Science of Knowing

Mathematics

Posina Venkata Rayudu

Consciousness Studies Programme

National Institute of Advanced Studies

Science of Knowing: Mathematics

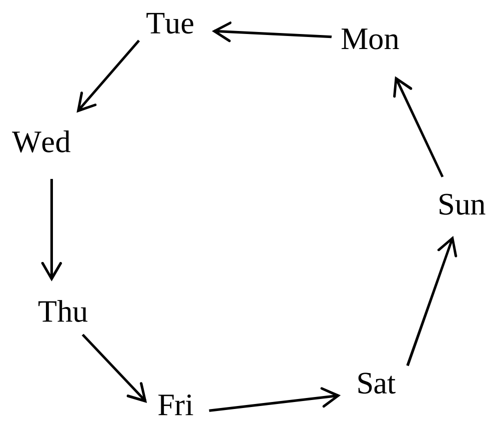
Overview

The **Science of Knowing: Mathematics** textbook is as much about cognition as it is about mathematics. Mathematics, for our present purposes, is a means to understand the mind, brain, and consciousness. As is often the case with any tool, before we begin to use mathematics to probe the workings of the mind or of the brain, we need to know how math works. Though mathematical calculations have been used as a means of knowing since antiquity, it is only during the past fifty years or so that we began to develop a clear mathematical understanding of acquiring mathematical knowledge (Lawvere, 2004, 2013). Category theory, which embodies these mathematical advances, is the main focus of our study (Lawvere and Schanuel, 2009, p. 3). The basic idea underlying the definition of mathematical category is: objects (of a given universe of discourse) are not specified until the relations between these objects are clearly spelled out (Lawvere, 1991). The reach of this mathematical insight extends far and wide—all the way to the current applications of category theory to cognitive neuroscience (Ehresmann and Vanbremeersch, 2007). As an illustration, let us consider the standard neuroscientific account of the mind: "mind is a set of processes carried out by the brain" (Kandel, 2013). The contents of the mind, however, are unlike the discrete elements of a set, and are related to one another in well-defined ways. For example, thinking changes the way we see a given visual image (Albright, 2012), and emotions change the way we think about a given situation (Helmuth, 2001). One of the objectives of the present course is to begin refining, with the help of category theory, the contemporary scientific conceptualization of the mind.

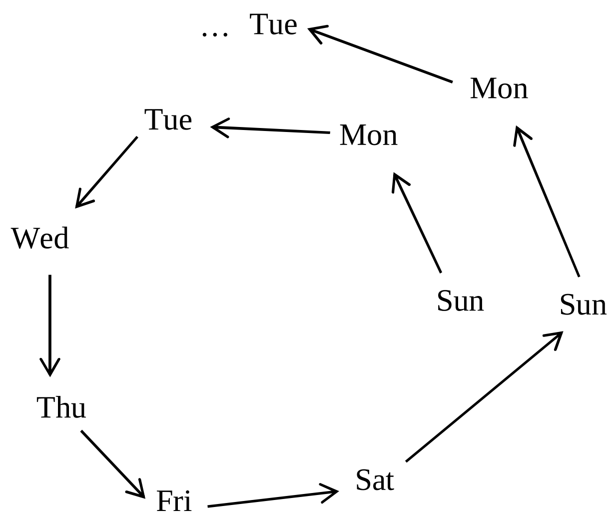
How are we going to refine our understanding of the mind using category theory? Let us begin with category theory. The categorical method is, in essence, a way of describing an object of interest such that (i) the description is reflective of the object and (ii) a systematic study of the description, hopefully, reveals something we did not know about the object (Lawvere, 1972, p. iv; Lawvere and Schanuel, 2009, pp. 135-136). These abstract generalities can be understood in terms of an everyday example: WEEK. As a first approximation, a week can be mathematically described using a number: 7 (as in 7 days). A more refined description of the week is in terms of sets:

W = {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}

Looking at our above description, we notice that the days of a week, unlike the discrete elements of a set, are related to one another. For instance, Tuesday comes after Monday, Monday comes after Sunday… We can capture this additional structure by way of describing the week as a set with added structure i.e. the set W equipped with an endomap *w*: W –> W, which specifies for each day of the week the following day (as depicted below):

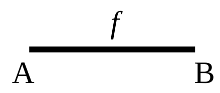


Looking at the above mathematical description of the days of a week, we realize that, unlike days in the above description, the Sunday that comes after Saturday is not the same Sunday that came before Monday. This added realization leads us to a further refined description in terms of a product of structured sets: $w \times n$ (n: N –> N, n(n) = n + 1, N = {0, 1, 2…}; Lawvere and Schnauel, 2009, pp. 239-240) as shown below:

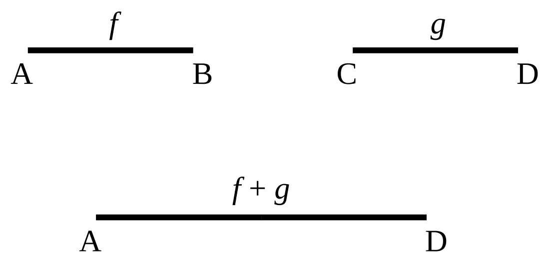


Now that we have seen a rough sketch of the categorical method, let us go over the category theory that we are going to study in the present course. We will begin with collections of things or sets. Sets are little less abstract than numbers and have zero structure; these attributes make them particularly suitable to build mathematical structures of all sorts (Lawvere, 1994a; Lawvere and Rosebrugh, 2003, p. 1; Lawvere and Schanuel, 2009, p. 81, 146). Next, we will look at functions. More specifically, we will study various types of functions such as sections, retracts, and isomorphisms. These functions can be thought of as specifications of various possible relationships between sets (Lawvere and Schanuel, 2009, pp. 39-59). Another useful way of thinking about functions is to think of a function f: A –> B as an A-shaped figure in B (Lawvere and Schanuel, 2009, pp. 81-85).

A function *f*: A –> B from a domain set A to a codomain set B can also be imagined as a line segment *f* with the two sets A, B as its two endpoints (as shown below):



Carrying this imagination further, we notice that we can join two line segments (*f*, *g*) into a line segment (*f* + *g*; with '+' denoting joining) if an endpoint (C) of one line segment (*g*) coincides with an endpoint (B) of the other line segment (*f*):



Formalization of the above geometric idea gives us the definition of composition of functions. The all-important definition of CATEGORY can be obtained by noting the properties of functions with respect to composition (Lawvere and Schanuel, 2009, p. 21).

Next, we will begin the study of the category of dynamical systems (also known as endomaps), wherein the far-reaching idea of structure-preserving morphism is introduced (Lawvere and Schanuel, 2009, pp. 136-137). Structure-preserving morphisms are ways of transforming one object into another without tearing apart the structure of objects (Lawvere and Schanuel, 2009, p. 210). Neurons switching between firing and resting states readily lend themselves to be modeled in terms of the category of dynamical systems. A subcategory of the category of dynamical systems is the category of idempotents, which captures an important aspect of learning (Lawvere and Schanuel, 2009, p. 106).

In addition to the notions of category and structure-preserving morphism, universal mapping property is another profound concept that has been abstracted from the practice of mathematics. Objects, which are often described in terms of their content, can also be described in terms of their relations to other objects (Lawvere, 1966, 2005). For example, a single-element set **1** = {•} can be described as a set to which there is exactly one function from every set (Lawvere and Schanuel, 2009, p. 213).

We all know 1 + 1 = 2. In asking what is special about the two-element set **2** = {•, •} obtained as the sum of two one-element sets ({•} + {•}), we arrive at the definition of SUM as a universal mapping property (Lawvere and Schanuel, 2009, pp. 265-267). We will also look at the duality between SUM and PRODUCT. Sums and products of objects more structured than sets such as dynamical systems and graphs can be calculated using the universal mapping property definitions. We will also see how arithmetic differs across different categories. For example, in the category of pointed sets we find that 1 + 1 = 1 (Lawvere and Schanuel, 2009, pp. 295-298). Further generalization of the definition of sum so as to accommodate putting together of not only objects but also the relations between objects leads to the notion of colimit (Lawvere and Rosebrugh, pp. 72-73). We will see how colimits are used to formalize the binding problem of putting together colors and shapes into the colored-shapes of our conscious experience (Ehresmann and Vanbremeersch, 2007).

Objective logic of a given universe of discourse can be calculated in terms of the whole-part relations of the objects of the given universe (Lawvere and Rosebrugh, 2003, pp. 193-195, 239-240). We will see how logic varies across different domains of discourse (Lawvere and Schanuel, 2009, pp. 339-347). In stark contrast to the Boolean logic of the category of sets, the logics of dynamical systems and of graphs admit degrees of truth and varieties of negation, wherein, for example, the familiar not(not(A)) = A can fail (Lawvere and Schanuel, 2009, p. 355; Lawvere and Rosebrugh, 2003, pp. 200-201).

With the aforementioned basics of category theory—category, structure-preserving morphism, universal mapping property, and objective logic—in place, we proceed to study the applications of category theory to cognitive neuroscience. Beginning with the objectification of everyday observations such as ‘everybody has a father and a mother’ (Lawvere, 2002), we will work towards characterizing the essence and objective logic (Lawvere, 1994b, 2004) of perception and memory (Albright, 2015; Albright and Stoner, 1995; Hertz, Krogh and Palmer, 1991, pp. 11-24; Hopfield, 1982; Jazayeri and Movshon, 2007; Kandel et al., 2012, pp. 556-637).

References

Albright, T. D. (2012) On the perception of probable things: Neural substrates of associative memory, imagery, and perception. *Neuron* 74: 227-245.

Albright, T. D. (2015) Perceiving. *Daedalus* 144: 22-41.

Albright, T. D. and Stoner, G. R. (1995) Visual motion perception. *Proc. Natl. Acad. Sci.* 92: 2433-2440.

Ehresmann, A. C. and Vanbremeersch, J. P. (2007) *Memory Evolutive Systems: Hierarchy, Emergence, Cognition,* Amsterdam: Elsevier Science.

Helmuth, L. (2001). Moral reasoning relies on emotion. *Science* 293: 1971-1972.

Hertz, J. A., Krogh, A. S., and Palmer, R. G. (1991) *Introduction to the Theory of Neural Computation*, New York: Addison-Wesley.

Hopfield, J. J. (1982) Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci.* 79: 2554-2558.

Jazayeri, M. and Movshon, J. A. (2007) A new perceptual illusion reveals mechanisms of sensory decoding. *Nature* 466: 912-915.

Kandel, E. (2013) The new science of mind and the future of knowledge. *Neuron* 80: 546-560.

Kandel, E. R., Schwartz, J. H., Jessell, T. M., Siegelbaum, S. A. and Hudspeth, A. J. (2012) *Principles of Neural Science* (5th ed.), New York: McGraw Hill Education.

Lawvere, F. W. (1966) The category of categories as a foundation for mathematics. In *Proceedings of the conference on categorical algebra*, S. Eilenberg, D.K. Harrison, S. MacLane, and H. Röhrl (Eds.), New York: Springer-Verlag, pp. 1-20.

Lawvere, F.W. (1972) *Perugia notes: Theory of categories over a base topos*, Perugia: Universita' di Perugia.

Lawvere, F. W. (1991) Some thoughts on the future of category theory. *Lecture Notes in Mathematics* 1488: 1-13.

Lawvere, F. W. (1994a) Cohesive toposes and Cantor's 'lauter einsen'. *Philosophia Mathematica* 2: 5-15.

Lawvere, F. W. (1994b) Tools for the advancement of objective logic: Closed categories and toposes. In *The Logical Foundations of Cognition*, J. Macnamara and G.E. Reyes (Eds.), New York: Oxford University Press, pp. 43-56.

Lawvere, F. W. (2002) Kinship and mathematical categories. In *Language, Logic, and Concepts*, R.S. Jackendoff, P. Bloom and K. Wynn (Eds.), Cambridge, MA: MIT Press, pp. 411-425.

Lawvere, F. W. (2004) Functorial semantics of algebraic theories and some algebraic problems in the context of functorial semantics of algebraic theories. *Reprints in Theory and Applications of Categories* 5: 1-121.

Lawvere, F. W. (2005) An elementary theory of the category of sets (long version) with commentary. *Reprints in Theory and Applications of Categories* 11: 1-35.

Lawvere, F. W. (2013) Fifty years of functorial semantics. *Celebrating Bill Lawvere and Fifty Years of Functorial Semantics*, Union College mathematics conference lecture transcript, http://www.math.union.edu/~niefiels/13conference/Web/Slides/Fifty_Years_of_Functorial_Semantics.pdf, Cited 07 May 2015.

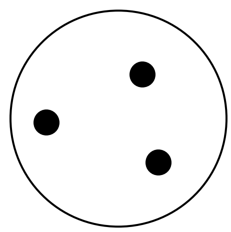
Lawvere, F. W. and Schanuel, S. H. (2009) *Conceptual Mathematics: A First Introduction to Categories* (2nd ed.), Cambridge, UK: Cambridge University Press.

Lawvere, F. W. and Rosebrugh, R. (2003) *Sets for Mathematics*, Cambridge, UK: Cambridge University Press.

A Bag of Contradictions

Let's begin with something familiar: gift-bags. Gift-bags, as we all know, come in all shapes and sizes, so our definition of gift-bags, if it were to be good enough to describe each and everything that we think of as a gift-bag, should not involve any specification of a particular shape, or size, or color, to list a few properties. What, then, should the definition of gift-bag involve? Something—some property—that's true of all gift-bags, which is nothing more than the fact that they contain gifts. To the extent that there is nothing more to a gift-bag than the gifts that it contains, we might as well identify gift-bag with the collection of gifts. How about gifts? Well, each gift is something that's distinct so that we can grab it from amongst other gifts in the collection of gifts and yet is indistinguishable from other gifts in the sense that it is just a gift—nothing more, nothing less—just like every other thing in the collection of gifts. This is all (to be qualified later on) that we need to lay the foundations of the monument called math. Set is a collection of elements and elements are distinct and yet indistinguishable—a contradiction that did not sit well with many mathematicians when Cantor put forward his Set theory. Another familiar contradiction that might help us feel at home with contradictions is boundary: A and not A; if we denote a region, say, India with A, not A would be its neighbors Pakistan, China, Nepal, Tibet, Bhutan, Bangladesh, etc. A and not A, then, is the boundary of India or the border between India and its neighbors (no wonder we have so many border disputes; it’s high time we make peace with contradictions).

Sets are often depicted as circles with dots inside denoting elements as shown below:

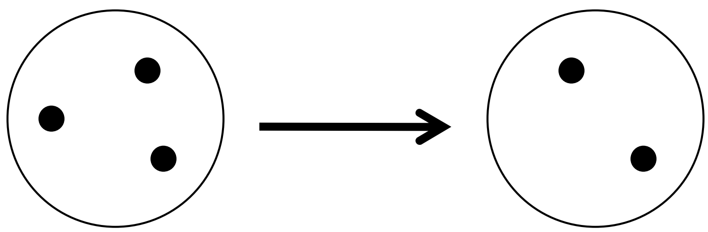


Since any collection of distinct and indistinguishable elements is a set, people in this room, to give an example, form a set. The number of people i.e. the number of elements in a set is a property of the set. So we can have sets with one element, two elements … and then there are infinite sets (e.g. collection of all numbers) and, of course, empty set.

Who got more gifts: me or you?

I want to see who got more gifts: me or my neighbor. So is with sets. Once we have the notion of set in place, we look for something to compare one set with another. To get a feel for what that something we are looking for might look like, let’s look at numbers. In the case of numbers, we can specify the relation between two numbers using, for example, ‘=’ (e.g. 3 = 3), ‘>’ (3 > 2), or ‘<’ (3

< 4). Since there is more to sets than the property of size (number of elements) that they have (just as there is more to you and me than the property of, say, height that we have), we need something more (general) than equality or 'less than' or 'greater than' to relate one set to another and that thing is function. Since the comparison is between sets, the function sits in between sets and is depicted as an arrow going from one set to another:



Denoting the sets with capital letters A and B, and the function with lower case f, we use the following notation for function

$$f: A \rightarrow B$$

There are many ways to think about function: we can think of a function as a journey from a town A to a town B noting that the town A where the journey began and the town B where the journey ended are integral to the journey just as the two endpoints of a line segment

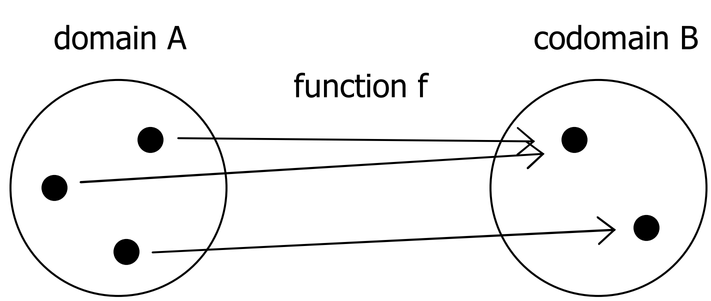
are part of the line;

we can also think of function as a process that transforms a set A into another set B, or even more concretely we can think of function as life with birth and death as endpoints (oh no, I don't want to get started on this one). The two sets need not be different just as there can be a tour of Salk which begins at our Gift shop and ends the Gift shop (how subtle of Salk Institute).

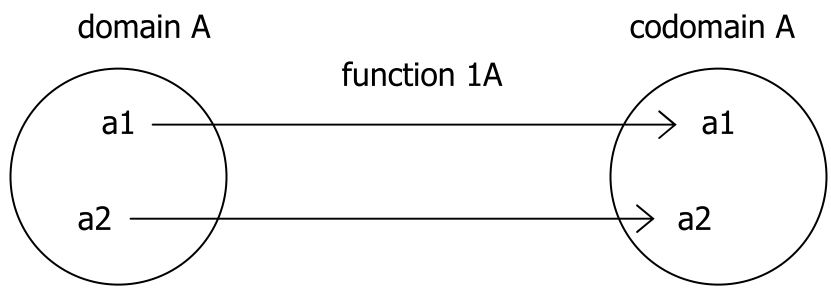
Not too long ago function was thought of as a collection or set of pairs of elements one from each of the two sets of a function. Now we think of a function f not as a set but as an arrow with a set A as source (or domain) and a set B as target (or codomain). Changing the way we think about, the way we represent, and the name we give to function not only brought about a sea change unveiling hitherto unimagined vast landscapes of mathematics, but also enabled exploration of the newfound lands. Shakespeare, had he been around to see this, would have thought twice before asserting: "What's in a name? / That which we call a rose / By any other name would smell as sweet."

Well, all this thinking, depicting, naming stuff about function is all fine and dandy, but what exactly does this function thing do (besides sitting in between two sets)?

A function f assigns to each and every element of its domain (a set A) one and only one element of its codomain (a set B). For example:

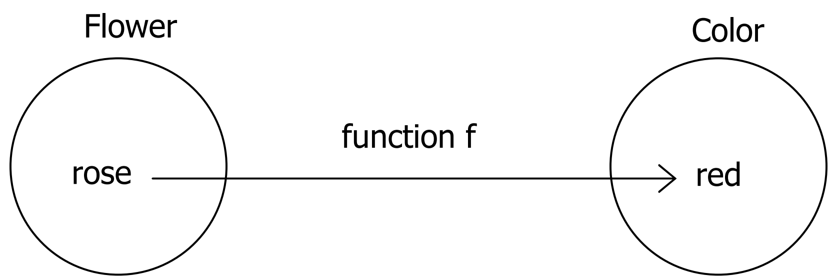


Drawing parallels to numbers, there's a function that acts like number 1 (with respect to multiplication), which is rather unimaginatively denoted by 1 and called identity function. For each set A there is an identity function 1A: A → A, which takes each element 'a' of domain A to the very same element 'a' of the codomain A i.e. 1A (a) = a for all 'a' in A. Pictorially,



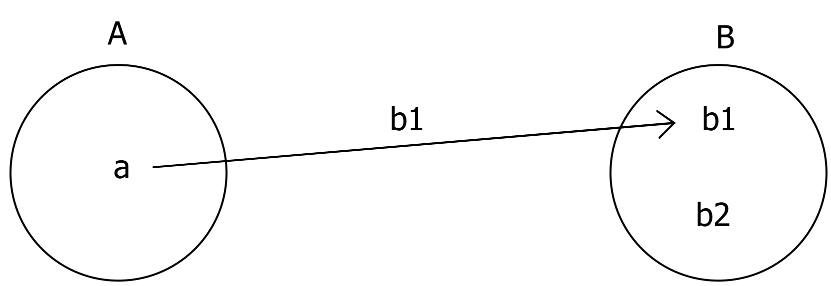
A Jest and Its Point

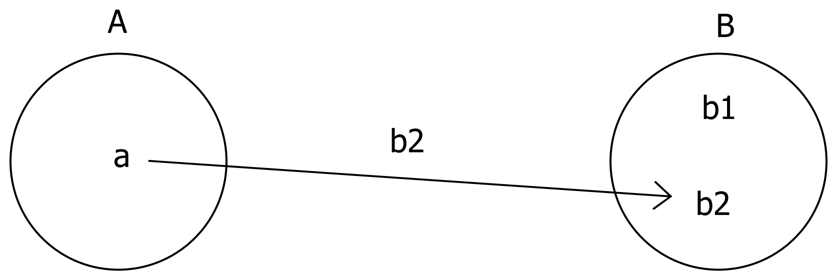
If assigning a point of codomain to every point of domain is all that a function does, then we can readily think of very many types of functions simply by changing the domain or codomain or by changing the assignment rule. Let's look at some of the various kinds of functions. Starting with a simple case of a function between two sets each of which has just one element: a set called Flower, which has only one element i.e. one flower, say, 'rose' as domain, and another set called color, which also has only one element i.e. one color, say, 'red', we find that there is just one function between two single-element or singleton sets as shown below.



If, in looking at the function f, you read rose –is-> red, then you made an observation pregnant with profound consequences, which is that the appropriate formal interpretation of everyday 'is' is function (or arrow; MacNamara, a psychologist, suggested this interpretation to Lawvere) just as the formal interpretation of everyday 'and' is multiplication (e.g. the probability of head in a coin toss is 0.5; the probability of head and head in two coin tosses is 0.5 x 0.5 = 0.25).

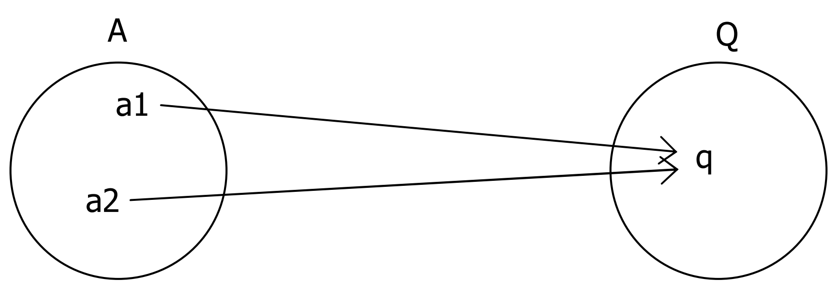
If we keep the singleton set as domain and change the codomain set, we find that a point is not a point but an arrow from a point to a point!

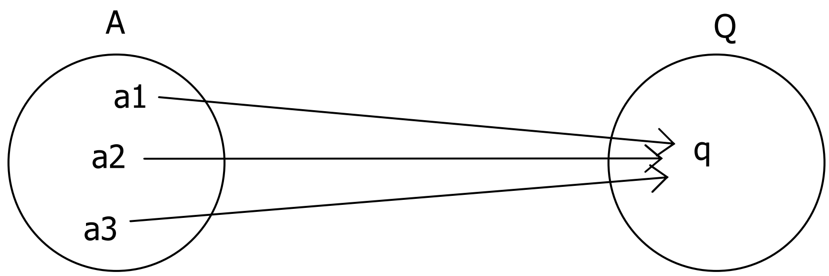




If the codomain has 2 elements b1 and b2, then there are 2 functions: one assigns b1 to the only one element 'a' of the domain, and the other assigns b2 to 'a' as shown above. In view of this one-to-one correspondence between elements of a set and functions from singleton set to the set, we can think of elements as functions, albeit special (just as we can think of a point as a line of zero length).

Now let's see what happens if we take singleton set as codomain and change the domain set as shown below:

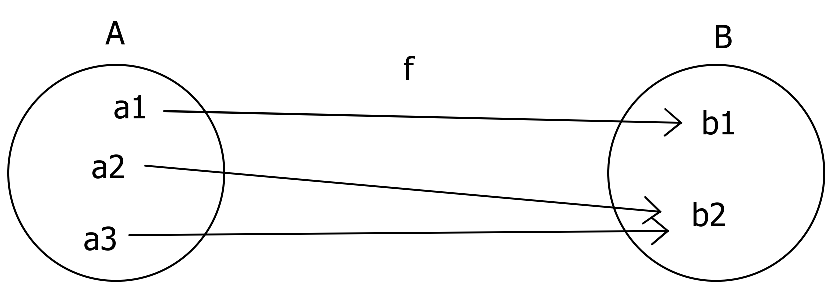


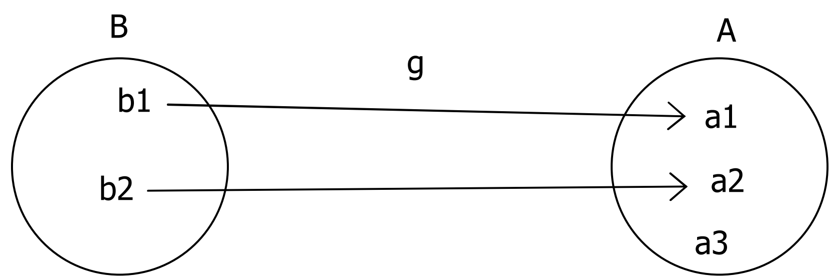


Since there is only one element in the codomain Q (see above), all of the elements of the domain set A (irrespective of the number elements it has) are assigned to that single element 'q' and this is the only function. In other words, there is only 1 function from any set to the singleton set.

