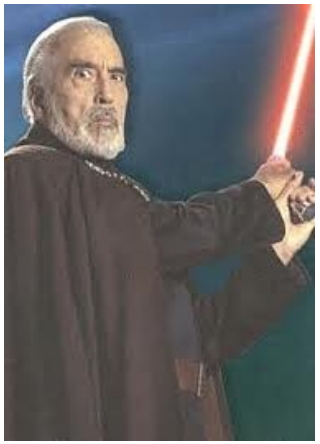
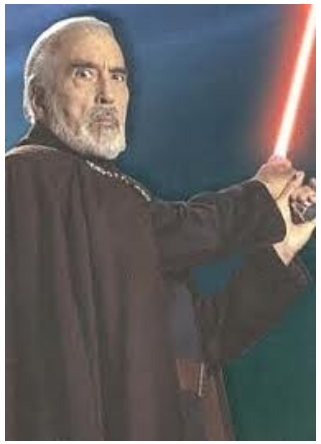
Functions

Announcements

- Problem Set 3 due Friday, October 21
  - Drop by office hours!
  - Ask questions at **cs103@cs.stanford.edu**.
- **Alternate Midterm Times**
  - Wednesday, October 26, 7:00PM – 10:00PM
  - Thursday, October 27, 9:00AM – 12:00 noon
  - Please email us with a preference ASAP so that we can get appropriately-sized rooms.

A **function** is a means of associating each object in one set with an object in some other set.





STAR WARS

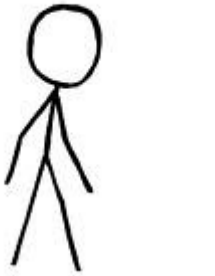
THE LORD OF THE RINGS

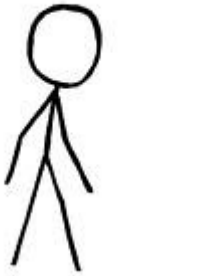
PORTAL™

STAR WARS

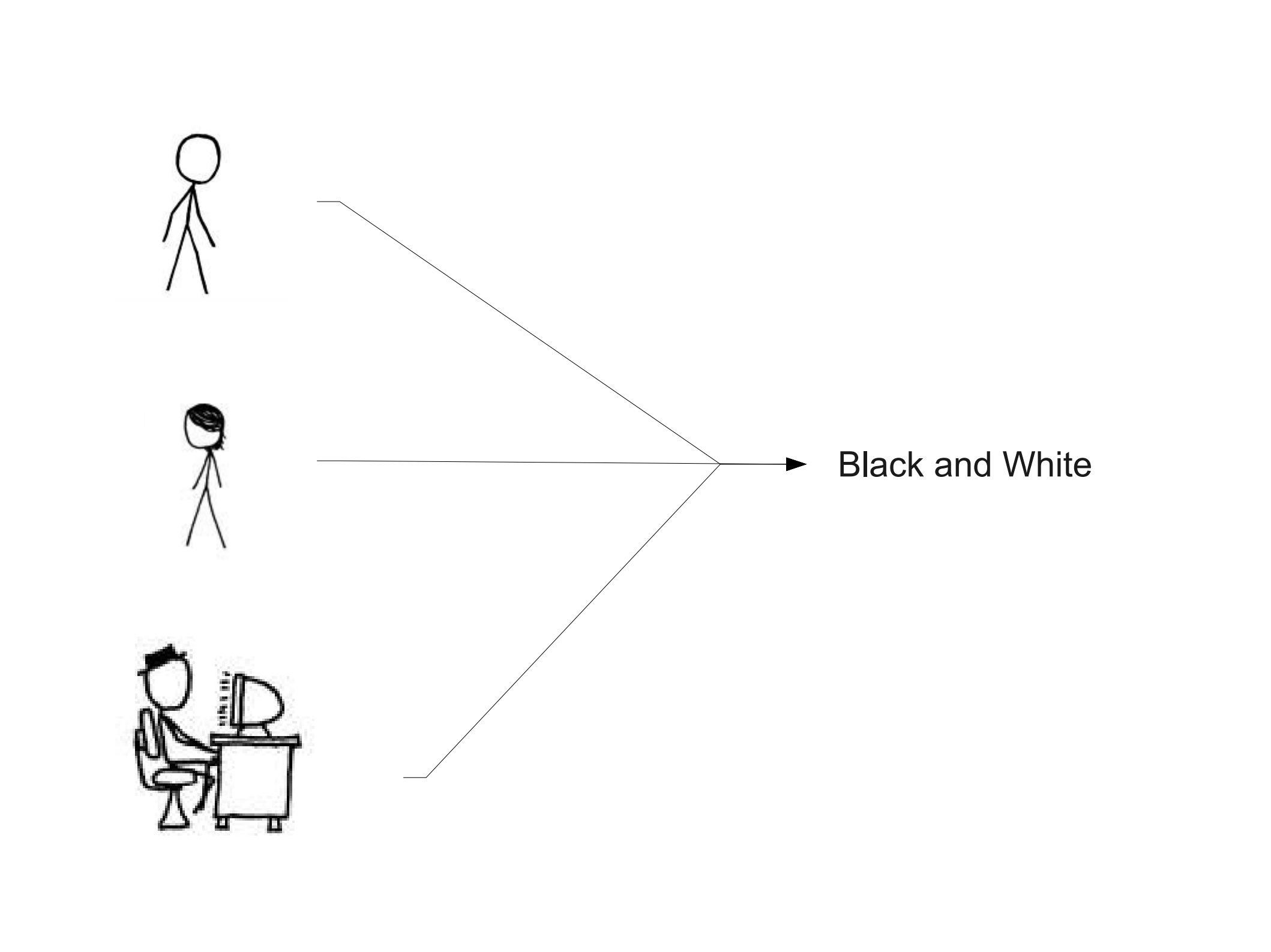
THE LORD OF THE RINGS

PORTAL™





Black and White


Black and White

Formally

- A **function** is a set $f \subseteq A \times B$ such that every value in A is associated with some value in B.
- More formally:
  - If $(a, b) \in f$ and $(a, c) \in f$, then $b = c$.
  - For any $a \in A$, there is some $b \in B$ such that $(a, b) \in f$.
- Even more formally:

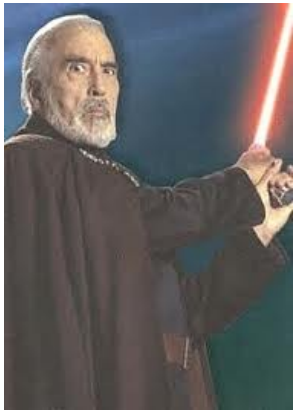
  $\forall a \in A.\ \exists b \in B.\ ((a, b) \in f \wedge \forall c \in B.\ ((a, c) \in f \rightarrow c = b))$
- If $(a, b) \in f$, we write **f(a) = b**.

Terminology

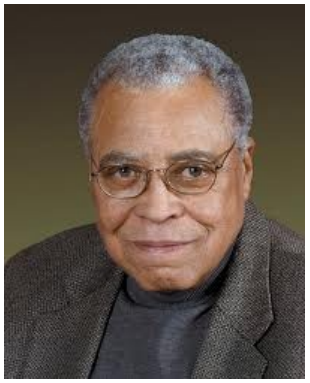
- If f is a function from A to B, we sometimes say that f is a **mapping** from A to B.
- We call A the **domain** of f.
- We call B the **codomain** of f.
- We denote that f is a function from A to B by writing

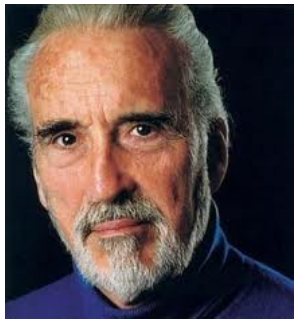
$$f : A \to B$$

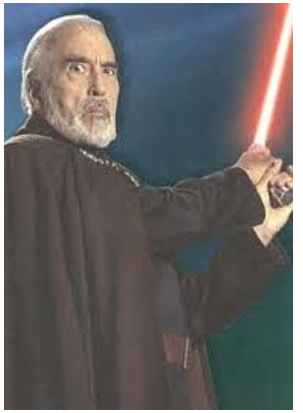
Is this a function?



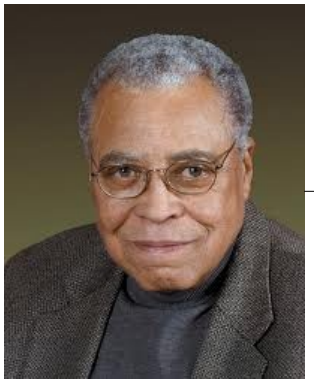
Is this a function?

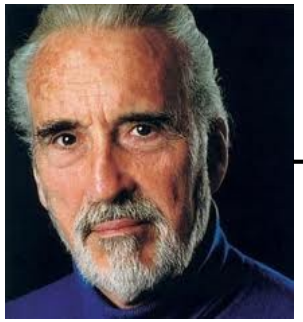




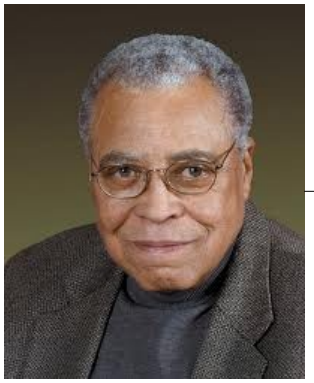


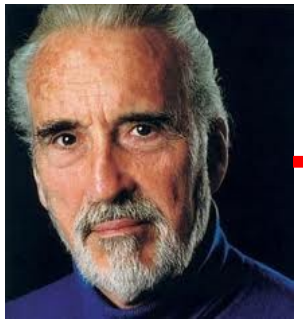
Is this a function?





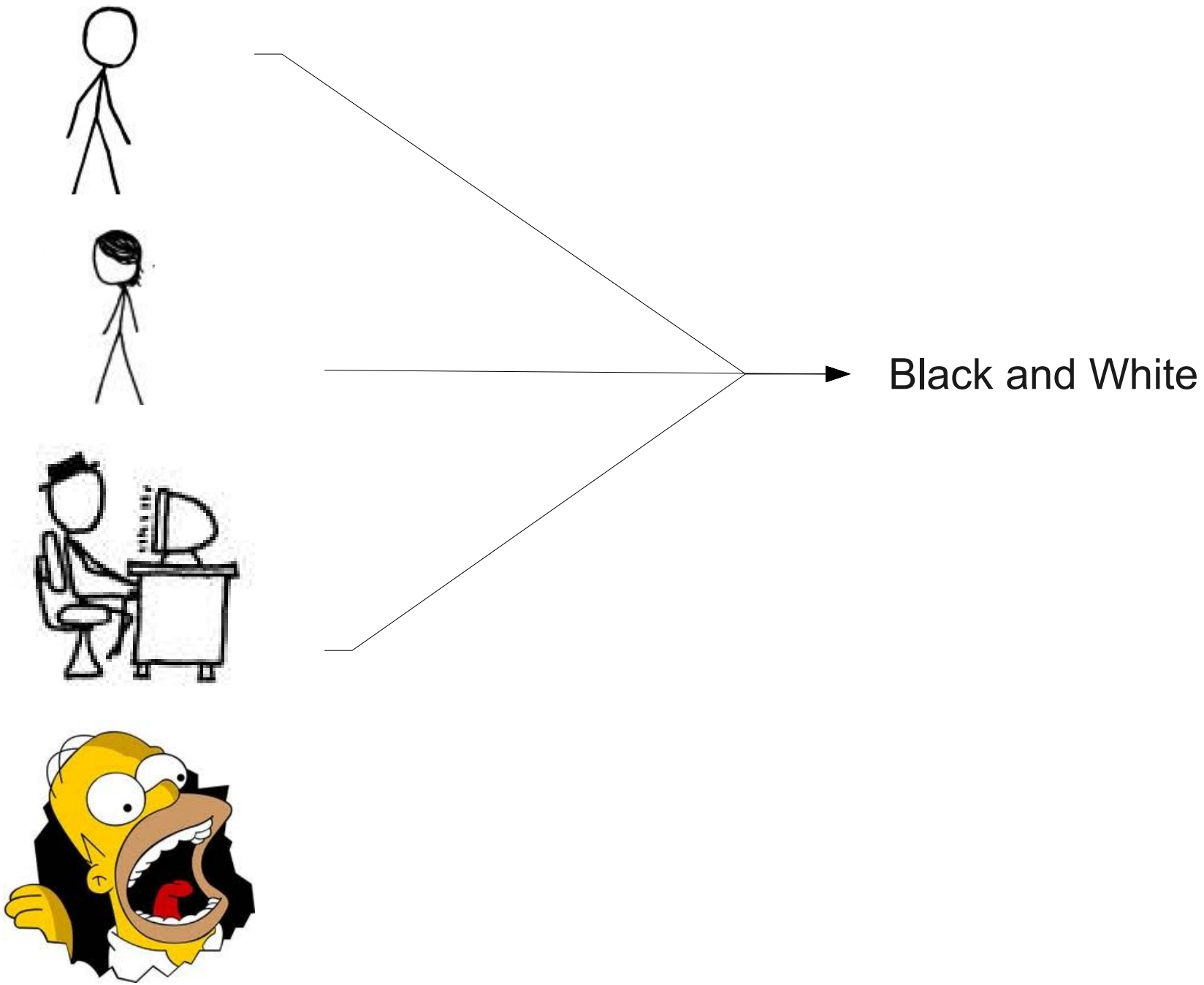
Is this a function?



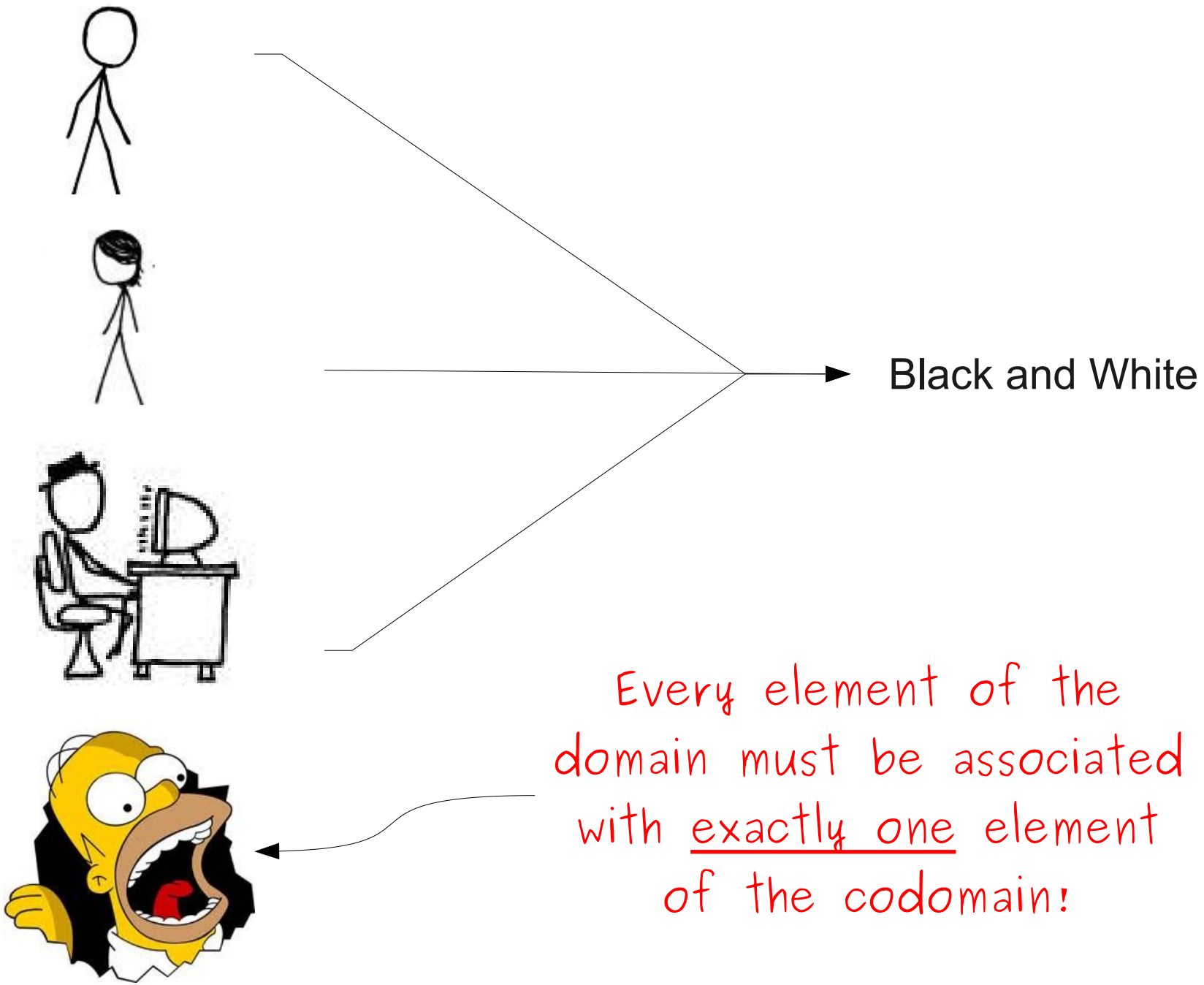


Each object in the domain has to be associated with exactly one object in the codomain!

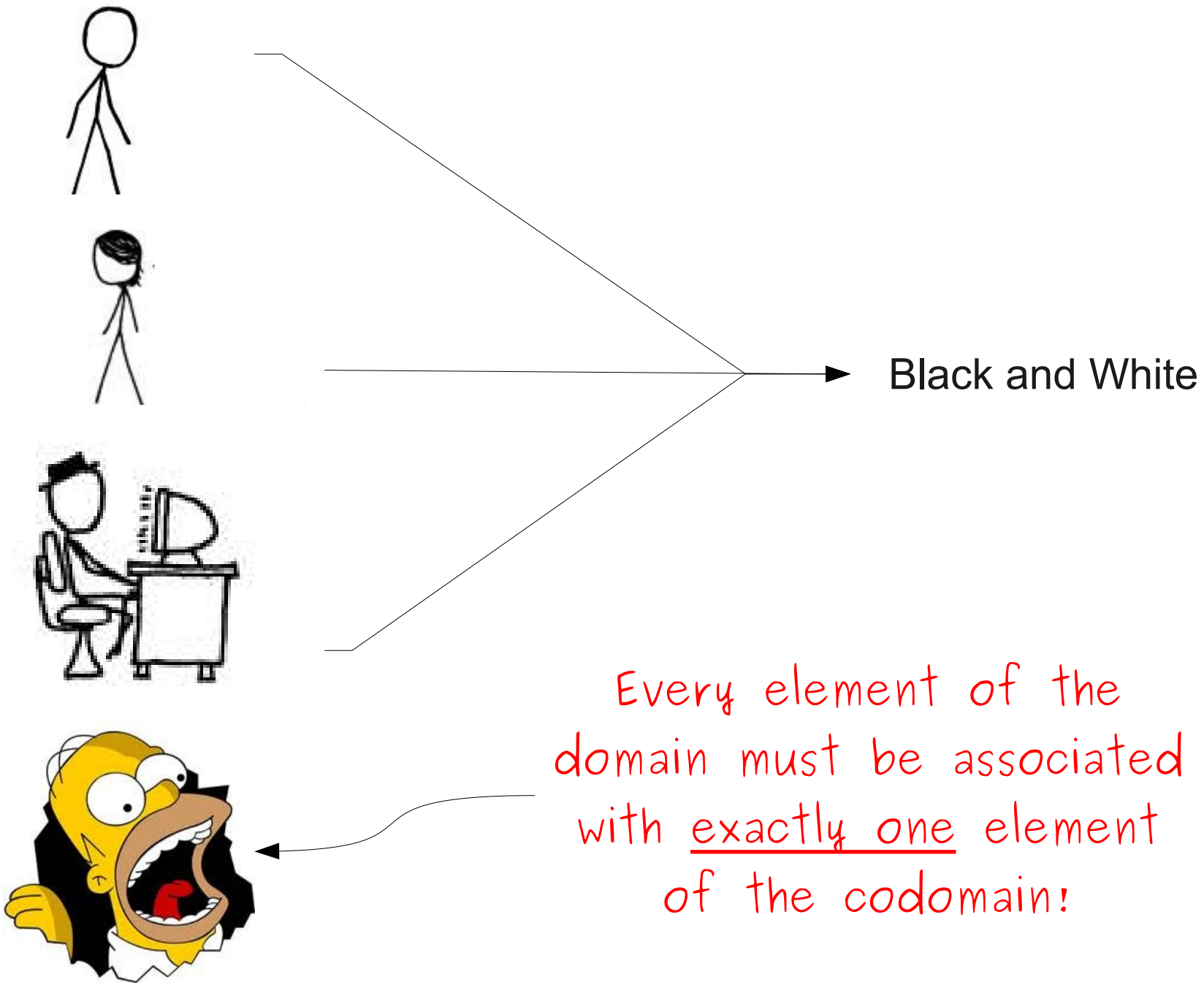
Is this a function?



Is this a function?



Is this a function?



