Relations and Functions

Recap from Last Time

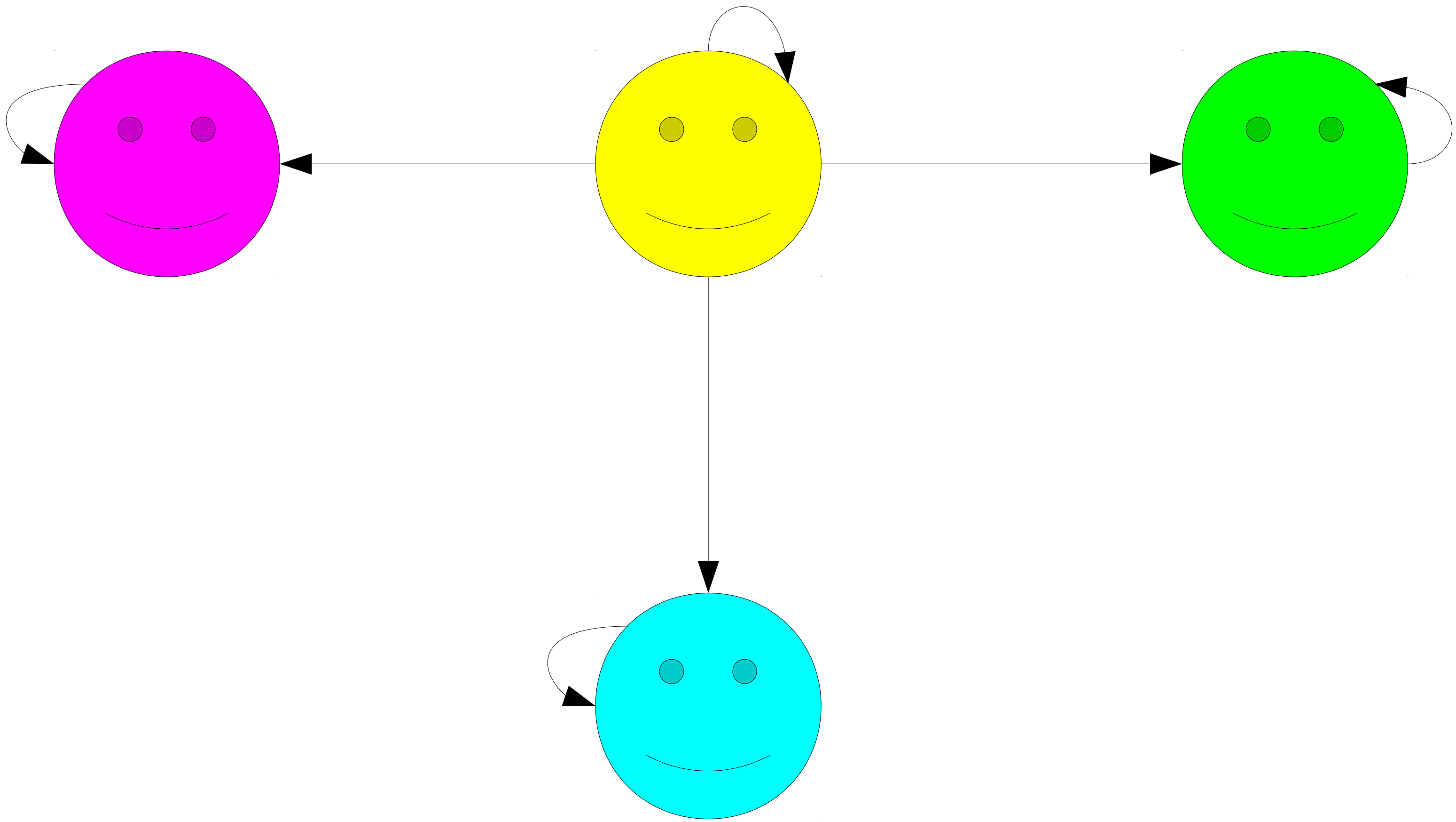
Reflexivity

- Some relations always hold from any element to itself.
- Examples:
  - $x = x$ for any x.
  - $A \subseteq A$ for any set A.
  - $x \equiv_k x$ for any x.
- Relations of this sort are called ***reflexive***.
- Formally speaking, a binary relation R over a set A is reflexive if the following is true:

$$\forall a \in A.\ aRa$$

("Every element is related to itself.")

Reflexivity Visualized



$$\forall a \in A.\ aRa$$

("Every element is related to itself.")

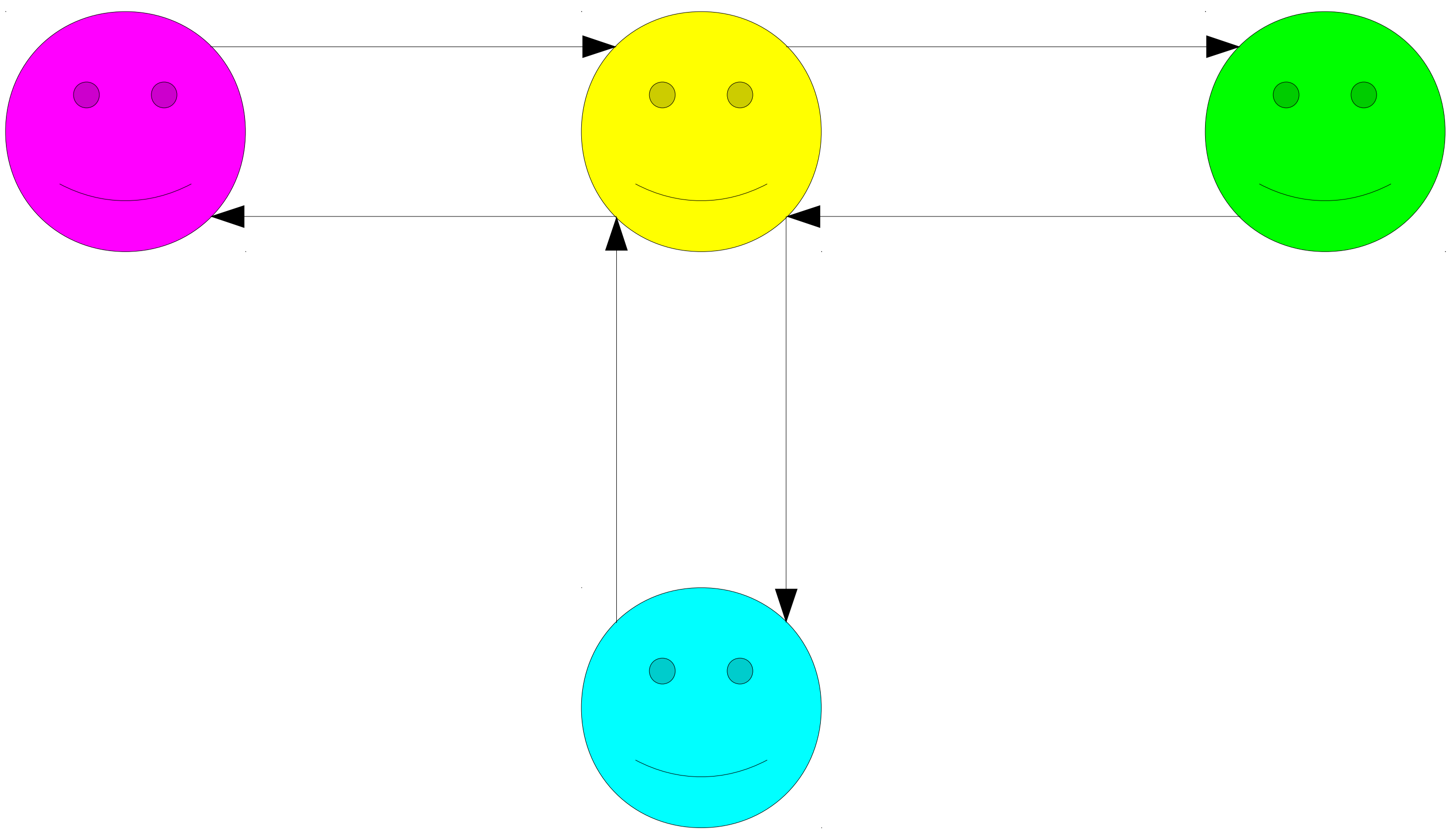
Symmetry

- In some relations, the relative order of the objects doesn't matter.
- Examples:
  - If $x = y$, then $y = x$.
  - If $x \equiv_k y$, then $y \equiv_k x$.
- These relations are called ***symmetric***.
- Formally: a binary relation R over a set A is called *symmetric* if

$$\forall a \in A.\ \forall b \in A.\ (aRb \to bRa)$$

("If a is related to b, then b is related to a.")

Symmetry Visualized



$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$

("If a is related to b, then b is related to a.")

$a \sim b$ if $a+b$ is even

Lemma 2: The binary relation $\sim$ is symmetric.

Proof: Consider any integers a and b where $a \sim b$. We need to show that $b \sim a$.

Since $a \sim b$, we know that $a+b$ is even. Because $a+b = b+a$, this means that $b+a$ is even. Since $b+a$ is even, we know that $b \sim a$, as required. ■

$$\forall a \in A.\ \forall b \in A.\ (aRb \rightarrow bRa)$$

(*"If a is related to b, then b is related to a."*)

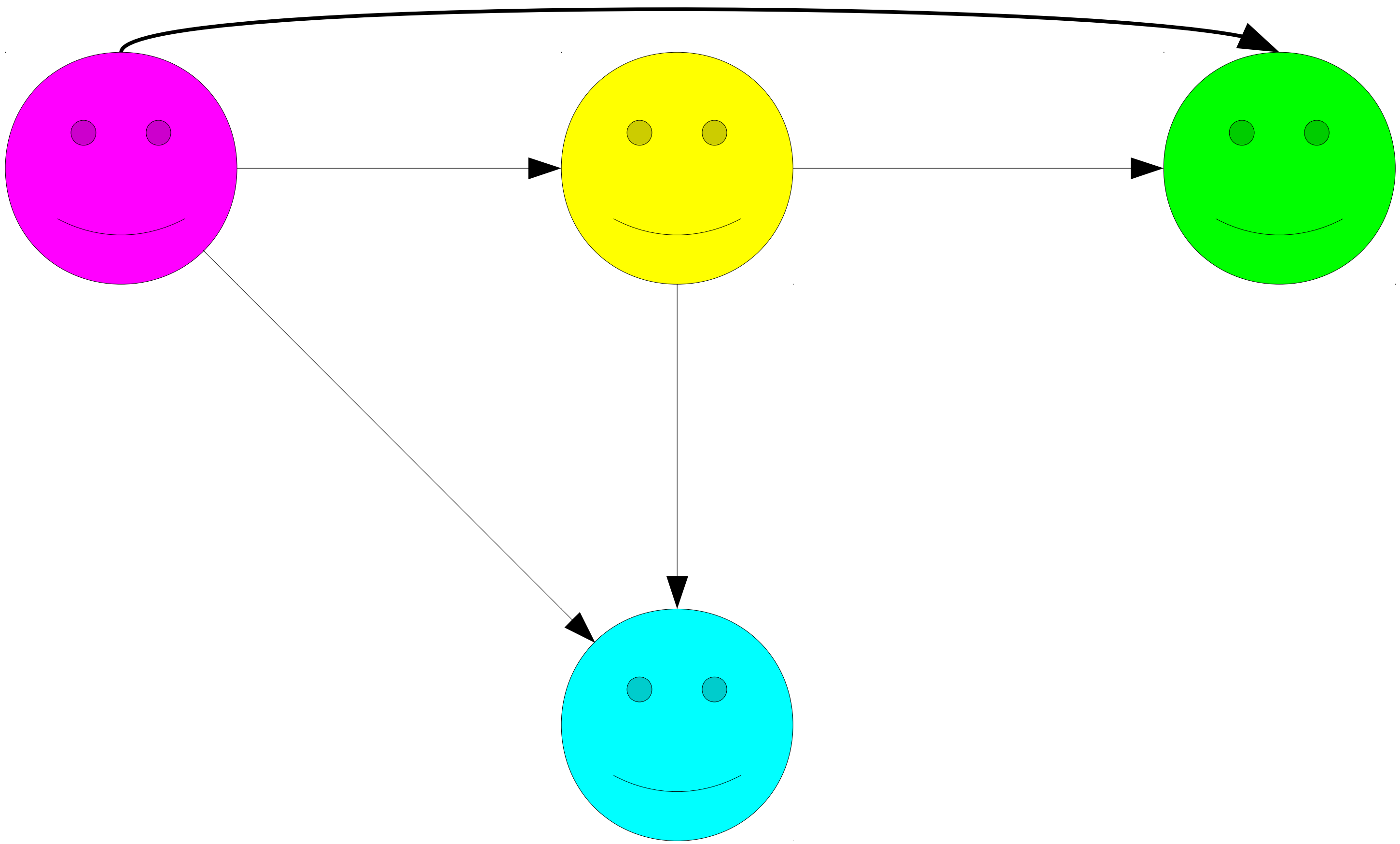
Transitivity

- Many relations can be chained together.
- Examples:
  - If $x = y$ and $y = z$, then $x = z$.
  - If $R \subseteq S$ and $S \subseteq T$, then $R \subseteq T$.
  - If $x \equiv_k y$ and $y \equiv_k z$, then $x \equiv_k z$.
- These relations are called ***transitive***.
- A binary relation R over a set A is called *transitive* if

$$\forall a \in A.\ \forall b \in A.\ \forall c \in A.\ (aRb \land bRc \to aRc)$$

(*"Whenever a is related to b and b is related to c, we know a is related to c.*)

Transitivity Visualized



$\forall a \in A. \forall b \in A. \forall c \in A. (aRb \land bRc \rightarrow aRc)$

("Whenever a is related to b and b is related to c, we know a is related to c.)

Equivalence Relations

- An ***equivalence relation*** is a relation that is reflexive, symmetric and transitive.
- Some examples:
  - $x = y$
  - $x \equiv_k y$
  - x has the same color as y
  - x has the same shape as y.

New Stuff!

Prerequisite Structures

Pancakes

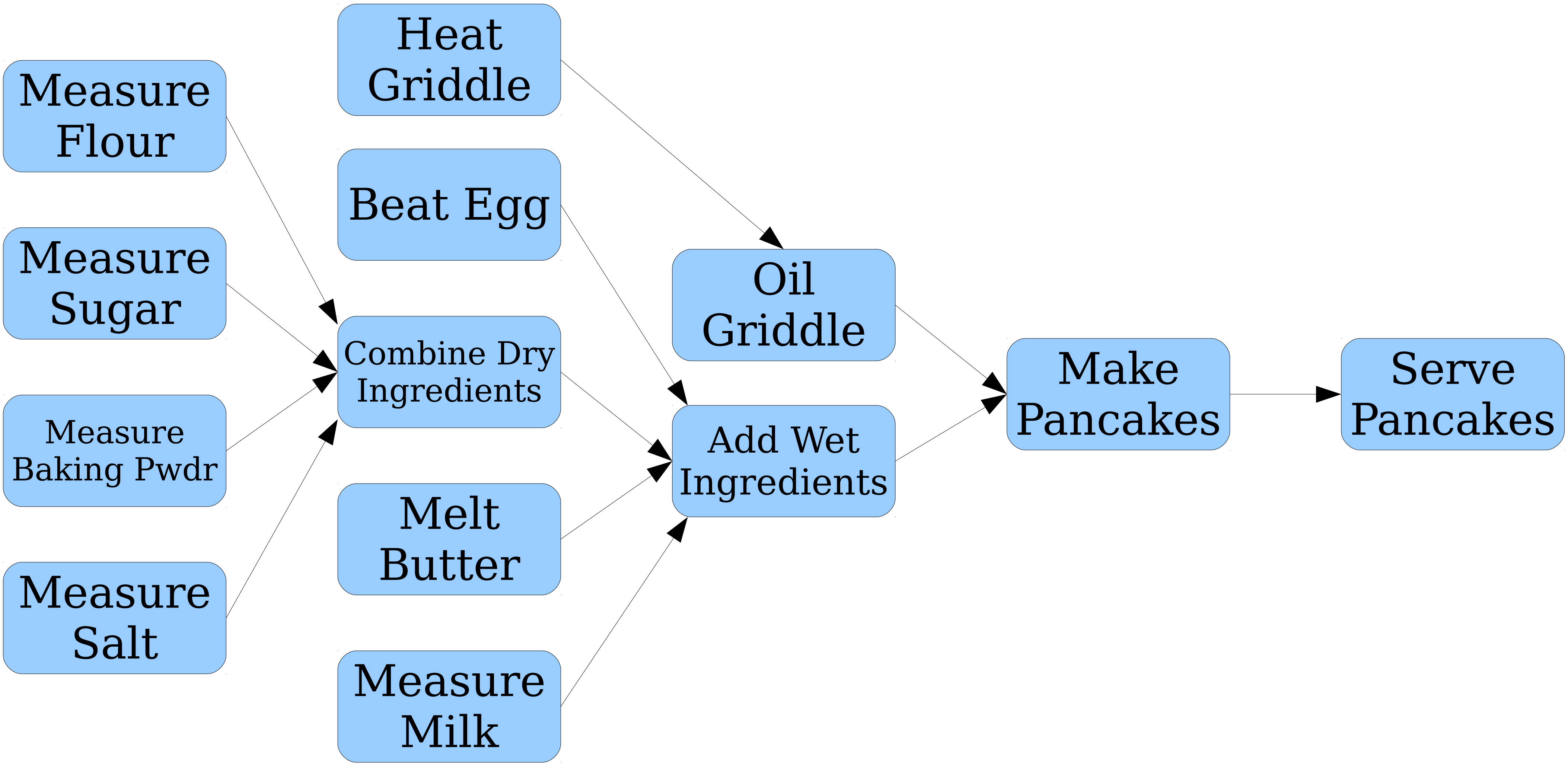
Everyone's got a pancake recipe. This one comes from Food Wishes (http://foodwishes.blogspot.com/2011/08/grandma-kellys-good-old-fashioned.html).

