Binary Relations

Outline for Today

- **Binary Relations**
  - Reasoning about connections between objects.
- **Equivalence Relations**
  - Reasoning about clusters.
- **Partial Orders**
  - Reasoning about prerequisites.

Relationships

- In CS103, you've seen examples of relationships
  - between sets:
    - $A \subseteq B$
  - between numbers:
    - $x < y$ $x \equiv_k y$ $x \leq y$
  - between nodes in a graph:
    - u is connected to v.
- Many other relations exist in more specialized contexts, and we'll see a bunch of them over the course of the quarter.

Binary Relations

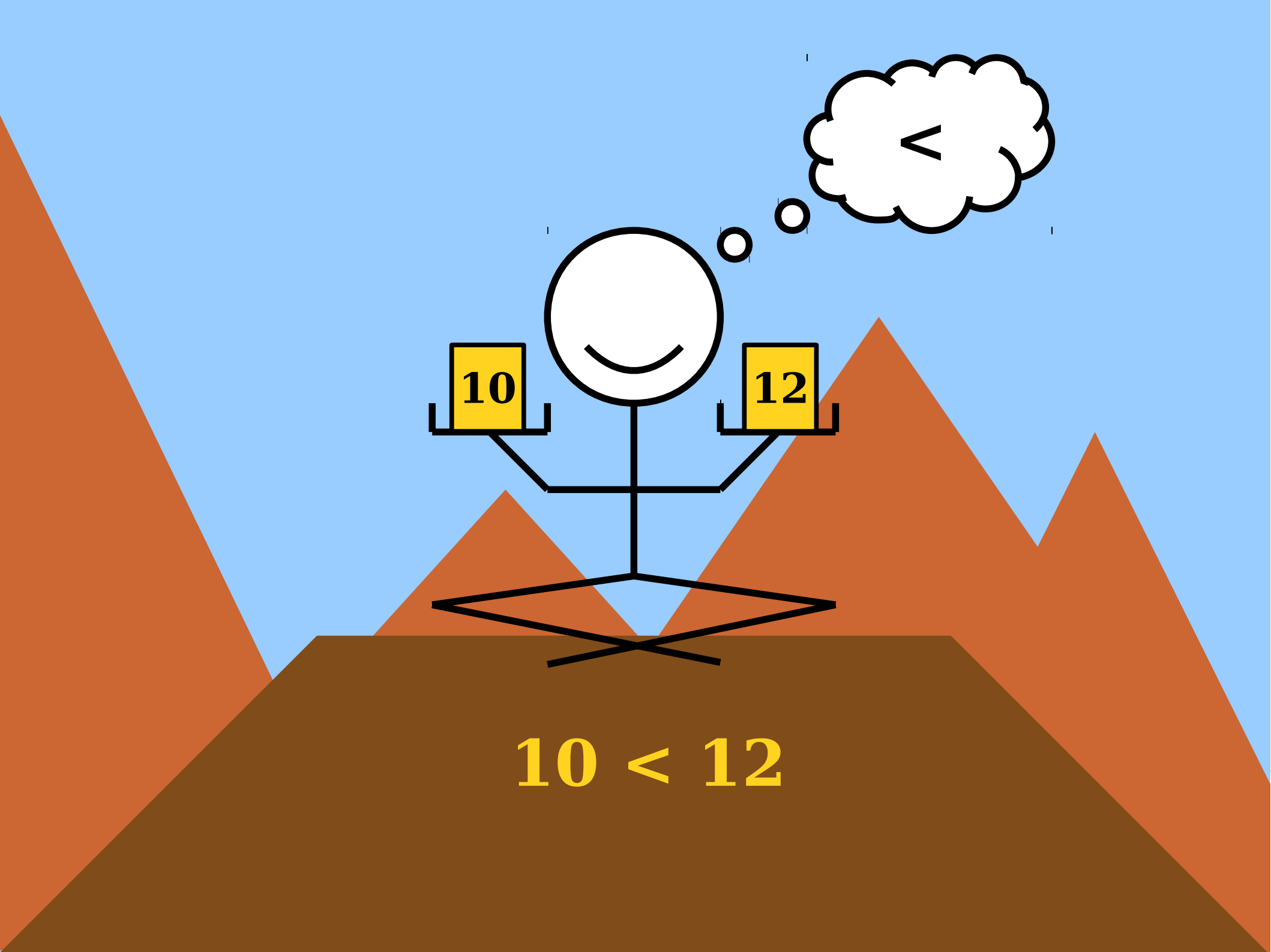
- Consider the set $\mathbb{N}$.
- We say $\leq$ is a ***binary relation over*** $\mathbb{N}$ because, given any $a, b \in \mathbb{N}$, the following questions are meaningful and have definitive yes or no answers:
  - Is $a \leq b$ true?
  - Is $b \leq a$ true?

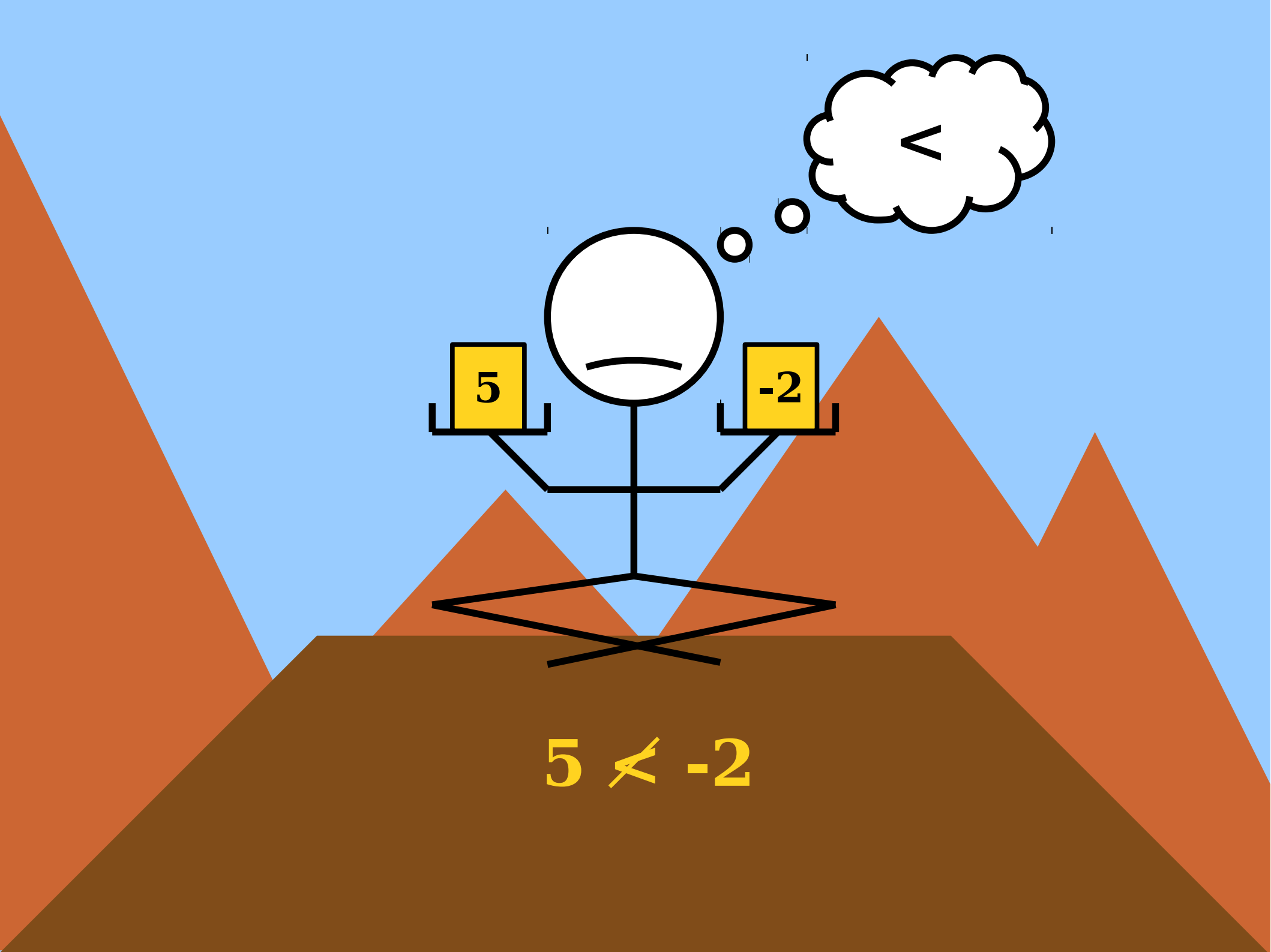
Binary Relations

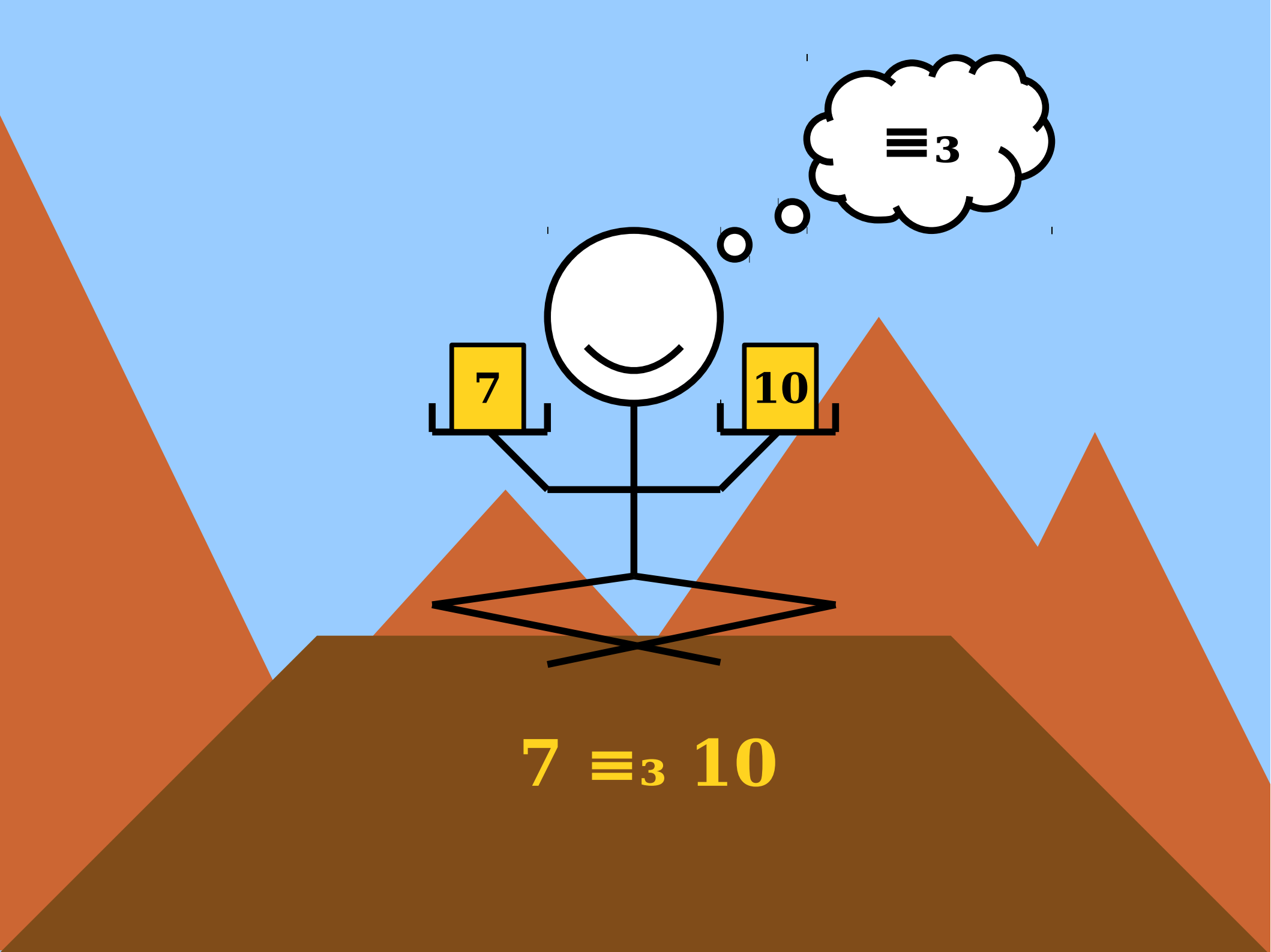
- Consider the set $\mathbb{Z}$.
- We say $\equiv_3$ is a ***binary relation over*** $\mathbb{Z}$ because, given any $a, b \in \mathbb{Z}$, the following questions are meaningful and have definitive yes or no answers:
  - Is $a \equiv_3 b$ true?
  - Is $b \equiv_3 a$ true?