Every element of the domain must be associated with exactly one element of the codomain!

Is this a function?

Is this a function?

Wish

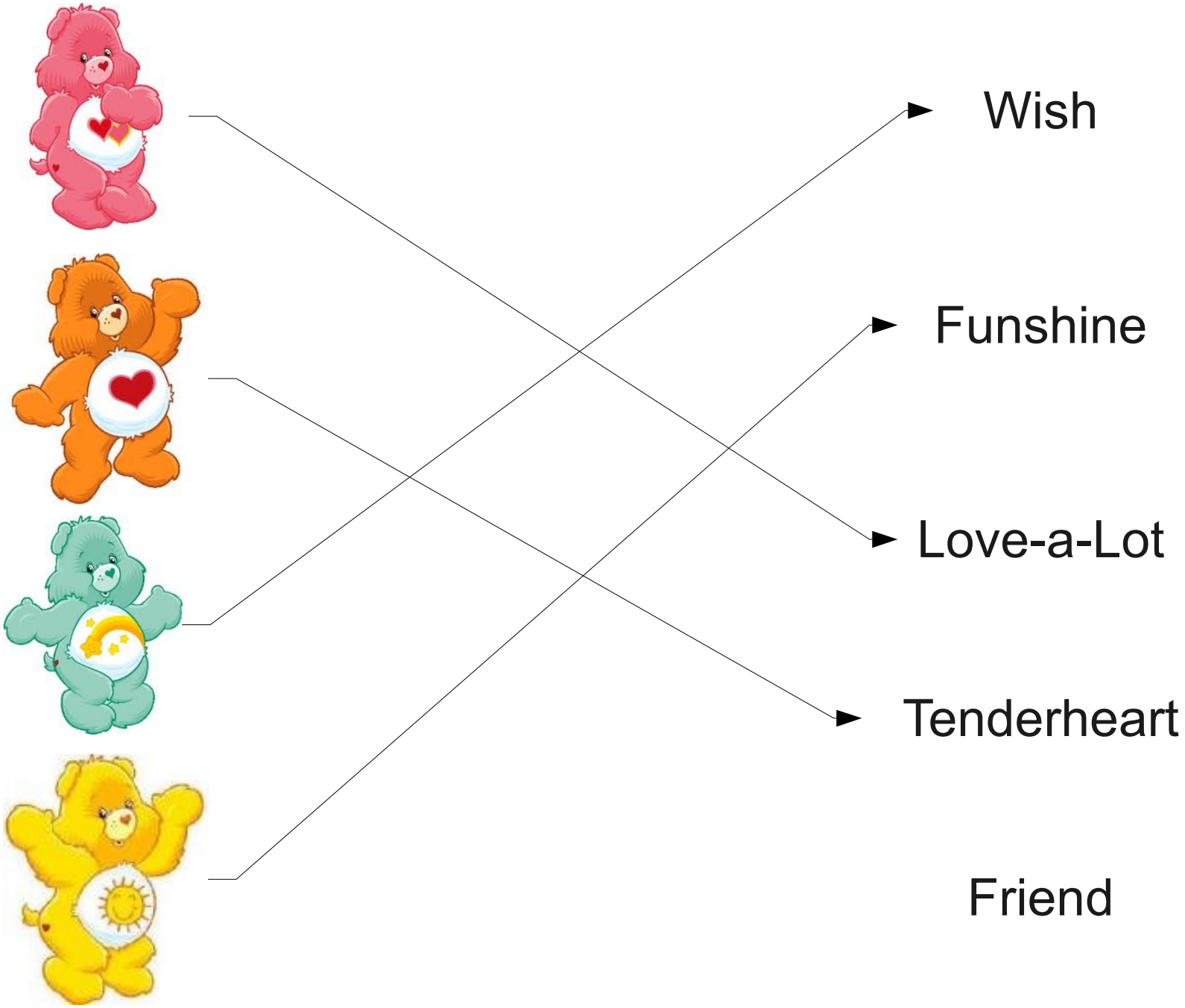
Funshine

Love-a-Lot

Tenderheart

Friend

Is this a function?



Is this a function?

Wish

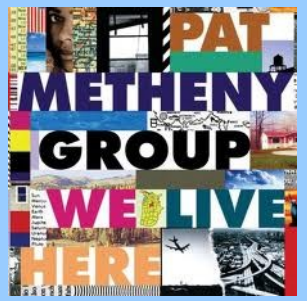
Funshine

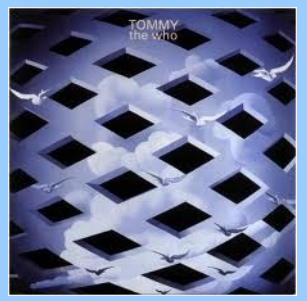
Love-a-Lot

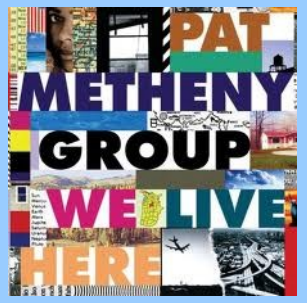
Tenderheart

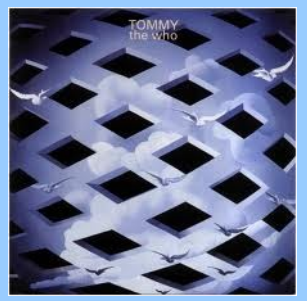
Friend

It's fine that nothing is associated with Friend; functions do not need to use the entire codomain.


