Unit 5
Basic structures: relations and strings

Computer Science 1002

Introduction to Logic for Computer Scientists

1

Sets and strings

- How do we represent sets on a computer?
  - We can list names of a set's elements, but that would make operations with sets less efficient: even to check if an element is in a set, we'd need to scan through the whole list.
- Instead, when the universe is finite (and reasonably small), we can represent a set by its *characteristic string,* stating whether each element of the universe is in the set or not.
  - but first, let us recall and define what is a string.

2

Alphabet

- A finite set of symbols is called an **alphabet**.
  - English alphabet contains 26 letters (ignoring case): {a, .., z}
  - {0,1} is the *binary alphabet*. This is the most popular alphabet in computer science.
  - An alphabet containing only one letter, for example {a}, is called a $unary$ alphabet.

3

Strings

- A **string** (over an alphabet A) is a sequence (list) of symbols from A
  - Usually with repetition; order matters.
    - We will talk about 1st, 2nd, 3rd and so on symbol in a given string.
  - "mun" is a string over English alphabet (lowercase). 2nd symbol in "mun" is "u".
    - English words are strings over English alphabet.
  - "1002" is a string over the alphabet of digits {0,1,2}.
  - A natural number is a string over the alphabet of digits {0,1,2,3,4,5,6,7,8,9}

4

Finite strings, binary strings

- A *finite string* of *length* n is a string containing n symbols
  - The length of a string s is denoted |s|
    - The same notation as cardinality of a set (number of elements in a set): |A|
  - |mun|=3. |1002|=4.
  - There is an empty (null) string with the length 0
    - Some books, including our textbook, denote the empty string by a Greek letter "lambda" λ, others by the Greek letter "epsilon" ϵ.
- A *binary string* is a finite string over the alphabet {0,1}
  - 0011011, 1111, λ, 0 are binary strings
    - with lengths 7,4,0,1 respectively.

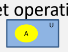
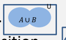
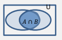
5

Characteristic string of a set

- Let U be a universe. List its elements in some order
  - It does not matter which order, as long as it is fixed and you know which element is 1st, which is 2nd and so on.
  - For example, in $U = \{a, b, c, d, e\}$ we can arbitrarily call a the first element, b second, c third, d fourth and e fifth.
- A **characteristic string** of a set $A \subseteq U$ is a binary string s of length |U| which for every position i in s has a 1 iff the i^{th} element of U is in A, and 0 iff the i^{th} element of U is not in A.
  - Let U={a,b,c,d,e}, in this order. Then
    - the characteristic string of a set A={a,c,d} is 10110, since 1st, 3rd and 4th elements of U are in A.
    - The characteristic string of $\emptyset$ is 00000. The characteristic string of U is 11111.

6

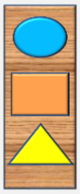
Set operations with characteristic strings

- Characteristic strings make it easy to do set operations
  - Complement: flip all 0s to 1s, and all 1s to 0s. 
  - Union: put 1 if at least one of the strings has 1 in that position. 
  - Intersection: put 0 if at least one of the strings has 0 in that position. 

- Let U={a,b,c,d,e}, A={a,c,d}, B= {a,b}.
  - Then the characteristic string for A is 10110, for B is 11000.
  - The characteristic string of $\bar{A}$ is 01001, corresponding to {b,e}
  - The characteristic string of $A \cup B$ is 11110, corresponding to set {a,b,c,d}
  - The characteristic string of $A \cap B$ is 10000, encoding the set {a}.

7

Sets and strings are different types!



- Though strings can be used to represent sets
  - with respect to a given finite universe and a given order of elements in it

strings and sets are different types!
  - The main operation on a set is to check whether a given element is in a set: $a \in S$.
    - There are no duplicates, and no intrinsic order of elements.
  - The main operation on a string is to see what symbol is in a specific position i.
- A string is a special type of a sequence, where at every position the sequence is an element of the alphabet.

8

9

Pairs, triples, tuples

- For sets S_1, S_2 define (ordered) **pairs** of elements (x_1, x_2) as a sequence of length 2 where the first element $x_1 \in S_1$ and second element $x_2 \in S_2$
  - Notation (x_1, x_2), with round brackets () indicates that it is an sequence, ordered with distinguishable first and second elements.
    - E.g. $(1,2)$ or $(5,5)$; here $(1,2) \neq (2,1)$
  - as opposed to curly brackets for sets: $\{1,2\} = \{2,1\}$
    - So {1,2} and (1,2) mean different things!

