symmetric, antisymmetric, or neither?

Transitive Relations

- A relation R is **transitive** if for all a, b, c, if aRb and bRc, then aRc
- **Example:**

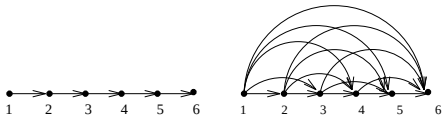


- If there is a path from a to c in a transitive relation, there must be a single arc from a to c.
- **Examples:**

  Is the **ancestor-of** relation transitive or not?

  Is the **friend-of** relation transitive or not?

  Is the $\geq$ relation transitive or not?

- The **transitive closure** of a relation R adds all the arcs to R necessary to make it transitive.
- **Examples:**
  - 

  - **Ancestor** is the transitive closure of **parent**.

Special and Equivalence Relations

- The **universal relation** $U = A \times A$.
- The **empty relation** $R = \emptyset$.
- The **identity relation** $I = \{(a, a) \text{ for } a \in A\}$
-

property	Universal	Empty	Identity
reflexive	yes	no	yes
irreflexive	no	yes	no
symmetric	yes	yes	yes
antisymmetric	no	yes	yes
transitive	yes	yes	yes

- $R \subset A \times A$ is an **equivalence relation** if it is reflexive, symmetric, and transitive.

  Example: The universal and identity relations are both equivalence relations.
- Equivalence relations partition the elements into **equivalence classes** of identically-behaving elements.

General (n-ary) Relations

- Suppose A_1, A_2, . . . , A_n are sets. A relation of A_1, A_2, . . . , A_n is a set $R \subseteq A_1 \times A_2 \times \ldots \times A_n$.
- Thus R is a set of ordered n-tuples $(a_1, a_2, \ldots, a_n)$ where $a_i \in A_i$.
- **Example:**

  $A_1 = N$, $A_2 = names$, $A_3 = majors$

  "Student number x is named y and majors in z"

  $(124324443, Mary, CSE), (563565426, Mary, PSY)$ are tuples of the relation.
- Such structures are modeled by **hypergraphs**, a graph structure where each "edge" represents a subset of more than two vertices.

Relational Databases

- The most important commercial database systems today employ the **relational** model, meaning that the data is stored as tables of tuples, i.e. relations.
- A Shakespearian **killed** relation would be:

Killer	Victim
Brutus	Caesar
Hamlet	Laertes
Hamlet	Polonius
Laertes	Hamlet
Brutus	Brutus
Cassius	Caesar

- **Requests** for information from the database is made in a query language like **SQL** which is based on the notations of set theory and the predicate calculus.
- **Example 1: Who killed Caesar?**

  In SQL:

  SELECT Killer from Killed where victim='Caesar'

  This reads "select from relation 'killed' all tuples where the victim was Caesar, and report only the killer field from each.

- **Example 2: Who was both a killer and a victim?**

  In SQL:

  (SELECT Killer from Killed) INTERSECT (SELECT Victim from Killed)

- Much of the power of relational databases comes from the fact that we can **combine different relations**.

- For example, suppose we also have a **died-by** relation:

Victim	Method
Caesar	Daggers
Hamlet	Sword
Laertes	Sword
Polonius	Sword
Brutus	Sword

  We can combine the two tables with a **join** operation, which the tables based on **common fields**. For example, the join of **killed** and **died-by** is:

Killer	Victim	Method
Brutus	Caesar	Daggers
Hamlet	Laertes	Sword
Hamlet	Polonius	Sword
Laertes	Hamlet	Sword
Brutus	Brutus	Sword
Cassius	Caesar	Daggers

- **Example 3: Which killers used daggers?**

  In SQL:

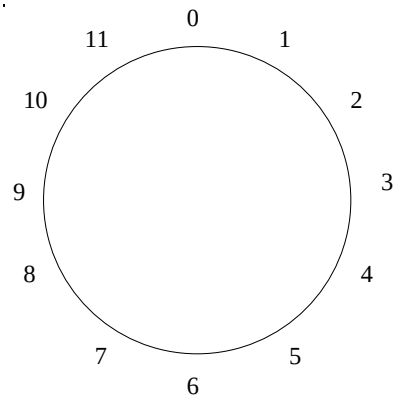
  SELECT Killer FROM Killed, Died_by WHERE Killed.victim=died_by.victim AND Method='Daggers'

- Note that this database design assumes that each victim can only be killed by one weapon (sorry, Rasputin).