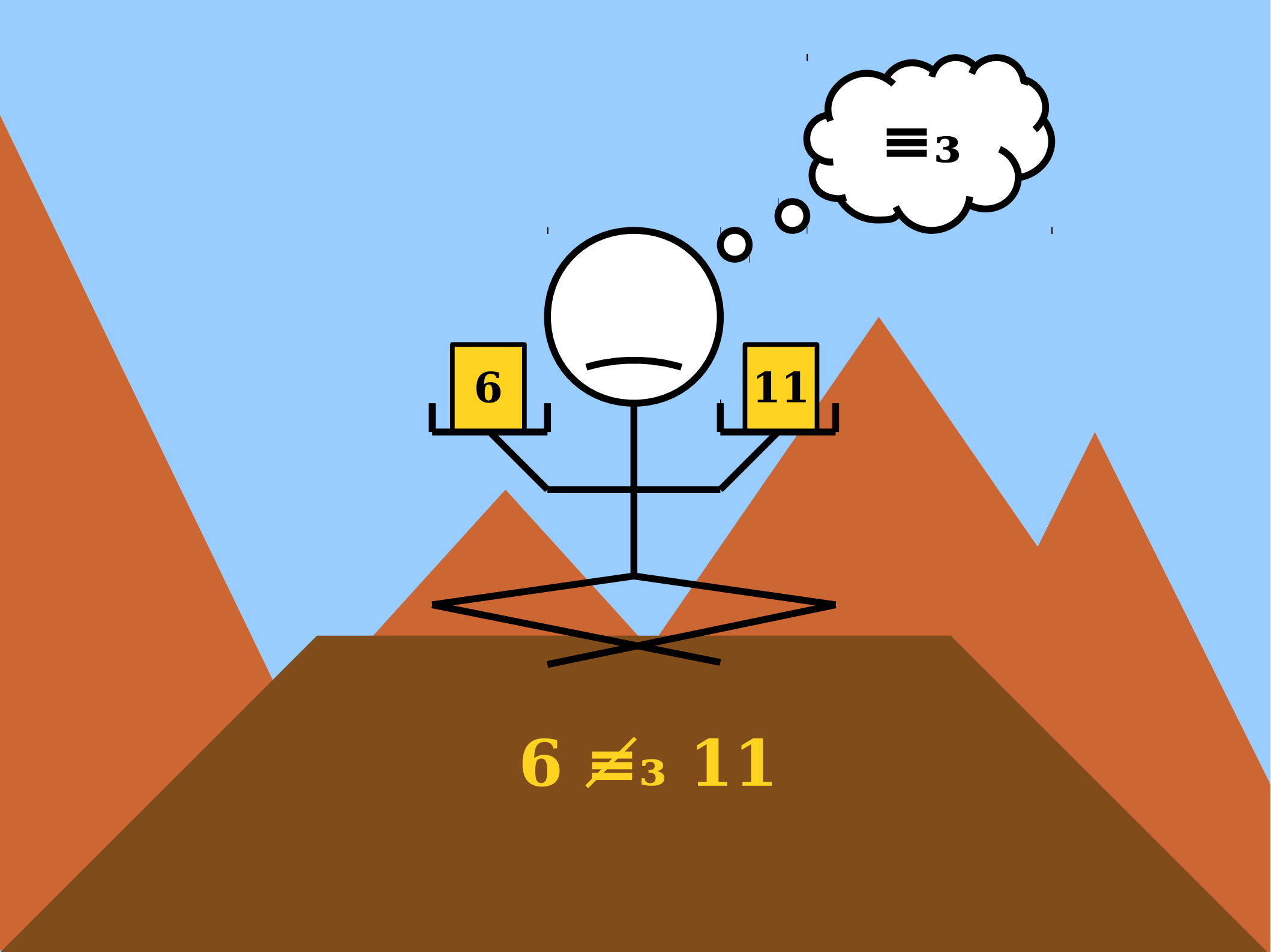
Binary Relations

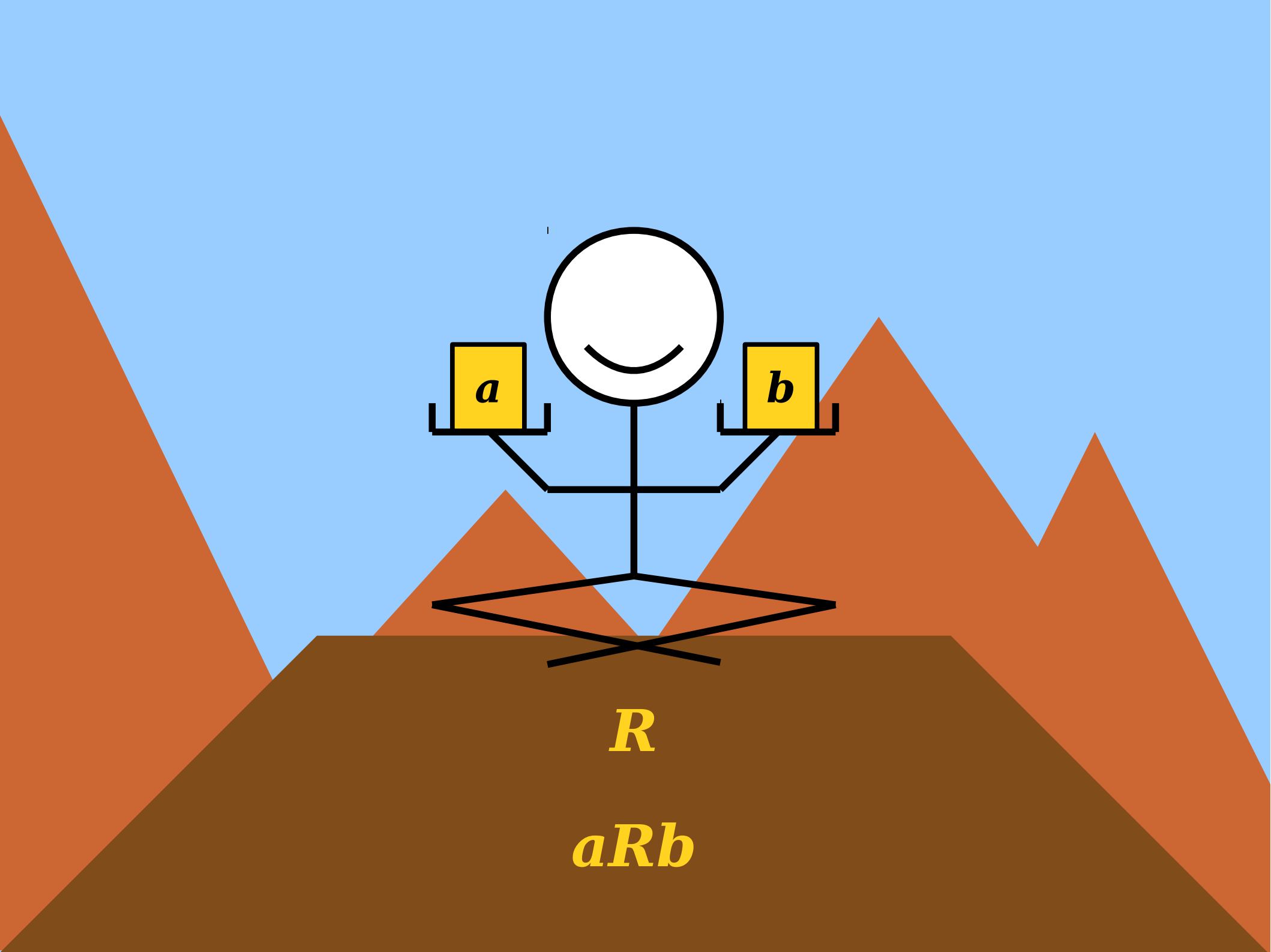
- Consider the set $A = \{a, b, c\}$.
- We say $=$ is a ***binary relation over A*** because, given any $a, b \in A$, the following questions are meaningful and have definitive yes or no answers:
  - Is $a = b$ true?
  - Is $b = a$ true?

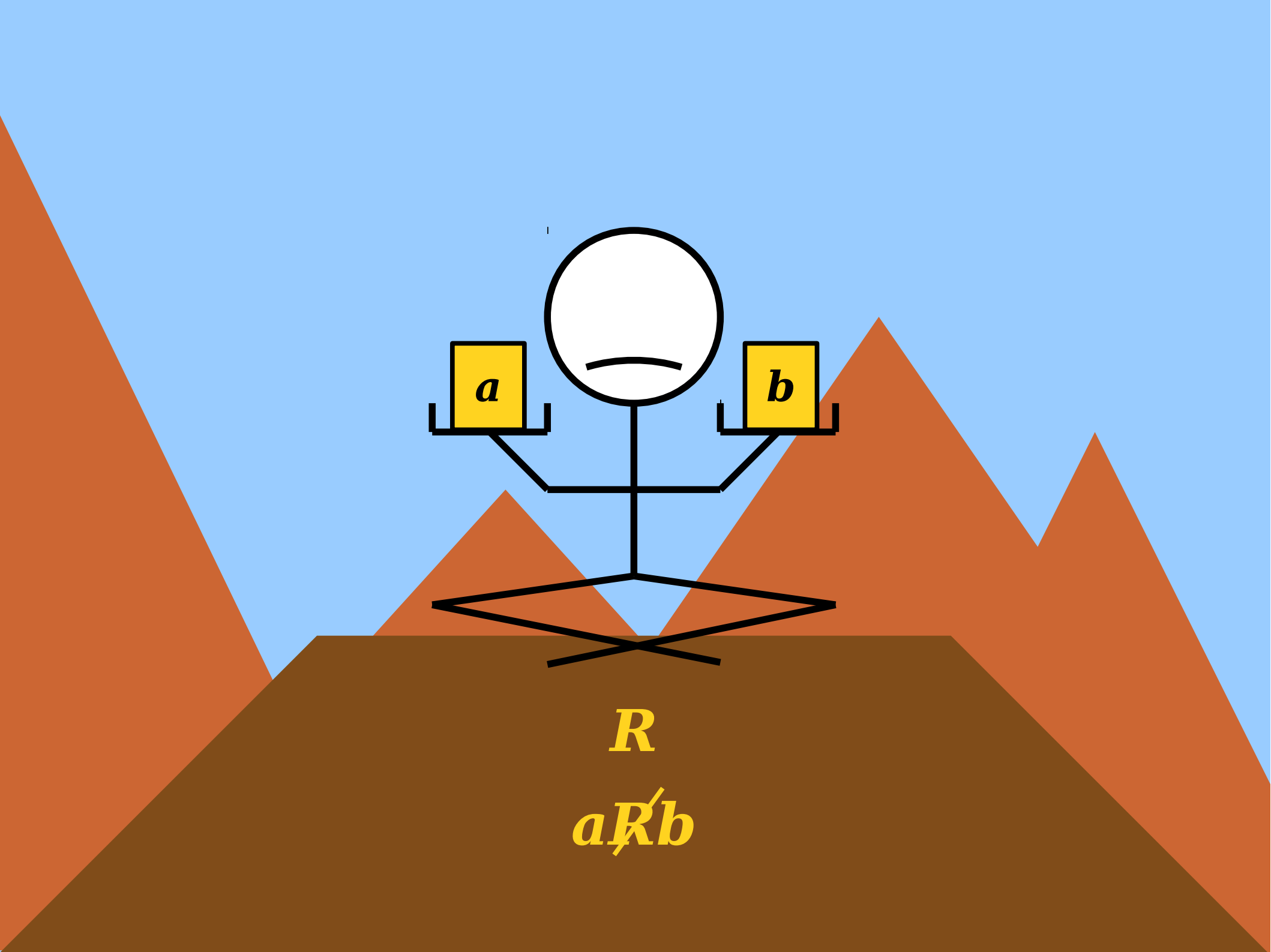

10
12
<
10 < 12


5
-2
<
5 ≮ -2


≡3
7
10
7 ≡3 10


≡3
6
11
6 ≢3 11


a
b
R
aRb

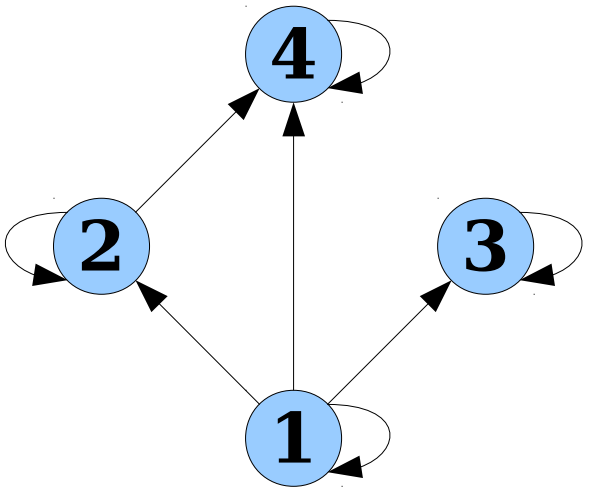

a
b
R
aRb

Binary Relations

- A ***binary relation over a set A*** is a structure that indicates properties about pairs of objects drawn from a set A.
- The particular property indicated depends on the choice of binary relation.
  - Some examples: $<$, $\leq$, $\subseteq$, $\equiv_k$, etc.
- If R is a binary relation over a set A and a, $b \in A$, we write aRb to indicate that the relation given by R holds from a to b.
  - For example, $42 < 137$, $\varnothing \subseteq \mathbb{N}$, etc.
- *Order matters in binary relations*. If we write aRb, we mean that a relation holds between a and b in that order.
  - For example, $42 < 137$, but $137 \not< 42$.
  - It's possible that order doesn't matter ($5 \equiv_3 2$ and $2 \equiv_3 5$, for example), but it depends on the particular relation.
  - The relation isn't required to hold in either direction. For example, the statements $\{a, b\} \subseteq \mathbb{N}$ and $\mathbb{N} \subseteq \{a, b\}$ are both false.

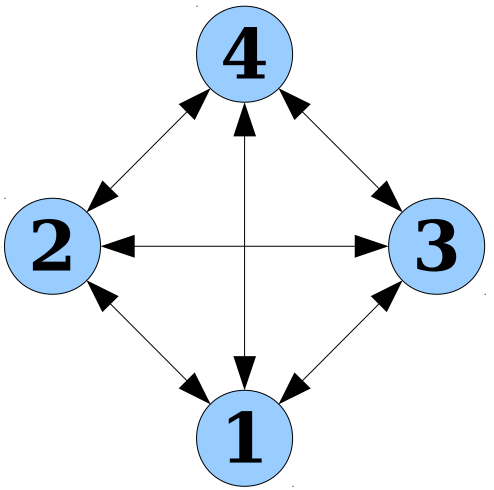
Visualizing Relations

- We can visualize a binary relation R over a set A as a graph:
  - The nodes are the elements of A.
  - There is an edge from x to y if and only if xRy.
- Example: the relation $a \mid b$ (meaning "a divides b") over the set {1, 2, 3, 4} looks like this:



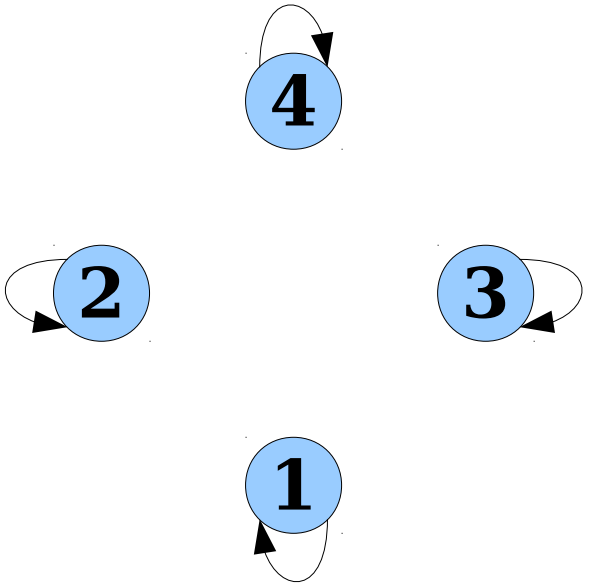
Visualizing Relations

- We can visualize a binary relation R over a set A as a graph:
  - The nodes are the elements of A.
  - There is an edge from x to y if and only if xRy.
- Example: the relation $a \neq b$ over {1, 2, 3, 4} looks like this:



Visualizing Relations

- We can visualize a binary relation R over a set A as a graph:
  - The nodes are the elements of A.
  - There is an edge from x to y if and only if xRy.
- Example: the relation $a = b$ over $\{1, 2, 3, 4\}$ looks like this:

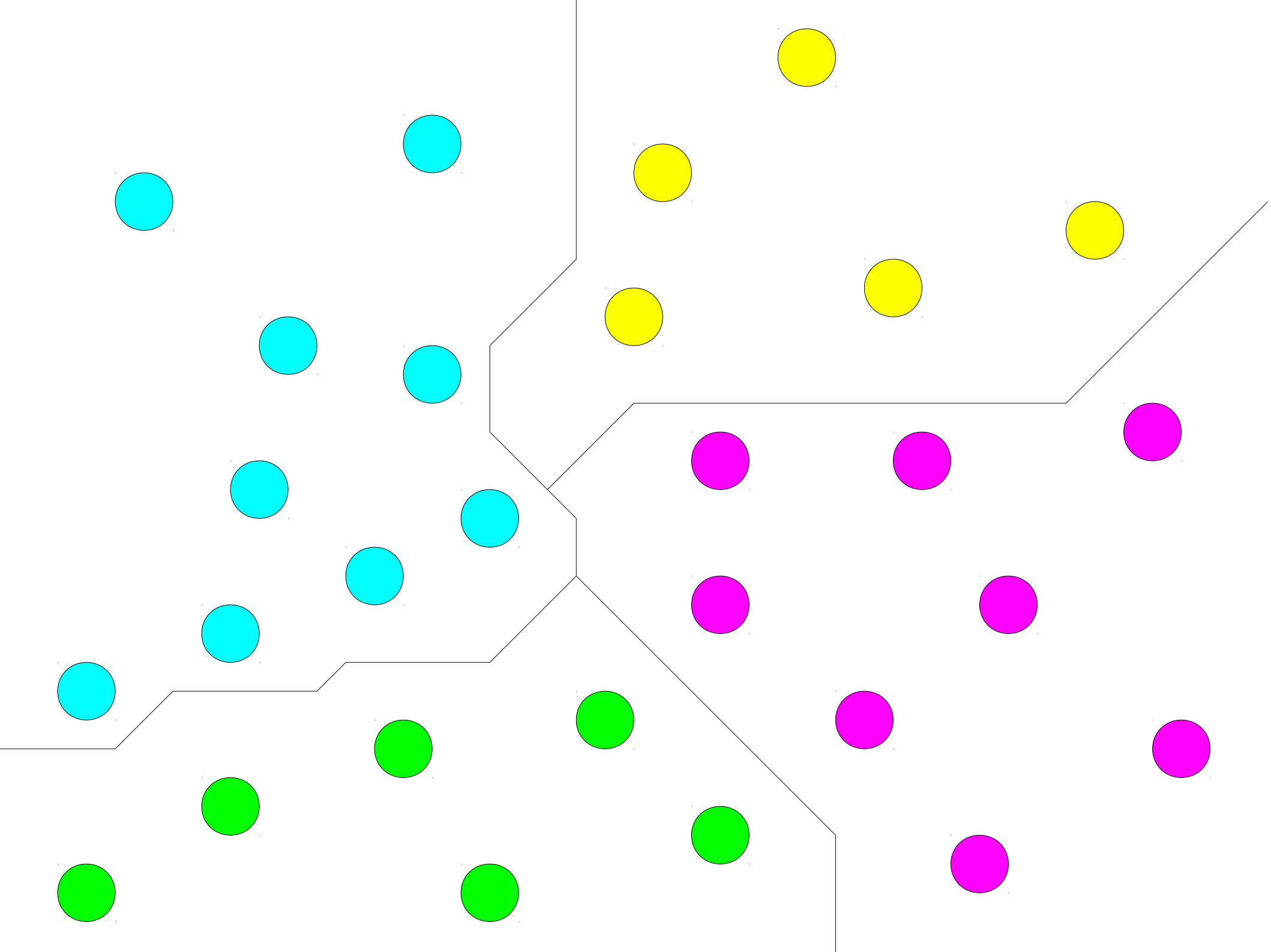


Capturing Structure

Capturing Structure

- Binary relations are an excellent way for capturing certain structures that appear in computer science.
- Today, we'll look at two of them (and possibly one more if we have time.)
  - **Partitions**
  - **Prerequisites**

Partitions



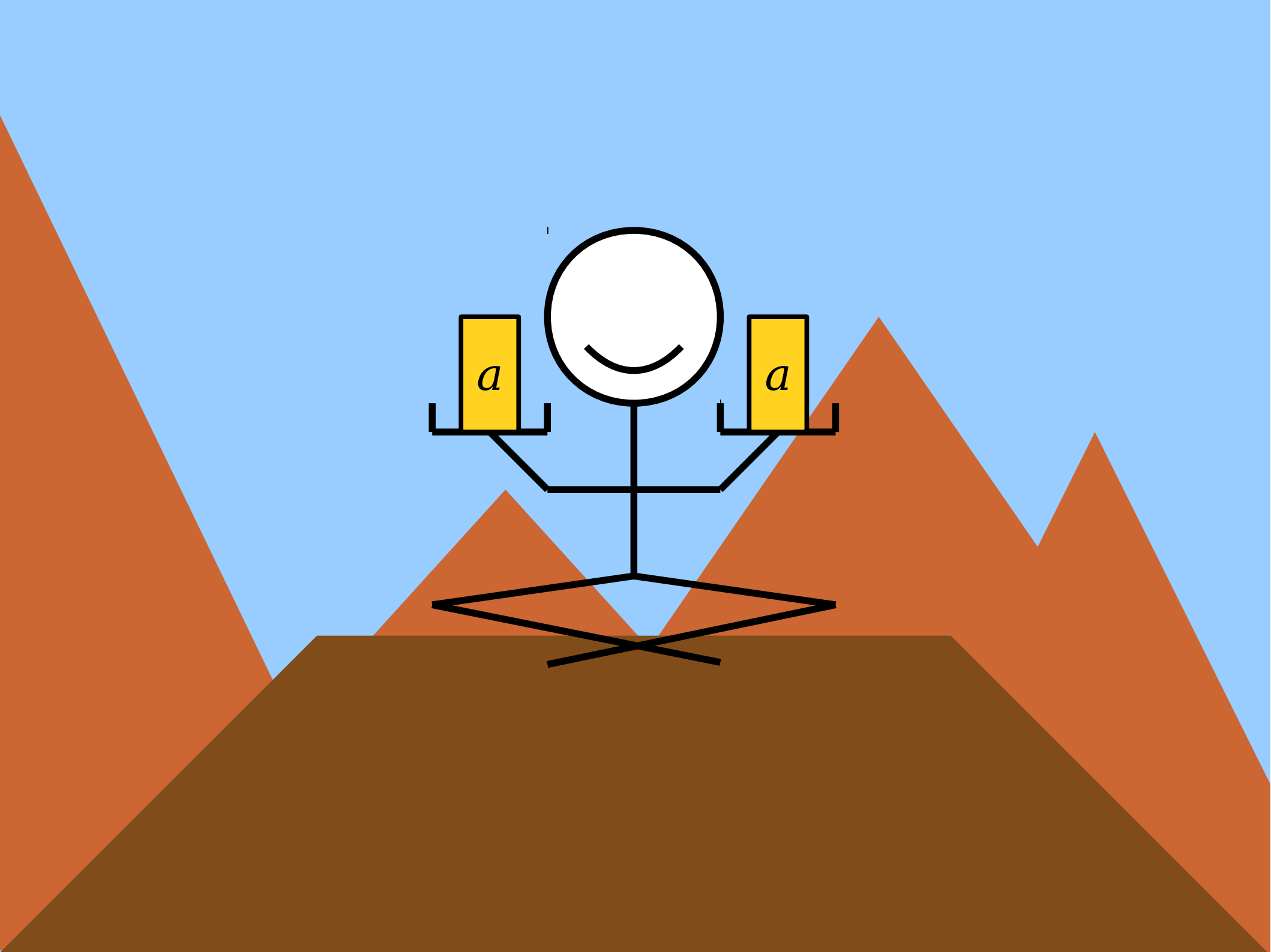
Partitions

- A ***partition of a set*** is a way of splitting the set into disjoint, nonempty subsets so that every element belongs to exactly one subset.
- Intuitively, a partition of a set breaks the set apart into smaller pieces.
- There doesn't have to be any rhyme or reason to what those pieces are, though often there is one.

Partitions and Clustering

- Partitions are useful for data mining.
- If you have a set of data, you can often learn something from the data by finding a “good” clustering of that data and inspecting the clusters.
- Interested to learn more? Take CS161 or CS246!

What's the connection between partitions and binary relations?


a
a

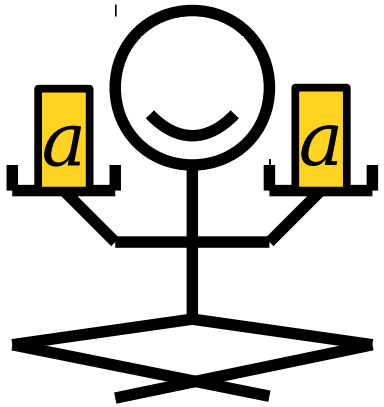

a
b

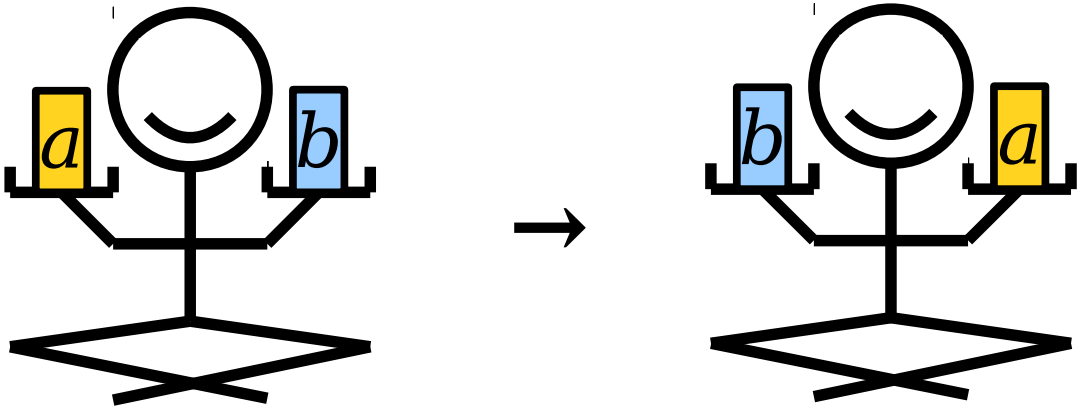

b
a

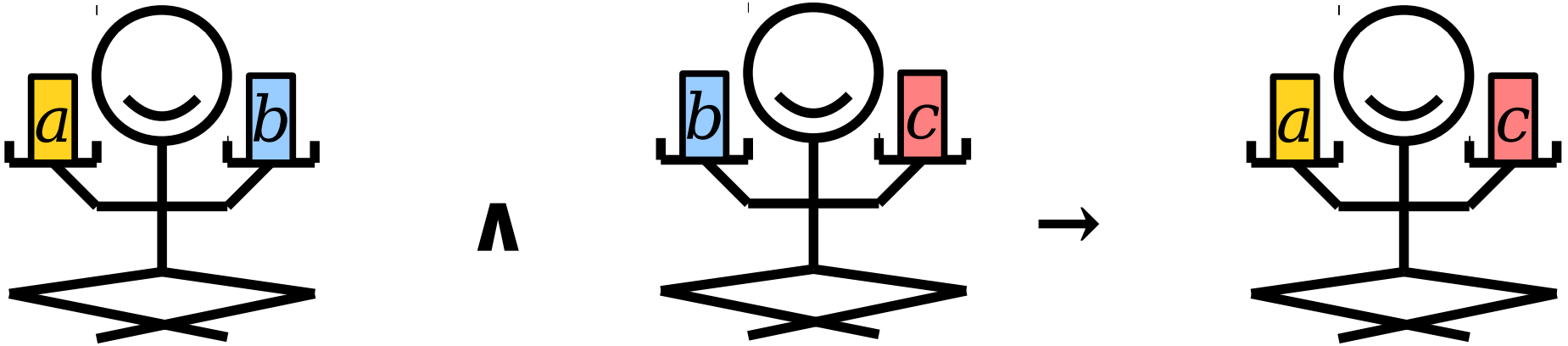

a
b


b
c


a
c


a
a


a
b
b
a


a
b
∧
b
c
a
c

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

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

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

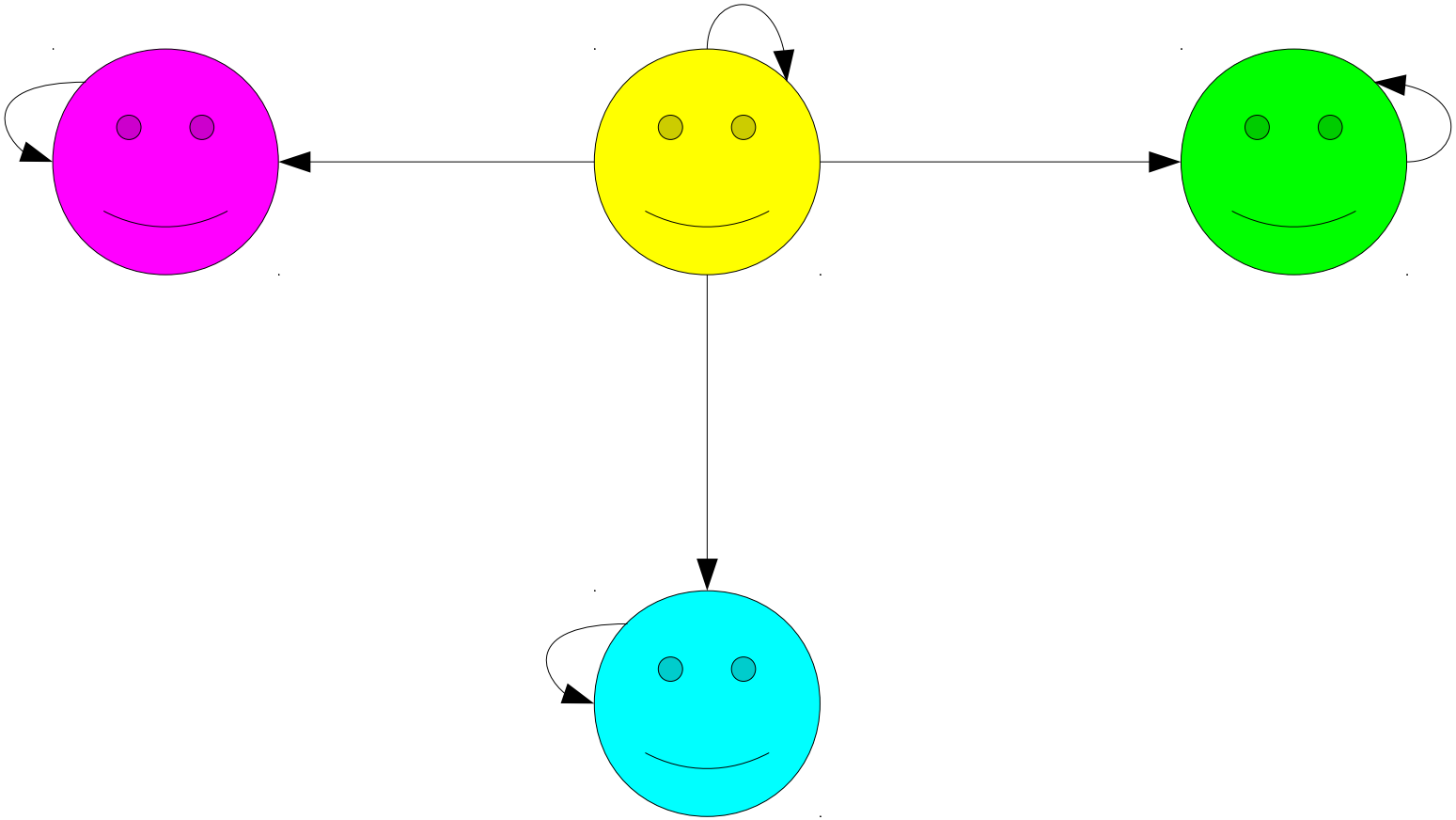
Reflexivity

- Some relations always 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.
  - u is connected to u for any node u.
- 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.”)