10

Pairs, triples, tuples

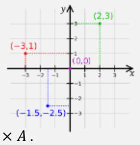
- Can generalize pairs to ordered sequences of any (fixed) length: triples, quadruples, generally **tuples** of elements from sets $S_1, S_2, \dots S_n$
  - A tuple with n elements is called an n-tuple. A triple is a 3-tuple.
    - (2,3,2,1,3) is a 5-tuple, where all 5 positions are elements of {1,2,3}.
- Tuples are similar to strings in that they are both ordered sequences.
  - In strings, elements come from a (fairly small) alphabet, each is commonly one symbol, and they are written one after another without spaces or brackets.
    - We often look at a bunch of strings of different length together.
  - In tuples, as in general sequences, elements are listed in round brackets () and separated by commas.
    - We often look at tuples of specific length together (eg a bunch of pairs).
  - In both cases, we often ask what element is in the position i.

11

Cartesian products

- **Cartesian product** of A and B is a set of all (ordered) pairs of elements with the first element from A, and the second element from B:
  - A x B = $\{(x,y) | x \in A \wedge y \in B\}$

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

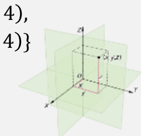


  - $A = \{1,2,3\},\ B = \{a,b\}$
  - $A \times B = \{(1,a),(1,b),(2,a),(2,b),(3,a),(3,b)\}$
  - $B \times B = \{(a,a),(a,b),(b,a),(b,b)\}$
- Order of pairs does not matter, order within pairs does: $A \times B \neq B \times A$.
  - The name "Cartesian" is the same as in Cartesian coordinate system: every point is described by a pair of numbers in 2d (triple of numbers in 3d).
- Number of elements in $A \times B$ is $|A \times B| = |A| \cdot |B|$
  - Here, |A| is the **cardinality** of A (number of elements in A)

12

Cartesian products

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



13

14

Relations

- A set of tuples of elements (that is, a subset of a Cartesian product) is called a **relation.** In particular, a set of pairs of elements is called a **binary relation**.
  - LESSTHAN = $\{(x, y)|\ x$ and y are real numbers such that $x < y\}$
    - So $(1,2) \in LESSTHAN$, but $(2,1) \notin LESSTHAN$
    - LESSTHAN $\subseteq \mathbb{R} \times \mathbb{R}$
  - LIKES = $\{(x, y) \mid$ person x likes person $y\}$
  - PARENT =$\{(x, y) \mid$ person x is a parent of person $y\}$
  - REGISTRATIONS = $\{(name, cour, sem)|$ student $name$ takes course $cour$ in semester $sem\}$
    - If Wei Lee takes COMP1002 in Fall 2020, then (Wei Lee, COMP1002, Fall 2020) $\in REGISTRATIONS$

15

Relational databases

PROFDATA = $\{(firstname, lastname, office)\}$ =

	A	B	C
1	Manrique	Mata-Montero	EN-2033
2	Sharene	Bungay	ER-6032
3	Antonina	Kolokolova	ER-6033

COURSEDATA = $\{(course, days, time, room, instructor)\}$ =

	A	B	C	D	E
1	COMP1000	MWF	11:00-11:50	EN-1054	Mata-Montero
2	COMP1001	MWF	12:00-12:50	EN-2040	Bungay
3	COMP1002	MTR	13:00-13:50	EN-2007	Kolokolova

In the language of databases, each tuple in a relation is called a **record**. Each position in the tuple called a **field**. And the relation itself is called a **table**.

- Records are rows in the table, and fields are columns.
- (Sharene, Bungay,ER-6032) is a record. "Office" is a field. PROFDATA is a table.
- Usually, a database consists of several tables.

16

Predicates vs. sets

- Predicates and sets are two sides of the same coin
  - For each set S there is a predicate which is true exactly on elements of S
  - For each predicate P there is a set S of values of x on which P is true.