PAT
METHENY
GROUP
WE LIVE
HERE


TOMMY
the who





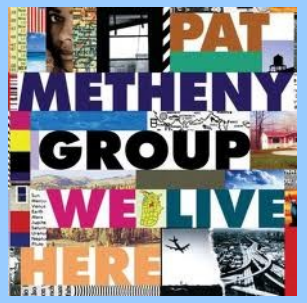
Classical

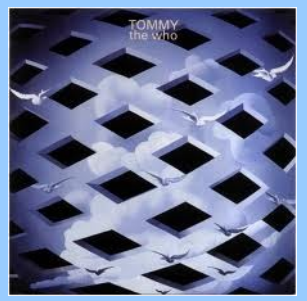
Rock

Jazz

Country

R&B





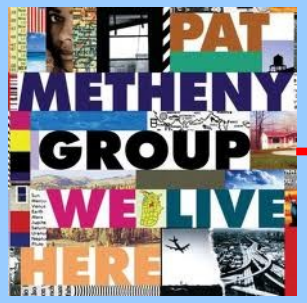
Classical

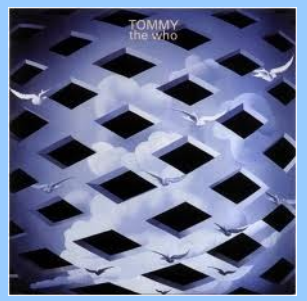
Rock

Jazz

Country

R&B





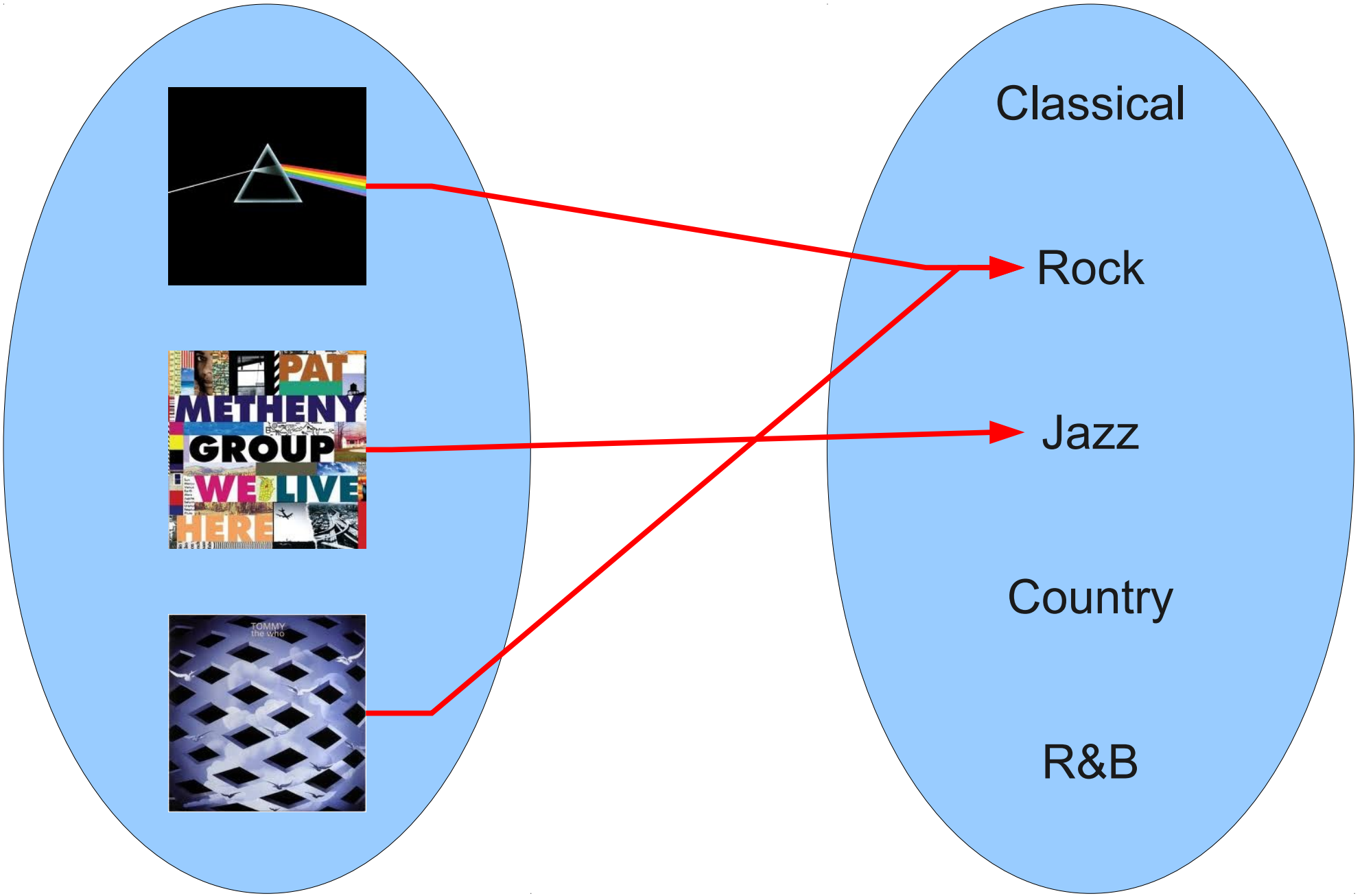
Classical

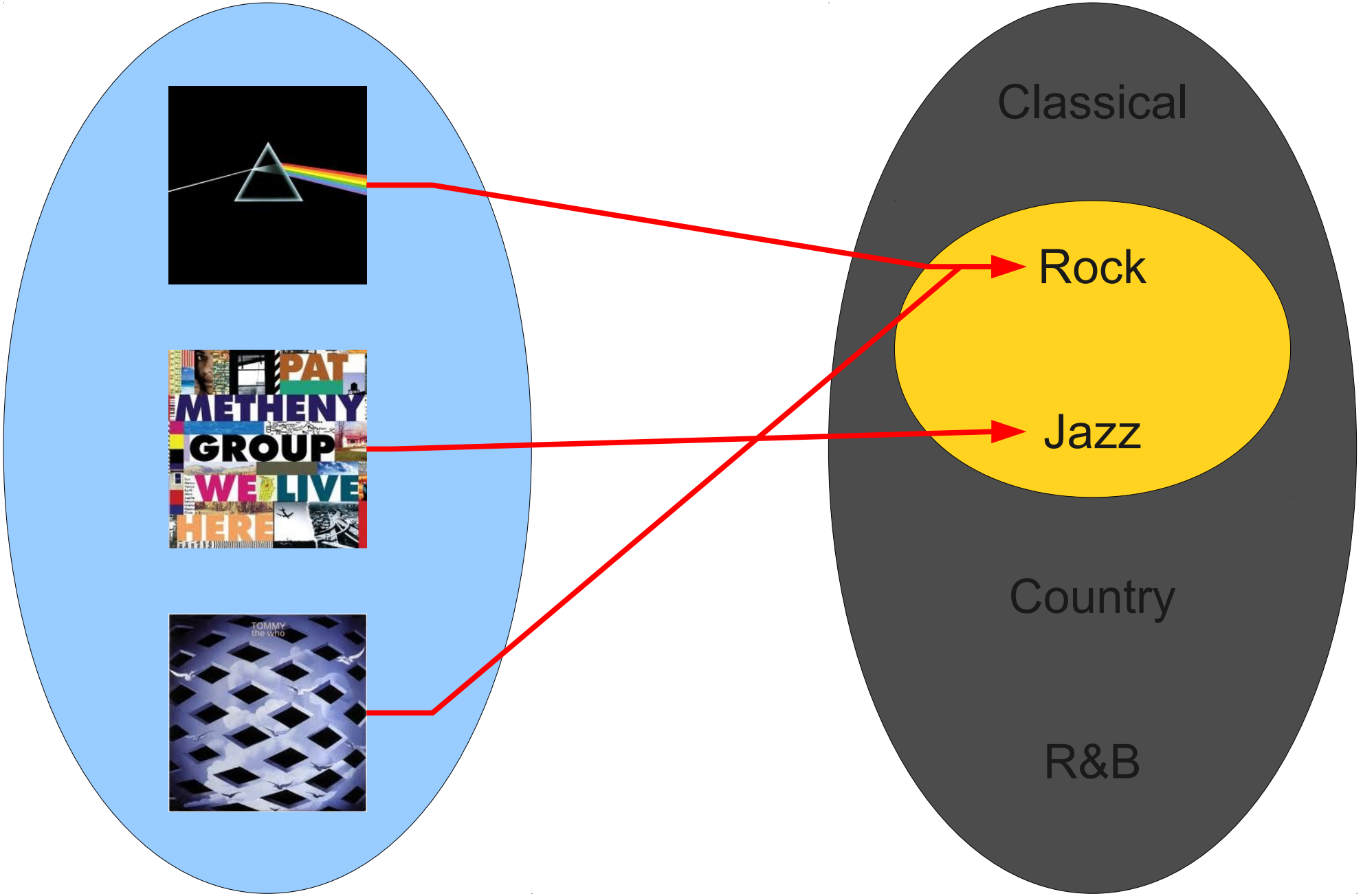
Rock

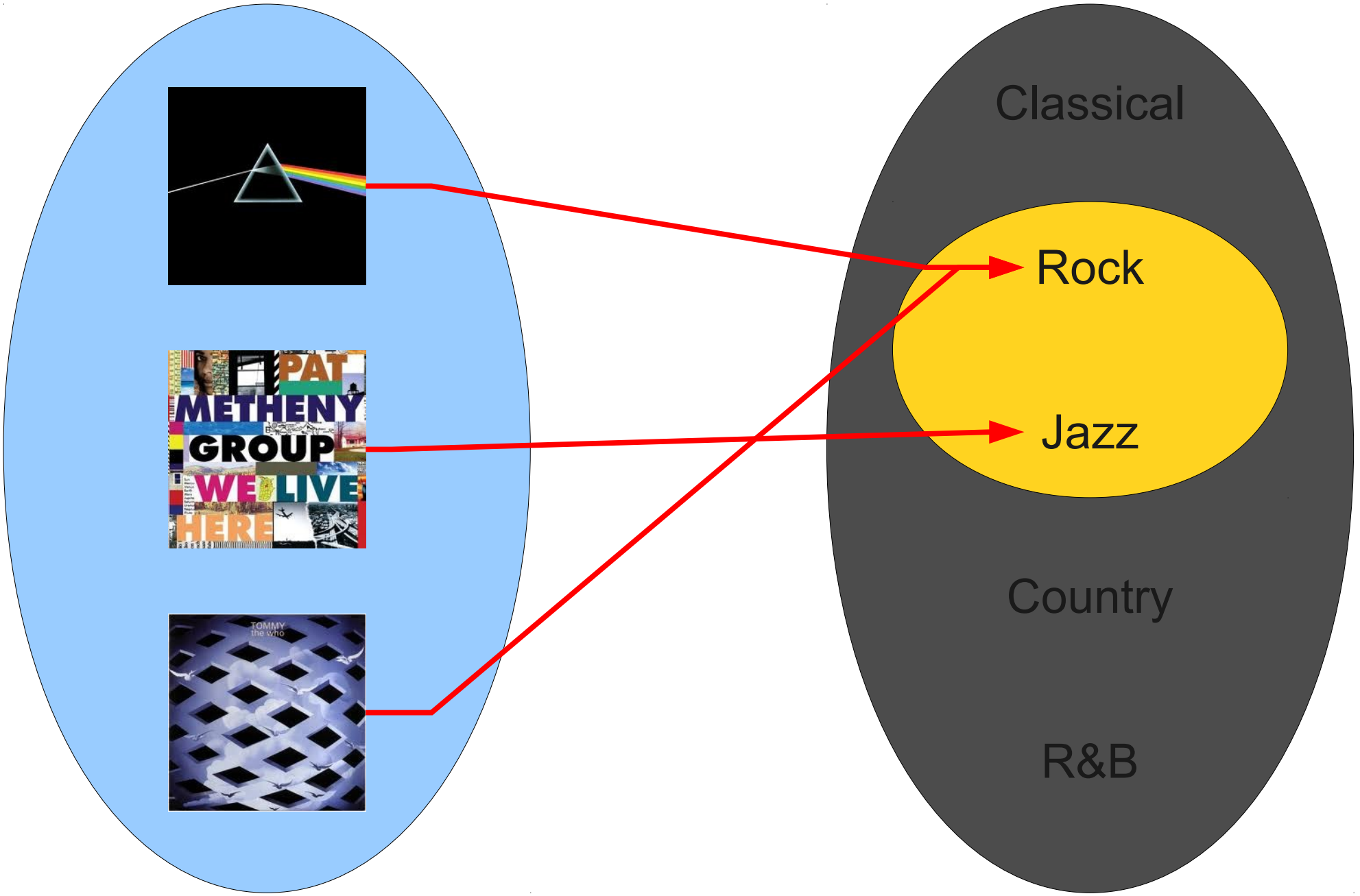
Jazz

Country

R&B

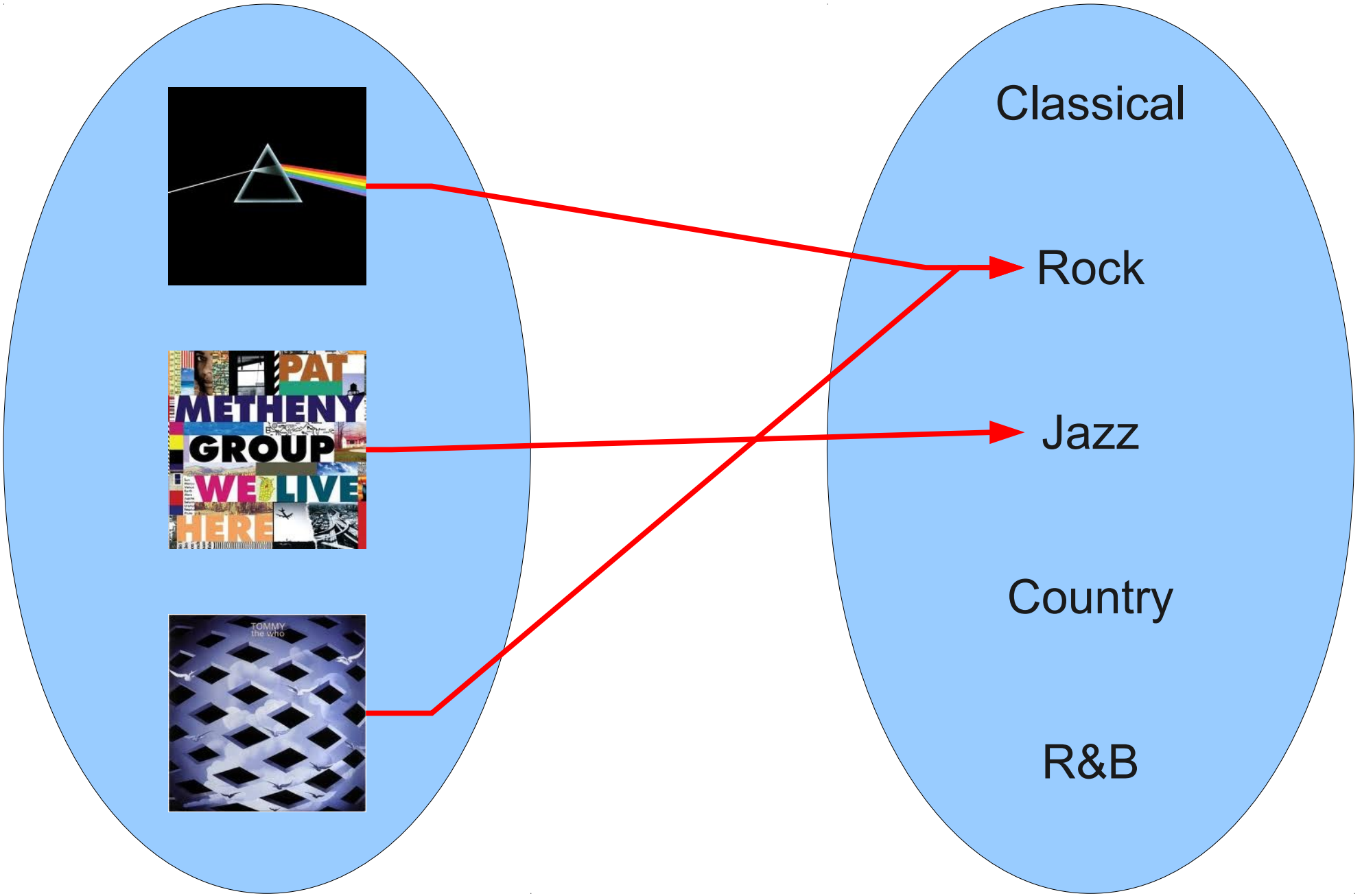

Classical
Rock
Jazz
Country
R&B
PAT
METHENY
GROUP
WE LIVE
HERE
TOMMY
the who

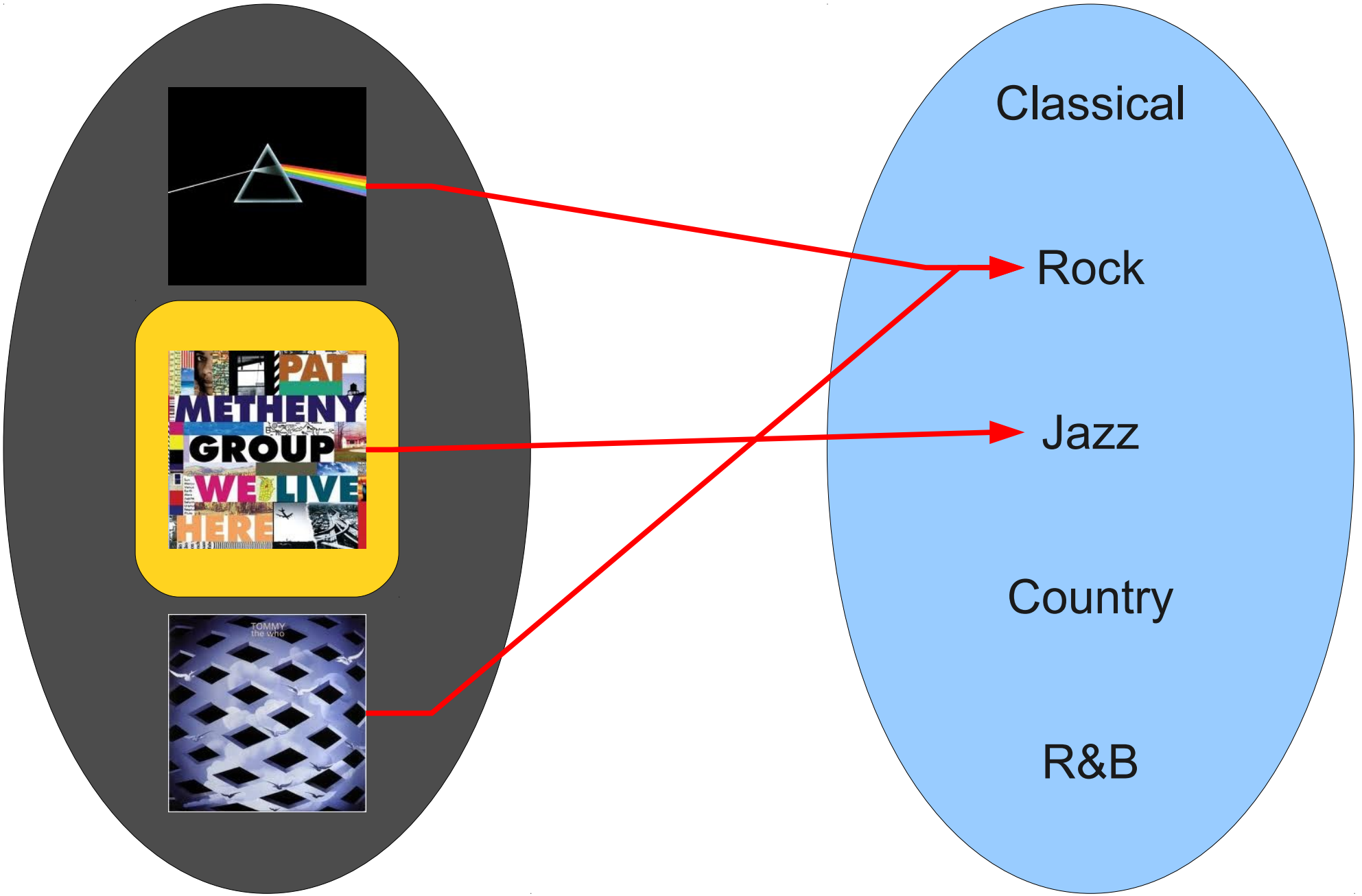

Classical
Rock
Jazz
Country
R&B
PAT
METHENY
GROUP
WE LIVE
HERE
TOMMY
the who

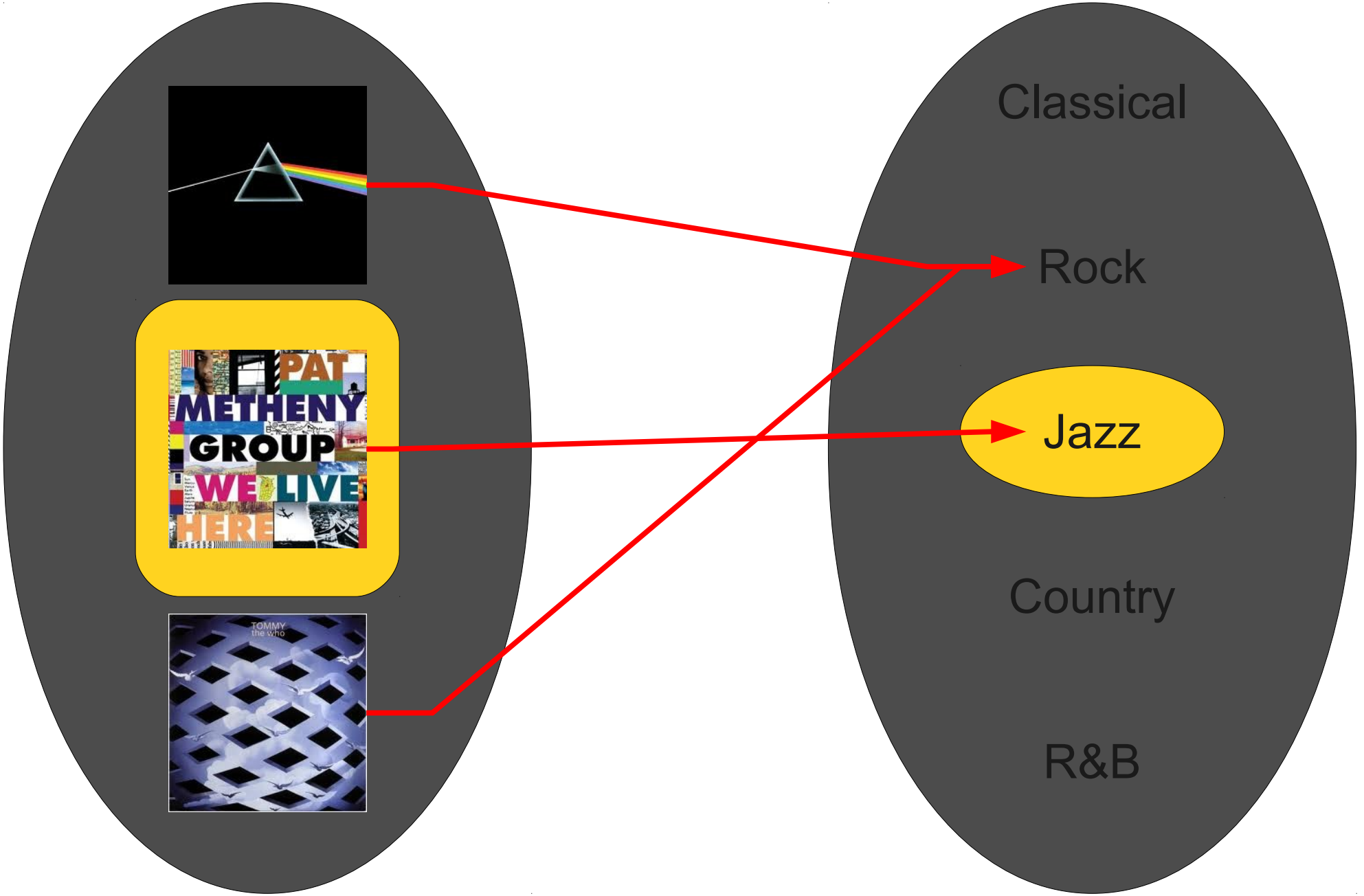


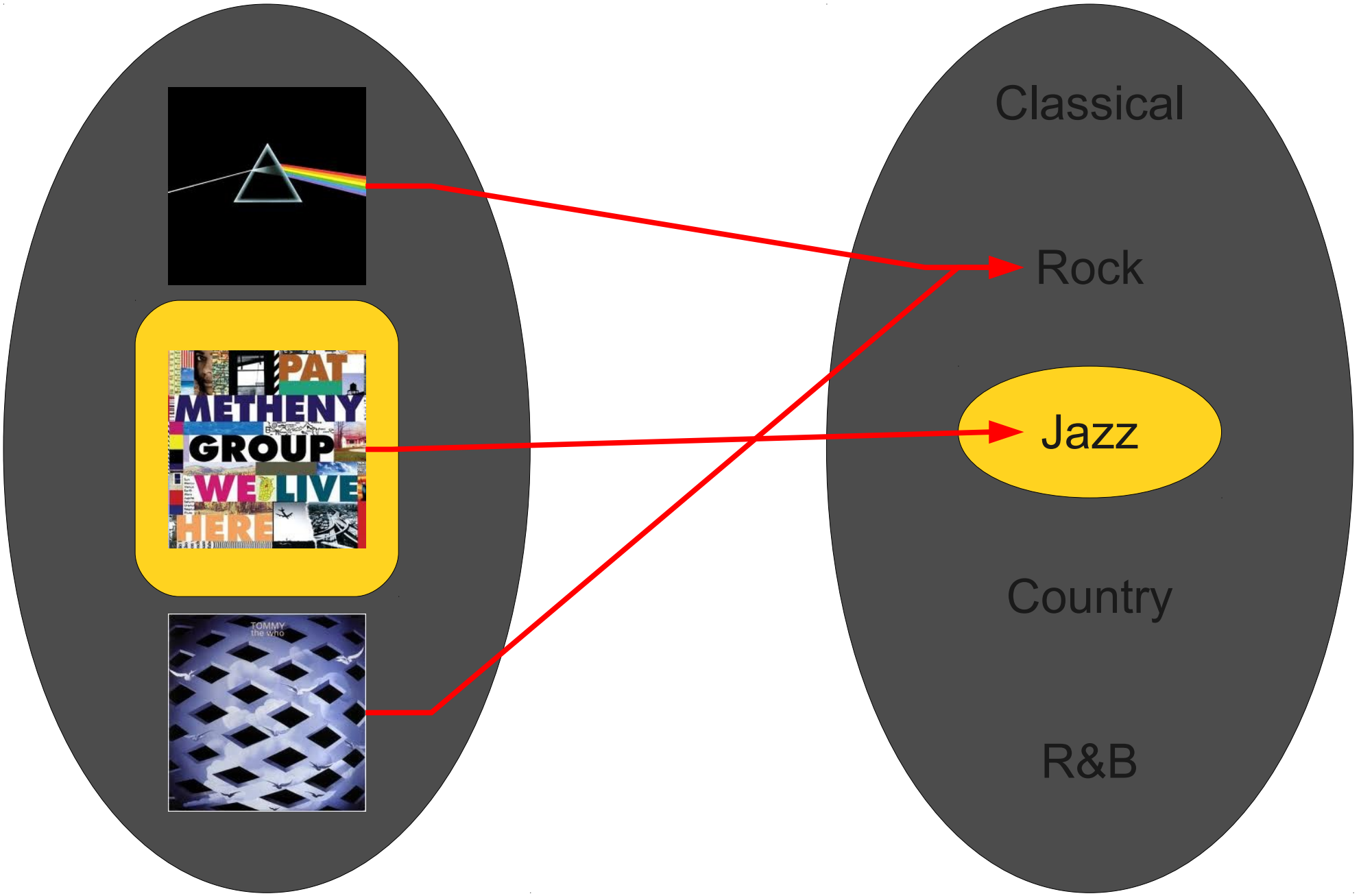
The **image** of a function $f : A \to B$ is the set

