CS2209A 2017
Applied Logic for Computer Science

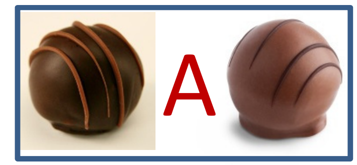
Lecture 17

Relations, first-order formula and database application

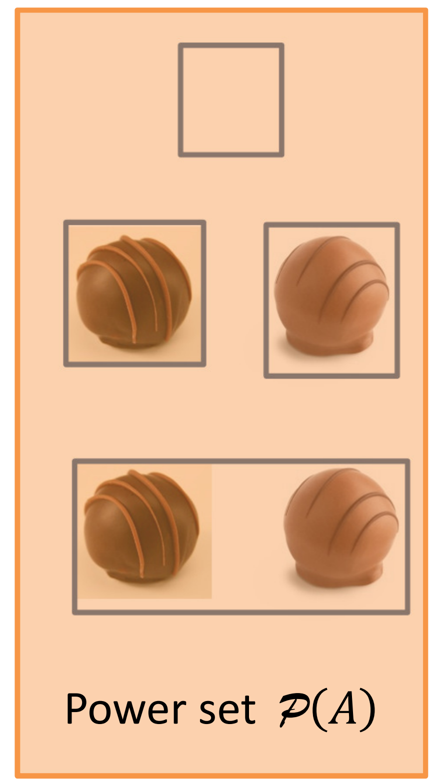
Instructor: Marc Moreno Maza

Power sets

- A **power set** of a set A, $\mathcal{P}(A)$, is a set of **all subsets** of A.
  - Think of sets as boxes of elements.
  - A subset of a set A is a box with elements of A (maybe all, maybe none, maybe some).
  - Then $\mathcal{P}(A)$ is a box containing boxes with elements of A.
  - When you open the box $\mathcal{P}(A)$, you don't see chocolates (elements of A), you see boxes.
  - A={1,2}, $\mathcal{P}(A) = \{\emptyset, \{1\}, \{2\}, \{1,2\}\}$
  - $A = \emptyset$, $\mathcal{P}(A) = \{\emptyset\}$.
    - They are not the same! There is nothing in A, and there is one element, an empty box, in $\mathcal{P}(A)$
- If A has n elements, then $\mathcal{P}(A)$ has 2^n elements.



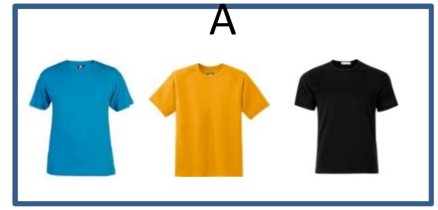
Subsets of A:

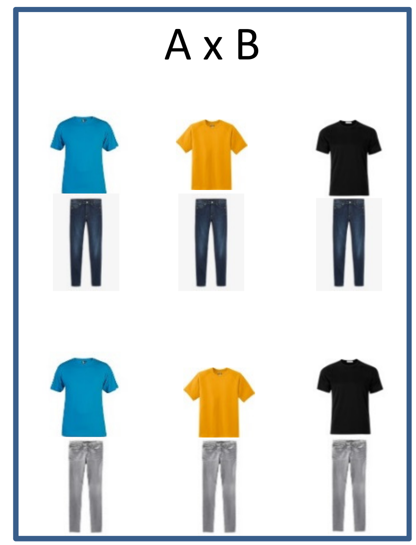


Cartesian products

- **Cartesian product** of A and B is a set of all pairs of elements with the first from A, and the second from B:
  - A x B = $\{(x, y) \mid x \in A,\ y \in B\}$
  - A={1,2,3}, B={a,b}
  - $A \times B =$ $\{(1, a), (1, b), (2, a), (2, b), (3, a), (3, b)\}$
  - A={1,2}, $A \times A = \{(1,1), (1,2), (2,1), (2,2)\}$

	a	b
1	(1,a)	(1,b)
2	(2,a)	(2,b)
3	(3,a)	(3,b)





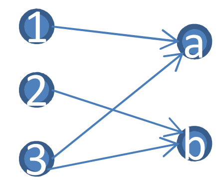
- Order of pairs does not matter, order within pairs does: $A \times B \neq B \times A$.
- Number of elements in $A \times B$ is $|A \times B| = |A| \cdot |B|$

