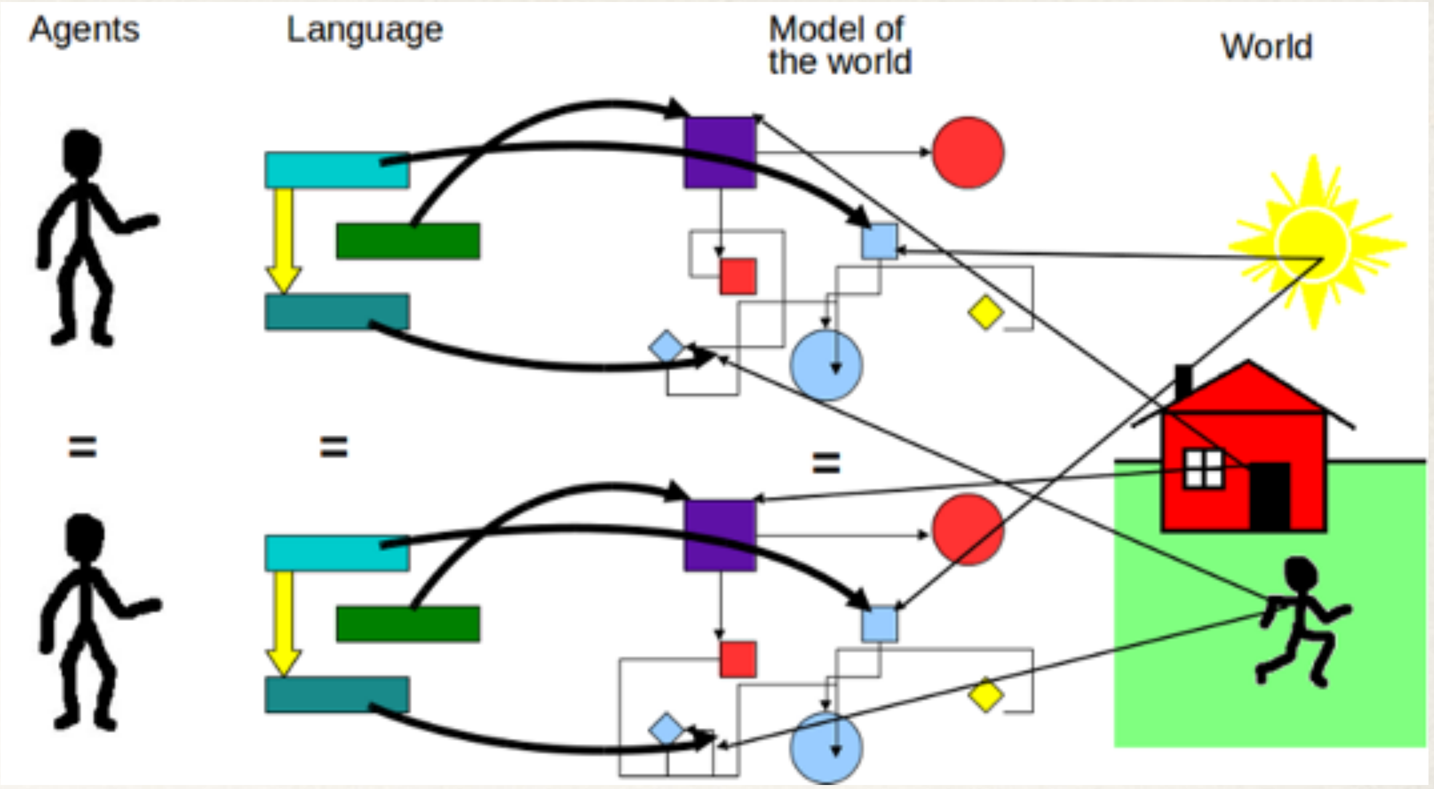
"The river is flowing surrounded by trees"



Language, meaning, and the world

PHIL 50 - Introduction to Logic

Marcello Di Bello, Stanford University, Spring 2014

Week 7 — Monday Class - Syntax and Semantics of Predicate Logic

This Week We'll Look at **Predicate Logic** More Closely

Syntax: rules to build well-formed formulas

Semantics: rules to assign (truth) values to these formulas

SYNTAX of Predicate Logic

The Ingredients of the Language of Predicate Logic

Constant symbols: ***a, b, c, …***

Variable symbols: ***x, y, z, …***

We shall refer to constant and variable symbols as **terms**

Predicate symbols: ***A, B, C, …***

Logical connectives or operators: $\neg, \wedge, \vee, \rightarrow$

Existential quantifier: $\exists x$ (*"there is an x"*)

Universal quantifier: $\forall x$ (*"for all x"*)

Inductive Definition of Formulas of the Language of Predicate Logic

Base case:

If $t_1, t_2, \ldots, t_k$ are terms, and P is a predicate, $P(t_1, t_2, \ldots, t_k)$ is a formula.