Set S	Predicate P
A collection of elements	Becomes true/false on a given element
$S_P = \{x \mid P(x)$ is true$\}$	$P_S(x) \equiv$ "$x \in S$"

17

Predicates vs relations

Relation R	Predicate P
A set (collection) of pairs (generally, n-tuples) of elements	True/false on a given pair (tuple) of elements
$R_P = \{(x, y) \mid P(x, y)$ is true$\}$ $LESSTHAN =$ $\{(x, y) \mid (x \in \mathbb{R}) \wedge (y \in \mathbb{R}) \wedge (x < y)\}$ $= \{(x, y) \mid LessThan(x, y)$ is true $\}$ $(3,5) \in LESSTHAN$ $(3,2) \notin LESSTHAN$	$P_R(x, y) \equiv$ "$(x, y) \in R$" $LessThan(x, y) \equiv$ $(x \in \mathbb{R}) \wedge (y \in \mathbb{R}) \wedge (x < y)$ $\equiv (x, y) \in LESSTHAN$ $LessThan(3,5)$ is true $LessThan(3,2)$ is false

18

19

Database queries

- Predicate logic gives us a way to query relational databases!
  - Using predicates for database relations.
    - Then build a query as a formula.
  - Input of the query:
    - Can be nothing (if no free variables)
    - Or specific values for its free variables.
  - Output of the query:
    - Either true/false
    - Or a set of elements (values of its free variables) satisfying the query.

20

Database queries

PROFDATA $= \{(firstname, lastname, office)\} =$

	A	B	C
1	Manrique	Mata-Montero	EN-2033
2	Sharene	Bungay	ER-6032
3	Antonina	Kolokolova	ER-6033

COURSEDATA $= \{(course, days, time, room, instructor)\} =$

	A	B	C	D	E
1	COMP1000	MWF	11:00-11:50	EN-1054	Mata-Montero
2	COMP1001	MWF	12:00-12:50	EN-2040	Bungay
3	COMP1002	MTR	13:00-13:50	EN-2007	Kolokolova

How can we query such databases?

21

Querying databases

PROFDATA

	A	B	C
1	Manrique	Mata-Montero	EN-2033
2	Sharene	Bungay	ER-6032
3	Antonina	Kolokolova	ER-6033

COURSEDATA

	A	B	C	D	E
1	COMP1000	MWF	11:00-11:50	EN-1054	Mata-Montero
2	COMP1001	MWF	12:00-12:50	EN-2040	Bungay
3	COMP1002	MTR	13:00-13:50	EN-2007	Kolokolova

- Suppose we want to find out if someone called Manrique is teaching COMP1001.
  - First, define predicates corresponding to the relations:
    - $ProfData(firstname, lastname, office)$
    - $CourseData(course, days, time, room, instructor)$
  - Now, let's write a formula using these predicates which would be true if and only if Manrique is teaching COMP1001.

22

PROFDATA

	A	B	C
1	Manrique	Mata-Montero	EN-2033
2	Sharene	Bungay	ER-6032
3	Antonina	Kolokolova	ER-6033

COURSEDATA

	A	B	C	D	E
1	COMP1000	MWF	11:00-11:50	EN-1054	Mata-Montero
2	COMP1001	MWF	12:00-12:50	EN-2040	Bungay
3	COMP1002	MTR	13:00-13:50	EN-2007	Kolokolova

- We want to find out if Manrique is teaching COMP1001.
  - Ask: is there somebody with the same last name teaching COMP1001?
    - Try $\exists\, ln\ \ ProfData(Manrique, ln,) \land \mathrm{CourseData(COMP1001, ln)}$
      - Error! ProfData takes 3 inputs, and CourseData takes 5!
    - Try $\exists\, ln\ \ ProfData(Manrique, ln, ?) \land \mathrm{CourseData(COMP1001, ?, ?, ?, ln)}$
      - Almost there, but the database does not like our question marks…
      - How do we fill them if we don't know their values? Use ∃ again

$$\exists\, \boldsymbol{ln}\ \exists \boldsymbol{o}\ \exists \boldsymbol{d}\ \exists \boldsymbol{t}\ \exists \boldsymbol{r}\ \boldsymbol{ProfData(Manrique, ln, o)} \land \mathbf{CourseData(COMP1001, d, t, r, ln)}$$

- In SQL, a popular database language,
  - Putting a condition which records to select (such as ProfData(Manrique,…)) is called a *selection*
  - the operation of the kind $\exists \mathrm{x}\ \mathrm{P(x)} \land Q(x)$ is called a *join*
  - Adding extra existential quantifiers for "don't care" fields is called a *projection*