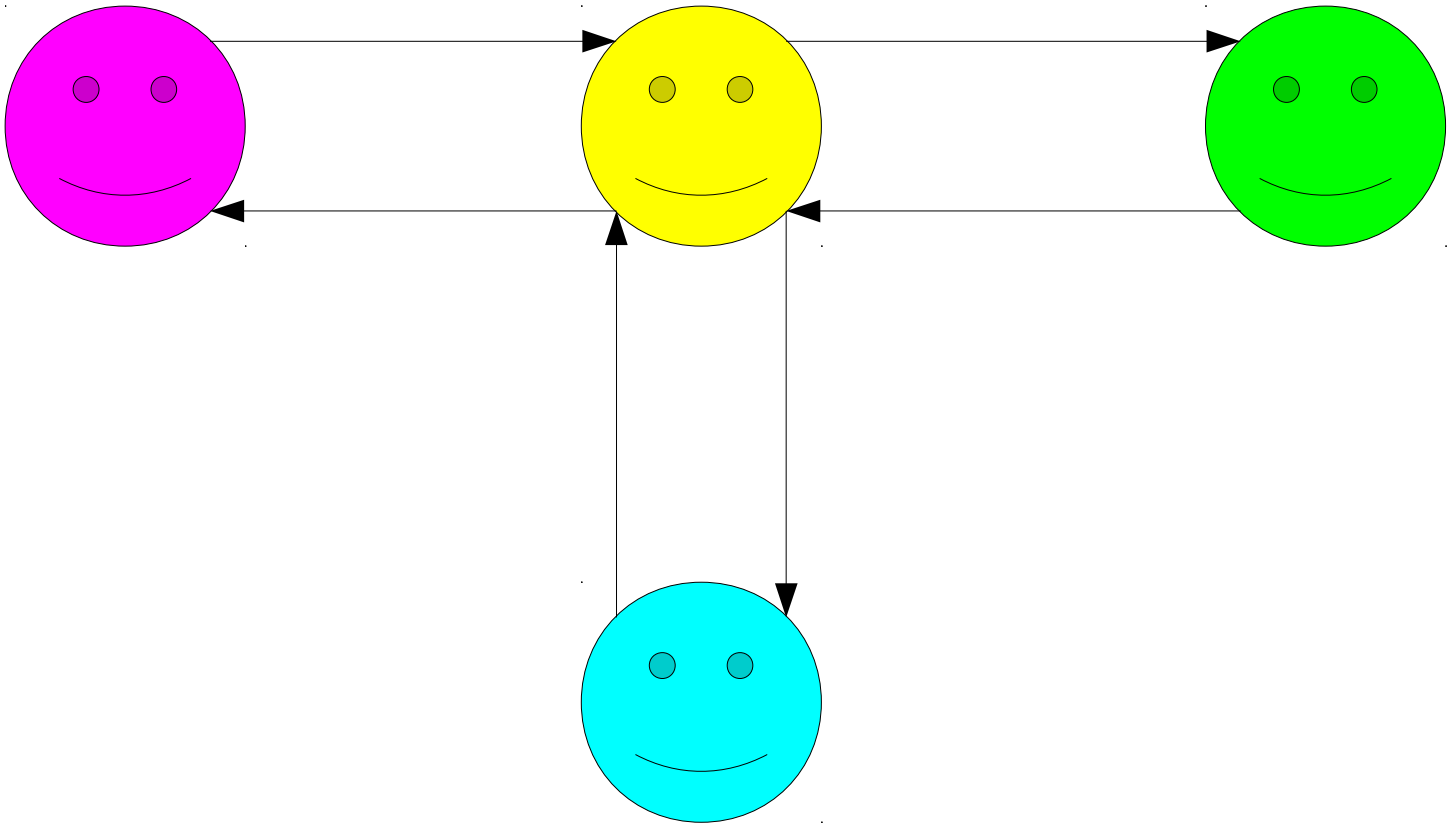
Symmetry

- In some relations, the relative order of the objects doesn't matter.
- Examples:
  - If $x = y$, then $y = x$.
  - If u is connected to v, then v is connected to u.
  - 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

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

("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.")

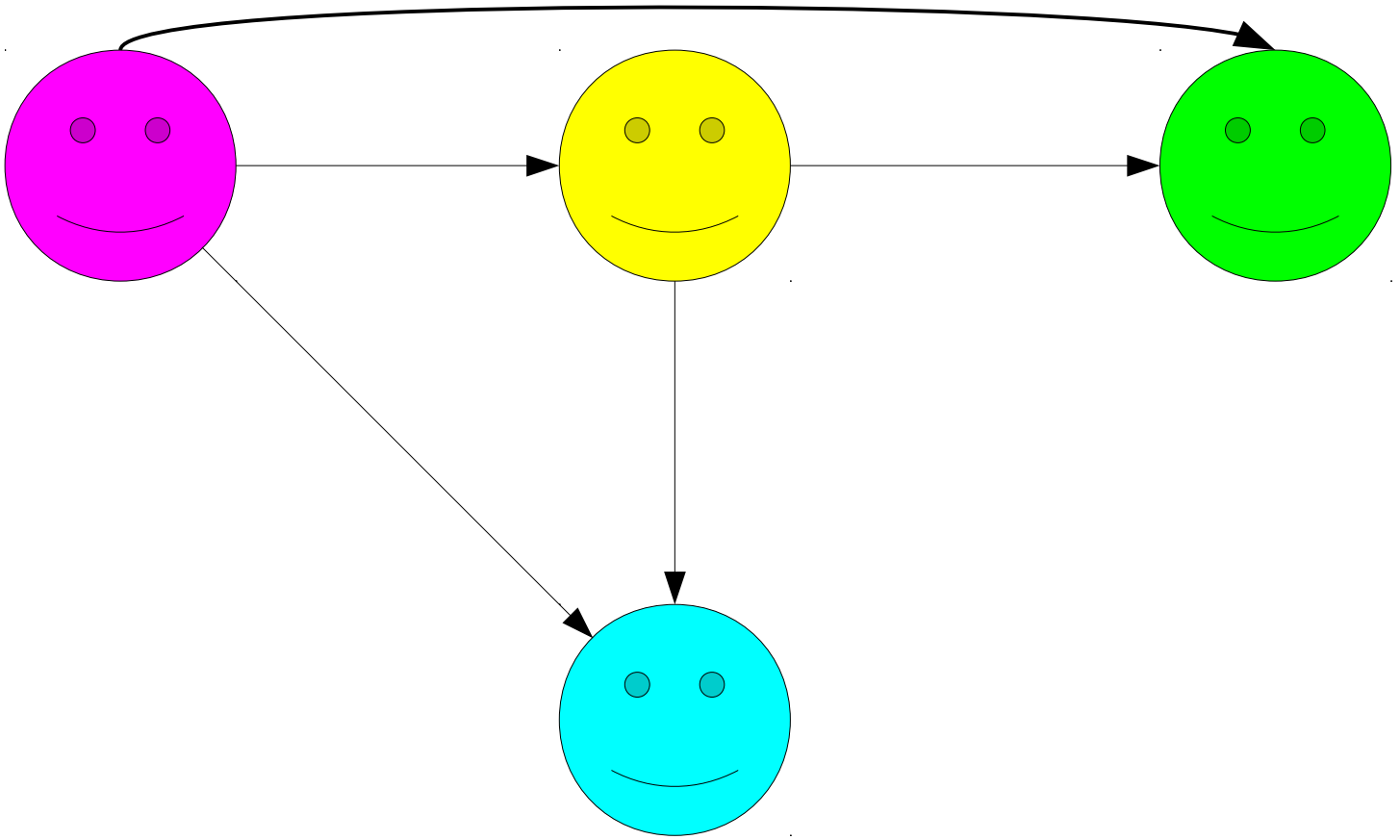
Transitivity

- 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

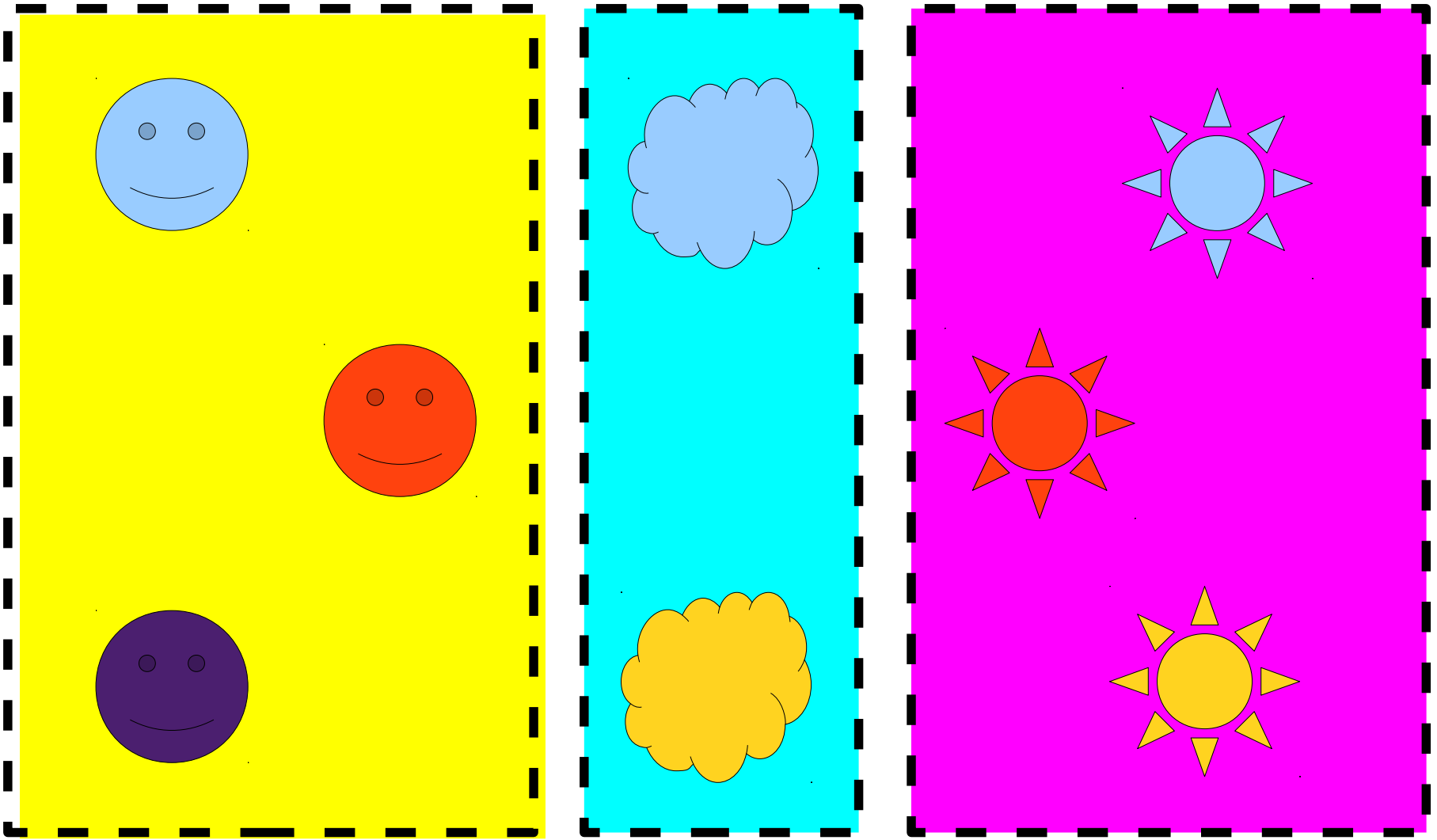


$$\forall a \in A.\ \forall b \in A.\ \forall c \in A.\ (aRb \wedge bRc \rightarrow aRc)$$

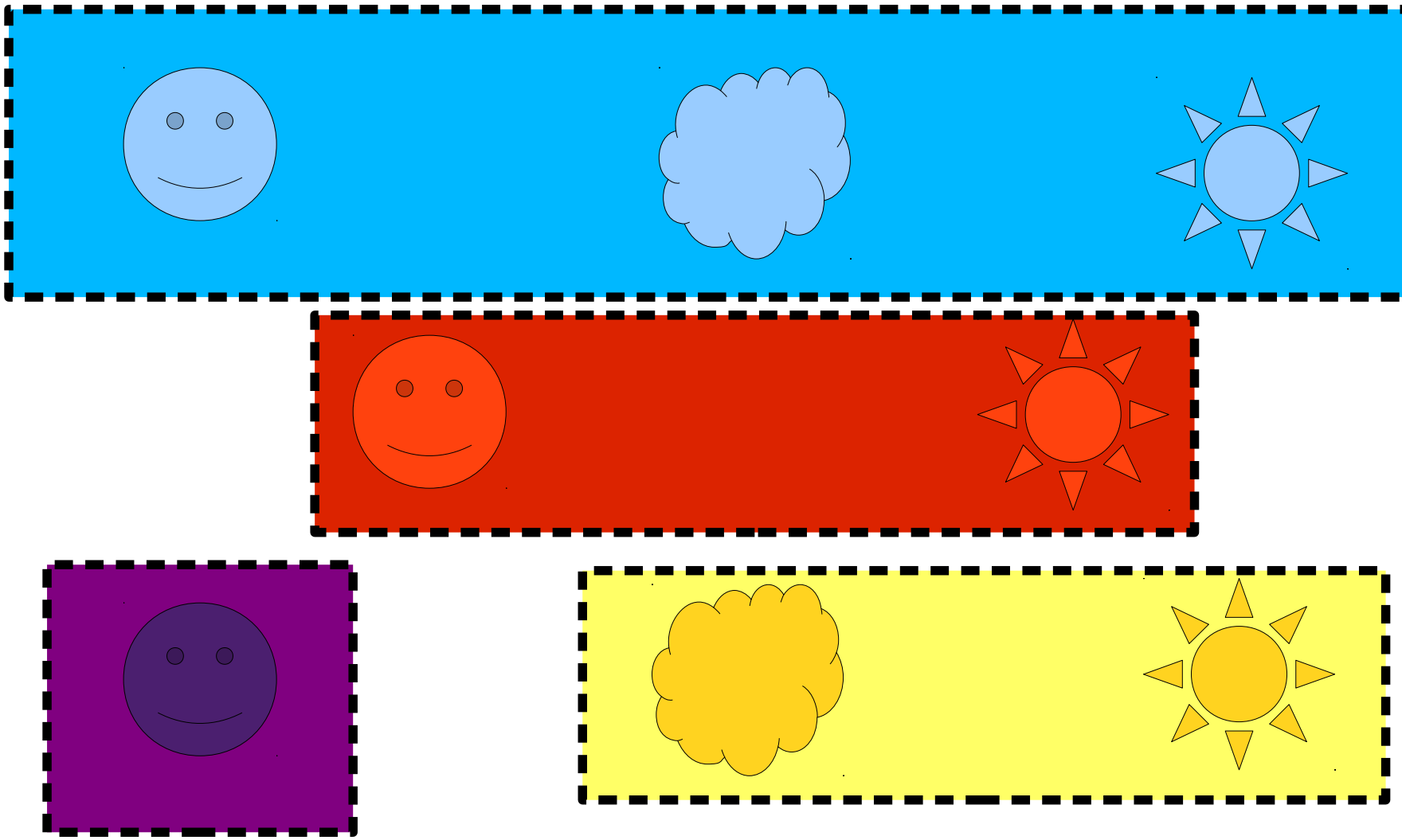
("Whenever a is related to b and b is related to c, we know a is related to c.)