Cartesian products

- We can define the Cartesian product for any number of sets:
  - $A_1 \times A_2 \times \cdots \times A_k = \{(x_1, x_2, \ldots x_k) | \, x_1 \in A_1 \ldots x_k \in A_k\}$
  - $A = \{1,2,3\}$, $B =$ {a,b}, C={3,4}
  - $A \times B \times C = \{(1, a, 3), (1, a, 4), \ (1, b, 3), (1, b, 4),$
    $(2, a, 3), (2, a, 4), \ (2, b, 3), (2, b, 4),$
    $(3, a, 3), (3, a, 4), \ (3, b, 3), (3, b, 4)\}$

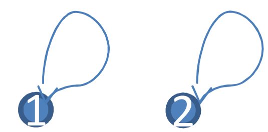
Relations

- **A relation** is a subset of a Cartesian product of sets.
  - If of two sets (set of pairs), call it a **binary** relation.
  - Of 3 sets (set of triples), **ternary**. Of k sets (set of tuples), **k-nary**
  - A={1,2,3}, B={a,b}
    - $A \times B = \{(1,a),(1,b),(2,a),(2,b),(3,a),(3,b)\}$
    - R = {(1,a), (2,b),(3,a), (3,b)} is a relation. So is R={(1,b)}.



Graph of R (bipartite)

  - A={1,2},
    - $A \times A = \{(1,1),(1,2),(2,1),(2,2)\}$
    - R={(1,1), (2,2)} (all pairs (x,y) where x=y)
    - R={(1,1),(1,2),(2,2)} (all pairs (x,y) where $x \leq y$).

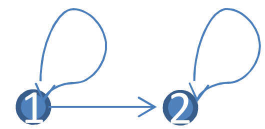


Graph of {(1,1),(2,2)}

  - A=PEOPLE
    - COUPLES ={(x,y) | Loves(x,y)}
    - PARENTS ={(x,y) | Parent(x,y)}
  - A=PEOPLE, B=DOGS, C=PLACES
    - WALKS = {(x,y,z) | x walks y in z}
      - Jane walks Buddy in spring bank park.

Types of binary relations

- A binary relation $R \subseteq A \times A$ is
  - **Reflexive** if $\forall x \in A, R(x,x)$



R_1

    - Every x is related to itself.
    - E.g. A={1,2}, $R_1 = \{ (1,1), (2,2), (1,2)\}$
    - On A = $\mathbb{Z}$, $R_2 = \{(x,y)|x = y\}$ is reflexive
    - But not $R_3 = \{(x,y)|\ x < y\}$
  - **Symmetric** if $\forall x, y \in A,\ (x,y) \in R \leftrightarrow (y,x) \in R$
    - R_1 and R_3 above are not symmetric. R_2 is.
    - A = $\mathbb{Z}$, $R_4 = \{(x,y)|x \equiv y \ mod\ 3 \}$ is symmetric.

Types of binary relations

- A binary relation $R \subseteq A \times A$ is
  - **Transitive**
    if $\forall x, y, z \in A,\ (x, y) \in R \wedge (y, z) \in R \rightarrow\ (x, z) \in R$
    - R_1, R_2, R_3, R_4 are all transitive.
    - $R_5 = \{(x, y) | x, y \in \mathbb{Z} \wedge x + 1 = y\}$ is not transitive.
    - PARENT = $\{(x, y) | x, y \in PEOPLE \wedge x\ \ is\ a\ parent\ of\ y\}$ is not.
    - A **transitive closure** of a relation R is a relation $R^* = \{(x, z)\ |\ \exists k \in \mathbb{N}\ \ \exists y_0, \ldots, y_k \in A\ \ (x = y_0 \wedge z = y_k) \wedge \forall i \in \{0, \ldots, k-1\}\ R(y_i, y_{i+1})\}$
      - That is, can get from x to z following R arrows.