23

PROFDATA

	A	B	C
1	Manrique	Mata-Montero	EN-2033
2	Sharene	Bungay	ER-6032
3	Antonina	Kolokolova	ER-6033

COURSEDATA

	A	B	C	D	E
1	COMP1000	MWF	11:00-11:50	EN-1054	Mata-Montero
2	COMP1001	MWF	12:00-12:50	EN-2040	Bungay
3	COMP1002	MTR	13:00-13:50	EN-2007	Kolokolova

- We want to find out if Manrique is teaching COMP1001.
  $\exists\, \boldsymbol{ln}\ \exists \boldsymbol{o}\ \exists \boldsymbol{d}\ \exists \boldsymbol{t}\ \exists \boldsymbol{r}\ \boldsymbol{ProfData(Manrique, ln, o)} \land \mathbf{CourseData(COMP1001, d, t, r, ln)}$

- Now let's make it more general: given person's name x and course name y, is person x teaching course y?
  $\exists\, ln\ \exists o\ \exists d\ \exists t\ \exists r\ ProfData(\boldsymbol{x}, ln, o) \land CourseData(\mathbf{y}, \mathrm{d, t, r}, ln)$
  - Here, x and y are free variables.

24

PROFDATA

	A	B	C
1	Manrique	Mata-Montero	EN-2033
2	Sharene	Bungay	ER-6032
3	Antonina	Kolokolova	ER-6033

COURSEDATA

	A	B	C	D	E
1	COMP1000	MWF	11:00-11:50	EN-1054	Mata-Montero
2	COMP1001	MWF	12:00-12:50	EN-2040	Bungay
3	COMP1002	MTR	13:00-13:50	EN-2007	Kolokolova

- Now let's make it more general: given person's name x and course name y, is person x teaching course y?

  $\exists\, ln\ \exists o\ \exists d\ \exists t\ \exists r\ ProfData(\mathbf{x}, ln, o) \wedge CourseData(\mathbf{y}, \mathrm{d}, \mathrm{t}, \mathrm{r}, ln)$
  - Here, x and y are free variables.
- Now let's write a query that check that everybody teaches something.

  $\forall x\ \exists y\ \exists\, ln\ \exists o\ \exists d\ \exists t\ \exists r\ ProfData(\mathbf{x}, ln, o) \wedge CourseData(\mathbf{y}, \mathrm{d}, \mathrm{t}, \mathrm{r}, ln)$
- Finally, let's return a relation containing all pairs (x, y) such that x teaches y
  - TEACHES = $\{ (x, y) \mid \exists\, ln\ \exists o\ \exists d\ \exists t\ \exists r\ ProfData(\mathbf{x}, ln, o) \wedge CourseData(\mathbf{y}, \mathrm{d}, \mathrm{t}, \mathrm{r}, ln) \}$

25

26

Building sets from sets (and other stuff).

- A bunch of tuples is a set called a *relation*.
  - A set of all possible tuples of elements of given sets is a Cartesian product
- A bunch of strings is just a set of strings.
  - The set of all binary strings has a special notation: $\{0,1\}^*$
    - Star stands for "repeat elements of the set {0,1} zero or more times".
  - In general, a set of all strings over an alphabet A is denoted A^* (A star).
    - You will see the reason for this notation in COMP 1003.
- What is a bunch of sets? And a set of all sets?

27

Power sets

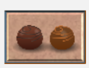
Set A

- A **power set** of a set A, denoted $\mathcal{P}(A)$, is a set elements of which are 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 some, maybe all, maybe none
  - Then $\mathcal{P}(A)$ is a box containing all possible boxes corresponding to subsets of A
  - When you open the box $\mathcal{P}(A)$, you don't see chocolates (elements of A), you see boxes.
  - A={1,2}, $\mathcal{P}(A) = \{\emptyset, \{1\}, \{2\}, \{1,2\}\}$
- If A has n elements, then $\mathcal{P}(A)$ has 2^n elements

Power set $\mathcal{P}(A)$

28

Power sets



Set A

- A **power set** of a set A, denoted $\mathcal{P}(A)$, is a set elements of which are all subsets of A.
  - 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, but there is one element, an empty box, in $\mathcal{P}(A)$
  - $A = \{ a, \{1, b\}\}$, $\mathcal{P}(A) = \{\emptyset, \{a\}, \{\{1, b\}\}, \{a, \{1, b\}\}\}$.
    - Here, A contains two elements, a and $\{1, b\}$. It is irrelevant for the power set that the second element is a set (a box), it treats it just like any other element.
    - That's why we need an extra pair of parentheses in the third element of $\mathcal{P}(A)$, $\{\{1, b\}\}$. This tells us that the 3rd element of $\mathcal{P}(A)$, is a set (box) containing a single element: $\{1, b\}$, just like $\{a\}$ is a set with a single element a.