Learning to Think with Arrows (Arbib & Manes)

Walking through the very many sizes of sets that could be domain and codomain and the numerous rules of assigning one element of codomain to each element of domain, we come across two types of functions that look interesting; so we look closely:





If we compare and contrast the two functions f: A → B and g: B → A i.e. stare at the above internal diagrams of the two functions long enough we slowly begin to see the differences such as: in the case of f all elements of the codomain B are assigned (to some element of the domain A), whereas in the case of the function g not all of the elements of codomain A get assigned (element a3 is left out). Here is one more difference: in the case of the function g different elements of domain B are assigned to different elements of codomain A, whereas in the case of f two different elements of domain A are both assigned to the same element i.e. f(a2) = b2 and f(a3) = b2.

Let's use these differences as definitions (if we don't like them later on we can always throw them away):

1. A function f: A → B is onto if every element b of codomain B is a f(a) for some 'a' in the domain A.

2. A function g: B → A is 1-1 if 2 elements b1 and b2 of domain B are assigned to the same element a1 of codomain A, then they are equal i.e. if g(b1) = a1 and g(b2) = a1; g(b1) = g(b2) =implies=> b1 = b2.

Going by these definitions of onto and 1-1 functions; it seems as if, there isn't much more to say beyond that which is stated in the definition. Since we are trying to go beyond mere knowing of the definitions, which we cooked up anyways, to gain some understanding of how these two different kinds of functions relate to one another (if they do), and since goggling does not seem to help anymore, let's ask couple of questions, which have answers built into their formulation (more or less).

Equality and Composition of Functions

When are two functions f: A → B and g: P → Q equal? Two functions f and g are equal i.e. f = g if they have same domain i.e. A = P, same codomain i.e. B = Q, and f(p) = g(p) for all of the elements p in the set P (equivalently, f(a) = g(a) for all a in A).

Since we can think of a function f: A → B as a journey from a town A to a town B, let's look at journeys to get a feel for composition of functions. Consider a journey, say, from San Diego to Amsterdam and from Amsterdam to Hyderabad. Now let's throw in some colons and arrows to make it look formal.

Journey1: San Diego → Amsterdam

Journey2: Amsterdam → Hyderabad

We can combine these two journeys to get a composite journey called

Journey = Journey2 Journey1: San Diego → Hyderabad

(read Journey 2 after Journey1)

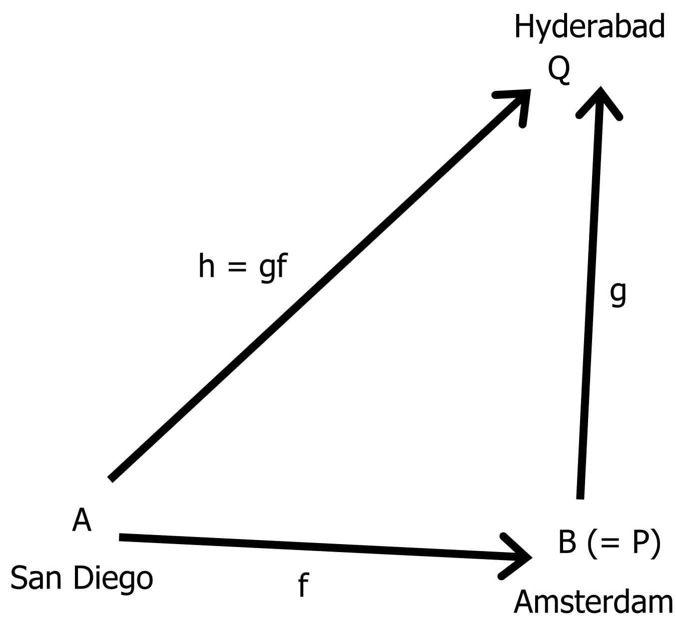
In other words, if the end of a journey is the beginning of another journey, then we can combine the two journeys to get a composite journey beginning at the beginning of the first journey and ending at the end of second journey (ah, what a joy it is to state the obvious).

Drawing parallels, given two functions

$f: A \to B$

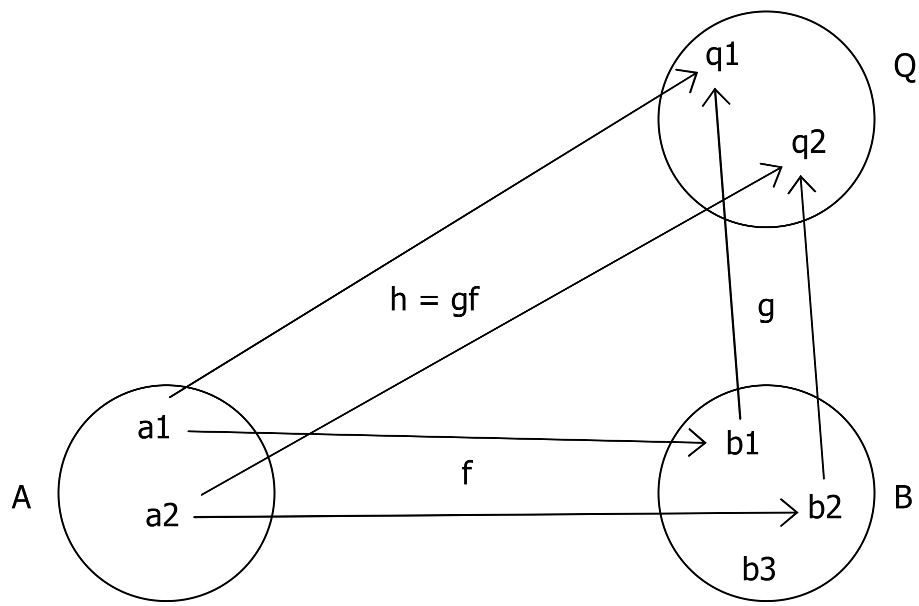
$g: P \to Q$

the composite $h = gf: A \to Q$ is defined if and only if the domain of g is same as the codomain of f i.e. $P = B$. (Within the monastic simplicity of composition there's an unstudied beauty: put together if and only if they fit-together, which may one day whisper in some lucky dude's ear how the symbolic concepts and the spatial intuition are put together into the conscious experience that simply refuses to speak, and it is of no consolation to see mathematicians struggling since the time of Descartes' analytical geometry to define the composition of as is with as if.) Back to terrestrial business: diagrammatically,



Let’s calculate the composite h: A → Q of two functions f: A → B and g: P → Q,

taking P = B and A, B, Q, f and g as shown below:



Expressing the above diagram in equations:

h(a1) = (gf)(a1) = g(f(a1)) = g(b1) = q1

h(a2) = (gf)(a2) = g(f(a2)) = g(b2) = q2

Identity and Associative Laws

The promised analogy between multiplication of numbers and composition of functions is here at last. Analogous to 1 x a = a for all a (number), we have the following composites with identity functions. Given any function f: A → B two identity functions corresponding to the two sets A and B can be constructed.

1A: A → A, 1A(a) = a

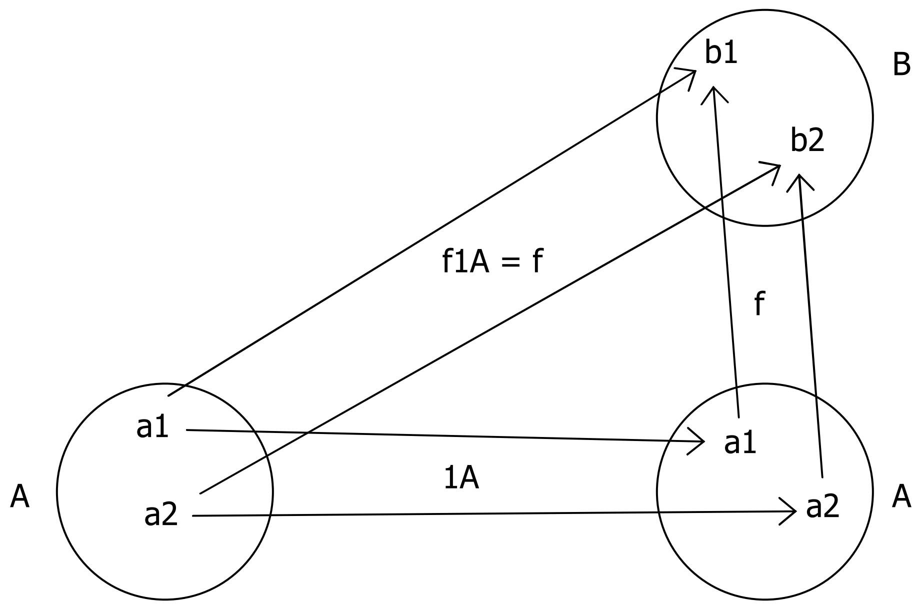
1B: B → B, 1B(b) = b

Since the domain of f is same as the codomain of 1A we can compose them to get the composite

f1A: A → A → B

f1A = f

as depicted below:

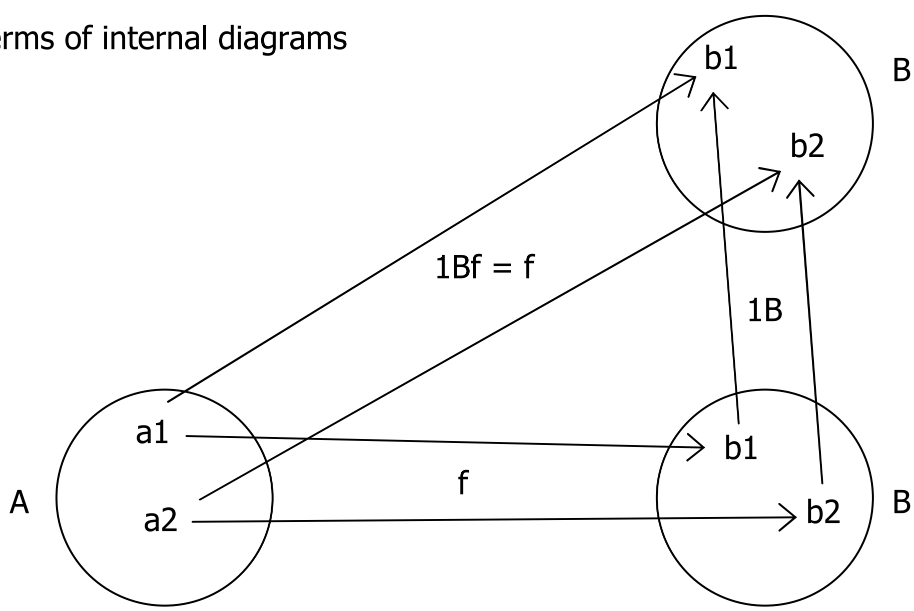


Since the domain of 1B is the same as the codomain of f, we can compose them to get the composite

1Bf: A → B → B

1Bf = f

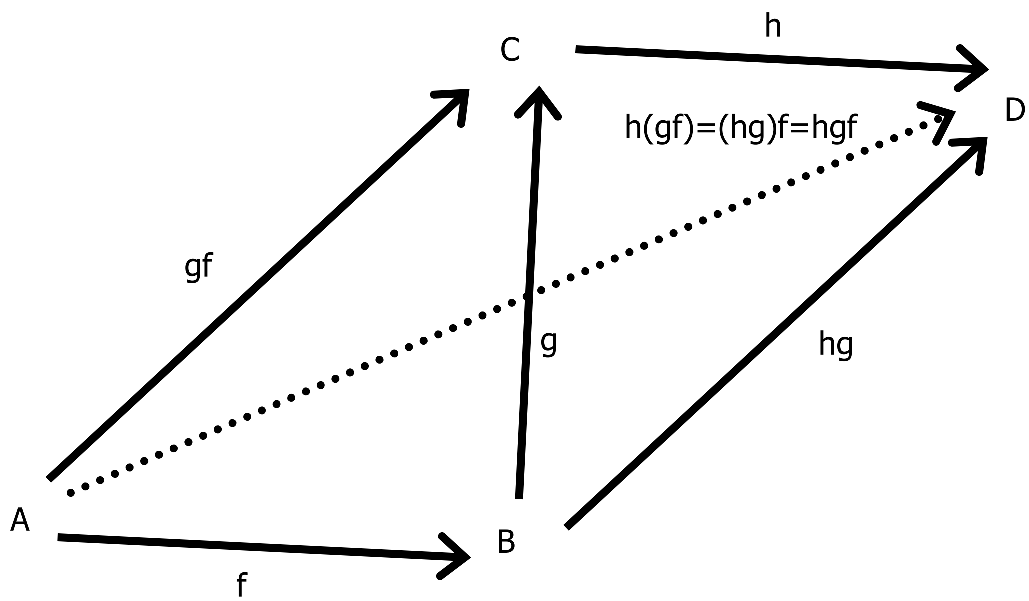
In terms of internal diagrams



Moving along, given three functions

f: A → B, g: B → C, and h: C → D

we can compose the three functions in two different ways



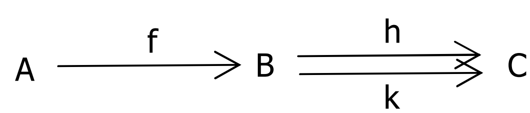
First combine g and f to get the composite gf and then combine gf and h to get h(gf). Or, first calculate the composite of h and g to get hg and then calculate the composite of hg and f to get (hg)f. Are these two composites equal? One could readily think of scenarios wherein h(gf) ≠ (hg)f. For example, if f, g, and h are life forms, mating f and g to get offspring gf which could then be mated with h to get h(gf), which need not necessarily be same (genetically) as the offspring obtained by mating f with the offspring of h and g. In other words, genetic algebra is non-associative since it doesn't satisfy the associative law:

h(gf) = (hg)f = hgf

In the case of sets and functions, associative law holds true as depicted in the above commutative diagram.

Opposites: Onto and 1-1 Functions

The opposition between onto and 1-1 functions is not apparent in our earlier definitions. To see the duality, let's consider composites of functions with certain specified functions. Let f: A → B be onto and furthermore consider two functions h, k as depicted below:



and satisfying

$$h(f(a)) = k(f(a))$$

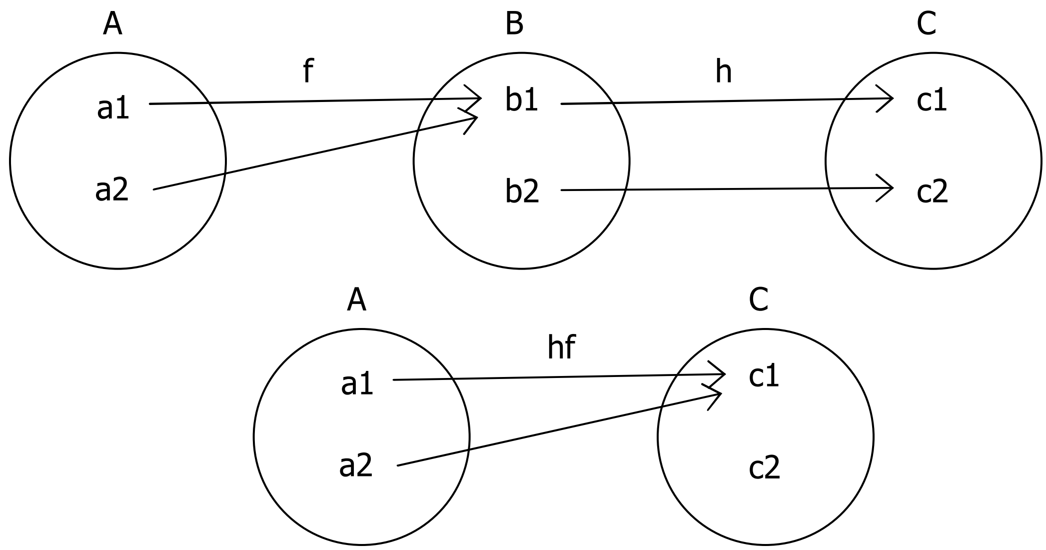
for all a in A. Since f is onto B = f(A)

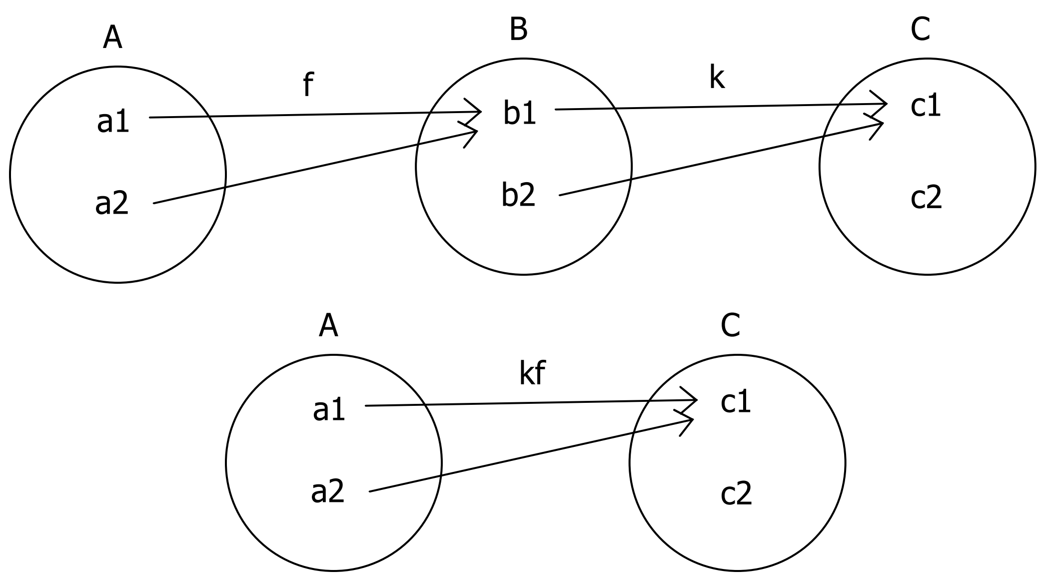
$$h(b) = k(b)$$

Since the functions h and k have same domain and same codomain and agree on all values b of domain B

$$h = k$$

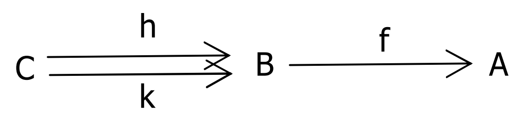
We can show the converse i.e. if f is not onto, then two functions h, k can be constructed such that hf = kf but h ≠ k as follows:





Therefore f is onto if hf = kf =implies=> h = k.

Now turning to 1-1 function: let f: B → A be 1-1 function and furthermore consider two functions h and k as shown below:



satisfying

$$f(h(c)) = f(k(c))$$

given that f is 1-1

$$f(x) = f(y) \text{ =implies=> } x = y$$

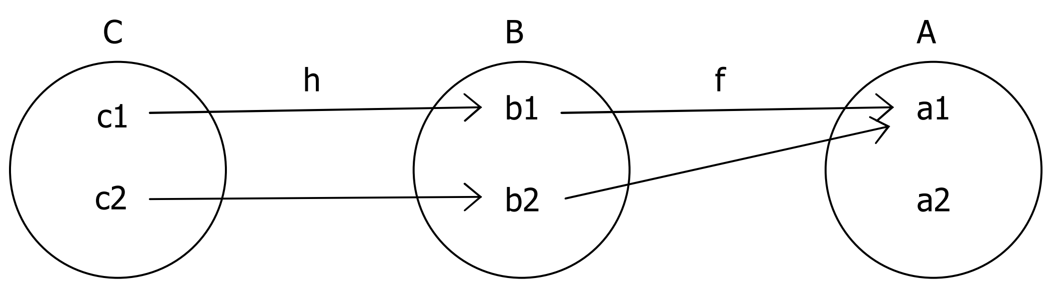
therefore

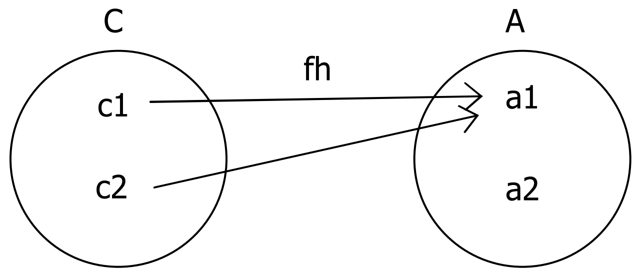
$$f(h(c)) = f(k(c)) \text{ =implies=> } h(c) = k(c)$$

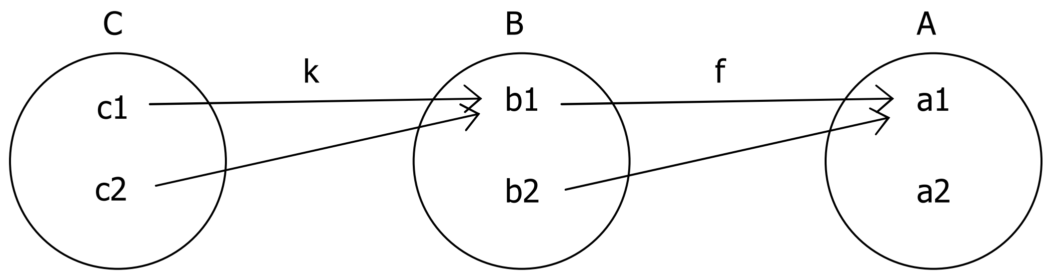
since the two functions h and k have same domain, codomain, and agree on all values c of domain C, we have

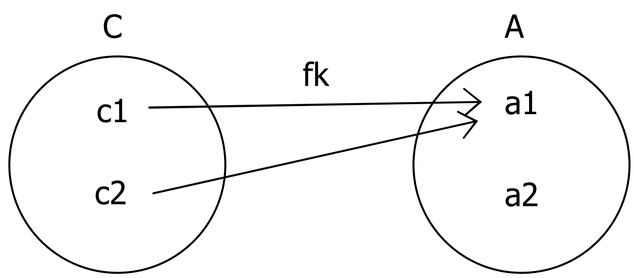
$$h = k$$

Now let's show the converse i.e. if f is not 1-1, then two functions h, k can be constructed such that $fh = fk$ but $h \neq k$ as follows:





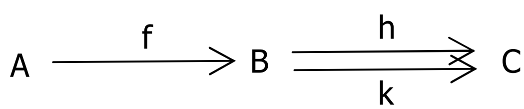




Therefore f is 1-1 if fh = fk =implies=> h =k.

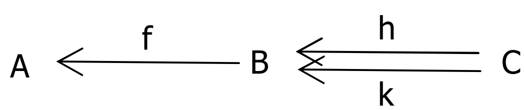
To see the opposition between onto and 1-1 functions

f is onto if



implies h = k.

f is 1-1 if



implies h = k.

follow the arrows.

An Introduction to Function

Abstract

Guided by the pedagogical principle of introducing unfamiliar notions in terms of familiar concepts, the notion of FUNCTION is presented as an abstraction of the concept of NUMBER, or more explicitly as number without the ORDER intrinsic to numbers. Order can then be introduced into the concept of function in two ways; one of which counts the number of functions between sets, while the other classifies the functions.

Thanks to our love of money we all know all that we need to know about numbers; our understanding of numbers is of an abstract nature in the following sense. Grab a person on the street and give that person two numbers and that person can readily tell you which of the two numbers is bigger, or if they are both equal; what’s fascinating is that the person can tell you the relation (‘=’ or ‘<’ or ‘>’) between the given two numbers even if the person has never seen or heard or imagined or thought of those numbers. How does the person-on-the-street do it? Since the familiarity of the particular numbers doesn’t seem to figure in the exercise, there must be something about the numbers that the person-on-the-street has familiarity with and it’s that property of numbers that the person-on-the-street has a deep understanding of, and which the person uses to tell whether the given two numbers are equal or which one of them is smaller. Well, it's ORDER. The numbers are ordered: 1, 2, 3… Where do numbers get this property of order from? In other words, is there something beyond the superficial order that we readily recognize in numbers. Yes, there is, and for this we have to go to the corner store cash register.

Before we go to the corner store which involves considerable work, let's sit down and look at a couple of numbers or better yet let's look at a number, say, '10'. As we are about to utter the number in words, we realize we need to know whether it's in binary number system or decimal system or, simply put, 'what number system is '10' in? If binary, then '10' means two; while in decimal system, '10' means ten. In other words, we got numbers and numbers got place-value systems. Let's get little bit more explicit about place-value system. Let's consider a number, say, 295 and let's say it's in decimal number system. So saying I have 295 dollars means, in decimal number system, that you have five 1's, nine 10's, and two 100's. Now this looks like a cash register with a slot (I tried to ask a person behind the counter what exactly it's called the other day by way of leaning over the counter and extended my hand close to the register; needless to add it didn't go well) for 1's and a slot for 10's etc. Now we can vaguely see another order under the surfacial order of the numbers, both of which will be made explicit below.

