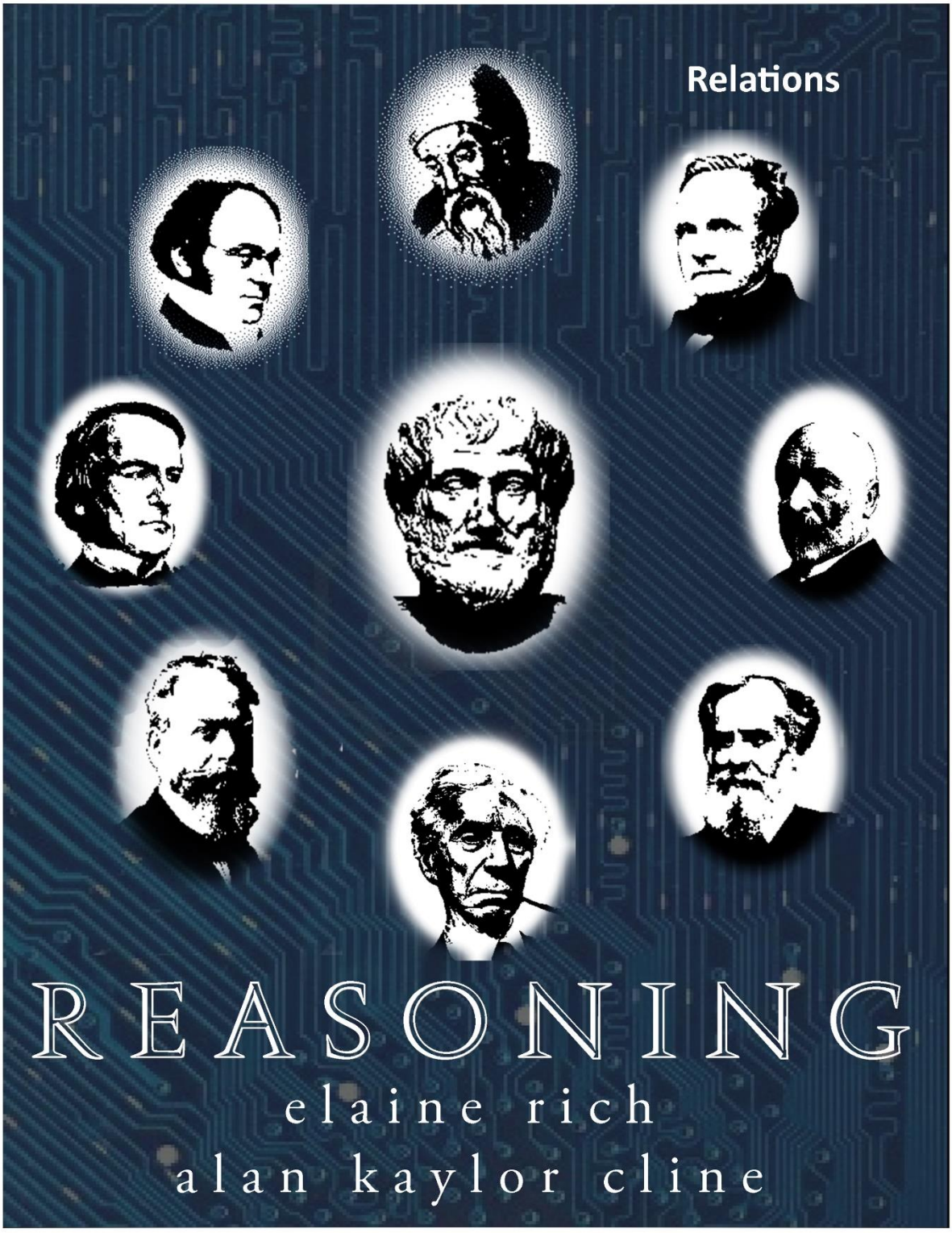

Relations
REASONING
elaine rich
alan kaylor cline

The Logicians on our cover are:

Euclid (? - ?)

Augustus De Morgan (1806 – 1871)

Charles Babbage (1791 – 1871)

George Boole (1815 – 1864)

Aristotle (384 BCE – 322 BCE)

George Cantor (1845 – 1918)

Gottlob Frege (1848 – 1925)

John Venn (1834 – 1923)

Bertand Russell (1872 – 1970)

REASONING

AN INTRODUCTION TO LOGIC, SETS, AND FUNCTIONS

CHAPTER 10

RELATIONS

Elaine Rich
Alan Kaylor Cline

The University of Texas at Austin

Image credits:

Swan and cygnets: http://www.telegraph.co.uk/earth/earthpicturegalleries/5989838/Animal-pictures-of-the-week-7-August-2009.html?image=1

Molecule: http://office.microsoft.com/en-us/images/results.aspx?qu=molecule&ex=1&origin=HA010237914#ai:MC900436918

Computer network: http://office.microsoft.com/en-us/images/results.aspx?qu=molecule&ex=1&origin=HA010237914#ai:MC900436918

City (Glasgow) road map: http://glasgowtransport.co.uk/gct_oneway_map.jpg

Web icons: http://icons.webtoolhub.com/icon-n70185f178026-detail.aspx

Cake: http://www.arthursclipart.org/foods/food/cake%203.gif

Pie: http://www.arthursclipart.org/foods/food/icecream%20coe%202.gif

Ice cream: http://4vector.com/i/free-vector-cherry-pie-clip-art_113043_Cherry_Pie_clip_art_hight.png

Clock: http://www.crateandbarrel.com/decorating-and-accessories/clocks/1

http://www.cs.utexas.edu/learnlogic
Library of Congress Cataloging-in-Publication Data

Rich, Elaine, 1950 -
Reasoning—An Introduction to Logic Sets and Functions / Elaine Rich.— 1st ed. p. cm.
ISBN x-xxx-xxxxx-x 1

Table of Contents

Ordered Tuples and Cartesian Products

Relating Elements of Sets

At this point, we've seen examples of many kinds of *sets*. And we've worked with operations on those sets. What we'd like to do now is to define some tools for reasoning about *elements* and their relationships to each other.

Consider the following pairs of elements:

(Paris, France), (Paris, Texas), (London, England), (Rome, Italy)

In each pair, the first element is drawn from the set *Cities*. The second is drawn from the set *Countries*. And the named city is in the country indicated by the second element.

Consider the following pairs of elements, all of which are drawn from **N** (the natural numbers):

(1, 2), (2, 4), (3, 6), (100, 200)

In each pair, the second element is twice the first.

Consider the following triples of elements, all of which are drawn from **Z** (the integers):

(2, 3, 5), (-5, 8, 3), (90, 16, 106), (-8, -2, -10), (9, 0, 9)

In each triple, the third element is the sum of the first two.

Ordered Pairs

An ***ordered pair*** is a sequence of two objects. Given any two objects, x and y, there are two ordered pairs that can be formed. We write them as (x, y) and (y, x). As the name implies, in an ordered pair (as opposed to in a set), order matters (unless x and y happen to be equal).

All of the following are ordered pairs:

(Paris, France), (Paris, Texas), (London, England), (Rome, Italy)

(1, 2), (2, 4), (3, 6), (100, 200)

(cat, kitten), (dog, puppy), (bat, pup), (swan, cygnet), (armadillo, pup)

We can put any two arbitrary elements together to form an ordered pair. But we rarely do that. Generally we'll use ordered pairs to capture useful connections. What relationships exist between the pairs of elements in the example above?

Two ordered pairs (a, b) and (x, y) are equal if and only $a = x$ and $b = y$.

(cat, kitten) ≠ (cat, kitty) (cat, kitten) ≠ (kitten, cat)

Problems

1. Indicate whether each of the following statements is true or false:

a) (Paris, France) = (France, Paris)
b) (17, 2) = (17, 2)
c) (2, 2) is an ordered pair.
d) (Paris, Italy) is an ordered pair.

Ordered N-Tuples

Sometimes we want to relate more than two elements. An ***ordered n-tuple*** is an ordered sequence of *n* objects. When *n* is 3, we'll say that we have an ***ordered triple***.

Consider the following set of equations:

$$\begin{aligned} 2x + 3y - z &= 9 \\ x - y + z &= 0 \\ 3x - y + 2z &= 4 \end{aligned}$$

We can say that the ordered triple (1, 3, 2) is a solution to this system of equations if we interpret it as containing values for the variables x, y, and z, in that order.

As for ordered pairs, two ordered tuples $(a_1, a_2, \ldots, a_k)$ and $(b_1, b_2, \ldots, b_n)$ are equal if and only if:

$k = n$ and, for every i, where $1 \le i \le n$, $a_i = b_i$

For example, $(1, 3, 2) \neq (1, 2, 3) \neq (1, 3, 2, 5)$

Problems

1. Indicate whether each of the following statements is true or false:

a) (Paris, Paris, France) = (Paris, France, Paris)
b) (1, 2, 2) = (1, 2)
c) (2, 2, 2) is an ordered triple.
d) (Paris, Rome, London, Athens) is an ordered 4-tuple

Cartesian Products

The ***Cartesian product*** of two sets A and B (written $A \times B$) is the set of all ordered pairs (a, b) such that $a \in A$ and $b \in B$.

For example, let A be a set of people: {Dave, Sara, Billy}, and let B be a set of desserts: {cake, pie, ice cream}. Then:

$A \times B$ = { (Dave, cake), (Dave, pie), (Dave, ice cream),
(Sara, cake), (Sara, pie), (Sara, ice cream),
(Billy, cake), (Billy, pie), (Billy, ice cream)}.

As you can see from this example, the Cartesian product of two sets contains elements that represent all the ways of pairing some element from the first set with some element from the second. Note that $A \times B$ is not the same as $B \times A$.

Continuing with our example:

$B \times A$ = { (cake, Dave), (pie, Dave), (ice cream, Dave),
(cake, Sara), (pie, Sara), (ice cream, Sara),
(cake, Billy), (pie, Billy), (ice cream, Billy)}.

Thus $A \times B \neq B \times A$.

In the special case in which A and B are the same set, we can write (by analogy with $x \cdot x = x^2$):

$$A \times A = A^2$$

If A and B are finite, then the cardinality of their Cartesian product is given by:

$$|A \times B| = |A| \cdot |B|.$$

This is straightforward to prove by induction. Try to do so.

Again, let A = {Dave, Sara, Billy}, and B = {cake, pie, ice cream}. Then:

$|A \times B| = |A| \cdot |B| = 3 \cdot 3 = 9$

Nifty Aside

The name "Cartesian product" is derived from the name of the important French philosopher and mathematician, René Descartes (1596 -1650). Descartes may be best known as the author of the phrase, "I think, therefore I am." But his other contributions both to philosophy and to mathematics were substantial.

When we take a Cartesian product, it's not necessary for the two sets to be different.

Let A = {rock, paper, scissors}. Then:

$$A \times A = \{ \begin{array}{l} \text{(rock, rock), (rock, paper), (rock, scissors),} \\ \text{(paper, rock), (paper, paper), (paper, scissors),} \\ \text{(scissors, rock), (scissors, paper), (scissors, scissors)}\} \end{array}$$

In this example, $A \times A$ is the set of possible outcomes of one round if two players play the rock, paper, scissors game.

Problems

1. We want to prove, by counterexample, that, for arbitrary sets A and B, it need not be true that $A \times B = B \times A$. Let $A = \{2, 4, 8, 16\}$. Let $B = \{2, 4, 6, 8, 10, 12\}$. Consider each of the following ordered pairs. If it proves our claim (by being an element of one of $A \times B$ or $B \times A$, but not both), mark true. Otherwise, mark false.

a) (2, 4)
b) (2, 6)
c) (8, 8)
d) (12, 4)
e) (12, 14)

2. Let $A = \{2, 4, 8, 16\}$. Let $B = \{2, 4, 6, 8, 10, 12\}$.

(Part 1) What is $|A \times B|$?

(Part 2) Insert 10 into B. Now what is $|A \times B|$?

(Part 3) Insert 10 into A. Now what is $|A \times B|$?

3. Let *Presidents* be the set of U.S. Presidents. Recall that $\mathbf{Z}^+$ is the set of positive integers. For each of the following ordered pairs, mark true if it is in $\mathbf{Z}^+ \times$ *Presidents*. Mark false otherwise.

a) (1, Washington)
b) (12, Lincoln)
c) (Garfield, 20)
d) (Taft, Wilson)
e) (35, Washington)

4. Suppose that we have three sets, A, B and C and we want to say that (peanuts, popcorn) is an element of both $A \times B$ and $A \times C$. At least one of the following statements makes that claim. The rest are nonsense (meaning that they're not either true or false; they simply aren't syntactically correct). For each statement, mark true if it makes the intended claim. Mark false otherwise.

(peanuts, popcorn) $\in A \times (B \wedge C)$
((peanuts, popcorn) $\in A \times B$) $\wedge$ ((peanuts, popcorn) $\in A \times C$))
(peanuts, popcorn) $\in (A \times B) \wedge (A \times C)$
(peanuts, popcorn) $\subseteq (A \times B) \wedge (A \times C)$

5. For any set A, what is $|A \times A|$? You should be able to prove your answer before you write it.

a) $|A|$
b) $2 \cdot |A|$
c) $|A|^2$
d) $|A|^2 - |A|$
e) $|A|^2 + |A|$

Generalizing to More Sets

We can take the Cartesian product of any finite number of sets.

For example, let: *Products* = {screws, nails, hooks}
Suppliers = {WidgetCorp, CoolBiz}
PriceRanges = {Low, Middle, High}

Then *Products* × *Suppliers* × *PriceRanges* contains elements such as:

(screws, CoolBiz, High) and (nails, WidgetCorp, Low)

Define the Cartesian product of sets $A_1, A_2, \ldots A_n$, written:

$$A_1 \times A_2 \times \ \ldots \ \times A_n$$

to be:

$$\{(e_1, e_2, \ldots e_n) : \text{for all } 1 \le k \le n,\ e_k \in A_k\}$$

If $A_1, A_2, \ldots A_n$ are finite, then the cardinality of their Cartesian product is given by:

$$| A_1 \times A_2 \times \ \ldots \ \times A_n | = | A_1| \cdot | A_2| \cdot \ldots | A_n| \cdot$$

Problems

1. Let: A = {apple, artichoke, almond}
B = {bacon, boxty}
C = {caramel, custard, coconut, carrot}
D = {duck, donut, dumpling, date}

Indicate whether each of the following statements is true or false:

a) (apple, bacon, caramel) $\in A \times B \times C$
b) (artichoke, boxty, coconut) $\in (A \times B) \times C$
c) ((almond, bacon), (custard, dumpling)) = (almond, bacon, custard, dumpling)
d) ((artichoke,boxty), (carrot,donut)) $\in (A \times B) \times (C \times D)$

2. Let: A = {apple, artichoke, almond}
B = {bacon, boxty}
C = {caramel, custard, coconut, carrot}
D = {duck, donut, dumpling, date}

(Part 1) What is $|A \times B \times C|$?
(Part 2) What is the cardinality of $|B \times B \times D|$?

Relations

Binary Relations

When we take the Cartesian product of two sets A and B, we get *all* ordered pairs that can be formed by taking one element of A and one element of B.

Recall the dessert choice example: Let A be a set of people: {Dave, Sara, Billy}, and let B be a set of desserts: {cake, pie, ice cream}. Then:

$A \times B$ = { (Dave, cake), (Dave, pie), (Dave, ice cream),
(Sara, cake), (Sara, pie), (Sara, ice cream),
(Billy, cake), (Billy, pie), (Billy, ice cream)}.

But suppose that we want to consider only those pairs that correspond to some situation we are trying to describe.

For example, suppose that we want to talk about desserts that people actually like. Perhaps Billy and Sara hate pie and Dave hates cake. So we want to write:

Likes = { (Dave, pie), (Dave, ice cream),
(Sara, cake), (Sara, ice cream),
(Billy, cake), (Billy, ice cream)}

A ***binary relation*** over two sets A and B is a *subset* of $A \times B$. In this case, we'll say that we have a relation *from* the set A *to* the set B.

Likes (as described above) is a binary relation.

When we have a relation R from A to B, we'll call A the ***domain*** of R and B its ***codomain***.

Likes has {Dave, Sara, Billy} as its domain and {cake, pie, ice cream} as its codomain.

Sometimes the two sets involved are the same (call it A). In this special case, we'll say that we have a relation *on* the set A. (In other words, a relation from A to A is a relation "on A".)

MotherOf is a relation on the set *People*. So *MotherOf* $\subseteq$ *People* $\times$ *People*. Suppose that Chris is the mother of Kelly. Then:

(Chris, Kelly) $\in$ *MotherOf*

Note that we don't require that a relation from A to B be a *proper* subset of $A \times B$.

Maybe Dave, Sara and Billy simply love food. Then we could have:

LovesItAll = $A \times B$ = { (Dave, cake), (Dave, pie), (Dave, ice cream),
(Sara, cake), (Sara, pie), (Sara, ice cream),
(Billy, cake), (Billy, pie), (Billy, ice cream)}

In this case, we'll say that the relation is a ***total relation*** from *A* to *B*.

Common binary relations include:

- = (equality, defined on many domains),
- < (defined on numbers and some other domains), and
- ≤ (also defined on numbers and some other domains).