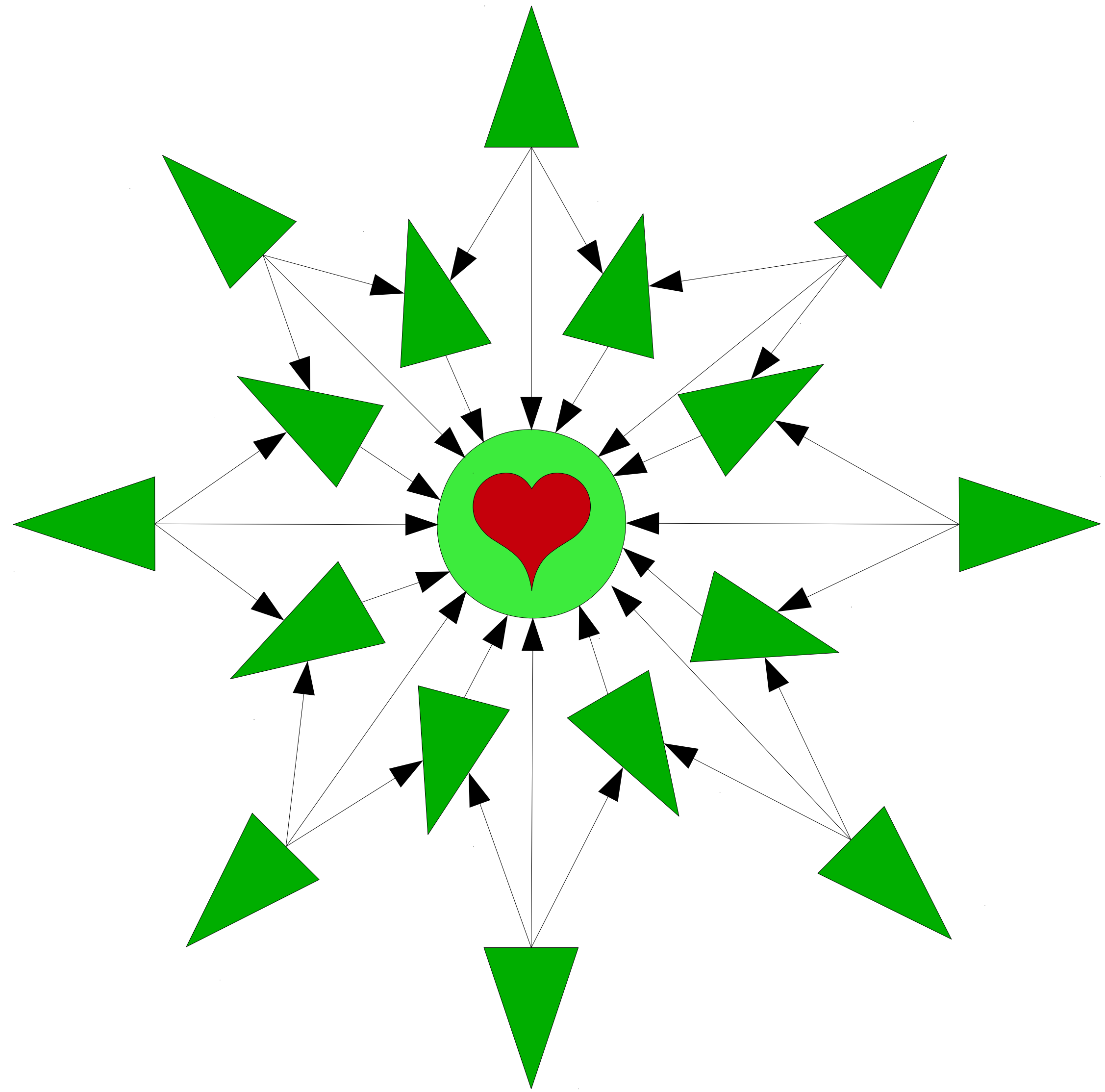
Ingredients

- 1 1/2 cups all-purpose flour
- 3 1/2 tsp baking powder
- 1 tsp salt
- 1 tbsp sugar
- 1 1/4 cup milk
- 1 egg
- 3 tbsp butter, melted

Directions

1. Sift the dry ingredients together.
2. Stir in the butter, egg, and milk. Whisk together to form the batter.
3. Heat a large pan or griddle on medium-high heat. Add some oil.
4. Make pancakes one at a time using 1/4 cup batter each. They're ready to flip when the centers of the pancakes start to bubble.

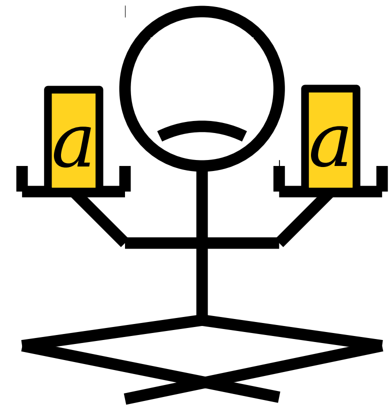

Measure Flour
Measure Sugar
Measure Baking Pwdr
Measure Salt
Heat Griddle
Beat Egg
Combine Dry Ingredients
Melt Butter
Measure Milk
Oil Griddle
Add Wet Ingredients
Make Pancakes
Serve Pancakes



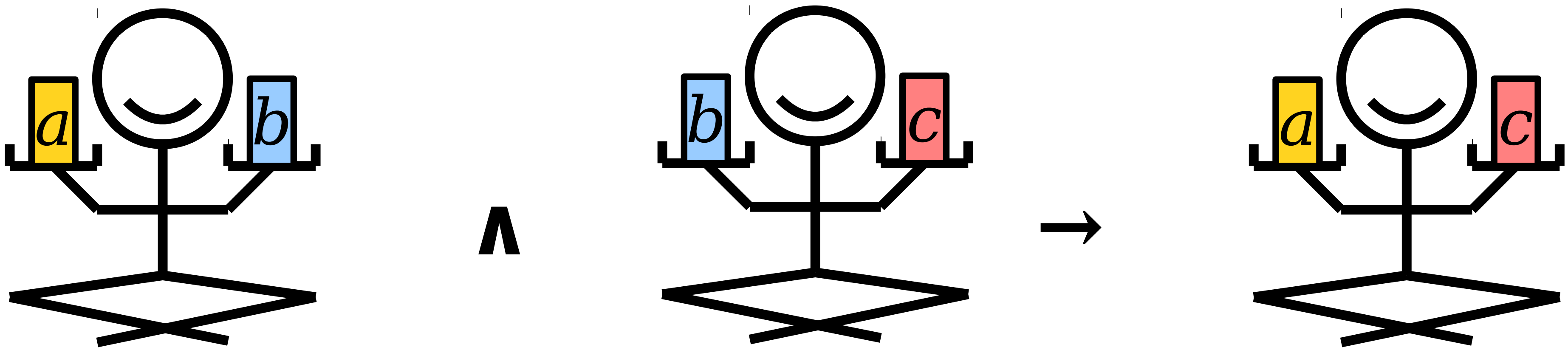
Relations and Prerequisites

- Let's imagine that we have a prerequisite structure with no circular dependencies.
- We can think about a binary relation R where aRb means

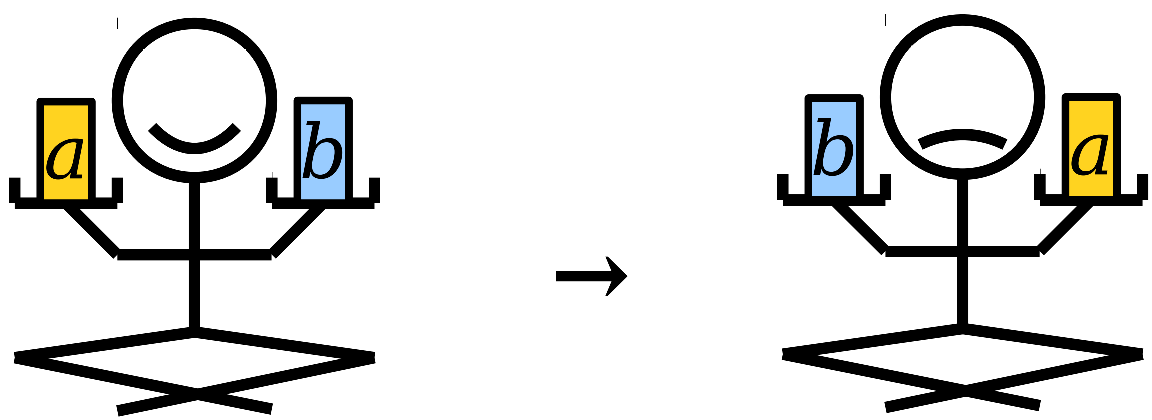
  "***a* must happen before *b***"
- What properties of R could we deduce just from this?



a
a



a
b
b
c
a
c



a
b
b
a

$$\forall a \in A.\ a \not R a$$

$$\forall a \in A.\ \forall b \in A.\ \forall c \in A.\ (aRb \land bRc \to aRc)$$

$$\forall a \in A.\ \forall b \in A.\ (aRb \to b \not R a)$$

$$\forall a \in A.\ a \cancel{R} a$$

Transitivity

$$\forall a \in A.\ \forall b \in A.\ (aRb \to b \cancel{R} a)$$

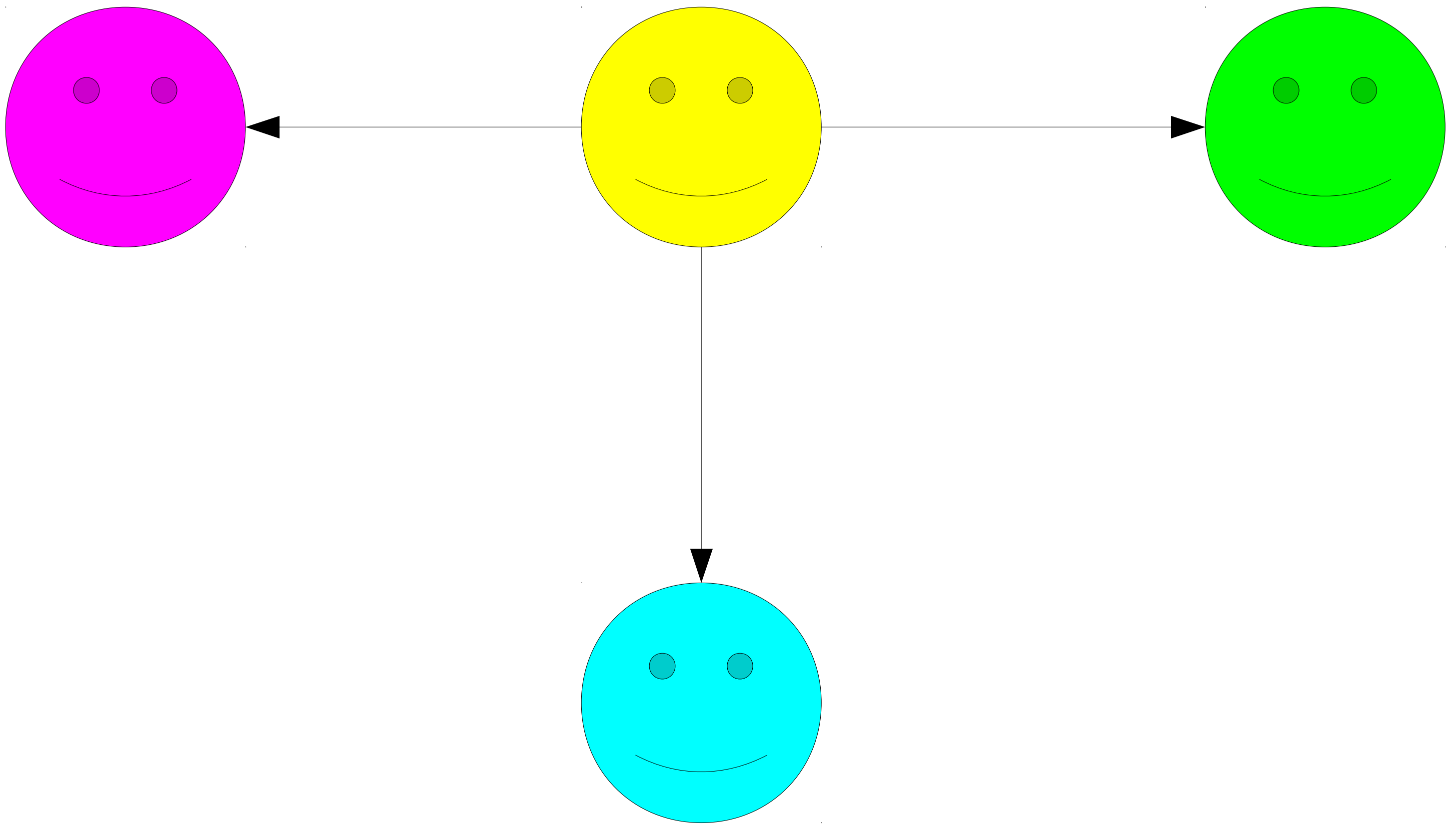
Irreflexivity

- Some relations *never* hold from any element to itself.
- As an example, $x \not< x$ for any x.
- Relations of this sort are called ***irreflexive***.
- Formally speaking, a binary relation R over a set A is ifreflexive if the following is true:

$$\forall a \in A.\ a \not R a$$

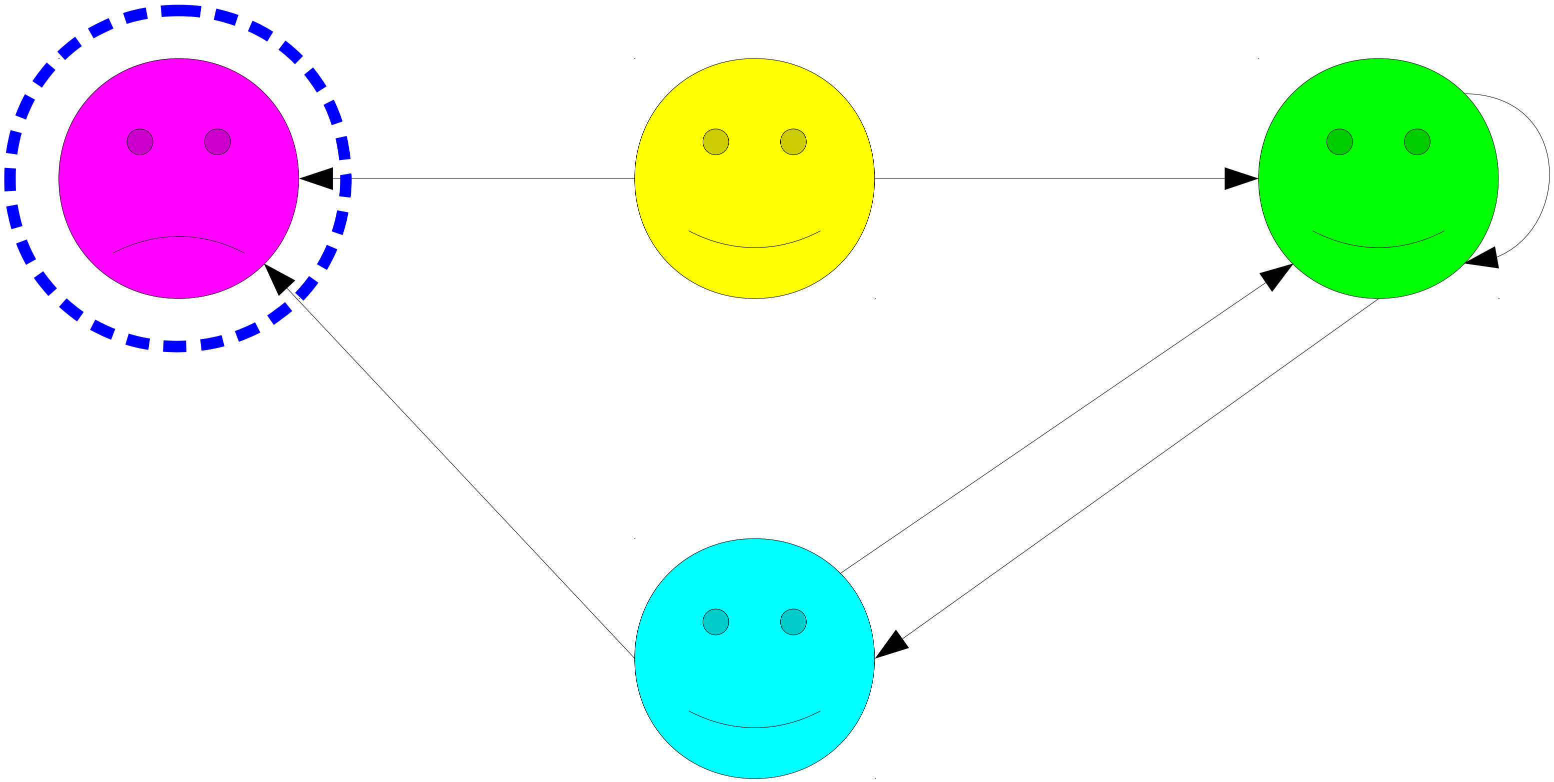
(*"No element is related to itself."*)