Inductive steps or cases:

If ϕ is a formula, then $\neg\phi$ is a formula.

If ϕ and ψ are formulas, then $(\phi \wedge \psi)$ is a formula.

If ϕ and ψ are formulas, then $(\phi \vee \psi)$ is a formula.

If ϕ and ψ are formulas, then $(\phi \rightarrow \psi)$ is a formula.

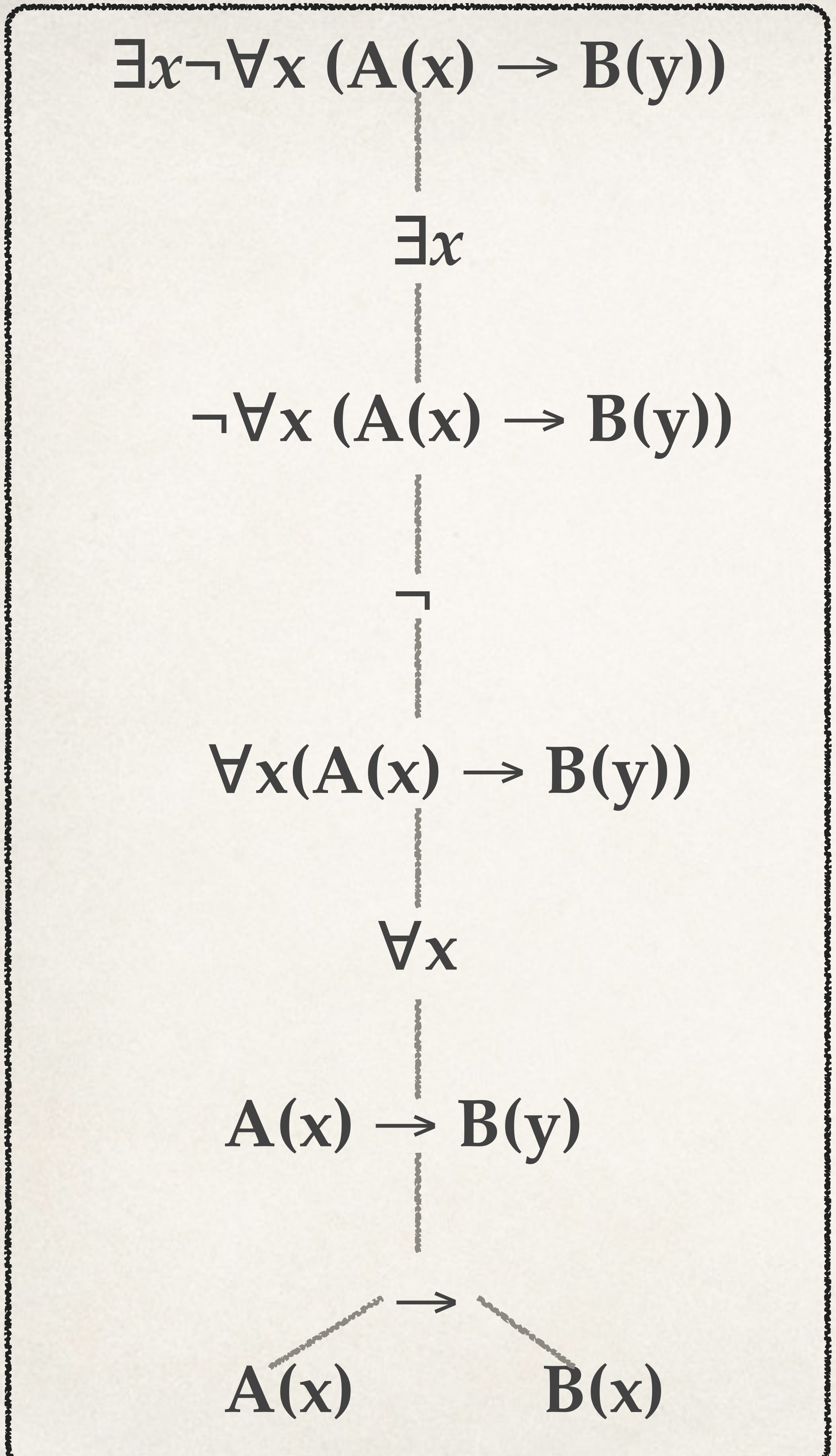
ϕ and ψ are not formulas, but *placeholders for formulas*