$$f[A] = \{ y \mid \exists x \in A.\ f(x) = y \}$$


Classical
Rock
Jazz
Country
R&B
PAT
METHENY
GROUP
WE LIVE
HERE
TOMMY
the who

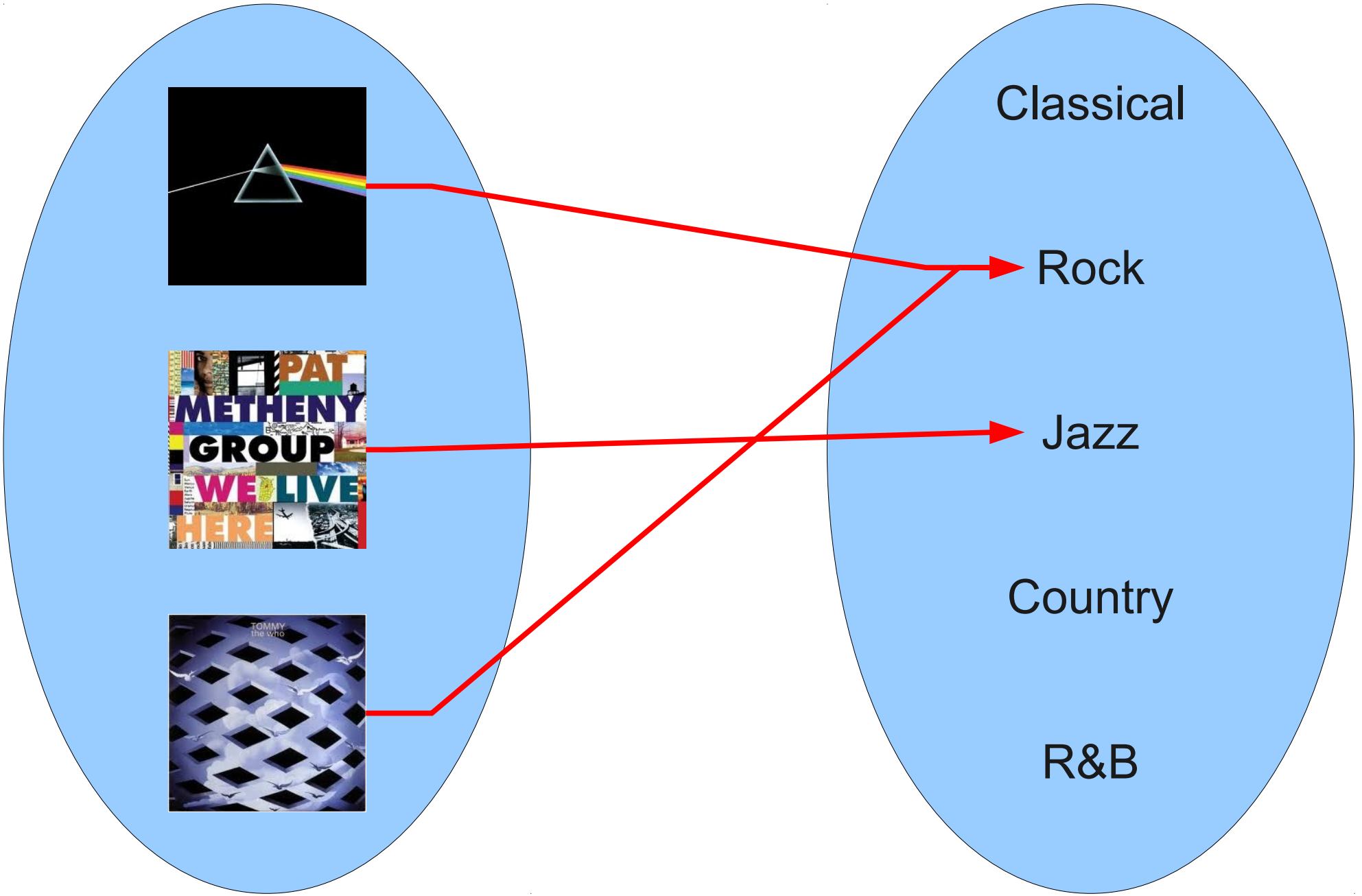

Classical
Rock
Jazz
Country
R&B
PAT
METHENY
GROUP
WE LIVE
HERE
TOMMY
the who

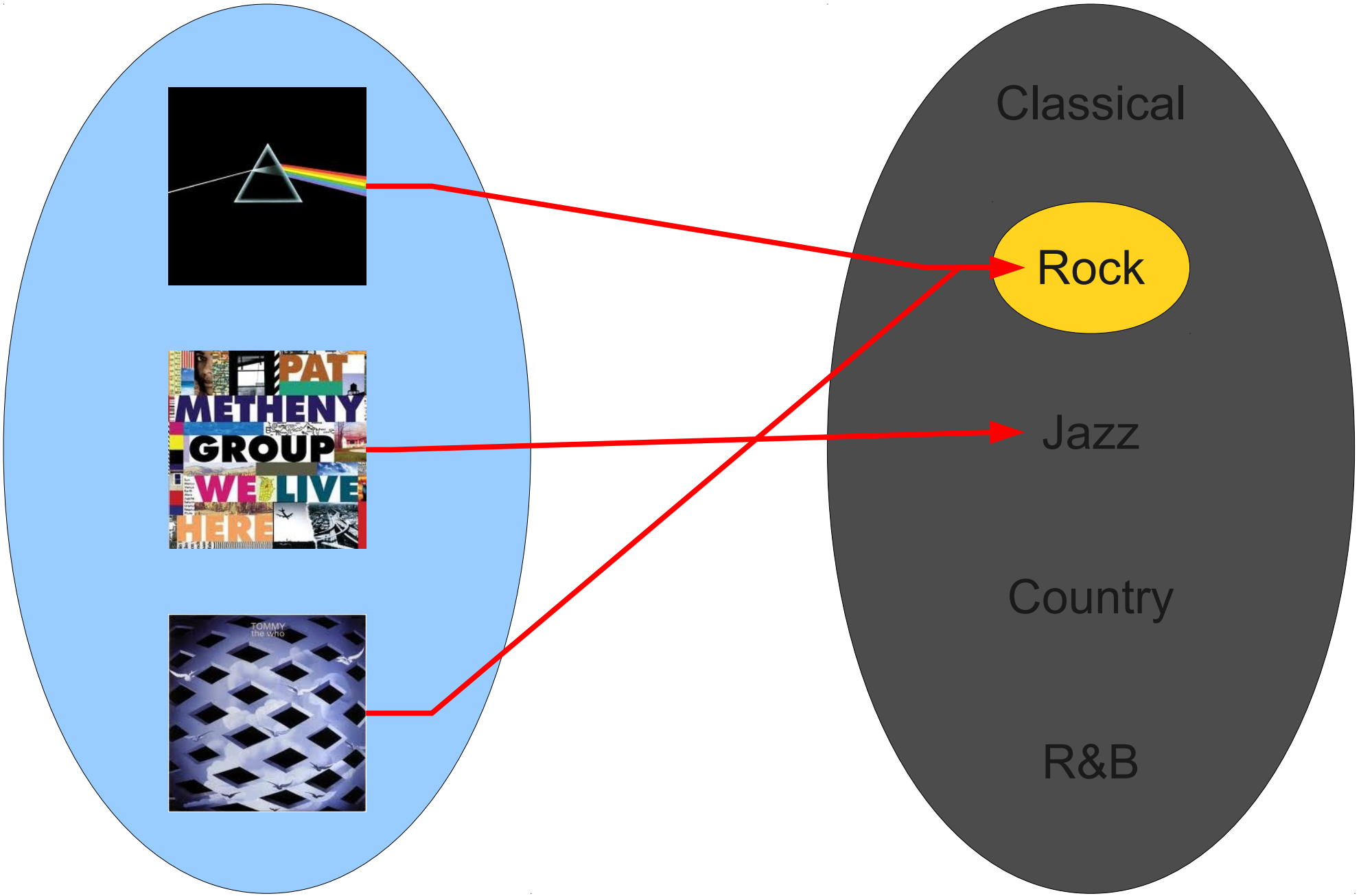

Classical
Rock
Jazz
Country
R&B
PAT
METHENY
GROUP
WE LIVE
HERE
TOMMY
the who

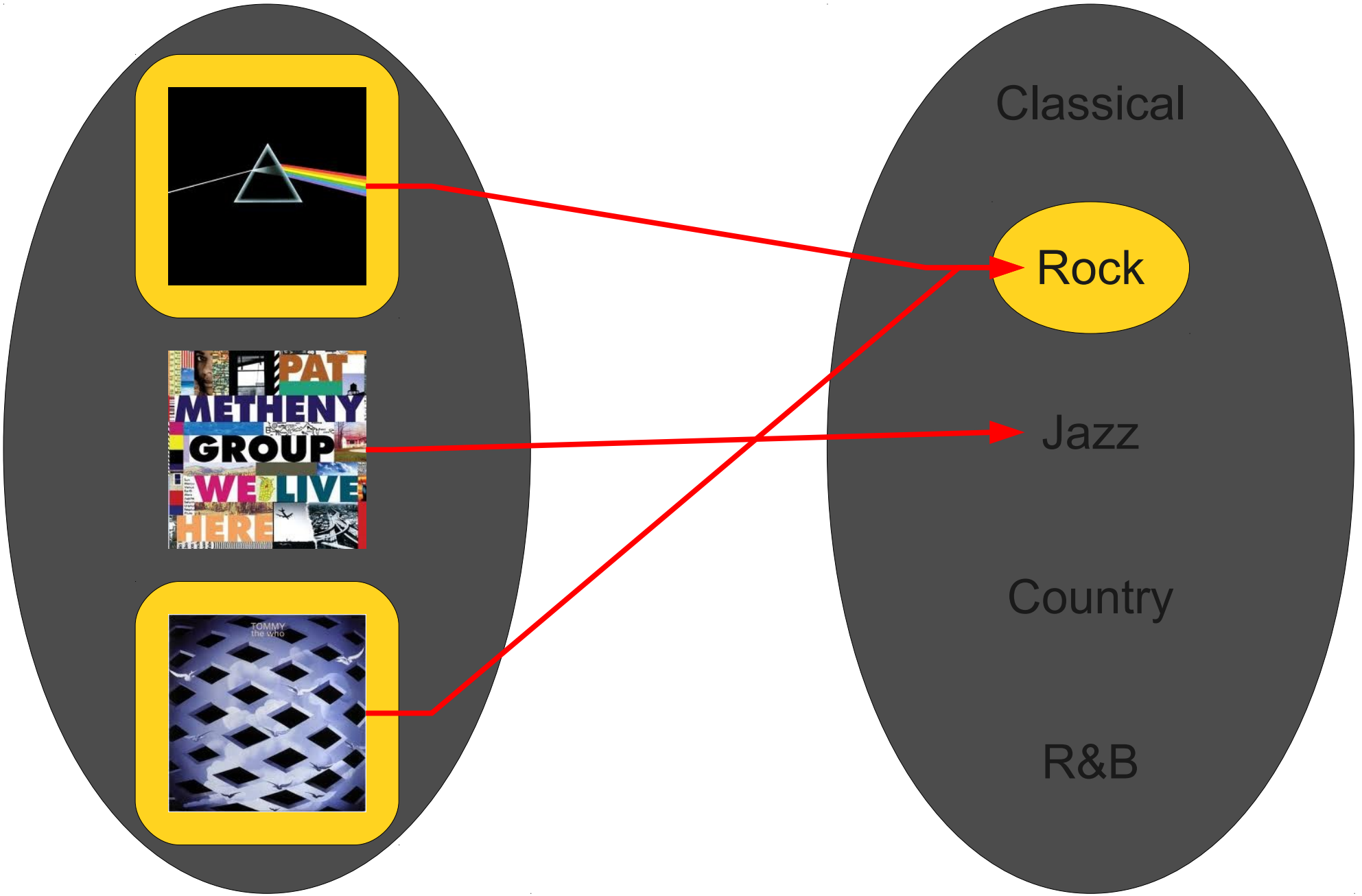


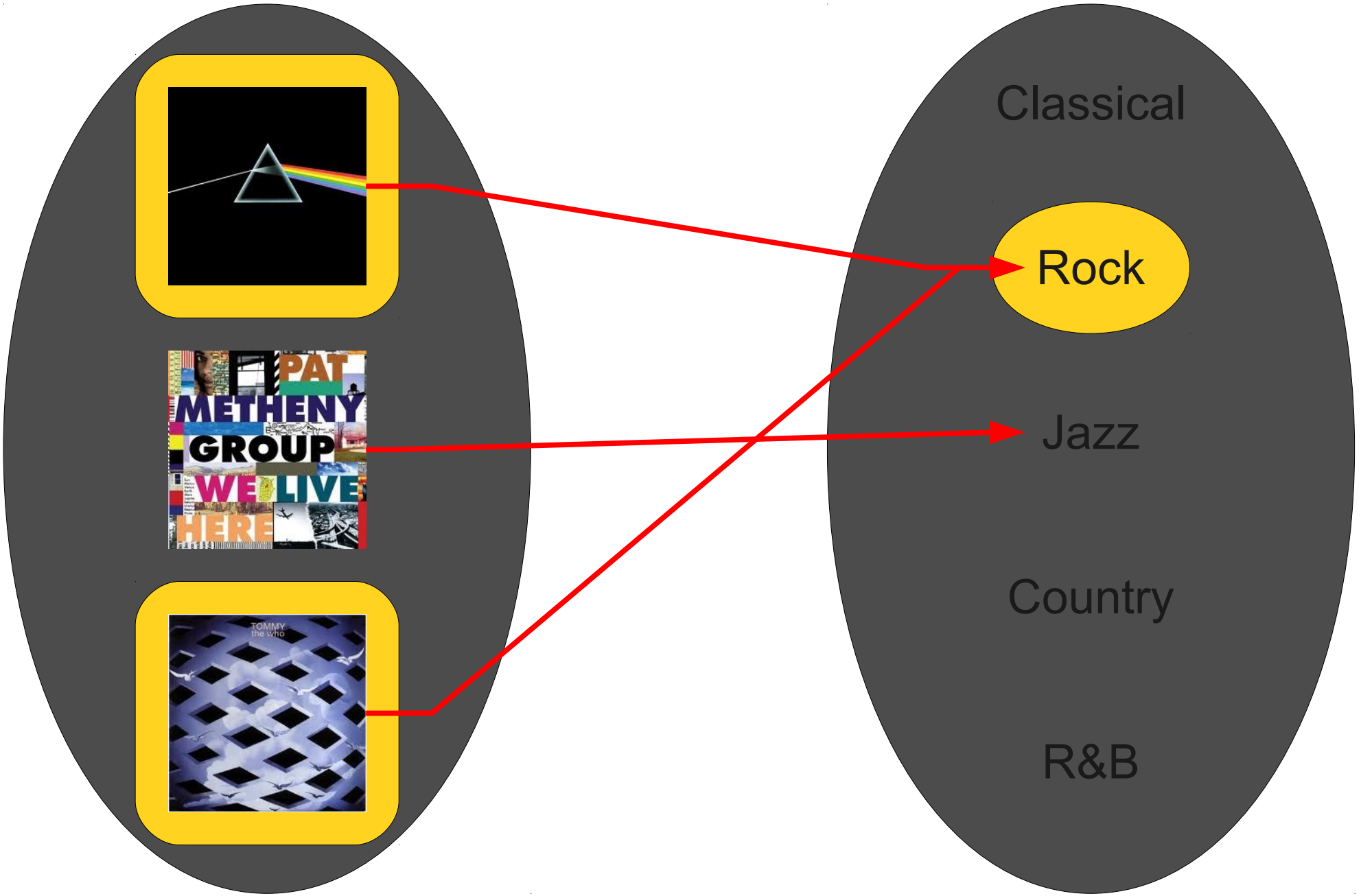
The **image** of a set $X \subseteq A$ is the set

$$f[X] = \{ y \mid \exists x \in X.\ f(x) = y \}$$


Classical
Rock
Jazz
Country
R&B
PAT
METHENY
GROUP
WE LIVE
HERE
TOMMY
the who

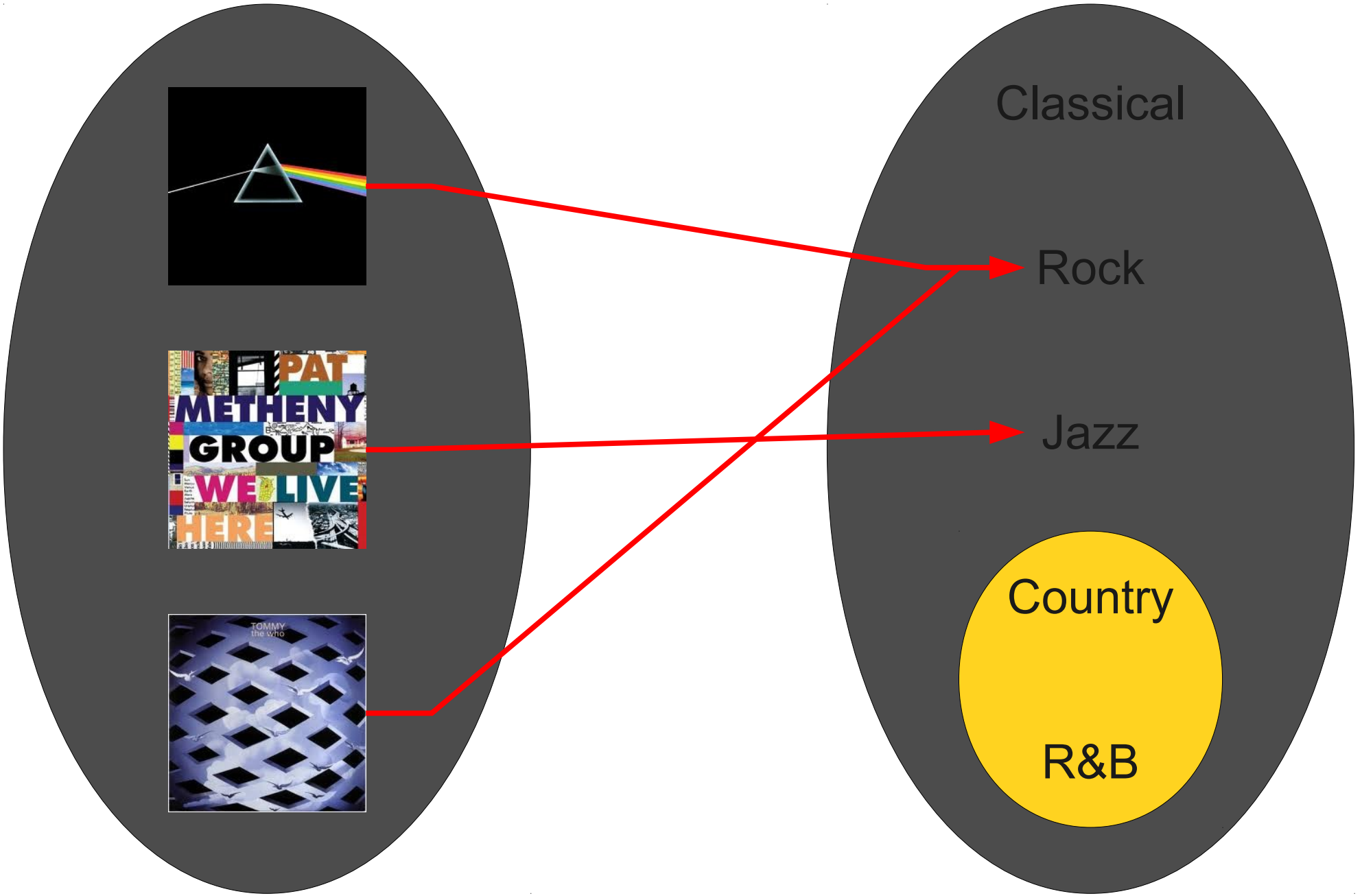

PAT
METHENY
GROUP
WE LIVE
HERE
TOMMY
the who
Classical
Rock
Jazz
Country
R&B

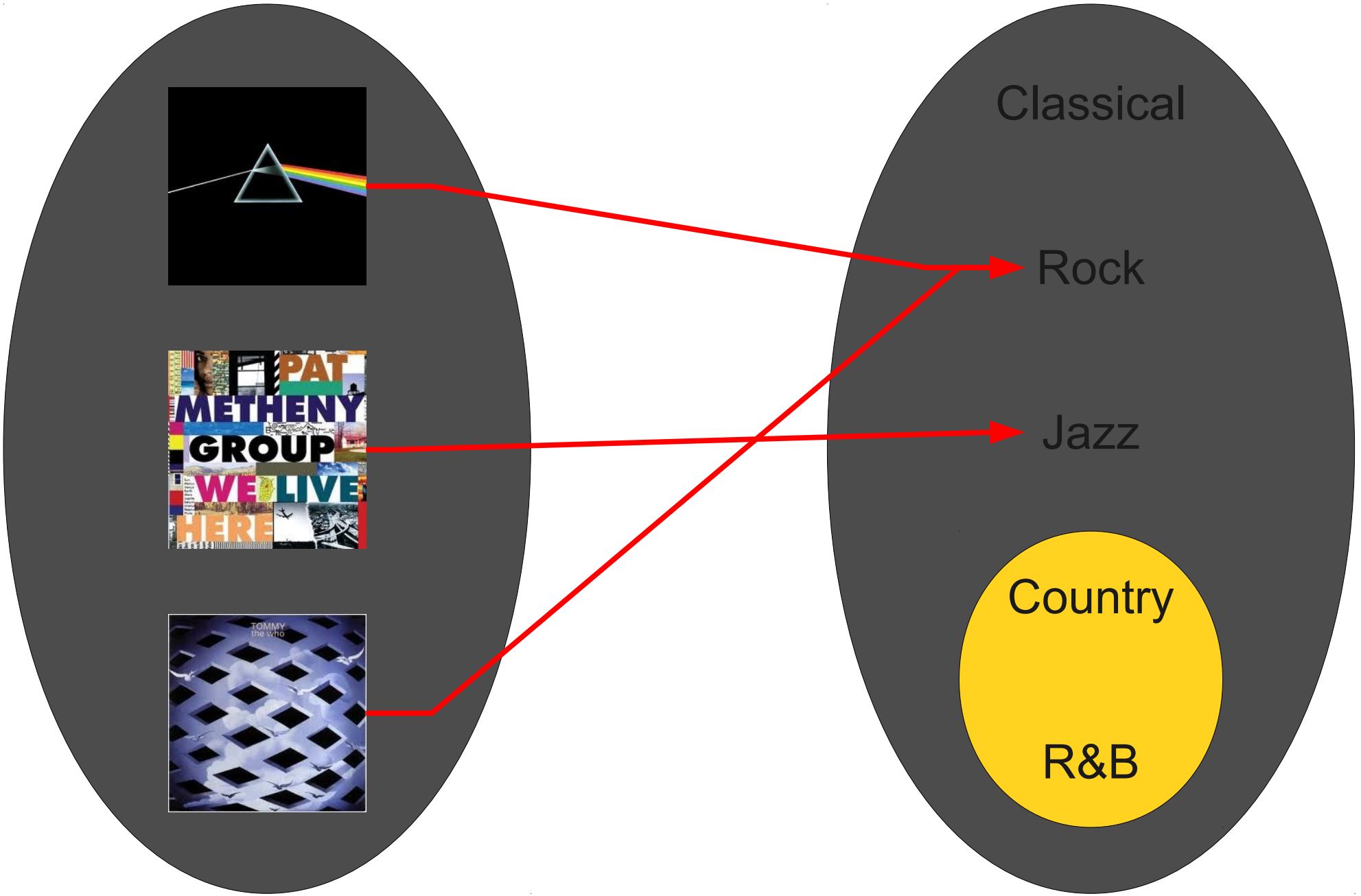

PAT
METHENY
GROUP
WE LIVE
HERE
TOMMY
the who
Classical
Rock
Jazz
Country
R&B



The **preimage** of a set $Y \subseteq B$ is the set

$$f^{-1}[Y] = \{ x \mid \exists y \in Y.\ f(x) = y \}$$


Classical
Rock
Jazz
Country
R&B
PAT
METHENY
GROUP
WE LIVE
HERE
TOMMY
the who



The preimage of a set may be empty.

Images and Preimages

- Let $f : A \to B$.
- The **image** of a set $X \subseteq A$ is the set

$$f[X] = \{ y \mid \exists x \in X.\ f(x) = y \}$$

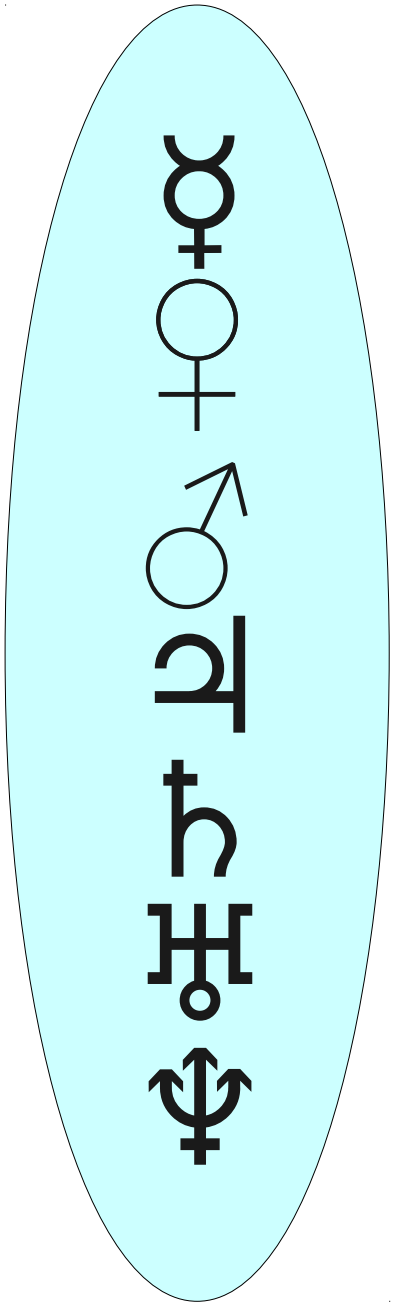
- The **preimage** of a set $Y \subseteq B$ is the set

$$f^{-1}[Y] = \{ x \mid \exists y \in Y.\ f(x) = y \}$$

- The term **range** is sometimes used to refer to the image of A, though sometimes range refers to the entire codomain.