First, there's the order of numbers: 1, 2, 3 … and then there is the ordering of the places in place-value number system: 1, 10, 100 … (in decimal system). Let's depict a number, say, 295, in the cash-register format:

100	10	1
2	9	5

The ‘1’, ‘10’, ‘100’ above the line can be thought of as names for three different places and the corresponding numbers below the line can be thought of as the values of the places above. Alternatively, one can think of ‘1’ above the line as a bin of size 1; the ‘5’ below the line as the number of things of size 1. Put succinctly, both the ‘names’ of the places and the ‘values’ of the places are ordered.

Now let’s consider another number, say, 255 and put in it the cash-register format:

100	10	1
2	5	5

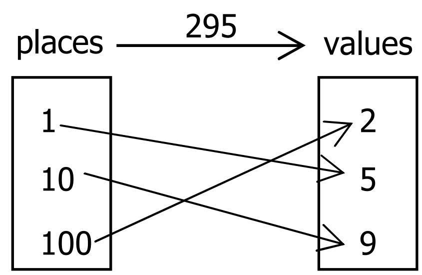
One thing we can readily see is that two distinct places (‘1’ and ‘10’) can both have the same value (‘5’). Let’s make a note of it and move on to another number, say 55, which when put in the above format:

100	10	1
0	5	5

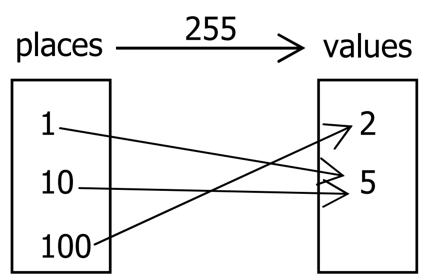
Now we see that each and every one of the places must take a value (if the 100's slot in the cash register is empty, we say we got 0 hundreds). While we are at the cash-register we might as well note that any slot in the register can have only one number of whatever it has; for example, you can't have both two '10' dollar notes and seven '10' dollar notes in the '10' dollar slot. Put differently, each place can take one and only one value (the total number of notes in any slot can be, for instance, five, but never be both five and six).

Now let's do a [partial or intermediary] summing up: (i). we have a collection of PLACES and a collection of VALUES, (ii). both collections are ordered, (iii) each and every place in the collection of places has one and only one value from amongst the collection of values, and (iv) two different places can have the same value, but no place can have two different values.

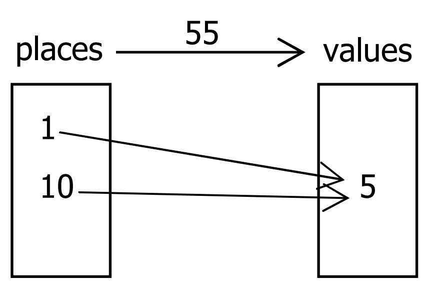
With these observations in place, let's look at the numbers we looked at earlier in a different format. Let's start with 295, which can be decomposed into a collection of places P = {1, 10, 100} and a collection of values V = {2, 5, 9}, which you can readily see that the members of each one of the collections can be ordered (in the above listing we ordered them in increasing order). Now let's place the collection of places on the left and the collection of values on the right with an arrow beginning at a place and ending at its corresponding value (which is simply a change of orientation from places on top and the corresponding values at the bottom to left → right orientation as depicted below):



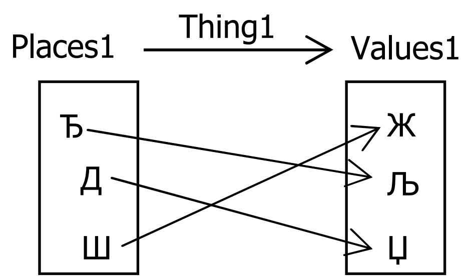
Now let's depict another number 255 in the above 'number: places → values' format as follows:

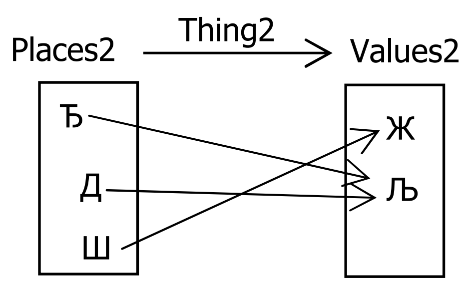


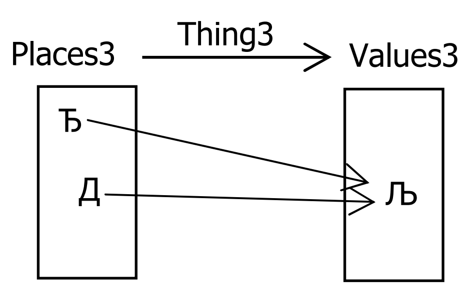
While we are at it, we might as well depict the number 55 in the above format:



Now let's abstract ORDER away from NUMBER i.e. replace places and values with things which have no order relationship between them. For example, let places be P = {Ђ, Д, Ш} and values be V = {Ж, Љ, Џ}, from which i.e. from the collections of places and the collections of values it is clear that we got rid of the order that's intrinsic to the numbers. Having gotten rid of order, we want to make sure that we got saved everything else about the number in our new construct that we got whatever that might be. To get to see what this new construct i.e. what 'NUMBER – ORDER' is, let's depict the above 'number: places → values' depiction of numbers with unordered symbols as follows:







The above diagrams look like the internal diagrams of functions f: A → B. Does that mean 'NUMBER – ORDER = FUNCTION'? If we look at the definition of function f: A → B, where f assigns to each element (place) of the domain A (places) an element (value) of the codomain B (values), we find that function can indeed be thought of as an abstraction of number or number-sans-order.

Summing up, numbers and functions are not much different. However the value of thinking of FUNCTION as 'NUMBER – ORDER' is something that I don't feel qualified to evaluate. I'd appreciate very much any comments, critique, clarifications, and corrections that you may have. In a subsequent note we will see how we can introduce ORDER back into FUNCTION in two different ways: one of which counts the functions and the other classifies the functions.

A Study of Function

Recently I came across a paper, according to which retrieving the material that's been read once is equivalent to re-reading the material 5 times or so. It seemed to make some sense in the sense retrieval inevitably highlights what I forgot, which in turn forces me to formulate questions such as 'what's that condition that has to be satisfied in order for the composite of two functions to be defined?', which in turn focuses my attention and help structure and glue the material to be learned into a coherent and cohesive unity.

Be that as it may, I thought of retrieving what I have been discussing for the past few weeks. Here I go. We have been talking a lot about functions such as

$$f: A \to B$$

also depicted as

$$A \xrightarrow{f} B$$

where A is the domain set and B is the codomain set. Even though 'A' and 'B' are depicted as disconnected from the arrow representing function, they i.e. domain A and codomain B are integral to the function f just as the end-points of a line-segment are integral to the line-segment. Yet another useful metaphor to keep in mind when thinking about functions is to think of a function as a journey 'j' with domain and codomain of the function corresponding to beginning (e.g. La Jolla) and destination (e.g. Amsterdam) of the journey,

$$j\text{: La Jolla} \rightarrow \text{Amsterdam}$$

also depicted as

$$\text{La Jolla} \xrightarrow{\quad j \quad} \text{Amsterdam}$$

We also noted that domain and codomain sets, and sets in general can be identified with identity functions such as $1_A: A \rightarrow A$, which when translated to our line-segment metaphor says that the end-points of a line-segment can be thought of as line-segments of zero length. In terms of our journey metaphor, the beginning and destination can be thought of as journeys that go nowhere (or stay wherever they are; $1_{\text{La Jolla}}\text{: La Jolla} \rightarrow \text{La Jolla}$). When we put down or formalized our thought of thinking of a set as an identity function, we found ourselves on the one hand simplifying the conceptual repertoire needed to speak of functions; on being able to speak of functions in terms of functions alone, albeit special functions, in the sense we can now say that a function has an identify function as domain and an identity function as codomain as in,

$$f: 1_A \rightarrow 1_B$$

and on the other hand confronting a problem as depicted below:

$$A \xrightarrow{\quad 1_A \quad} A \xrightarrow{\quad f \quad} B \xrightarrow{\quad 1_B \quad} B$$

Looking at the above diagram, in an effort to make sense of it, one immediate question we had was 'how do we put-together or compose two functions?' Here again we found that our journey metaphor is instructive. To elaborate, consider two journeys

$$j: \text{La Jolla} \rightarrow \text{Amsterdam}$$

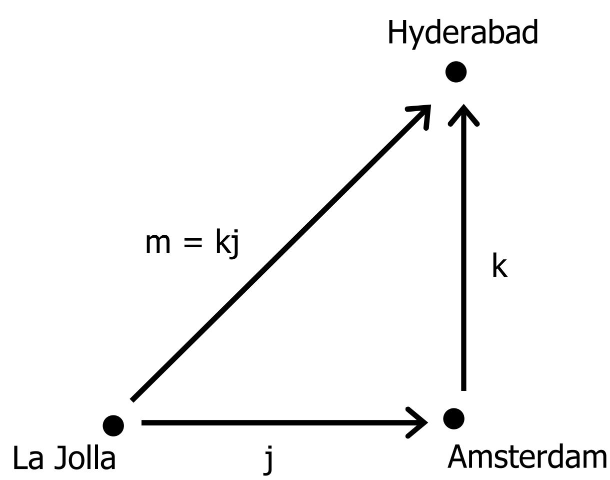
and

$$k: \text{Amsterdam} \rightarrow \text{Hyderabad}$$

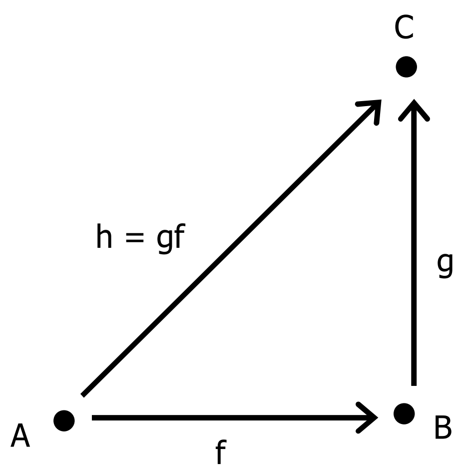
the composite kj (read as journey k after journey j) is, taking the most obvious take on journeys, the journey from La Jolla to Hyderabad. We also noted that the composite journey

$$kj: \text{La Jolla} \rightarrow \text{Hyderabad}$$

of two journeys such as j and k is possible if and only if the destination of the first journey j, Amsterdam, is the same as the beginning of the second journey k, Amsterdam. Pictorially we can depict as follows:



Finally we noted that taking the journey j from La Jolla to Amsterdam and journey k from Amsterdam to Hyderabad is same as taking the composite journey m from La Jolla to Hyderabad. Now let's translate these everyday intuitions into the terminology of functions. Drawing on the above metaphor, we say that the composite of two functions f: A → B and g: C → D is defined if and only if the codomain set of the first function is same as the domain of the second function i.e. B = C, and that the domain of the composite is same as the domain of the first function and the codomain of the composite is same as the codomain of the second function. More explicitly the composite of f: A → B and g: B → C is gf: A → C.



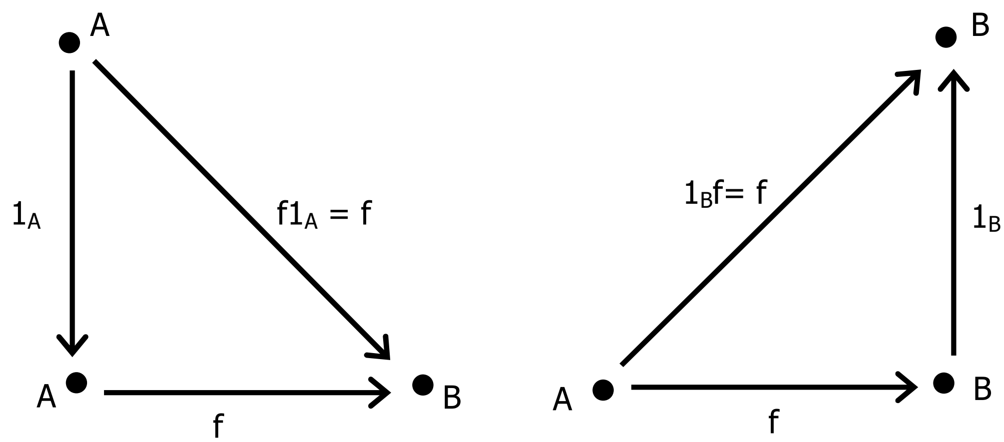
Most importantly, the composite function h is equal to the 'function g after function f'. Now we find ourselves ready to answer the question raised by our representation of function f: A → B

$$A \xrightarrow{f} B$$

as

$$A \xrightarrow{1_A} A \xrightarrow{f} B \xrightarrow{1_B} B$$

under the pretext of terminological austerity. We calculated the composites $f1_A$: A → B and 1_Bf: A → B and found that $f1_A = f$ and $1_Bf = f$ as depicted below:



Now given that $f1_A$: A → B and 1_Bf: A → B are defined, we found that the composite 1_Bf1_A: A → B can be defined. Given that the following two pair-wise composites

$$A \xrightarrow{1_A} A \xrightarrow{f} B \qquad A \xrightarrow{f} B \xrightarrow{1_B} B$$

are defined, it is easy to see that the composite of all three functions is defined by way of imagining f: A → B segments of the above two pair-wise composites overlap (which is somewhat analogous to the condition that codomain of the first function f must coincide with the domain of the second function g in order for the composite gf to be defined) so that we get

$$A \xrightarrow{1_A} A \xrightarrow{f} B \xrightarrow{1_B} B$$

We can also be more specific and state that the composite $1_B f 1_A$ can be evaluated either by first evaluating $f1_A$, which is f which when composed with 1_B gives f as the composite, which is exactly what we get when we first evaluate $1_B f$ and then compose the composite f with 1_A. Or even more explicitly the composite $1_B f 1_A$ can be calculated either as a composite of

$f1_A: A \rightarrow B$ and $1_B: B \rightarrow B$

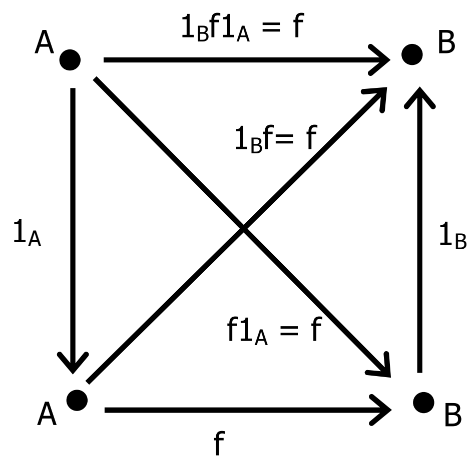
or as a composite of

$1_A: A \rightarrow A$ and $1_B f: A \rightarrow B$

and both ways of calculating $1_B f 1_A$ give the same result i.e.

$$1_B(f1_A) = (1_B f)1_A = 1_B f 1_A = f$$

as shown below:



Generalizing from identity functions to functions in general, we note that whenever two composites gf and hg are defined, then the composite hgf is defined, which can be thought of as a generalization of given 'B = C' the composite gf: A → C of functions f: A → B and g: C → D is defined, and can be calculated as the composite of gf and h i.e. h(gf) or as the composite of f and hg i.e. (hg)f is as illustrated below in terms of our favorite journeys.

j: La Jolla → Amsterdam

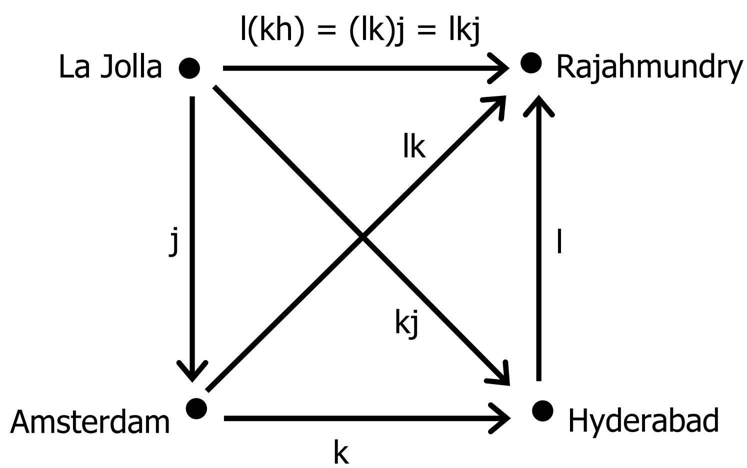
and

k: Amsterdam → Hyderabad

and

l: Hyderabad → Rajahmundry

Since the journey l's beginning is Hyderabad, which is the same as the destination of journey k, whose beginning is Amsterdam, which is the same as the destination of journey j, we can clearly form pair-wise composites (lk)j and l(kj) to obtain lkj, with, of course, (lk)j = l(kj) = lkj, when the l, k, and j are interpreted as functions.



Now let's collate our recollections of the properties of function—properties that are true of all functions—each and every function.

1. Function $f: A \rightarrow B$ has a domain A and codomain B, which are identity functions $1_A: A \rightarrow A$ and $1_B: B \rightarrow B$, respectively
2. Given two functions $f: A \rightarrow B$ and $g: C \rightarrow D$ composition of f and g is defined if B = C and the composite $h: A \rightarrow C$ is given by $h = gf$
3. Composite of a function $f: A \rightarrow B$ with identities $1_A: A \rightarrow A$ and $1_B: B \rightarrow B$ satisfies: $f1_A = f = 1_Bf$

4. Given three functions f: A → B, g: C → D, and h: E → F, the triple composite hgf: A → F is defined if the pair-wise composites gf: A → D and hg: C → F are defined, or in other words hgf is defined if B = C and D = E and is given as h(gf) = (hg)f = hgf

Now let's give a name to the collection of the above list of properties; since they are dealing with functions and only functions, let's call the structure formed of this list a category of functions (in naming 'category of functions' instead of 'category of sets', I seem to think I am following Ehresmann's naming convention, which is more revealing of the category we are dealing with). If we replace function with arrow to denote anything that satisfies the above 4 conditions, we have an arbitrary category or a generic category. With little rewriting we have the general notion of CATEGORY defined:

1. Arrow f: A → B has a domain A and a codomain B, which are identity arrows 1_A: A → A and 1_B: B → B, respectively
2. Given two arrows f: A → B and g: C → D composition of f and g is defined if B = C and the composite h: A → C is given by h = gf
3. Composite of an arrow f: A → B with identities 1_A: A → A and 1_B: B → B satisfies: $f1_A = f = 1_Bf$
4. Given three arrows f: A → B, g: C → D, and h: E → F, the triple composite hgf: A → F is defined if the pair-wise composites gf: A → D and hg: C → F are defined, or in other words hgf is defined if B = C and D = E and is given as h(gf) = (hg)f = hgf

In passing we may note that, with isomorphisms, a subset of arbitrary functions, as arrows we obtain the notion of groupoid, and with automorphisms, a subset of isomorphisms, as arrows we obtain the notion of group.

It might be helpful to state what we mean by a CATEGORY in plain English. A CATEGORY, in plain English, is a mathematical universe or a domain of mathematical discourse. For example, the category of functions that we were talking about in this session is the mathematical universe inhabited by sets, functions, and composition of functions. Alternatively, the category of functions is a mathematical discourse about sets, functions, and composite of functions. In a sense the mathematical notion of CATEGORY is not inconsistent with its everyday usage.

In the spirit of complete disclosure, since I am not so sure about the legitimacy of the way we arrived at the notion of CATEGORY as a collection of properties of functions, I'll go over the textbook definition of CATEGORY, which on the surface does not seem to be much different, but may differ in matters that matter most.

Before we close let's look at a concrete illustration of the notion of CATEGORY, especially one in which arrows are not functions (Arbib & Manes). Before we get to the category, we need to have a definition in place.

A poset (or partially ordered set) is a set A with a structure of ≥, which is

Reflexive: $a \geq a$ for all a in A

Antisymmetric: $a \geq a'$ and $a' \geq a \Rightarrow a = a'$ for all a, a' in A

Transitive: a ≥ a′ and a′ ≥ a″ => a ≥ a″ for all a, a′, a″ in A

Consider the set A = {1, 2, 3, 4} along with the structure '≥', so that we have as arrows 2 ≥ 1, 3 ≥ 2, etc., where 1, 2, 3, and 4 are considered objects or identity arrows. The identity arrows such as 4 ≥ 4 are given by the reflexivity of the structure of ≥. The composite of two composable arrows 3 ≥ 2 and 2 ≥ 1 is 3 ≥ 1 by virtue of transitivity of ≥, and is in accord with the definition of composition of arrows of a category. We can also note that the composite of an arrow with its identities is the arrow as in the composite of 3 ≥ 3 and 3 ≥ 2 is 3 ≥ 2, and the composite of 3 ≥ 2 and 2 ≥ 2 is 3 ≥ 2. Having checked the identity laws, let's check to see if associativity holds true. The composite of three composable arrows: 4 ≥ 3, 3 ≥ 2, and 2 ≥ 1 can be evaluated by first evaluating the composite of 3 ≥ 2, and 2 ≥ 1, which is 3 ≥ 1, and then evaluating the composite of 4 ≥ 3 and 3 ≥ 1, which is 4 ≥ 1. Alternatively, we could first evaluate the composite of 4 ≥ 3 and 3 ≥ 2, which is 4 ≥ 2, and then evaluate the composite of 4 ≥ 2 and 2 ≥ 1, which is 4 ≥ 1; thereby upholding associativity. Thus we have a category (a poset) in which a ≥ b is an arrow (and not a function) and a ≥ a is the identity arrow on a in A. This example clearly shows that arrows of category need domain and codomain, which could be identities, and as long as there is composition of arrows defined satisfying identity and associative laws, we have a category.

COMPOSITION

Algebraic Composition Defined under Geometric Condition (Brown)

Let's start with something more mundane: 'what's it about 'coffee' and about 'cup' that allows one to 'add' coffee to cup?'

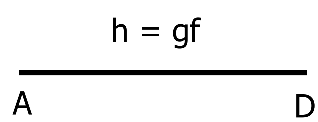
Before we answer the above question, let's go back to the notion of 'algebraic composition defined under geometric condition.' To make this notion little bit more concrete, consider 2 geometric objects, say, 2 line segments f and g, depicted as below:

There are many things about the above geometric objects that one could talk about, but there are two things about each of the two geometric objects f and g that are of particular relevance for our present purposes; they are the endpoints of each of the two line segments f and g. Let's denote the endpoints of f as 'A' and 'B', and the endpoints of g as 'C' and 'D' as shown below:

Now let's go back to where it all started i.e. composition. How do we compose, or add, or put-together the two objects f and g, or more explicitly the two line segments f and g? We can put f and g together by bringing them close to one another such that the end-point B of f coincides with the end-point C of g. In other words, we can put-together f and g if and only if B = C. The composite of f and g is a line-segment gf with A and D as end-points.

Composition of 

is defined if and only if B = C, and the composite is



The most important thing to note is that when we want to compose two things, the two things to be composed must have something in common to form the composite.

Now let's go back to coffee and cup and try to answer how we get to add coffee to cup using the condition for composition: something common! Since we all add coffee to cup all the time, there must be something in common between coffee and cup, which is volume. But for the fact that both coffee and cup have the common property of volume, we wouldn't be able to add coffee to cup.

Functions, Rules, and Equality

Now let's go back (I hope all this going back is not too uncomfortable) to functions. Consider 2 functions f: A → B and g: C → D. Now let's find out under what conditions or what are the conditions that the above two functions 'f' and 'g' have to satisfy in order for us to form the composite of the two functions. With the geometric conditions fresh in mind, we could take a guess at it, but before we do that, let's get very clear about what a function is. The notation for a function is

$$f: A \rightarrow B$$

The function

$$f: A \rightarrow B$$

has 3 things:

1. a domain object 'A', a set
2. a codomain object 'B', a set
3. a rule

Let's dig little bit deeper into the 'rule' that's part of the definition of function.

rule:

for each element 'a' of set 'A'

there is exactly one element 'b' of set 'B'

such that 'b' is the value of the function 'f' at 'a'

$$f(a) = b$$

Any 'f' that satisfies the above property (labeled 'rule') is considered a function from a set 'A' to a set 'B'.

Before we get back to composition, there is one more thing that we need to get clear about function, which also happens to be about the 'rule'. Recall that a function has 3 things:

$f: A \rightarrow B$

1. a domain set A
2. a codomain set B
3. a rule (which assigns an element of the codomain to every element of the domain)

From this definition of function, there is much we can say; to begin:

Two functions 'f' and 'g' are different if:

1. they have different domains; e.g. $f: A \rightarrow B$, $g: C \rightarrow B$
2. they have different codomains; e.g. $f: A \rightarrow B$, $g: A \rightarrow C$
3. they have different domains & codomains; e.g. $f: A \rightarrow B$, $g: C \rightarrow D$

How about rules? Are two functions different if they have different rules? For this we go back to the definition of 'equality of functions'. Two functions 'f' and 'g' are equal if they have the same domain, the same codomain, and if and only if they have the same value for each and every argument. Given $f: A \rightarrow B$ and $g: A \rightarrow B$, function $f = g$ if and only if $f(a) = g(a)$ for all 'a' in the domain A. With this definition of equality at the front of our minds, let's try to answer the question: 'are two functions different if they have different rules?'