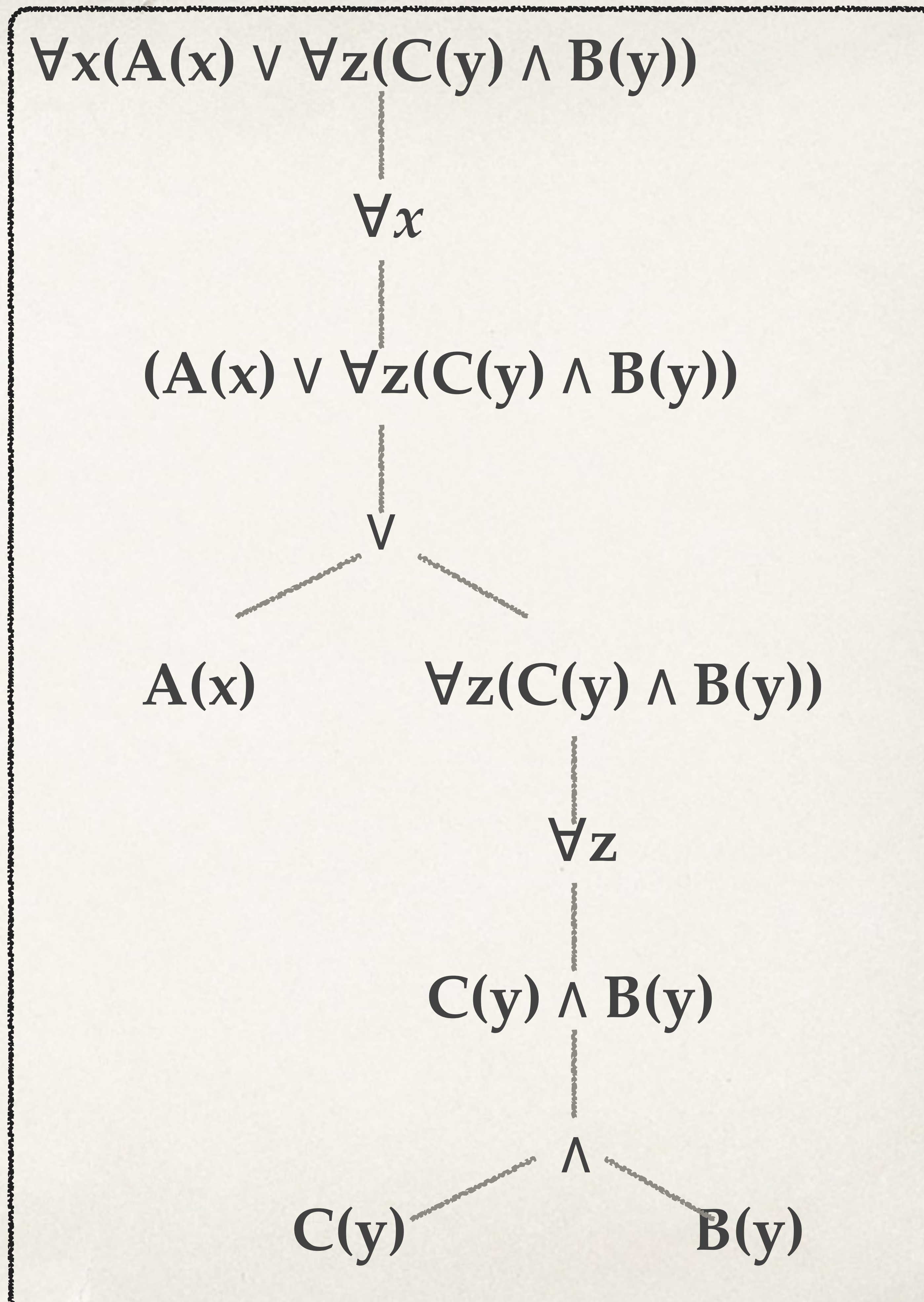
If ϕ is a formula and x is a variable, then $\exists x(\phi)$ is a formula.

If ϕ is a formula and x is a variable, then $\forall x(\phi)$ is a formula.

Final clause: Nothing else is a formula.

Think of the Syntax of Predicate Logic as a Set of Grammar Rules to Check Whether a Formula is Grammatical (i.e. Well-formed) or not.


∃x¬∀x (A(x) → B(y))
∃x
¬∀x (A(x) → B(y))
¬
∀x(A(x) → B(y))
∀x
A(x) → B(y)
→
A(x)
B(x)


∀x(A(x) ∨ ∀z(C(y) ∧ B(y))
∀x
(A(x) ∨ ∀z(C(y) ∧ B(y))
∨
A(x)
∀z(C(y) ∧ B(y))
∀z
C(y) ∧ B(y)
∧
C(y)
B(y)

SEMANTICS of Predicate Logic

$$A(a) \land B(c)$$

$$Eat(a, b) \rightarrow Sleep(a)$$

$$\neg Eat(a, b) \rightarrow Starve(a)$$

*Today we will only consider how to interpret formulas **without** variables and quantifiers. We will look at the semantics for quantified formulas on Wednesday.*

Before we Painstakingly Delve into the Semantics of Predicate Logic, Let's Think About it for a Moment

A Language as a Combination of Syntax and Semantics

Any language—be it a natural language or a formal language—has grammatical rules for how to construct grammatical sentences (or formulas, in the case of a formal language). This is the **syntax.**

From the syntactic point of view, a language is just a bunch of symbols put together according to grammar rules.

But in order to serve any purpose at all, the symbols must be assigned a meaning. This is the **semantics** of the language.