☿
♀
♂
♃
♄
♅
♆

☿
♀
♂
♃
♄
♅
♆



Mercury

Venus

Earth

Mars

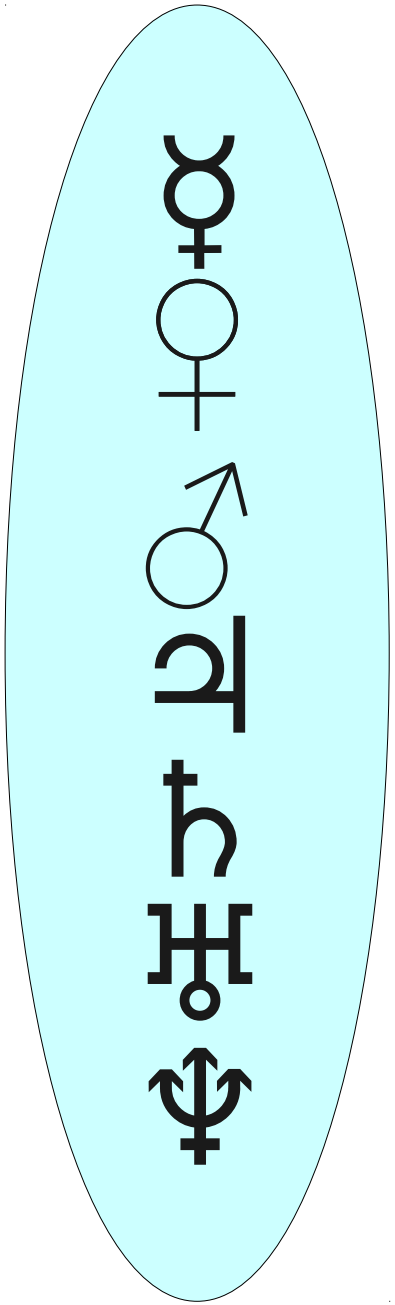
Jupiter

Saturn

Uranus

Neptune

Pluto



Mercury

Venus

Earth

Mars

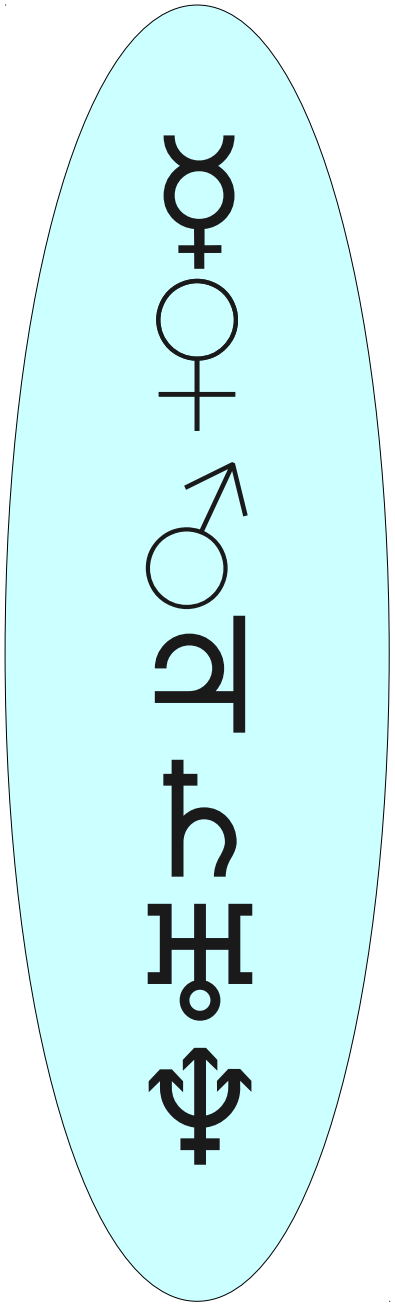
Jupiter

Saturn

Uranus

Neptune

Pluto



Mercury

Venus

Earth

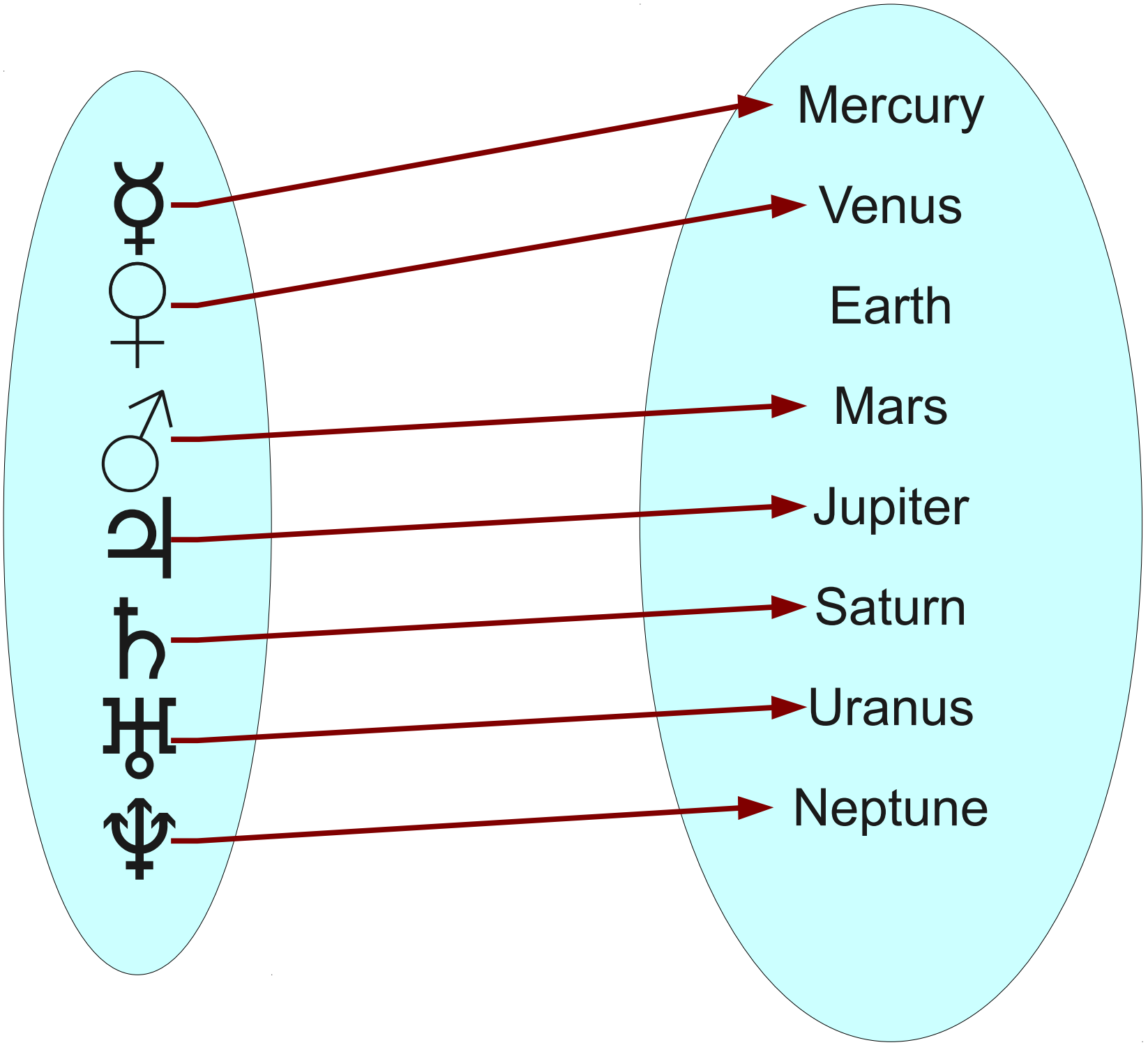
Mars

Jupiter

Saturn

Uranus

Neptune

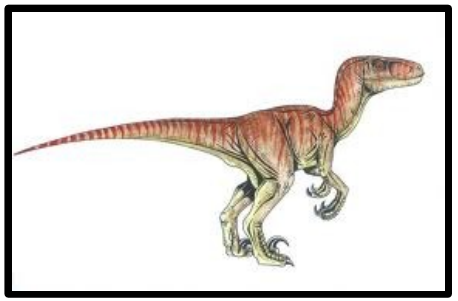

☿
♀
♂
♃
♄
♅
♆
Mercury
Venus
Earth
Mars
Jupiter
Saturn
Uranus
Neptune

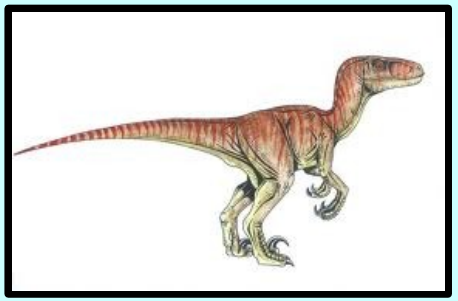
Injective Functions

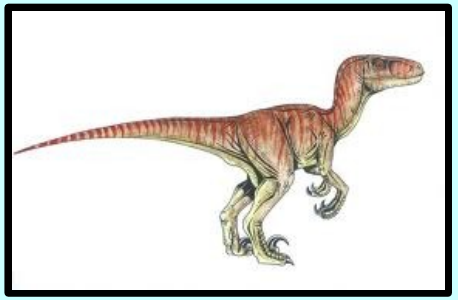
- A function $f : A \to B$ is called **injective** (or **one-to-one**) if each element of the codomain has at most one element of the domain associated with it.
  - A function with this property is called an **injection**.
- More formally:

$$\forall x \in A.\ \forall x' \in A.\ (f(x) = f(x') \to x = x'))$$

- An intuition: injective functions assign names from B to objects in A.



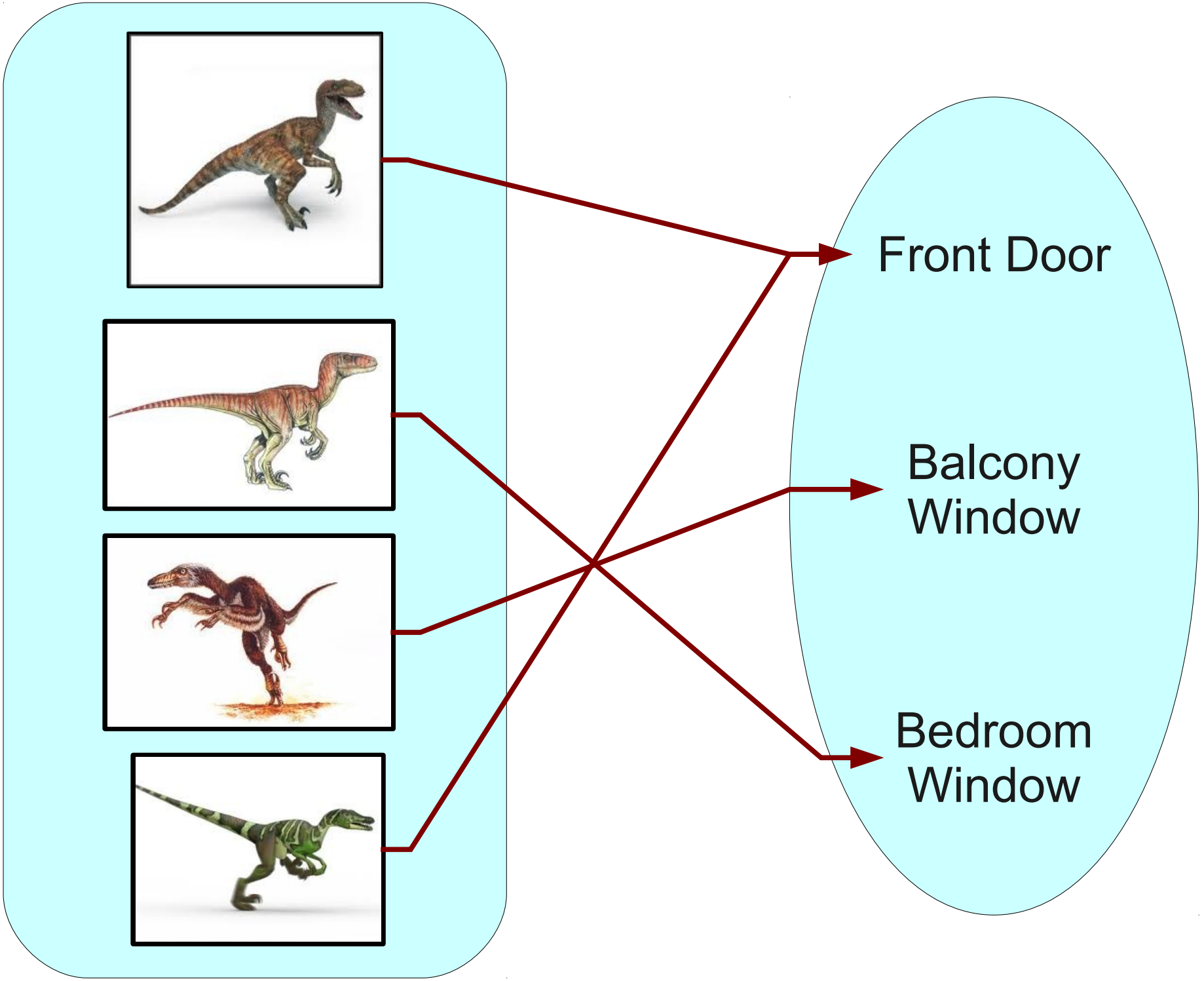




Front Door

Balcony Window

Bedroom Window


Front Door
Balcony Window
Bedroom Window

Surjective Functions

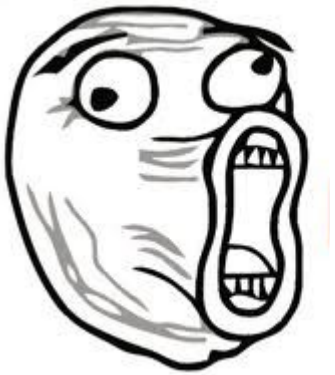
- A function $f : A \to B$ is called **surjective** (or **onto**) if each element of the codomain has at least one element of the domain associated with it.
  - A function with this property is called a **surjection**.
- More formally:

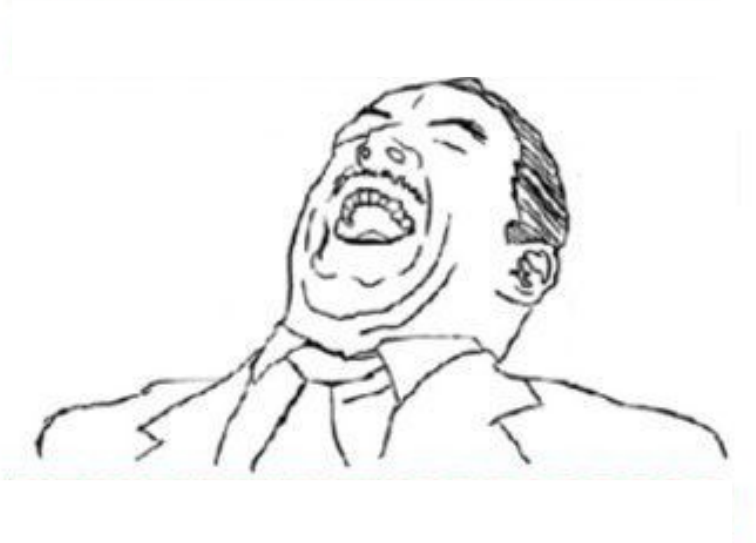
$$\forall y \in B.\ \exists x \in A.\ f(x) = y$$

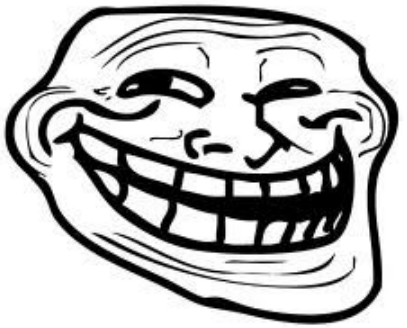
- An intuition: surjective functions cover every element of B with at least one element of A.

Recap

- An injective function associates **at most** one element of the domain with each element of the codomain.
- A surjective function associates **at least** one element of the domain with each element of the codomain.
- What about functions that associate **exactly one** element of the domain with each element of the codomain?



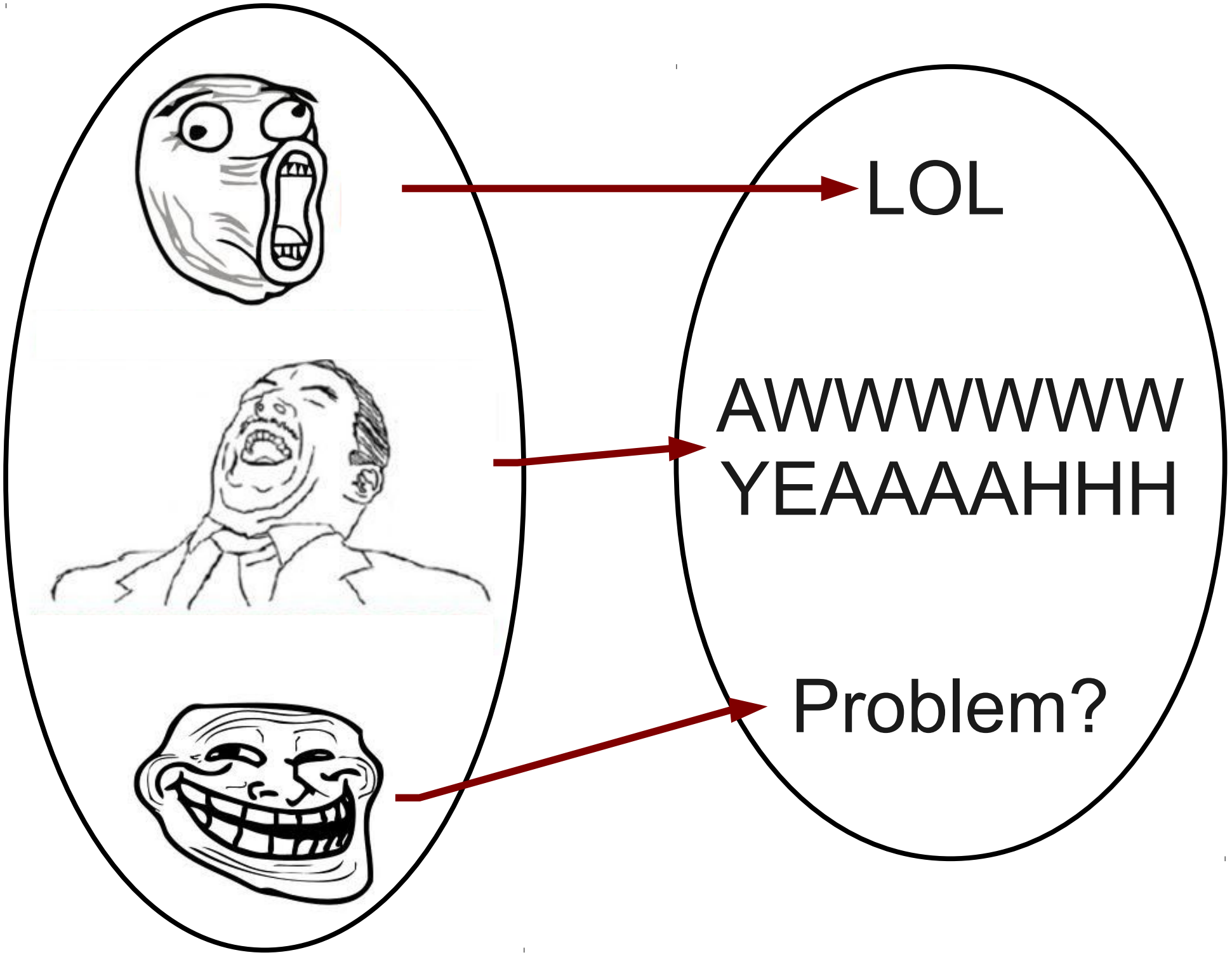




LOL

AWWWWWW
YEAAAAHHH

Problem?


LOL
AWWWWWW
YEAAAAHHH
Problem?

Bijections

- A function that associates each element of the codomain with a unique element of the domain is called **bijective**.
  - Such a function is a **bijection**.
- Formally, a bijection is a function that is both **injective** and **surjective**.
- A bijection is a one-to-one correspondence between two sets.

Functions and Sets

Cardinality

- Recall (from *lecture one!*) that the **cardinality** of a set is the number of elements it contains.
  - Denoted |S|.
- |{1, 2, 3}| = 3
- |{100, 200, 300}| = 3
- $|\mathbb{N}| = \aleph_0$

Functions and Cardinality

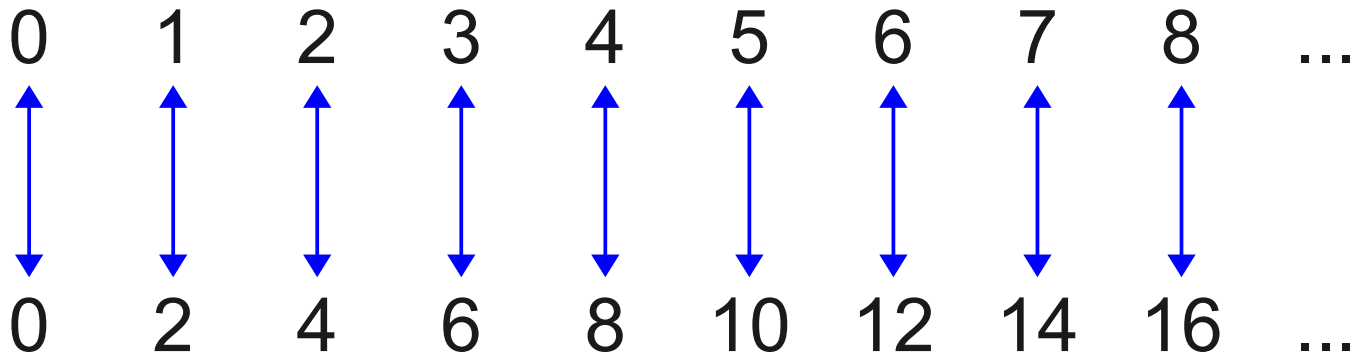
- Formally:

$$|S| = |T| \equiv \text{There is a bijection } f : S \rightarrow T$$

Functions and Cardinality

- Formally:

$$|S| = |T| \equiv \text{There is a bijection } f : S \rightarrow T$$



$$f(n) = 2n$$

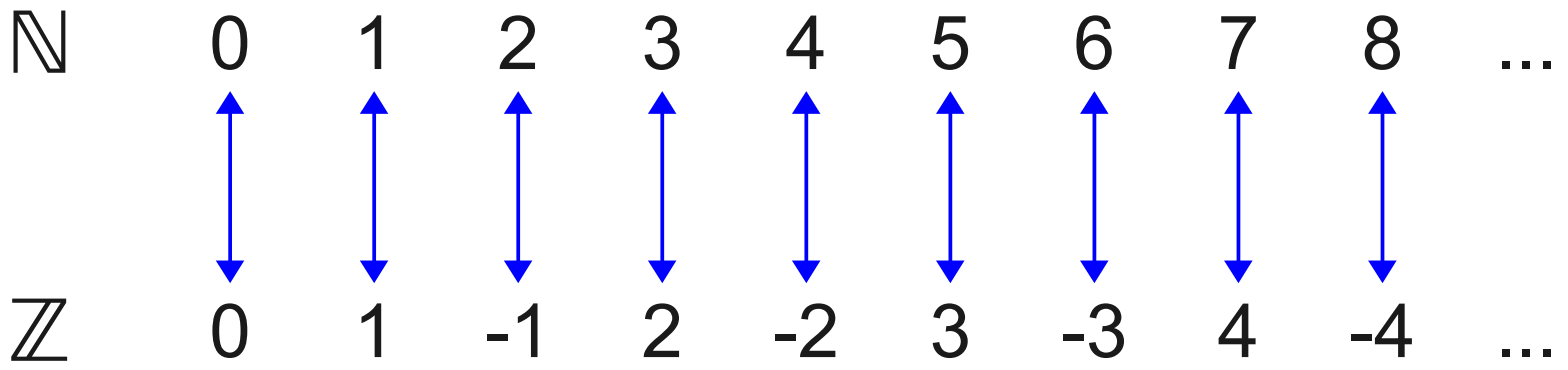
$$S = \{ x \mid x \in \mathbb{N} \text{ and } x \text{ is even} \}$$

$$|S| = |\mathbb{N}| = \aleph_0$$

Functions and Cardinality

- Formally:

$|S| = |T| \equiv$ There is a bijection $f : S \rightarrow T$



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

$|\mathbb{Z}| = |\mathbb{N}| = \aleph_0$

Comparing Cardinalities

- $|S| = |T|$ is defined using bijections.
- We define $|S| \leq |T|$ as follows:

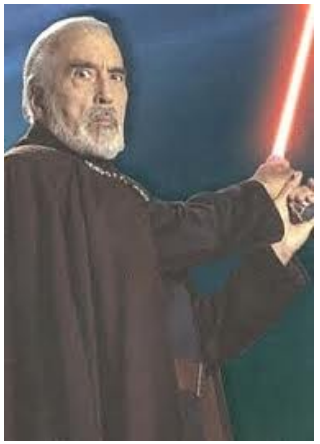
  $|S| \leq |T| \equiv$ There is a **surjection** $f : T \to S$

- Intuitively, we can "cover" set S using elements of T.
- Equivalently:

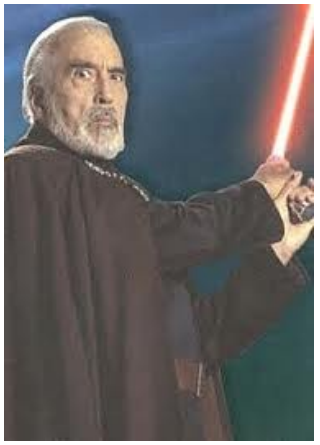
  $|S| \leq |T| \equiv$ There is an **injection** $f : S \to T$

- Intuitively, there are enough elements of T to assign one to each of the elements of S without running out.