Modular Arithmetic

- An important example of an equivalence relation is grouping integers into classes based on their remainder mod k.

  Think of grouping integers by counting around a clock.



- Note that $23 = 11 (\text{mod } 12)$. If you start at 0 and count to 23 you end up at 11.

  This is because 11 and 23 have the same remainder when divided by 12. 11 and 23 are in the same equivalence class.

- Note that if $m = n (\text{mod } k)$ and $x = y (\text{mod } k)$ then $m + x = n + y (\text{mod } k)$ – think about tracing out a path on the clock to prove it.

- Consider the relation $x = y (\text{mod} 12)$.

- This relation is reflexive, symmetric, and transitive, and hence is an equivalence relation.

- We have 12 equivalence classes.

- What an equivalence relation **means** is that the elements can be partitioned into equivalence classes (or blocks) such that all elements in the same block have the same properties.

  $\{0, 12, 24, 36, \ldots\}$, $\{1, 13, 25, 37, \ldots\}$, ...

Total and Partial Orders

- Many sets of objects, such as the integers, have a natural **total order (relation)** defined on them.

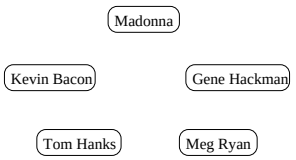
  For each pair of x and y, either (1) $x > y$, (2) $x < y$, or (3) $x = y$.

- Such total orders can be very useful.

  For example, we can sort any subset of totally ordered objects uniquely, e.g. $(1, 5, 12, 14, 23)$.

  The 'next' element of any finite, totally ordered set is completely defined.

- However, not all sets of objects have naturally defined total orders on them. Consider the set of entertainers ordered according to who is the bigger star:



  Note the confusion because different types are stars are **incomparable**, which is not the same as equals. Several different orders are consistent with this data.

Partial Orders

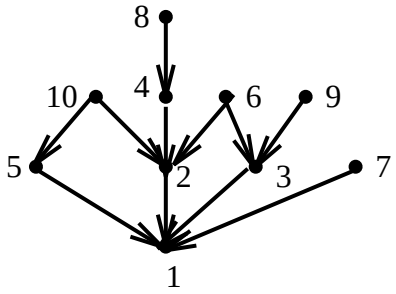
- $R \subset A \times A$ is a **partial order** if it is reflexive, antisymmetric, and transitive.

- An example of partial orders is "less-than".

- Partial orders are represented by **Hasse Diagrams**. Transitivity and reflexivity edges are not represented on Hasse Diagrams.

- **Example:**

  $R = \{(x, y) \mid x, y \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}\ and\ y|x\}$



- Because R is antisymmetric, reflexive and transitive, there can be no cycle in R. Thus there can be no inconsistency in the order (such as Redford-Bogart-Newman-Redford).

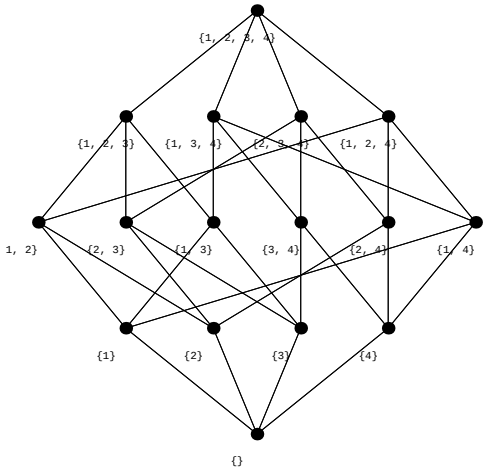
- A partial order on n objects can have only one consistent total ordering, or as many as $n!$ total orderings, depending upon the pairs in the relation.

Subsets by Inclusion

- For any set A, the relation $\subseteq$ on the power set of A defines a partial order.

- Representation by a Hasse Diagram:

  $A = \{1, 2, 3, 4\}$



- The structure of this subset inclusion relation defines the graph known as a **hypercube**.

  Note the recursive structure of a hypercube; a d-dimensional cube is composed by connecting two $(d-1)$-dimensional cubes.