Let's consider two rules, wait, before we do that let's consider two functions with same domain and same codomain so that the only thing different is the rule corresponding to the function, so that we can clearly see the relation between 'rule' and 'function' in terms of 'same' and 'different'.

Consider two functions f: N → N and g: N → N, where N is the set of numbers; N = {1, 2 …}. Now let's consider two different rules, one for 'f' and another for 'g'. For 'f', let the rule be 'take the input 'n' and add 1 i.e. (n + 1), and keep it aside for now, take the input and subtract 1 i.e. (n – 1); now multiply them both i.e. (n + 1)(n – 1); in terms of equations:

$$f(n) = (n + 1)(n - 1)$$

For 'g', let the rule be 'take the input n, square it i.e. (n^2), and subtract 1 i.e. (n^2 - 1); in the format of equations:

$$g(n) = (n^2 - 1)$$

Now if we look at the two rules for 'f' and 'g', they are different: simplistically speaking, one (f) requires the operations of addition, subtraction, and multiplication, while the other (g) requires just multiplication and subtraction. Clearly we have two different rules for f: N → N and g: N → N, but going by the definition of equality of functions,

$$f = g \text{ if and only if } \quad f(n) = g(n)$$

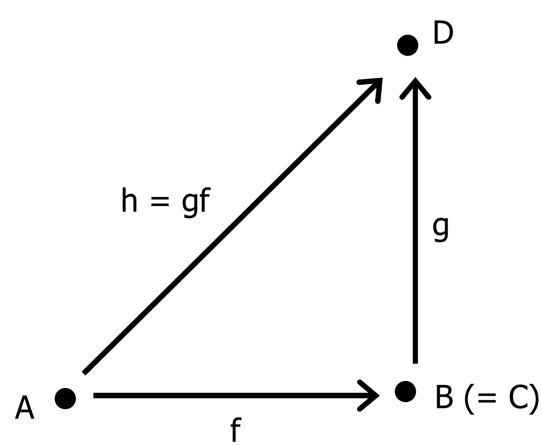
From the above

$$f(n) = (n + 1)(n - 1) = (n^2 - 1) = g(n)$$

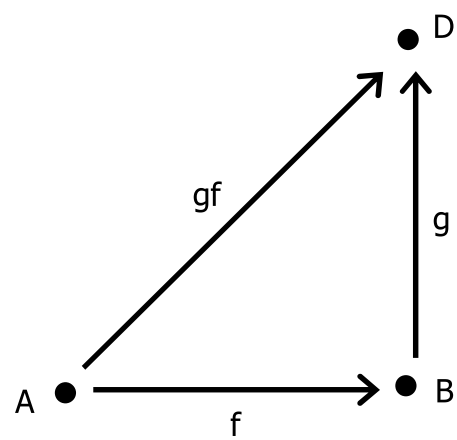
So, we find that even though the functions 'f' and 'g' have different rules, they satisfy the conditions for equality of functions. Thus we note that functions with different rules (but not different domains, or different codomains, or both i.e. different domains and different codomains) can be equal.

Composition and Associative Law

Now with the 'equality of functions' in place, let's return to 'composition of functions.' Consider two functions f: A → B and g: C → D. Taking a cue from 'algebraic composition defined under geometric conditions', we say the composite of two functions f: A → B and g: C → D is defined if and only if the codomain of the first function f is same as the domain of the second function g i.e. B = C. Once this condition in terms of domain and codomain of functions is satisfied so as to make the functions composable, we are now in a position to find the composite of 'f' and 'g'.



The composite of f: A → B and g: B → D written as 'gf' (and read 'g after f') has as domain the domain of 'f' i.e. 'A', and as codomain the codomain of 'g' i.e. ‘D’. Now looking at the above diagram we notice that there are two pathways to go from ‘A’ to ‘D’: 1. first go from ‘A’ to ‘B’ and then from ‘B’ to ‘D’; and 2. go from ‘A’ to ‘D’ directly. Looking back at the diagram, redrawn below



Now, first we can find the value of ‘f’ at, say, ‘a’ of ‘A’; f(a) = b, and then evaluate the function ‘g’ at ‘b’ i.e. g(b) = d or expanding we have

g(b) = d

g(f(a)) = d

since element ‘a’ of ‘A’ is a function from singleton set to ‘A’

g(fa) = d

Now let’s look at the second route gf: A → D. Evaluating (gf) at ‘a’, we get (gf)a = d. In order for the two paths to lead to the same destination, they have to evaluate to the same‘d’ of D for the same ‘a’ of A i.e.

g(fa) = d and (gf)a = d or

g(fa) = (gf)a

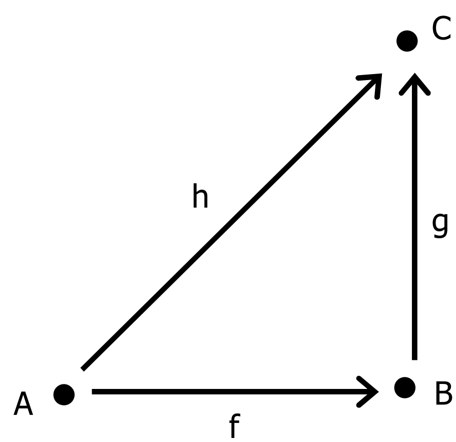
i.e. the above expression or identity or law has to be satisfied.

g(fa) = (gf)a = gfa

is nothing but the associative law. So, we may say composition of two functions is a special case of associative law.

Commutative Diagram

No, we are not quite done with our diagram (external diagram; since we are not looking at the innards i.e. elements of objects A, B …):



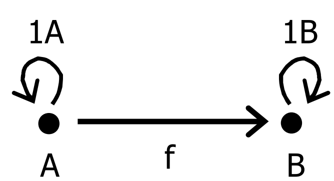
The above diagram, in which there are two paths from a point 'A' to another point 'C' and in which taking either one of the two paths is same i.e. taking 'gf' from A to B to C is same as (or equal to) taking the other path 'h' from A to C, is called commutative diagram. Making it crisp, if, in the above diagram, h = gf, then the diagram is said to be commutative.

Identity Laws

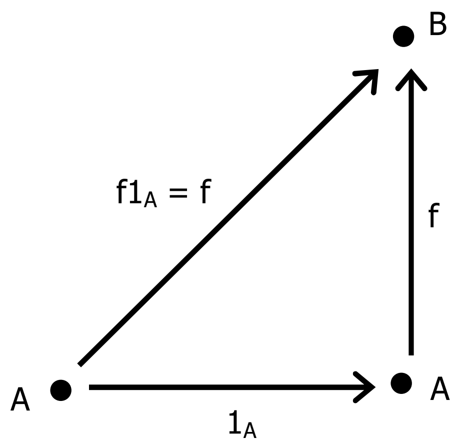
Given any function f: A → B we can readily construct two commutative diagrams along the following lines: First note that to each set A we can associate a function called identity function, which has the set A as both domain and codomain, and which assigns to each element 'a' of the domain A the same element 'a' of codomain A. Thus given a function f: A → B, we can construct two functions 1A: A → A and 1B: B → B.

Given the three functions f: A → B, 1A: A → A, and 1B: B → B, how many composites can we form?

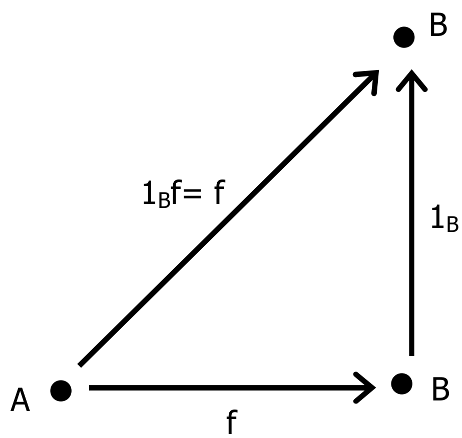
Recollect that in order for a composite of 2 functions to be defined the codomain of the first function must be same as the domain of the second function. Looking back at the above 3 functions, we note that (i) the codomain of 1A is same as the domain of f, so we can form the composite f1A of 1A and f; and (ii) the codomain of f is same as the domain of 1B, so we can form the composite 1Bf of 1B and f. In terms of external diagrams:



or more explicitly



and

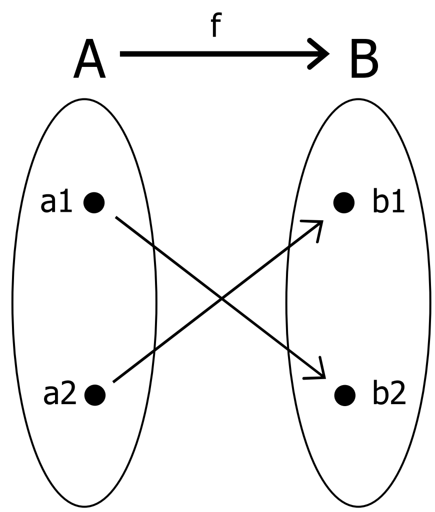


Now let's look at a simple illustration of the commutativity of the above two diagrams i.e. let's see an example of the following two identities:

(i) f1A = f

(ii) 1Bf = f

Let's look at the internal diagram of a simple function f: A → B

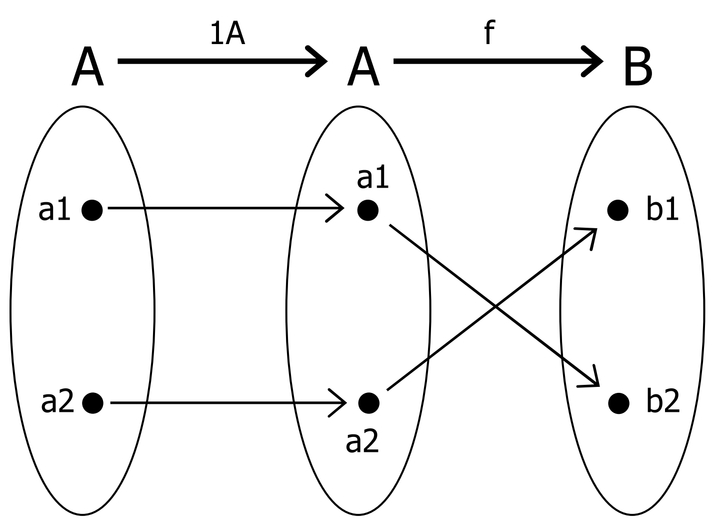


f(a1) = b2

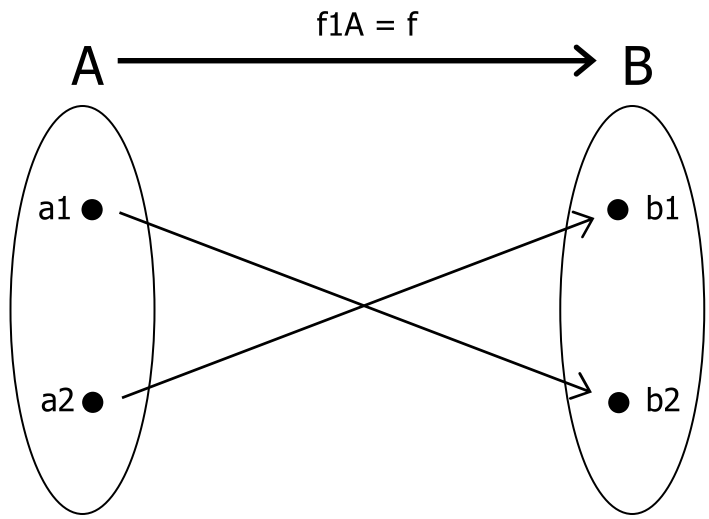
f(a2) = b1

To see that f1A = f

The composite of

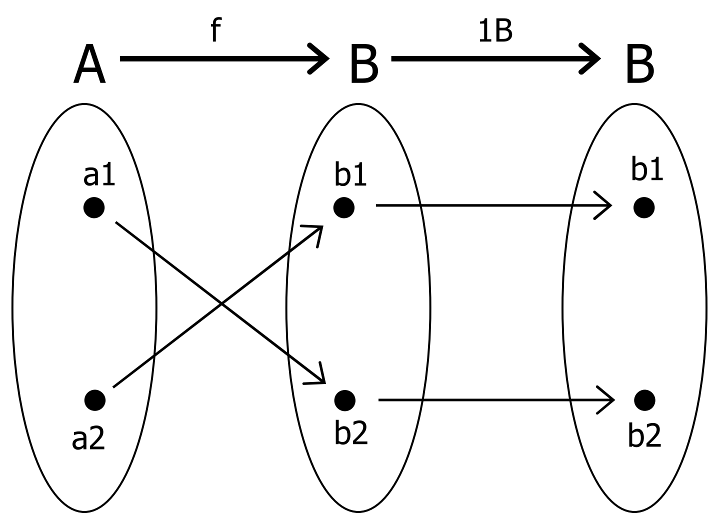


is

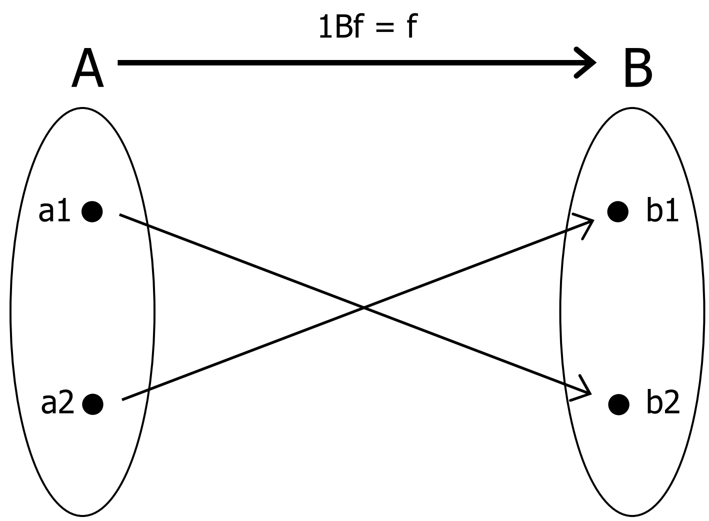


To see that 1Bf = f

The composite of



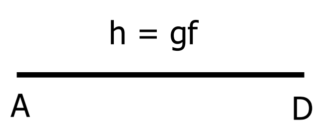
is



Composing Squares

Early on we began by noting 'algebraic composition defined under geometric conditions', which led to the condition that to compose 2 line segments, the target of the 1st line segment must be the same as the source of the 2nd line segment. Composition of

is defined if and only if B = C, and the composite is given by

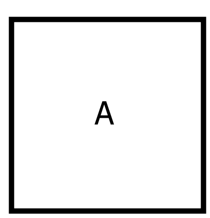


Now let's see if we can stretch the geometric intuition into additional algebraic concepts. Since we started with line-segments, which are 1-dimensional and found that the composition of 1-dimensional geometric line-segments or algebraic functions is not always defined; it is defined if and only if the 0-dimensional source of the 2nd line-segment (function) is the (0-dimensional) target of the 1st line-segment (function). So far so good or so well.

Now that we are at 1-dimensional line-land, we can go either 0-dimensional point-land or 2-dimensional flat-land (c.f. Abbot), and ask the same question of composition or putting together.

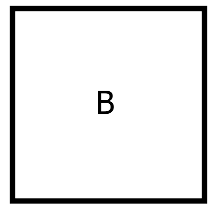
First, let's look at the case of 0-dimensional points. Composition or putting-together of structure-less points or elements, unlike the case of functions, where composition is not always defined for a pair of arbitrary functions, is always defined, which is nothing but the collection of elements into a set. In other words, in the case of 0-dimensional points or elements "composition" of elements into collections or sets is always defined.

Now, let's go in the other direction: from 1-dimensional line-segments to 2-dimensional squares. Consider a square A

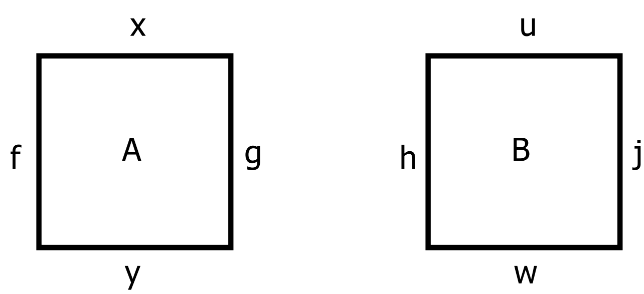


and note that the square A has (i) horizontal edges, and (ii) vertical edges.

Now consider another square B



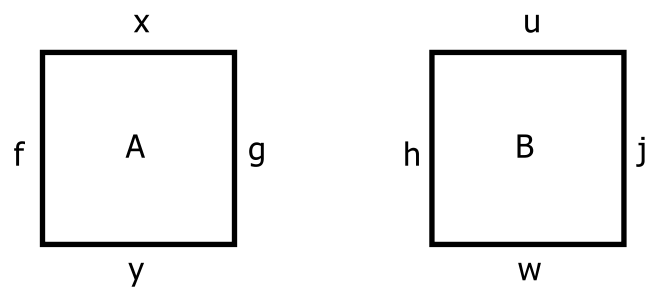
Now back to our good-old question of composition. Given 2 squares A and B, how and under what conditions can we compose them and what are the composites? Consider the two squares:



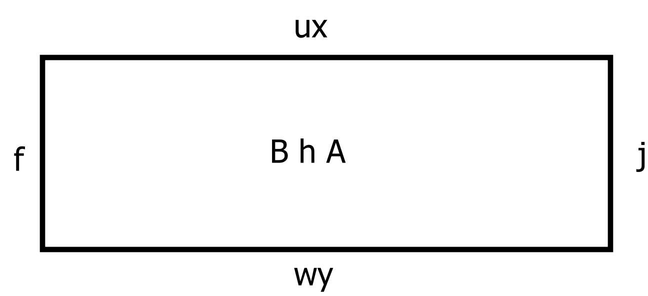
The square A has as vertical edges f and g, and as horizontal edges x and y. The square B has as vertical edges h and j, and as horizontal edges u and w.

Taking a cue or extending the reasoning employed in defining composite in 1-dimensional line-segments case, we try to find the conditions under which composition of squares is defined. Those of you who played Lego might have the answers. In any case we find that in the case of 2-dimensional squares, given two squares we can form (i) a composite by stacking the squares vertically or (ii) a composite by stacking the squares horizontally. Let's now get specific or concrete.

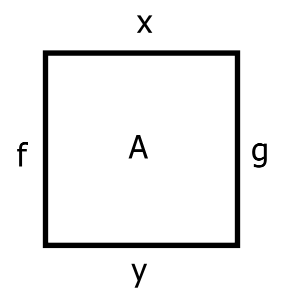
Given 2 squares A and B

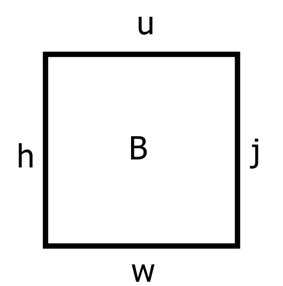


the horizontal composite B h A is defined if and only if the vertical edges g and h coincide, and the composite is:

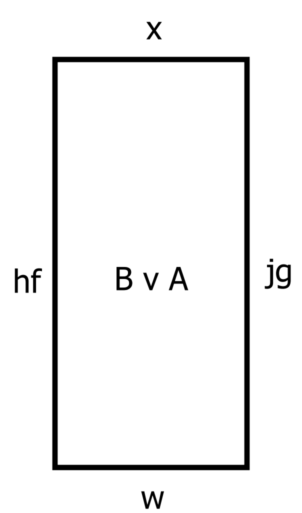


Similarly, given two squares

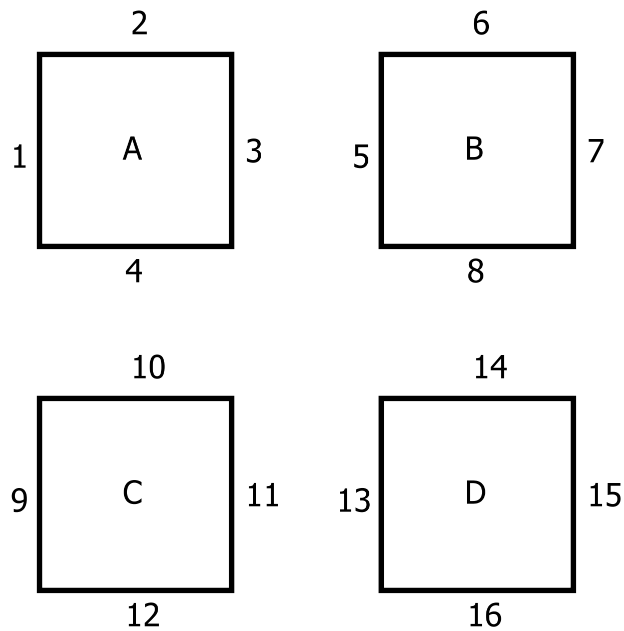




the vertical composite of A and B, B v A is defined if and only if the horizontal edge y of A coincides with the horizontal edge u of B, and the composite B v A is:

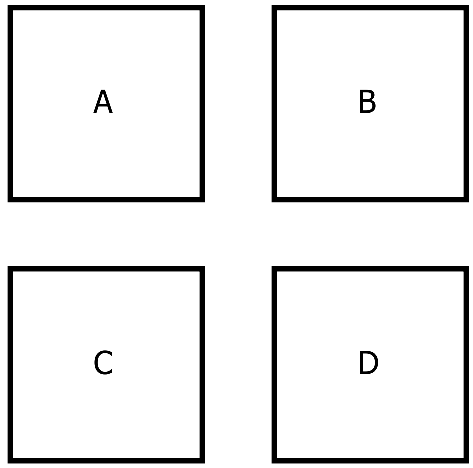


Now let's spice up squares a bit. Consider 4 squares A, B, C, and D, and the two operations v and h (vertical and horizontal composition, respectively):

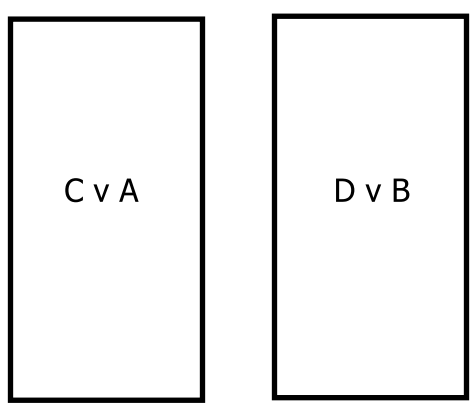


The squares are labeled with capital letters as usual, and the edges are labeled with numbers (used simply as distinct symbols). Looking at the 4 squares above we can think of forming the composite of 4 squares in 2 ways: (i) forming 2 vertical composites first and then forming the horizontal composite of the 2 vertical composites; and (ii) forming 2 horizontal composites first and then forming the vertical composite of the 2 horizontal composites.