Equivalence Relations

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



Let R be “has the same shape as”



Let T be “is the same **color** as”

Equivalences and Partitions

- Our definition of equivalence relations was motivated by the idea of partitioning elements into groups.

  Partition of Elements ⇒ Equivalence Relation

- In the previous examples, we've seen several equivalence relations that then give rise to a partition. It turns out that this is not a coincidence!

  Equivalence Relation ⇒ Partition of Elements

Equivalence Classes

- Given an equivalence relation R over a set A, for any $x \in A$, the ***equivalence class of x*** is the set

$$[x]_R = \{ y \in A \mid xRy \}$$

- $[x]_R$ is the set of all elements of A that are related to x by relation R.
- ***Theorem:*** If R is an equivalence relation over A, then every $a \in A$ belongs to exactly one equivalence class.

Closing the Loop

- In any graph $G = (V, E)$, we saw that the connected component containing a node $v \in V$ is given by

$$\{ x \in V \mid v \text{ is connected to } x \}$$

- What is the equivalence class for some node $v \in V$ under the relation $\leftrightarrow$?

$$[v]_{\leftrightarrow} = \{ x \in V \mid v \text{ is connected to } x \}$$

- *Connected components are just equivalence classes of* $\leftrightarrow$!

Time-Out for Announcements!

Problem Set Four

- Problem Set Three due today at 5:00PM (sorry about the Scoryst issues – we're working on it!)
- Problem Set Four goes out today.
  - Checkpoint due on Monday at the start of class.
  - Remaining problems due on Monday, May 4 at the start of class.
  - Explore functions, cardinality, binary relations, and the pigeonhole principle!
  - See applications to foundational mathematics, program optimization, and parallel computation.

Extra Practice Problems

- There's a new set of extra practice problems up on the course website. Feel free to check it out!
- Solutions to the extra practice problems from Wednesday are available in hardcopy at lecture or in the solution set drawer.
- We'll take a poll about what topics you'd like more practice on over the weekend.

Practice Midterm Exam

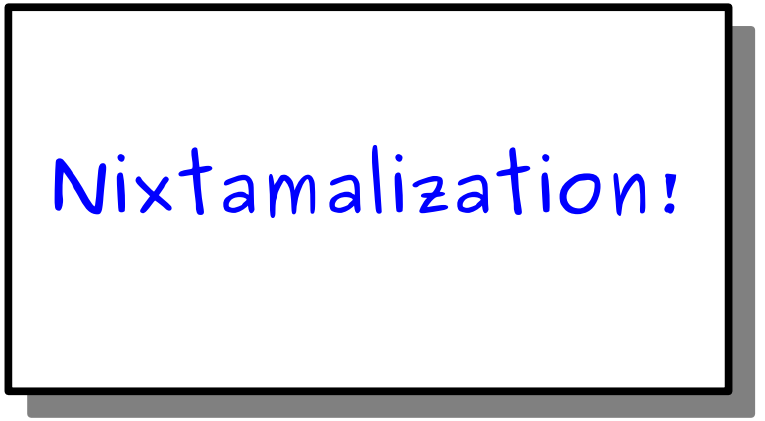
- We'll be holding a practice exam this Monday from 7PM – 10PM in **Annenberg Auditorium**.
- *Highly recommended!* This is a great way to get practice and assess what you need to focus your studying on.

Friday Four Square

- Friday Four Square is happening today at 4:15PM outside the Gates building.
- Fun way to meet a lot of people, both in and out of the CS department.
- Everyone is welcome!

Your Questions

“Speak on a topic of your choice which you find fascinating (inside or outside the realm of Computer Science). Teach us something new!”

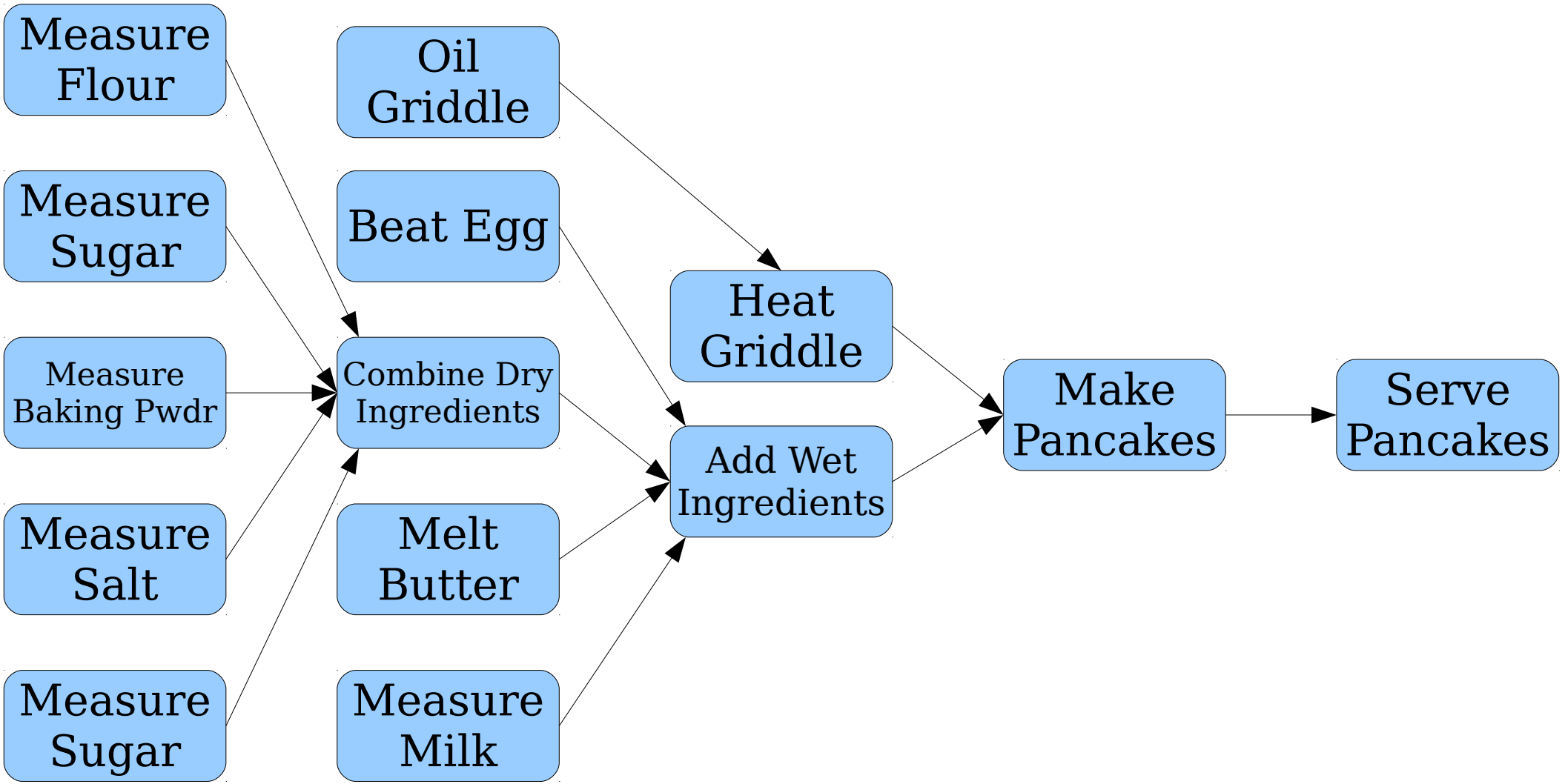


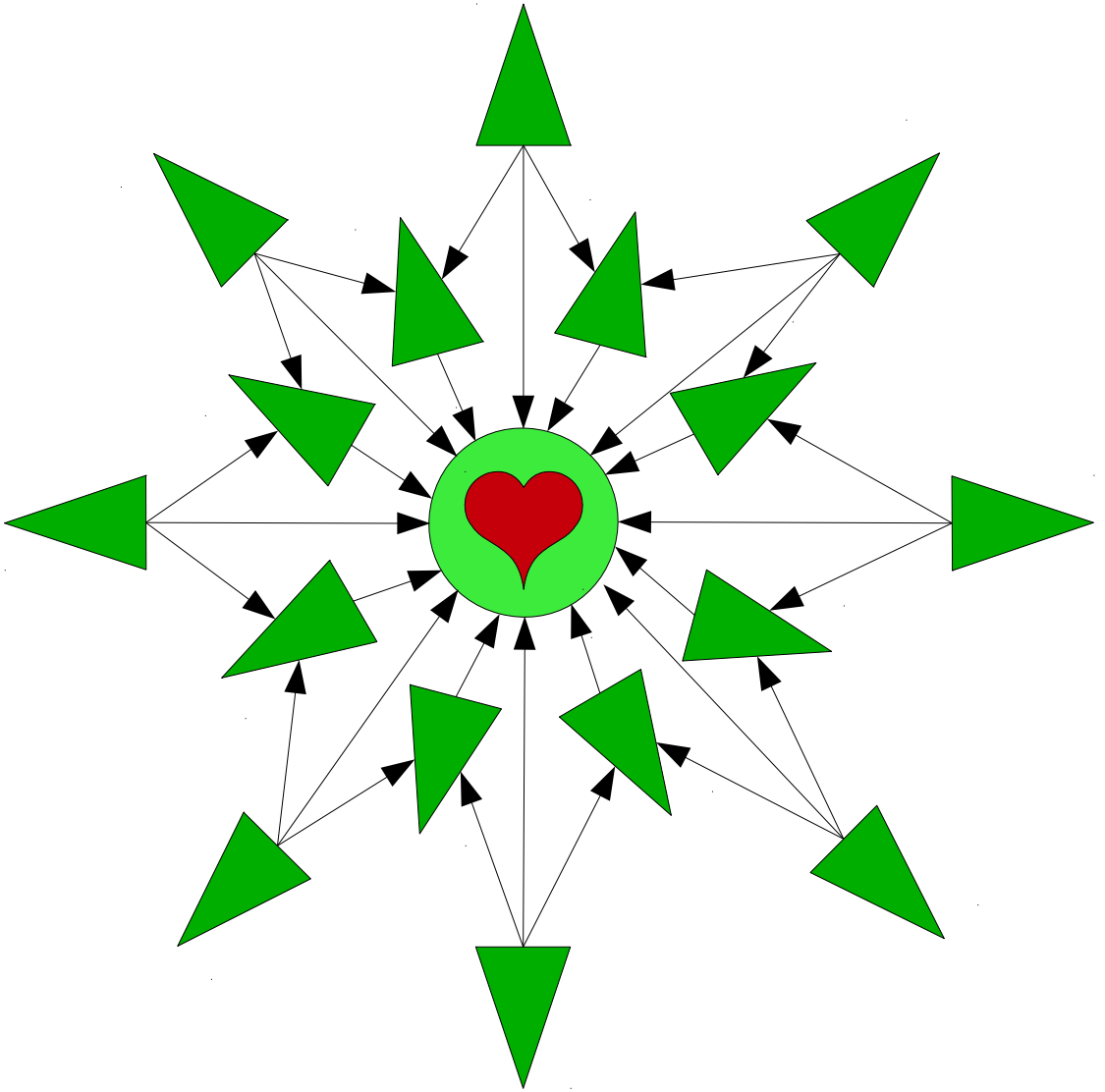
“Which is more important to you? Leading a happy and easy life or leading a meaningful and possibly challenging one? Of course the best situation is having both happiness and meaning, but if you could only pick one which will you pick?”

I'm not sure I'm qualified to give an expert opinion on this, but I can tell you a story!

Back to CS103!