The relation < on the integers contains an infinite number of elements drawn from the Cartesian product of the set of integers with itself. For instance, 2 < 7.

Let *R* be any binary relation and let (*a*, *b*) be some element of *R*. Four common ways to indicate this are:

$(a, b) \in R$	
Rab	Common in books, but only useful when *R*, *a*, and *b* can all be written as single characters. We won't use this one.
R(*a*, *b*)	Note the similarity to predicate notation. Think of *R*(*a*, *b*) as being true if and only if $(a, b) \in R$.
a R b	Often used for common relations to which we assign special symbols. This is called ***infix notation*** (because the relation symbol occurs *in* between the two arguments).

Let *R* be < (less than) defined on the integers. Then we can write all of the following:

- $(1, 6) \in <$
- $< (3, 327)$
- $4 < 82$

Let *R* be the *Likes* relation defined above. Then we can write:

- (Dave, pie) ∈ *Likes*
- *Likes*(Billy, cake)
- Sara *Likes* pie

Notice a couple of important properties of relations as we have defined them. First, a relation may be equal to the empty set (in which case, we can call it an ***empty relation***).

If Dave, Sue, and Billy all hate dessert, then the *Likes* relation would be {} or ∅.

Second, there are no constraints on how many times a particular element may occur in a relation.

In the *Likes* example, pie occurs once, cake occurs twice, and ice cream occurs three times.

Problems

1. Assume the standard definition of the relation ≤ defined on the natural numbers. In other words, $(i, j) \in \leq$ if and only if $i \leq j$. Mark each of the following statements as true or false:

a) $6 \leq 8$
b) $12 \leq 12$
c) $\leq (-9, 6)$
d) $(7, 5) \in \leq$

2. Let *Cities* be the set of cities in the world. (We'll skip over the details of how large something has to be to qualify as a "city". Also, we'll assume, for simplicity, and counter to fact, that a name uniquely identifies a city.) Let *Countries* be the set of countries in the world. Define:

$$Places \subseteq Countries \times Cities = \{(a, b) : b \text{ is in } a\}$$

Mark each of the following statements as true or false:

a) France is in the domain of *Places*
b) France is in the codomain of *Places*
c) (France, Paris) ∈ *Places*
d) (France, Rouen) ∈ *Places*
e) (Rome, Italy) ∈ *Places*
f) *Places*(Spain, Madrid)

3. Let *Cities* be the set of cities in the world. (We'll skip over the details of how large something has to be to qualify as a "city". Also, we'll assume, for simplicity, and counter to fact, that a name uniquely identifies a city.) Let *Countries* be the set of countries in the world. Define:

$$Capitals \subseteq Countries \times Cities = \{(a, b) : b \text{ is the capital of } a\}$$

Mark each of the following statements as true or false:

a) *Capitals*(France, Paris)
b) (France, Rouen) ∈ *Capitals*
c) *Capitals*(Turkey, Istanbul)
d) *Capitals*(Turkey, Ankara)

4. Let *Presidents* be the set of U.S. Presidents. Recall that $\mathbf{Z}^+$ is the set of positive integers. We've already seen that $\mathbf{Z}^+ \times$ *Presidents* includes, among other things, (1, Washington), (12, Lincoln) and (35, Washington). But now let's define the relation:

$$NthPresident \subseteq \mathbf{Z}^+ \times Presidents = \{(n, p) : p \text{ was the } n^{th} \text{ president of the USA}\}$$

For each of the following ordered pairs, mark true if it is in *NthPresident*. Mark false otherwise.

a) (1, Washington)
b) (12, Lincoln)
c) (Garfield, 20)
d) (Taft, Wilson)
e) (35, Washington)

n-ary Relations

Not all relations are binary. We define an ***n-ary relation*** over sets A_1, A_2, … A_n to be a subset of $A_1 \times A_2 \times \ldots \times A_n$. The n sets may be different, or they may be the same.

In the special case in which $n = 3$, we'll say that we have a ***ternary relation***.

Recall the parts and suppliers example:

Products = {screws, nails, hooks}
Suppliers = {WidgetCorp, CoolBiz}
PriceRanges = {Low, Middle, High}

Products × *Suppliers* × *PriceRanges* contains all ordered triples of the form:

(product, supplier, price range)

But now consider just those triples that correspond to our business reality. We can define a ternary relation *CanBuy*, which contains only those triples that describe suppliers and what they sell. So, for example, perhaps it's the case that:

(screws, CoolBiz, Low) ∈ *CanBuy*, but:
(hooks, WidgetCorp, Low) ∉ *CanBuy*

Nifty Aside

Many of the databases that run our modern world are implemented as ***relational databases***: the facts that they contain are represented as relations like *CanBuy*.

Problems

1. Define the ternary relation *Plus* on **Z** (the integers) as:

$$\{(i, j, k) : k = i + j\}$$

Mark each of the following statements as true or false:

a) $(4, 5, 20) \in Plus$
b) $(7, 8, 15) \in Plus$

2. Assume the existence of the following sets:

firsts:	First names
lasts:	Last names
eids:	strings of up to 8 characters, corresponding to electronic student ids
gpas:	numbers between 0 and 4

Assume that each student has exactly one eid and no two students share an eid.

Suppose that we want to define a relation *StudentData* that will associate, with each eid, a first name, a last name, and a gpa. Indicate which of the following is/are true of *StudentData*:

a) *StudentData* is a 4-ary relation.
b) *StudentData* is a relation on the set *eids*.
c) $|StudentData| \leq |eids|$.
d) (hhht000, Casey, Studentman, 3.78) could be an element of *StudentData*
e) (hhht000, 4.00, 2.65, 3.78) could be an element of *StudentData*.

The Inverse of a Binary Relation

The ***inverse*** of a binary relation R, written R^{-1}, is simply the set of ordered pairs in R with the elements of each pair reversed. Formally, if $R \subseteq A \times B$, then:

$$R^{-1} \subseteq B \times A = \{(b, a): (a, b) \in R\}.$$

If a relation is a way of associating with each element of A with a corresponding element of B, then think of its inverse as a way of associating with elements of B their corresponding elements in A. Note that the domain of R^{-1} is the codomain of R, and vice versa.

Recall the relation:

Likes = { (Dave, pie), (Dave, ice cream),
(Sara, cake), (Sara, ice cream),
(Billy, cake), (Billy, ice cream)}

Likes associates, with each person, their preferred desserts.

Likes^{-1} = *Fans-Of* = { (pie, Dave),
(cake, Sara), (cake, Billy),
(ice cream, Dave), (ice cream, Sara), (ice cream, Billy)}

Likes^{-1} (perhaps more helpfully called *Fans-Of*), associates, with each dessert, the people who like it.

Every binary relation has an inverse.

We've just seen that *Fans-Of* is the inverse of *Likes*.

The inverse of < (in the usual sense, defined on numbers) is >.

It is important to keep in mind the definition of an inverse. It is not either complement or negation.

Our company's supplier/customer database contains the relation:

BuysFrom $\subseteq$ *Companies* $\times$ *Companies* = $\{(c_1, c_2): c_1$ buys from $c_2\}$.

Suppose that *BuysFrom* contains, among other elements:

(MegaCorp, SuperBuy), (MegaCorp, BuyCheap), (SuperBuy, MegaCorp)

Then *BuysFrom*$^{-1}$ (which we might want to call *SellsTo*) contains, among other elements:

(SuperBuy, MegaCorp), (BuyCheap, MegaCorp), (MegaCorp, SuperBuy)

Notice that (SuperBuy, MegaCorp) is an element of both *BuysFrom* and *BuysFrom*$^{-1}$. This means that the companies both buy and sell from/to each other.

The inverse of R^{-1} is R. Said another way, $(R^{-1})^{-1} = R$. This follows immediately from the definition of R^{-1}. By the way, notice that this isn't the first time you've seen this phenomenon.

Let n be a number. Then $-(-n) = n$. For example, $- -3 = 3$.

Let n be a non-zero number. Then $\frac{1}{\frac{1}{n}} = n$.

Problems

1. Consider the relation ≤ on **Z** (the integers). Mark each of the following true if it is in ≤, false otherwise:

(4, 8) (7, 2) (5, 5)

2. Mark each of the following true if it is in the *inverse* of ≤ (on **Z**), false otherwise:

(4, 8) (7, 2) (5, 5)

3. Which of the following symbols is commonly used to denote the inverse of ≤ (on **Z**):

< > ≤ ≥ =

4. Mark each of the following true if it is in both ≤ and the inverse of ≤ (on **Z**), false otherwise:

(2, 9) (9, 2) (8, 8) (8, -8)

5. Consider the relation *Speaks*, a relation from *People* to *Languages*, defined as:

Speaks = {(*a*, *b*) : *a* speaks *b*}

So we might have, for example, the following pairs in *Speaks*:

(Pierre, French), (Ramon, Spanish), (Megumi, Japanese), (Lola, French), (Ramon, English), (Ava, Farsi)

Then $Speaks^{-1}$ (which we might want to call *Speakers*) is a relation from *Languages* to *People*, defined as:

$Speaks^{-1}$ = *Speakers* = {(*b*, *a*) : *b* is spoken by *a*}

For each of the following ordered pairs, mark true if it must be in *Speakers* (given what we know) and false otherwise:

(Pierre, French) (French, Lola) (Spanish, Arturo) (Japanese, Megumi)

6. The Human Relations department of our company maintains a database that might contain (among other things) this relation:

$Works\text{-}in \subseteq EmployeeIDs \times Departments =$
{(e, d) : employee with id number e works in department d}

Suppose that it contains the following ordered pairs (among others):

(32415, TechSupport), (45672, Finance), (56421, Purchasing), (65532, Design)
(71005, Marketing), (87821, TechSupport), (88100, Finance), (89991, Finance)

Then $Works\text{-}in^{-1}$ (which we might want to call *EmployeesOf*) is a relation from *Departments* to *EmployeeIDs*, defined as:

$Works\text{-}in^{-1} = EmployeesOf$ = {(d, e) : employee with id number e works in department d }

Write out the elements that we know are in *EmployeesOf*. Compare this set to the one that contains elements of *Works-in*. Notice something interesting.

(Part 1) First consider *Works-in*. Suppose that I give you half of an ordered pair:

(id, _____)

For some given value of id, what's the maximum number of values that might be able to fill in the blank (given the data that you've seen)?

(Part 2) Now consider *EmployeesOf*. Suppose that I give you half of an ordered pair:

(department, _____)

Now for some given value of department, what's the maximum number of values that might be able to fill in the blank (again, given the data that you've seen)?

7. Recall that *Works-in* contains (among other things):

(32415, TechSupport), (45672, Finance), (56421, Purchasing), (65532, Design)
(71005, Marketing), (87821, TechSupport), (88100, Finance), (89991, Finance)

We have just seen that, for every employee, there is a unique department. But it's not true that for every department there's a unique employee. (That would be a very small company indeed.)

We call a relation like *Works-in* $\subseteq A \times B$, which associates with every element of *A* a unique element of *B*, a ***function***. We'll say a lot more about functions soon. For now, we just observe that if we start out with a function (such as *Works-in*) and take its inverse, we don't necessarily get a function.

Just to experiment with this idea, suppose that we want $Works\text{-}in^{-1}$ to be a function. And we're willing to fire people to make that happen. For each of the following sets of ordered pairs, mark true if removing all of its elements from *Works-in* would result in $Works\text{-}in^{-1}$ = *EmployeesOf* being a function. Mark false otherwise:

a) {(32415, TechSupport), (87821, TechSupport)}
b) {(45672, Finance), (56421, Purchasing), (88100, Finance), (89991, Finance)}
c) {(32415, TechSupport), (45672, Finance), (88100, Finance), (89991, Finance)}
d) {(56421, Purchasing), (32415, TechSupport), (45672, Finance), (65532, Design), (71005, Marketing), (87821, TechSupport), (89991, Finance)}

Composing Binary Relations

If we have two or more binary relations, we may be able combine them in a particularly useful way.

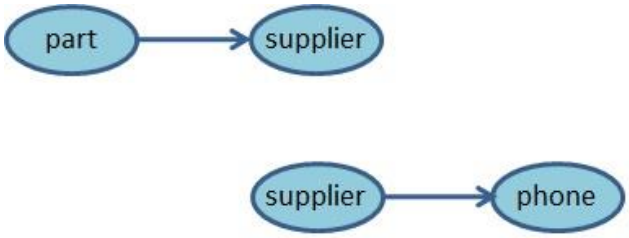
Our company maintains a database of parts we buy and suppliers from whom we can buy them. This database contains (among other things) the following two relations:

$Sources \subseteq Parts \times Suppliers = \{(p, s) : p \text{ can be bought from } s\}$

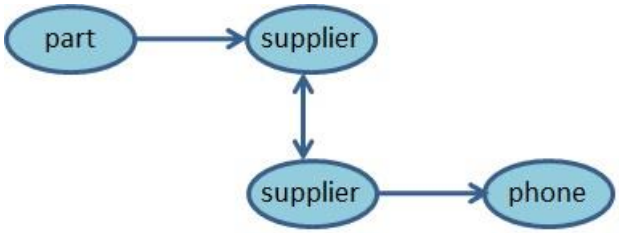
$Contacts \subseteq Suppliers \times PhoneNumbers = \{(s, n)\} : \text{the phone number of } s \text{ is } n\}$

Now suppose that we want to order a part. We need to find the phone number of some supplier who sells it.

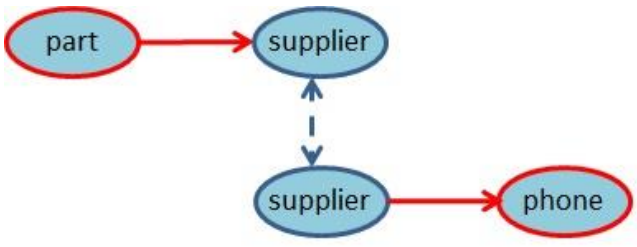
We can visualize the two relations that we've got as:



What we need to do is to connect them by linking the two ovals that correspond to suppliers:



Finally, we need to connect parts with phone numbers by reasoning that if $part_i$ is related to $supplier_j$ and $supplier_j$ is related to $phonenumber_k$, then $part_i$ is related to $phonenumber_k$. In other words, we need an operation that does this:



You might ask, at this point, why don't we just start with a single, ternary relation:

$$HowToOrderParts \subseteq Parts \times Suppliers \times PhoneNumbers.$$

Such a relation would contain elements such as (Nails, ZCorp, 555-333-4444). Then we'd be done.

But this would be a bad idea for a couple of reasons, both of which are tied to our desire to represent each piece of information exactly once. To see this, consider the following tables that contain fragments of our two original relations *Sources* and *Contacts*. (Read each row of one of these tables as corresponding to an ordered pair that is in the corresponding relation.)

SourcesContacts

Part	***Supplier***
Brushes	PaintLand
Buckets	ToolLand
Drill Bits	ToolLand
Drill Bits	ZCorp
Filler	ZCorp
Knobs	ZCorp
Nails	ZCorp
Scrapers	PaintLand
Screws	ZCorp

Contacts

Supplier	***Phone Number***
PaintLand	777-888-9999
ToolLand	666-111-2222
ZCorp	555-333-4444

If we just combined these two relations and stored *HowToOrderParts*, we'd represent many times the fact that ZCorp's contact phone number is 555-333-4444.

Definition of Composition

Define the ***composition*** of two relations $R_1 \subseteq A \times B$ and $R_2 \subseteq B \times C$, written $R_2 \circ R_1$, as:

$$R_2 \circ R_1 = \{(a, c) : \exists b\ ((a, b) \in R_1 \wedge ((b, c) \in R_2)\}.$$

In other words, (a, c) is in $R_2 \circ R_1$ if and only if there's some b that connects them as described above.

Note that this definition tells us that, if we want to apply R_1 (for example *Sources*) first, and then R_2 (for example *Contacts*), as shown in the pictures above, we must write $R_2 \circ R_1$.

In other words, we read a composition expression right to left. (Imagine that you were reading Hebrew or Arabic.) It might seem a bit more natural to write it the other way. But for consistency reasons that will become clear later, once we've introduced functions and their composition, this is the way it is done.

Not all expressions of the form $R_2 \circ R_1$ are well-defined. $R_2 \circ R_1$ makes sense only when the codomain of R_1 is a subset of the domain of R_2.

For example *Contacts* ∘ *Sources* makes sense. (The codomain of *Sources* is *Suppliers*. The domain of *Contacts* is also *Suppliers*.) But *Sources* ∘ *Contacts* doesn't make sense. If we apply *Contacts* first, we get phone numbers. We can't apply *Sources* to phone numbers.

By composing relations as we need them, we can solve the multiple representation problem that we noticed above. And this is important. Representing the same fact more than once is bad for at least two reasons:

- We'd waste a lot of space. Perhaps you're thinking that space doesn't matter much in modern computing systems. Think about the size of an everyday flash drive. But it turns out that it still matters if you are working on the scales that routinely confront Facebook and Google and Twitter and many others.