How is it that symbols can be assigned meaning? How can they acquire meaning?

How can THIS mean THAT?

"The river is flowing surrounded by trees"

A Simple Suggestion

"The river is flowing surrounded by trees"

The word "river" means RIVER-IN-REALITY

The word "trees" means TREES-IN-REALITY

The word "surrounded" means SURROUNDED-IN-REALITY

The word "flowing" means FLOWING-IN-REALITY

But is that all there is to the linguistic meaning of words and sentences?

What about the Meaning of "Is" and of the Logical Connectives?

"The river IS flowing"

"The river IS NOT flowing"

"The river is flowing AND the trees are surrounding it.

Etc.

It seems that words such as "is", "not", "and" etc. do not have any direct correspondence to things in reality.

Another Surprising Aspect of Language is that Sentences Can Be True or False. How Can That Be?

How can a Sentence be TRUE or FALSE?

"The river is flowing surrounded by trees"

NB: Words alone cannot be true or false, but sentences can. How can that be?

What is truth?

As You Learn the Semantics of Predicate Logic, Think of the Questions:

(Q1)
How is it that words and sentences end up possessing meaning?

(Q2) How is it that sentences (but not words alone) can be true or false? And more generally, what is truth?

We Now Need to Introduce the Notion of a **MODEL**

Model (and Truth in a Model)

A model M is a tuple $\langle D, I, g \rangle$ where

D is the **domain**, i.e. *D* is a non-empty set of objects

I is an **interpretation function** that behaves as follows:

I assigns to every constant symbol an element of *D*

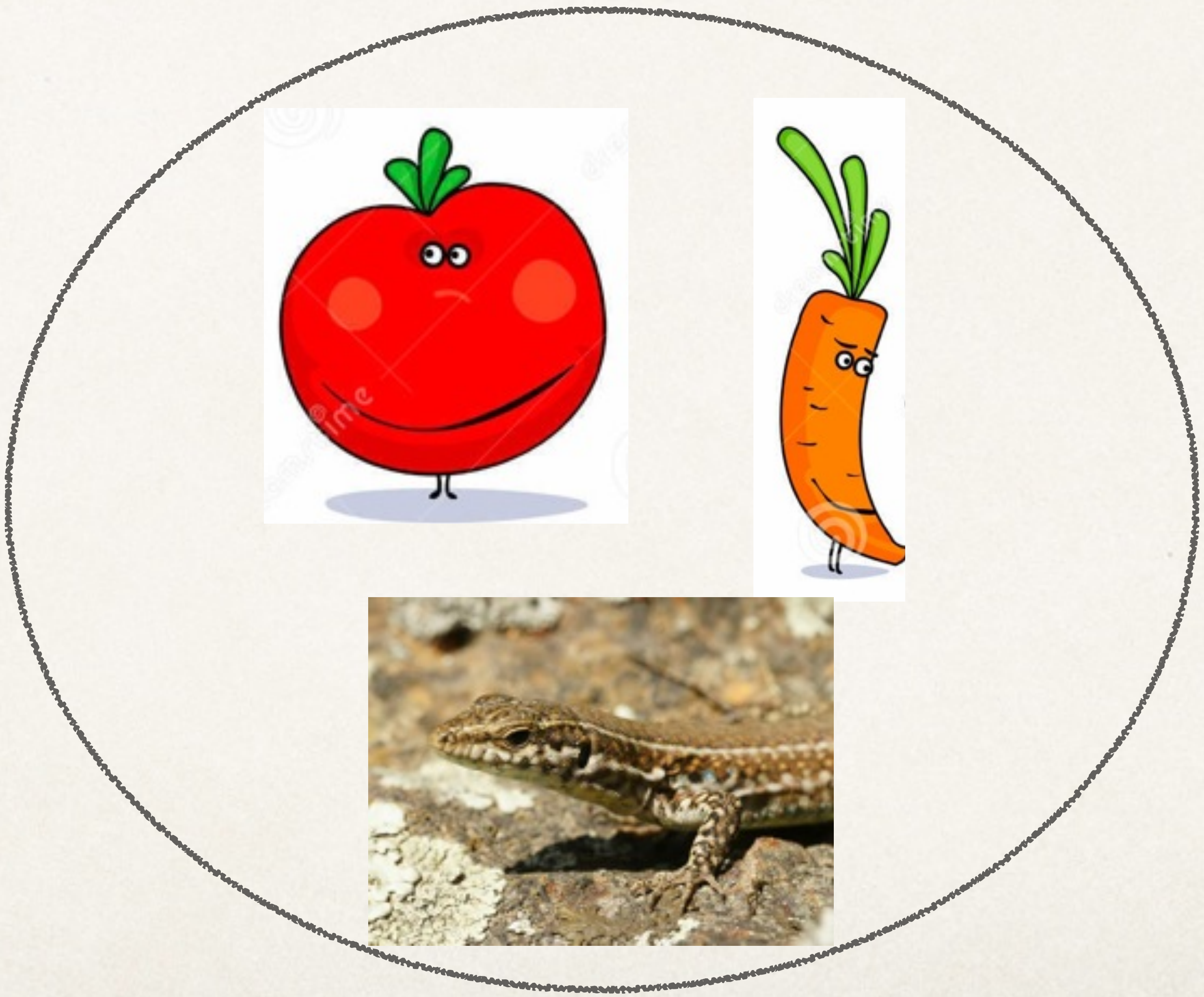
I assigns to every *1*-place predicate symbol a subset of *D*