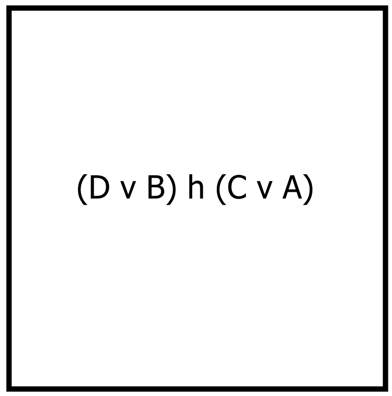
Pictorially, we can, given



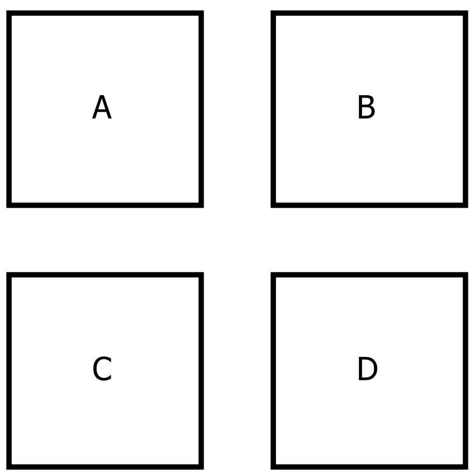
(i) first form 2 vertical composites (here and later on we assume that the conditions for vertical and horizontal compositions i.e. coincidence of the appropriate edges is satisfied or given):



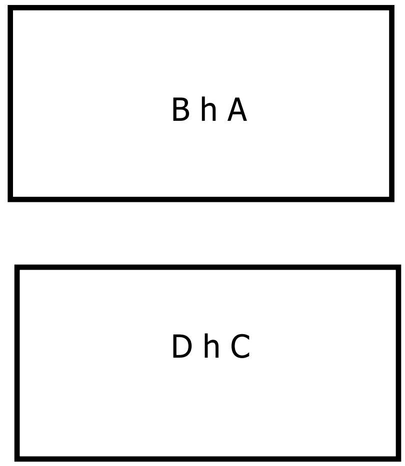
and then form the horizontal composite of the above 2 vertical composites:



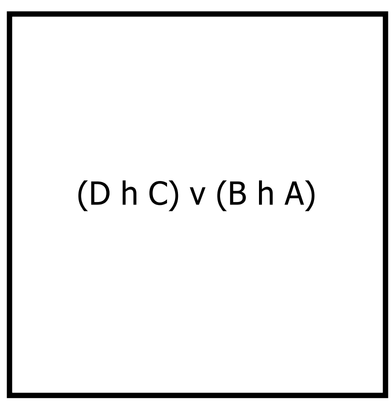
Redrawing the 4 squares again:



(ii) first form 2 horizontal composites (remember the conditions for compositions are assumed to be given or satisfied):



and then form the vertical composite of the above 2 horizontal composites:

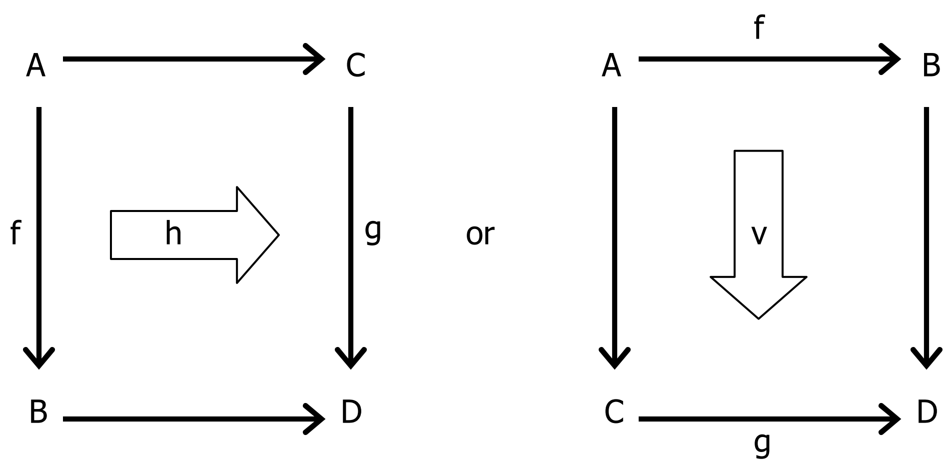


Now to the obvious question: Given that, given 4 squares, (i) we can first form vertical composites and then form their horizontal composite; or (ii) we can first form horizontal composites and then form their vertical composite, are the two composites equal? In other words, does the following identity

(D v B) h (C v A) = (D h C) v (B h A)

hold true? Well, if it holds true, we say it—the interchange law—holds true; if not, we say the interchange law doesn't hold true in the given case.

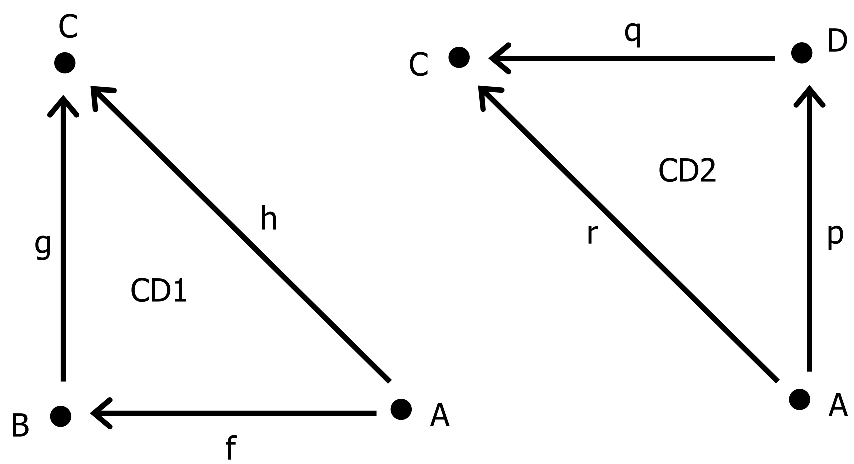
Finally all this geometry, i.e. squares, edges, horizontal, and vertical composition is but to get to the notion of a function which takes functions to functions. Our familiar function takes elements (of a domain set) to elements (of a codomain set). If we think of a function f: A → B as a process that transforms an object A into an object B, it's not much of a leap of imagination to think of a process which transforms a process f: A → B into a process g: C → D as shown below:



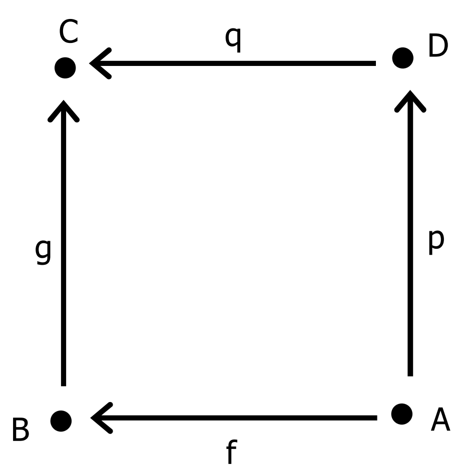
These have the usual identities, associative laws, compositions, etc., but that's all for later.

Composing Commutative Diagrams

Before we close, let's go back to commutative diagrams. Consider 2 commutative diagrams CD1 and CD2:



with h = gf and r = qp. We can form the composite of the 2 commutative diagrams if h: A → C is equal to or coincides with r: A → C. Let's say it does i.e. h = r, so that we can form



and we note that there are 2 paths from A to C: 1. from A to B to C and 2. from A to D to C. The first path is the composite ‘gf’ which is equal to ‘h’ which in turn is equal to ‘r’ which in a final turn is equal to ‘qp’. Written crisply,

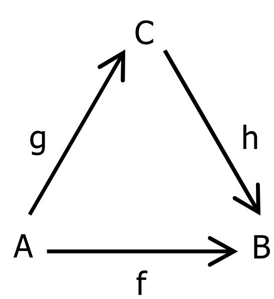
$$\begin{aligned} gf &= h \\ &= r \\ &= qp \end{aligned}$$

Thus we find that a big diagram formed of smaller commutative diagrams is also commutative, or the composite of commutative diagrams is commutative. In closing, if it seems as though the presentation is too pedantic, it’s simply to, in the words of Lawvere, “discern the germ of nontrivial in the trivial.”

A Tale of Three Maps

Let's start with something elementary. If we are given '2 x a = 10', we can calculate the value of 'a', and it is 'a = 5'. If we are given 'a x 2 = 10', we get the same value for 'a', which is 'a = 5'. First thing I'd like to note is that given two knowns (2, 10) and one unknown (a), we can find the unknown using the knowns. Second thing to note is that it doesn't matter whether the unknown 'a' is on the left (a x 2 = 10) or on the right (2 x a = 10), we get the same answer: a = 5.

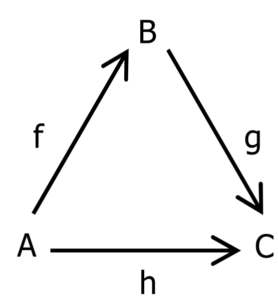
Drawing parallels to functions, the equation '2 x a = 10' is suggestive of the equational presentation (hg = f) of the commutative triangle depicted below:



commutes.

In terms of equations we have hg = f. For the rest of this note we'll be concerned with finding the unknown map, when two maps in the equation hg = f are given. The unknown map could be 'h' or 'g'. However, unlike the case of numbers, in the case of functions, the fact that the composite 'hg' is defined doesn't in any way indicate that 'gh' is also defined, and needless to add given ‘hg = f’, we, in general, get different answers depending on whether the map on the left i.e. ‘h’ is unknown or whether the map on the right i.e. ‘g’ is unknown. So we’ll look at the 2 cases in parallel. Instead of treating the most general case wherein all three maps or at least both of the known maps are arbitrary maps, we begin with a very special map i.e. identity map, and work our way to more general maps. To keep matters manageable we take one of the two known maps as arbitrary map and vary the other known map from the special case of identity map to relatively more and more general maps in the following sequence: identity, involution, automorphism, isomorphism, and section/retract.

Consider the following commutative diagram

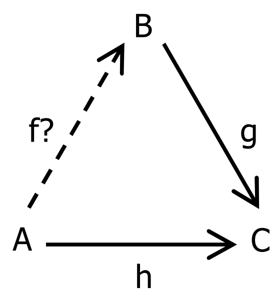


Given that the above triangle commutes, we have

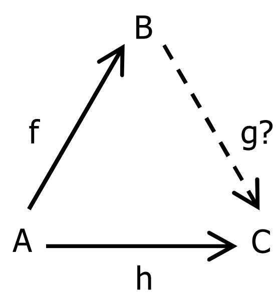
gf = h

Now of the three maps, let’s say, either ‘f’ can be the unknown map or ‘g’ can be the unknown map. Let’s denote unknown maps with dashed arrows as shown below:

Case R: Map ‘f’ in ‘gf = h’ is unknown (unknown map ‘f’ is on the right in ‘gf’; hence the label Case R)

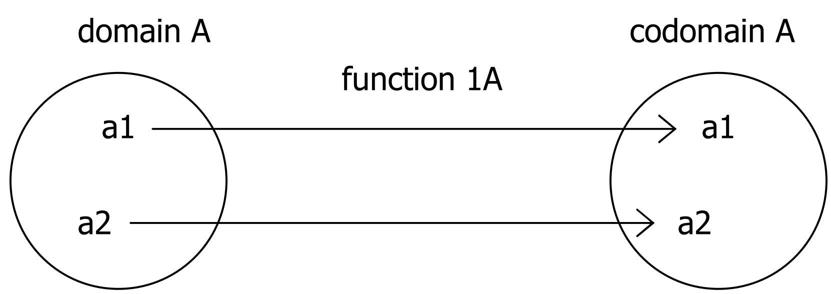


Case L: Map ‘g’ in ‘gf = h’ is unknown (unknown map ‘g’ is on the left in ‘gf’; hence the label Case L)



CASE 1. Identity Map

An identity function 1A: A → A takes each element 'a' of the domain A to the very same element 'a' of the codomain A i.e. 1A (a) = a for all 'a' in A. The identity map is pictorially depicted below:

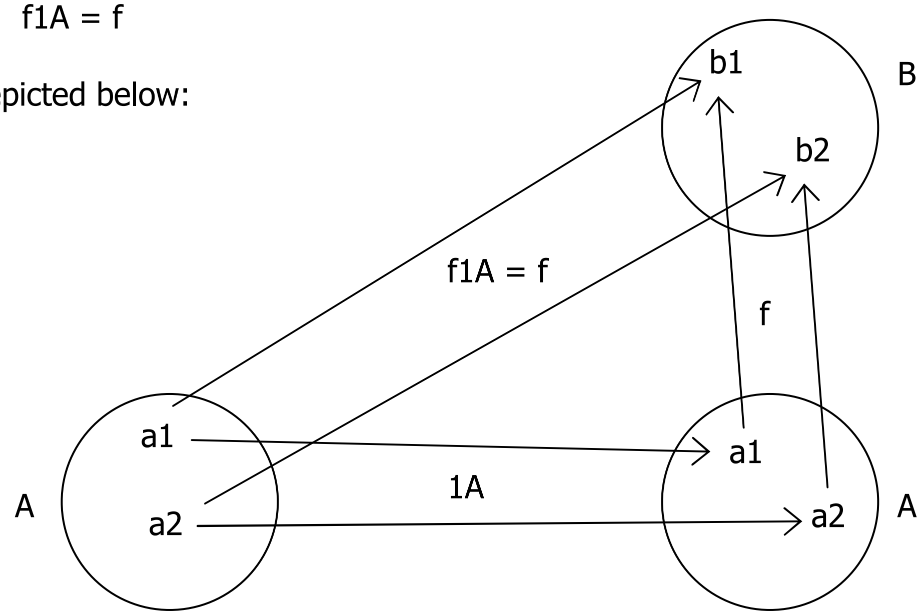


Furthermore analogous to 1 x a = a for all a (number), we have the following

f1A: A → A → B

f1A = f

as depicted below:

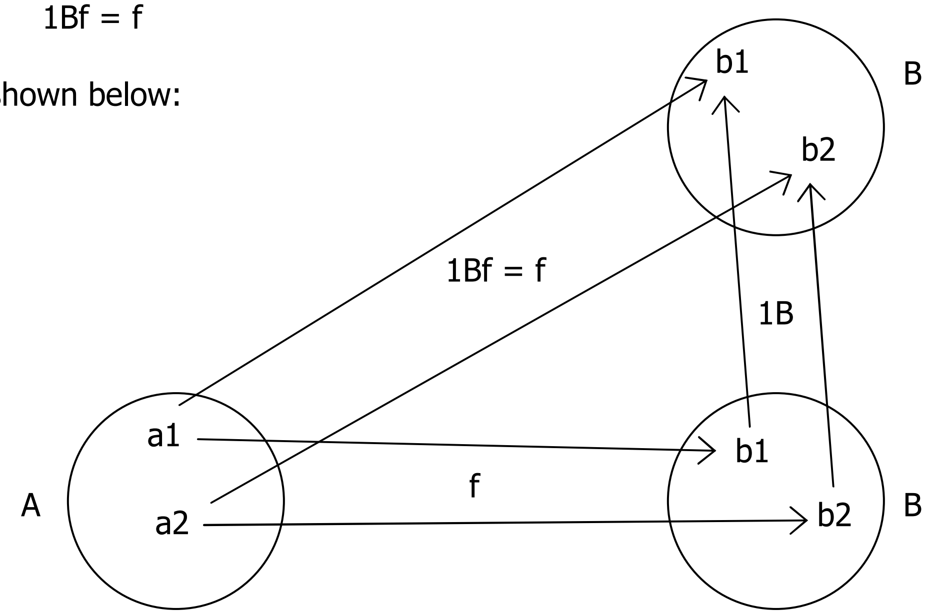


Analogous to a x 1 = a, we have

1Bf: A → B → B

1Bf = f

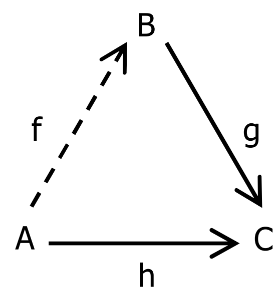
as shown below:



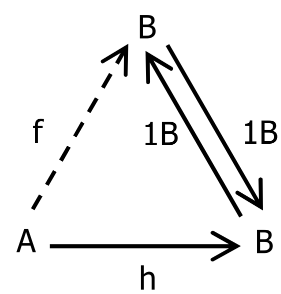
Equally important to note is the fact that 1A1A = 1A. With the definition of identity function and identity laws in place, let's consider case R first and then we can go to case L.

<u>CASE 1R.</u>

Given that the following triangle



commutes i.e. gf = h, we have to find 'f' satisfying gf = h. Given that 'g' is identity i.e. g = 1B: B → B, there's going to be the same identity arrow in the opposite direction as shown below:



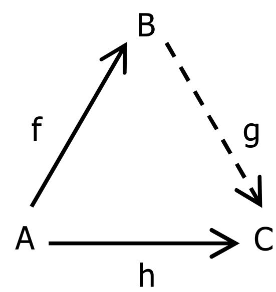
Given that the domain and the codomain of f are the same as that of the composite 1Bh, let's try f = 1Bh and see if it satisfies gf = h.

gf = g1Bh = 1B1Bh = 1Bh = h

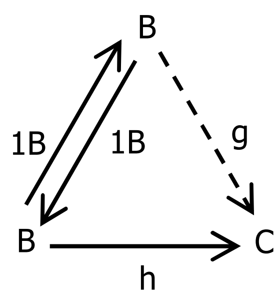
Thus f = h if g is identity.

CASE 1L.

Given that the following triangle



commutes i.e. gf = h, we have to find 'g' satisfying gf = h. Given that 'f' is identity i.e. f = 1B: B → B, there's going to be the same identity arrow in the opposite direction as shown below:



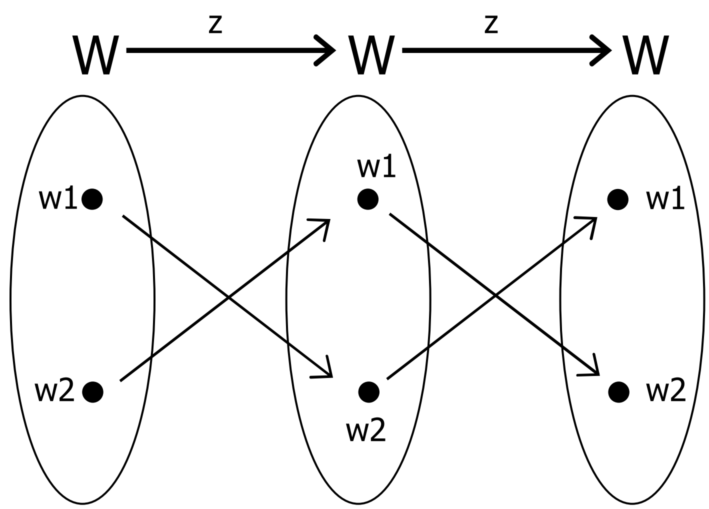
Given that the domain and the codomain of g are the same as that of the composite h1B, let's try g = h1B and see if it satisfies gf = h.

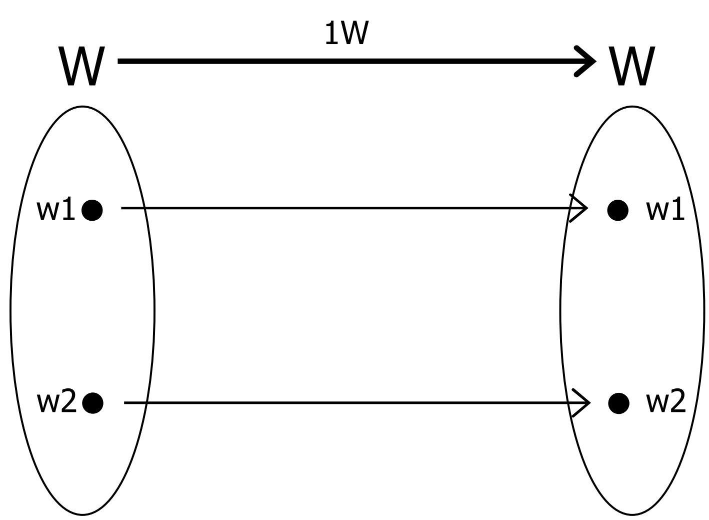
gf = h1B1B = h1B = h

Thus g = h if f is identity.

CASE 2. Involution

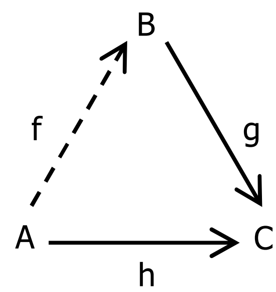
Given a map z: W → W, if zz = 1W, then z is an involution. The internal diagram of an involution is shown below, with the composition first followed by the composite.



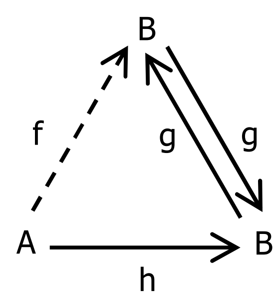


<u>CASE 2R.</u>

Given that the following triangle



commutes i.e. gf = h, we have to find 'f' satisfying gf = h. Given that 'g' is an involution i.e. g: B → B and gg = 1B, there's going to be the same arrow g in the opposite direction as shown below:



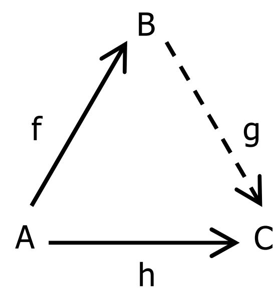
Let's try f = gh and see if it satisfies gf = h.

gf = ggh = 1Bh = h

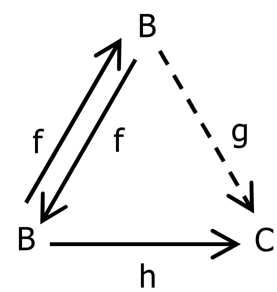
Thus f = gh if g is an involution.

CASE 2L.

Given that the following triangle



commutes i.e. gf = h, we have to find 'g' satisfying gf = h. Given that 'f' is an involution i.e. ff = 1B, we have the following diagram:



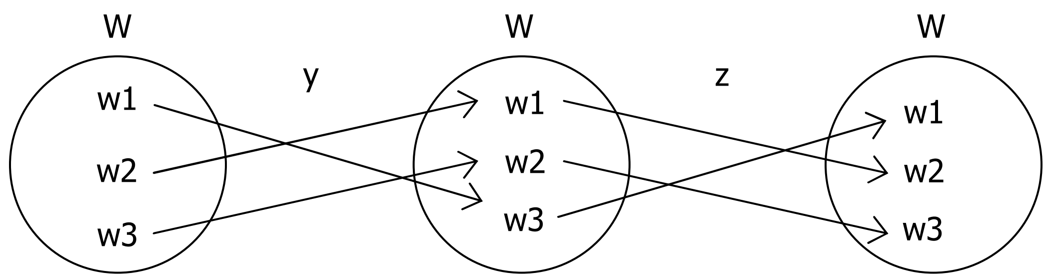
Let's try g = hf and see if it satisfies gf = h.

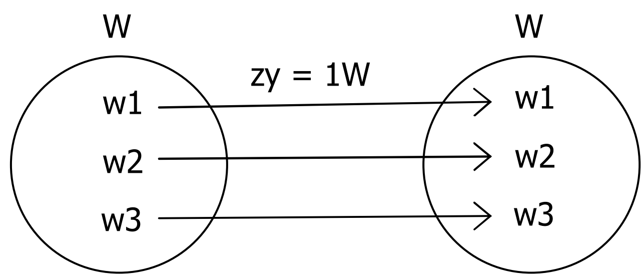
gf = hff = h1B = h

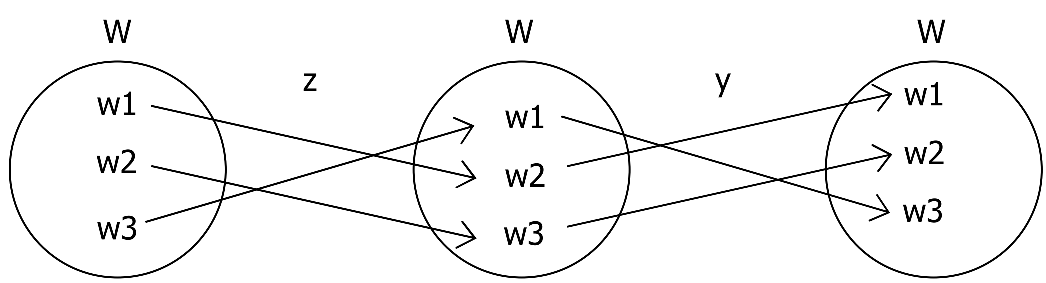
Thus g = hf if f is an involution.

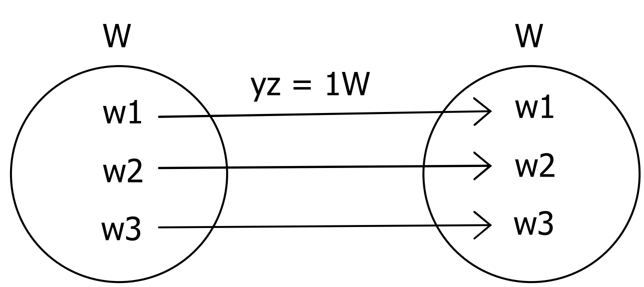
CASE 3. Automorphism

Given a map z: W → W, if there's a map y: W → W such that zy = 1W and yz = 1W, then z is an automorphism. The internal diagram of an automorphism is shown below, with the composition first followed by the composites.



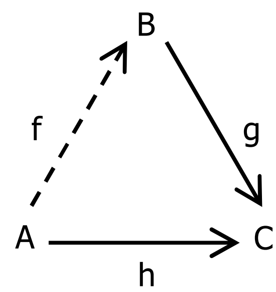




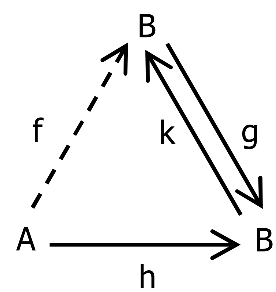


CASE 3R.

Given that the following triangle



commutes i.e. gf = h, we have to find 'f' satisfying gf = h. Given that 'g' is an automorphism g: B → B, there exists a k: B → B such that gk = 1B and kg = 1B.



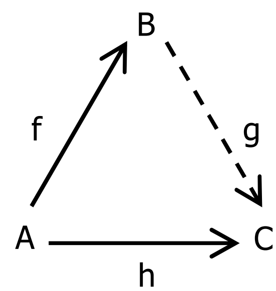
Let's try f = kh and see if it satisfies gf = h.

gf = gkh = 1Bh = h

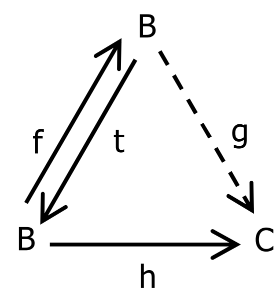
Thus f = kh if g is an automorphism.

CASE 3L.

Given that the following triangle



commutes i.e. gf = h, we have to find 'g' satisfying gf = h. Given that 'f' is an automorphism $f: B \rightarrow B$, there exists a $t: B \rightarrow B$ such that ft = 1B and tf = 1B.