- We'd have to do a lot of work to keep our data up to date. For example, if ZCorp moves or if we get assigned a different sales rep, we'd have to change the phone number. If we must do that in many places, we increase the chance that we'll make mistakes.

And the situation would be even worse if we needed to use one relation in several different ways.

For example, we might need to know our suppliers' phone numbers for other reasons than just ordering specific parts. Let's say we also needed to associate phone numbers with billing questions. We wouldn't want to store them in yet *another* place.

So the operation of relation composition is of significant practical significance.

Nifty Aside

Much of the world's structured data (in contrast to raw text of the sort we typically read on the Web) is stored in commercial relational data bases. One of the fundamental operations in such databases is ***join***, which is a generalization, to multiple fields that can be connected at once, of composition as we've just defined it.

Problems

1. Recall our definitions of the relations *Sources* and *Contacts*:

Sources

Part	***Supplier***
Brushes	PaintLand
Buckets	ToolLand
Drill Bits	ToolLand
Drill Bits	ZCorp
Filler	ZCorp
Knobs	ZCorp
Nails	ZCorp
Scrapers	PaintLand
Screws	ZCorp

Contacts

Supplier	***Phone Number***
PaintLand	777-888-9999
ToolLand	666-111-2222
ZCorp	555-333-4444

Mark each of the following claims true or false:

a) (Brushes, Paintland) $\in$ *Contacts* $\circ$ *Sources*.
b) (Brushes, 777-888-9999) $\in$ *Contacts* $\circ$ *Sources*.
c) (Scrapers, 777-888-9999) $\in$ *Sources* $\circ$ *Contacts*.
d) (Filler, 777-888-9999) $\in$ *Contacts* $\circ$ *Sources*.
e) (Screws, 555-333-4444) $\in$ *Sources* $\circ$ *Contacts*.

2. Suppose that we are a catering company. Every event we produce has a menu, consisting of a set of dishes. And every dish is defined by a recipe that contains a set of ingredients. So we have three sets (*Events*, *Dishes*, *Ingredients*) and two relations (*Menus*, *RequiredIngredients*), parts of which are shown in these tables:

Menus

Events	***Dishes***
Wedding23	Quesadillas
Wedding23	Guacamole
Wedding23	Fajitas
Birthday57	Quesadillas
Birthday57	Fried Chicken
Birthday57	Cole Slaw

RequiredIngredients

Dishes	***Ingredients***
Quesadillas	Tortillas
Quesadillas	Cheese
Quesadillas	Onions
Guacamole	Avocados
Guacamole	Tomatoes
Guacamole	Onions
Fried Chicken	Chicken
Fried Chicken	Corn Flakes
Cole Slaw	Cabbage
Cole Slaw	Mayonnaise

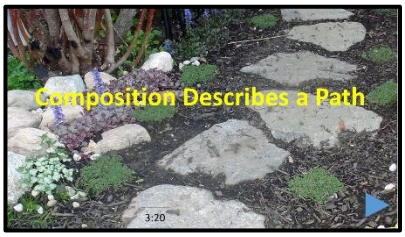
We are getting ready to prepare for an event and we need to gather all of its required ingredients. So we want to generate pairs drawn from *Events* $\times$ *Ingredients*. Which of the following expressions produces what we need:

a) *RequiredIngredients* $\circ$ *Menus*
b) *Menus* $\circ$ *RequiredIngredients*
c) $Menus^{-1} \circ RequiredIngredients$
d) *Events* $\times$ *Ingredients*
e) *Events* $\circ$ *Ingredients*

Composition Describes a Path

One way to think about a binary relation is that it describes single-step paths that connect elements of its domain to elements of its codomain.

Extending that idea: One way to think about composing binary relations is that we can build longer paths.



https://www.youtube.com/watch?v=sBo9M9gkhl4

Imagine a standard part of the database of every college:

Define: $EnrolledIn \subseteq Students \times Classes = \{(s, c) : \text{student } s \text{ is in class } c\}$
$OfferedBy \subseteq Classes \times Departments = \{(c, d) : \text{class } c \text{ is offered by department } d\}$

Suppose that we want to relate students to the departments in which they are taking classes.

Step 1: $Students \rightarrow Classes$ *EnrolledIn* takes this step

Step 2: $Classes \rightarrow Departments$ *OfferedBy* takes this step

Summarizing: $Students \rightarrow Classes \rightarrow Departments$

So we apply *EnrolledIn* (Step 1) first. Then *OfferedBy* (Step 2). But remember that the standard notation for writing this composition applies the relations right to left. So we write:

$OfferedBy \circ EnrolledIn$

Composing a Binary Relation with Itself

Composition combines a relation S with a relation R. What happens if S and R are the same relation? Then we'll say that we have composed a relation with itself. So we have:

$$R \circ R = \{(a, c) : \exists b\ ((a, b) \in R \wedge ((b, c) \in R)\}$$

By analogy with: $x \cdot x = x^2$, $x \cdot x \cdot x = x^3$, and so forth, we'll write:

$R \circ R = R^2$, $R \circ R \circ R = R^3$, and so forth.

(Just as an aside: This notation could be confusing. A relation is a set and we've already defined what A^2 means when A is a set. It means $A \times A$. If A happened to be a binary relation (thus its elements looked like (a, b)) and we used that definition, we'd get elements that look like $((a, b), (c, d))$. In the definition that we've just given here (where $A^2 = A \circ A$), on the other hand, we get elements that look like (a, d). Sadly we're stuck with this problem since the notation really is used to mean both things. But, in our discussions, we will avoid confusion by, whenever R is a relation, taking R^2 to be $R \circ R$.)

Sometimes something interesting happens when we compose a relation with itself.

ParentOf is a relation on the set *People*. So:

$$ParentOf \subseteq People \times People = \{(p, r): p \text{ is the parent of } r\}$$

What is $ParentOf^2$?

Problems

1. There's a famous hypothesis that all pairs of people on the planet are no more than six steps removed from each other, where one step corresponds to a link between one person and someone that person actually knows. Think, "friend of a friend".

Let's describe this claim in terms of the composition of a relation with itself (several times). Assume the domain *People*.

(Part 1) First we need a relation that describes the primitive connection that can exist between a pair of people. Fill in the blank in this definition:

Knows = {(*x*, *y*): _____}

(Part 2) Now we need to form a relation that contains all pairs (*a*, *b*) such that *a* and *b* are connected by no more than six steps of the *Knows* relation.

We already know how to say that two people are connected by exactly one step of the *Knows* relation: That's exactly what *Knows* does.

Next we need to define a relation that contains all pairs (*a*, *b*) such that *a* and *b* are connected by exactly two steps of the *Knows* relation. Write down a concise way of describing this set.

(Part 3) Now we need to combine *Knows* and $Knows^2$ to get all pairs that are connected by either one or two steps. The obvious way to do that is to exploit the fact that relations are sets. So we can operate on them using all the standard set operations. The one we need here is *union*. So this relation does what we need:

$$Knows \cup Knows^2$$

Suppose that *Knows* contains the following elements (among many, many others):

(Morgan, Chip), (Chip, Morgan), (Taylor, Brett), (Brett, Taylor), (Jan, Chris), (Chris, Jan), (Davis, Jan), (Jan, Davis), (Kelly, Sasha), (Sasha, Kelly), (Brett, Kelly), (Kelly, Brett),
(Kelly, Davis), (Davis, Kelly), (Jean, Collette), (Collette, Jean)

For each of the following ordered pairs, mark true if it is in $Knows \cup Knows^2$. Mark false otherwise.

a) (Sasha, Brett)
b) (Sasha, Collette)
c) (Sasha, Kelly)
d) (Taylor, Davis)

2. We can now finish our discussion of the six degrees of separation claim. Write out a concise description, in terms of *Knows*, for the set of ordered pairs of people who are no more than six steps away from each other.

Composing a Binary Relation with its Inverse

Sometimes interesting things happen when we start with some relation R and then we compute:

$$R^{-1} \circ R$$

https://www.youtube.com/watch?v=tdo12pjrnww

In other words, we apply R's inverse to R itself.

Let *Speaks* $\subseteq$ *People* $\times$ *Languages* = {(*p*, *g*): *p* speaks *g*}. So *Speaks* might contain elements such as:

(Kim, English), (Jean, French), (Lu, Chinese), (Terry, English), (Morgan, English)

And $Speaks^{-1}$ would then contain elements such as:

(English, Kim), (French, Jean), (Lu, Chinese), (English, Terry), (English, Morgan)

So, since Kim is related to English via *Speaks* and English is related to Kim, Terry, and Morgan via $Speaks^{-1}$, we have that $Speaks^{-1} \circ Speaks$ contains, among other things:

(Kim, Kim), (Kim, Terry), (Kim, Morgan)

So we see that $Speaks^{-1} \circ Speaks$ relates people to other people who speak the same language.

Sometimes, both $R^{-1} \circ R$ and $R \circ R^{-1}$ can be useful.

Recall this example from a standard college database:

Define: *EnrolledIn* $\subseteq$ *Students* $\times$ *Classes* = {(*s*, *c*) : student *s* is in class *c*}

Suppose that we want to relate pairs of students who share a common class. We want to go:

Students $\rightarrow$ *Classes* $\rightarrow$ *Students*

So (since we evaluate right to left) the composed relation that we need is:

$EnrolledIn^{-1} \circ EnrolledIn$

Now suppose that we want to relate pairs of classes with at least one student in common. We want to go:

Classes $\rightarrow$ *Students* $\rightarrow$ *Classes*

Now the composed relation is:

$EnrolledIn \circ EnrolledIn^{-1}$

Problems

1. Recall the example: $Speaks \subseteq People \times Languages = \{(p, g): p \text{ speaks } g\}$.

Which of the following composed relations relates people to other people with whom they can communicate via a single interpreter (i.e., no chain of interpretations is required):

a) $Speaks^{-1} \circ Speaks$
b) $Speaks^{-1} \circ Speaks^{-1}$
c) $Speaks^2$
d) $(Speaks^{-1} \circ Speaks)^2$
e) $(Speaks \circ Speaks^{-1})^2$

2. Define: $RequiredIngredients \subseteq Dishes \times Ingredients =$

$\{(d, i)$: ingredient i is required for dish $d\}$

Suppose that we know that, when we make a dish, there are likely to be some ingredients left over. So we want to find other dishes that could use those ingredients. Which of the following composed relations relates dishes to other dishes that call for the same ingredients:

a) $RequiredIngredients^2$
b) $RequiredIngredients^{-1} \circ RequiredIngredients$
c) $RequiredIngredients \circ RequiredIngredients^{-1}$
d) $(RequiredIngredients^{-1} \circ RequiredIngredients)^2$
e) $(RequiredIngredients \circ RequiredIngredients^{-1})^2$

Summarizing the Basics of Relations

Before we go on, we'll do a set of problems to review the basics of relations as we've defined them.

Problems

1. Let $A = \{1, 2, 3, 4, 5\}$ and $B = \{4, 6, 7, 8, 9\}$.

Let R be a relation between A and B defined by:

$$R = \{(1, 7), (4, 6), (5, 6), (2, 8)\}.$$

Compute the relation R^{-1}. Which of the following elements is *not* in R^{-1}?

a) (6, 5)
b) (5, 6)
c) (7, 1)
d) (8, 2)
e) (6, 4)

2. Let $A = \{1, 2, 3, 4, 5, 6\}$, $B = \{4, 6, 7, 8, 9\}$, and $C = \{2, 4, 6, 8, 10\}$.

Let R be a relation between A and B defined by:

$$R = \{(1, 7), (4, 6), (5, 6), (2, 8), (6, 7)\}.$$

Let S be a relation between B and C defined by:

$$S = \{(4, 2), (6, 2), (6, 4), (7, 10), (8, 2), (9, 6)\}$$

(Part 1) Compute the relation S^{-1}. Which of the following elements is *not* in S^{-1}?

a) (2, 4)
b) (6, 9)
c) (6, 2)
d) (4, 6)
e) (10, 7)

(Part 2) Compute the relation $S \circ R$. Which of the following pairs of elements are *not* both in $S \circ R$?

a) (1, 10), (4, 2)
b) (5, 4), (6, 10)
c) (8, 8), (6, 6)
d) (2, 2), (4, 4)
e) (4, 4), (5, 2)

3. Let $B = \{4, 6, 7, 8, 9\}$ and $C = \{2, 4, 6, 8, 10\}$.

Let S be a relation between B and C defined by:

$$S = \{(4, 2), (6, 2), (6, 4), (7, 10), (8, 2), (9, 6)\}$$

Recall that, therefore, $S^{-1} = \{(2, 4), (2, 6), (4, 6), (10, 7), (2, 8), (6, 9)\}$.

(Part 1) Compute the relation $S \circ S^{-1}$. Which of the following elements does it contain?

a) (4, 4)
b) (6, 8)
c) (8, 4)
d) (9, 9)
e) (6, 10)

(Part 2) Compute the relation $S^{-1} \circ S$. Which of the following elements does it contain?

a) (4, 8)
b) (10, 10)
c) (2, 4)
d) (10, 2)
e) (4, 2)

4. Let A = {1, 2, 3, 4, 5}, B = {6, 7, 8, 9}, C = {10, 11, 12, 13}, and D = {α, β, γ, δ}.

Let R be a relation between A and B defined by:

R = {(1, 7), (4, 6), (5, 6), (2, 8)}.

Let S be a relation between B and C defined by:

S = {6, 10), (6, 11), (7, 10), (8, 13)}

Let U be a relation between C and D defined by:

U = {(11, β), (10, β), (13, δ), (12, α), (13, γ)}

(Part 1) Compute the relation $U \circ S$. Which of the following elements does it contain?

a) (6, β)
b) (5, β)
c) (β, 6)
d) (γ, 8)
e) (β, 8)

(Part 2) Compute the relation $(U \circ S) \circ R$. Which of the following elements does it *not* contain?

a) (2, β)
b) (5, β)
c) (1, β)
d) (4, β)
e) (2, γ)

(Part 3) Compute the relation $U \circ (S \circ R)$. Assume that you've already computed:

$S \circ R$ = {(1, 10), (4, 10), (4, 11), (5, 10), (5, 11), (2, 13)}.

Which of the following elements is *not* in $U \circ (S \circ R)$?

a) (2, β)
b) (5, β)
c) (1, β)
d) (4, β)
e) (2, γ)

Notice that $(U \circ S) \circ R = U \circ (S \circ R)$. It is possible to prove the more general claim that relation composition is associative. In other words, for all A, B, and C, $(A \circ B) \circ C = A \circ (B \circ C)$.

Soundex

The Soundex system is an interesting example of the use of relations. It is a phonetic algorithm: it indexes names by their sounds. It is designed so that similar sounding names will be assigned the same index, thus making it possible to match names despite spelling variations. It is used to search genealogy databases since spellings of names have often been changed over the years (and the miles). It is now built into many commercial database products.

https://www.youtube.com/watch?v=V6cFEJsqQ40

Soundex takes a name and converts it to a four-symbol code that is designed so that similar sounding names will map to the same code. Once one has the code for one's name, one can run the system backwards to retrieve other names that map to the same code.

Here's the algorithm that converts a name to a code:

1. If two or more adjacent letters (including the first) would map to the same number, remove all but the first. Include as "adjacent" letters that are separated just by H's and W's.

2. Set the first letter of the code to the first letter of the name.

3. For all other letters of the name do:
   1. Convert the letters B, P, F, V, C, S, G, J, K, Q, X, Z, D, T, L, M, N, and R to numbers and tack the resulting number onto code (on the right):
      B, P, F, V = 1
      C, S, G, J, K, Q, X, Z = 2
      D, T = 3
      L = 4
      M, N = 5
      R = 6
   2. Delete all instances of the letters A, E, I, O, U, Y, H, and W.

4. If the code string contains more than three numbers, delete all but the leftmost three.

5. If the code string contains fewer than three numbers, pad with 0's on the right to get three.

Problems

1. Which of the following names doesn't have the same Soundex code as the name Fairley? You probably want to use the Soundex converter:

http://resources.rootsweb.ancestry.com/cgi-bin/soundexconverter

a) Farrell
b) Farwell
c) Friel
d) Fried
e) Frailey

2. Let *Names* be the set of last names. Let *Codes* be the set of strings that the Soundex conversion algorithm can produce. We can use the notations that we've developed to describe Soundex as a pair of relations. We need to define:

- *SoundexCode*, which relates a name to its code, and
- *SoundexEQ*, which relates two names if and only if they have the same Soundex code.

Read these statements carefully. Pay particular attention to whether each one "makes sense", by which we mean that set operators are applied only to sets and Boolean operators are applied only to Boolean values.