Compositions

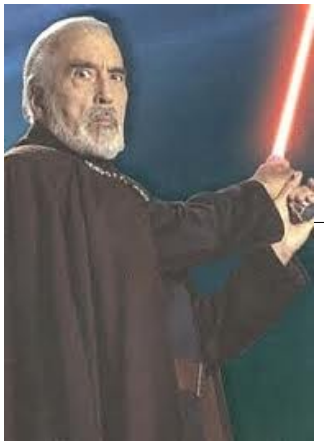


STAR WARS



THE LORD OF THE RINGS

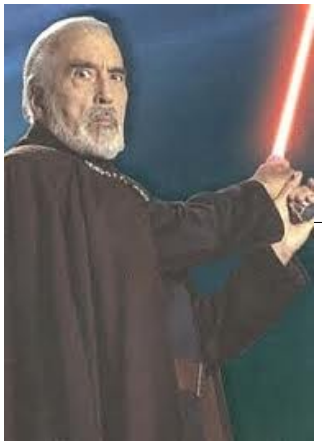
PORTAL™



STAR WARS

THE LORD OF THE RINGS

PORTAL™

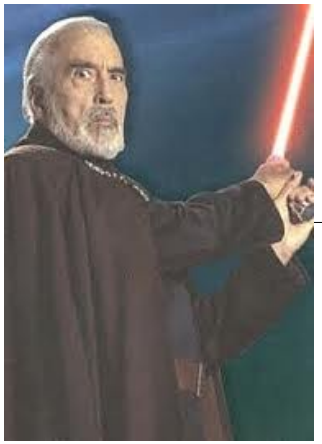


STAR WARS

THE LORD OF THE RINGS

PORTAL™

VALVE®



STAR WARS

THE LORD OF THE RINGS

PORTAL™

VALVE®

STAR WARS

THE LORD OF THE RINGS

PORTAL™

VALVE®

VALVE®

Function Composition

- Let $f : A \to B$ and $g : B \to C$
- The function $g \circ f : A \to C$ is defined as

$$(g \circ f)(x) = g(f(x))$$

- Note that f is applied first, but f is on the right side!
- Function composition is **associative**:

$$h \circ (g \circ f) = (h \circ g) \circ f$$

Function Composition

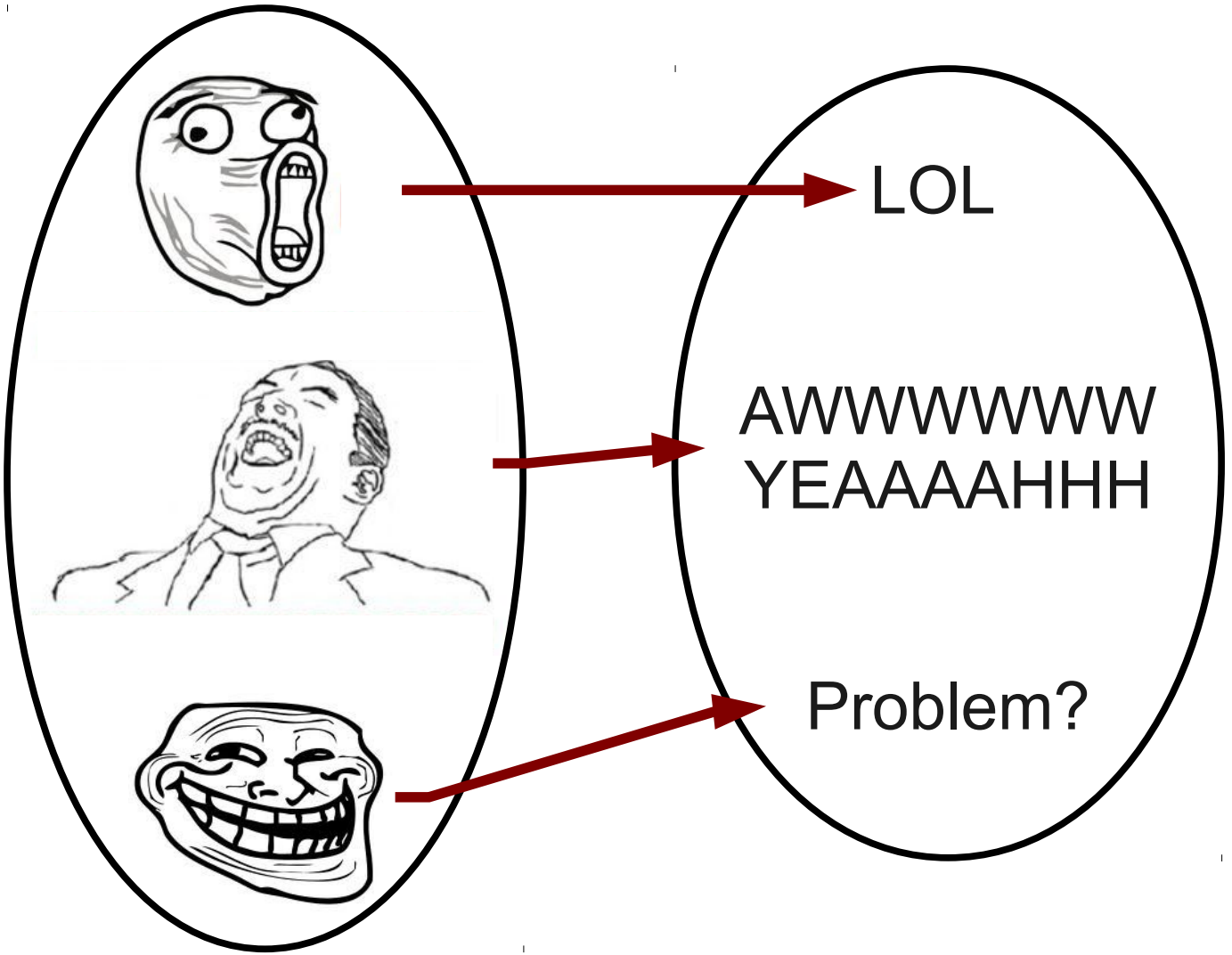
- Suppose $f : A \to A$ and $g : A \to A$.
- Then both $g \circ f$ and $f \circ g$ are defined.
- Does $g \circ f = f \circ g$?
- **In general, no:**
  - Let $f(x) = 2x$
  - Let $g(x) = x + 1$
  - $(g \circ f)(x) = g(f(x)) = g(2x) = 2x + 1$
  - $(f \circ g)(x) = f(g(x)) = f(x + 1) = 2x + 2$

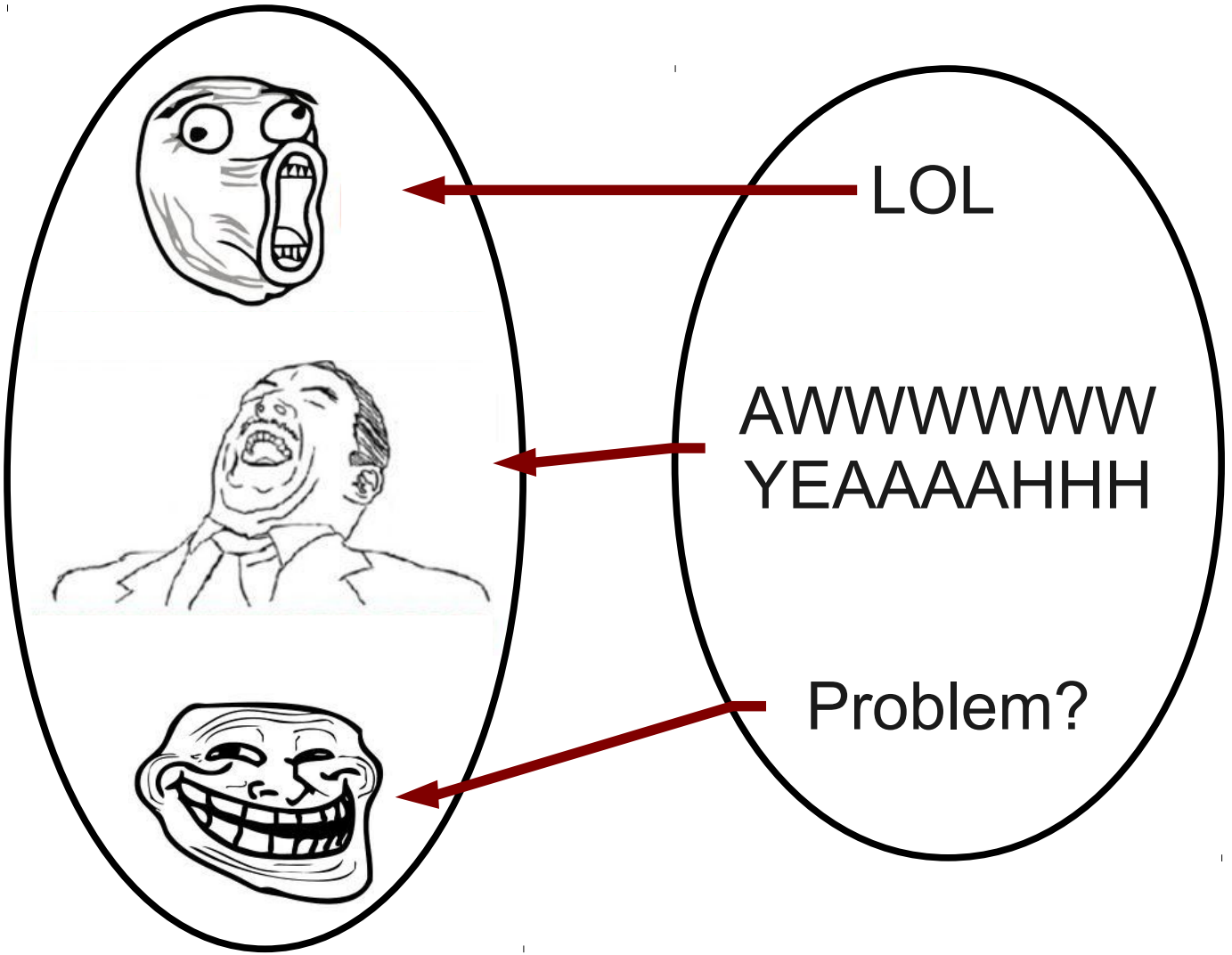
The Identity Function

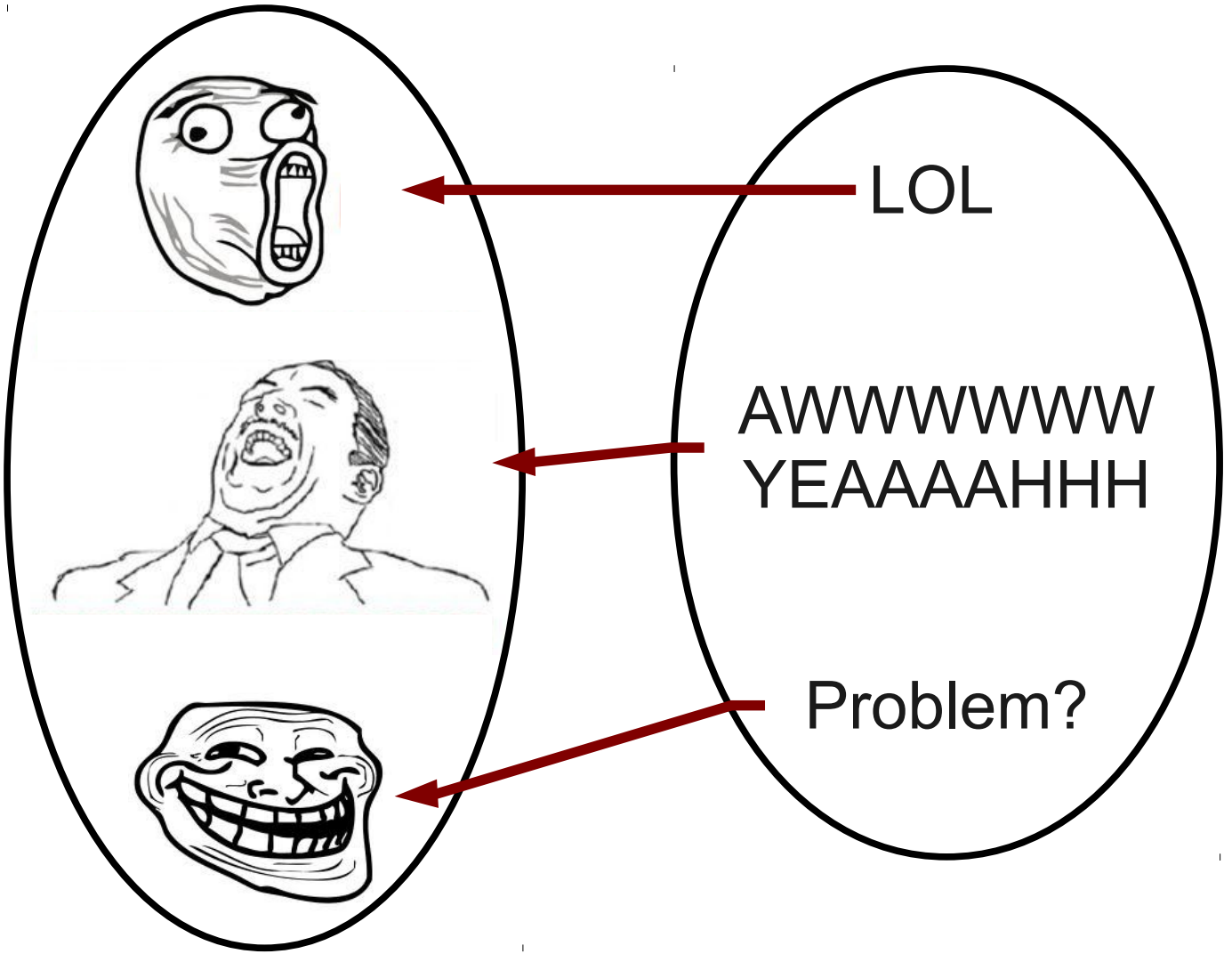
- Given a set A, the **identity function on A** is the special function $id_A : A \to A$ defined as

$$id_A(x) = x$$

- This function is not particularly exciting, but it does have its uses.


LOL
AWWWWWW
YEAAAAHHH
Problem?

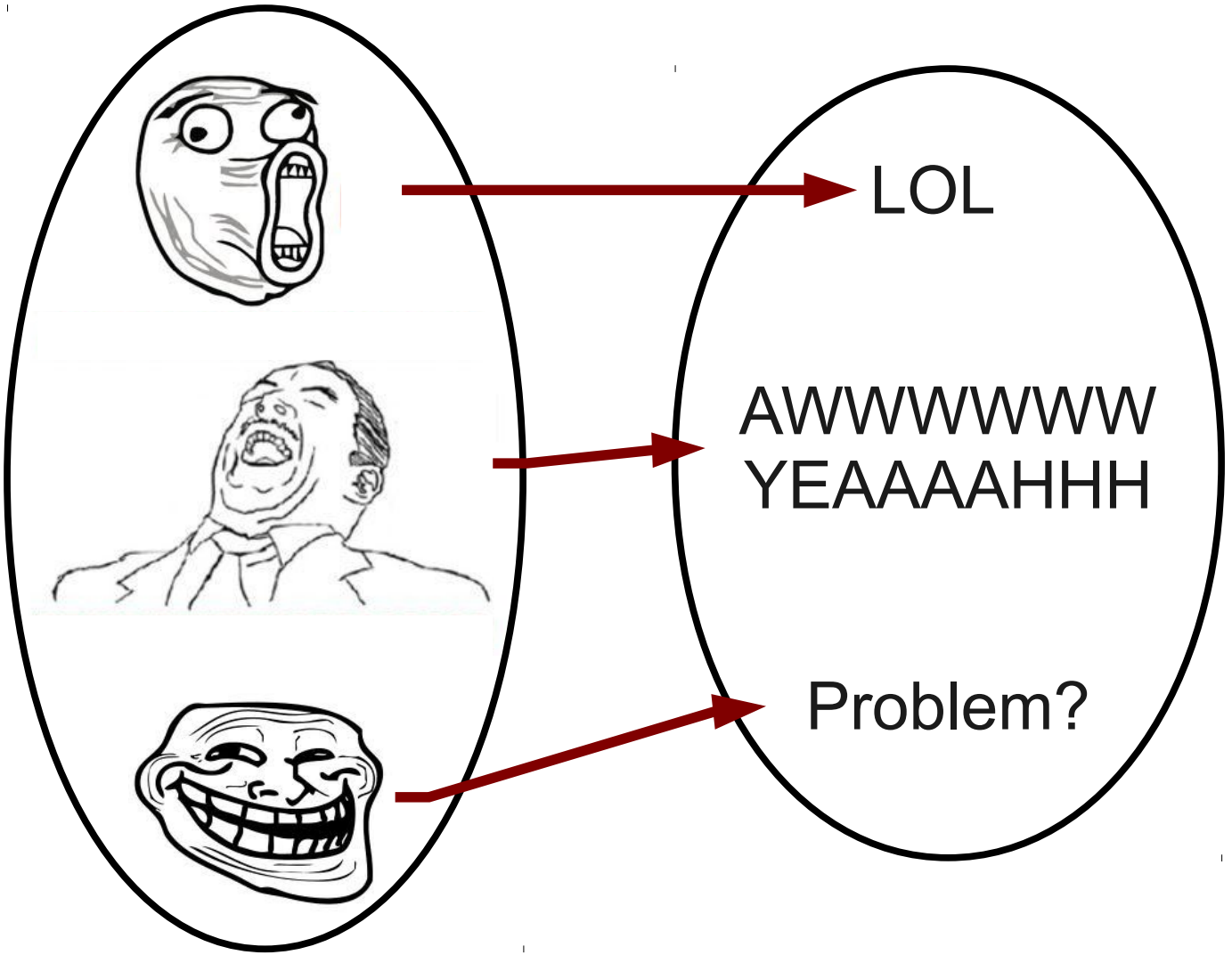

LOL
AWWWWWW
YEAAAAHHH
Problem?