Prerequisite Structures


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

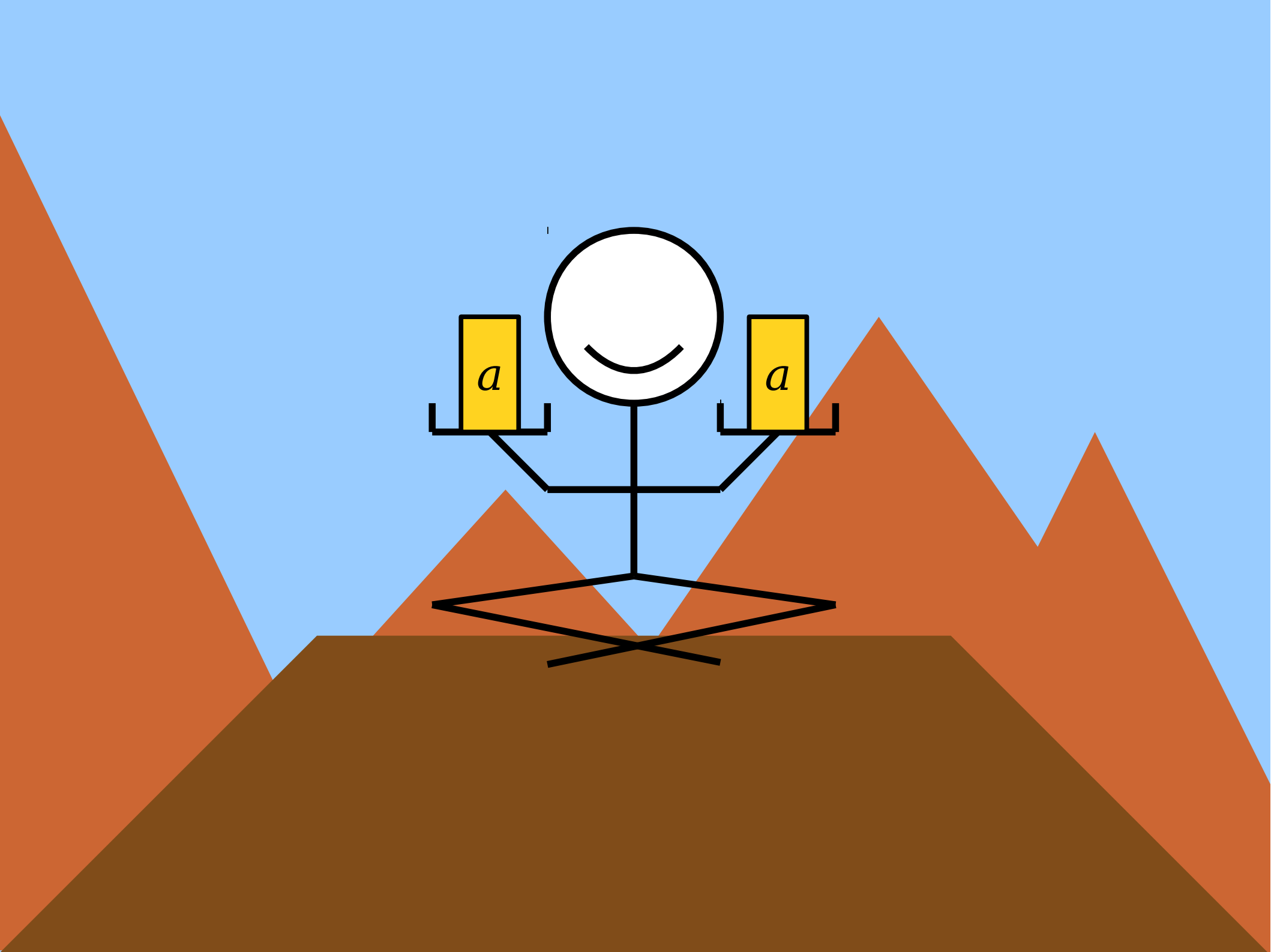


Relations and Prerequisites

- 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* can't be performed after *b***"

- What properties of R could we deduce just from this?

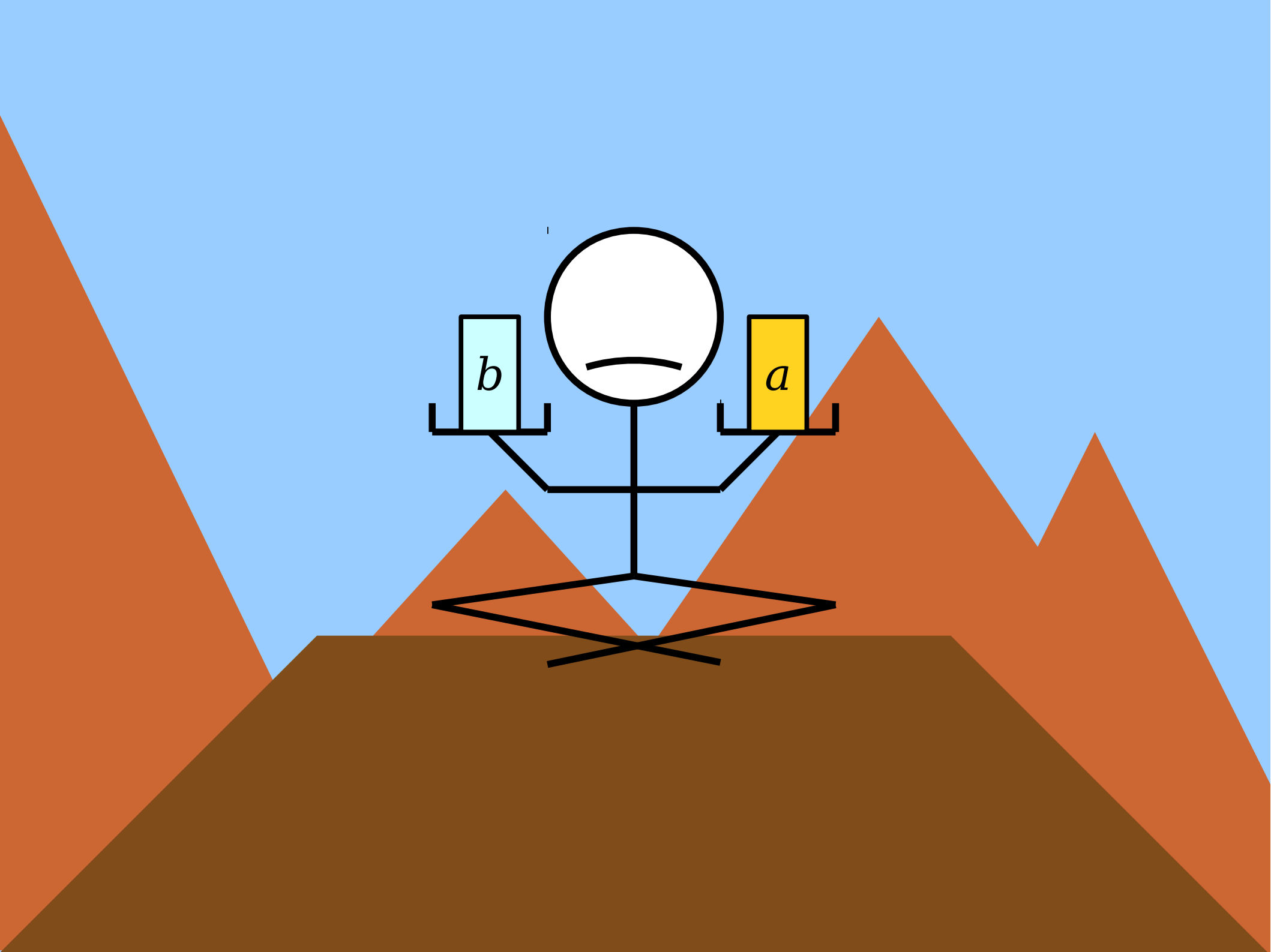

a
a

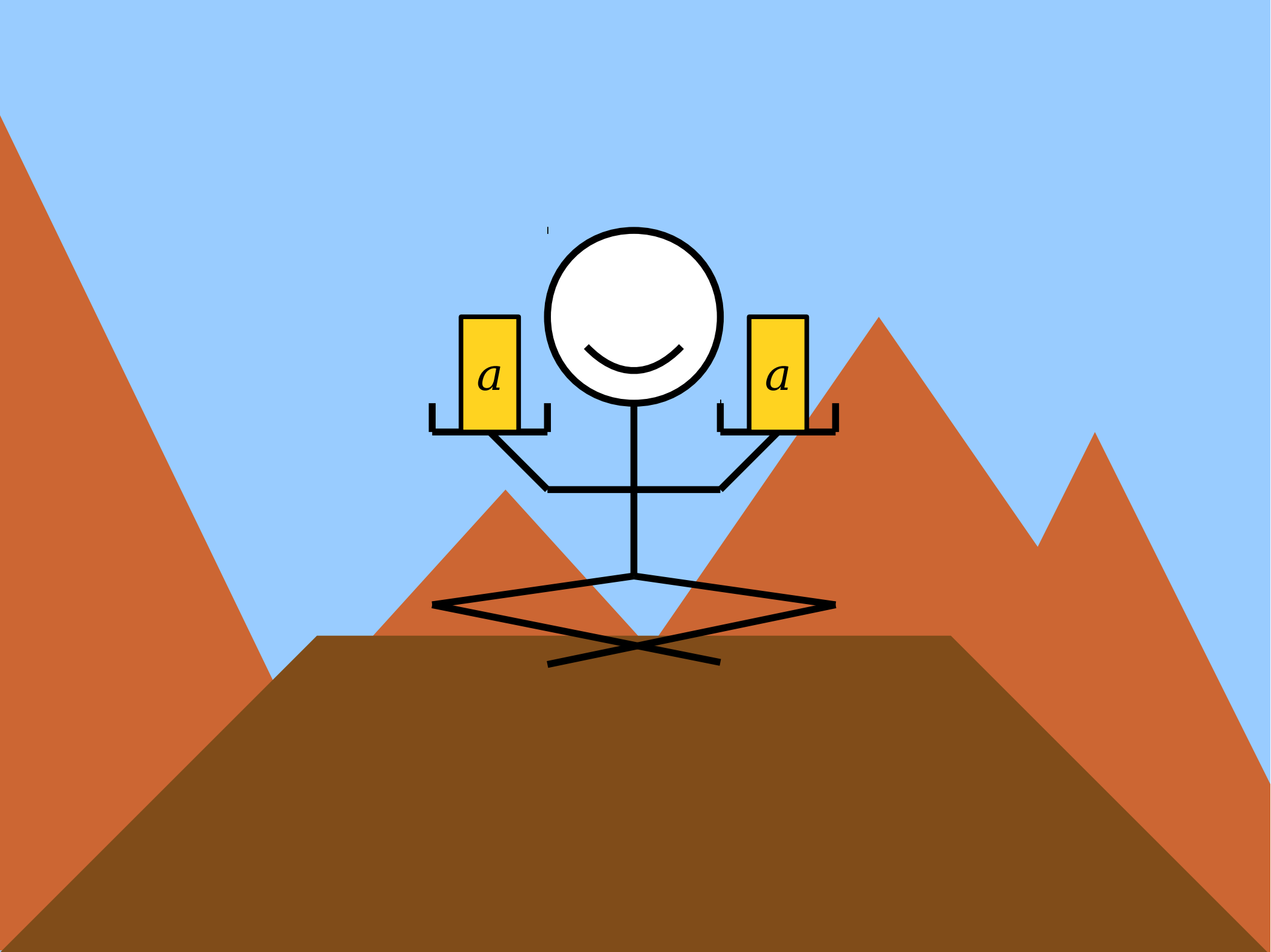

a
b

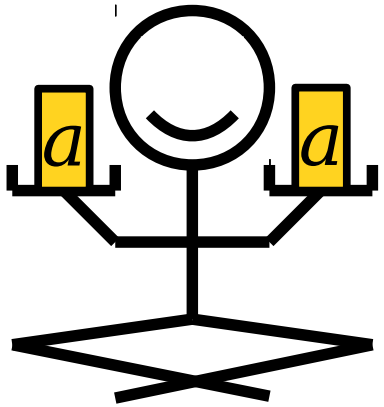

b
c

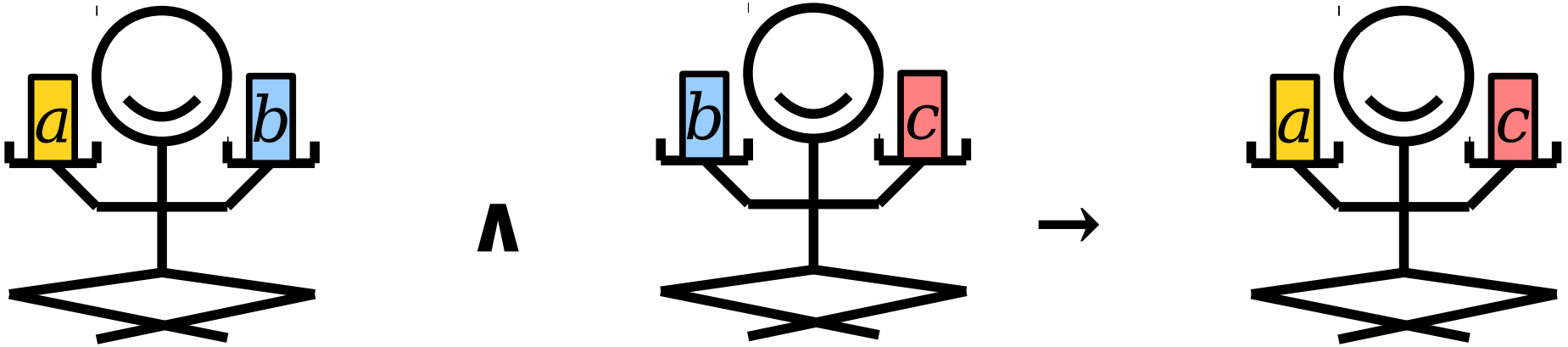

a
c

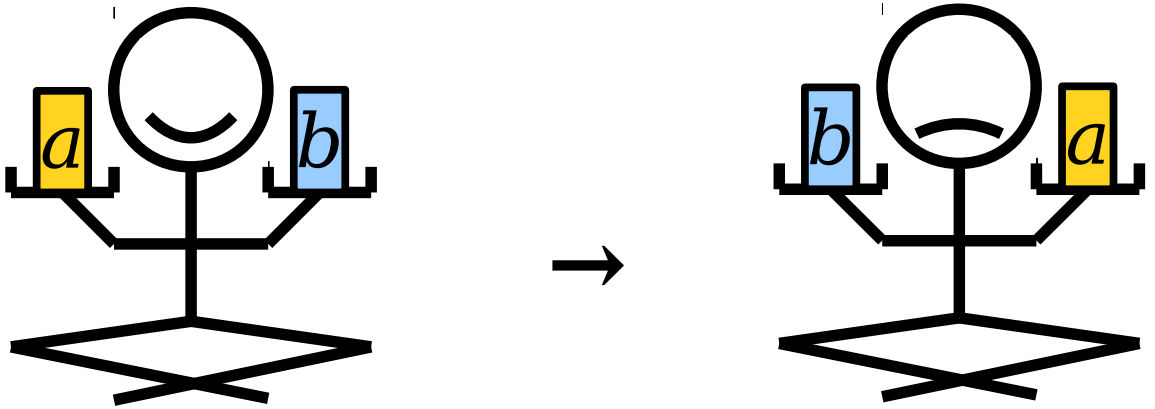

a
b


b
a


a
a







(*unless* $a = b$)

Reflexivity

Transitivity

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

Antisymmetry

- A binary relation R over a set A is called ***antisymmetric*** if the following is true:

$$\forall a \in A.\, \forall b \in A.\, (a \neq b \wedge aRb \rightarrow \neg(bRa))$$