Power set $\mathcal{P}(A)$

29

Type checking: power set and cartesian product

- Power set is a set, elements of which are themselves sets.
  - If A has n elements, then $\mathcal{P}(A)$ has 2^n elements, each of which is a set
  - Empty set $\emptyset$ is an element of $\mathcal{P}(A)$ for any A.
    - Because for every A, $\emptyset \subseteq A$
- Cartesian product of two sets is a set of pairs of elements
  - Of k sets, (ordered) k-tuples of elements.
  - Remember that (a, b) and $\{a, b\}$ mean very different things!
    - If a program expects a pair and you give it a set, you get an ERROR!
  - Power set: $\{\ \emptyset,\ \{\ \},\{\ \},\{\ \}\ \}$ ($2^{|A|}$ sets inside)
  - Cartesian product: $\{\ (,),\ (,),\ (,),\ (,)\ \}$ ($|A| \cdot |B|$ pairs inside)

30

31

Binary relations

- A special type of relation, with many interesting properties, is a binary relation $\subseteq A \times A$ for some set A
  - that is, a relation among elements of the same set, where every element can be in the 1st as well as 2nd position of a pair.
  - A={1,2},
    - $A \times A = \{(1,1), (1,2), (2,1), (2,2)\}$
    - R={(1,1), (2,2)} ={ (x,y) | $x, y \in \{1,2\} \wedge$ x=y}}
    - R={(1,1),(1,2),(2,2)} = (x,y) | $x, y \in \{1,2\} \wedge x \leq y\}$.
  - A=PEOPLE
    - COUPLES ={(x,y) | Loves(x,y)}
    - PARENTS ={(x,y) | Parent(x,y)}

32

Graphs of binary relations

- **A (directed) graph (digraph)** of a binary relation $R \subseteq A \times A$ is a diagram consisting of
  - |A| points, with a point (often drawn as a circle with a label, called a vertex or a node) for each element of A
  - An arrow (called an edge, an arc or a link) from point x to point y for each $(x, y) \in R$
    - We draw a loop with an arrow for each $x \in A$ such that $(x, x) \in R$
- Let A={1,2,3}

$R = \{(1,1), (1,2), (2,2), (3,1)\}$

$R = \{(1,2), (1,3), (3,1)\}$

$R = A \times A$

$R = \{(1,1)\}$

- This is a different notion of a graph than plotting a function on plane!

33

34

Reflexive relations

- A binary relation $R \subseteq A \times A$ is **reflexive** if $\forall x \in A, R(x, x)$
  - Every x is related to itself.
    - On a graph, every vertex has a loop
  - 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 $R_3 = \{(x, y) | \, x < y\}$

R_1

R_2

35

Anti-reflexive relations

- A binary relation $R \subseteq A \times A$ is **anti-reflexive** if $\forall x \in A, \neg R(x, x)$
  - Graph of R has no loops.
  - 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 $R_5 = \{(x, y) \in \mathbb{Z} \times \mathbb{Z} \,|\, x + 1 = y\}$
    - And PARENT = $\{(x, y) \in PEOPLE \times PEOPLE \,| x \;\; is \; a \; parent \; of \; y\}$
  - A relation R can be neither reflexive nor anti-reflexive.

R_6 ={(1,2)}

36

Symmetric relations

- A binary relation $R \subseteq A \times A$ is **symmetric** iff $\forall x, y \in A,\ (x,y) \in R \leftrightarrow (y,x) \in R$
  - For every arrow in a graph (except loops) another goes the opposite way
  - R_1 and R_3 from previous slides are not symmetric. R_2 is.
  - $R_8 = \{(1,2),(2,1),(1,1)\}$ is symmetric
  - A = $\mathbb{Z}$, $R_4 = \{(x,y) | x \equiv y \ mod\ 3\}$ is symmetric.