Types of binary relations

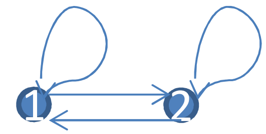
- A binary relation $R \subseteq A \times A$ is
  - **Anti-reflexive** if $\forall x \in A, \neg R(x,x)$
    - R can be neither reflexive nor anti-reflexive.
    - E.g. A={1,2}, $R_6 = \{(1,2)\}$
      - but not $R_1 = \{(1,1), (2,2), (1,2)\}$ (reflexive)
      - nor $R_7 = \{(1,1), (1,2)\}$ (neither)
    - For $A = \mathbb{Z}$, not $R_2 = \{(x,y) | x = y\}$
      - Nor $R_4 = \{(x,y) | x \equiv y \ mod\ 3\}$
    - But $R_3 = \{(x,y) |\ x < y\}$ is anti-reflexive.
      - So are $R_5 = \{(x,y) \in \mathbb{Z} \times \mathbb{Z} \mid x + 1 = y\}$
      - And PARENT = $\{(x,y) \in PEOPLE \times PEOPLE\ | x\ \ is\ a\ parent\ of\ y\}$

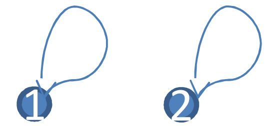
Graph of {(1,2)}

Types of binary relations

- A binary relation $R \subseteq A \times A$ is
  - **Anti-symmetric**
    if $\forall x, y \in A, (x, y) \in R \land (y, x) \in R \rightarrow x = y$
    - $R_1, R_3, R_5, R_6, R_7, PARENT$ are anti-symmetric. R_4 is not.
    - R_2 is both symmetric and anti-symmetric.
    - $R_8 = \{(1,2), (2,1), (1,3)\}$ is neither symmetric nor anti-symmetric.

Equivalence





- A binary relation $R \subseteq A \times A$ is an **equivalence** if R is reflexive, symmetric and transitive.
  - E.g. **A={1,2},** $R = \{(1,1),(2,2)\}$ or $R = A \times A$
  - Not $R_1 = \{ (1,1), (2,2), (1,2)\}$
    nor $R_3 = \{(x, y) |\ x < y\}$
  - On A = $\mathbb{Z}$, $R_2 = \{(x, y) | x = y\}$ is an equivalence
  - So is $R_4 = \{(x, y) | x \equiv y\ mod\ 3\ \}$
    - Reflexive: $\forall x \in \mathbb{Z},\ x \equiv x\ mod\ 3$
    - Symmetric: $\forall x, y \in \mathbb{Z},\ x \equiv y \bmod 3 \rightarrow y \equiv x\ \ mod\ 3$
    - Transitive: $\forall x, y, z \in \mathbb{Z},\ x \equiv y \bmod 3 \wedge y \equiv z\ \ mod\ 3 \rightarrow x \equiv z\ \ mod\ 3$

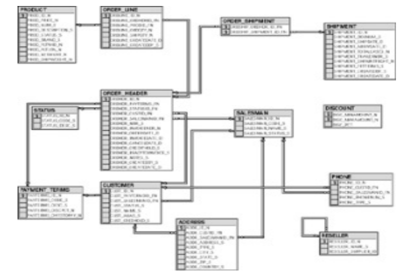
Equivalence

- An equivalence relation partitions A into **equivalence classes**:
  - Intersection of any two equivalence classes is ∅
  - Union of all equivalence classes is A.
  - $R_4 = \{(x, y) | x \equiv y \; mod \; 3 \}$:

$$\mathbb{Z} = \{x \mid x \equiv 0 \; mod \; 3\} \cup \{x \mid x \equiv 1 \; mod \; 3\} \cup \{x \mid x \equiv 2 \; mod \; 3\}$$

  - $R = A \times A$ gives rise to a single equivalence class.
  - $R = \{(1,1), (2,2)\}$ to two.

Databases and predicates



Relation R	Predicate P
A set of tuples	True/false on a given tuple
R=$\{(x_1, \ldots, x_k) \mid P(x_1, \ldots, x_k)$ is true$\}$	$P(x_1, \ldots, x_k) \equiv (x_1, \ldots, x_k) \in R$

– In a database, store relations as tables.

– Then ask queries as predicate logic formulas

- Return the set of all database elements satisfying the formula.

Well-Formed Formula for First Order Predicate Logic

- Not all strings can represent propositions of the predicate logic. Those which produce a proposition when their symbols are interpreted must follow the rules given below, and they are called **wffs** (well-formed formulas) of the first order predicate logic.

- A predicate name followed by a list of variables such as $P(x, y)$, where P is a predicate name, and x and y are variables, is called an **atomic formula**.

Well-Formed Formula for First Order Predicate Logic

- **wffs are constructed using the following rules:**
  - *True* and *False* are wffs.
  - Each propositional constant (i.e. specific proposition), and each propositional variable (i.e. a variable representing propositions) are wffs.
  - Each atomic formula (i.e. a specific predicate with variables) is a wff.
  - If A, B, C are wffs, then so are $\neg A$, $A \wedge B$, $A \vee B$, $A \rightarrow B$
  - If x is a variable (representing objects of the universe of discourse), and A is a wff, then so is $\exists x\ A$ and $\forall x\ A$.