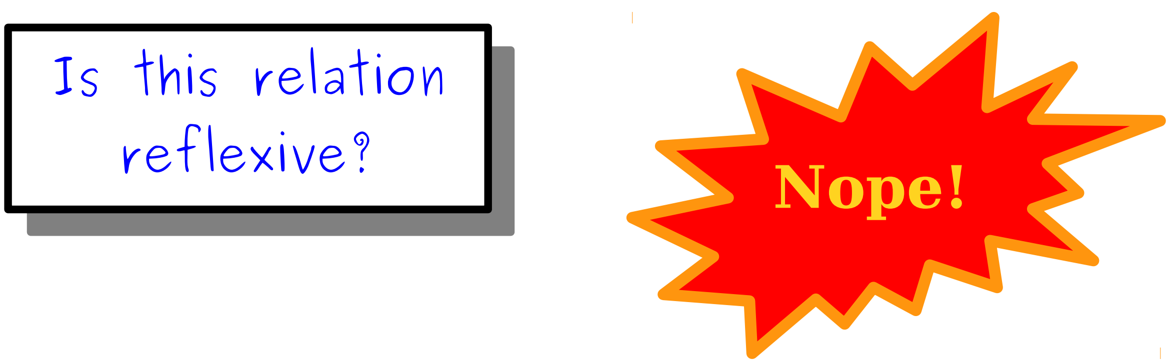
Irreflexivity Visualized



$$\forall a \in A.\ a \not R a$$

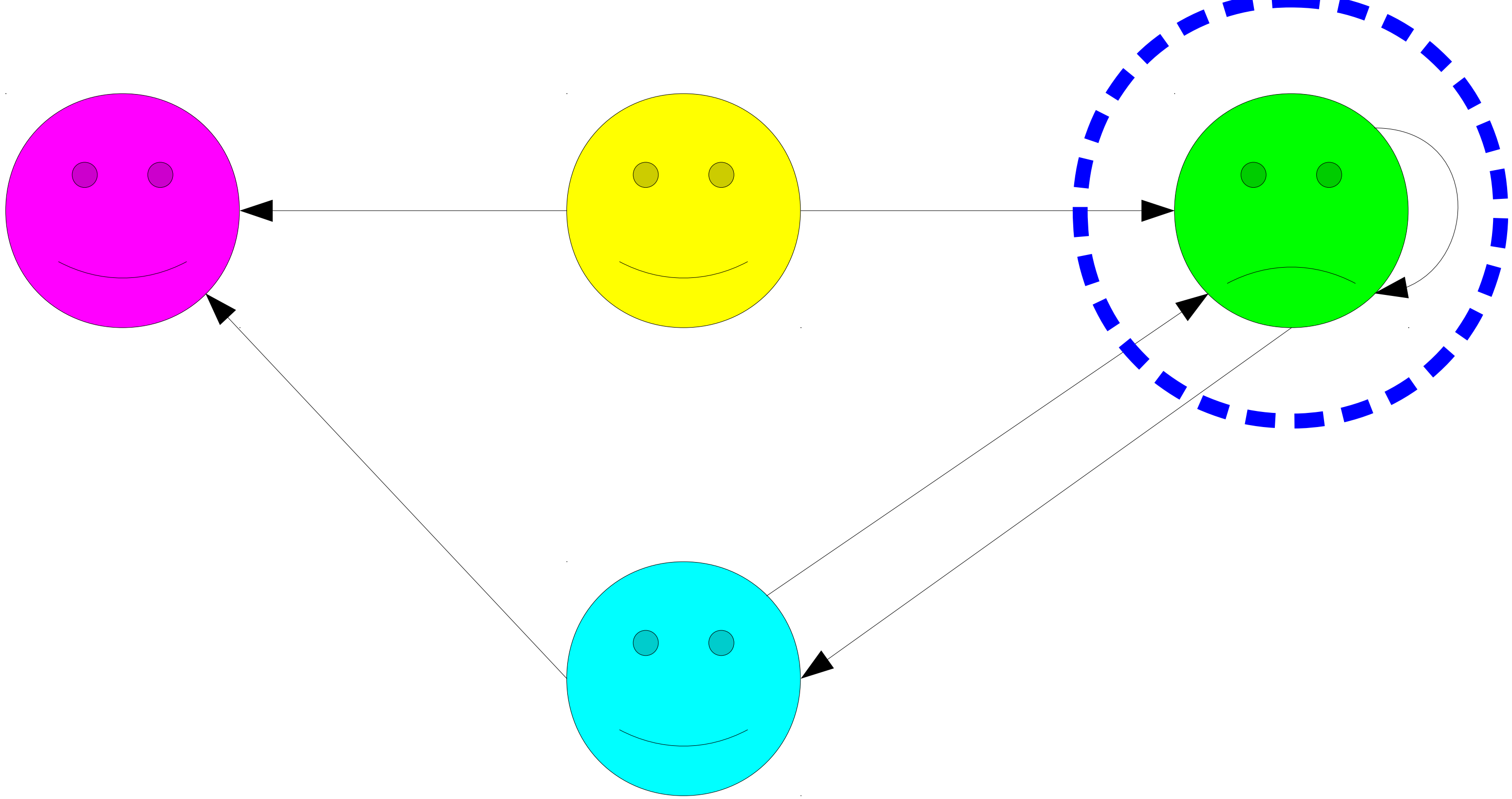
("No element is related to itself.")





$$\forall a \in A.\ aRa$$

(*"Every element is related to itself."*)



Is this relation irreflexive?

Nope!

$$\forall a \in A.\ a \not{R} a$$

(*“No element is related to itself.”*)

Reflexivity and Irreflexivity

- Reflexivity and irreflexivity are ***not*** opposites!
- Here's the definition of reflexivity:

$$\forall a \in A.\ aRa$$

- What is the negation of the above statement?

$$\exists a \in A.\ a\not{R}a$$

- What is the definition of irreflexivity?

$$\forall a \in A.\ a\not{R}a$$

Irreflexivity

Transitivity

$$\forall a \in A. \forall b \in A. (aRb \to b \not R a)$$

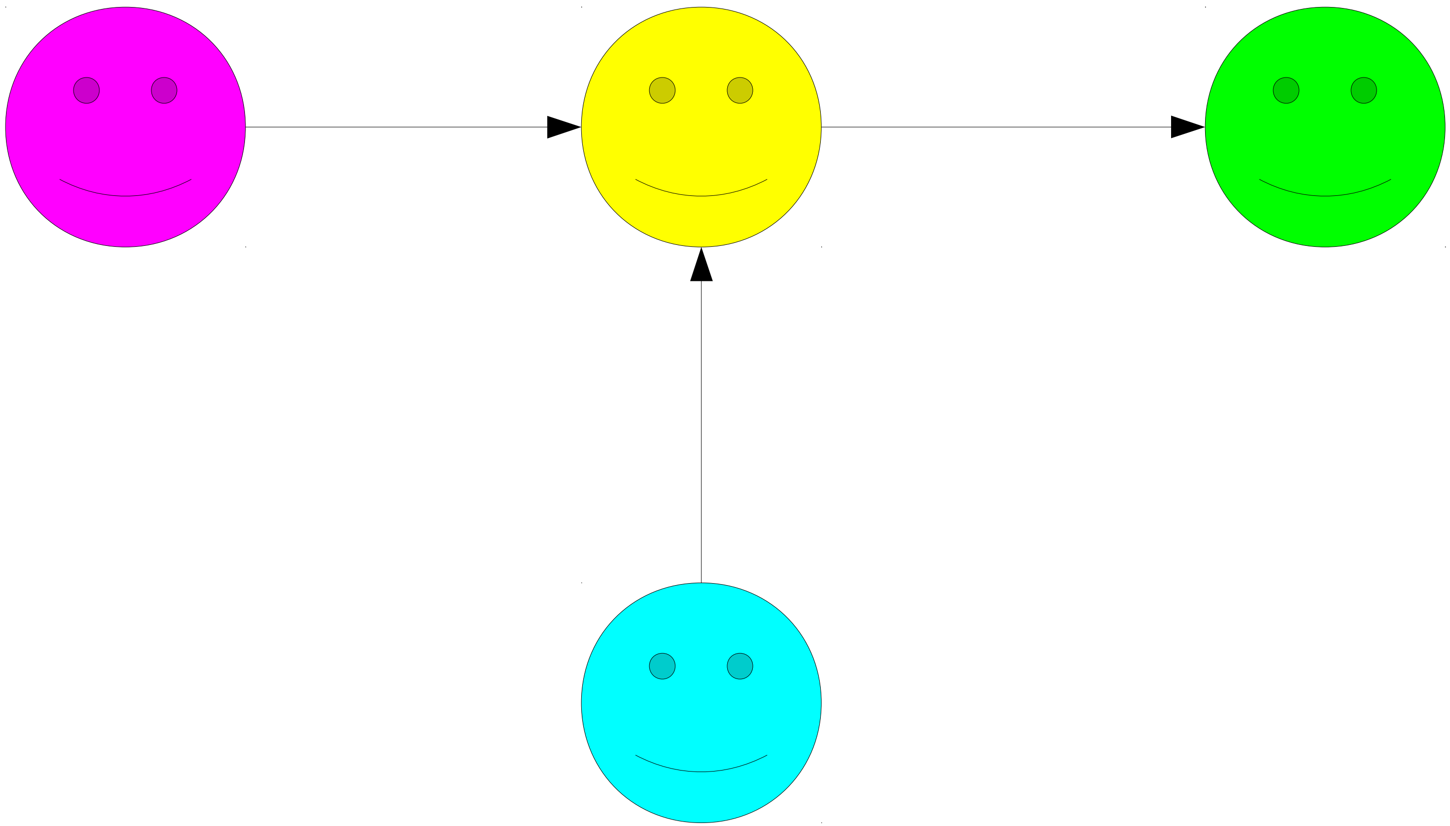
Asymmetry

- In some relations, the relative order of the objects can never be reversed.
- As an example, if $x < y$, then $y \not< x$.
- These relations are called ***asymmetric***.
- Formally: a binary relation R over a set A is called *asymmetric* if

$$\forall a \in A.\ \forall b \in A.\ (aRb \to b \not R a)$$

("If a relates to b, then b does not relate to a.")

Asymmetry Visualized



$$\forall a \in A.\ \forall b \in A.\ (aRb \rightarrow b \not R a)$$

(*"If a relates to b, then b does not relate to a."*)

Question to Ponder: Are symmetry and asymmetry opposites of one another?

Irreflexivity

Transitivity

Asymmetry

Strict Orders

- A ***strict order*** is a relation that is irreflexive, asymmetric and transitive.
- Some examples:
  - $x < y$.
  - a can run faster than b.
  - $A \subset B$ (that is, $A \subseteq B$ and $A \neq B$).
- Strict orders are useful for representing prerequisite structures and have applications in complexity theory (measuring notions of relative hardness) and algorithms (searching and sorting)

Strict Order Proofs

- Let's suppose that you're asked to prove that a binary relation is a strict order.
- Calling back to the definition, you could prove that the relation is asymmetric, irreflexive, and transitive.
- However, there's a slightly easier approach we can use instead.

Theorem: Let R be a binary relation over a set A. If R is asymmetric, then R is irreflexive.

Proof: Let R be an arbitrary asymmetric binary relation over a set A. We will prove that R is irreflexive.

What's the high-level structure of this proof?

$$\forall R.\ (\textit{Asymmetric}(R) \to \textit{Irreflexive}(R))$$

Therefore, we'll choose an arbitrary asymmetric relation R, then go and prove that R is irreflexive.

Theorem: Let R be a binary relation over a set A. If R is asymmetric, then R is irreflexive.

Proof: Let R be an arbitrary asymmetric binary relation over a set A. We will prove that R is irreflexive.

To do so, we will proceed by contradiction.

What is the definition of irreflexivity?

$$\forall x \in A.\ x \not{R} x$$

What is the negation of this statement?

$$\exists x \in A.\ xRx$$

So let's suppose that there is some element $x \in A$ such that xRx and proceed from there.

Theorem: Let R be a binary relation over a set A. If R is asymmetric, then R is irreflexive.

Proof: Let R be an arbitrary asymmetric binary relation over a set A. We will prove that R is irreflexive.

To do so, we will proceed by contradiction. Suppose that R is not irreflexive. That means that there must be some $x \in A$ such that xRx.

Since R is asymmetric, we know for any $a, b \in A$ that if aRb holds, then $b\not R a$ holds. Plugging in $a=x$ and $b=x$, we see that if xRx holds, then $x\not R x$ holds. We know by assumption that xRx is true, so we conclude that $x\not R x$ holds. However, this is impossible, since we can't have both xRx and $x\not R x$.

We have reached a contradiction, so our assumption must have been wrong. Thus R must be irreflexive. ■

Theorem: If a binary relation R is asymmetric and transitive, then R is a strict order.