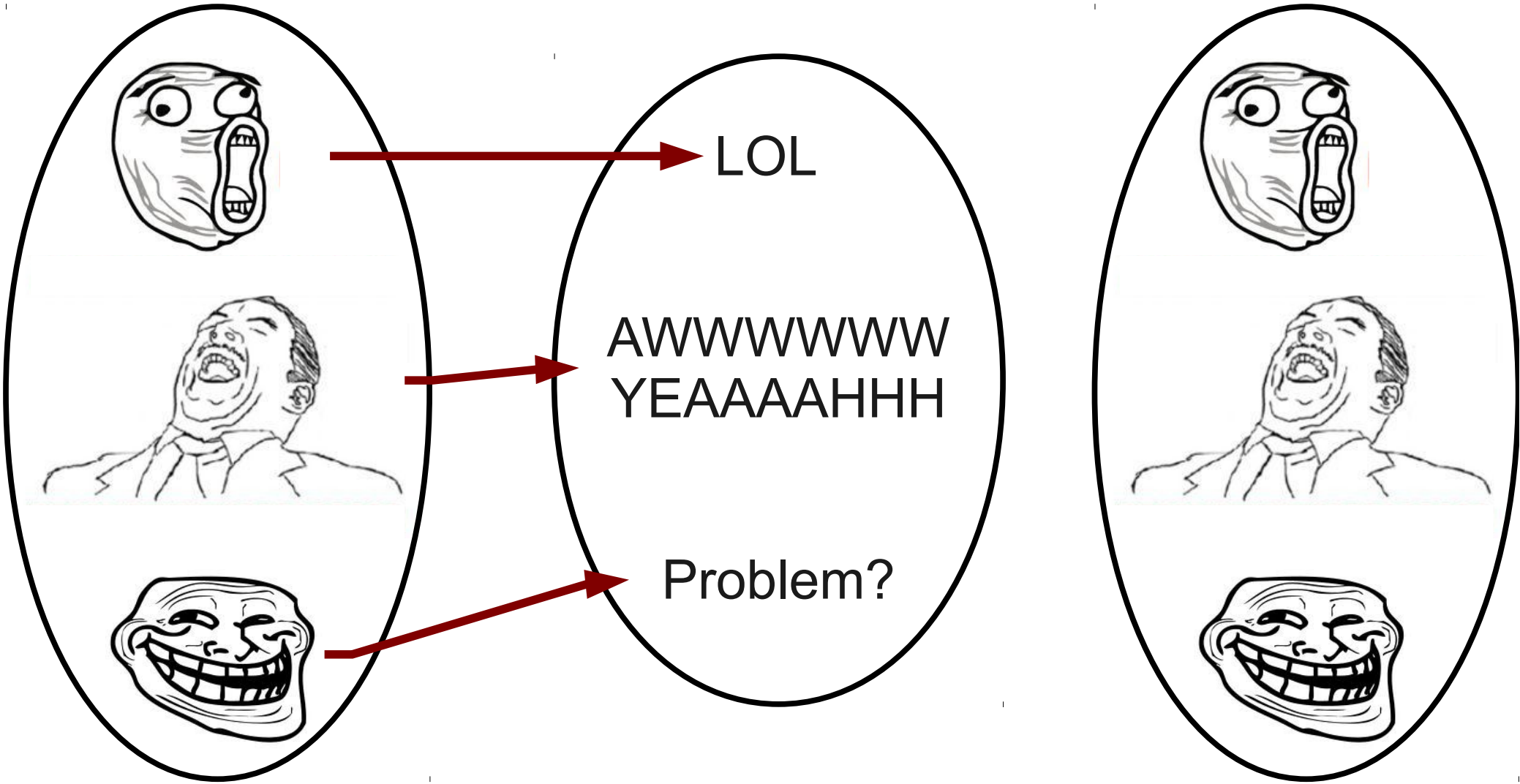


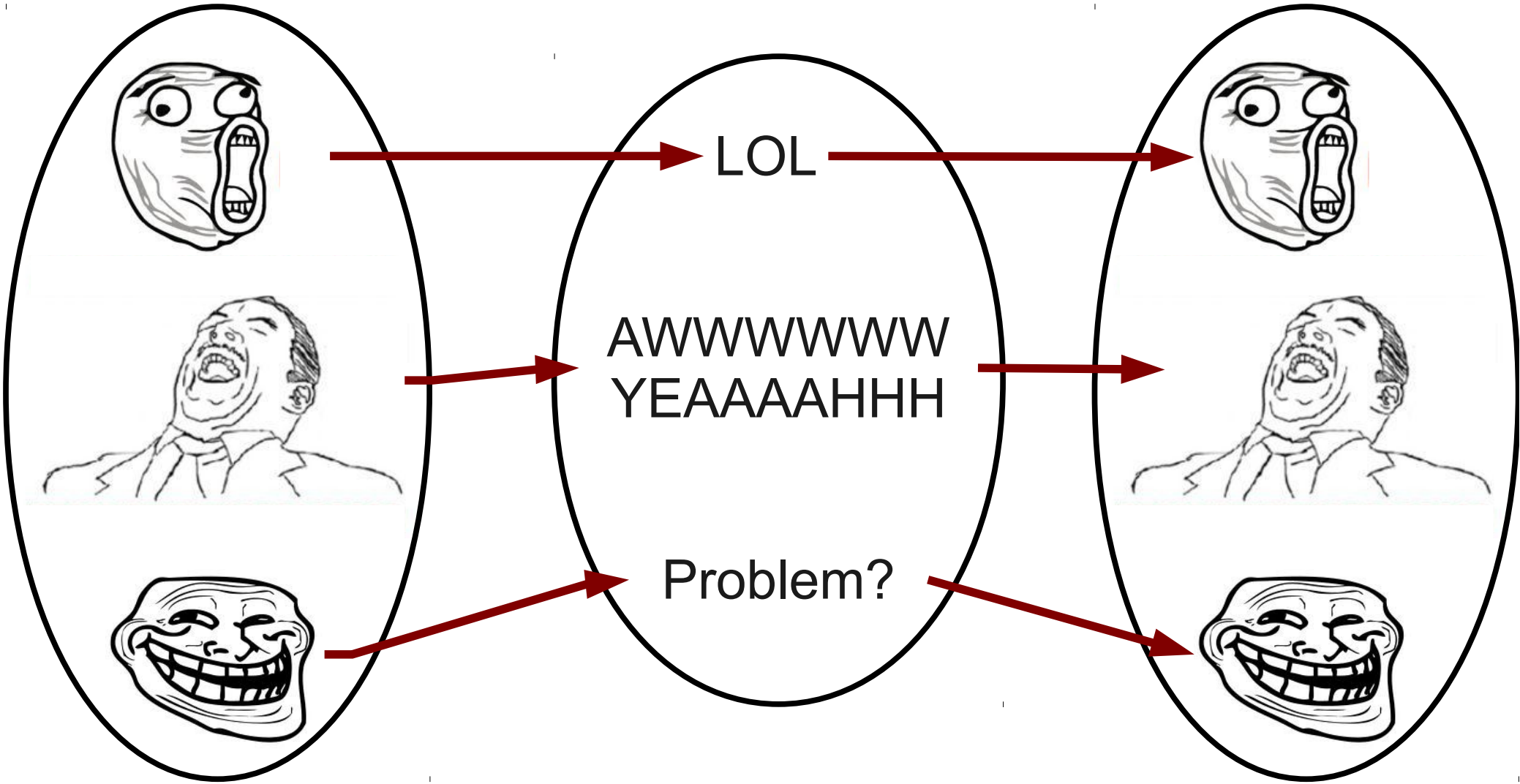
Is this a function?

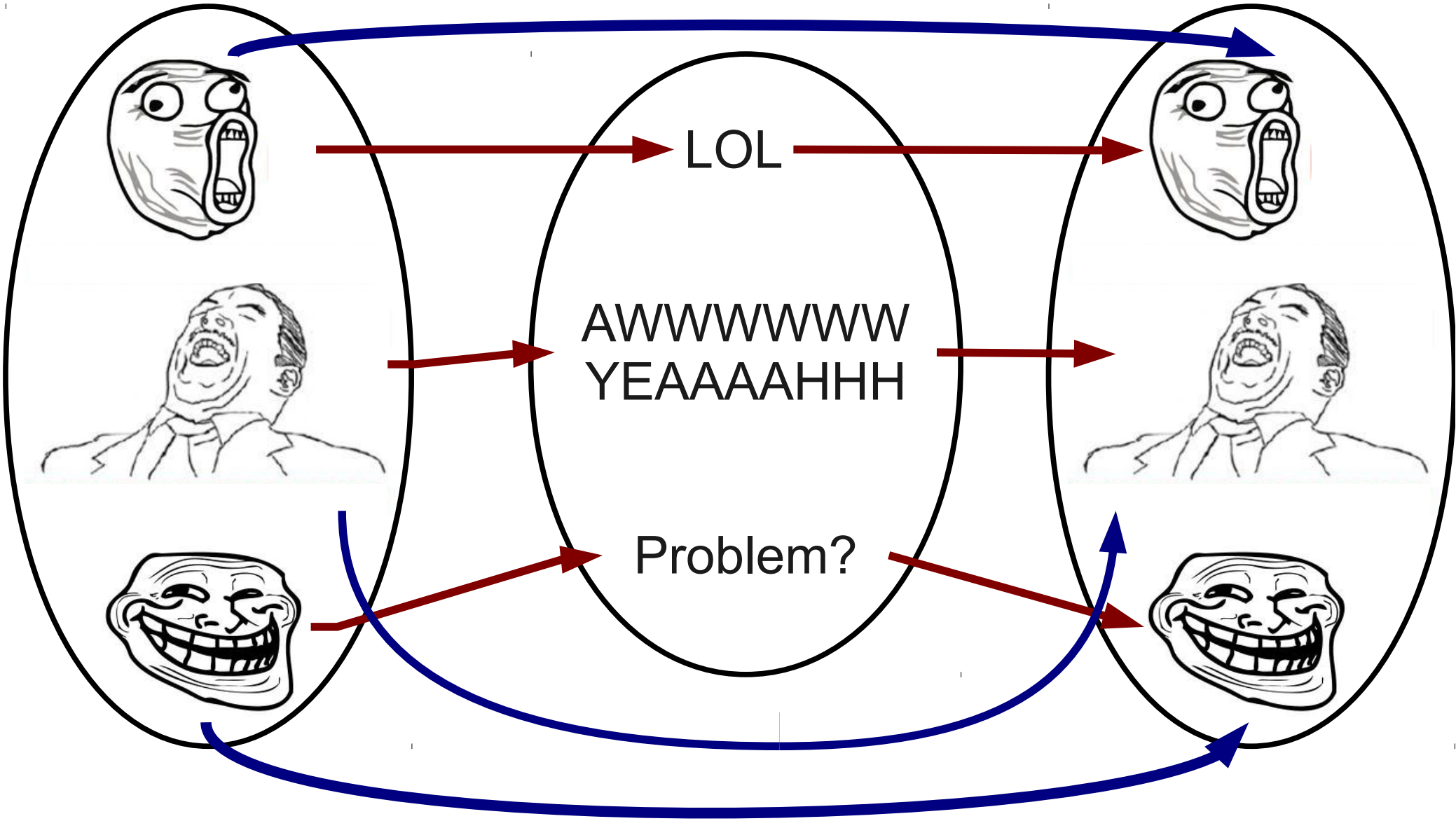
Inverses, Informally

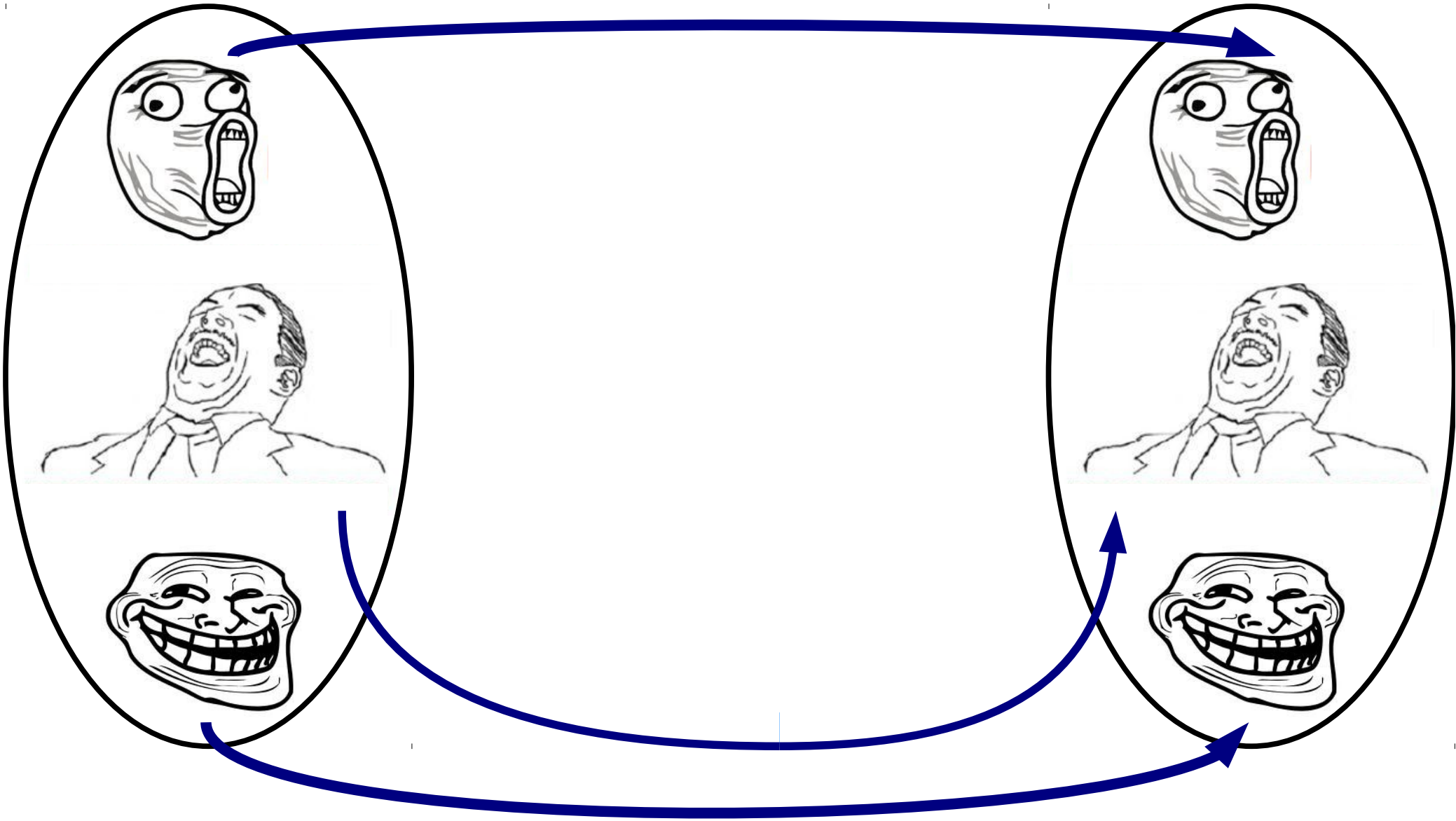
- Informally, the **inverse** of a function $f : A \to B$ is the function $f^{-1} : B \to A$ such that if $f(a) = b$, then $f^{-1}(b) = a$.
- Not all functions have inverses:
- If the function isn't **injective**, then for some $a_0 \neq a_1$, $f(a_0) = b = f(a_1)$.
  - What is $f^{-1}(b)$?
- If the function isn't **surjective**, some element of $b \in B$ has nothing mapping to it.
  - What is $f^{-1}(b)$?
- If a function is not bijective, it is not invertible.


LOL
AWWWWWW
YEAAAAHHH
Problem?


LOL
AWWWWWW
YEAAAAHHH
Problem?


LOL
AWWWWWW
YEAAAAHHH
Problem?


LOL
AWWWWWW
YEAAAAHHH
Problem?



Inverses, Formally

- The **inverse** of a function $f : A \to B$ is a function $f^{-1} : B \to A$ such that

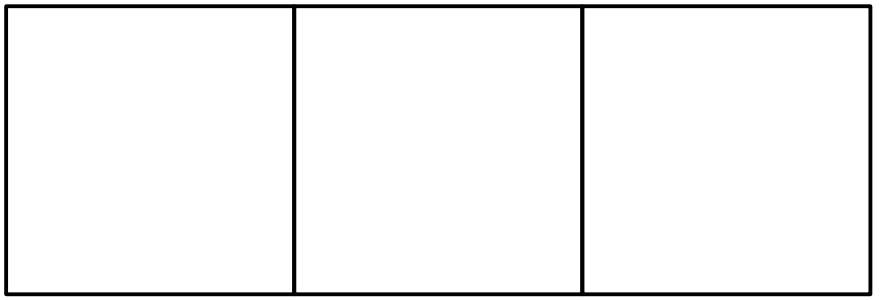
$$f^{-1} \circ f = id_A$$

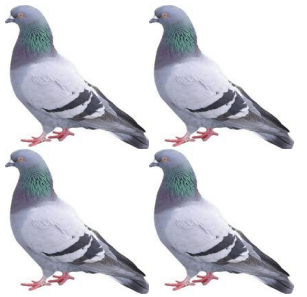
- That is, $f^{-1}(f(x)) = x$
- Inverse functions are **unique**.
- **Theorem**: A function has an inverse iff it is bijective.
  - Interesting exercise: why is this?

The Pigeonhole Principle

The Pigeonhole Principle

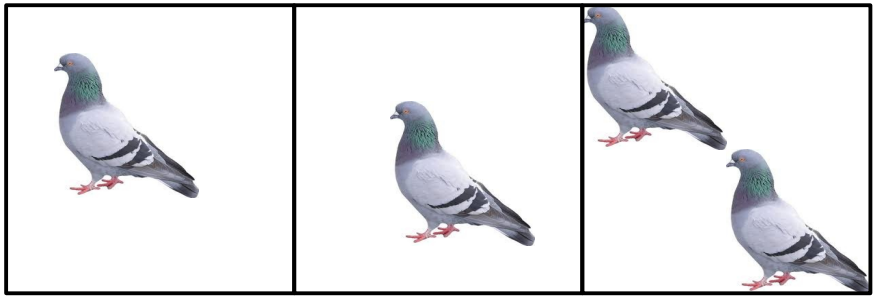
- Suppose that you have n pigeonholes.
- Suppose that you have m > n pigeons.
- If you put the pigeons into the pigeonholes, some pigeonhole will have more than one pigeon in it.





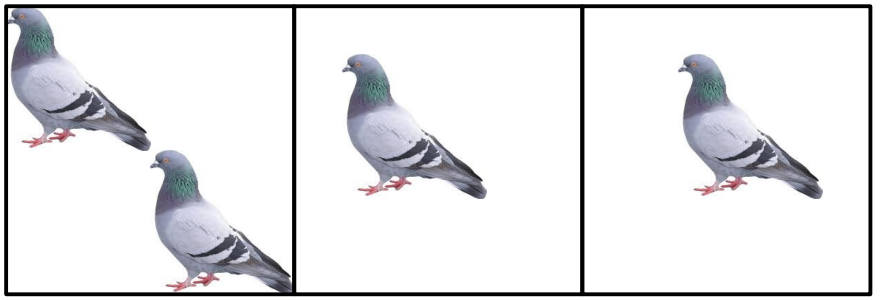
The Pigeonhole Principle

- Suppose that you have n pigeonholes.
- Suppose that you have m > n pigeons.
- If you put the pigeons into the pigeonholes, some pigeonhole will have more than one pigeon in it.



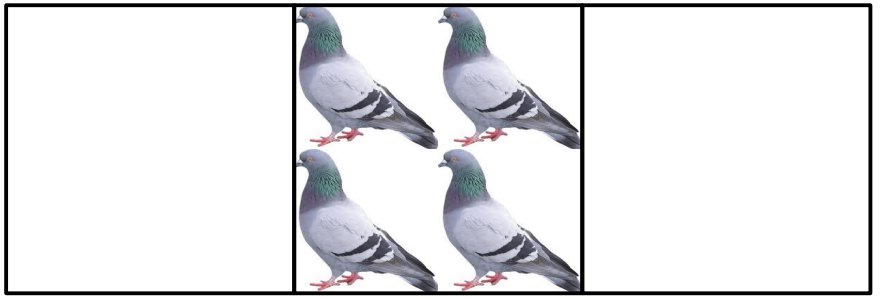
The Pigeonhole Principle

- Suppose that you have n pigeonholes.
- Suppose that you have m > n pigeons.
- If you put the pigeons into the pigeonholes, some pigeonhole will have more than one pigeon in it.



The Pigeonhole Principle

- Suppose that you have n pigeonholes.
- Suppose that you have m > n pigeons.
- If you put the pigeons into the pigeonholes, some pigeonhole will have more than one pigeon in it.



Proof of the Pigeonhole Principle

Theorem: If m > n objects are put into n bins, there must be some bin with at least two objects in it.

Proof of the Pigeonhole Principle

Theorem: If m > n objects are put into n bins, there must be some bin with at least two objects in it.

Proof: By contradiction;

Proof of the Pigeonhole Principle

Theorem: If m > n objects are put into n bins, there must be some bin with at least two objects in it.

Proof: By contradiction; assume all n bins have at most one element in them.

Proof of the Pigeonhole Principle

Theorem: If m > n objects are put into n bins, there must be some bin with at least two objects in it.

Proof: By contradiction; assume all n bins have at most one element in them. But then there are are at most n elements distributed across them, contradicting the fact that m > n.

Proof of the Pigeonhole Principle

Theorem: If m > n objects are put into n bins, there must be some bin with at least two objects in it.

Proof: By contradiction; assume all n bins have at most one element in them. But then there are are at most n elements distributed across them, contradicting the fact that m > n. We have reached a contradiction, so our assumption was wrong and some bin has at least two objects in it.

Proof of the Pigeonhole Principle

Theorem: If m > n objects are put into n bins, there must be some bin with at least two objects in it.