Relational database and query

Example: three tables, *student*, *course*, and *enroll*

COURSE	
CNO	TITLE
CSc2310	Java
CSc2510	Theory
CSc4710	Database

STUDENT	
SID	NAME
1111	Jones
2222	Smith
3333	Blake
4444	Damon
5555	Bell

ENROLL		
SID	CNO	GRADE
1111	CSc2310	F
1111	CSc2510	F
2222	CSc2510	A
2222	CSc4710	B
3333	CSc2310	A
3333	CSc2510	A
3333	CSc4710	F

- A query using predicate logic is of the form { X1,...,Xn | P(X1,...,Xn) } , where P(X1,...,Xn) is a **wff** in predicate logic with free variables X1,...,Xn.

Relational database and query

COURSE	
CNO	TITLE
CSc2310	Java
CSc2510	Theory
CSc4710	Database

STUDENT	
SID	NAME
1111	Jones
2222	Smith
3333	Blake
4444	Damon
5555	Bell

ENROLL		
SID	CNO	GRADE
1111	CSc2310	F
1111	CSc2510	F
2222	CSc2510	A
2222	CSc4710	B
3333	CSc2310	A
3333	CSc2510	A
3333	CSc4710	F

- **Now, some query examples:**
  - Get names of students enrolled in CSc2510.
    { N | (exists S,G)(student(S,N) and enroll(S,'CSc2510',G)) }
  - Get CNO and TITLE of courses in which Smith is enrolled.
    { C,T | (exists S,G)(course(C,T) and enroll(S,C,G) and student(S,'Smith')) }
  - Get names of students who have enrolled in at least 2 courses.
    { N | (exists S,C1,G1,C2,G2)(student(S,N) and enroll(S,C1,G1) and enroll(S,C2,G2) and C1 <> C2) }
  - Get names of students who have an "A" grade in CSc2510.
    { N | (exists S)(student(S,N) and enroll(S,'CSc2510','A')) }

Limitations of first-order logic

- With predicate logic, can we express everything?
- The fact is that some very natural properties cannot be expressed in predicate logic, because here the notion of “*expressing*" is quite different.
- A **propositional formula** only talks about **finitely many** things, and if worst comes to worst it can just list all the cases, describing a truth table.
- Whereas in the **predicate logic** case we have the ability to talk about **infinitely many** things at once.
- And it is not possible to list all infinitely many cases of relationships among, say, natural numbers, at least not with a finite-length formula.

Limitations of first-order logic

- Here is an example of a very natural property which is not *expressible* in first-order logic.
  - **Transitivity in databases**: Consider an airline database such as the one that travel agents use to find and book flights.

- Now ask the database to "*give me a way to fly from London Ontario to London UK, and I don't care how many times I need to change planes*".

- This kind of query, called "**transitivity**", is not expressible in first-order logic.

Limitations of first-order logic

- A very natural property which is not *expressible* in first-order logic.
  - **Transitivity in databases**: Consider an airline database such as the one that travel agents use to find and book flights.
- For any fixed number of plane changes you can, indeed, ask if there is a way to get from London ON to London UK, however you always need to set a limit on the number of intermediate locations.
- For example, to ask if there is a way to get to London UK via four intermediate cities you would write
  - $\exists y_1, y_2, y_3, y_4 \ Flight(London\ ON, y_1) \wedge Flight(y_1, y_2) \wedge Flight(y_2, y_3) \wedge Flight(y_3, y_4) \wedge Flight(y_4, London\ UK)$

Transitivity in databases

- Some databases do have special ways of computing this kind of relation, but it is an addition to the language of first-order logic and can be quite computationally intensive, especially in large databases.

- More often, it seems, the database system just tries the first several queries, until they reach a number of intermediate steps that seems too large for them

Transitivity in databases

- You might have noticed that you pretty much never see a route with more than three intermediate cities, and sometimes get an answer “there is no way“

- You can also ask a combination of such queries: for example, “Is there a direct flight from London ON to London UK, or a flight with one intermediate city”?
  - $\exists y\ Flight(London\ ON, London\ UK) \vee Flight(London\ ON, y) \wedge\ Flight(y, London\ UK)$