Which one or more of the following pairs of definitions correctly defines *SoundexCode* and *SoundexEQ*:

I. $SoundexCode \subseteq Names \times Codes = \{(name, c) : c \text{ is the code produced by Soundex for } name\}$
$SoundexEQ \subseteq Names \times Names =$
$\{(name_1, name_2) : \exists c \in Codes\ ((name_1, c) \in SoundexCode \wedge (name_2, c) \in SoundexCode)\}$

II. $SoundexCode \subseteq Names \times Codes = \{(name, c) : c \text{ is the code produced by Soundex for } name\}$
$SoundexEQ \subseteq Names \times Names =$
$\{(name_1, name_2) : \exists c \in Codes\ ((name_1, c) \wedge (name_2, c) \in SoundexCode)\}$

III. $SoundexCode \subseteq Names \times Codes = \{(name, c) : c \text{ is the code produced by Soundex for } name\}$
$SoundexEQ \subseteq Names \times Names =$
$\{(name_1, name_2) : \exists c_1, c_2 \in Codes\ ((name_1, c_1) \in SoundexCode \wedge (name_2, c_2) \in SoundexCode)\}$

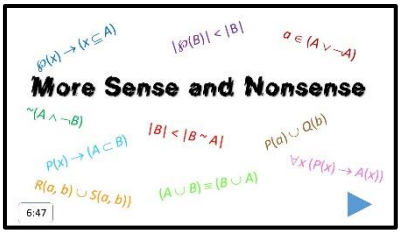
IV. $SoundexCode \subseteq Names \times Codes = \{(name, c) : c \text{ is the code produced by Soundex for } name\}$
$SoundexEQ \subseteq Names \times Names = SoundexCode^{-1} \circ SoundexCode$

Making Sure that We Don't Write Nonsense

At this point, we will often find ourselves working with several different ***types*** of things, including:

- Primitive objects like numbers, people and courses.
- Sets of objects.
- Logical statements.
- Cross products and relations

It's important, if we don't want to spout nonsense, that we use all of these things only in ways that are defined for their types. We talked about this back in our discussion of sets, but it's worth looking at it again.



https://www.youtube.com/watch?v=uyBk03YYx_g

Assume that a is a primitive object, A and B are sets, and P is a logical predicate. Consider the statement:

$$a \in P(A \times B)$$

This expression is nonsense. P is a predicate. It's fine to have a predicate that evaluates a property of a set (such as $A \times B$). But P itself has a value (*True* or *False*). It does not have elements (such as a).

Problems

1. Mark each of the following claims as true if it is well-defined and false if it is nonsense. (Note that we're not actually asking whether or not the claim is true. Just whether it is syntactically well-formed and whether the types of the objects match the roles that they play in the expression.) Assume that a and b are primitive objects, A and B are sets, P and Q are logical predicates, and R and S are binary relations. (Remember the three notations that we are using for relations.)

a) $(A \times B) \in (A \times B)$
b) $P((a, b)) \wedge Q(a)$
c) $a \in R \wedge S$
d) $\forall x\,(P(x) \cup Q(x))$
e) $(R \cup S) \subseteq (A \times B)$

Representing Binary Relations

Approaches to Representation

Commercial relational databases encode *n*-ary relations (for arbitrary values of *n*) in a variety of efficient ways. We won't go into those here.

> ***Nifty Aside***
>
> One estimate put the size of the relational database market at about $24 billion in 2011. And it's growing.

But binary relations are so important in many kinds of applications that it does seem worth it to look at ways to represent them (on paper for ourselves as we study them and, more importantly, within programs that exploit them).

Let's consider four techniques for representing an arbitrary binary relation *R*:

1) We can simply list the elements of *R*.
2) We can encode *R* as a computational procedure.
3) We can encode *R* using a structure we'll call a directed graph.
4) We can encode *R* using a structure we'll call an adjacency matrix.

Let's consider option 1 first. It seems the simplest. We'll just write down all the elements of *R*.

Likes = { (Dave, pie), (Dave, ice cream),
(Sara, cake), (Sara, ice cream),
(Billy, cake), (Billy, ice cream)}

There are three reasons that we want to consider alternatives to this:

- It only works for finite relations.

Try doing this for a relation like < on the integers.

- It isn't very computationally efficient in terms either of time or of space.
- It isn't easy to visualize.

Next let's try option 2. We'll encode *R* as a computational procedure. As with any set, there are at least two ways in which a computational procedure can define *R*. It may:

- ***enumerate*** the elements of *R*, or
- ***recognize*** the elements of *R* by implementing *R*'s characteristic function. It must return *True* when given a pair that is in *R* and *False* when given anything else.

Define the relation *BiggerPrime* on the natural numbers as follows:

$BiggerPrime \subseteq \mathbf{N} \times \mathbf{N} = \{(i, j) : j \text{ is prime and } j > i\}$

For example $(8, 11) \in BiggerPrime$ because 11 is a prime number that is greater than 8.

Here's a simple Python program that, when given an ordered pair, returns True if the pair is in *BiggerPrime* and False otherwise. Assume that the function Prime(n) has already been defined; it returns True if n is prime and False otherwise.

```
def bigger_prime(i, j):
    if i == j or i > j:
        return (False)

    # Now we know that j is bigger.  Now check prime.
    if prime(j):
        return (True)
    else:
        return (False)
```

As we'll soon see, this computational approach is the only thing that works when we're dealing with infinite relations.

Problems

1. Recall the relation *BiggerPrime* that we defined with the following Python code:

```
def bigger_prime(i, j):
    if i == j or i > j:
        return (False)

    if prime(j):      # We know that j is bigger.
        return (True)
    else:
   return (
```

Mark each of the following ordered pairs true if it is an element of *BiggerPrime* and false otherwise:

(5, 11) (11, 5) (13, 13) (7, 12)

2. Mark each of the following relations true if it would be possible to represent it by writing down all its elements. Mark it false otherwise.

a) Equality defined on the integers.
b) Equality defined on the natural numbers less than 10.
c) $SocialSecurityNumber \subseteq USResidents \times$ 9-digit numbers =
   $\{(x, n) : n \text{ is the social security number of } x\}$

Undirected Graphs

Our third idea is to represent a relation as a graph. Before we can talk about how to do that, we need a brief digression: We need to learn about graphs.

The simplest kind of graph is an undirected graph. Formally, an ***undirected graph*** is an ordered pair of sets:

$G = (V, E)$

where:

- V is a set of ***vertices*** (also often called ***nodes***), and
- E is a set of ***edges***, each of which is an unordered pair (v_1, v_2) of vertices. Note that, since an edge is an unordered pair, if $(v_1, v_2) \in E$, (v_2, v_1) is also in E.

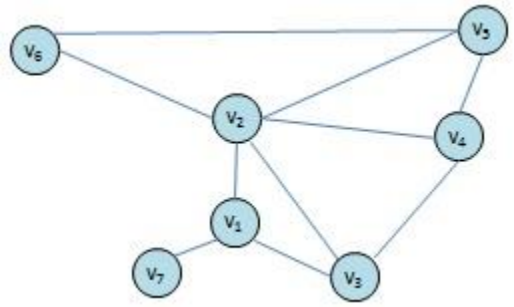
Given a graph $G = (V, E)$, a sequence $p = (v_{i_1}, v_{i_2}, \ldots, v_{i_n})$ is a ***path*** from v_{i_1} to v_{i_n} if, for all $k = 1, 2, \ldots, n\text{-}1$, $(v_{i_k}, v_{i_{k+1}}) \in E$. The ***length*** of such a path is $n - 1$.

Graphs are easy to visualize.

An airline's route map can be represented as a graph where:

- The vertices represent cities, and
- The edges represent flights.

Let: v_1 represent Austin,
v_2 represent Dallas,
v_3 represent Houston,
v_4 represent New York,
v_5 represent Boston,
v_6 represent Seattle, and
v_7 represent Fredericksburg.



An edge between two cities means that there is a direct flight between them.

Then this graph tells us, for example, that there are paths (v_1, v_2), (v_1, v_3), and (v_1, v_2, v_6). But there is no path (v_1, v_6). So you can fly directly from Austin to Dallas and to Houston, but if you want to go to Seattle, you'll have to go to Dallas first. The shortest path between Austin and Seattle has length 2.

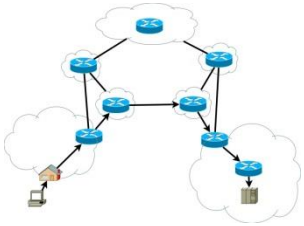
A molecule can be represented as a graph where:

- The vertices are atoms, and
- The edges are bonds.

A computer network can be represented as a graph where:

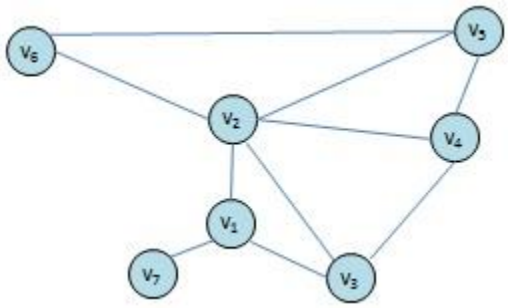
The vertices are computers and routers, and
The edges are cables.

The Internet is an example of a VERY large network. Represented as a graph, it has more than 30 billion vertices.



Problems

1. Recall the airline route map graph that we just considered:



Let: v_1 represent Austin,
v_2 represent Dallas,
v_3 represent Houston,
v_4 represent New York,
v_5 represent Boston,
v_6 represent Seattle, and
v_7 represent Fredericksburg.

Let an edge between two cities mean that there is a direct flight between them.

(Part 1) What is the maximum number of legs that a passenger might have to fly to get one from city to another?

(Part 2) What is the length of the shortest path between Houston and Seattle?

(Part 3) Suppose that the airline were willing to add one new city pair to its flight list. The goal is to decrease the worst case (i.e., the maximum number of legs that a passenger would have to fly to get from one city to another.) What two cities should be connected by this new leg? Describe a city by its vertex number. Enter your answer as two numbers, separated by a single space. Enter the smaller number first.

Directed Graphs

While undirected graphs are useful in many problem contexts (as we've just seen), they're not exactly what we need for representing relations. The issue is that an element of a relation is an *ordered* pair. Given two objects o_1 and o_2, it is possible that a relation R contains:

- both (o_1, o_2) and (o_2, o_1),
- just (o_1, o_2),
- just (o_2, o_1), or
- neither (o_1, o_2) nor (o_2, o_1).

So we need a new structure that captures order.

A ***directed graph*** (sometimes called a ***digraph***) is an ordered pair:

$$G = (V, E)$$

where:

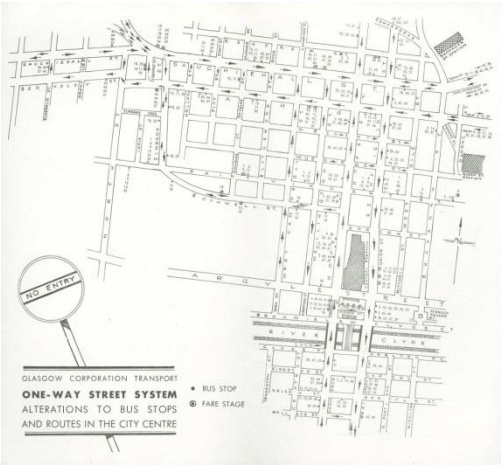
- V is a finite set of vertices (or nodes), and
- E is a finite set of edges, each of which is an *ordered* pair (v_1, v_2) of vertices.

Define path and path length as for an undirected graph (except, of course, that it is now possible that E contains the edge (v_1, v_2) but not the edge (v_2, v_1)).

The road map of a city can be represented as a graph where:

The vertices are the intersections and end points, and
The edges are the roads. The edge (v_1, v_2) exists whenever there is a road such that one can drive directly from v_1 to v_2.

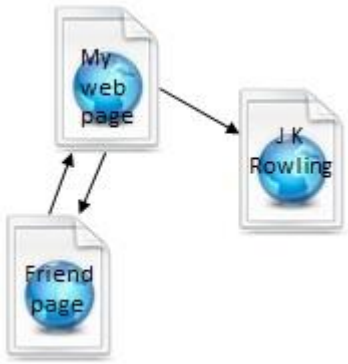
We use a directed graph because one-way streets exist. If a street is two-way, then there will be two edges that connect its intersections: one going one way and one going the other.



The World Wide Web can be represented as a graph where:

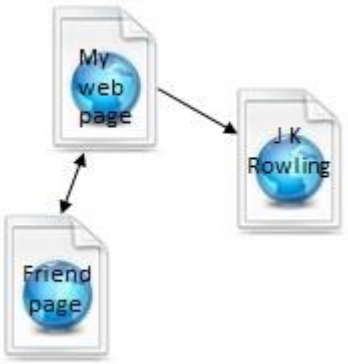
The vertices are the web pages, and
The edges are the links from one page to another. The edge (p_1, p_2) exists whenever $page_1$ contains a link to $page_2$.

We use a directed graph because $page_1$ can link to $page_2$ without $page_2$ linking back to $page_1$. For example, I can, on my web page, put a link to J. K. Rowling's page. But I probably can't convince her to link to me. On the other hand, I can link to my best friend's page and she can link to mine.



We'll often use a notational shorthand whenever both the edge (v_1, v_2) and the edge (v_2, v_1) exist in a graph. Instead of drawing them separately, we'll draw a single edge with an arrow at both ends.

Using this convention, the small web fragment that we showed above would be drawn as:



Using a Directed Graph to Represent a Relation

We can now see how to represent a relation as a graph. In fact, it should be clear that a directed graph corresponds exactly to a relation: It is a set *V* and a set of ordered pairs whose elements are drawn from *V*.

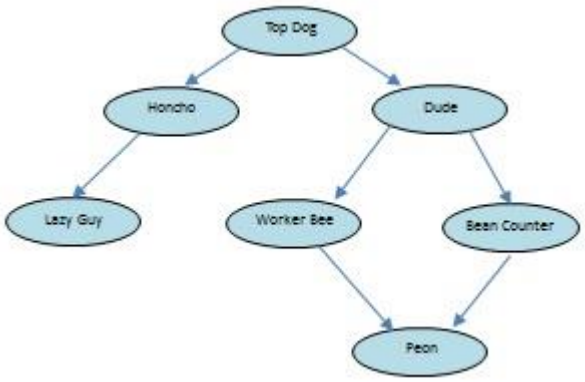
The simplest case arises when *R* is a relation on a single set *A*. Then, in a graph representation of *R*:

- The vertices are the elements of *A*, and
- The edges are the elements of *R*. Specifically, there is an edge from v_i to v_j if and only if $(v_i, v_j) \in R$.

For example, the *Supervises* relation in our corporate database might contain elements such as:

(Top Dog, Honcho), (Honcho, Lazy Guy), and (Bean Counter, Peon).

It might look like this when represented as a directed graph:



We can also represent relations from one set to another. In that case, we must create vertices for the elements of both sets.

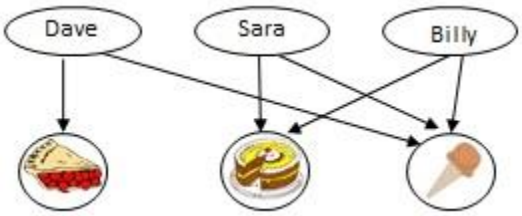
Recall the relation *Likes* from a set *A* of people to a set *B* of desserts:

Likes = { (Dave, pie), (Dave, ice cream),
(Sara, cake), (Sara, ice cream),
(Billy, cake), (Billy, ice cream)}

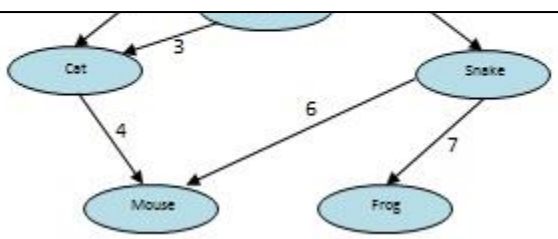
We can represent *Likes* as a directed graph:

People:

Desserts:



Predator/Prey if and only if, in nature, a_1 is a predator of a_2. Consider this proposed graph representation of a subset of the relation:



Which edge is incorrect?

2. WordNet® is a large lexical database of English. It contains entries for nouns, verbs, adjectives, and adverbs. It encodes several important relations:

- *Synonym*: ($word_1$, $word_2$) $\in$ *Synonym* if they share a meaning.
- *Hypernym*: ($word_1$, $word_2$) $\in$ *Hypernym* if a meaning of $word_2$ is a generalization of $word_1$.
- *Hyponym*: ($word_1$, $word_2$) $\in$ *Hyponym* if a meaning of $word_2$ is a specialization of $word_1$.

Notice that *Hyponym* = $Hypernym^{-1}$. For example:

(textbook, book) $\in$ *Hypernym*
(book, textbook) $\in$ *Hyponym*

You can play with WordNet online (or download it if you want to use it in a program). You'll find it here: http://wordnet.princeton.edu/ . To see hypernyms and hyponyms of a word, click on the S at the left of an entry.

It helps to visualize the Wordnet relations by drawing (part of) them as graphs, just as we've been doing.

Now consider the relation composition: *Hyponym* $\circ$ *Hypernym*.

Look up the word "telephone". Find its hypernyms. Now find the hyponyms of those. You'll get a set of words whose meanings are, in some sense, siblings of "telephone". They are other similar things.

In Wordnet, which of the following elements is not in *Hyponym* $\circ$ *Hypernym*:

a) (telephone, lens)
b) (telephone, CD player)
c) (telephone, laptop)
d) (telephone, monitor)
e) (telephone, amplifier)

Using an Adjacency Matrix

Now consider another way to represent a directed graph (and thus a relation).