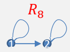

R_8

37

Anti-symmetric relations

- A binary relation $R \subseteq A \times A$ is **anti-symmetric** iff $\forall x, y \in A, (x,y) \in R \wedge (y,x) \in R \rightarrow x = y$
  - For every arrow, there is no arrow the other way. Loops OK.
  - $R_1, R_3, R_5, R_6, R_7, PARENT$ are anti-symmetric.
  - R_4 is not.
  - R_2 is both symmetric and anti-symmetric.
  - $R_9 = \{(1,2),(1,3),(3,1)\}$ is neither symmetric nor anti-symmetric.


R_8

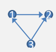

R_1


R_9

38

Transitive relations

- A binary relation $R \subseteq A \times A$ is **transitive** if $\forall x, y, z \in A,\ (x,y) \in R \wedge (y,z) \in R \rightarrow (x,z) \in R$
  - In the graph of R, if there is a way to get from x to z by following a sequence of edges (arrows), then there is a way to get from x to z in one step.
  - 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 = $\{(x,y) | x, y \in PEOPLE \wedge x \text{ is a parent of } y\}$ is not transitive either.


$R = \{(1,2),(3,1),(3,2)\}$


$R = \{(1,2),(3,1)\}$

39

40

- Sometimes a relation we have does not quite give us what we want:
  - A $PARENT$ relation is not reflexive, not transitive – we have to come up with all sorts of new formulas for grandparent/great-grandparent..
  - Maybe what we want is an *ANCESTOR* relation:
    - It is transitive and reflexive, behaves nicely…
    - Though it requires more space to store than $PARENT$…
- Is there a way to compute $ANCESTOR$ from $PARENT$ when we need it?
- And how to do this in general?

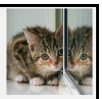
41

Extending relations to have desired properties

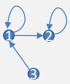
- Often, we start with a relation which is not reflexive
  - Or not symmetric, or not transitive
- But we'd really prefer if it were reflexive (symmetric, transitive)!
- In this case, we change our relation to get a relation with the property we want.
  - We always want to keep all the original edges: never lose information.
  - And we want to make as few changes as possible.
- If R is a relation, then its transitive, reflexive or symmetric *closure* is the smallest transitive, reflexive or symmetric relation, respectively, containing R.

42

A reflexive closure of a relation

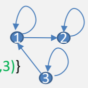


- Let $R \subseteq \{1,2,3\} \times \{1,2,3\}$, R={(1,1), (1,2), (2,2), (3,1)}
- To compute a reflexive closure of R, we just need to add all missing self-loops.
  - In this case, there is only one: (3,3)
  - This will give us the smallest possible reflexive relation containing R.
- So to compute a reflexive closure of a given relation, add all missing pairs (x, x), and nothing else.
- Another example: a reflexive closure of $R_3 = \{(x,y) | \; x < y\}$ is $\{(x,y) | \; x \leq y\}$



$R = \{(1,1), (1,2), (2,2), (3,1)\}$

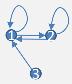
Reflexive closure of R:
{(1,1), (1,2), (2,2), (3,1), (3,3)}



43

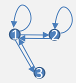
A symmetric closure of a relation

- Let $R \subseteq \{1,2,3\} \times \{1,2,3\}$, R={(1,1), (1,2), (2,1), (2,2), (3,1)}
- To compute a symmetric closure of R, check every pair $(x, y) \in R$ for which $x \neq y$. If $(y, x) \notin R$, then add (y, x) to the symmetric closure of R.
  - In this case, (1,2) and (2,1) are both in R, but $(1,3) \notin R$ even though $(3,1) \in R$, so we add (1,3) to the symmetric closure of R.
  - We don't need to bother with (2,3), (3,2) since neither is in R.
  - This will give us the smallest possible symmetric relation containing R.
- Example: a symmetric closure of $R_3 = \{(x,y) | \; x < y\}$ is $\{(x,y) | \; x \neq y\}$



$R = \{(1,1), (1,2), (2,1), (2,2), (3,1)\}$

Symmetric closure of R:
{(1,1), (1,2), (1,3), (2,1)(2,2), (3,1) }



44

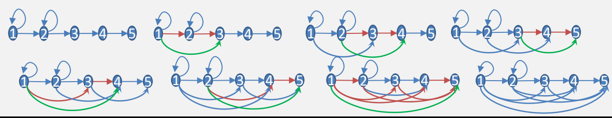
Transitive closure: routes from flights

- If you are trying to get to Boston from Gander, you would be more interested in a sequence of flights that would get you there than whether there is a direct flight.
- You need a *transitive closure* of the FLIGHT relation.