Proof: Let R be a binary relation that is asymmetric and transitive. Since R is asymmetric, by our previous theorem we know that R is also irreflexive. Therefore, R is asymmetric, irreflexive, and transitive, so by definition R is a strict order. ■

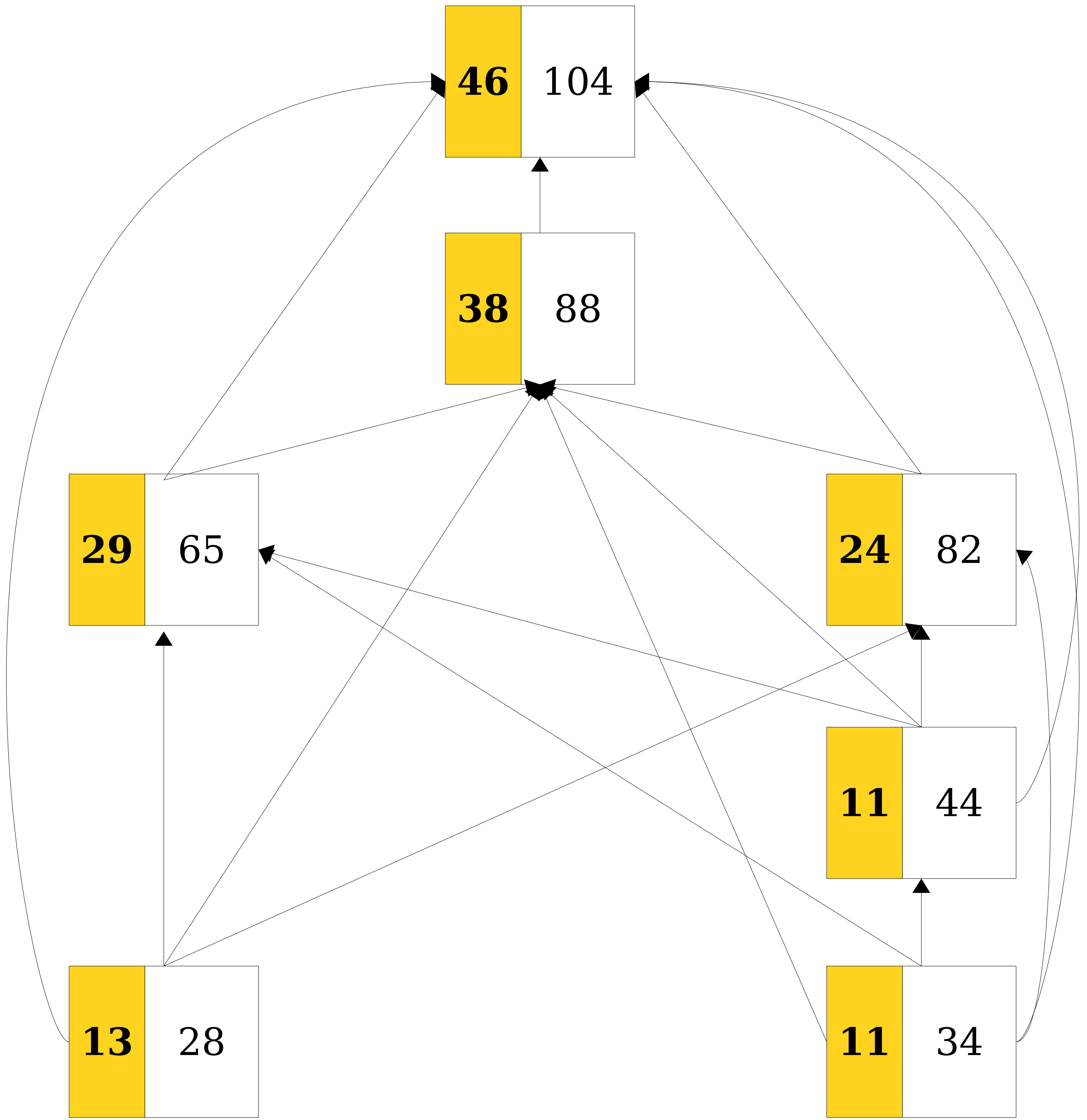
To prove that some binary relation R is a strict order, just prove that R is asymmetric and transitive.

Drawing Strict Orders

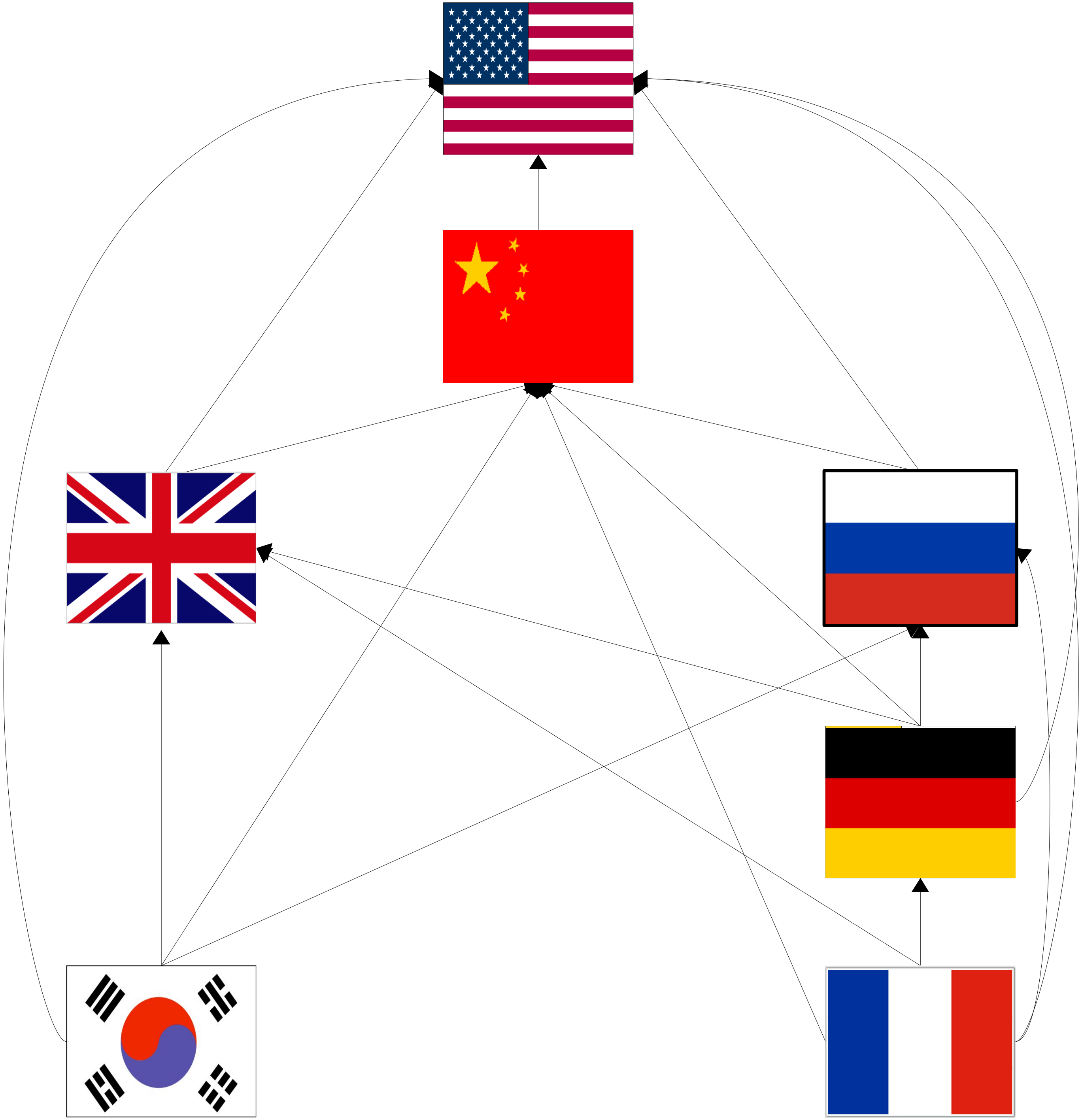
2012 Summer Olympics

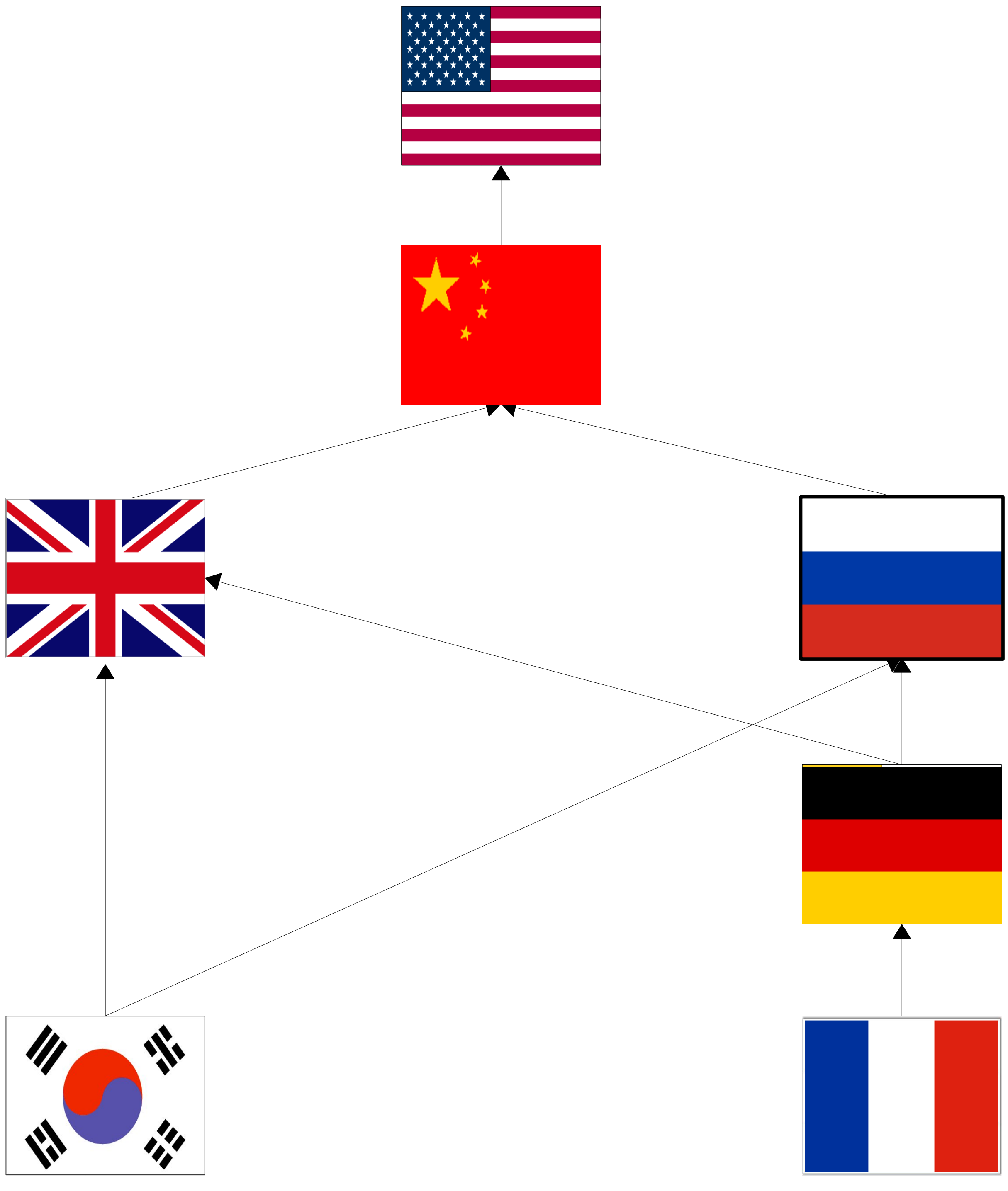
Gold	Silver	Bronze	Total
46	29	29	104
38	27	23	88
29	17	19	65
24	26	32	82
13	8	7	28
11	19	14	44
11	11	12	34

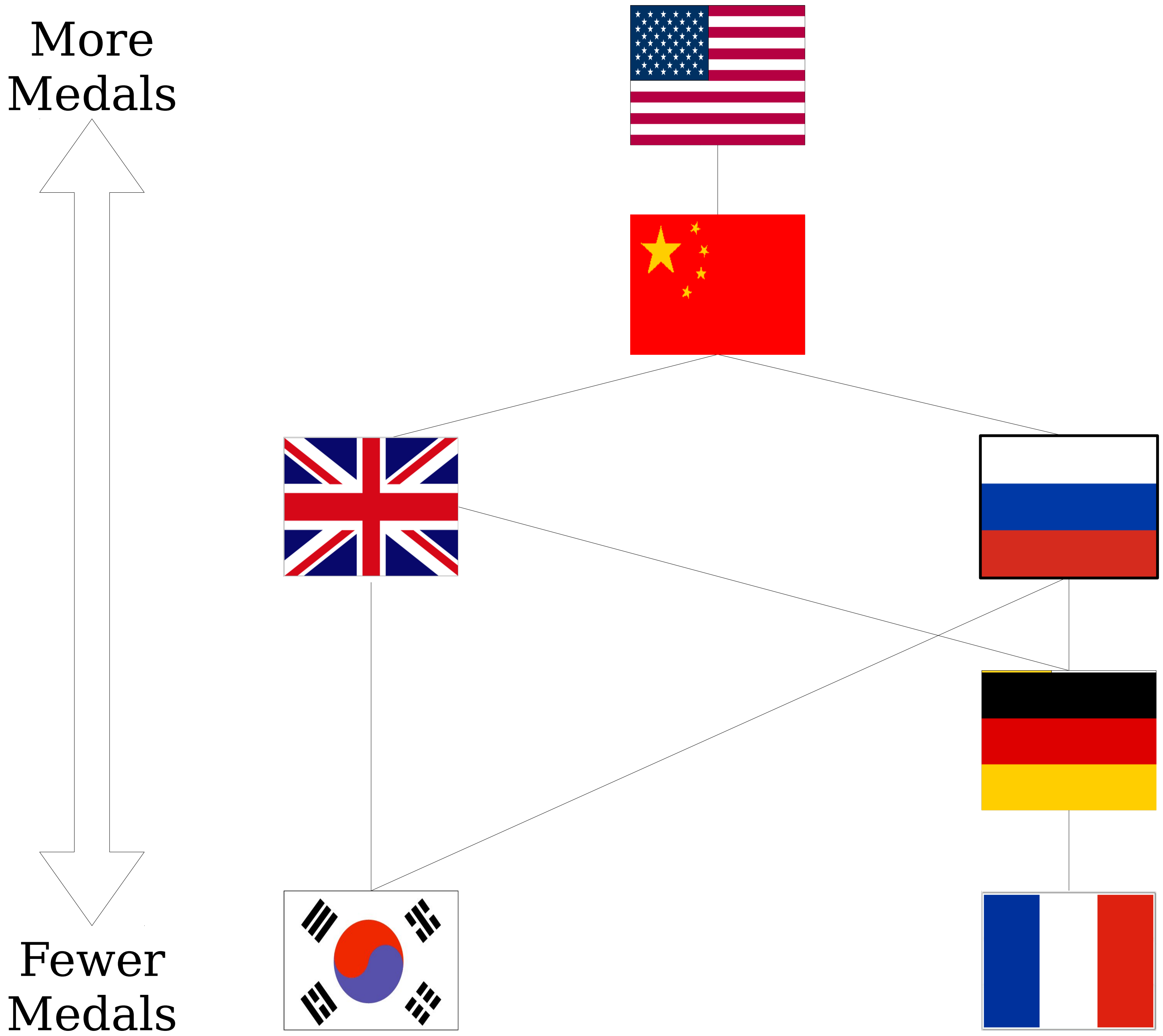
Inspired by http://tartarus.org/simon/2008-olympics-hasse/
Data from http://www.london2012.com/medals/medal-count/