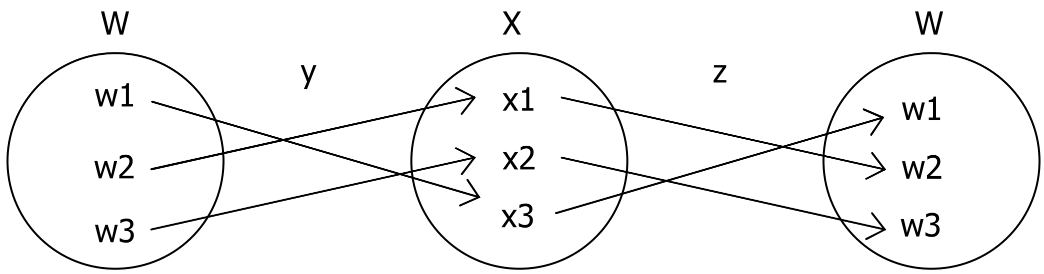
Let's try g = ht and see if it satisfies gf = h.

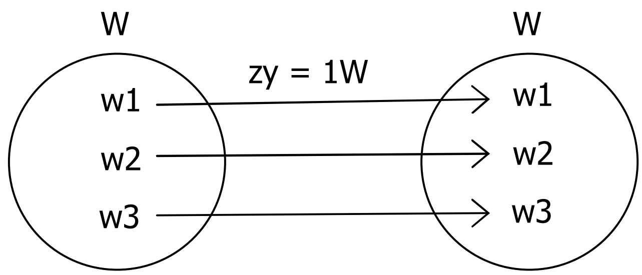
gf = htf = h1B = h

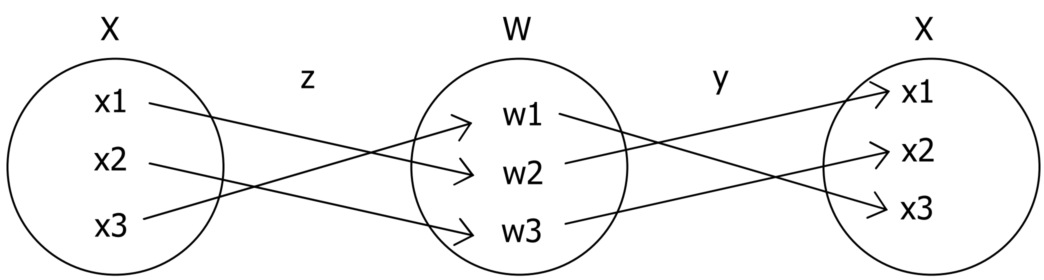
Thus g = ht if f is an automorphism.

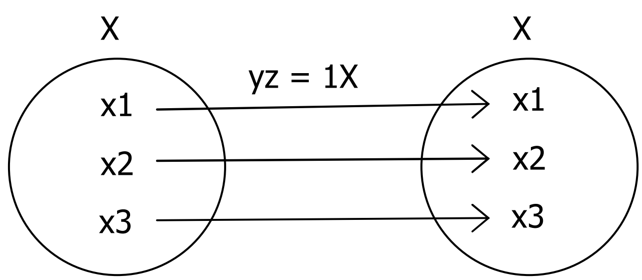
CASE 4. Isomorphism

Given a map y: W → X, if there's a map z: X → W such that zy = 1W and yz = 1X, then y is an isomorphism. The internal diagram of an isomorphism is shown below, with the composition first followed by the composites.



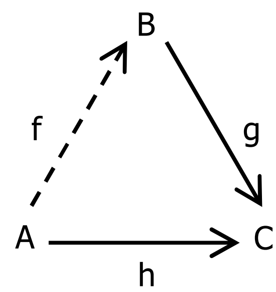




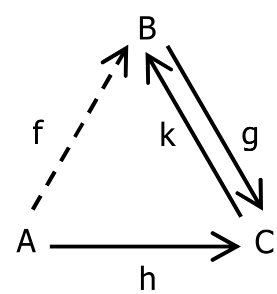


CASE 4R.

Given that the following triangle



commutes i.e. gf = h, we have to find 'f' satisfying gf = h. Given that 'g' is an isomorphism g: B → C, there exists a k: C → B such that kg = 1B and gk = 1C.



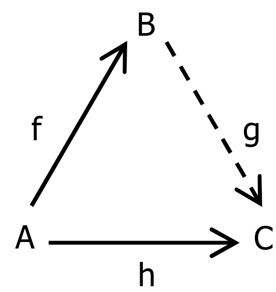
Let's try f = kh and see if it satisfies gf = h.

gf = gkh = 1Ch = h

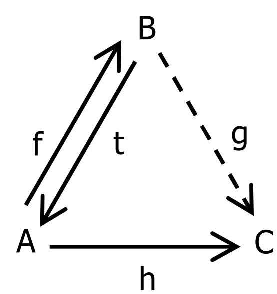
Thus f = kh if g is an isomorphism.

CASE 4L.

Given that the following triangle



commutes i.e. gf = h, we have to find 'g' satisfying gf = h. Given that 'f' is an isomorphism f: A → B, there exists a t: B → A such that tf = 1A and ft = 1B.



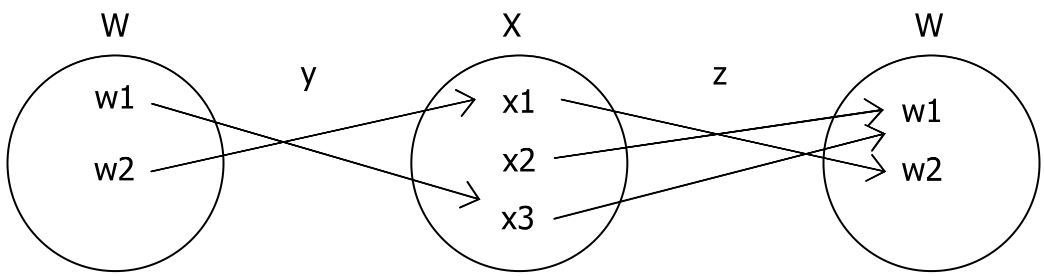
Let's try g = ht and see if it satisfies gf = h.

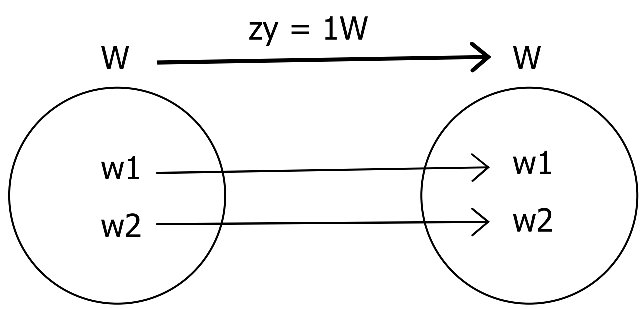
gf = htf = h1A = h

Thus g = ht if f is an isomorphism.

CASE 5. Retract-Section

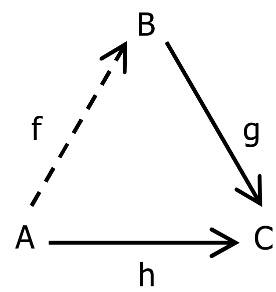
Given a map y: W → X and a map z: X → W such that zy = 1W, then y is a section of z, and z is a retract of y. The internal diagram of a retract-section pair is shown below.



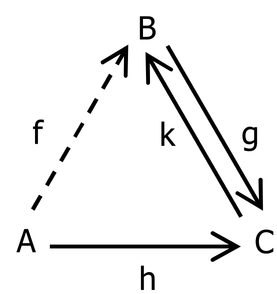


CASE 5R.

Given that the following triangle



commutes i.e. gf = h, we have to find 'f' satisfying gf = h. Given that 'g' is a retract g: B → C, there exists a section k: C → B such that gk = 1C.



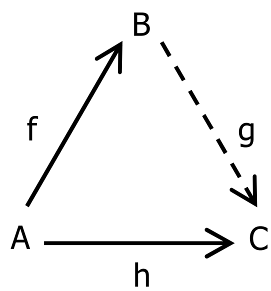
Let's try f = kh and see if it satisfies gf = h.

gf = gkh = 1Ch = h

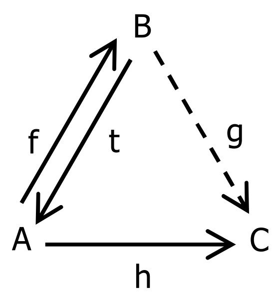
Thus f = kh if g has a section.

CASE 5L.

Given that the following triangle



commutes i.e. gf = h, we have to find 'g' satisfying gf = h. Given that 'f' is a section f: A → B with a retract t: B → A such that tf = 1A.



Let's try g = ht and see if it satisfies gf = h.

gf = htf = h1A = h

Thus g = ht if f has a retract.

Let's conclude here noting that we could have let the second known map also vary in which case we would have a total of 72 cases of which we looked at 10 cases, which hopefully are sensible.

Exercises: Understanding beyond Knowing

Never have I been so wrong and never have I been so happy at being wrong. It all started when I thought of doing the exercises in the Sets, maps, composition part of Conceptual Mathematics. Let me collect the exercises for your quick perusal:

Given A = {John, Mary, Sam} and B = {eggs, coffee}

Exercise 2: How many maps f are there with domain A and codomain B?

Exercise 3: Same, but for maps f: A → A

Exercise 4: Same, but for maps f: B → A

Exercise 5: Same, but for maps f: B → B

Exercise 6: How many maps f: A → A satisfy ff = f?

Exercise 7: How many maps f: B → B satisfy gg = g?

Exercise 8: Can you find a pair of maps A –f-> B –g-> A for which gf = 1A?

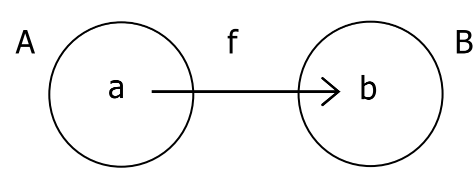
Exercise 9: Can you find a pair of maps B –h-> A –k-> B for which kh = 1B? If so, how many such pairs?

Looking at the exercises, which ask, in general terms, for the number of functions f: A → B from a given set A to another given set B, I said to myself: 'what could I, in doing the exercises, possibly understand beyond knowing the number of functions?' I thought, upon doing the exercises, I'd, at best, get to know the correct answer; I might get the number wrong if I count a function twice or miss a function. What more can there be to counting functions? If you count the number of eggs in a basket, you either get the count right or wrong; what else is there to understand, right? Well, I don't know much about counting eggs, but after counting the number of functions from one set to another set, and as I kept counting, I began to wake up to something universal across the numerous particular counts of functions, some understanding did develop beyond that of mere knowing the correct number of functions. Here is my understanding, without much further ado, for you to judge for yourself.

Counting Functions and Place-Value Notation

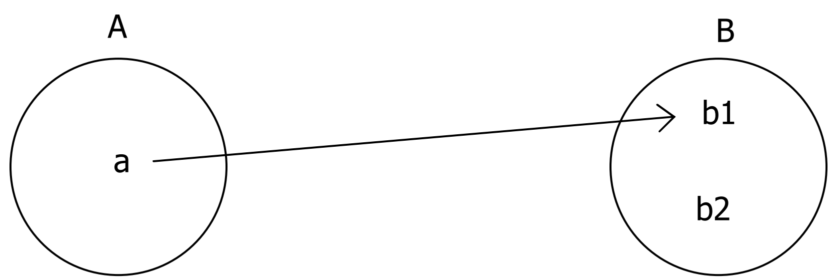
Please allow me to start simple and ramp it up slowly. Let's consider two sets A = {a} and B = {b} each with one element. How many functions f: A → B are there from set A to set B? This is easy. There is one element 'a' in the domain set A and there is one element 'b' in the codomain set B. Obviously there is just one function f: A → B, which takes the only element 'a' of A to the only element 'b' of B as shown below, which we label as Case 1&1 (with the 1st '1' denoting the number of elements in the domain and the second '1' denoting the number of elements in the codomain):

Case 1&1

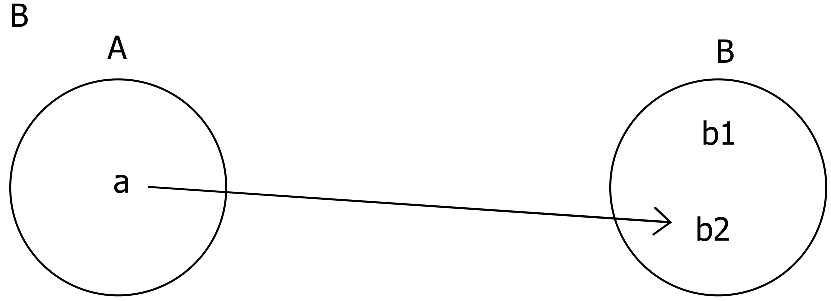


Looking at the internal diagram of Case 1&1, it is clear that there are no other functions f: A → B. Now let's consider little bit more, but not too much, complicated scenario, where A = {a} and B = {b1, b2}. In this case there are two functions: one function f1 taking the only element 'a' of domain to one of the elements 'b1' of the codomain and the other function f2 taking the only element 'a' of the domain to the other element 'b2' of the codomain. The corresponding (Case 1&2) internal diagrams are shown below:

f1: A → B

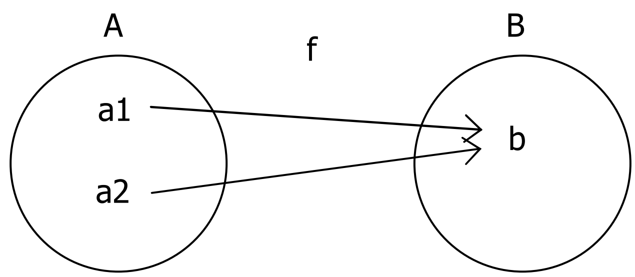


f2: A → B



Looking at the above internal diagrams and recollecting from previous sessions, we can assure ourselves that the number of functions from a single-element set to any codomain set B is equal to the number of elements in the codomain. In other words, the functions from singleton to a set are in one-to-one correspondence with the number of elements of the set. Now let's look at Case 2&1 i.e. the case where the domain A has 2 elements and the codomain B has just one element. The internal diagram of Case 2&1 is shown below:

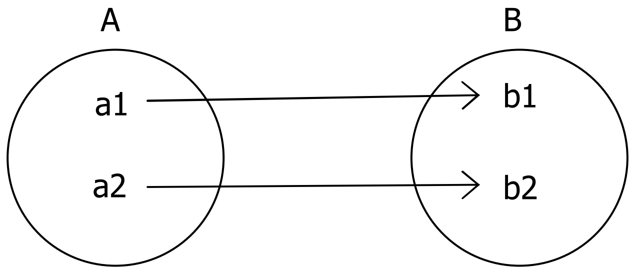
Case 2&1



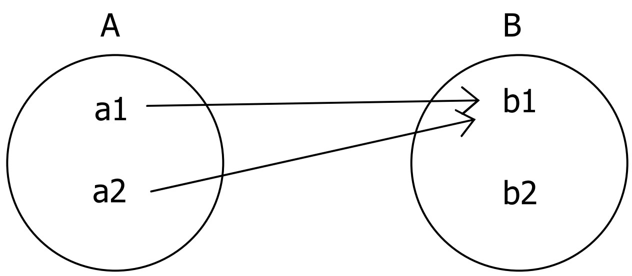
Here again, drawing on our previous experience, and of course going by the definition of function, there is only 1 function f: A → B, which takes both elements of domain A to the only element of the codomain B. There's little more we can say here, which is that there is only one function f: A → B irrespective of however many elements the domain set A has, as long as the codomain B has just one element.

If all of the above feels like lettuce and more lettuce making you wonder when things are going to get spicy, we are almost there and hold your 'are we there yet?' for a minute, please. Let's, now, look at the case of number of functions f: A → B, where both A and B has 2 elements each (Case 2&2). How many functions are there from a 2-element domain set to a 2-element codomain set? Let's list:

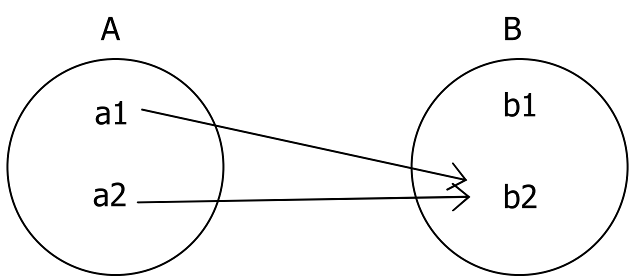
f1: A → B



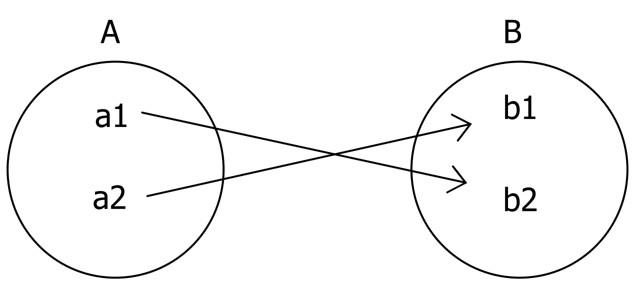
f2: A → B



f3: A → B



f4: A → B



Looking at the above internal diagrams, it looks as though all possible functions from A = {a1, a2} to B = {b1, b2} are accounted for; it doesn't seem like we missed counting a function or counted a function twice. Following on the heels of this reassuring thought, I had a not so comforting thought that as the domain and/or codomain sizes increases the risk of counting a function twice or forgetting a function also increases, which led me think of a systematic way to enumerate all functions from any given set to any given set. Whatever this systematic way might be it should always give (i) the total number of functions and (ii) an orderly listing of all functions. With no way in sight, I closed my eyes and said to myself silently (on a tangential note, isn't it amazing that silence—the absence of sound—the "zero" of auditory modality has so much structure; there's so much to understand and it's high time we make 'comprehension' fashionable again): 'there's no way other than studying the definition of function in order to find a procedure to calculate the total number of functions and to orderly list the total number of functions from any given set to any given set.' So, let's look at the definition of function:

A function f: A → B assigns for each element 'a' of domain A exactly one element 'b' of codomain B such that 'b' is the value of the function f at 'a'. Two words that stood out in the definition of function calling for my attention are 'value' and 'at'. The elements 'a' of the domain A can be thought of as labels for locations in the space A or places in A where the function f takes values, different values. These two words: place and value, possible, by way of association reminded me of the place-value notation of numbers (I must admit for some reason I can not reconstruct the thread of thought that led me from the definition of function to the notion of place-value notation step-by-systematic-step; hence the recourse to association). Now let's see how far the parallels between functions and numbers go. There is domain, which corresponds to place and there is codomain corresponding to value and the total number of functions corresponds to the total number of numbers for a given number of places and values. Do they? Let's see. Once again, let's begin simple, with the simplest possible case (in case you are wondering what's with this obsession with starting simple, the virtues of starting simple can be readily discerned by comparing and contrasting the methods and deliverables of science and religion both of which are in the business of making 'sense of it all.' Science, in trying to make sense of it all, begins with making sense of simple things and builds on these sensible things, whereas religion takes on the most incomprehensible of all and instead of admitting the failing of the method comes up with some cock-and-bull stories), where there is only one place (1 i.e. unit's place) and only one value (0). Numbers are obtained by assigning all possible values to all the places. Since there is only one place and only one value '0', the value '0' is assigned to the only place giving the only number 0. Thus if we have one place and one value we get one number. How about if we have one place and two values {0, 1}? The only one place can take the value of '0' or the value of '1', giving rise to two numbers '0' and '1'. Now let's look at the case where we have two places {2, 1} and only one value {0}. Since there is only one value, both the places have to take the only one value as a result of which we have only one number '00'. In case you haven't noticed, these cases mirror Case 1&1, Case 1&2, and Case 2&1 (above). Here also we are confronted with the question of given a number of places and a number of values, how many numbers are there. Let's now consider the case of 2 places P = {2, 1} and 2 values V = {0, 1}. The total number of numbers N = |values|^|places| = |V|^|P|. Therefore N = 4 and the four numbers are:

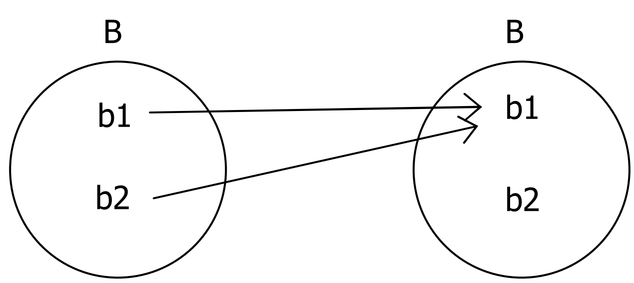
2^1	2^0	
2	1	
0	0	0
0	1	1
1	0	2
1	1	3

In the above depiction {2, 1} depict places and each row below the labels of places is a number, which is noted to the right of each row. We can readily recognize the above as binary number system. As expected this is exactly the number of functions f: A → B, with |A| = 2 and |B| = 2 is same as the total number of numbers and is given by #f = |B|^|A| = 2^2 = 4. The place-value notation analogy not only helped us get a formula to calculate the total number of functions from any given set to any given set, but also, as a bonus, gave us a systematic procedure to enumerate each and every function. Now, knowing that I'll never make the mistake of counting a function twice or of missing a function, I can sleep tight. Now it's straight-forward to do Exercises 2 through 5. All we have to do, in order to calculate the number of functions f: A → B is use the above formula i.e. #f = |codomain|^|domain|.

Discerning UNIVERSALS in Particulars

To give a foretaste, let's consider 'Mary taught John', which can be readily interpreted in terms of the Mary and the John (things out-there) and the taught (events out-there); we can, if we like, go to Mary and thank her on behalf of John or go to John and tell him how lucky he is to have Mary teach him. This is all 'looking at the given particulars (Mary taught John) and seeing particulars (things and events). This is not all there is. We can, however read into the very 'Mary taught John.' a sentence, a statement, or subject, verb, object, or nouns and verb and many more UNIVERSALS that are not out-there until one imagines or creates or constructs grammar (on yet another tangential note, I wonder, in these purposeful [satirically speaking] days, would we as a kind—mankind encourage, nurture, promote endeavors that resulted in, to name one, grammar—a product of the pursuit of comprehension). At this point you probably are thinking: 'yes, we got the point; in trying to count the number of functions you found an analogy between numbers and functions. Great! Can we go now? No, not yet. Understanding, as if not to be outdone by reality, is infinitely deep; the more you focus the deeper your understanding. Moving along from this mumbo-jumbo, let's look at the other exercises. The remaining exercises also ask for the number of functions f: B → B, with the additional constraint that the functions f should satisfy ff = f. Whenever I run into a problem, for some reason, I start looking around, in part because I can't quite figure out how to solve the problem or to see the big picture of which the given problem is a part, to situate the problem in its natural habitat. In plain English, we are not just going to say 'there are x number of functions f: B → B that satisfy ff = f and walk away. Let the celebrations begin. With ease of typing in mind, let's take B = {b1, b2}. With this B as both domain set and as codomain set, the total number of functions f: B → B is given by #f = |B|^|B| = 2^2 = 4. Let's draw the internal diagrams of all four functions.