I assigns to every **2**-place predicate symbol a subset of $D \times D$

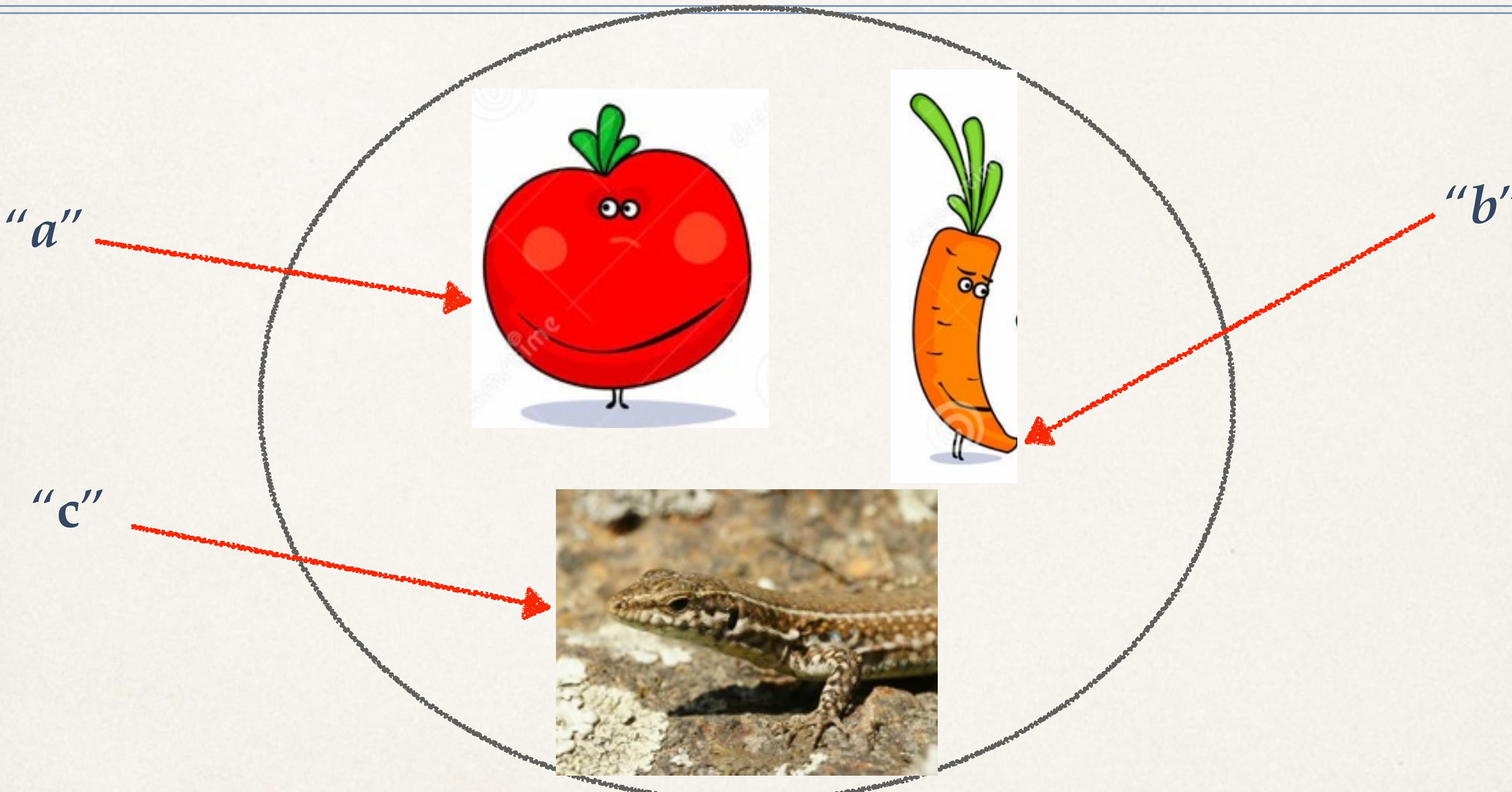
***g* [we will discuss "g" tomorrow; "g" interprets variables]**