46
104
38
88
29
65
24
82
11
44
13
28
11
34



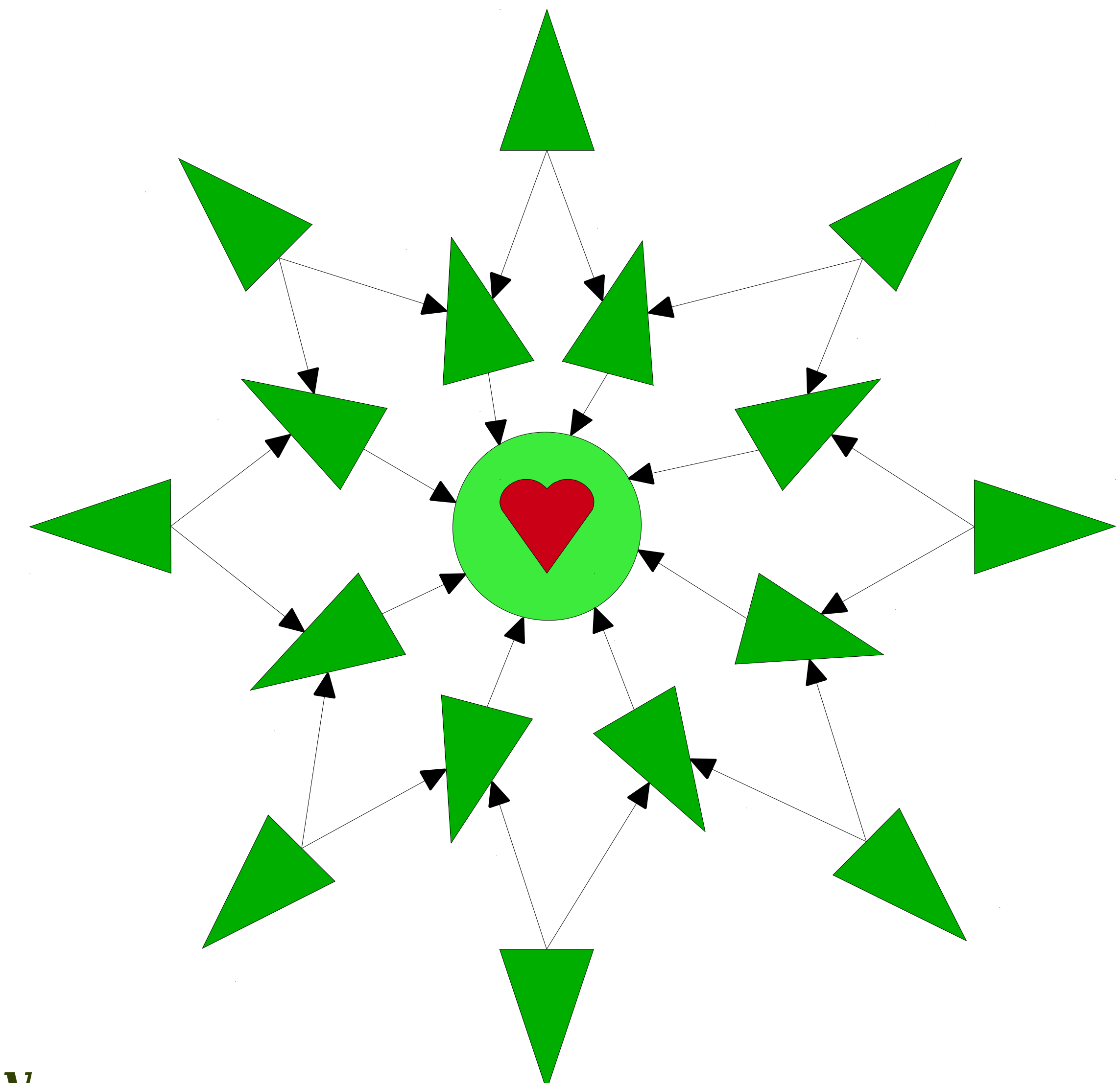



More Medals
Fewer Medals

Hasse Diagrams

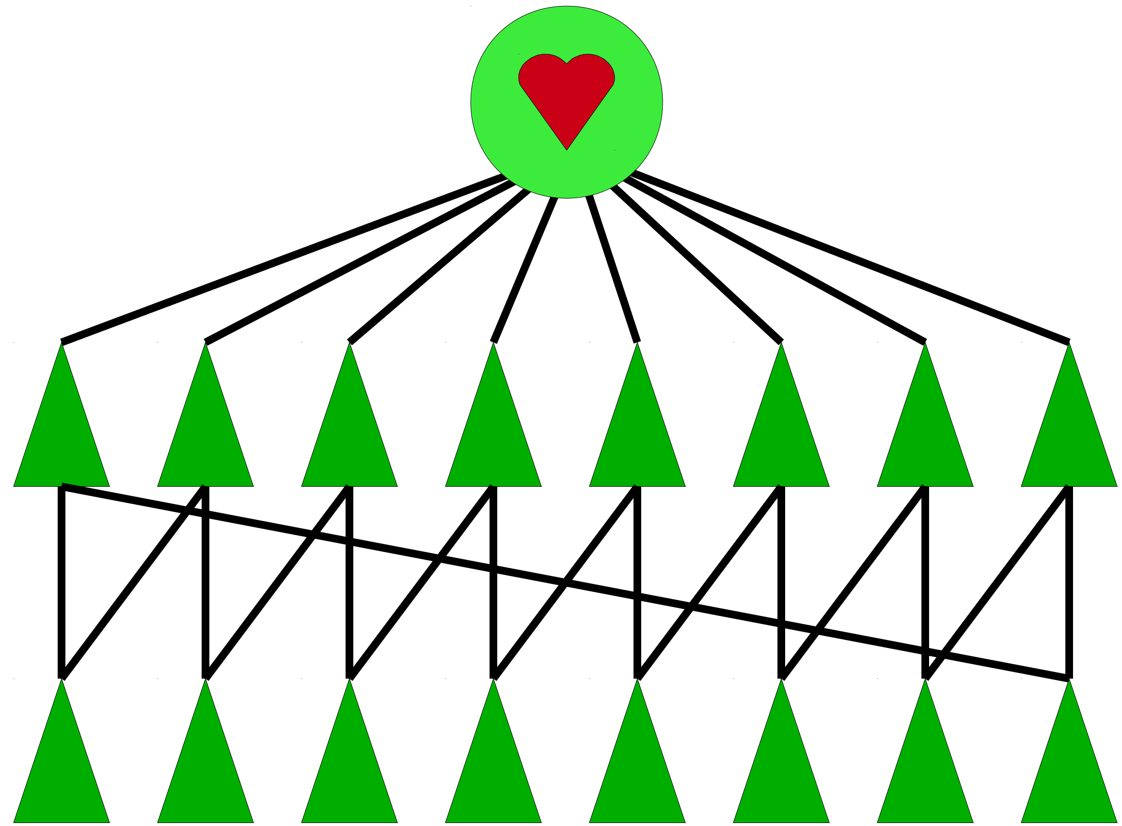
- A ***Hasse diagram*** is a graphical representation of a strict order.
- Elements are drawn from bottom-to-top.
- Higher elements are bigger than lower elements: by ***asymmetry***, the edges can only go in one direction.
- No redundant edges: by ***transitivity***, we can infer the missing edges.

Hasse Artichokes



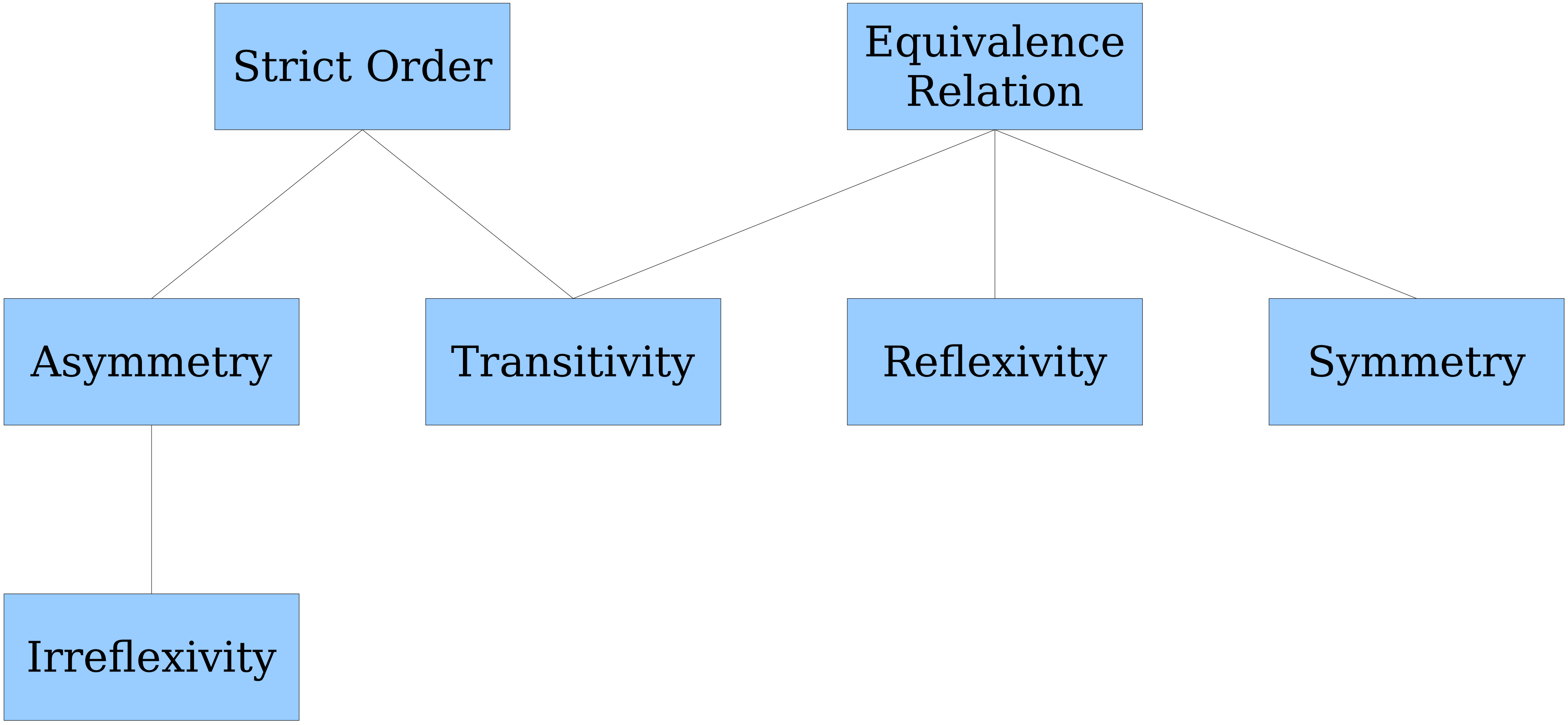
xRy   if   x must be eaten before y

Hasse Artichokes



xRy if x must be eaten before y

The Meta Strict Order



aRb **if** ***a*** **is less specific than** ***b***

The Binary Relation Editor

Time-Out for Announcements!

PS1 Graded

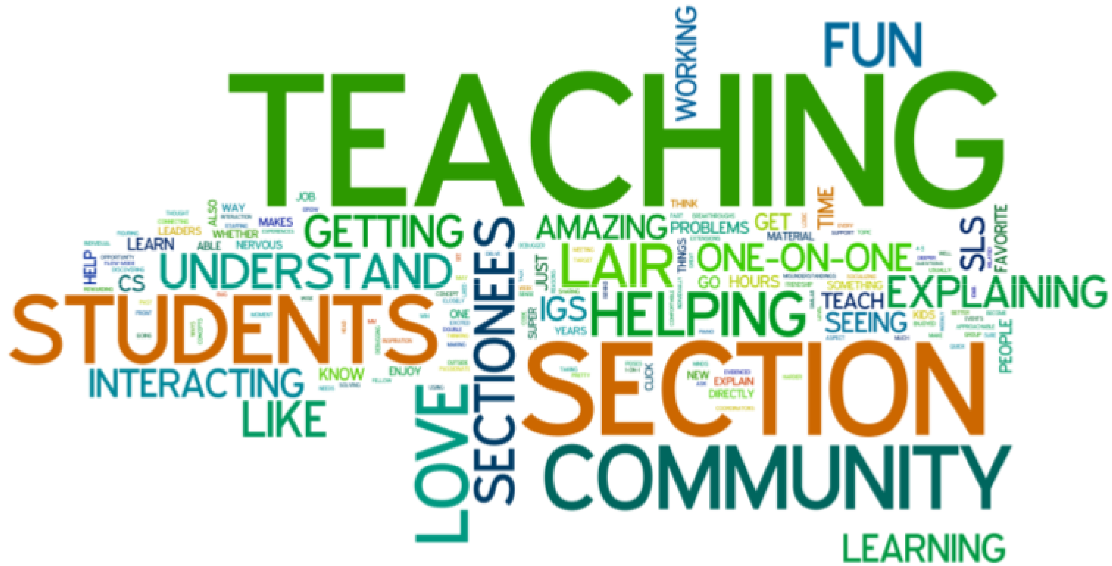
- Problem Set 1 has been graded. Grades and feedback will be released after class.
- Please review your feedback – it's critical for improving your mathematical and proof-writing skills.
- We'll send out an email about regrade requests later today.

PS2 Checkpoint Graded

- We also graded the PS2 checkpoint.
- Look at that feedback as well!
- The rest of PS2 is due on Friday.

Get a Flu Shot!

- Vaden is offering flu shots every day from 3PM – 6PM.
- Highly recommended – you're protecting yourself, your friends, your dormmates, and everyone else in this room!
- Plus, this year's shots don't hurt at all. Yay!



CS106 SECTION LEADING

Deadline for students who have completed CS106B/X:

Thursday, October 22nd

Current CS106B students only: Thursday, November 5th

All majors welcome!

Applicants must either be enrolled in CS106B, or have completed CS106B/X, or CS107.

Apply online…

cs198.stanford.edu

Questions? Contact the CS198 Coordinators (Aaron Broder, Danielle Kain, and Nhien Tran) at cs198@cs.stanford.edu

A CS Department Milestone

Undergraduate Majors

Sorted by Number of Women

214	***Computer Science***
208	Human Biology
153	Engineering Special Programs
100	Biology
99	Science Technology and Society
85	International Relations
75	Psychology
68	English
67	Economics
67	Mechanical Engineering

Source: Eric Roberts, citing university official figures

Your Questions

“What do you think of the gender imbalance in the tech industry?”

It's there. It needs to change. But it's getting better thanks to a concerted effort by a ton of wonderful people.

"I loved 106A, but I'm constantly afraid I'm going to take a class sometime soon that will weed me out of the major (possibly even this one). When do you think is the right time to declare? I'm scared I won't realize I can't handle CS until it's too late."

I'm worried that you're thinking about CS and choosing a a major in the wrong way. It's not like there's some "secret special skill" you need to have in order to do CS and that you can't major in CS if you don't have it. There are hard classes, sure, but nothing that's designed to weed out people missing some skill. We think of our courses as funnels rather than filters.

That said, if you want to declare CS, any time is a good time! ☺

"Can you share some of your hobbies? ☺"

Sure! I love cooking, reading Korean food blogs, reading (mostly nonfiction), and traveling.

“How do you motivate yourself to work so hard? We know you love CS but you have bad days, too -- right? How do you get through the days when propositional logic is getting you down?”

I have great coworkers and really supportive friends and family. This job is hard – there are a lot of times where it's super stressful and I need a break – and they help me get through it. Plus, all it takes is one or two good interactions with students and everything seems great again.

“How do you motivate yourself to work so hard? We know you love CS but you have bad days, too -- right? How do you get through the days when propositional logic is getting you down?”

I have great coworkers and really supportive friends and family. This job is hard - there are a lot of times where it's super stressful and I need a break - and they help me get through it. Plus, all it takes is one or two good interactions with students and everything seems great again.

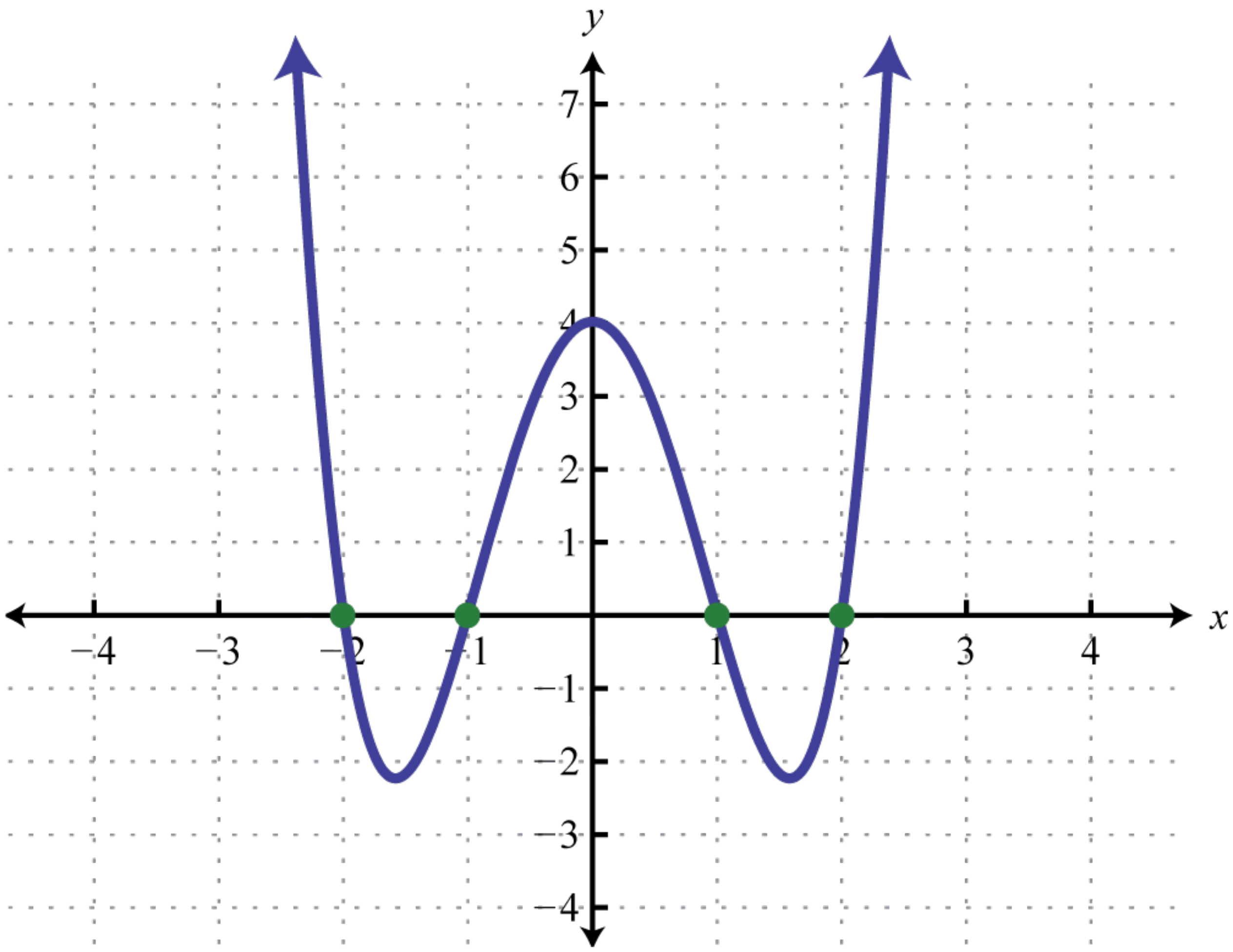
Also, sometimes I make soondubu for dinner. Or make bread. Kneading bread is cathartic.