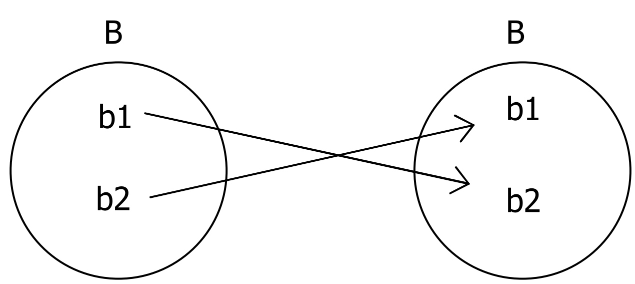
f1: B → B



<u>2^1 2^0</u>

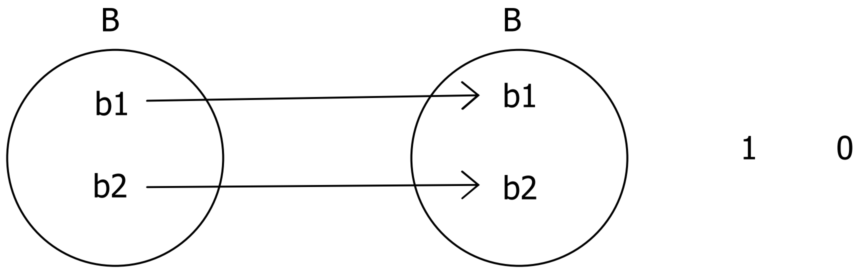
0 0

f2: B → B

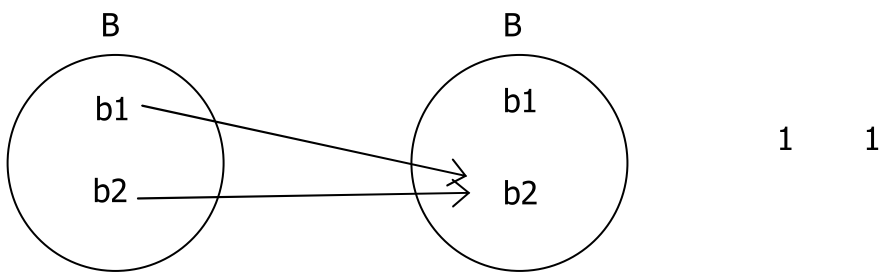


0 1

f3: B → B



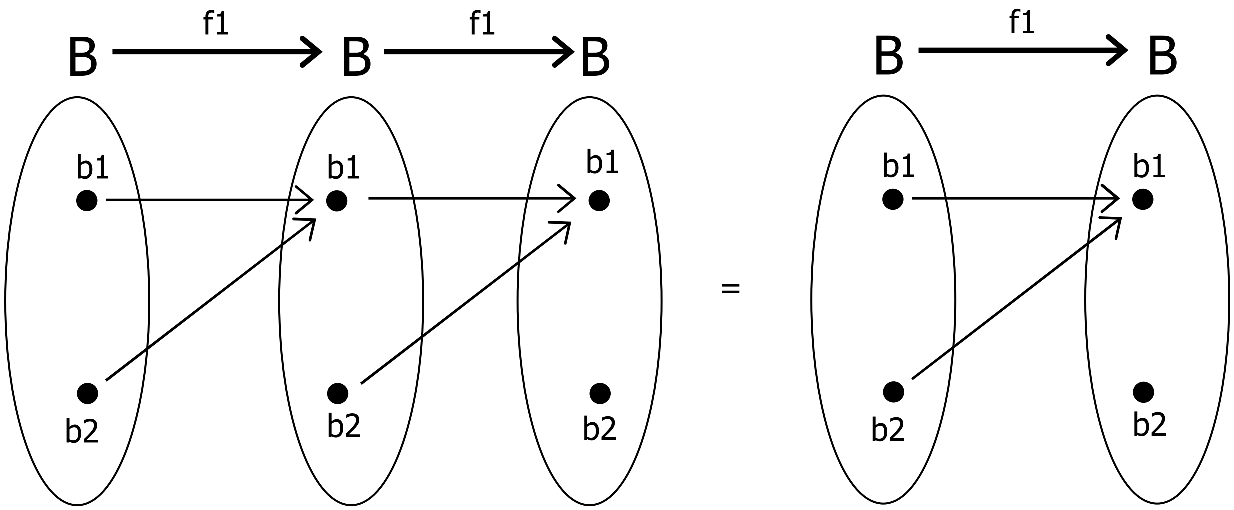
f4: B → B



Next to each internal diagram is drawn the corresponding number to highlight the correspondence between numbers and functions.

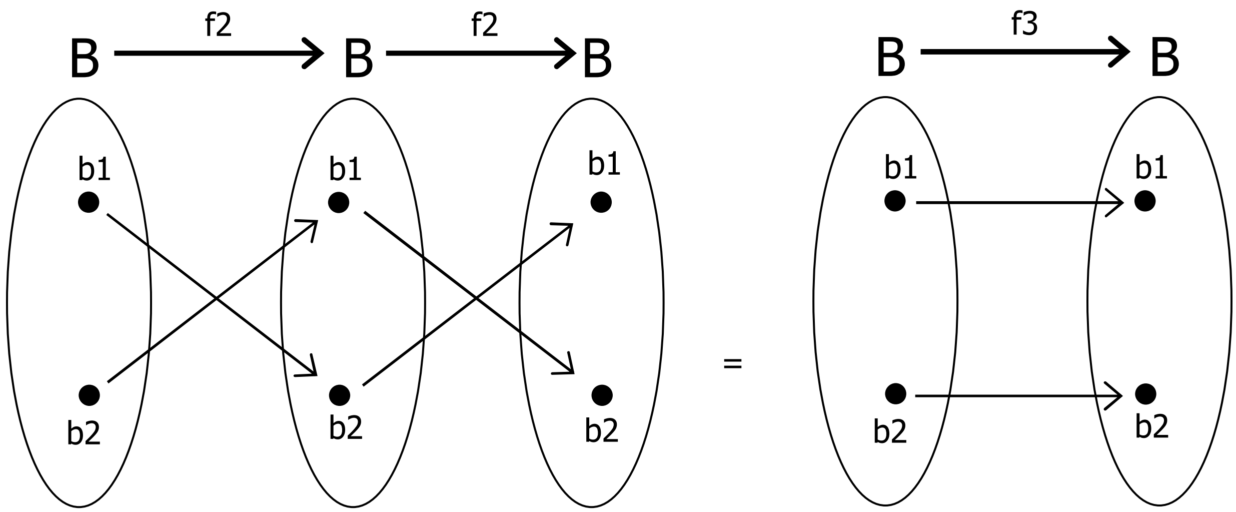
Now back to the question: we were asked to find functions f: B → B, which satisfy ff = f. Since there are four functions, given |B| = 2, f1, f2, f3, and f4, we have to evaluate four composites f1f1, f2f2, f3f3, and f4f4 and see what the result is. However these are not the only pair-wise composites. There are a total of 4*4 = 16 composites (to which we will come later on) of which 4 composites have the potential to satisfy ff =f. Let's draw the internal diagrams of each and every one of the 16 composites.

Case: f1f1



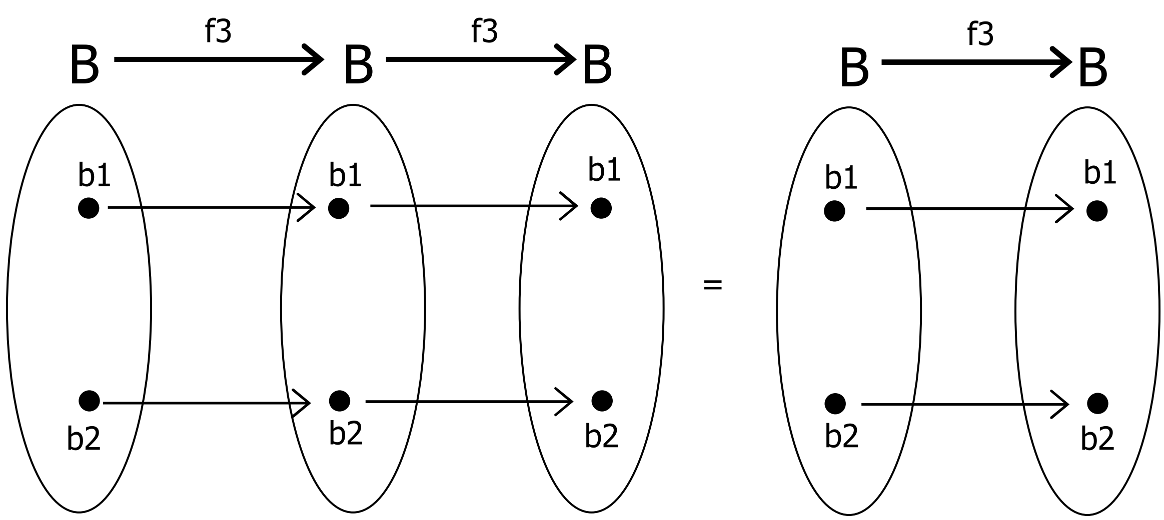
Out of the four possible functions which could potentially satisfy ff = f, we already found one that satisfies ff = f, and now we have to see if any of the other three satisfy ff = f. Since, as we have been implicitly or explicitly, stating that the point of doing these exercises is not just to know the number of this or that, but to participate in the practice of mathematics i.e. the exercise of discerning universals in particulars, and in turn use the universals to structure or organize the particulars so that this place we call home begins to look more and more sensible—comprehensible. Alright, let's draw all the rest i.e. f2f2, f3f3, and f4f4.

Case: f2f2



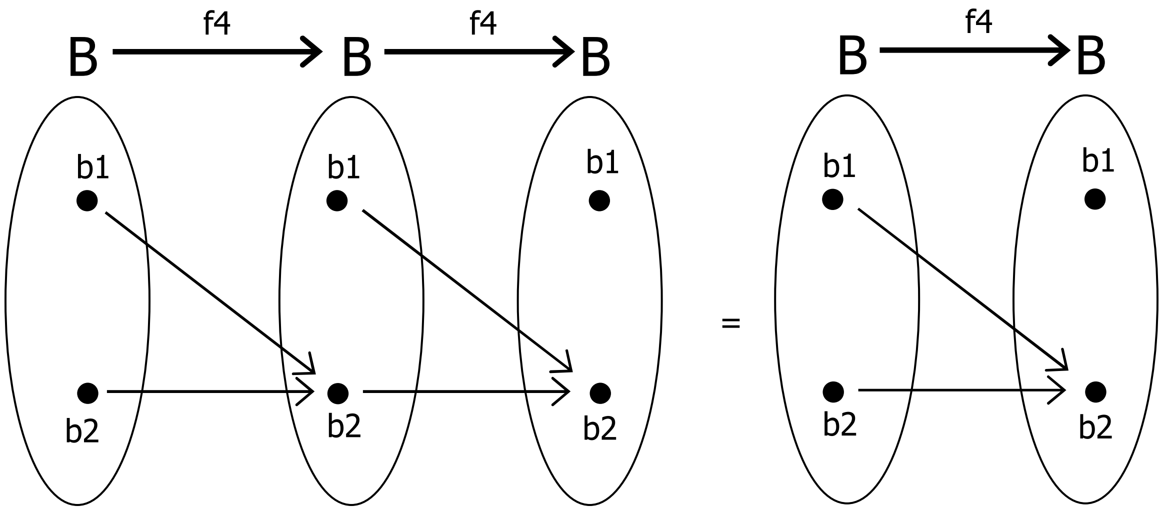
From the above internal diagrams, it is clear that f2f2 ≠ f2. How about f3? Let find out.

Case: f3f3



From the above diagram we clearly see that f3f3 = f3. Now let's look at the final composite f4f4.

Case: f4f4



Here again f4f4 = f4. Thus out of the four f: B → B, three, f1, f3, and f4 satisfy ff = f. Yes, we did find out that out of the four endomaps of a 2-element set, only three endomaps satisfy idempotence, but, in the wee hours of the dark night of soul (conscious experience) one tends to feel that there wasn't much of enlightenment or is there. Well, in bright daylight, that's not quite true.

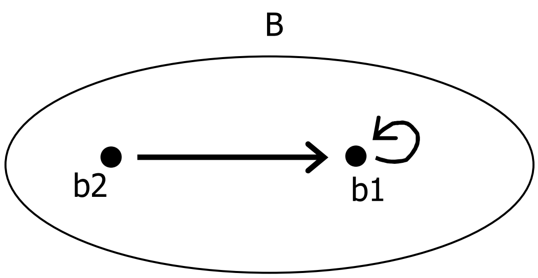
1. Case f1f1 and case f4f4 tell us that the composite of two non-onto and non-one-to-one functions is also non-onto and non-one-to-one.
2. Case f2f2 and case f3f3 tell us that the composite of two onto and 1-1 functions is also onto and 1-1 function.

Is this all the enlightenment; how about the composite of a non-onto and non-1-1 function and onto and 1-1 function.

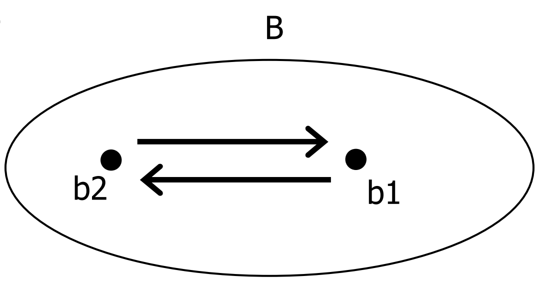
Fortunately, the above simple exercise is rich enough to answer this and many more questions. Before we get into evaluating all 16 pair-wise composites, let's see if there's more juice in the four composites we have already evaluated.

Since most of the time there is more than one way to do an exercise, let's try the above exercise in a slightly different way. Since we are dealing with endomaps f: B → B, which have a special type of internal diagram, let's see how the internal diagrams of the four endomaps f: B → B look.

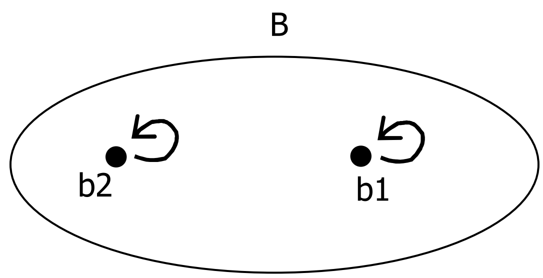
Case: f1



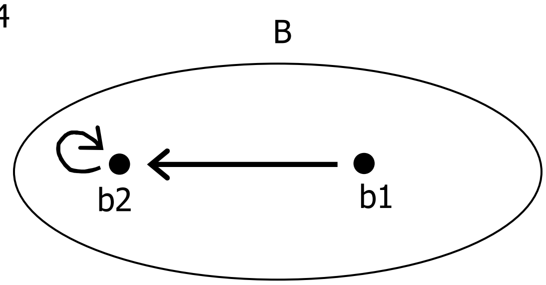
Case: f2



Case: f3



Case: f4

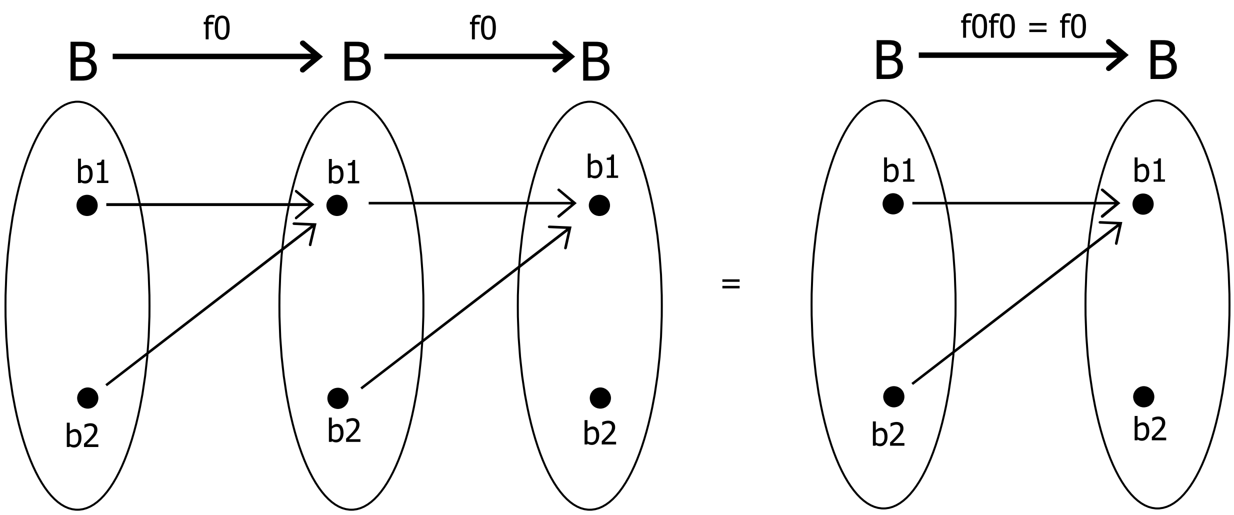


Looking at the above four internal diagrams of the endomaps f: B → B, we can discern some characteristics of endomaps which lend to satisfying ff = f.

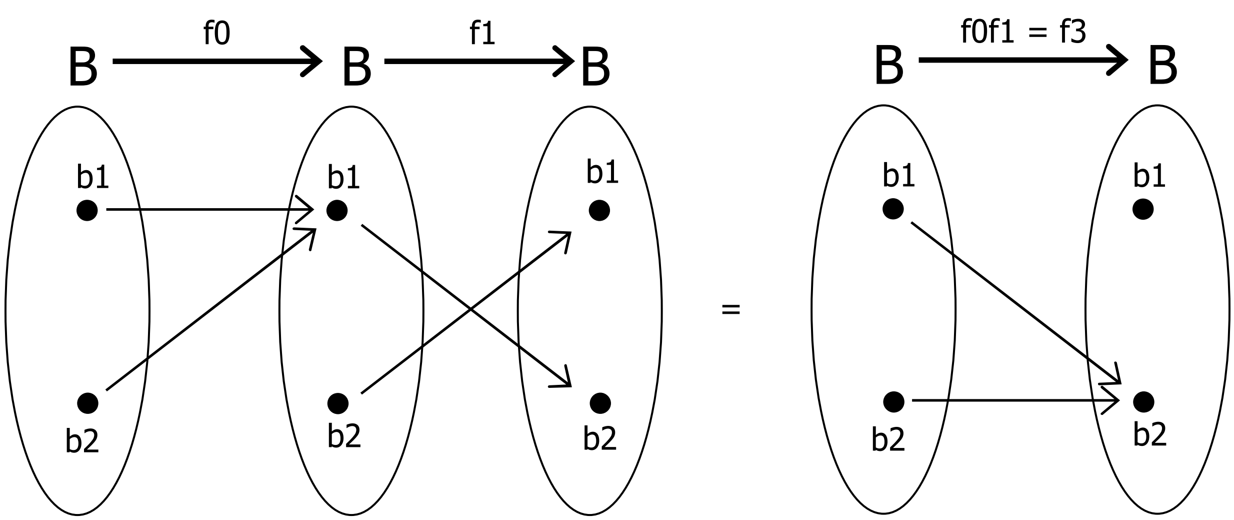
1. Endomaps with cycles do not satisfy idempotence as is the case with f2f2 ≠ f2.
2. Endomaps with loops and fixed-points with a stem of at most length 1 (e.g. f1 and f4) satisfy idempotence: f1f1 = f1, f3f3 = f3, & f4f4 = f4

Now let's evaluate all 16 pair-wise composites to see what happens when we form the composite of a non-onto and non-1-1 function and a onto and 1-1 function, among others. But, first, let's list out the 16 composites: f1f1, f1f2, f1f3, f1f4, and f2f1, f2f2, f2f3, f2f4, and f3f1, f3f2, f3f3, f3f4, and f4f1, f4f2, f4f3, f4f4. Furthermore there is a one-to-one correspondence between functions and numbers; more specifically function has domain and codomain and number has places and values corresponding to the domain and codomain, respectively. Once we identify function with number, we found that the total number of functions (for a given domain and codomain) is equal to the total number of numbers (for a given number of places and a given number of values). However our (possibly misguided) program of arithmetizing the algebra of composition runs into a major (eye-opening?) hurdle as we fail to find an analog for composition in arithmetic operations. Be that as may, we will we will place the numbers corresponding to the functions corresponding to the numbers in a given place-value notation (determined by domain and codomain) at their respective headers. In other words the numbering of functions starts from 0 as in f0 to indicate that it corresponds to the number 0 0; f1 corresponds to 0 1; f2 corresponds to 1 0; f3 corresponds to 1 1.

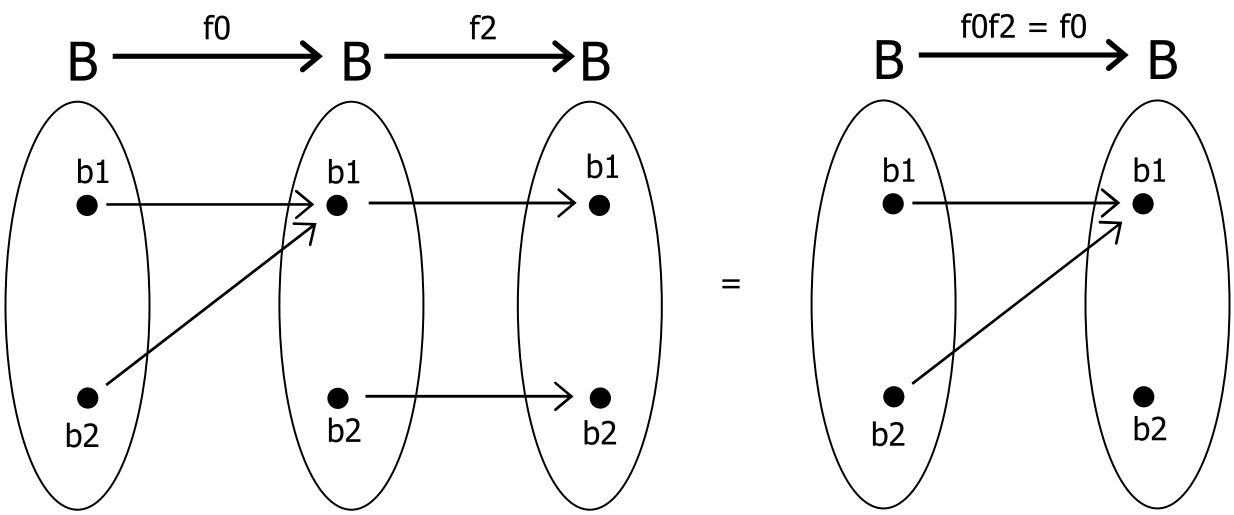
1. f0f0 = f0 (0 o 0 = 0; 'o' stands for some undefined operation)



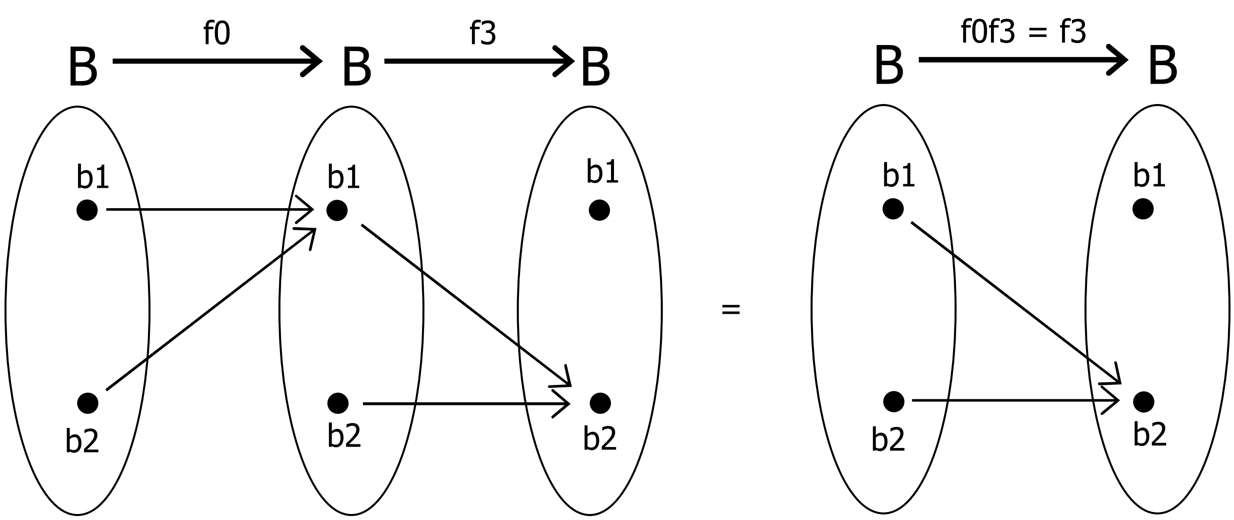
2. f0f1 = f3 (0 o 1 = 3; 'o' stands for some undefined operation)



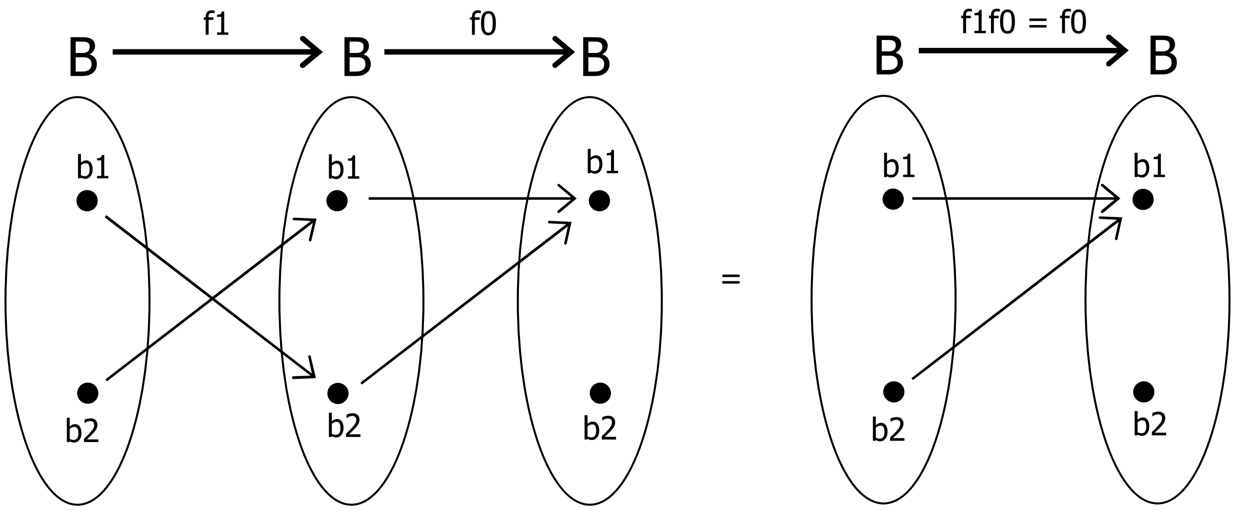
3. f0f2 = f0 (0 o 2 = 0; ‘o’ stands for some undefined operation)