We've already seen that we can represent a finite set S as a bit vector. If the universe U from which S is drawn has n elements, so will the bit vector, which we can call V. Then, to represent S, we'll fill V with Boolean values as follows:

- If $U_k \in S$: $V[k] = True$
- If $U_k \notin S$: $V[k] = False$

Let U be {11, 22, 33, 44, 55, 66, 77, 88, 99}. Let S = {99, 55}. Then we can represent S with the following bit vector (where, for ease of reading, we let blank stand for *False* and • stand for *True*):

11	22	33	44	55	66	77	88	99
				•				•

Assume two sets, A and B. (They can be different sets or the same set.) Now consider either:

- R, a relation from A to B (i.e., $R \subseteq A \times B$), or
- G, a directed graph all of whose edges start in some element of A and end in some element of B.

Then, to represent R or G, we construct an ***adjacency matrix*** M with:

- one row for every element of A, and
- one column for every element of B.

Let $M(i, j)$ refer to the i^{th} row, j^{th} column of M. (We'll label the rows and columns of our matrices. Don't count those labels as a row or a column.)

	cake	okra	pie	ice cream
Billy				$M(1, 4)$
Dave				
Sara				

Then, to represent R (or G):

- If $(A_i, B_j) \in R$ (or there is an edge from A_i to B_j in G): $M(i, j) = True$
- If $(A_i, B_j) \notin R$ (or there is not an edge from A_i to B_j in G): $M(i, j) = False$

Let *Likes* be a relation from a set A = {Billy, Dave, Sara,} of people to a set B = {cake, okra, pie, ice cream} of foods:

Likes = { (Dave, pie), (Dave, ice cream),
(Sara, cake), (Sara, ice cream),
(Billy, cake), (Billy, ice cream)}

We can represent *Likes* as the following adjacency matrix M (where, for ease of reading, we again let blank stand for *False* and • stand for *True*):

	cake	okra	pie	ice cream
Billy	•			•
Dave			•	•
Sara	•			•

The name "adjacency matrix" is natural if you think of $M(i, j)$ as being *True* whenever, in a directed graph, A_i is adjacent to B_j in the sense that it is possible to get from A_i to B_j in a single step.

Problems

1. Consider the following tiny fragment of a possible *MotherOf* relation on the set of people:

{(Doreen, Ann), (Ann, Mark), (Ann, Cathy), (Doreen, Marie), (Marie, Christy)}

Suppose that we propose the following matrix *M* to represent this piece of *MotherOf*. We have numbered the cells that contain the value *True*.

	Ann	Cathy	Christy	Doreen	Marie	Mark
Ann		1		2		3
Cathy						
Christy						
Doreen	4				5	
Marie			6			
Mark						

One cell is *True* but it shouldn't be. Enter the number of the incorrect cell.

2. Consider the following tiny fragment of a possible *SiblingOf* relation, also defined on the set of people:

{(Ann, Marie), (Beth, Ann), (Rich, Marie), (Steve, Trish), (Mike, Beth)}

Suppose that we propose the following matrix *M* to represent this piece of *SiblingOf*.

	Ann	Beth	Marie	Mike	Rich	Steve	Trish
Ann			•				
Beth	•						
Marie							
Mike							
Rich			•				
Steve							•
Trish							

We've left out one *True* value that should be there. Into which of the following cells of *M* does it need to go:

a) (2, 4)
b) (4, 2)
c) (3, 5)
d) (3, 1)
e) (7, 6)

Properties of Binary Relations on a Set

The Special Nature of Binary Relations on a Set

Relations need not be binary. And they may map from elements of one set to elements of some different set.

But the special case of *binary* relations on a *single set* is so important that it's worth additional study.

Consider the set P of people. Here are some useful binary relations on P:

- *MotherOf*
- *SiblingOf*
- *BossOf*
- *FriendOf*

Consider the set $\mathbf{Z}$ (the integers). Here are some useful binary relations on $\mathbf{Z}$:

- $Succ(i, j) \subseteq (\mathbf{Z} \times \mathbf{Z}) = \{(i, j) : j = i + 1\}$. For example, 5 is the successor of 4.
- $GreaterThan(i, j) \subseteq (\mathbf{Z} \times \mathbf{Z}) = \{(i, j) : i > j\}$. We often write *GreaterThan* using infix notation. So we can write, for example, 10 > -2.
- $DivisibleBy(i, j) \subseteq (\mathbf{Z} \times \mathbf{Z}) = \{(i, j) : i$ is evenly divisible by nonzero $j\}$. For example:

  DivisibleBy(12, 3) and *DivisibleBy*(12, 4).

Problems

1. (Part 1) Suppose that you wanted to encode some of your real world knowledge as relations. Which of the following things would *not* naturally be able to be described using a *binary* relation on a *single set*:

a) *CousinOf*
b) *SocialSecurityNumberOf*
c) *IngredientOf*
d) *BorderingCountries*
e) *SubsetOf*

(Part 2) Suppose that you wanted to encode, as relations, some of your mathematical knowledge about the integers (the set **Z**). Which of the following things would *not* naturally be able to be described using a binary relation on **Z**:

a) *HasMorePrimeFactorsThan*
b) *GreaterThanOrEqualTo*
c) *SquareOf*
d) *IsHalfOf*
e) *IsRemainderAfterDividingBy10*

Modular Arithmetic

Modular arithmetic, defined on the natural numbers **N**, is based on the notion of remainder after division.

Define *n* modulo (or just "mod") *k* as the remainder after *n* is divided evenly by *k*.

For example, 7 mod 3 is 1 because, if we divide 7 by 3, we get a remainder of 1.

Notice that, if *n* mod *k* is *j* then: $\exists i \in \mathbf{N}\ (n = (i \cdot k) + j)$

In other words, *k* goes into *n* *i* times, with a remainder of *j* left over.

For example: 15 mod 5 = 0 because 15 = (3 · 5) + 0 (*i* = 3)
32 mod 5 = 2 because 32 = (6 · 5) + 2 (*i* = 6)

Nifty Aside

Traditional clocks display hours modulo 12 (except that they write 0 as 12). Clocks don't tell us how many times the hour hand has gone all the way around. For example, if it's midnight now, then 28 hours from now, a clock will display the hour as 4 since 28 mod 12 = 4.

Modular arithmetic also plays a crucial role (as do prime numbers) in the modern cryptography systems that make secure communication on the Internet possible.

Modular arithmetic is the basis for a very useful set of relations on the set **N**. In particular, define an entire family of relations (one for each positive integer value of *k*) on the natural numbers as follows:

$$\equiv_k = \{(i, j) : (i \bmod k) = (j \bmod k)\}$$

If $(i, j) \in \equiv_k$, then we'll write $i \equiv_k j$. Read this as, "*i* is equivalent to *j*, mod *k*."

For example:

$7 \equiv_3 31 \equiv_3 1$ (7, 31, and 1 all have the same remainder, 1, when divided by 3)
$6 \equiv_2 32 \equiv_2 0$ (6, 32, and 0 all have the same remainder, 0, when divided by 2)

Problems

1. Mark each of the following statements true or false:

$16 \equiv_{10} 2$ $8 \equiv_3 35$ $50 \equiv_5 0$ $17 \equiv_4 2$

Properties of Binary Relations on a Set

In this section, we'll explore three important properties that some binary relations on a set possess:

- A binary relation is ***reflexive*** if every element is related to itself.
- A binary relation is ***symmetric*** if, whenever *a* is related to *b*, *b* is also related to *a*.
- A binary relation is ***transitive*** if, whenever *a* is related to *b* and *b* is related to *c*, *a* is also related to *c*.

These terms help us to describe relations that we wish to deal with. They are important in mathematics.

And, in many practical applications, checking for these properties can be important as part of the process of checking, say, for data errors in databases.

Reflexivity

A binary relation is ***reflexive*** if and only if every element is related to itself. In other words, if R is a binary relation on a set A, it is reflexive if and only if:

$$\forall a \in A\ ((a, a) \in R)$$

Consider the relation *GreaterThanOrEqualTo* defined on the integers (or the rationals or the reals). It is reflexive since:

$$\forall a \in \mathbf{Z}\ (a \geq a)$$

Consider the corporate database relation *WorksInSameDepartmentAs*, defined on the set of *Employees*. It is reflexive since everyone works in the same department as him/herself.

A binary relation is ***antireflexive*** (sometimes also called ***irreflexive***) if and only if *no* element is related to itself. In other words, if R is a binary relation on a set A, it is antireflexive if and only if:

$$\forall a \in A\ ((a, a) \notin R)$$

Consider the relation *GreaterThan* defined on the integers (or the rationals or the reals). It is antireflexive since no element is greater than itself:

$$\forall a \in \mathbf{Z}\ (\neg(a > a))$$

Consider the relation *SupervisorOf*, defined on the set of *Employees*. It could play an important role in a large corporate Human Relations database. *SupervisorOf* is antireflexive since no one can be his/her own supervisor.

The ***Identity*** relation is the special reflexive relation in which every element is related to itself and to nothing else. In other words, given a set A, the relation I on A is defined as:

$$I = \{(a, a) : a \in A\}$$

You may be wondering why I is worth bothering to define. Answer:

- Sometimes when we compose relations we discover that we're back where we started. We can then say that the composition is equal to I (and that may be a useful observation).

Define the successor relation (as we have been doing):

$Succ(i, j) \subseteq (\mathbf{Z} \times \mathbf{Z}) = \{(i, j) : j = i + 1\}$.

And define the predecessor relation:

$Pred(i, j) \subseteq (\mathbf{Z} \times \mathbf{Z}) = \{(i, j) : j = i - 1\}$.

Then observe:

$Pred \circ Succ = I = Succ \circ Pred$

For example, the successor of 2 is 3. Then the predecessor of 3 is 2.

- If we want to develop an algebra of relations (in the same sense in which you already know about an algebra on the integers), then I is the identity for composition (in the same sense in which 1 is the identity for multiplication).

Notice that a relation is reflexive if and only if I is a subset of it.

Problems

1. Consider the relation *MotherOf* (in the real world). Which of the following describes it:

a) It is reflexive.
b) It is antireflexive.
c) It is neither reflexive nor antireflexive. In other words, in some, but not all cases, an element is related to itself.

2. Consider the following relation on the *even* integers:

$R = \{(i, j) : i \cdot j \text{ is even}\}$

For example, $(4, 10) \in R$ because $4 \cdot 10 = 40$, which is even.

Which of the following describes R:

a) It is reflexive.
b) It is antireflexive.
c) It is neither reflexive nor antireflexive. In other words, in some, but not all cases, an element is related to itself.

3. Consider the following relation on the *odd* integers:

$R = \{(i, j) : i \cdot j \text{ is even}\}$

For example, $(3, 7) \notin R$ because $3 \cdot 7 = 21$, which is not even.

Which of the following describes R:

a) It is reflexive.
b) It is antireflexive.
c) It is neither reflexive nor antireflexive. In other words, in some, but not all cases, an element is related to itself.

4. Consider the following relation on the integers (all of them):

$R = \{(i, j) : i \cdot j \text{ is even}\}$

For example, $(3, 8) \in R$ because $3 \cdot 8 = 24$, which is even.

Which of the following describes R:

a) It is reflexive.
b) It is antireflexive.
c) It is neither reflexive nor antireflexive. In other words, in some, but not all cases, an element is related to itself.

Graphs and Adjacency Matrices of Reflexive Relations

Suppose that R is a relation on some finite set A. Then we know that we can represent R as both a graph and as an adjacency matrix.

Problems

1. (Part 1) Let R be a reflexive relation on some finite set A. Which of the following must be true of any graph that represents R:

a) For no vertex v is there an edge from v to v.
b) For every vertex v, there is an edge from v to v.
c) For every vertex v, there is an edge to no more than one other vertex.
d) Every pair of vertices v_1 and v_2 is connected by an edge.
e) None of these.

(Part 2) Let R be a reflexive relation on some finite set A. Consider the following claims that we could make about any adjacency matrix M that represents R. Which one or more of these is/are true?

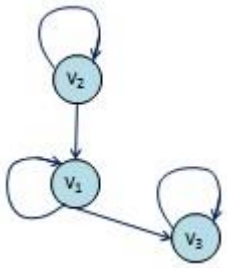
I. If $|A| = n$, at least n cells of M contain the value *True*.
II. Every cell on the diagonal from upper left to lower right contains the value *True*.
III. Every cell on the diagonal from upper right to lower left contains the value *True*.

More on Graphs and Adjacency Matrices of Reflexive Relations

The last two problems suggest that it is easy, using either a graph or an adjacency matrix, to determine whether a relation is reflexive.

Using a graph: If a relation is reflexive and represented as a graph *G*, there will be, for every vertex in *G*, an edge going back to itself. We often call such edges ***loops***.

So we know that this graph corresponds to a reflexive relation:



Using an adjacency matrix: If a relation is reflexive and represented as a matrix *M*, the value *True* will appear in every element of the major diagonal (upper left to lower right) of *M*.

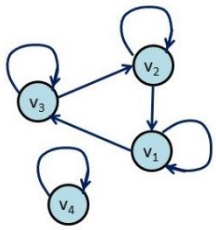
So we know that this matrix corresponds to a reflexive relation (a superset of the one we just saw the graph of):

	v_1	v_2	v_3	v_4
v_1	•			•
v_2	•	•		
v_3			•	
v_4				•

Notice that, using either of these structures, we can check a relation to see whether or not it is reflexive in time proportional to the number of elements in the set on which it is based. We just need to examine each vertex (in a graph) or each element of the major diagonal in an adjacency matrix.

Problems

1. Consider the following graph *G*:



Let *G* correspond to a relation *R* on the set {1, 2, 3, 4}. Is *R* reflexive?

Symmetry

A binary relation is ***symmetric*** if and only if, whenever a is related to b, b is also related to a. In other words, if R is a binary relation on a set A, it is symmetric if and only if:

$$\forall a,b \in A\ (((a, b) \in R)) \rightarrow ((b, a) \in R))$$

Define the relation *CommonFactor* as:

CommonFactor = {(i, j) $\in$ **Z** $\times$ **Z**: i and j share a common integer factor}

For example, (10, 12) $\in$ *CommonFactor*, since 10 and 12 share the factor 2.

CommonFactor is symmetric. If 10 and 12 share a common factor, so do 12 and 10.

Problems

1. Define the ***parity*** of an integer as either:

- Even (in the case of an even integer), or
- Odd (in the case of an odd integer).

Now consider the relation *EqualParity*:

EqualParity = {(i, j) $\in$ **Z** $\times$ **Z**: i and j have the same parity}

True or false: *EqualParity* is symmetric.

2. Assume an arbitrary set A and two symmetric relations R and S on A. Prove or disprove with a simple counter-example the claim: $R \circ S$ must be symmetric.

Which of the following is true:

a) The claim can be proved true.
b) The claim can be proved false with a counterexample.

Antisymmetry

A binary relation is ***antisymmetric*** if and only if, whenever a and b are different and a is related to b, b is *not* related to a. In other words, if R is a binary relation on a set A, it is antisymmetric if and only if:

$$\forall a,b \in A\ ((a \neq b) \wedge ((a, b) \in R)) \rightarrow ((b, a) \notin R))$$

Or, here's an equivalent definition that may be more convenient in some cases:

$$\forall a,b \in A\ ((a, b) \in R) \wedge ((b, a) \in R)) \rightarrow a = b)$$

Again, consider the relation *GreaterThan* defined on the integers (or the rationals or the reals). It is antisymmetric since, if $a > b$, it cannot be true that $b > a$:

$$\forall a,b \in \mathbf{Z}\ (\ (a > b) \rightarrow \neg(b > a))$$

It follows from the fact that *GreaterThan* is antisymmetric that any relation R that is defined by a greater-than relationship on some numeric quantity must also be antisymmetric.

Define the relation *BetterGPAThan* on the set S of students as:

$$BetterGPAThan = \{(r, s) \in S \times S : r \text{ has a higher GPA than } s\}$$

BetterGPAThan is antisymmetric.

There are also interesting antisymmetric relations that don't involve numbers.

Define the antisymmetric relation $ComesBefore \subseteq Strings \times Strings$ =

$\{(s, w) : s$ comes before w in alphabetical order$\}$

For example:

("goblin", "gremlin") $\in ComesBefore$, but ("gremlin", "goblin") $\notin ComesBefore$

Define the relation *Implies* on the set P of Boolean formulas as:

$Implies = \{(p, q) \in P \times P : p \rightarrow q\}$

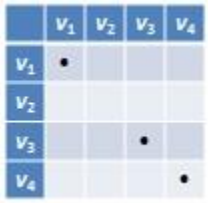
Recall the truth table definition of *Implies*:

	p	q	$p \rightarrow q$
1	T	T	T
2	T	F	F
3	F	T	T
4	F	F	T

To determine whether a relation is antisymmetric, we ignore elements of the form (a, a). In the case of Booleans, that means that we ignore (T, T) and (F, F). So we ignore rows 1 and 4. Now observe that row 3 tells us that $(F, T) \in Implies$. But we know, from row 2, that $(T, F) \notin Implies$. So *Implies* is antisymmetric.