Back to CS103!

Functions

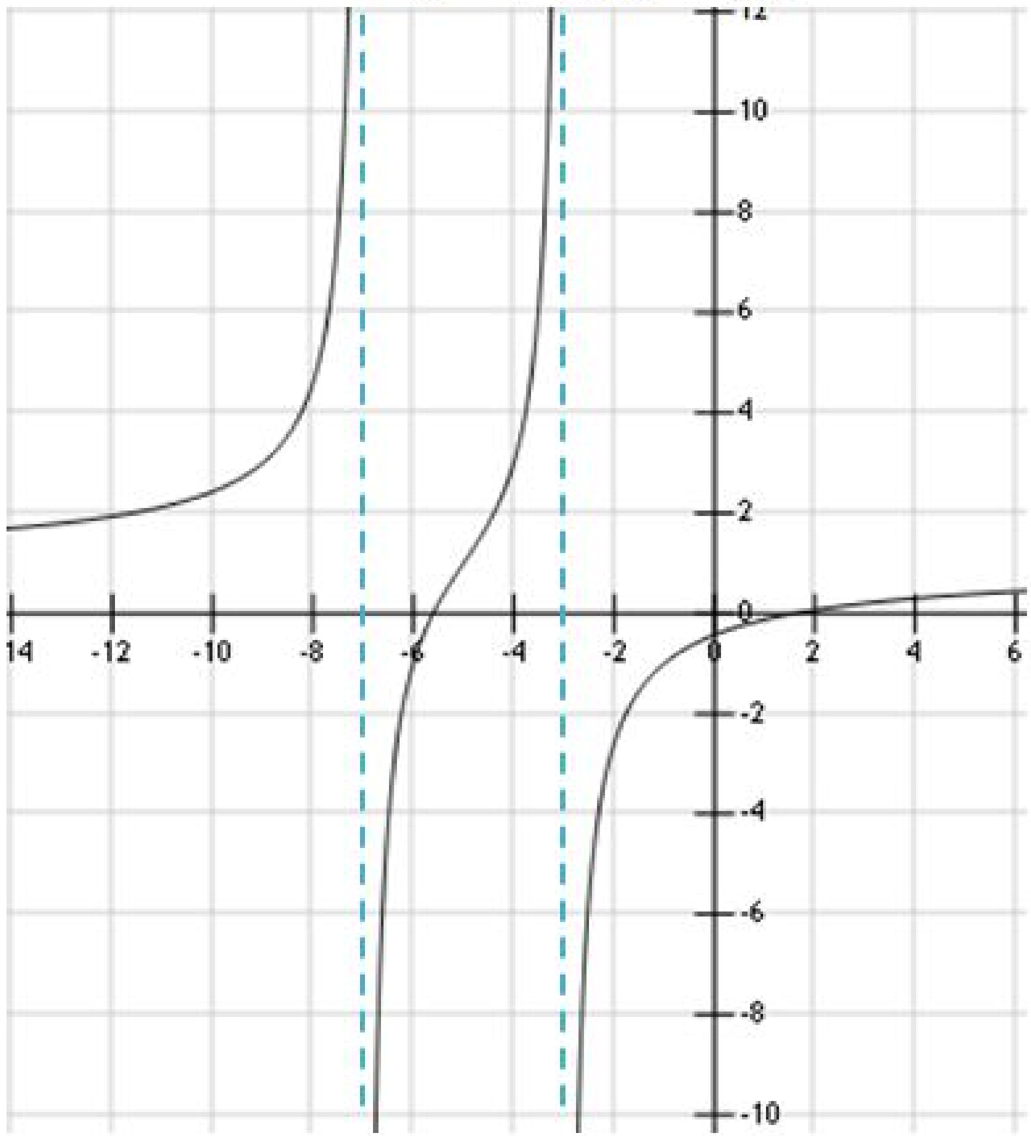
What is a function?

Functions, High-School Edition



$$f(x) = x^4 - 5x^2 + 4$$

$$f(x) = \frac{x^2 + 4x - 9}{x^2 + 10x + 21}$$



source: http://study.com/cimages/multimages/16/asymptote_1.JPG

Functions, High-School Edition

- In high school, functions are usually given as objects of the form

$$f(x) = \frac{x^3+3x^2+15x+7}{1-x^{137}}$$

- What does a function do?
  - Takes in as input a real number.
  - Outputs a real number.
  - … except when there are vertical asymptotes or other discontinuities, in which case the function doesn't output anything.

Functions, CS Edition

```
int flipUntil(int n) {
  int numHeads = 0;
  int numTries = 0;

  while (numHeads < n) {
    if (randomBoolean()) numHeads++;

    numTries++;
  }

  return numTries;
}
```

Functions, CS Edition

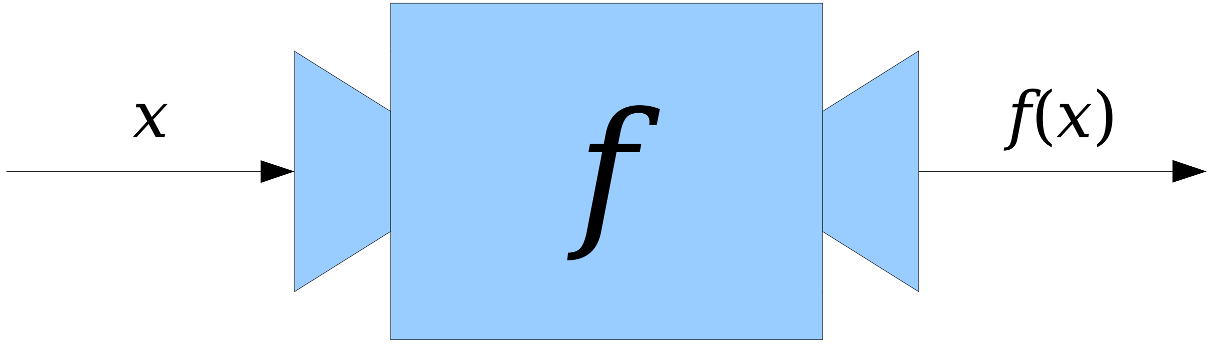
- In programming, functions
  - might take in inputs,
  - might return values,
  - might have side effects,
  - might never return anything,
  - might crash, and
  - might return different values when called multiple times.

What's Common?

- Although high-school math functions and CS functions are pretty different, they have two key aspects in common:
  - They take in inputs.
  - They produce outputs.
- In math, we like to keep things easy, so that's pretty much how we're going to define a function.

Rough Idea of a Function:

A function is an object *f* that takes in one input and produces exactly one output.



(This is not a complete definition – we'll revisit this in a bit.)

High School versus CS Functions

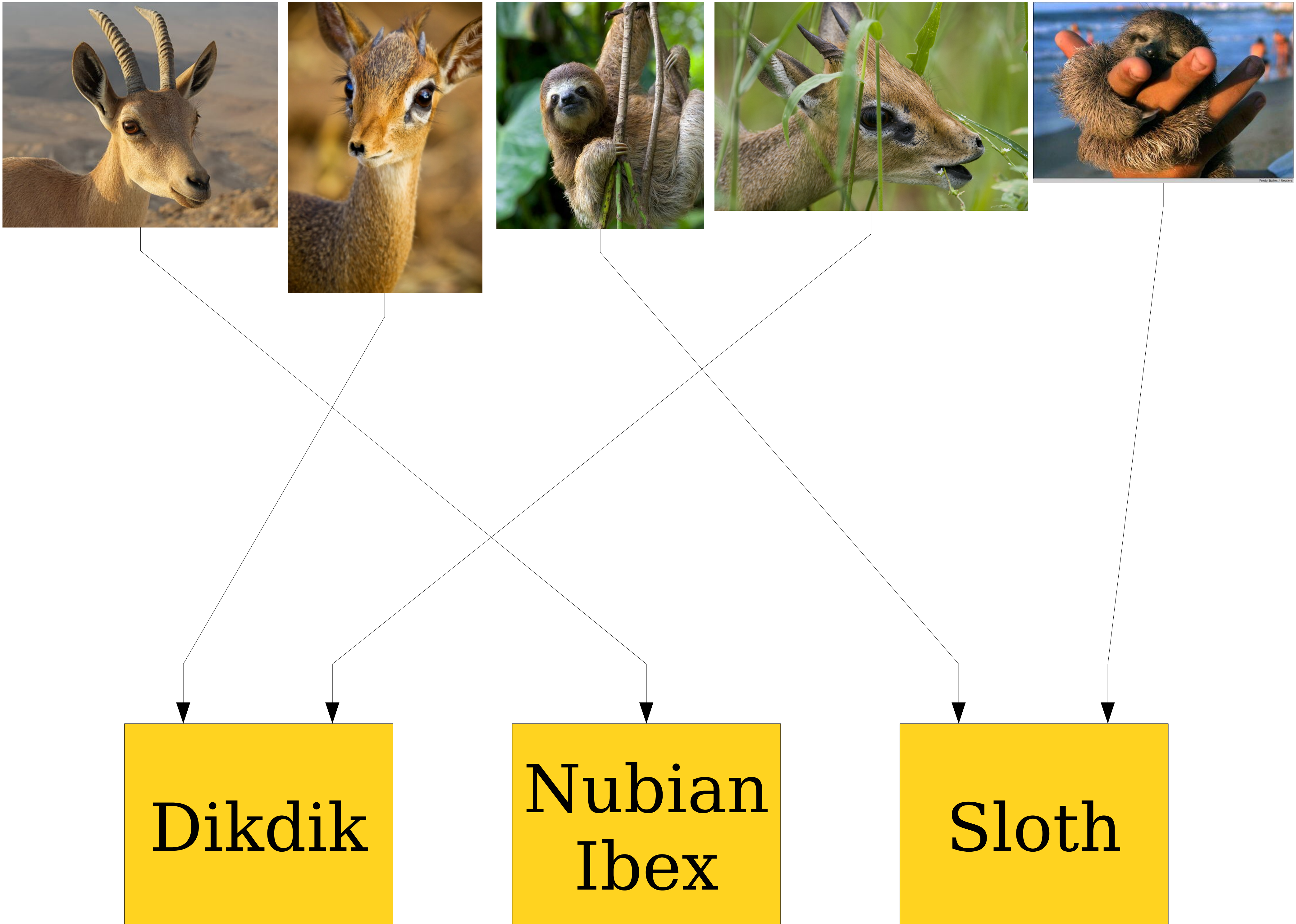
- In high school, functions usually were given by a rule:

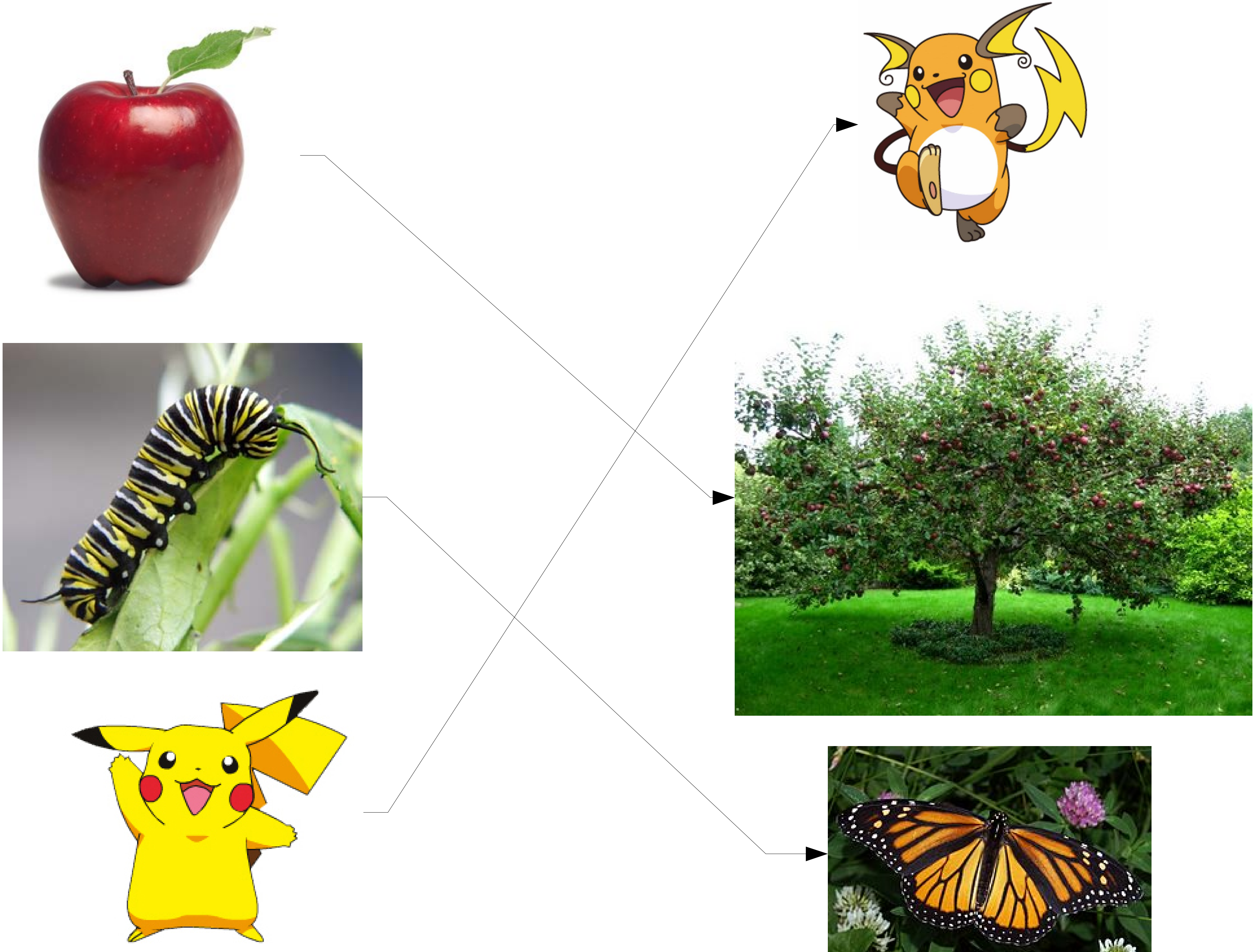
$$f(x) = 4x + 15$$

- In CS, functions are usually given by code:

```
int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

- What sorts of functions are we going to allow from a mathematical perspective?


Dikdik
Nubian Ibex
Sloth



… but also …

$$f(x) = x^2 + 3x - 15$$

$$f(n) = \begin{cases} -n/2 & \text{if } n \text{ is even} \\ (n+1)/2 & \text{otherwise} \end{cases}$$

Functions like these are called ***piecewise functions***.

To define a function, typically you will either

- draw a picture, or
- give a rule for determining the output.

In mathematics, functions are ***deterministic***.

That is, given the same input, a function must always produce the same output.

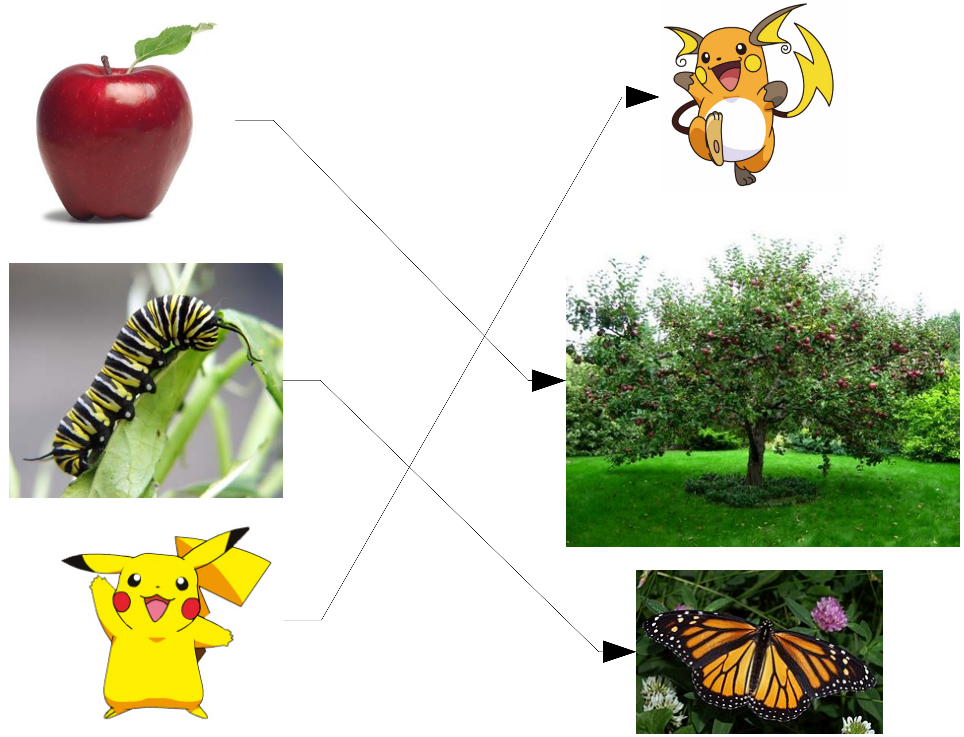
One Challenge

$$f(x) = x^2 + 2x + 5$$

$$f(\ 3\ ) = 3^2 + 3 \cdot 2 + 5 = 20$$

$$f(\ 0\ ) = 0^2 + 0 \cdot 2 + 5 = 5$$

$f($  $) = \ldots ?$



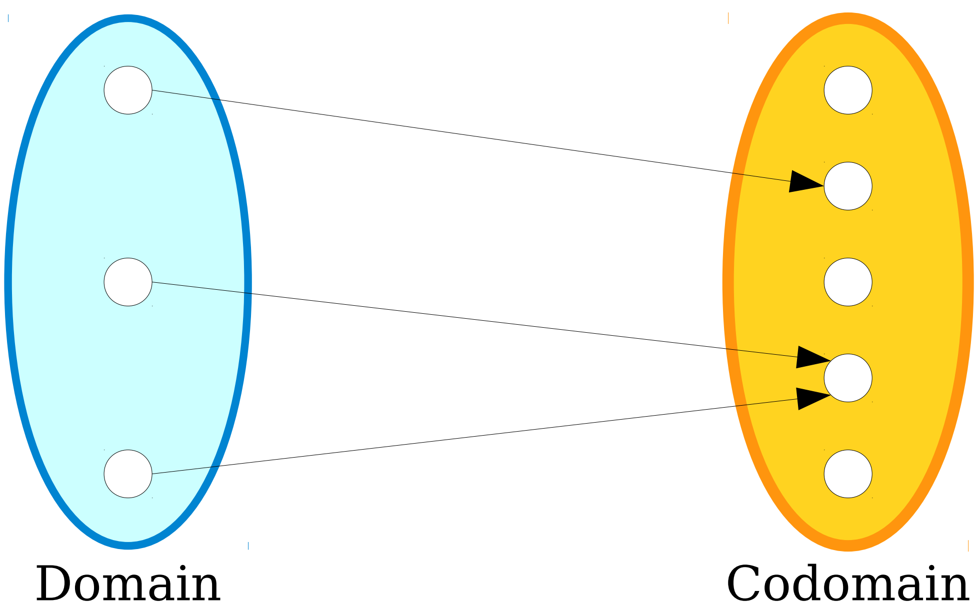
$f($  $) =$ 

$$f(137) = \ldots?$$

We need to make sure we can't apply functions to meaningless inputs.

Domains and Codomains

- Every function f has two sets associated with it: its ***domain*** and its ***codomain***.
- A function f can only be applied to elements of its domain. For any x in the domain, $f(x)$ belongs to the codomain.



Domains and Codomains

- If f is a function whose domain is A and whose codomain is B, we write $\boldsymbol{f : A \to B}$.
- This notation just says what the domain and codomain of the function is. It doesn't say how the function is evaluated.
- Think of it like a "function prototype" in C or C++. The notation $f : A \to B$ is like writing

  B f(A `argument);`

  We know that f takes in an A and returns a B, but we don't know exactly which B it's going to return for a given A.

Domains and Codomains

- A function f must be defined for every element of the domain.
  - For example, if $f : \mathbb{R} \to \mathbb{R}$, then the following function is ***not*** a valid choice for f:

$$f(x) = 1 / x$$

- The output of f on any element of its domain must be an element of the codomain.
  - For example, if $f : \mathbb{R} \to \mathbb{N}$, then the following function is ***not*** a valid choice for f:

$$f(x) = x$$

- However, a function f does not have to produce all possible values in its codomain.
  - For example, if $f : \mathbb{N} \to \mathbb{N}$, then the following function is a valid choice for f:

$$f(n) = n^2$$

Defining Functions

- Typically, we specify a function by describing a rule that maps every element of the domain to some element of the codomain.
- Examples:
  - $f(n) = n + 1$, where $f : \mathbb{Z} \to \mathbb{Z}$
  - $f(x) = \sin x$, where $f : \mathbb{R} \to \mathbb{R}$
  - $f(x) = \lceil x \rceil$, where $f : \mathbb{R} \to \mathbb{Z}$
- Notice that we're giving both a rule and the domain/codomain.

Defining Functions