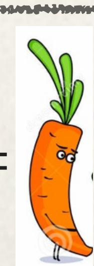
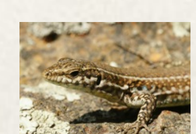
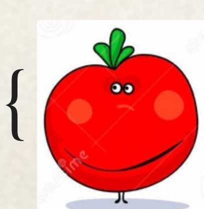
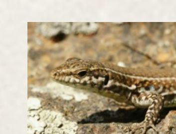
$M \vDash \phi$ *iff* ϕ is true in (relative to) model **M** which is $\langle D, I, g \rangle$

Example of a Domain **D**

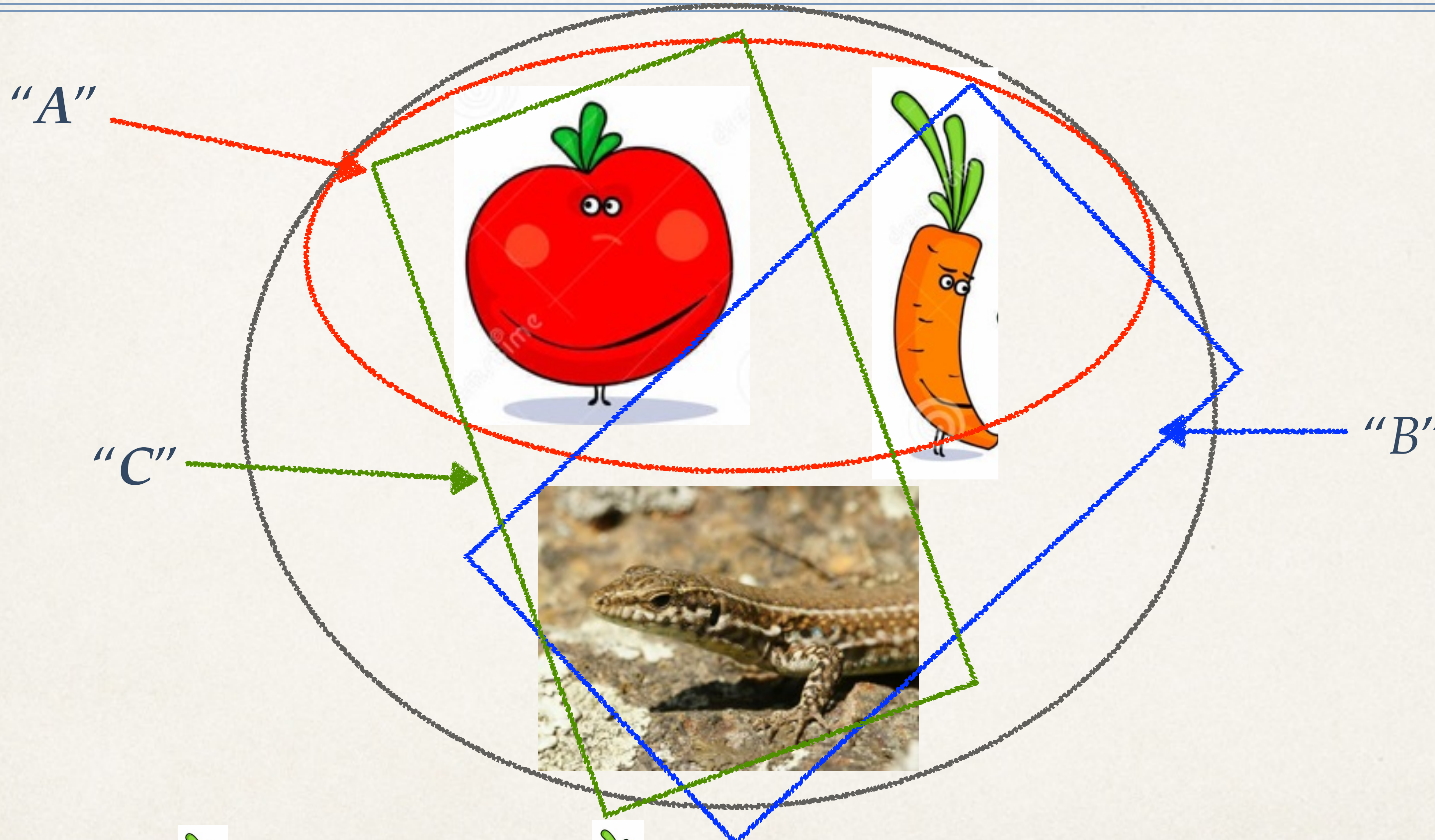


The Interpretation Function for Constant Symbols



$I(a) =$  $I(b) =$  $I(c) =$  $D = \{$  ,  ,  $\}$

The Interpretation Function for 1-place Predicate Symbols



$I(A) = \{$  ,  $\}$ $I(B) = \{$  ,  $\}$ $I(C) = \{$  ,  $\}$

The Basic Idea

Constant symbols refer to **objects**

(i.e. ***I*** assigns an object to every constant symbol)

Predicate symbols refer to **sets of objects**

(I.e. ***I*** assigns a set of objects to every predicate symbol)

How To Assess the Truth of Formulas Containing Constant Symbols and 1-place Predicates

The rough idea is that the formula ***A(a)*** is true whenever the objects which corresponds to the constant symbol ***a*** is in the set of objects which correspond to the predicate symbol ***A***.