Notice that, somewhat counterintuitively, it is possible for a relation R to be both symmetric and antisymmetric. Suppose that R contains no elements (a, b) where a and b are different. Then, trivially, it is symmetric. Also (and this is most clear from the first definition of antisymmetry given above), R is also antisymmetric.

For example, let R be the relation encoded as the following adjacency matrix:



R is both symmetric (trivially) and antisymmetric (trivially).

Problems

1. Define the ***parity*** of an integer as either:

- Even (in the case of an even integer), or
- Odd (in the case of an odd integer).

Now consider the relation *EqualParity*:

$$EqualParity = \{(i, j) \in \mathbf{Z} \times \mathbf{Z}: i \text{ and } j \text{ have the same parity}\}$$

Indicate, for each of the following claims, whether it is true or false:

a) *EqualParity* is symmetric.
b) *EqualParity* is antisymmetric.
c) *EqualParity* is neither symmetric nor antisymmetric. In other words, in some, but not all cases, $(i, j) \in EqualParity$ implies that $(j, i) \in EqualParity$.

2. Assume an arbitrary set A and two symmetric relations R and S on A. Prove or disprove with a simple counter-example the claim: $R \circ S$ must be symmetric.

Which of the following is true:

c) The claim can be proved true.
d) The claim can be proved false with a counterexample.

3. Assume that some arbitrary relation R on some set A is both symmetric and antisymmetric. Recall that the identity relation, I, on a set A is defined as:

$$I = \{(a, a) : a \in A\}$$

Indicate, for each of the following claims, whether it is true or false:

a) R must be equal to I.
b) R must be a subset of I.
c) R can be any superset of I.
d) It must be the case that $R \cap I = \varnothing$

Graphs and Adjacency Matrices of Symmetric Relations

Suppose that R is a relation on some finite set A. Then we know that we can represent R as both a graph and as an adjacency matrix.

Problems

1. Let R be a symmetric relation on some finite set A. Which of the following must be true of any graph that represents R:

a) For no vertex v is there an edge from v to v.
b) For every vertex v, there is an edge from v to v.
c) For every edge (v_1, v_2) there is also the edge that goes in the other direction.
d) Every pair of vertices v_1 and v_2 is connected by an edge.
e) None of these.

2. Let R be a symmetric relation on some finite set A. Consider the following claims that we could make about any adjacency matrix M that represents R. Which one or more of these claims must be true?

I. If $|A| = n$, at least n cells of M contain the value *True*.
II. Fold M along its major diagonal (the one from the upper left to the lower right). Cells that line up on top of each other must contain the same value.
III. Fold M along its other diagonal (the one from the upper right to the lower left). Cells that line up on top of each other must contain the same value.

More on Graphs and Adjacency Matrices of Symmetric Relations

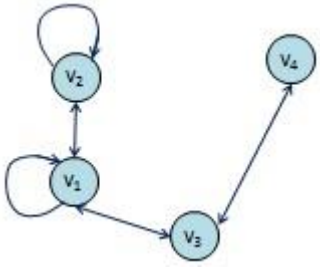
The last two problems suggest that it is easy, using either a graph or an adjacency matrix, to determine whether a relation is symmetric.

Using a graph: If a relation is symmetric and represented as a graph *G*, every edge in *G* will go in both directions. (Recall that we may draw such a pair of edges as a single edge with arrows at both ends.)

So we can easily see that the first of these graphs corresponds to a symmetric relation, but the second one does not (since the edge (v_1, v_3) goes in only one direction).

The only exception to this is that we won't always bother to draw reflexive edges in both directions (we don't need to since there is no difference between (*a*, *a*) and (*a*, *a*)).

So this graph corresponds to a symmetric relation:



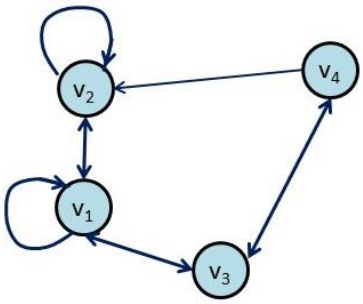
Using an adjacency matrix: If a relation is symmetric and represented as a matrix *M*, then if we fold *M* along its major diagonal, cells that land on top of each other must have the same value.

So we know that this matrix corresponds to a symmetric relation. If you can't visualize this, print it, fold it, and see what happens.

	v_1	v_2	v_3	v_4	v_5
v_1	•		•	•	•
v_2		•			•
v_3	•		•		
v_4	•				
v_5	•	•			•

Problems

1. Consider the following graph *G*:



Let *G* correspond to a relation *R* on the set {1, 2, 3, 4}. Is *R* symmetric?

Transitivity

A binary relation is ***transitive*** if, whenever a is related to b and b is related to c, a is also related to c. In other words, if R is a binary relation on a set A, it is transitive if and only if:

$$\forall a,b,c \in A\ (((a, b) \in R) \wedge ((b, c) \in R)) \rightarrow ((a, c) \in R))$$

One way to think of a transitive relation R is this: If, for two elements a and c, it is possible to get (somehow, via some number of intermediate steps) from a to c, then (a, c) must be in R.

Consider the relation *GreaterThan* defined on the integers (or the rationals or the reals). It is transitive since, if $a > b$, and $b > c$, it must be true that $a > c$:

$$\forall a,b,c \in \mathbf{Z}\ (((a > b) \wedge (b > c)) \rightarrow (a > c))$$

It follows from the fact that *GreaterThan* is transitive that any relation R that is defined by a greater-than relationship on some numeric quantity must also be transitive.

Recall the relation *BetterGPAThan* on the set S of students as:

BetterGPAThan = $\{(r, s) \in S \times S : r$ has a higher GPA than $s\}$

BetterGPAThan is transitive.

If we represent a relation as either a graph or an adjacency matrix, there exist algorithms for deciding whether the relation is transitive. Unfortunately, they're not as straightforward as the ones for deciding whether a relation is reflexive or symmetric.

Reviewing the Properties

The three properties that we've just presented play important roles in mathematics, as well as in many kinds of applications.

Big Idea:

We bother to define these properties (as well as others to come) for at least two important reasons:

- By giving names to the properties, we can talk about them both concisely and precisely.
- Names help us focus on properties that are often very important.

For example, if we assert that a relation R possesses one or more of the properties we've discussed, then we can check for errors in entering R into a database by checking to see whether the property(s) hold.

Suppose that we have a database of sales data. We use it for various things, including giving commissions and awards to the sales staff. Consider the relation:

$$Outsells \subseteq SalesStaff \times SalesStaff = \{(a, b) : a \text{ has outsold } b \text{ in the current period}\}$$

Outsells is:

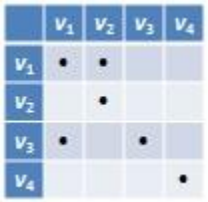
- Antireflexive: No one can outsell him/herself.
- Antisymmetric: If a outsells b, then b does not outsell a.
- Transitive: If a outsells b and b outsells c, then a outsells c.

Notice, by the way, that *Outsells* is really *GreaterThan* in disguise (the quantity being measured is sales volume). So it has the same properties that *GreaterThan* does.

If we check our database and find that our records don't show *Outsells* as having those properties, then we know that a mistake has been made in entering or computing the data.

Problems

1. Let R be the relation described by the following adjacency matrix:



	v_1	v_2	v_3	v_4
v_1	•	•		
v_2		•		
v_3	•		•	
v_4				•

Mark each of the following claims as true or false:

a) R is reflexive.
b) R is antireflexive.
c) R is symmetric.
d) R is antisymmetric.
e) R is transitive.

2. Define the relation *SubsetOf* on the set S of sets (of anything):

$$SubsetOf = \{(A, B) \in S \times S : A \subseteq B\}$$

Notice that saying that $(A, B) \in SubsetOf$ is the same as saying that $A \subseteq B$. This is such an important relation that we've defined an infix notation for it. In fact, when we first defined $\subseteq$, we were defining a relation. We just hadn't formalized that notion in the way that we now have.

Mark each of the following claims as true or false:

a) *SubsetOf* is reflexive.
b) *SubsetOf* is antireflexive.
c) *SubsetOf* is symmetric.
d) *SubsetOf* is antisymmetric.
e) *SubsetOf* is transitive.

3. Consider the relation $\equiv_{10}$, defined on the natural numbers and based on the notion of modular arithmetic that we described several slides ago. Mark each of the following claims as true or false:

a) $\equiv_{10}$ is reflexive.
b) $\equiv_{10}$ is antireflexive.
c) $\equiv_{10}$ is symmetric.
d) $\equiv_{10}$ is antisymmetric.
e) $\equiv_{10}$ is transitive.

4. Consider the relation *TakesAUniversityClassWith* defined on the set of University students.

Mark each of the following claims as true or false:

a) *TakesAUniversityClassWith* is reflexive.
b) *TakesAUniversityClassWith* is antireflexive.
c) *TakesAUniversityClassWith* is symmetric.
d) *TakesAUniversityClassWith* is antisymmetric.
e) *TakesAUniversityClassWith* is transitive.

5. Consider the playoff games of your favorite sport. Let A be the set of teams who are competing. Assume that each pair of teams plays against each other exactly once and that there are no ties. Define the relation:

$$PlayedAndBeat = \{(a, b) \in A \times A: \text{Team } a \text{ played team } b \text{ and beat them}\}$$

(Part 1) Which of the following statements is correct:

a) *PlayedAndBeat* must be reflexive.
b) *PlayedAndBeat* must be antireflexive.
c) *PlayedAndBeat* must be neither reflexive nor antireflexive.
d) We can't make any claim about the reflexivity of *PlayedAndBeat*.

(Part 2) Which of the following statements is correct:

a) *PlayedAndBeat* must be symmetric but not antisymmetric.
b) *PlayedAndBeat* must be antisymmetric but not symmetric.
c) *PlayedAndBeat* must be both symmetric and antisymmetric.
d) *PlayedAndBeat* must be neither symmetric nor antisymmetric.
e) We can't make any claim about the symmetry of *PlayedAndBeat*.

(Part 3) Which of the following statements is correct:

a) *PlayedAndBeat* must be transitive.
b) We can't make any claim about the transitiviy of *PlayedAndBeat*.

6. Let G be any directed graph. Define a ***path*** from vertex a to vertex b as a sequence of edges:

$$(a, v_2), (v_2, v_3), (v_3, v_4), \ldots (v_{n-1}, b)$$

such that, for every edge e_k after the first, the first element of e_k is the second element of e_{k-1}.

Define the relation:

$$Reachable_G = \{(a, b) : a = b \text{ or there exists a path from } a \text{ to } b \text{ in } G\}$$

In other words, b is ***reachable*** from a if and only if it is possible to get from a to b in 0 or more steps.

(Part 1) Which of the following statements is correct:

a) $Reachable_G$ must be reflexive.
b) $Reachable_G$ must be antireflexive.
c) $Reachable_G$ must be neither reflexive nor antireflexive.
d) We can't make any claim about the reflexivity of $Reachable_G$.

(Part 2) Which of the following statements is correct:

a) $Reachable_G$ must be symmetric but not antisymmetric.
b) $Reachable_G$ must be antisymmetric but not symmetric.
c) $Reachable_G$ must be both symmetric and antisymmetric.
d) $Reachable_G$ must be neither symmetric nor antisymmetric.
e) We can't make any claim about the symmetry of $Reachable_G$.

(Part 3) Which of the following statements is correct:

a) $Reachable_G$ must be transitive.
b) We can't make any claim about the transitivity of $Reachable_G$.

7. Imagine this sensible definition of what *other* people can see your Facebook posts:

> Only your friends.

Of course, we also assume that you can view your own posts. Let F be the set of Facebook users. Define the relation:

> $\mathit{VisibleTo} \subseteq F \times F = \{(a, b)$: User a's posts are visible to user $b\}$

(Part 1) Which of the following statements is correct?

a) *VisibleTo* must be reflexive.
b) *VisibleTo* must be antireflexive.
c) *VisibleTo* must be neither reflexive nor antireflexive.
d) We can't make any claim about the reflexivity of *VisibleTo*.

(Part 2) Which of the following statements is correct?

a) *VisibleTo* must be symmetric but not antisymmetric.
b) *VisibleTo* must be antisymmetric but not symmetric.
c) *VisibleTo* must be both symmetric and antisymmetric.
d) *VisibleTo* must be neither symmetric nor antisymmetric.
e) We can't make any claim about the symmetry of *VisibleTo*.

(Part 3) Which of the following statements is correct?

a) *VisibleTo* must be transitive.
b) *VisibleTo* cannot be transitive.
c) We can't make any claim about the transitivity of *VisibleTo*.

8. Consider the classic children's two-person game, Rock Paper Scissors. At each play, the two players simultaneously choose one of rock, paper, or scissors.

If the two players choose the same move, the play is a tie and doesn't count. If they choose different moves, the rule for determining who wins is given by:

- Rock smashes scissors.
- Scissors cut paper.
- Paper covers rock.

So the following relation *Beats* contains one element for each way that $player_1$ can beat $player_2$:

$$Beats \subseteq \{\text{rock, paper, scissors}\} \times \{\text{rock, paper, scissors}\} =$$

$$\{(\text{rock, scissors}), (\text{scissors, paper}), (\text{paper, rock})\}$$

Read (rock, scissors) $\in$ *Beats* as saying that rock beats scissors.

Mark each of the following claims as true or false:

a) *Beats* is reflexive.
b) *Beats* is antireflexive.
c) *Beats* is symmetric.
d) *Beats* is antisymmetric.
e) *Beats* is transitive.

Note one interesting thing about *Beats*: $Beats^3 = I$. (In other words, apply it three times and you'll be back where you started.)

Patterns of Property Combinations

The three properties of reflexivity, symmetry, and transitivity are almost logically independent of each other. We can find simple, potentially useful relations that possess seven of the eight possible combinations of these properties:

Properties	***Domain***	***Example***
None	People	*Mother-of*
Just reflexive	People who can see	*WouldRecognizePictureOf*
Just symmetric	People	*FacebookFriendOf*
Just transitive	People	*Ancestor-of*
Just reflexive and symmetric	People	Facebook posts visible to
Just reflexive and transitive	Numbers	$\leq$
Just symmetric and transitive	Anything	$\varnothing$
All three	Almost anything	=

To see why we can't find a nontrivial (i.e., different from $\varnothing$) example of a relation that is symmetric and transitive but not reflexive, consider a simple relation R on {1, 2, 3, 4}. As soon as R contains a single element that relates two unequal objects (e.g., (1, 2)), it must, for symmetry, contain the matching element (2, 1). So now we have $R' = \{(1, 2), (2, 1)\}$. To make R' transitive, we must add (1, 1) and (2, 2). Call the resulting relation R''. But then R'' would also be reflexive, except that neither 3 nor 4 is related to itself. In fact, they are related to nothing. We cannot find an example of a relation R that is symmetric and transitive but not reflexive if we insist that all elements of the domain be related under R to something.

Proving Claims About Properties of Relations

How can we prove that some relation possesses some property that we care about?

Answer: A relation is a set (of tuples). So we prove properties of relations in the same way we prove properties of other kinds of sets:

To prove that relation R has some property:

1. Let e be an *arbitrary* element of R.
2. Write down what we know about e.
3. Use whatever else we're given and derive other facts about e until we've proved some relevant claim about e.
4. Generalize from e to *any* element of R.

Along the way, we can use other proof techniques as appropriate. For example, to prove that some claim P is true if and only if some other claim Q is true, we do two proofs:

P: Prove that $P \rightarrow Q$.
Q: Prove that $Q \rightarrow P$.

And recall that the usual way to prove that two sets A and B are equal is to do two proofs:

A: Every element of A must be in B.
B: Every element of B must be in A.

Let's put all of these general ideas together to prove a (possibly obvious) claim about symmetric relation s.

Prove: R is symmetric if and only if $R = R^{-1}$.

We must do two proofs (since we must prove both directions of this claim).

P: Prove that, if R is symmetric, then $R = R^{-1}$.

To prove that $R = R^{-1}$, we must prove that every element of R is in R^{-1} and that every element of R^{-1} is in R:

A: Every element of R is in R^{-1}:

Let (a, b) be an arbitrary element of R.
(b, a) is also in R, since R is symmetric.
(a, b) is in R^{-1}, by the definition of inverse.
Every element (x, y) in R is also in R^{-1}.

B: Every element of R^{-1} is in R:

Let (a, b) be an arbitrary element of R^{-1}.
(b, a) is in $(R^{-1})^{-1} = R$, by the definition of inverse.
(a, b) is also in R, since R is symmetric.
Every element (x, y) in R^{-1} is also in R.

Q: Prove that, if $R = R^{-1}$, then R is symmetric.

Let (a, b) be an arbitrary element of R.
(a, b) is in R^{-1} since $R = R^{-1}$.
(b, a) is in $(R^{-1})^{-1} = R$, by the definition of inverse.
If any element (x, y) is in R, then (y, x) is also in R.
R is symmetric (by definition).

Once people get facile with these kinds of proofs, they don't write them out in this much detail. But it will be good for us to practice doing it this way for a while.

Equivalence Relations

What Is an Equivalence Relation?