Typically, we specify a function by describing a rule that maps every element of the domain to some

codomain.

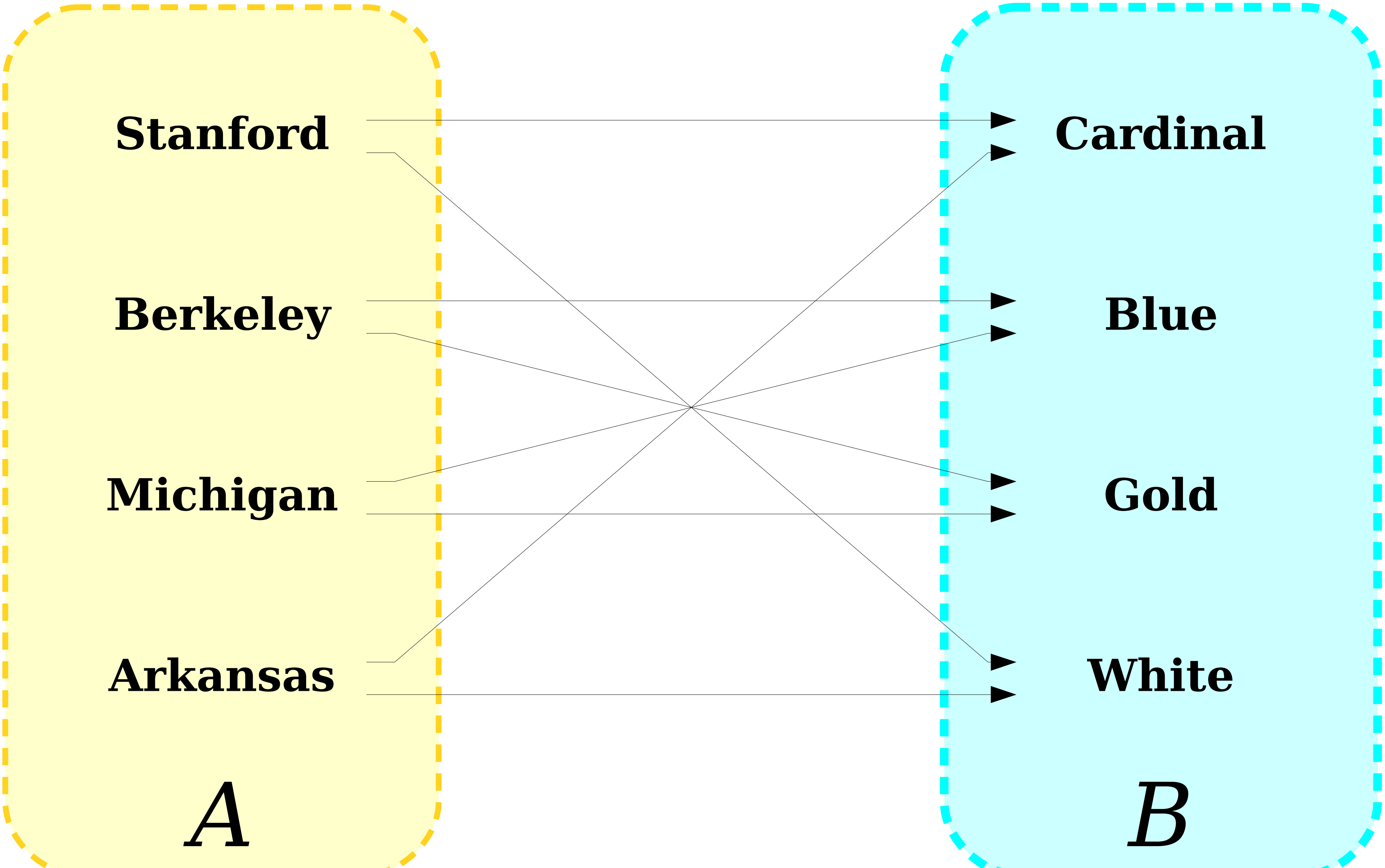
Examples:

- $f(n) = n + 1$, where $f : \mathbb{Z}$
- $f(x) = \sin x$, where $f : \mathbb{R} \to \mathbb{R}$
- $f(x) = \lceil x \rceil$, where $f : \mathbb{R} \to \mathbb{Z}$

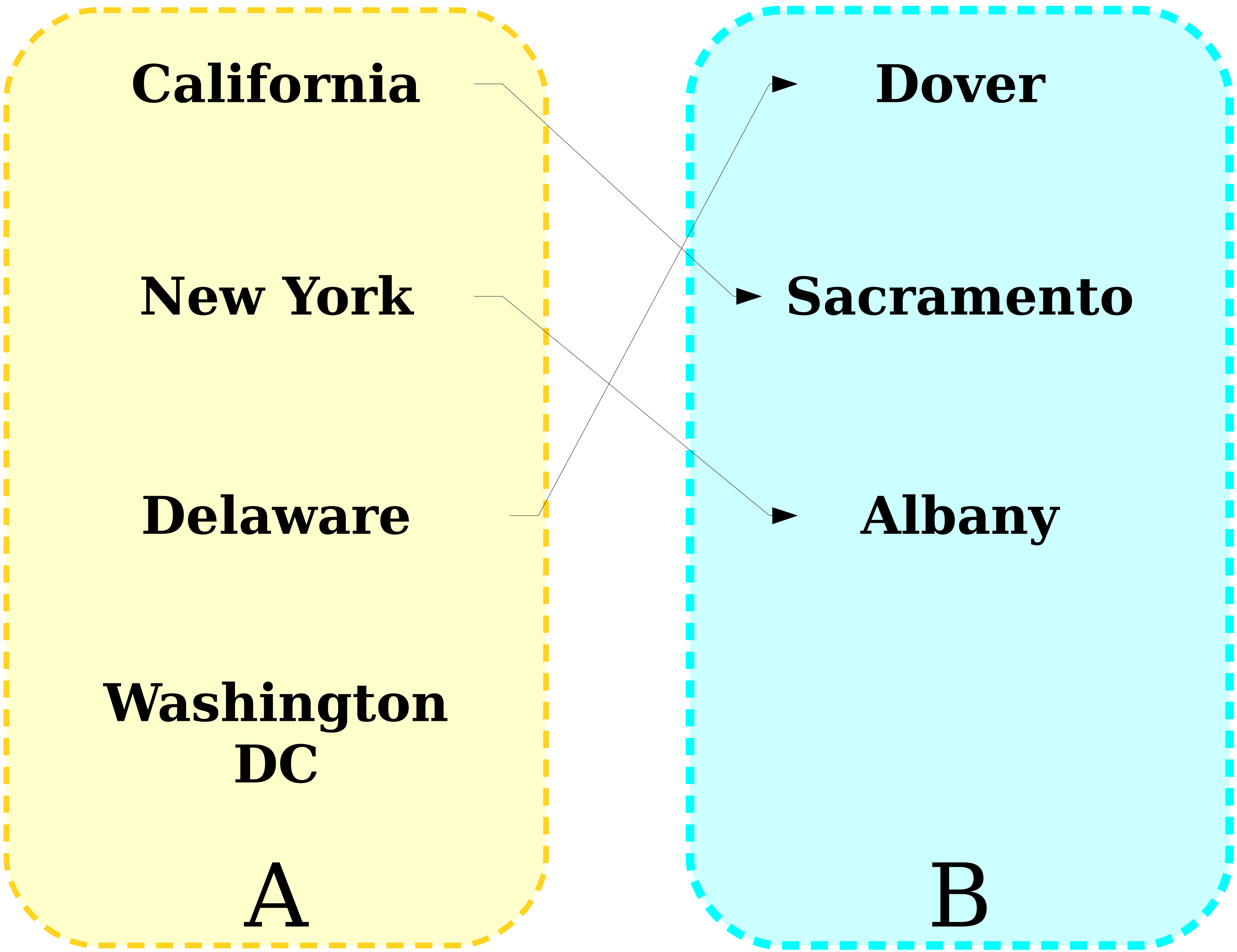
This is the ceiling function – the smallest integer greater than or equal to x. For example, $\lceil 1 \rceil = 1$, $\lceil 1.37 \rceil = 2$, and $\lceil \pi \rceil = 4$.

Notice that we're giving both a rule and the domain/codomain.

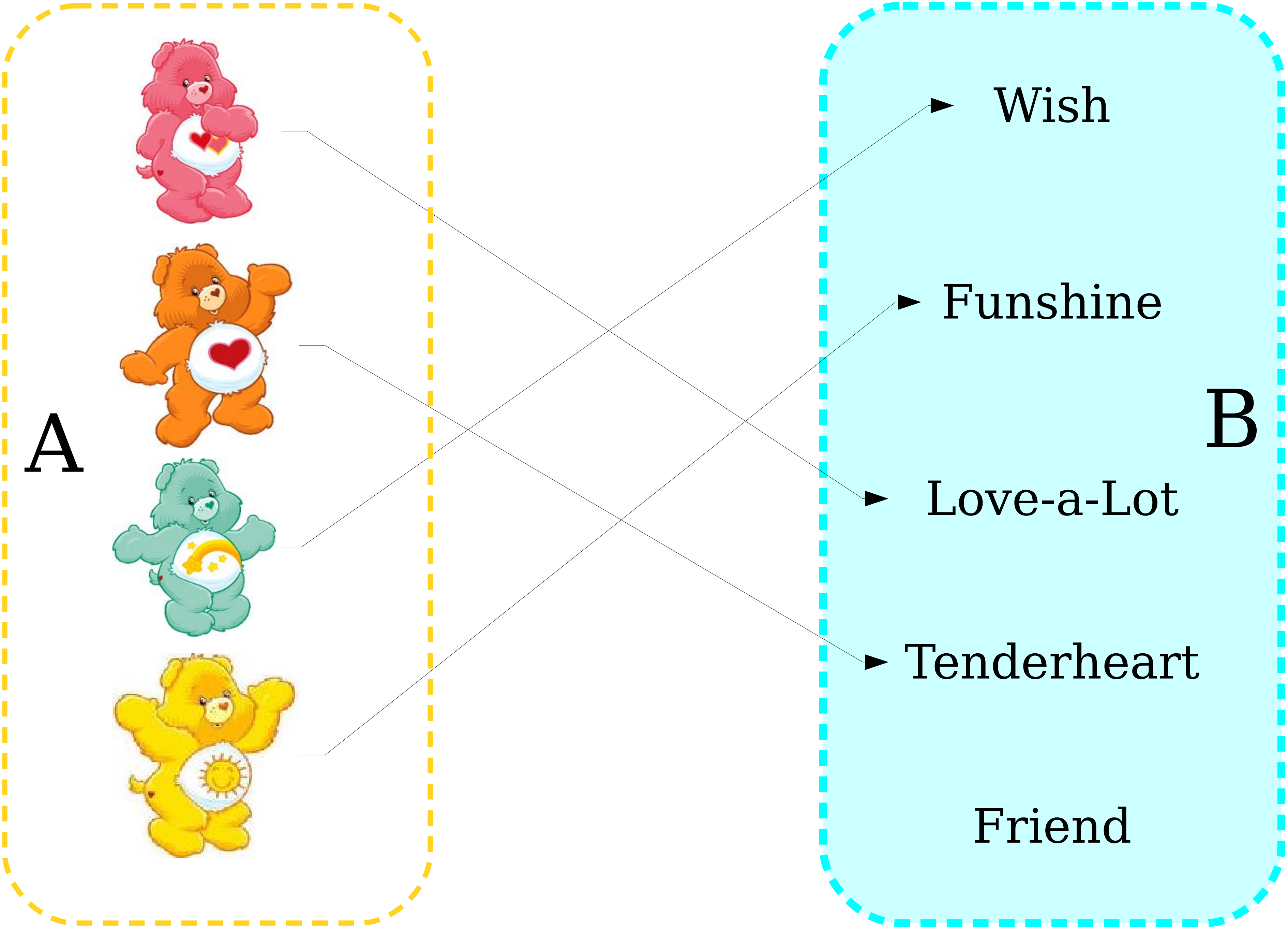
Is this a function from A to B?



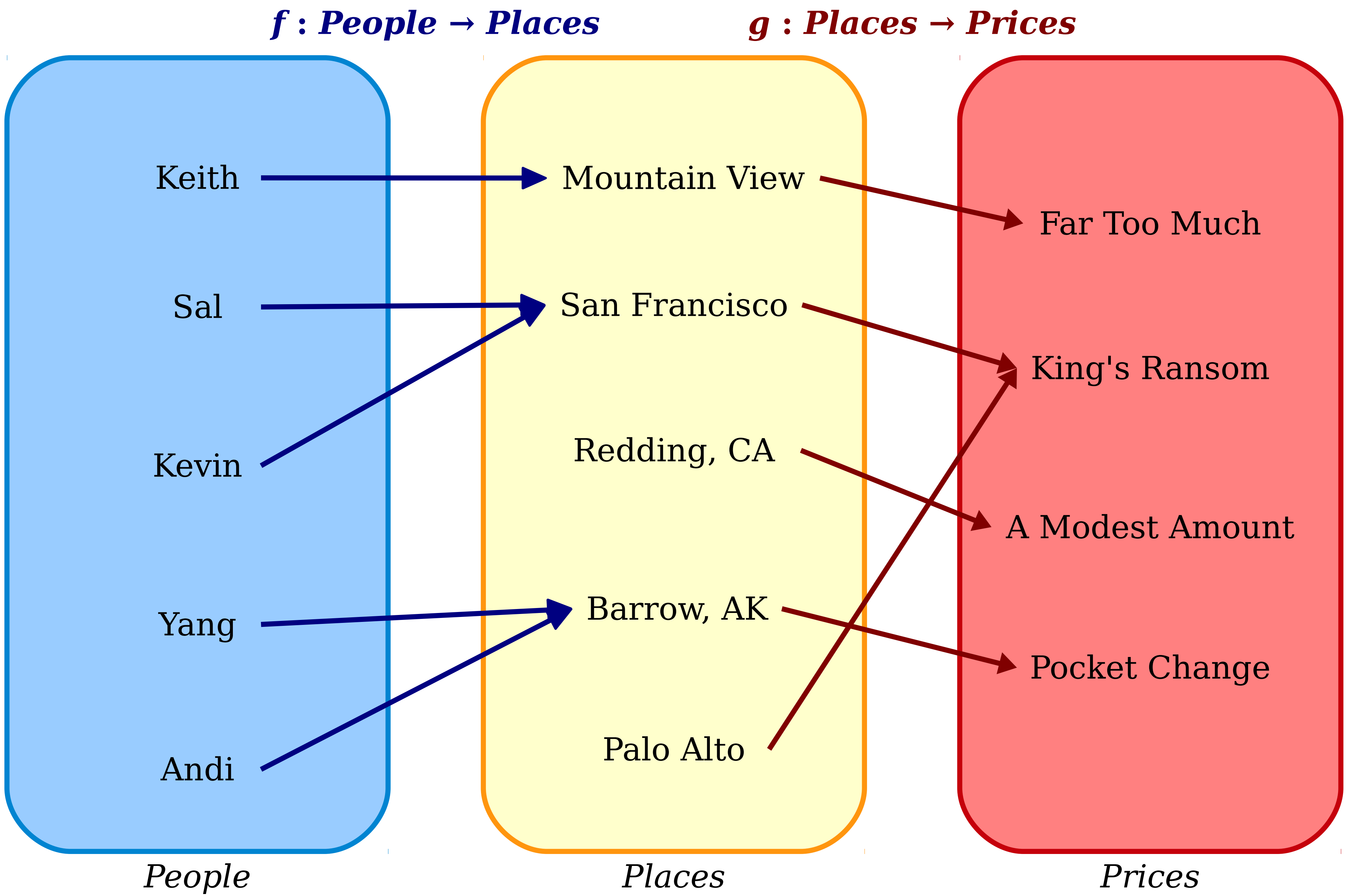
Is this a function from A to B?



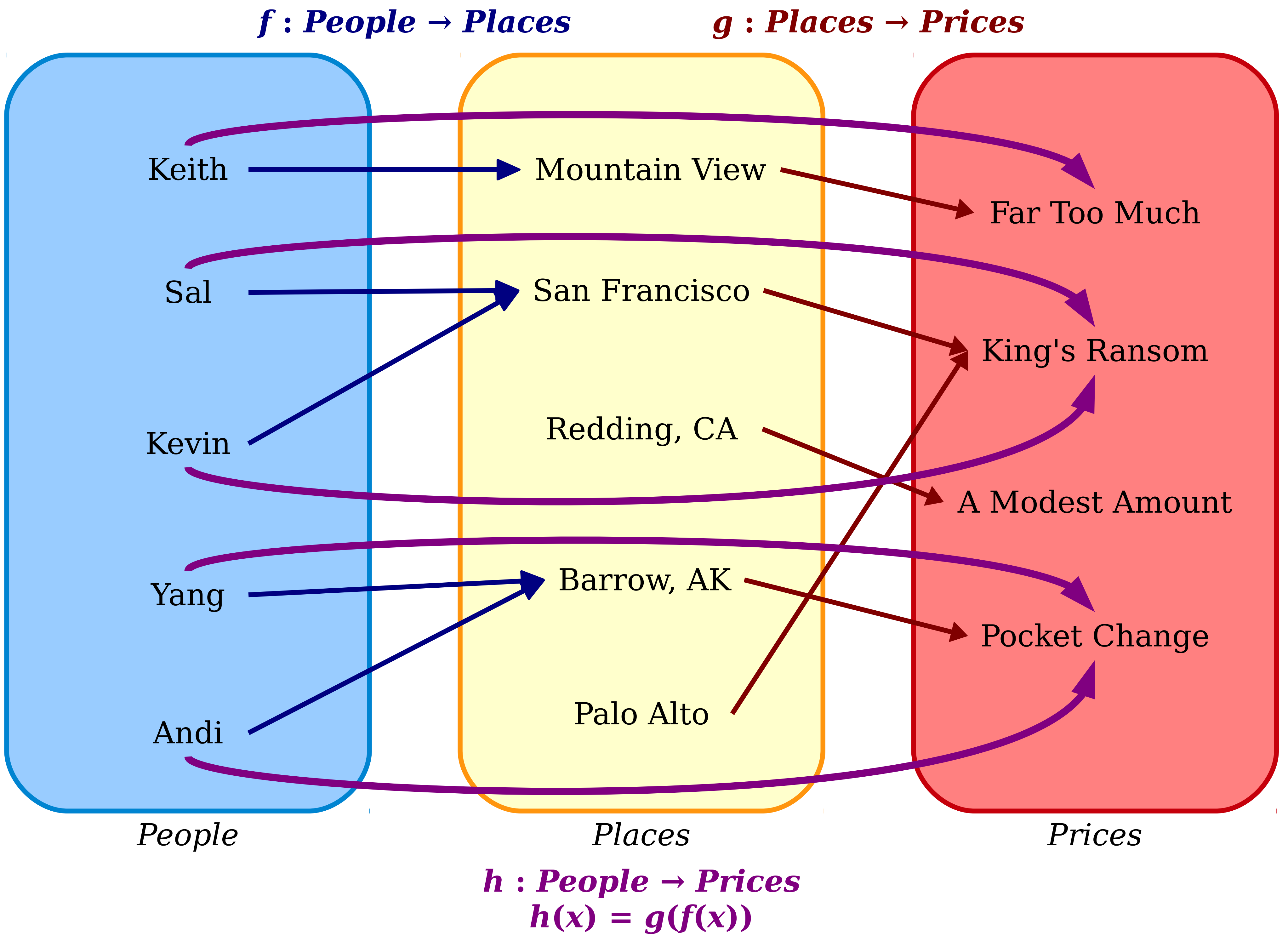
Is this a function from A to B?



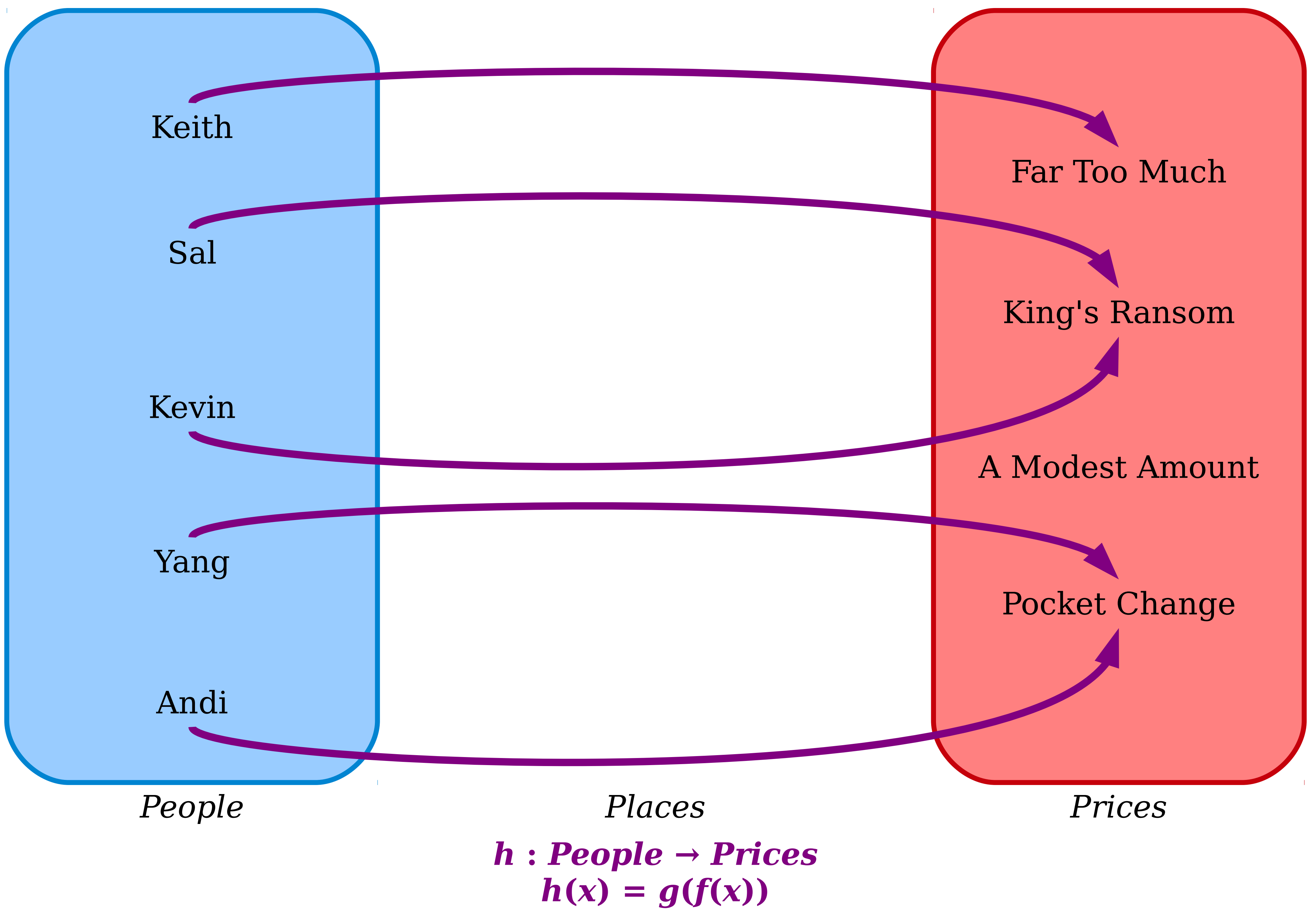
Combining Functions



f : People → Places
g : Places → Prices
Keith
Sal
Kevin
Yang
Andi
Mountain View
San Francisco
Redding, CA
Barrow, AK
Palo Alto
Far Too Much
King's Ransom
A Modest Amount
Pocket Change
People
Places
Prices

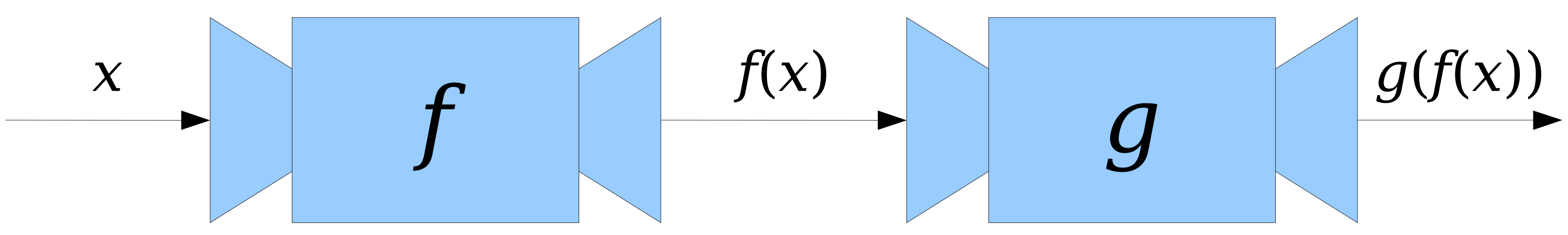


f : People → Places
g : Places → Prices
Keith
Sal
Kevin
Yang
Andi
Mountain View
San Francisco
Redding, CA
Barrow, AK
Palo Alto
Far Too Much
King's Ransom
A Modest Amount
Pocket Change
People
Places
Prices
h : People → Prices
h(x) = g(f(x))


Keith
Sal
Kevin
Yang
Andi
Far Too Much
King's Ransom
A Modest Amount
Pocket Change
People
Places
Prices
h : People → Prices
h(x) = g(f(x))

Function Composition

- Suppose that we have two functions $f : A \to B$ and $g : B \to C$.
- Notice that the codomain of f is the domain of g. This means that we can use outputs from f as inputs to g.



Function Composition

- Suppose that we have two functions $f : A \to B$ and $g : B \to C$.
- The ***composition of f and g***, denoted $\boldsymbol{g \circ f}$, is a function where
  - $g \circ f : A \to C$, and
  - $(g \circ f)(x) = g(f(x))$.

The name of the function is $g \circ f$. When we apply it to an input x, we write $(g \circ f)(x)$. I don't know why, but that's what we do.

- A few things to notice:
  - The domain of $g \circ f$ is the domain of f. Its codomain is the codomain of g.
  - Even though the composition is written $g \circ f$, when evaluating $(g \circ f)(x)$, the function f is evaluated first.

Function Composition

- Let $f : \mathbb{N} \to \mathbb{N}$ be defined as $f(n) = 2n + 1$ and $g : \mathbb{N} \to \mathbb{N}$ be defined as $g(n) = n^2$.
- What is $g \circ f$?

$$\begin{aligned}(g \circ f)(n) &= g(f(n)) \\ &= g(2n + 1) \\ &= (2n + 1)^2 = 4n^2 + 4n + 1\end{aligned}$$

- What is $f \circ g$?

$$\begin{aligned}(f \circ g)(n) &= f(g(n)) \\ &= f(n^2) \\ &= 2n^2 + 1\end{aligned}$$

- In general, if they exist, the functions $g \circ f$ and $f \circ g$ are usually not the same function. ***Order matters in function composition!***