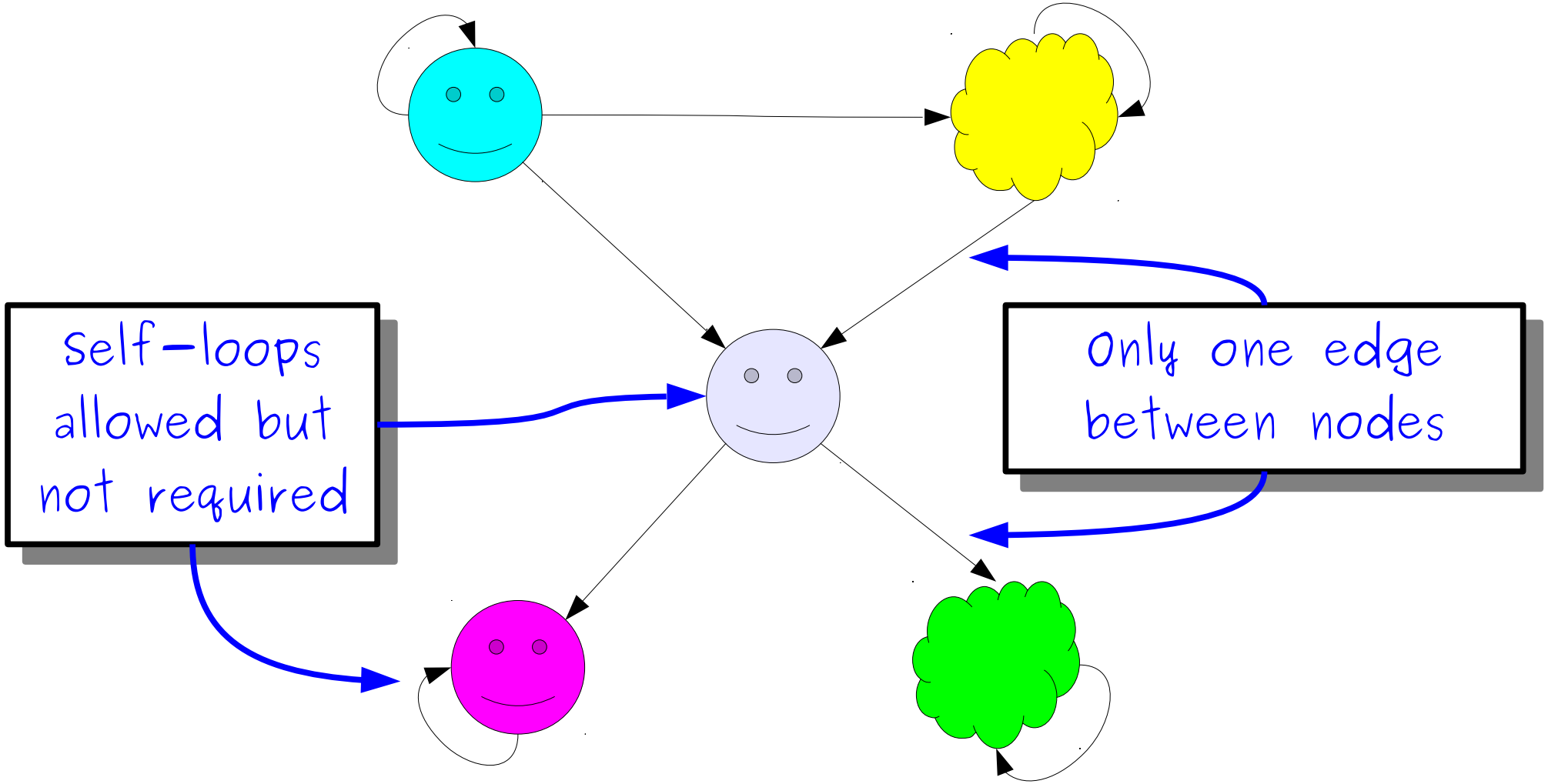
("If a is related to b and a ≠ b, then b is not related back to a.")

- Equivalently:

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

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

An Intuition for Antisymmetry



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

("If a is related to b and a ≠ b, then b is not related back to a.")

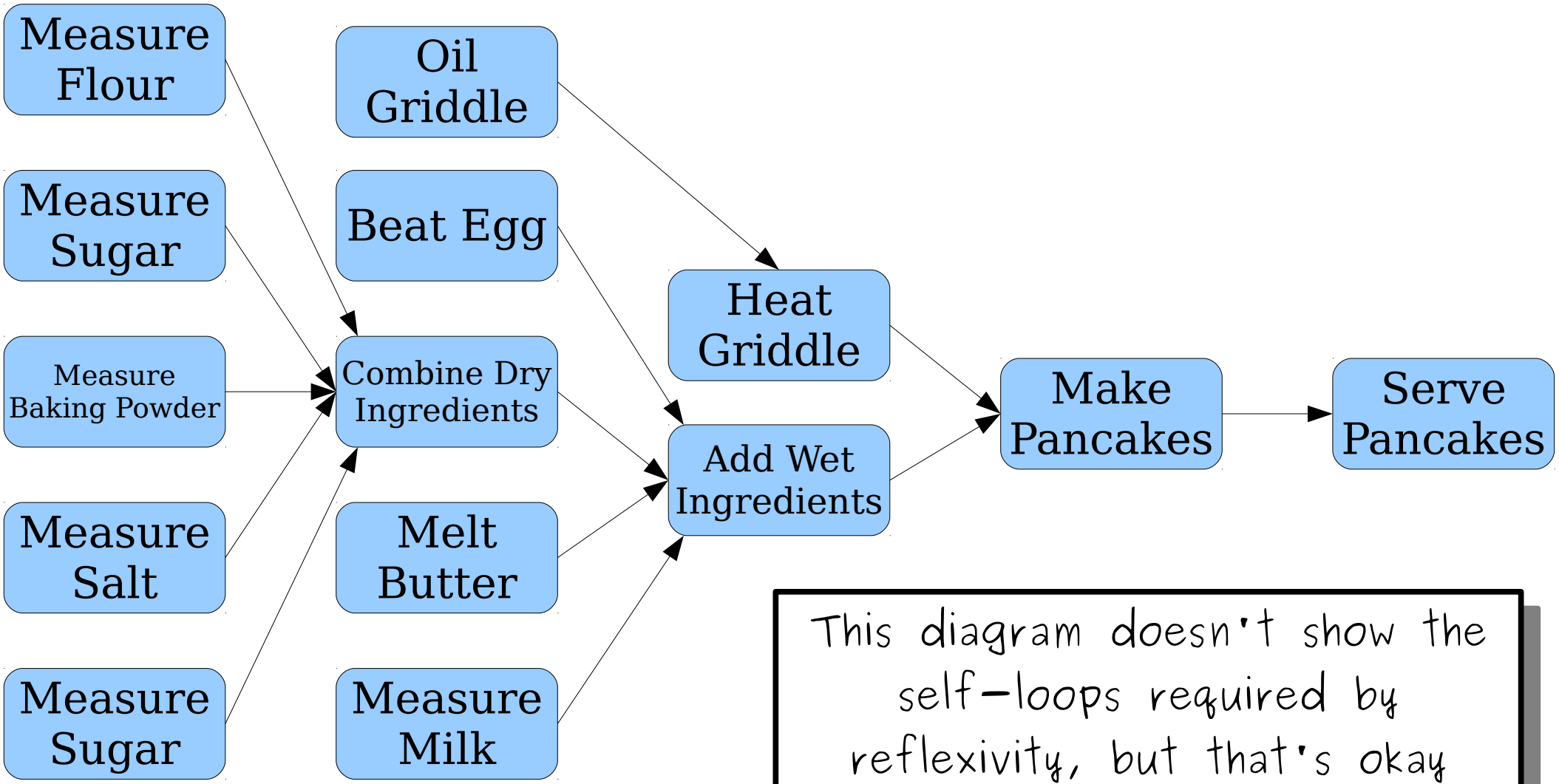
Antisymmetry

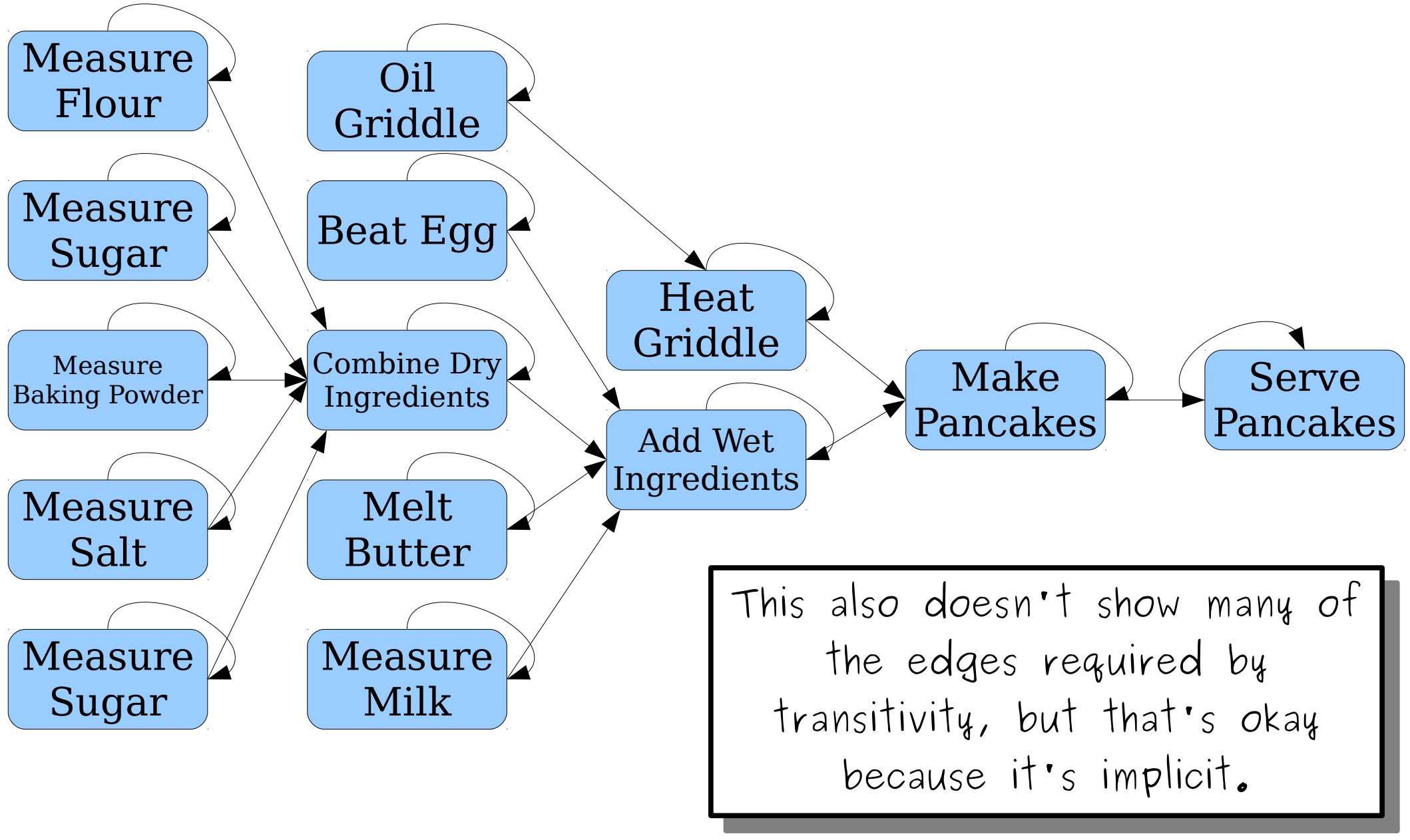
- Antisymmetry is probably the least intuitive of the four properties we've just seen.
- A few notes:
  - It's helpful to think of $\leq$ or $\subseteq$ as canonical examples of antisymmetric relations.
  - Antisymmetry is ***not*** the opposite of symmetry. There are relations that are both symmetric and antisymmetric (as you'll see in Problem Set Four).