45

A transitive closure of a relation

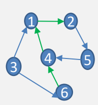
- A transitive closure of a relation R is the smallest transitive relation that contains R.
  - Add edge (x, z) whenever $(x, y) \in R, (y, z) \in R$, but $(x, z) \notin R$
  - Keep doing it again and again, until the resulting relation is transitive
    - Not enough to just go through edges once.
    - There are faster ways you will learn in your algorithms course.



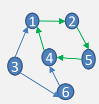
46

Paths in graphs

- A *path* in a graph G from a vertex x to a vertex y is a sequence $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$ of edges of G, where $v_0 = x$ and $v_k = y$.
  - If no vertex repeats, it is called a *simple path*.
  - Number of edges k is the *length* of the path.
  - If $x = y$, it is called a *cycle*.
- A pair (x, y), where $x \neq y$, is in the transitive closure of a relation R iff there is a path from x to y in the graph of R.
  - If (x, y) is in R, it is also in its transitive closure even if $x = y$.



Simple path from 6 to 2



Cycle 4 to 4

47

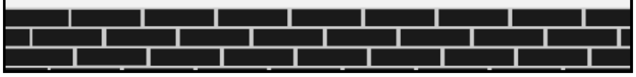
Combining closures

- To compute a closure that is both, for example, reflexive and transitive, compute one first and then the other of the result.
  - The order does not matter.
- For example, let $R = \{(x,y) | \; x + 1 = y\}$, for $x, y \in \mathbb{Z}$.
  - Transitive closure of R is $\{(x,y) | \; x < y\}$
  - Reflexive closure of R is $\{(x,y) | x + 1 = y \lor x = y\}$
  - Symmetric closure of R is $\{(x,y) | x + 1 = y \lor y + 1 = x\}$
  - Reflexive transitive closure of R is R is $\{(x,y) | \; x \leq y\}$
  - Symmetric transitive closure of R is $\{(x,y) | \; x \neq y\}$
  - Reflexive, symmetric, transitive closure of R is $\mathbb{Z} \times \mathbb{Z}$

48

Transitive closure and limitations of predicate logic

- One reason why computing transitive closures is so useful for databases is that it lets us circumvent a limitation of predicate logic:
  - We can write a formula saying that there is a path from x to y of length k
    - $Grandparent(x, y)$ says that there is a path of length 2 in the graph of PARENT
  - However, we need to use $k - 1$ new variables for a path of length k.
- There is no way to write a finite-length formula of predicate (first-order) logic saying that there is some path, of any length, from x to y.
  - Without going to second-order logic, which is beyond the scope of this course.

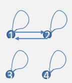


49

50

Equivalence relation

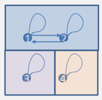
- A binary relation $R \subseteq A \times A$ is an **equivalence** if R is reflexive, symmetric and transitive.
  - E.g. A={1,2}, $R = \{(1,1), (2,2)\}$ or $R = A \times A$
  - Not R_1 = { (1,1), (2,2), (1,2)} nor $R_3 = \{(x, y) | x < y\}$
  - On A = $\mathbb{Z}$, $R_2 = \{(x, y) | x = y\}$ is an equivalence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



51

Equivalence classes

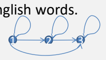
- An equivalence relation partitions A into **equivalence classes**:
  - Intersection of any two equivalence classes is $\emptyset$
  - Union of all equivalence classes is A.
  - R_4: $\mathbb{Z} = \{x \,| x \equiv 0 \; mod \; 3\} \cup$ {x | x ≡ 1 mod 3} $\cup$ $\{x \,| x \equiv 2 \; mod \; 3\}$
  - $R = A \times A$ gives rise to a single equivalence class. $R = \{(1,1), (2,2)\}$ on A ={1,2} to two equivalence classes.