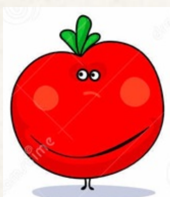
Let **P** be a *1-place predicate symbol* and let ***c*** be a constant symbol. We have:

$$\boldsymbol{M} \vDash \mathbf{P}(\boldsymbol{c}) \text{ iff } \boldsymbol{I}(\boldsymbol{c}) \in \boldsymbol{I}(\boldsymbol{P})$$

Recall: **M** is $\langle \boldsymbol{D}, \boldsymbol{I}, \boldsymbol{g} \rangle$

$$\boldsymbol{M} \nvDash \mathbf{P}(\boldsymbol{c}) \text{ iff } \boldsymbol{I}(\boldsymbol{c}) \notin \boldsymbol{I}(\boldsymbol{P})$$

Illustration

$I(a) =$  $I(b) =$  $I(c) =$ 

$I(A) = \{$  ,  $\}$ $I(B) = \{$  ,  $\}$ $I(C) = \{$  ,  $\}$

$\mathbf{M} \vDash \mathbf{A}(\boldsymbol{b})$ *because* $\boldsymbol{I(b) \in I(A)}$

$\mathbf{M} \vDash \mathbf{B}(\boldsymbol{c})$ *because* $\boldsymbol{I(c) \in I(B)}$

$\mathbf{M} \nvDash \mathbf{C}(\boldsymbol{b})$ *because* $\boldsymbol{I(b) \notin I(C)}$

From 1-place Predicates to 2-place Predicates

1-place predicates:

American(…)

Fruit(…)

2-place predicates:

Eat(…, …)

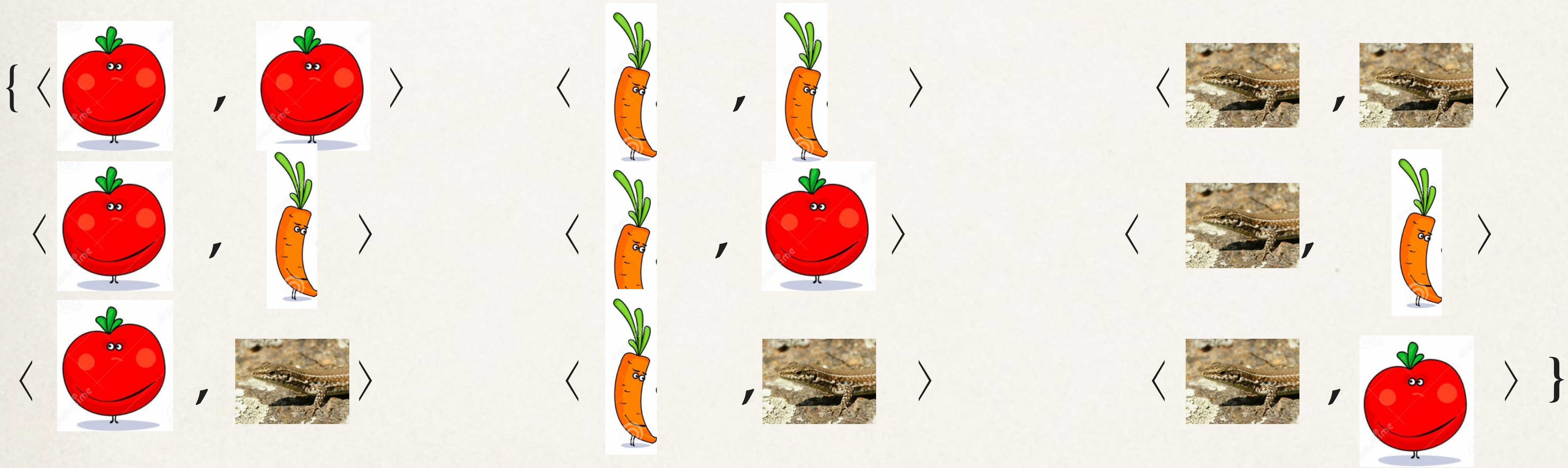
Like (…, …)

In order to give an interpretation for 2-place predicates, we should talk about sets of ordered pairs of objects

Ordered Pairs

Consider the domain D= { , ,  }.

The set $D \times D$ is the **<u>set</u> of all 9 ordered pairs**, as follows:



$D \times D$ is called the **Cartesian Product for** D and consists of all ordered pairs that can be obtained from D.

Sets of Objects versus Sets of Ordered Pairs of Objects

A *1-place predicate* is interpreted set-theoretically as a **set of objects** (of those objects which satisfy the 1-place predicate in question).