An ***equivalence relation*** on a set *A* is a relation that is:

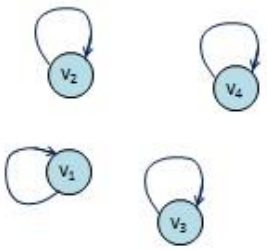
- reflexive,
- symmetric, and
- transitive.

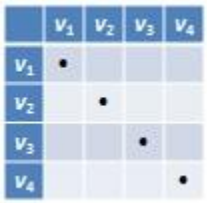
The most fundamental equivalence relation (on almost any set) is equality (=).

$7 = 7$	$7 \neq 8$
True = *True*	*True* ≠ *False*
$\{1, 2\} = \{2, 1\}$	$\{1, 2\} \neq \{1\}$
"abc" = "abc"	"abc" ≠ "cba"
$1 \bmod 4 = 5 \bmod 4$	$3 \bmod 4 \neq 5 \bmod 4$

While we can, of course, define = on some set *A* to mean whatever we want it to mean, in most common cases, = is exactly the Identity relation, *I*. An element $a \in A$ is equal to itself and nothing else.

Recall that we can represent *I* as either a graph or an adjacency matrix. When we do that, we get structures that look like this:

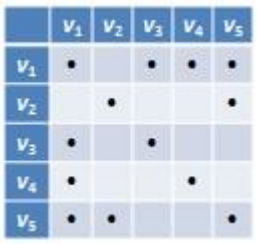




There are also many other useful equivalence relations. They are all, of course (because they are reflexive), supersets of *I*.

Problems

1. Consider this adjacency matrix:

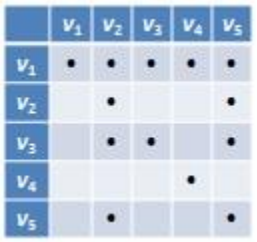


	v_1	v_2	v_3	v_4	v_5
v_1	•		•	•	•
v_2		•			•
v_3	•		•		
v_4	•			•	
v_5	•	•			•

Which of the following statements is true of the relation that it represents:

a) It is an equivalence relation.
b) It is not an equivalence relation because, while it is reflexive and transitive, it is not symmetric.
c) It is not an equivalence relation because, while it is symmetric and transitive, it is not reflexive.
d) It is not an equivalence relation because, while it is reflexive and symmetric, it is not transitive.
e) It is not an equivalence relation because it is missing at least two of the required properties.

2. Consider this adjacency matrix:



	v_1	v_2	v_3	v_4	v_5
v_1	•	•	•	•	•
v_2		•			•
v_3		•	•		•
v_4				•	
v_5		•			•

Which of the following statements is true of the relation that it represents:

a) It is an equivalence relation.
b) It is not an equivalence relation because, while it is reflexive and transitive, it is not symmetric.
c) It is not an equivalence relation because, while it is symmetric and transitive, it is not reflexive.
d) It is not an equivalence relation because, while it is reflexive and symmetric, it is not transitive.
e) It is not an equivalence relation because it is missing at least two of the required properties.

Examples of Equivalence Relations

Equivalence relations show up in a wide range of problem contexts.

Recall the family of relations $\equiv_k$ (defined on the natural numbers and based on the notion of modular arithmetic that we described several slides ago. For any integer k, $\equiv_k$ is an equivalence relation because it is:

- reflexive: $i \equiv_k i$, since there is a single remainder when i is divided by k.
- symmetric: if $i \equiv_k j$, then i and j have the same remainder when divided by k, so $j \equiv_k i$.
- transitive: if $i \equiv_k j$, then we have that, for some t, $i \bmod k = t$ and $j \bmod k = t$. Then, if $j \equiv_k n$, it must also be true that $n \bmod k = t$. So $i \bmod k = t = n \bmod k$. Thus, $i \equiv_k n$.

Consider again our company's personnel data base. Assume that it contains the following relation:

$Works\text{-}in \subseteq EmployeeIDs \times Departments =$

$\{(e, d)$: employee with id number e works in department $d\}$

Using that information, we could define the following relation:

$WorksInSameDepartmentAs \subseteq Employee \times Employee =$

$\{(e, f) : \exists d\ ((e, d) \in Works\text{-}In \wedge (f, d) \in Works\text{-}In)\}$

Assume that each employee works in exactly one department. Then *WorksInSameDepartmentAs* is an equivalence relation because it is:

- reflexive: Everyone works in the same department as him/herself.
- symmetric: If e and f work in some department d, then both (e, f) and (f, e) are in *WorksInSameDepartmentAs*. If there is no such department d, then neither (e, f) nor (f, e) is in *WorksInSameDepartmentAs*.
- transitive: Suppose that e and f work in the same department. Call it d. Suppose f and g also work in the same department. Since f only works in one department (d), g must also work in d. So e and g work in the same department.

Problems

1. Consider the relation *SiblingOf* $\subseteq$ *People* $\times$ *People* =

$$\{(a, b) : SisterOf(a, b) \vee BrotherOf(a, b)\}$$

Which of the following statements is true of *SiblingOf*:

a) It is an equivalence relation.
b) It is not an equivalence relation because, while it is reflexive and transitive, it is not symmetric.
c) It is not an equivalence relation because, while it is symmetric and transitive, it is not reflexive.
d) It is not an equivalence relation because, while it is reflexive and symmetric, it is not transitive.
e) It is not an equivalence relation because it is missing at least two of the required properties.

2. Let's return to the relation *WorksInSameDepartmentAs*. But now let's assume that it is possible for a single employee to work in more than one department at a time. Which of the following statements is true of *WorksInSameDepartment* in this case:

a) It is an equivalence relation.
b) It is not an equivalence relation because, while it is reflexive and transitive, it is not symmetric.
c) It is not an equivalence relation because, while it is symmetric and transitive, it is not reflexive.
d) It is not an equivalence relation because, while it is reflexive and symmetric, it is not transitive.
e) It is not an equivalence relation because it is missing at least two of the required properties.

3. Define the relation *FBFriendOf* $\subseteq$ *People* $\times$ *People* =

$$\{(a, b) : a \text{ is a Facebook friend of } b\}$$

Which of the following statements is true of *FBFriendOf*:

a) It is an equivalence relation.
b) It is not an equivalence relation because, while it is reflexive and transitive, it is not symmetric.
c) It is not an equivalence relation because, while it is symmetric and transitive, it is not reflexive.
d) It is not an equivalence relation because, while it is reflexive and symmetric, it is not transitive.
e) It is not an equivalence relation because it is missing at least two of the required properties.

4. Consider the large corporate database relation *SupervisorOf* $\subseteq$ *People* $\times$ *People* =

{(a, b) : a is the supervisor of b}

Which of the following statements is true of *SupervisorOf*:

a) It is an equivalence relation.
b) It is not an equivalence relation because, while it is reflexive and transitive, it is not symmetric.
c) It is not an equivalence relation because, while it is symmetric and transitive, it is not reflexive.
d) It is not an equivalence relation because, while it is reflexive and symmetric, it is not transitive.
e) It is not an equivalence relation because it is missing at least two of the required properties.

5. Define the relation *CommonFactor* as:

CommonFactor = {(i, j) $\in$ $\mathbf{Z} \times \mathbf{Z}$: i and j share a common integer factor}

For example, (12, 9) $\in$ *CommonFactor* since both 12 and 9 have 3 as a factor.

Which of the following statements is true of *CommonFactor*:

a) It is an equivalence relation.
b) It is not an equivalence relation because, while it is reflexive and transitive, it is not symmetric.
c) It is not an equivalence relation because, while it is symmetric and transitive, it is not reflexive.
d) It is not an equivalence relation because, while it is reflexive and symmetric, it is not transitive.
e) It is not an equivalence relation because it is missing at least two of the required properties.

6. Imagine that we are UPS. We need to pack as many boxes as possible into each truck. Assume that we don't worry about keeping things right side up. Also assume that all boxes are rectangles and that we measure them to the nearest inch. We describe each box as a three tuple:

(length, width, height)

Recall that $\mathbf{Z}^+$ is the set of positive integers. Then we might define the relation that contains all pairs of boxes of equal volume:

EQVolume $\subseteq (\mathbf{Z}^+ \times \mathbf{Z}^+ \times \mathbf{Z}^+) \times (\mathbf{Z}^+ \times \mathbf{Z}^+ \times \mathbf{Z}^+) = \{((a, b, c), (i, j, k)) : a \cdot b \cdot c = i \cdot j \cdot k\}$

Which of the following statements is true of *EQVolume*:

a) It is an equivalence relation.
b) It is not an equivalence relation because, while it is reflexive and transitive, it is not symmetric.
c) It is not an equivalence relation because, while it is symmetric and transitive, it is not reflexive.
d) It is not an equivalence relation because, while it is reflexive and symmetric, it is not transitive.
e) It is not an equivalence relation because it is missing at least two of the required properties.

Partitions

Recall the following definition: Given a nonempty set A, a ***partition*** of A is a set of nonempty, pairwise disjoint subsets of A whose union is A.

Another way to state this definition is to say that Π, a set of subsets of A, is a partition of A if and only if:

- No element of Π is empty: $\forall X \in \Pi\ (X \neq \varnothing)$.
- No elements of Π overlap: $\forall X, Y \in \Pi\ (X \cap Y = \varnothing)$.
- No element of A is left out of Π:

$$\bigcup_{X \in \Pi} X = A$$

Consider the set **Z** (the integers). Here's one partition Π of **Z**:

Π = {{even integers}, {odd integers}}

Neither set is empty. And every integer is in exactly one element of Π.

We must be careful, when we assert that some set Π is a partition of a set A, that all three of the required properties hold.

Again consider the set **Z** (the integers). Here's another proposed partition Π of **Z**:

Π = {{integers divisible by 2}, {integers divisible by 3}}

This time, Π is not a partition of **Z**. It has two problems. Some elements of **Z** (for example, 6) are in both elements of Π. And some other elements of **Z** (for example, 7) are in neither element of Π.

Problems

1. Let P be the set of people. For each of the following, sets S, mark true if S is a partition of P. Mark false otherwise.

a) {{residents of North America}, {residents of South America}, {residents of Europe}, {residents of Asia}, {residents of Africa}}
b) {{people like beets}, {people who like okra}, {people who like chocolate}, {people who like pears}}
c) {{people born in a leap year}, {people born in an odd numbered year}, {people born in an even numbered year}}
d) {{high school students} {high school graduates}, {high school dropouts}}
e) {{eldest girl in family}, {eldest boy in family}, {has/had older sibling}}

2. Let $\mathbf{Z}$ be the set of integers. For each of the following sets S, mark true if S is a partition of $\mathbf{Z}$. Mark false otherwise:

a) $\{\{x : x \equiv_3 0\}, \{x : x \equiv_3 1\}, \{x : x \equiv_3 2\}\}$
b) $\{\{x : x \equiv_4 0\}, \{x : x \equiv_4 1\}, \{x : x \equiv_4 2\}\}$
c) {{0}, {negative integers}, {positive integers}}
d) {∅, {prime numbers}, {composite numbers}}

Equivalence Classes

The word "equivalence" suggests equality. And, as we've seen, = (in the usual sense) is an equivalence relation.

But, in particular problem contexts, it may be useful to group together objects that, while not identical, share some property that is key in the current context. Notice that a partition (as just defined) is one way to carve up a set into subsets. So let's explore the relationship between equivalence relations and partitions.

Let A be a nonempty set and let R be an equivalence relation on A. Now consider an arbitrary element x of A. Define the ***equivalence class*** containing x as:

$$\{y \in A : (x, y) \in R\}$$

In other words, the equivalence class that contains x contains all and only those elements of A that are related to x via R. Note that, since R is an equivalence relation, it is reflexive. So x is guaranteed to be related to itself (and thus to be in this set).

We'll say that an equivalence class, as just defined, is ***induced*** by R.

Common notations for equivalence classes are:

$[x]_R$	(the class that contains x, in case we need to be explicit about R), or
$[x]$	(the class that contains x in case R is clear from context)
[description]	(the class that contains exactly the elements that satisfy description)

Notice that, in general, there may be different ways to describe the same equivalence class.

Let $A = \{1, 2, 3, 4, 5\}$. Let $R =$

$\{(1, 1), (1, 3), (1, 5), (2, 2), (2, 4), (3, 1), (3, 3), (3, 5), (4, 2), (4, 4), (5, 1), (5, 3), (5, 5)\}$

Then [1] = [3] = [5] = [odd positive integers less than 6] = $\{1, 3, 5\}$.

Now consider the set of equivalence classes induced by R. Using the equivalence relation properties of reflexivity, symmetry and transitivity, it is possible to prove that this set is a partition. In other words, it carves up A into Π, a set of nonempty subsets in such a way that every element of A is in exactly one subset. We'll say then that R induces Π.

So now we have that equivalence relations do something very useful. Each equivalence relation induces a partition on its domain. Interesting equivalence relations induce partitions that carve sets into equivalence classes with the property that elements in the same class are "equivalent" in some useful way.

Consider the equivalence relation $\equiv_{12}$. So we have, for example:

$7 \equiv_{12} 19 \equiv_{12} 31 \equiv_{12} 103$

The equivalence classes induced by $\equiv_{12}$ correspond to the positions of the hour hand on a traditional clock.

Problems

1. Assume that we roll a pair of dice and care only about the total of the two dice. We'll define *EqualDiceRoll* to relate pairs of dice values with each other. So it is a subset of *Rolls* =

$$\{(\{1, 2, 3, 4, 5, 6\} \times \{1, 2, 3, 4, 5, 6\}) \times (\{1, 2, 3, 4, 5, 6\} \times \{1, 2, 3, 4, 5, 6\})\}$$

EqualDiceRoll = $\{((i, j), (n, m)) : i + j = n + m\}$

EqualDiceRoll is an equivalence relation that induces a partition on *Rolls*. For many games, this partition captures exactly the property of a dice roll that we care about. Indicate whether each of the following statements is true or false:

a) $(4, 5) \in [(3, 6)]$
b) $[(2, 3)] \subseteq [(4, 1)]$
c) |[pairs that total 2]| = |[pairs that total 12]|
d) $[(1, 3)] \in [(2, 2)]$
e) $[(2, 6)] = [(6, 2)]$
f) $(6, 6) \in [(1, 1)]$

Equivalence Classes and Partitions

The equivalence relation properties of reflexivity, symmetry and transitivity are what guarantee that Π is a partition. Let's show that it must have the three required properties of a partition:

- No element of Π is empty: If $X \in \Pi$, then, for some element x of A:

  $$X = \{y \in A : (x, y) \in R\}$$

  Since R is reflexive, X must contain x. So it isn't empty.

- No elements of Π overlap: Consider two arbitrary elements of Π:

  $X = \{y \in A : (x, y) \in R\}$ (all elements related to x)
  $Z = \{y \in A : (z, y) \in R\}$ (all elements related to z)

  We need to show that the only way for X and Z to overlap is for them to be equal:

  Suppose that X and Z do overlap and there is some element a that is in both of them. In other words, a is related to both x and z. Formally, $(x, a) \in R$ and $(z, a) \in R$. Since R is symmetric, we have that $(a, z) \in R$. But now, since R is transitive, we have that $(x, z) \in R$.

  Using this fact, we'll show that X and Z are the same. Often to do this, we do two proofs: One shows that every element of X is in Z. The other goes the other way. In this case, we can collapse the two proofs because every step is reversible. We'll show equivalent expressions.

  Let p be an arbitrary element of the domain A. Then:

	$p \in X$	assumption for conditional proof
$\equiv$	$(x, p) \in R$	definition of X
$\equiv$	$(p, x) \in R$	since R is symmetric
$\equiv$	$(p, z) \in R$	since $(x, z) \in R$ (see above) and R is transitive
$\equiv$	$(z, p) \in R$	since R is symmetric
$\equiv$	$p \in Z$	definition of Z

  So being in X is equivalent to being in Z. The two sets are equal.

- Every element of A is in some element of Π: Let a be an arbitrary element of A. Then one element of Π must be the equivalence class that contains all and only the elements related to a. Since a is related to itself, it is in that set. It is thus in some element of Π.

Problems

1. Imagine that we are UPS. We need to pack as many boxes as possible into each truck. Assume that we don't worry about keeping things right side up. Also assume that all boxes are rectangles and that we measure them to the nearest inch. We describe each box as a three tuple:

(length, width, height)

Recall that $\mathbf{Z}^+$ is the set of positive integers. We have defined a relation that contains all pairs of boxes of equal volume:

$$EQVolume \subseteq (\mathbf{Z}^+ \times \mathbf{Z}^+ \times \mathbf{Z}^+) \times (\mathbf{Z}^+ \times \mathbf{Z}^+ \times \mathbf{Z}^+) = \{((a, b, c), (i, j, k)) : a \cdot b \cdot c = i \cdot j \cdot k\}$$

EQVolume is an equivalence relation. So it partitions boxes into equivalence classes. All boxes in a single class have the same volume.

For each of the following pairs, mark true if both are in the same equivalence class and false otherwise:

a) (2, 6, 7), (7, 3, 4)
b) (3, 7, 11), (11, 4, 6)
c) (2, 6, 9), (3, 3, 12)

2. WordNet® is a large lexical database of English. It contains entries for nouns, verbs, adjectives, and adverbs. It maps each word to one or more meanings. For example, the noun "party" could mean a social event or a political organization.