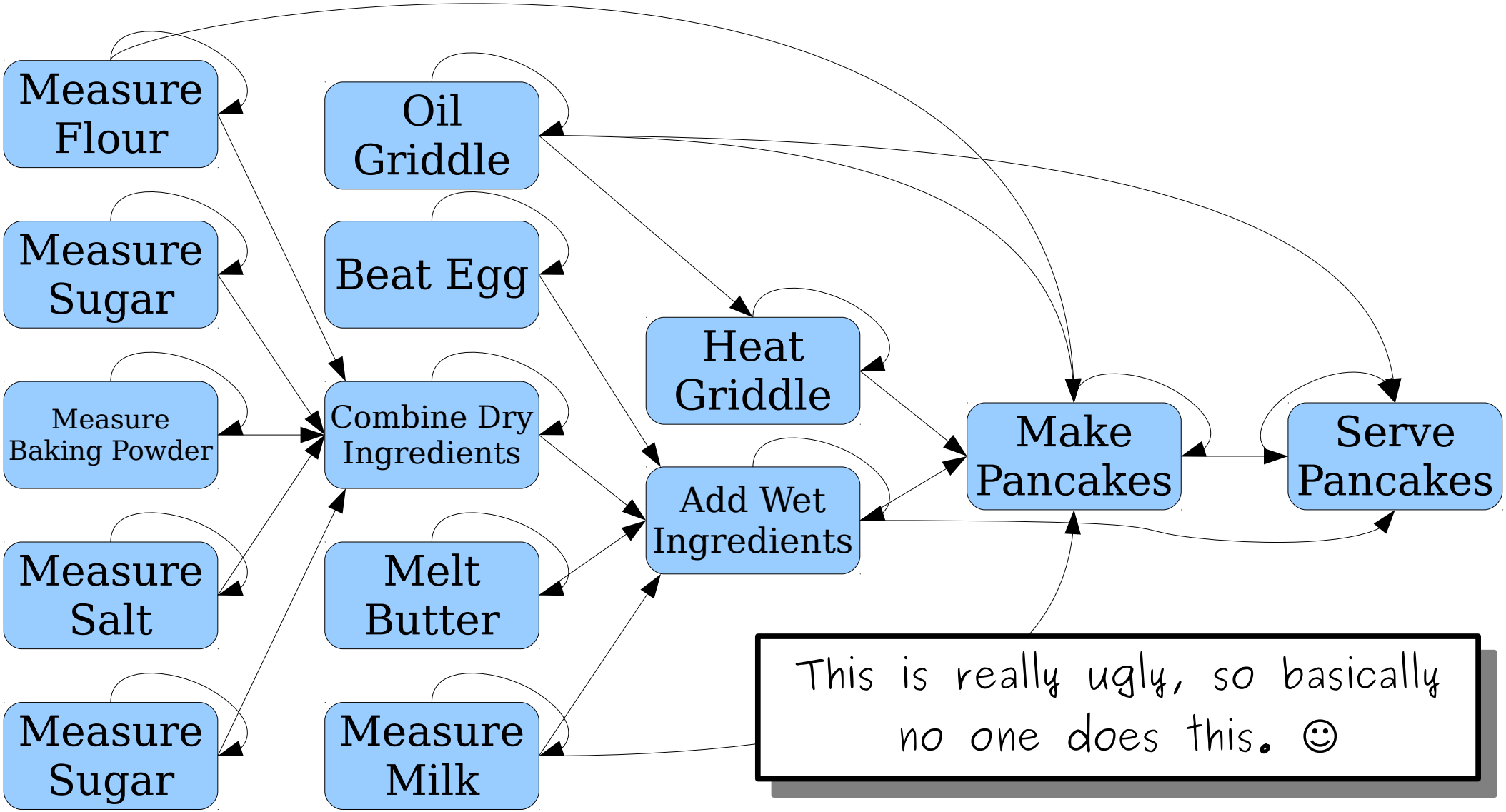
Partial Orders

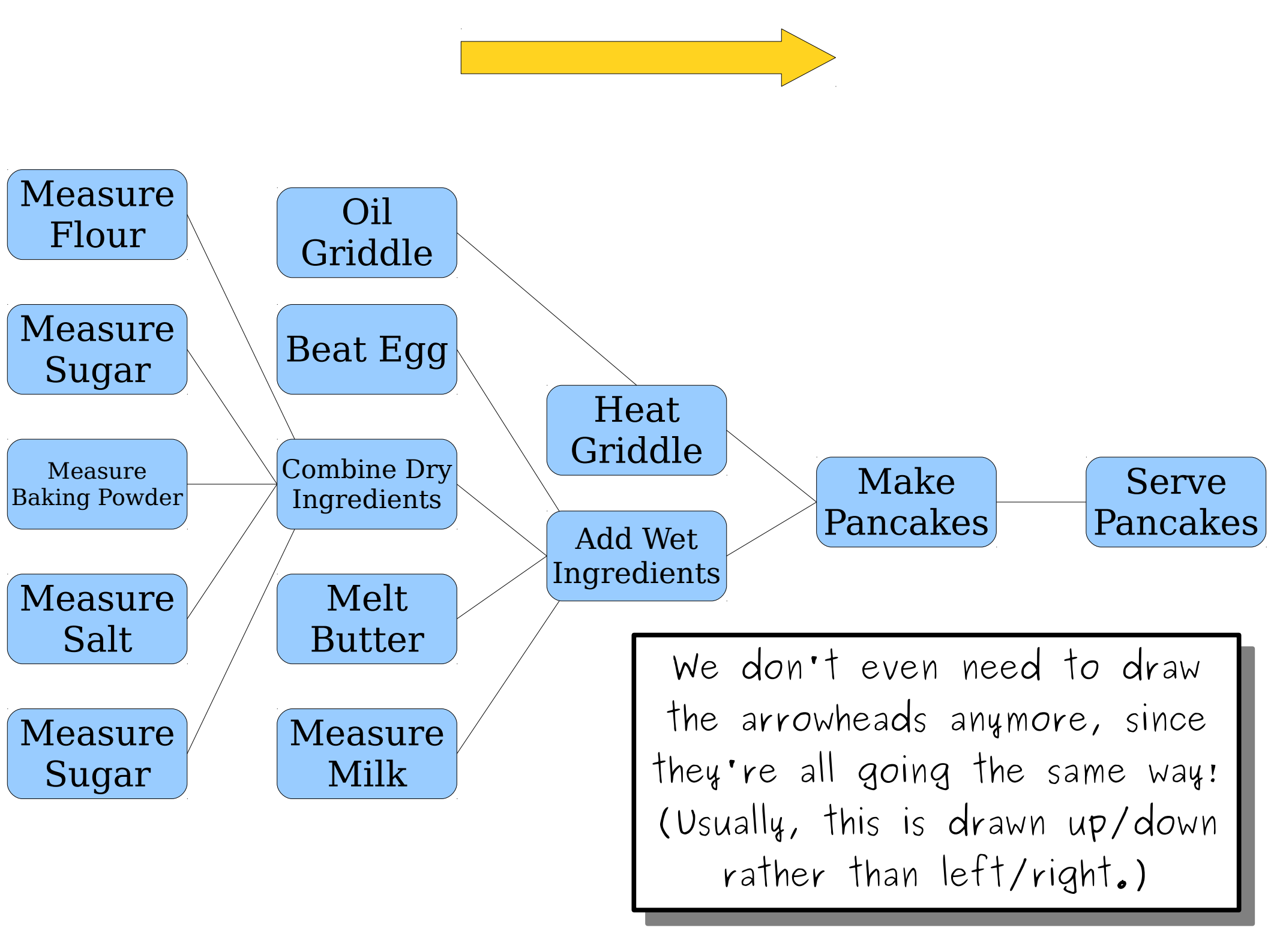
- A binary relation R over a set A is called a ***partial order*** if it is
  - reflexive,
  - **anti**symmetric, and
  - transitive.
- Intuitively, captures a prerequisite structure:
  - Everything must appear no later than itself.
  - If a must appear no later than b and $a \neq b$, then b can't appear earlier than a.
  - If a must appear no later than b and b must appear no later than c, then a must appear no later than c.

Drawing Partial Orders


Measure Flour
Measure Sugar
Measure Baking Powder
Measure Salt
Measure Sugar
Oil Griddle
Beat Egg
Combine Dry Ingredients
Melt Butter
Measure Milk
Heat Griddle
Add Wet Ingredients
Make Pancakes
Serve Pancakes
This diagram doesn't show the self-loops required by reflexivity, but that's okay because we implicitly know they're there.


Measure Flour
Measure Sugar
Measure Baking Powder
Measure Salt
Measure Sugar
Oil Griddle
Beat Egg
Combine Dry Ingredients
Melt Butter
Measure Milk
Heat Griddle
Add Wet Ingredients
Make Pancakes
Serve Pancakes
This also doesn't show many of the edges required by transitivity, but that's okay because it's implicit.

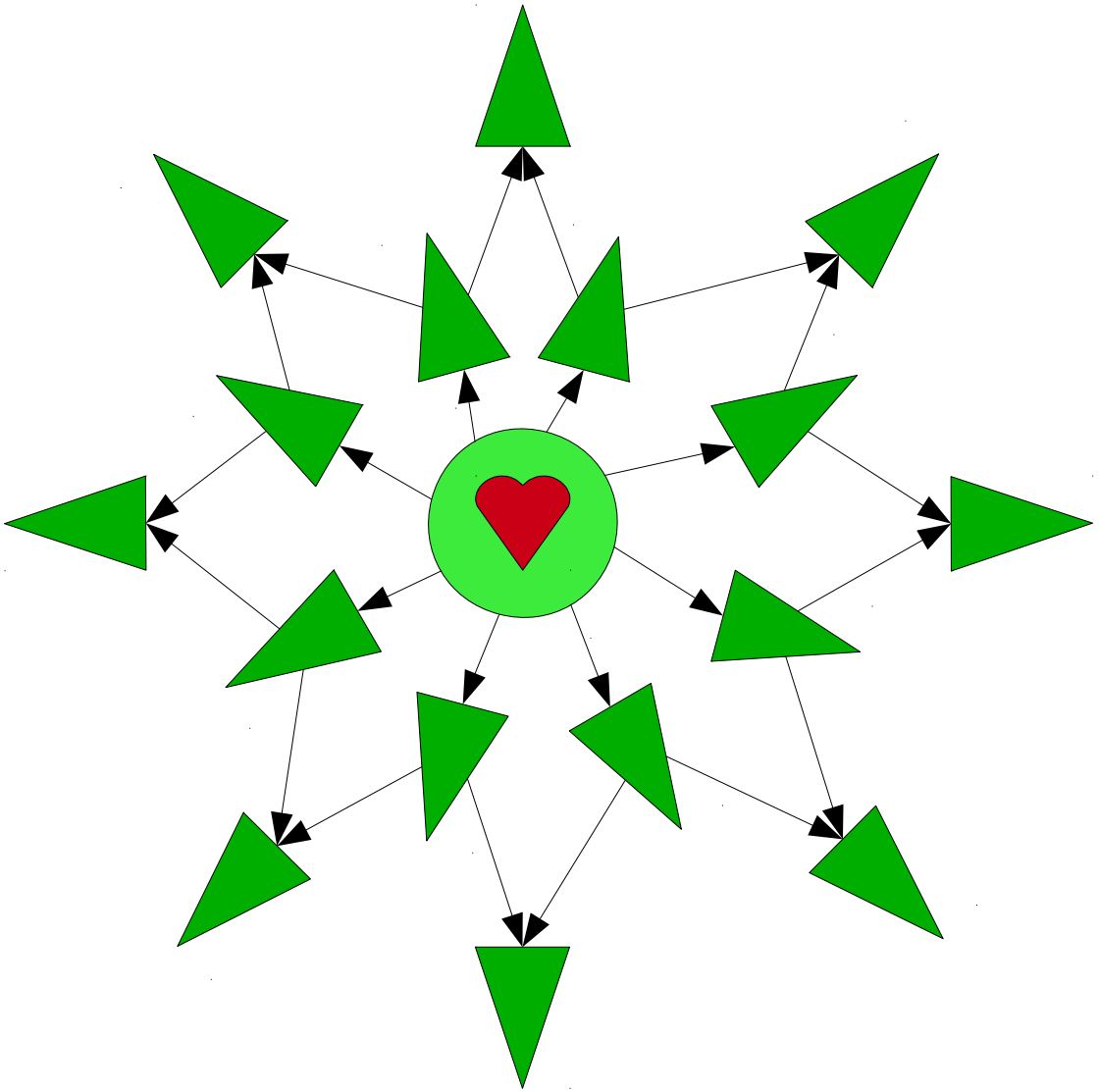

Measure Flour
Measure Sugar
Measure Baking Powder
Measure Salt
Measure Sugar
Oil Griddle
Beat Egg
Combine Dry Ingredients
Melt Butter
Measure Milk
Heat Griddle
Add Wet Ingredients
Make Pancakes
Serve Pancakes
This is really ugly, so basically no one does this. ☺


Measure Flour
Measure Sugar
Measure Baking Powder
Measure Salt
Measure Sugar
Oil Griddle
Beat Egg
Combine Dry Ingredients
Melt Butter
Measure Milk
Heat Griddle
Add Wet Ingredients
Make Pancakes
Serve Pancakes
We don't even need to draw the arrowheads anymore, since they're all going the same way! (Usually, this is drawn up/down rather than left/right.)

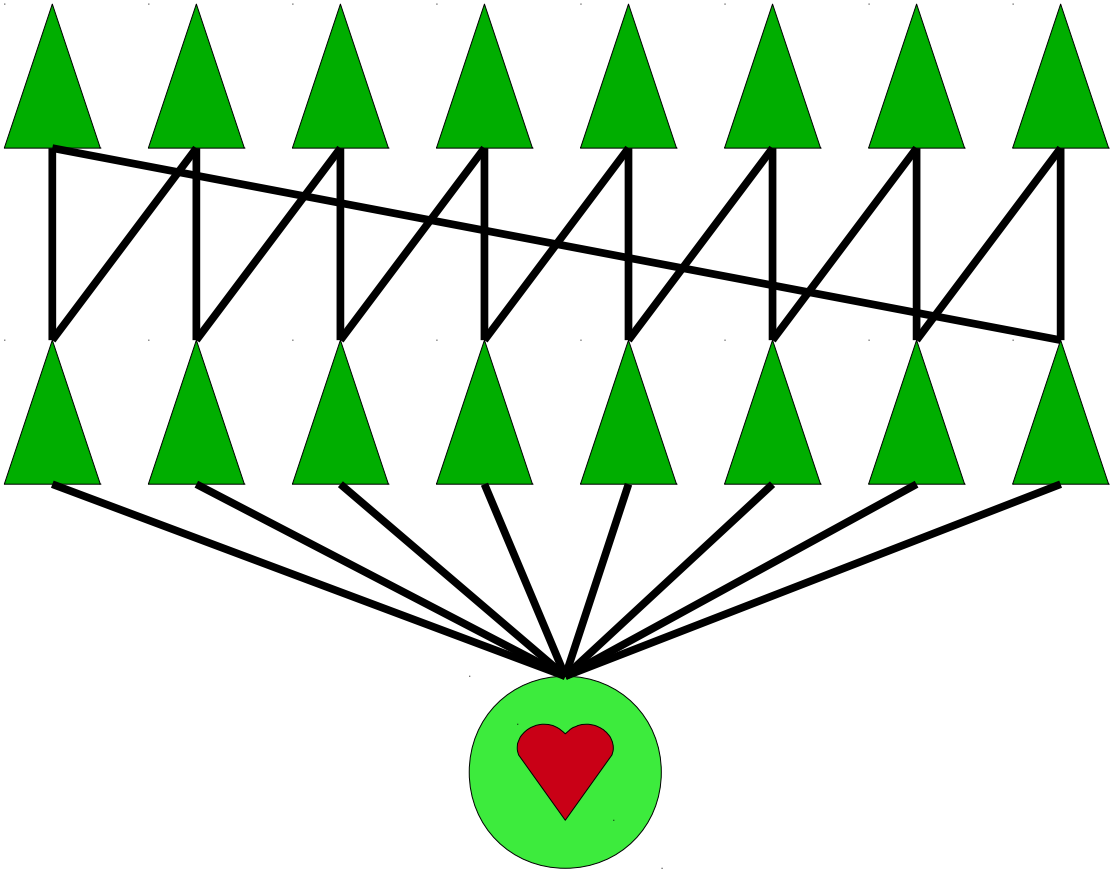
Hasse Diagrams

- A ***Hasse diagram*** is a graphical representation of a partial order.
- Elements are drawn from bottom-to-top (we used left-to-right in the previous slides, but that's the exception rather than the rule.)
- No self-loops: by ***reflexivity***, we can always add them back in.
- Higher elements are bigger than lower elements: by ***antisymmetry***, the edges can only go in one direction.
- No redundant edges: by ***transitivity***, we can infer the missing edges.

Hasse Artichokes



Hasse Artichokes



Summary

- Binary relations are another formalism for modeling relationships between objects.
- Different types of binary relations capture different structures:
  - ***Equivalence relations*** capture partitions and clusterings.
  - ***Partial orders*** capture prerequisite structures.
- Each of these terms has a formal definition based on simpler properties of binary relations.

The Binary Relation Editor

An Important Milestone

Recap: **Discrete Mathematics**

- The past four weeks have focused exclusively on discrete mathematics:

Induction	Functions
Graphs	The Pigeonhole Principle
Relations	Logic
Set Theory	Cardinality

- These are the building blocks we will use throughout the rest of the quarter.
- These are the building blocks you will use throughout the rest of your CS career.

Next Up: **Computability Theory**

- It's time to switch gears and address the limits of what can be computed.
- We'll explore these questions:
  - How do we model computation itself?
  - What exactly is a computing device?
  - What problems can be solved by computers?
  - What problems ***can't*** be solved by computers?
- ***Get ready to explore the boundaries of what computers could ever be made to do.***

Next Time

- **Formal Language Theory**
  - How are we going to formally model computation?
- **Finite Automata**
  - A simple but powerful computing device made entirely of math!
- **DFAs**
  - A fundamental building block in computing.