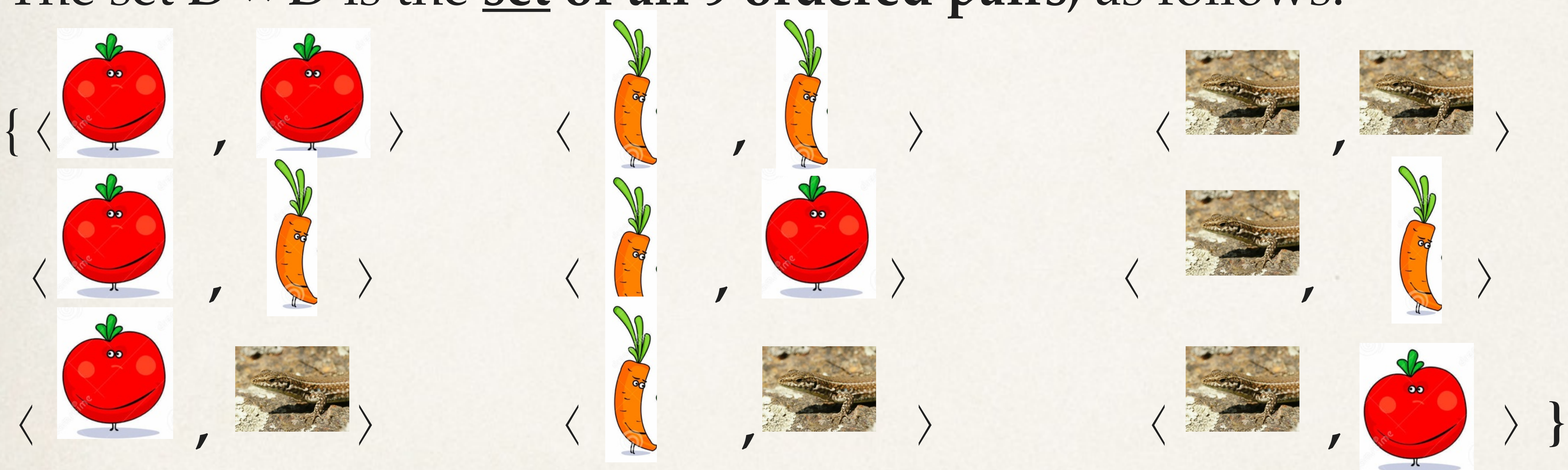
Similarly, a *2-place predicate* is interpreted set-theoretically as a **set of ordered pairs of objects** (of those ordered pairs which contain objects that satisfy the 2-place predicate in question).

<u>Example</u>: the interpretation of the 2-place predicate ***Eat(… , …)*** is the set of ordered pairs of objects such that the first object in the pair eats the second object in the pair.

Illustration

*The interpretation **I** assigns a subset of **D** × **D** to each 2-place predicate symbol.*

The set ***D*** × ***D*** is the **<u>set</u> of all 9 ordered pairs**, as follows:



Example*: *I(Eat) = {  }

Truth for Formulas Containing Constant Symbols and Predicates

The rough idea is that *e.g.* the formula ***Eat(a, b)*** is true whenever the object that corresponds to the constant symbols ***a*** and the object that corresponds to the constant symbol ***b*** form an ordered pair that is in set of ordered pairs that correspond to the *2-place* predicate ***Eat***.

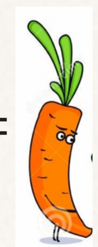
Let $\mathbf{P^2}$ be a *2-place predicate symbol* and let $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$ be constant symbols. We have:

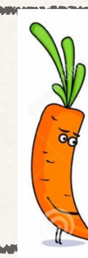
$\mathbf{M} \vDash \mathbf{P^2}(c_1, c_2)$ *iff* $\langle \mathbf{I(c_1), (I(c_2)} \rangle \in \mathbf{I(P^2)}$

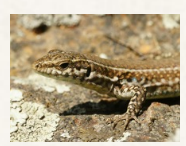
Recall: **M** is $\langle \boldsymbol{D, I, g} \rangle$

$\mathbf{M} \nvDash \mathbf{P^2}(c_1, c_1)$ *iff* $\langle \mathbf{I(c_1), (I(c_2)} \rangle \notin \mathbf{I(P^2)}$

Assessing the Truth of Formulas with Constants and Predicate Symbols

$I(a) =$  $I(b) =$  $I(c) =$ 

$I(A) = \{$  ,  $\}$ $I(B) = \{$  ,  $\}$ $I(C) = \{$  ,  $\}$

$I(Eat) = \{ \langle$  ,  $\rangle, \langle$  ,  $\rangle \}$

$\mathbf{M} \models \mathbf{A(b)}$

$\mathbf{M} \models \mathbf{Eat}(c, a)$

$\mathbf{M} \not\models \mathbf{Eat}(a, b)$

because $\mathbf{I(b)} \in I(A)$

because $\langle \mathbf{I(c), I(a)} \rangle \in \mathbf{I(Eat)}$

because $\langle \mathbf{I(a), I(b)} \rangle \notin \mathbf{I(Eat)}$