You can play with WordNet online (or download it if you want to use it in a program). You'll find it here: http://wordnet.princeton.edu/ .

Synonyms (words that share a meaning) are grouped into sets called synsets. Wordnet has about 117,000 synsets. Here are a couple of them:

[cab, hack, taxi, taxicab]
[café, coffeehouse, coffee shop, coffee bar]

Let's consider the question, "Do WordNet's synsets form a partition on English words and their meanings?" Put another way, is the relation "*InSameSynsetAs*" an equivalence relation that partitions words into equivalence classes such that all the elements of a single class share a meaning?"

For the rest of this discussion, we'll focus only on the noun meanings of the words we consider. You will need to look words up in Wordnet in order to answer these questions.

Question 1: Do Wordnet's synsets form a partition of the set of English *words*? Recall that, if *Synsets* is a partition, then every English word must occur in exactly one set.

(Part 1) To how many synsets does the English word "table" belong?

(Part 2) To how many synsets does the English word "chair" belong?

(Part 3) Is the claim that Wordnet's synsets form a partition of the set of English words true or false?

Question 2: Do Wordnet's synsets form a partition of the set of English word *meanings* (senses)? In other words, we begin by acknowledging that a given word (string of letters) may have more than one meaning. But now consider the set *Meanings*, which contains word meanings, as distinct from the word(s) we use to describe those meanings. Is *Synsets* a partition of *Meanings*?

(Part 4) Wordnet assigns the English word "cloth" a single meaning (i.e., a single synset). How many other words also have the meaning that corresponds to that synset?

(Part 5) Check each of the words you found in the last question. Considering only the "cloth" meaning of each word, is it the case that the meaning is assigned to a single synset (i.e., the one that contains the meaning of "cloth")? Mark yes or no.

Try this same exercise with a few more words and see what you observe.

(Part 6) Now suppose that we know that there are no empty synsets (why would they have bothered to create any?). And we know that every sense that Wordnet knows has been assigned to at least one synset. Then, on the basis of what we have observed, does it appear that *Synsets* is a partition of *Meanings*? (In other words, is every meaning in exactly one synset?) Mark yes or no.

Proving that a Relation is an Equivalence Relation

To prove that a relation R is an equivalence relation, we must show that it has the three required properties:

- symmetry
- reflexivity
- transitivity

Sometimes this is difficult. But in many cases our proof can piggyback on the fact that = is an equivalence relation. Many relations are actually = in disguise.

Make the assumption that everyone lives in exactly one place. Now consider the relation $LivesAtSameAddressAs \subseteq People \times People$ =

$\{(a, b) : a$ and b live at the same address$\}$

An alternative way to say this is that a's address equals b's address. So we have that *LivesAtSameAddressAs* is an equivalence relation:

- Reflexivity: Person a's address equals itself since equals is reflexive.
- Symmetry: If a's address equals b's address, then b's address equals a's address since equals is symmetric.
- Transitivity: If a's address equals b's address and b's address equals c's address, then a's address equals c's address, since equals is transitive.

But, of course, we don't always get off this easily.

Define: $R \subseteq \mathbf{Z} \times \mathbf{Z} = \{(x, y) : x + y \text{ is even}\}$.

R is an equivalence relation:

- Reflexivity: For all x, $x + x = 2x$ and thus is even. Thus $(x, x) \in R$.
- Symmetry: For all x and y, $x + y = y + x$.(since addition is commutative). So if $(x, y) \in R$, so is (y, x).
- Transitivity: For any x, y, and z: Assume that (x, y) and (y, z) are both in R. Then $x + y$ is even and $y + z$ is even. So there exist integers j and k such that:

[1]	$x + y = 2j$	
[2]	$y + z = 2k$	
[3]	$x + z + 2y = 2j + 2k$	Adding [1] and [2]
[4]	$x + z = 2j + 2k - 2y$	
[5]	$x + z = 2(j + k - y)$	

Since $j + k - y$ is an integer, $x + z$ is even and thus $(x, z) \in R$ if (x, y) and (y, z) are.

Soundex (Again)

So far, we've looked at a lot of simple examples of equivalence relations. But the power of the idea is that equivalence relations, and the partitions that they induce, are very useful when we're dealing with many kinds of complex, real world problems.

Recall the Soundex system, which assigns four-symbol codes to names in such a way that names that sound "similar" will get the same code. Once one has the code for one's name, one can run the system backwards to retrieve other names that map to the same code.

https://www.youtube.com/watch?v=V6cFEJsqQ40

Here's the conversion algorithm again, or you can review the video:

1. If two or more adjacent letters (including the first) would map to the same number, remove all but the first. Include as "adjacent" letters that are separated just by H's and W's.
2. Set the first letter of the code to the first letter of the name.
3. For all other letters of the name do:
    3. Convert the letters B, P, F, V, C, S, G, J, K, Q, X, Z, D, T, L, M, N, and R to numbers and tack the resulting number onto code (on the right):
        B, P, F, V = 1
        C, S, G, J, K, Q, X, Z = 2
        D, T = 3
        L = 4
        M, N = 5
        R = 6
    4. Delete all instances of the letters A, E, I, O, U, Y, H, and W.
4. If the code string contains more than three numbers, delete all but the leftmost three.
5. If the code string contains fewer than three numbers, pad with 0's on the right to get three.

When we introduced Soundex, we saw that we could use our relation notation to describe what it does. In particular, we could describe what it does as the composition of a relation with its inverse. We wrote:

$SoundexCode \subseteq Names \times Codes = \{(name, c) : c \text{ is the code produced by Soundex for } name\}$

$SoundexEQ \subseteq Names \times Names = SoundexCode^{-1} \circ SoundexCode$

We can now observe that *SoundexEQ* is an equivalence relation. It induces a partition on the set of names. Pairs of names in the same equivalence class "sound similar". So for example, here are some of its equivalence classes (using the Soundex code as the name for the equivalence class that contains all the names that map to that code):

[F640] contains such elements as Fairley, Farrell, Friel, Farwell and Frailey.

[M450] contains such elements as Mellon, Malone, Milne, Maloney, and Moylan.

[K460] contains such elements as Kaylor, Keller, Kyler, Kelleher, and Keeler.

Problems

1. You can compute Soundex codes here:

http://resources.rootsweb.ancestry.com/cgi-bin/soundexconverter

Using the Soundex code as the name for the equivalence class that contains all the names the map to that code, to what equivalence class does the name Miller belong? (Enter a four-character code.)

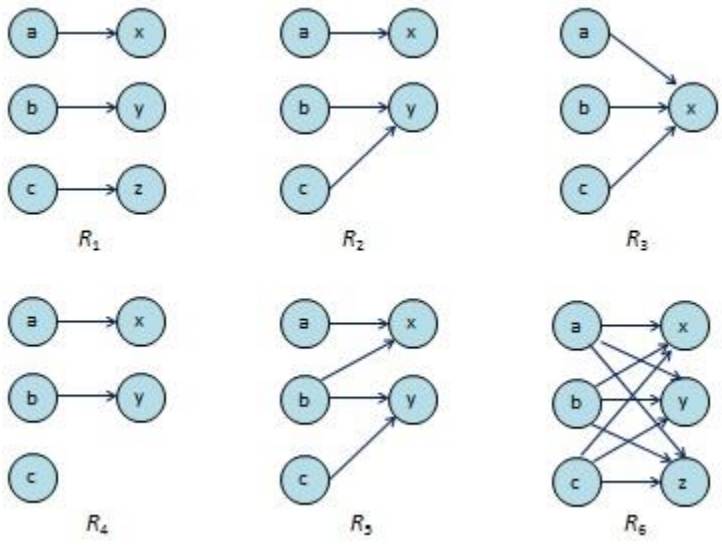
2. To show that *SoundexEQ* is an equivalence relation, we must show that it is:

- reflexive,
- symmetric, and
- transitive.

Consider each of these properties and argue to yourself that *SoundexEQ* possesses it. You can make use of the fact that *SoundexEQ* is the composition of *SoundexCode* and its inverse.

To get started, let's look at the more general question of when the composition of a relation with its inverse has the required properties.

(Part 1) Reflexivity: Consider the following relations:



Mark each of the following claims as true or false. Then try to generalize: When is the composition of a relation with its inverse reflexive?

a) $R_1^{-1} \circ R_1$ is reflexive.
b) $R_2^{-1} \circ R_2$ is reflexive.
c) $R_3^{-1} \circ R_3$ is reflexive.
d) $R_4^{-1} \circ R_4$ is reflexive.
e) $R_5^{-1} \circ R_5$ is reflexive.
f) $R_6^{-1} \circ R_6$ is reflexive.

(Part 2) Symmetry: Consider the same set of relations: Mark each of the following claims as True or False. Then try to generalize: When is the composition of a relation with its inverse symmetric?

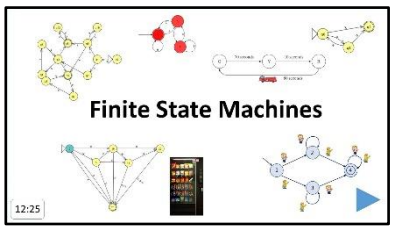
a) $R_1^{-1} \circ R_1$ is symmetric.
b) $R_2^{-1} \circ R_2$ is symmetric.
c) $R_3^{-1} \circ R_3$ is symmetric.
d) $R_4^{-1} \circ R_4$ is symmetric.
e) $R_5^{-1} \circ R_5$ is symmetric.
f) $R_6^{-1} \circ R_6$ is symmetric.

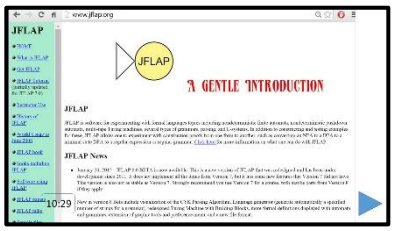
(Part 3) Transitivity: Consider the same set of relations: Mark each of the following claims as True or False. Then try to generalize: When is the composition of a relation with its inverse transitive?

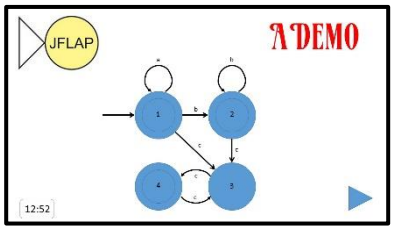
a) $R_1^{-1} \circ R_1$ is transitive.
b) $R_2^{-1} \circ R_2$ is transitive.
c) $R_3^{-1} \circ R_3$ is transitive.
d) $R_4^{-1} \circ R_4$ is transitive.
e) $R_5^{-1} \circ R_5$ is transitive.
f) $R_6^{-1} \circ R_6$ is transitive.

Finite State Machines (Again)

Recall that, when we first introduced the idea of a partition of a set, we mentioned finite state machines. You may want to review the videos that were presented there:







https://www.youtube.com/watch?v=lh2onWfBrxk
https://www.youtube.com/watch?v=irewHV3S0_M
https://www.youtube.com/watch?v=cEvWgEHd0pE

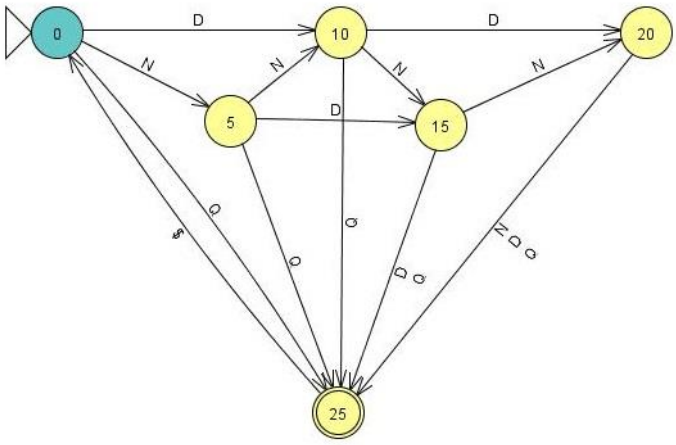
A finite state machine (or FSM) is a five-tuple:

- An alphabet of symbols that can be input to the machine. These symbols can be literal characters or they can correspond to events that could occur in the world.
- A set of states. In some applications, there is an output associated with each state.
- A set of transitions from one state to the next. Each is labeled with a symbol drawn from the alphabet. In some applications, there may also be an output associated with each transition.
- A start state (one of the states already mentioned).
- A set of accepting (often called "final") states. (All other states are nonaccepting/nonfinal.) Some applications make use of this idea of "acceptance".

An FSM begins in its start state. It consumes input symbols one at a time and follows its transitions to move from one state to the next. For some applications, the job of the machine is to generate outputs as this process proceeds. For other applications, the job of the machine is to decide whether or not an input string is acceptable. In that case, it will matter whether the machine lands in an accepting or a nonaccepting (rejecting) state.

A vending machine is a great example of an FSM. The simple one shown here has a four symbol alphabet:

- N (a nickel has been inserted)
- D (a dime has been inserted),
- Q (a quarter has been inserted), and
- B (the Buy button has been pushed).



The states correspond to the amount of money (regardless of the specific coins or their order) that has been input since the last sale.

The start state corresponds to 0¢. We'll assume that it costs 25¢ to buy something. So the 25¢ state is an accepting state. There (and nowhere else) can one push the Buy button. Notice that the machine shown here is "friendly"; it allows you to deposit more than 25¢ if you happen not to have the correct change. So you can push the Buy button whenever there's at least 25¢ in the machine. Whenever you do that, the machine resets to 0¢ (that transition isn't shown in the diagram).

An FSM *M* defines an equivalence relation on the set of possible input strings:

$string_1 \equiv_M string_2$ if and only if $string_1$ and $string_2$ drive *M* to the same state.

So the states of *M* correspond to the equivalence classes of $\equiv_M$.

In the vending machine example:

N D D ≡ D N D ≡ Q ≡ Q Q Q ≡ Q D D N Q

All of these inputs are in the equivalence class [25¢].

This equivalence relation is the key to the power of FSMs. It groups together inputs that share a future. The only thing that matters, in determining the future behavior of *M* is its current state. It doesn't matter how it got there.

Again, in the vending machine example:

If what has happened so far is any of N D D or D N D or Q or Q Q Q or Q D D N Q:

- If Q comes next, it is possible to buy a drink.
- If N comes next, the machine is 20¢ away from it being possible to buy a drink.
- And so forth.

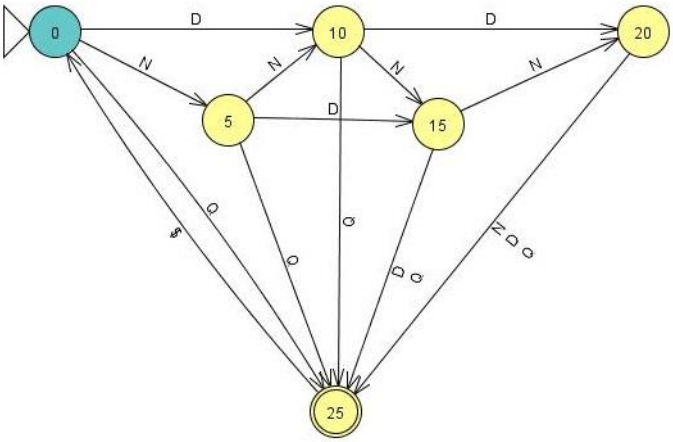
What this means is that, in a complex world, a simple finite state machine (with, of course a finite and often small amount of memory) can remember everything that matters. All the rest of the history information can be thrown away.

Another good example of an FSM is a traffic light. A simple American traffic light has three states, Green, Yellow and Red. (In other countries, there may be other states, such as Flashing Yellow.) There may also be more complex lights with more states that, for example, respond to time of day. The inputs that cause transitions from one state to the next include elapsed time and special signals from emergency vehicles or pedestrian walk buttons. In this example, there is output associated with the states (i.e., the color of the light). Again notice, that most history is thrown away. All that matters for what happens next is the current state.

Another example is a password checker. Many websites have rules for legal passwords. For example, they may require at least 6 characters, at least one of which is a number. An FSM can check a proposed new password to see if it satisfies the rules. Now the inputs are characters (letters, numbers, special symbols). The machine will begin checking the proposed password, reading the characters left to right. The states of the machine will record how many of each kind of character have been seen so far. Now we'll make use of the notion of an accepting state. Only those states that correspond to strings that satisfy the rules will be accepting. Notice that, as always, we throw away a lot of detail as we're processing an input. For example, the input strings "thinkso" and "maybeso" are equivalent since they both contain 7 lower case letters, no numbers, and no special characters. Thus, if the rule is 6 – 8 characters at least one of which is a number, then both of these will be accepted if exactly one character comes next and it is a number. Otherwise, both of them will end up being rejected.

Problems

1. Let *M* be the vending machine FSM that we have described:



Let $\equiv_M$ be the equivalence relation induced by *M*. Mark each of the following statements as true or false:

a) DDD $\equiv_M$ DDNN
b) QQQQ $\equiv_M$ QQQQQQQQ
c) QDN $\equiv_M$ NDQ
d) DDN $\equiv_M$ DNN