Proof: By contradiction; assume all n bins have at most one element in them. But then there are are at most n elements distributed across them, contradicting the fact that m > n. We have reached a contradiction, so our assumption was wrong and some bin has at least two objects in it. ■

Some Simple Applications

- Any group of 367 people must have a pair of people that share a birthday.
  - 366 possible birthdays (pigeonholes)
  - 367 people (pigeons)
- Two people in San Francisco have the exact same number of hairs on their head.
  - Maximum number of hairs ever found on a human head is no greater than 500,000.
  - There are over 800,000 people in San Francisco.

Nonconstructive Proofs

- The pigeonhole principle often is used in **nonconstructive proofs**.
- A proof is **constructive** if it shows how to find some particular object.
- Examples:
  - Proving any tournament graph has a tournament winner.
  - Showing that a $2^n \times 2^n$ board with a square missing can be tiled with right triominoes.
- A proof is **nonconstructive** if it guarantees that some object must exist, but does not say how to find it.
  - What two SF residents have the same number of hairs on their heads?

Proofs with Pigeonholes

- Proofs using the pigeonhole principle often work as follows:
  - Construct some set of n bins into which at least n + 1 elements must be distributed. **(This is the creative step of the proof)**
  - Claim, by the pigeonhole principle, that some bin must contain at least two distinct objects.
  - Argue that these two objects must have something in common, which can be used to assert that some object must exist.

A More Complex Application

Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance $d > 0$, there are two points exactly distance d from one another that are the same color.

A More Complex Application

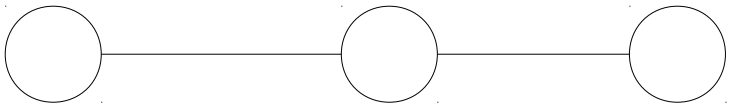
Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance d > 0, there are two points exactly distance d from one another that are the same color.

Thought: There are two colors here, so if we start picking points, we'll be dropping them into one of two buckets (red or blue).

How many points do we need to pick to guarantee that we get two of the same color?

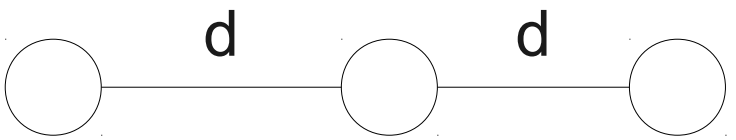
A More Complex Application

Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance d > 0, there are two points exactly distance d from one another that are the same color.



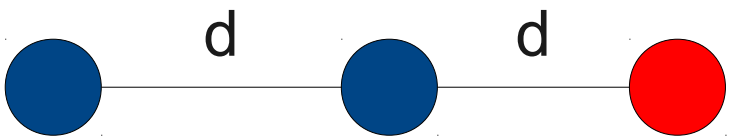
A More Complex Application

Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance d > 0, there are two points exactly distance d from one another that are the same color.



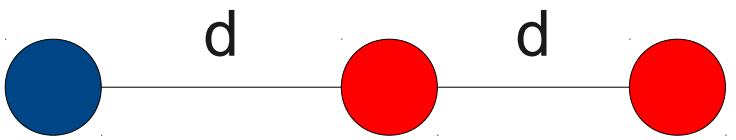
A More Complex Application

Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance d > 0, there are two points exactly distance d from one another that are the same color.



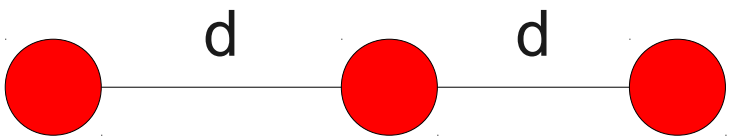
A More Complex Application

Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance d > 0, there are two points exactly distance d from one another that are the same color.



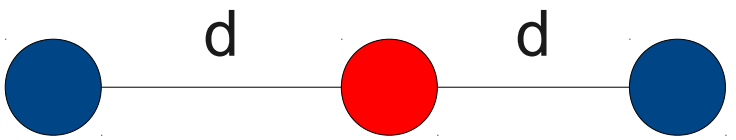
A More Complex Application

Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance d > 0, there are two points exactly distance d from one another that are the same color.



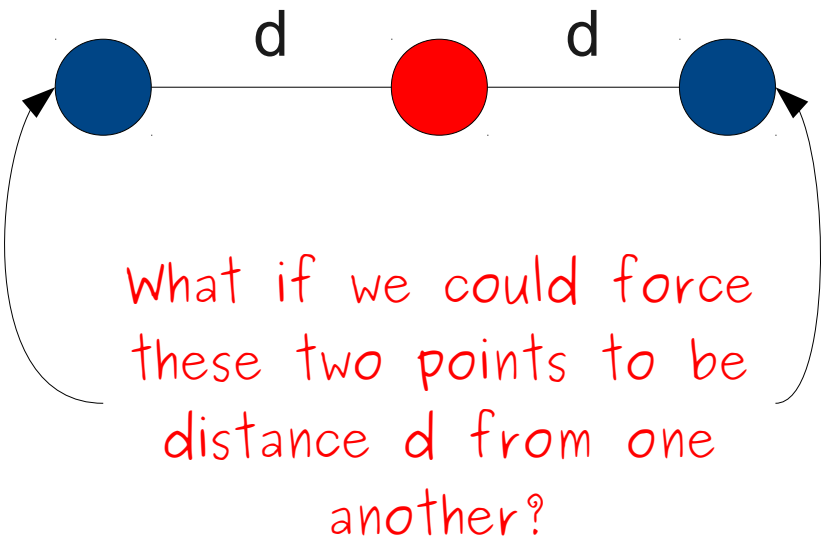
A More Complex Application

Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance d > 0, there are two points exactly distance d from one another that are the same color.



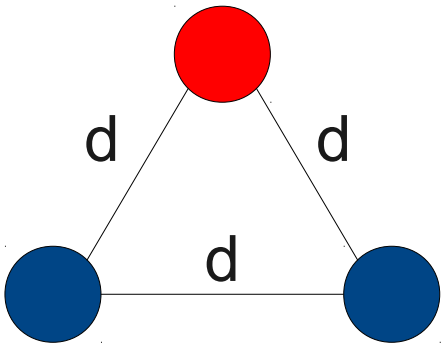
A More Complex Application

Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance d > 0, there are two points exactly distance d from one another that are the same color.



A More Complex Application

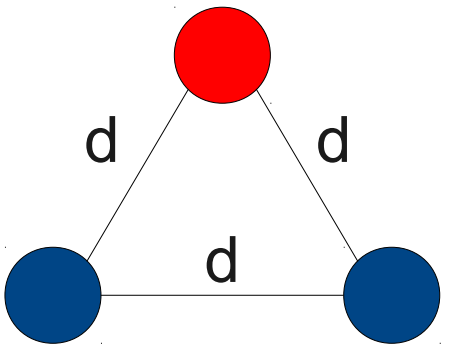
Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance d > 0, there are two points exactly distance d from one another that are the same color.



A More Complex Application

Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance d > 0, there are two points exactly distance d from one another that are the same color.

Any pair of points is at distance d from one another. Since two must be the same color, there is a pair of points of the same color at distance d!



A More Complex Application

Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance $d > 0$, there are two points exactly distance d from one another that are the same color.

Proof: Consider any equilateral triangle whose side lengths are d.

A More Complex Application

Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance $d > 0$, there are two points exactly distance d from one another that are the same color.

Proof: Consider any equilateral triangle whose side lengths are d. Put this triangle anywhere in the plane.

A More Complex Application

Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance $d > 0$, there are two points exactly distance d from one another that are the same color.

Proof: Consider any equilateral triangle whose side lengths are d. Put this triangle anywhere in the plane. By the pigeonhole principle, because there are three vertices, two of the vertices must have the same color.

A More Complex Application

Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance $d > 0$, there are two points exactly distance d from one another that are the same color.

Proof: Consider any equilateral triangle whose side lengths are d. Put this triangle anywhere in the plane. By the pigeonhole principle, because there are three vertices, two of the vertices must have the same color. These vertices are at distance d from each other, as required.

A More Complex Application

Theorem: Suppose that every point in the real plane is colored either red or blue. Then for any distance $d > 0$, there are two points exactly distance d from one another that are the same color.

Proof: Consider any equilateral triangle whose side lengths are d. Put this triangle anywhere in the plane. By the pigeonhole principle, because there are three vertices, two of the vertices must have the same color. These vertices are at distance d from each other, as required. ■

The Limits of Data Compression

The Pigeonhole Principle and Functions

- Consider a function $f : A \to B$ for finite sets A and B.
- If $|A| > |B|$, then by the pigeonhole principle some element of B has at least two elements of A that map to it.
- Thus f cannot be injective.

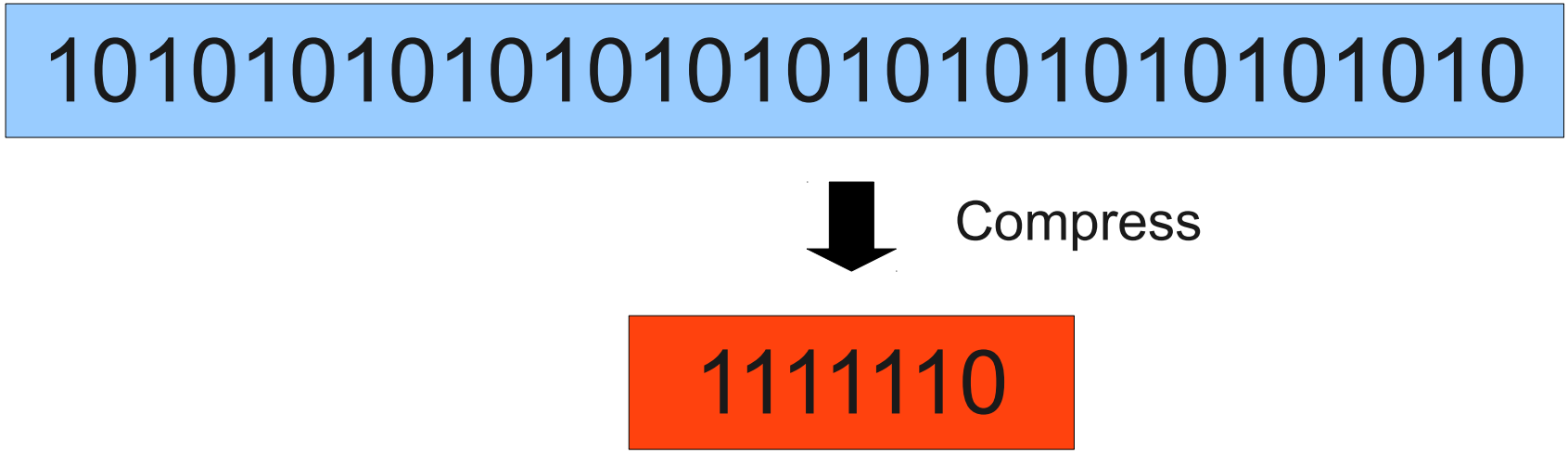
Bitstrings

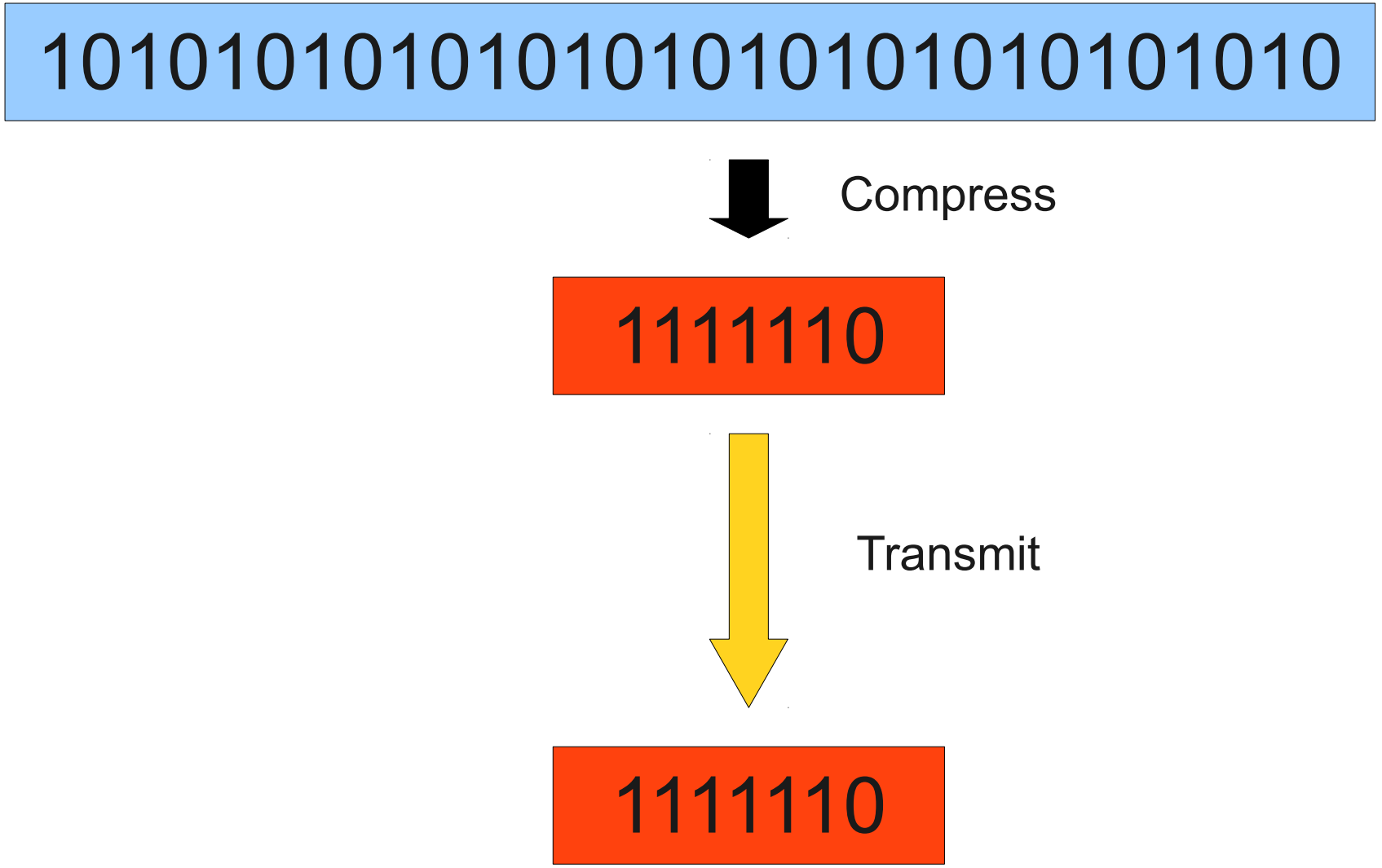
- A **bitstring** is a finite sequence of 0s and 1s.
- Examples:
  - 11011100
  - 010101010101
  - 0000
  - ε (the **empty string**)
- There are 2^n bitstrings of length exactly n.
- There are $2^0 + 2^1 + \ldots + 2^n = 2^{n+1} - 1$ bitstrings of length at most n.
  - We proved this by induction earlier in the quarter.

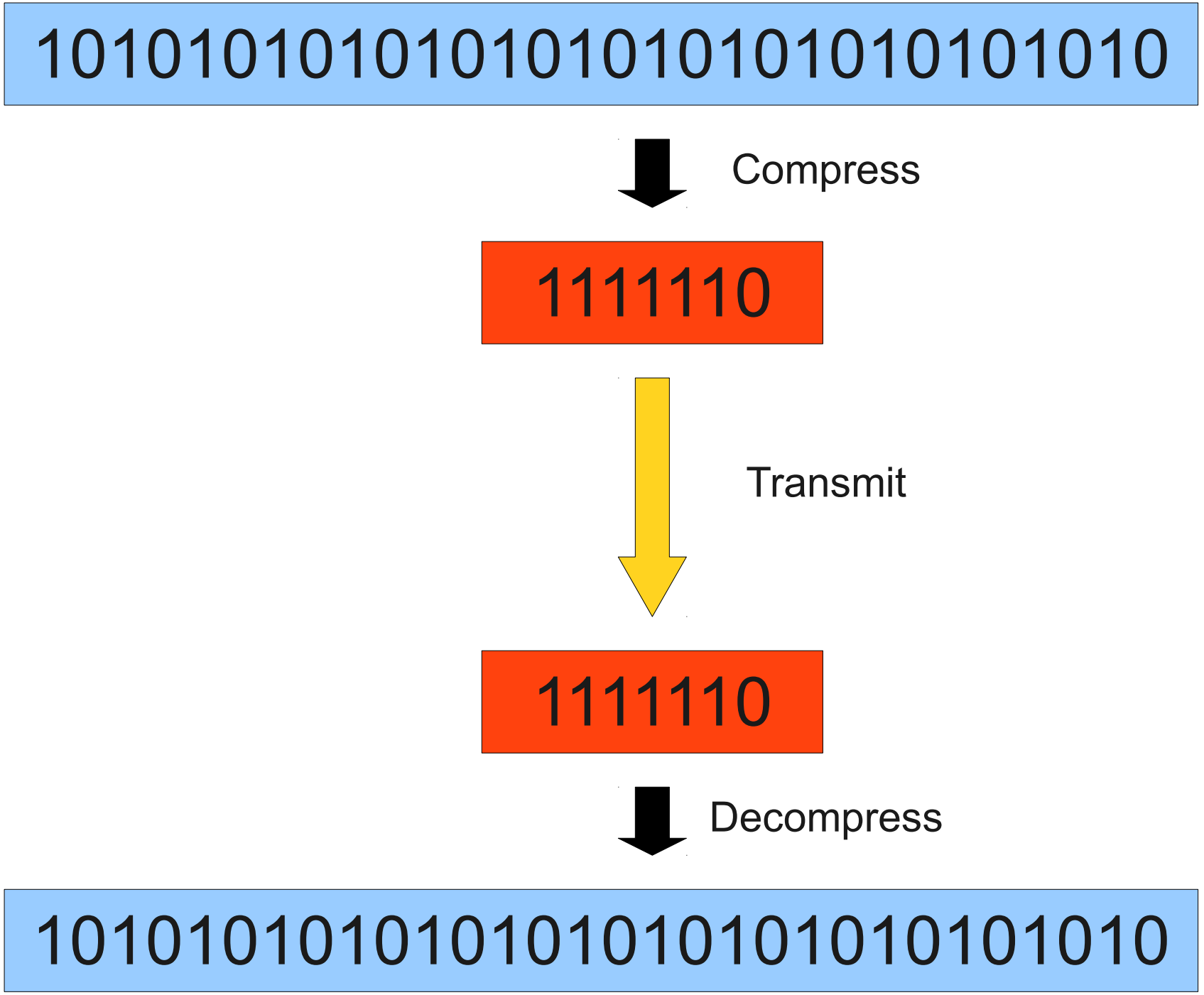
Data Compression

- Inside a computer, all data are represented as sequences of 0s and 1s (bitstrings)
- To transfer data (across a network, on DVDs, on a flash drive, etc.), it is advantageous to try to reduce the number of 0s and 1s before transferring it.
- Most real-world data can be compressed by exploiting redundancies.
  - Text repeats common patterns (“the”, “and”, etc.)
  - Bitmap images use similar colors throughout the image.
- **Idea**: Replace each bitstring with a *shorter* bitstring that contains all the original information.
  - This is called **lossless data compression**.

10101010101010101010101010101010


10101010101010101010101010101010
Compress
1111110


10101010101010101010101010101010
Compress
1111110
Transmit
1111110


10101010101010101010101010101010
Compress
1111110
Transmit
1111110
Decompress
10101010101010101010101010101010

Lossless Data Compression

- In order to losslessly compress data, we need two functions:
  - A **compression function** C, and
  - A **decompression function** D.
- These functions must be inverses of one another: $D = C^{-1}$.
  - Otherwise, we can't uniquely encode or decode some bitstring.
- Since C is invertible, C must be a bijection.

A Perfect Compression Function

- Ideally, the compressed version of a bitstring would always be shorter than the original bitstring.
- **Question**: Can we find a lossless compression algorithm that always compresses a string into a shorter string?
- Let's assume we only care about strings of length at least 10.

A Counting Argument

- Let $\mathbb{B}^n$ be the set of strings of length n, and $\mathbb{B}^{<n}$ be the set of strings of length less than n.
- How many bitstrings of length n are there?
  - **Answer**: 2^n
- How many bitstrings of length *less than n* are there?
  - **Answer:** $2^0 + 2^1 + \ldots + 2^{n-1} = 2^n - 1$
- Using our earlier result, by the pigeonhole principle, there cannot be an injective function from $\mathbb{B}^n$ to $\mathbb{B}^{<n}$.
- Since every bijection is an injection, there is no bijection between $\mathbb{B}^n$ and $\mathbb{B}^{<n}$.
- **There is no perfect compression function!**

Why this Result is Interesting

- Our result says that no matter how hard we try, it is **impossible** to compress every string into a shorter string.
- No matter how clever you are, you cannot write a lossless compression algorithm that always makes strings shorter.
- In practice, only highly redundant data can be compressed.
- The fields of **information theory** and **Kolmogorov complexity** explore the limits of compression; if you're interested, go explore!

Next Time

- Structural Induction
- Trees
- A Sorting Lower Bound