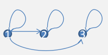


52

Order

- A binary relation $R \subseteq A \times A$ is an **order** if R is reflexive, anti-symmetric and transitive.
  - E.g. $R_1 = \{(x, y) | x, y \in \mathbb{Z} \wedge x \leq y\}$. So is alphabetical order of English words.
  - But not $R_2 = \{(x, y) | x, y \in \mathbb{Z} \wedge x < y\}$
    - not reflexive, so not an order.
  - $SUBSETS = \{(A, B) \mid A, B \; are \; sets \wedge \; A \subseteq B\}$ is an order.
    - Reflexive: $\forall A, \; A \subseteq A$
    - Anti-symmetric: $\forall \, A, B \;\; A \subseteq B \wedge B \subseteq A \rightarrow A = B$
    - Transitive: $\forall A, B, C \;\; A \subseteq B \wedge B \subseteq C \rightarrow A \subseteq C$
  - $DIVISORS = \{(\mathrm{x,y}) \mid x, y \in \mathbb{N} \wedge x, y \geq 2 \; \wedge \exists z \in \mathbb{N} \;\; y = z \cdot x\}$ is an order.
  - PARENT is not an order. But ANCESTOR would be, if defined so that each person is an ancestor of themselves.

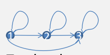




53

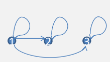
Total and partial order

- R is a **total order** if $\forall x, y \in A \;\; R(x, y) \vee R(y, x)$ (any two elements of A are related.)
  - $R_1 = \{(x, y) | x, y \in \mathbb{Z} \wedge x \leq y\}$ is a total order
  - So is alphabetical order of English words.



Total order

- Otherwise, R is a **partial order**.
  - $SUBSETS = \{(A, B) \mid A, B \; are \; sets \wedge \; A \subseteq B\}$ is a partial order.
    - Not total: if A ={1,2} and B ={1,3}, then neither $A \subseteq B$ nor $B \subseteq A$
  - $DIVISORS$ is a partial order.
    - 5 and 7 do not divide each other.
  - ANCESTOR is a partial order
    - Two people don't necessarily have to be related.

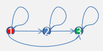


Partial order

54

Minimal and maximal

- An order may have **minimal** and **maximal** elements (maybe multiple)
  - $x \in A$ is minimal in R if $\forall y \in A \ \ y \neq x \rightarrow \ \neg R(y, x)$
    - and maximal if $\forall y \in A \ y \neq x \rightarrow \neg R(x, y)$
  - If there is only one minimal element, we call it a minimum
    - Unique maximal element is a maximum.
  - In R_{10}, 1 is a minimum, 3 is a maximum
  - In R_{11}, 1 is a minimum, and both 2 and 3 are maximal
  - $\emptyset$ is minimal in SUBSETS (its unique minimum); universe is maximal (its unique maximum).
  - All primes are minimal in DIVISORS, and there are no maximal elements.



R_{10}

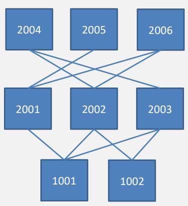
R_{11}

55

56

Drawing orders: diagrams

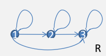
- A graph of an order relation has a lot of edges! It has all loops, and every time $(a, b) \in R$ and $(b, c) \in R$, we have $(a, c) \in R$
- When we know that a relation is an order, can we draw it simpler?
  - Can you figure out in which order you can take courses from this diagram?



57

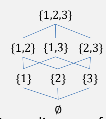
Hasse diagram

- A Hasse diagram is a way to draw a (partial or total) order without drawing loops or edges that have to be there by transitivity or reflexivity.
  - draw minimal elements on the bottom, then go up
  - don't draw arrowheads (assume arrows go upwards).
  - R={ $(x, y) \in \{1,2,3\} \times \{1,2,3\} | \ x \leq y\}$
    - On the Hasse diagram of R, only draw edges (1,2) and (2,3), as all the rest follow by reflexivity and transitivity. 1 is the minimal (bottom), 3 maximal (top).



R

Hasse diagram of R



Hasse diagram of SUBSETS on {1,2,3}

58

Puzzle: coins

- A not-too-far-away country recently got rid of a penny coin, and now everything needs to be rounded to the nearest multiple of 5 cents…
  - Suppose that instead of just dropping the penny, they would introduce a 3 cent coin.
    - Like British three pence.
  - What is the largest amount that cannot be paid by using only existing coins (5, 10, 25) and a 3c coin?

59