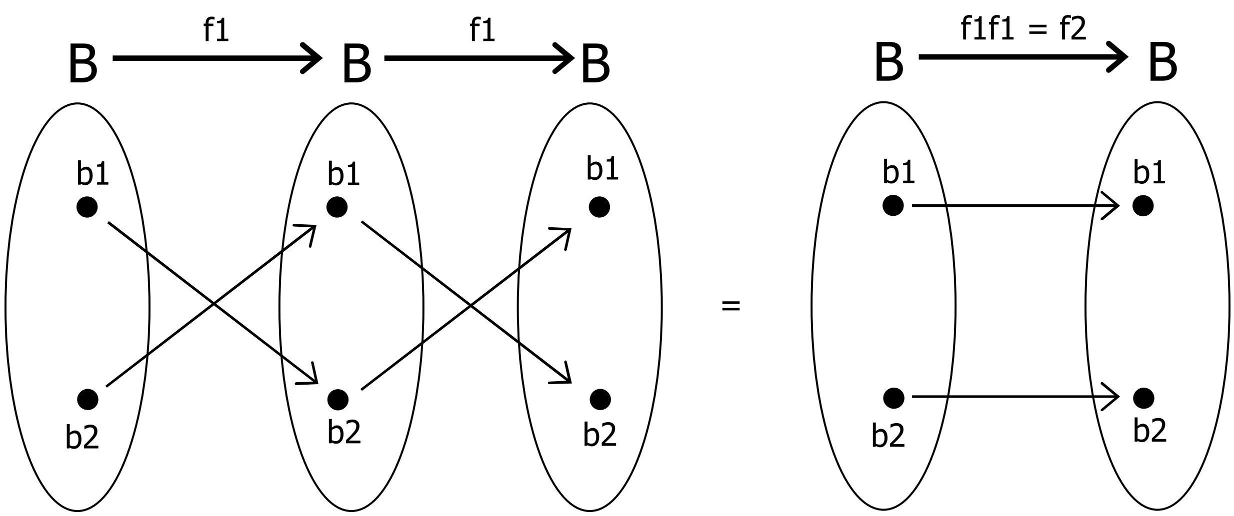
4. f0f3 = f3 (0 o 3 = 3; ‘o’ stands for some undefined operation)



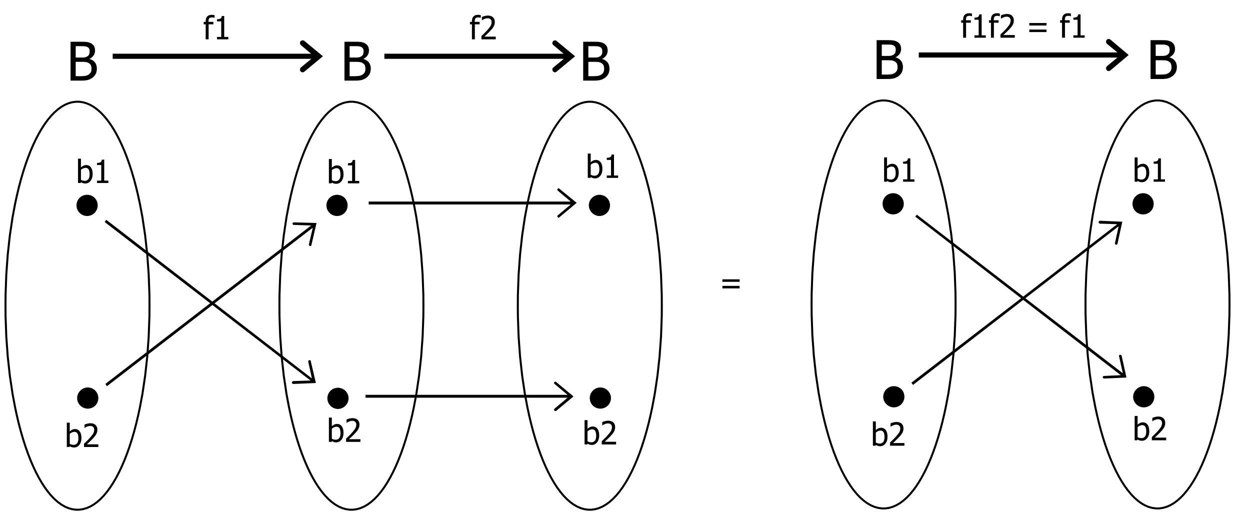
5. f1f0 = f0 (1 o 0 = 0; ‘o’ stands for some undefined operation)



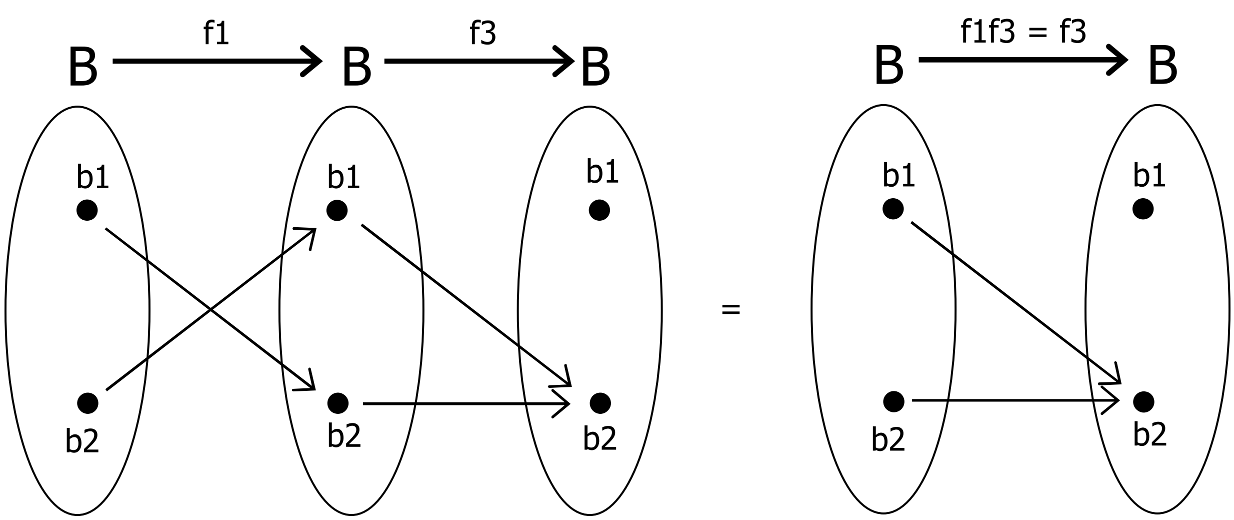
6. f1f1 = f3 (0 o 1 = 2; ‘o’ stands for some undefined operation)



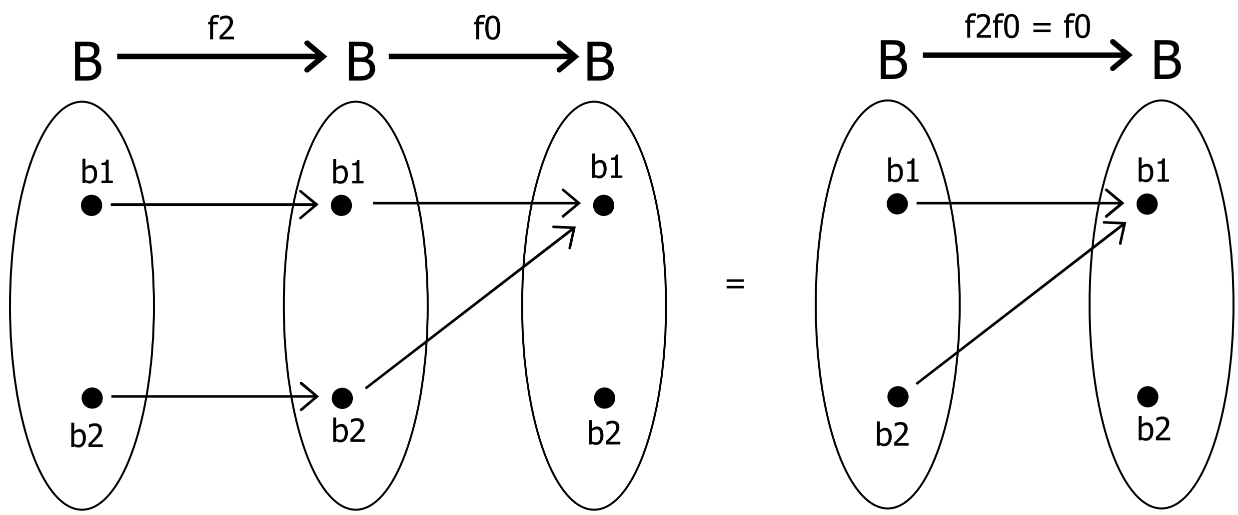
7. f1f2 = f2 (1 o 2 = 2; ‘o’ stands for some undefined operation)



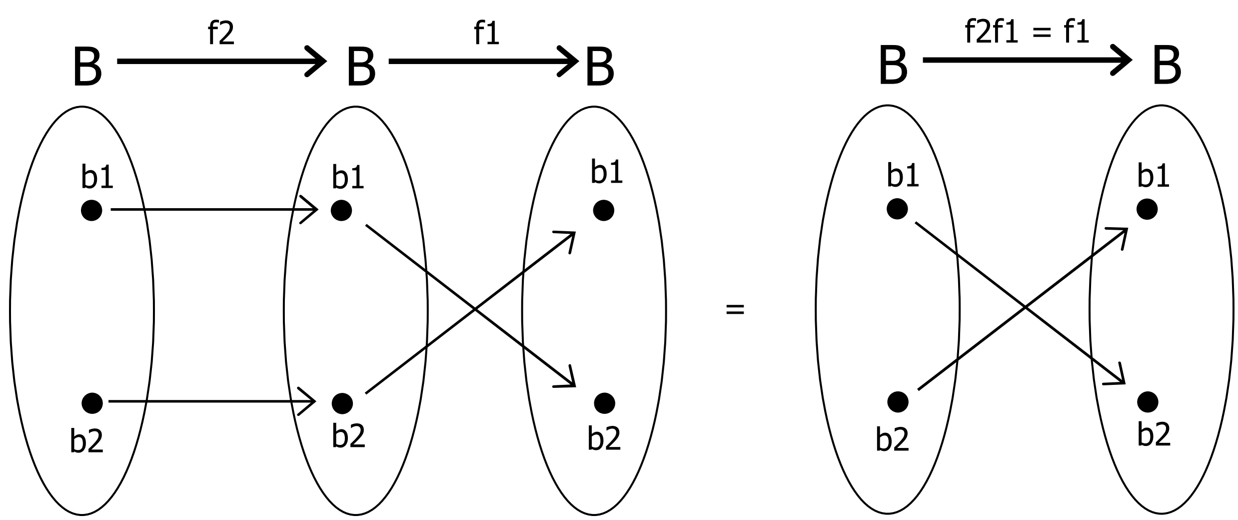
8. f1f3 = f3 (1 o 3 = 3; ‘o’ stands for some undefined operation)



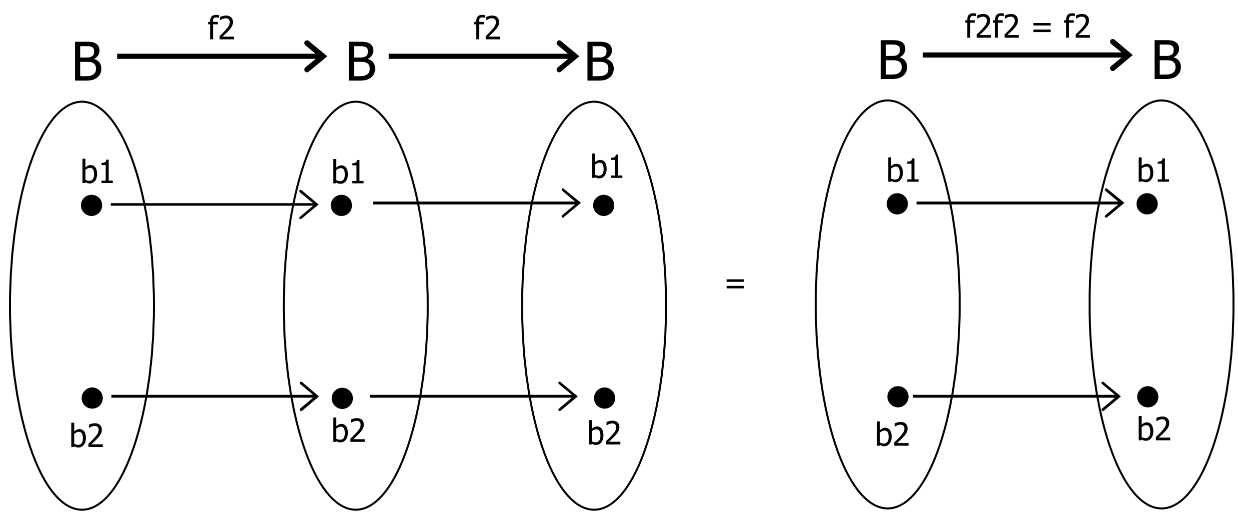
9. f2f0 = f0 (2 o 0 = 0; ‘o’ stands for some undefined operation)



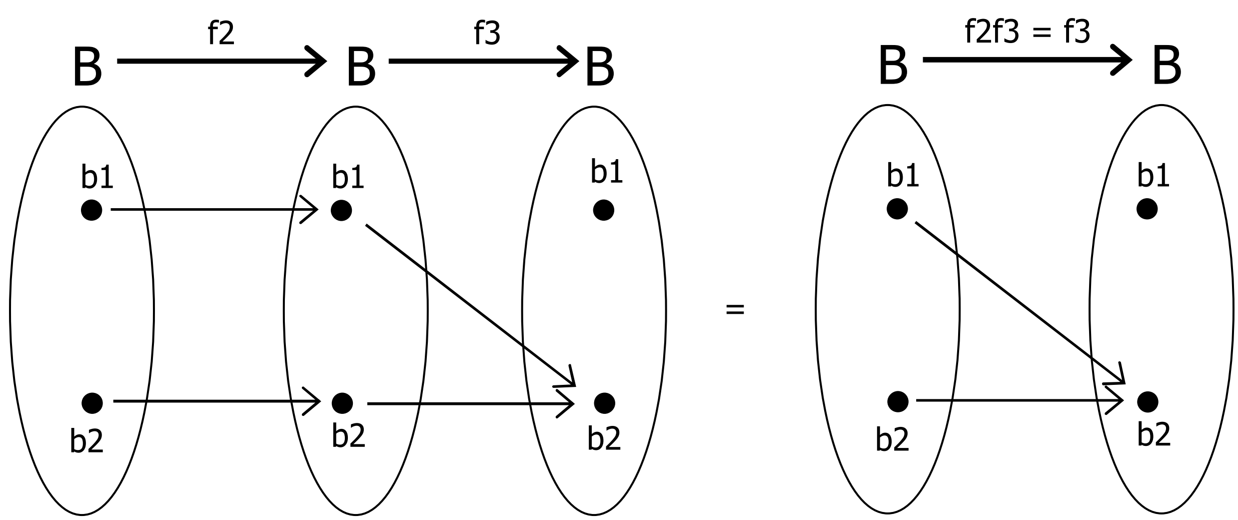
10. f2f1 = f1 (2 o 1 = 1; ‘o’ stands for some undefined operation)



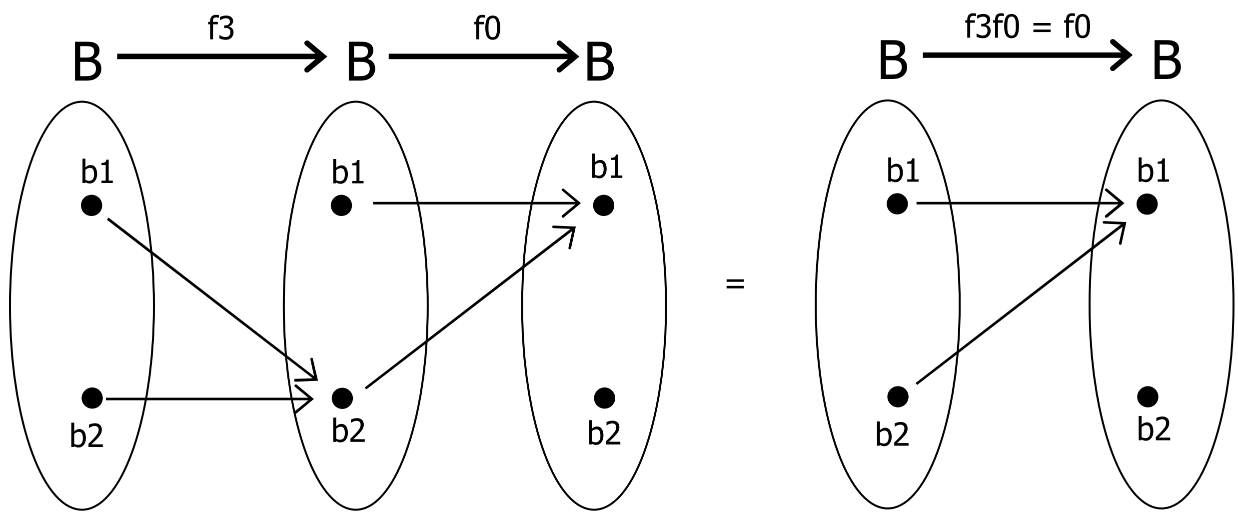
11. f2f2 = f0 (2 o 2 = 2; ‘o’ stands for some undefined operation)



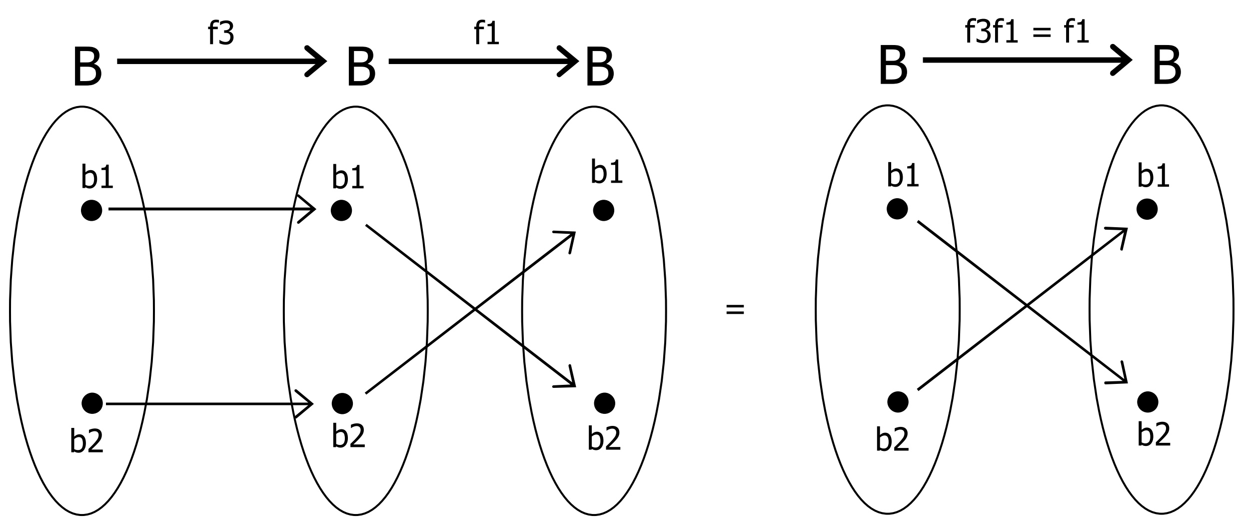
12. f2f3 = f3 (2 o 3 = 3; ‘o’ stands for some undefined operation)



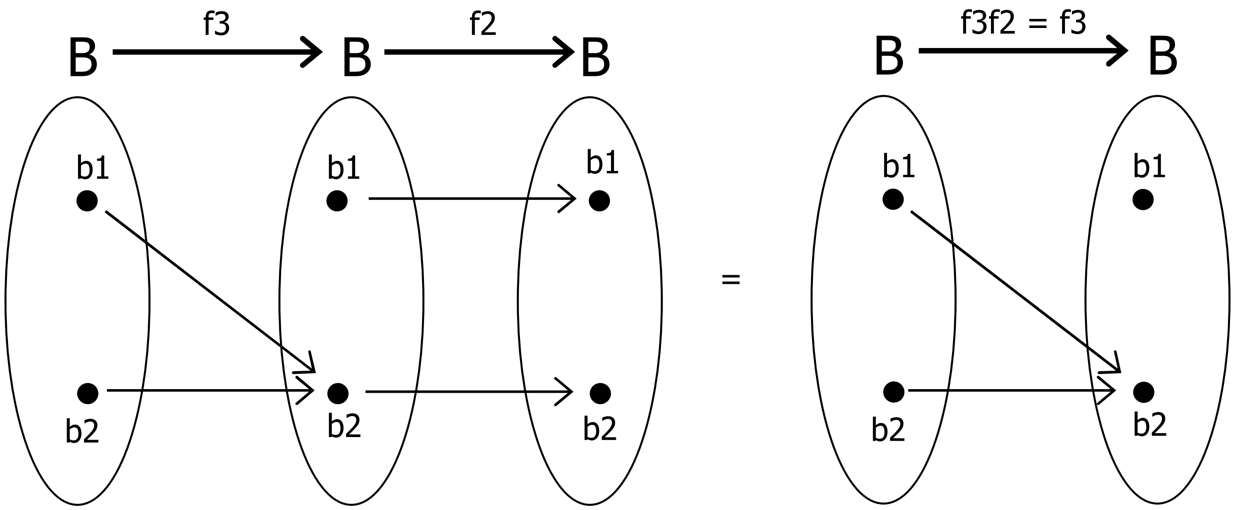
13. f3f0 = f0 (3 o 0 = 0; 'o' stands for some undefined operation)



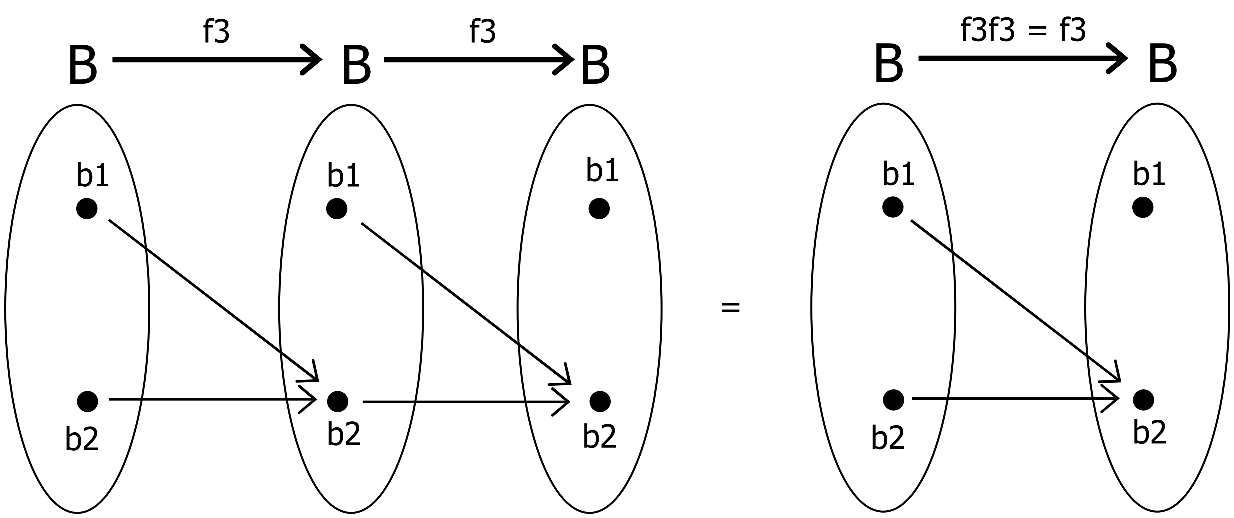
14. f3f1 = f1 (3 o 1 = 1; 'o' stands for some undefined operation)



15. f3f2 = f2 (3 o 2 = 3; ‘o’ stands for some undefined operation)



16. f3f3 = f3 (3 o 3 = 3; ‘o’ stands for some undefined operation)



Looking at the above examples, we notice some general features:

1. Composite of two non-onto and non-1-1 functions is a non-onto and non-1-1 function
2. Composite of non-onto and non-1-1 function with an isomorphism (onto and 1-1) is an non-onto and non-1-1 function
3. Composite of two isomorphisms is an isomorphism

I know of the proof of 3 (see below), but can't think of a proof for 1 and 2.

Composite of Isomorphisms

Just for the fun of it let's prove that the composite of two isomorphisms is an isomorphism. First what is an isomorphism? Isomorphism is a function f: A → B that is both 1-1 and onto. Speaking more verbally, a 1-1 function is a function which takes distinct elements of the domain to distinct elements of codomain B. Succinctly, if f(a1) = f(a2), then a1 = a2. Now on to onto function, which is a function such that every element 'b' of the codomain B is the value of some 'a' of the domain a, i.e. B = f(A). Let's get more concrete. Let's say f: A → B is an isomorphism with f(a) = b, and let's construct another function g: B → A such g(b) = a, which can be written as g(f(a)) = a simply by way of substituting b with f(a). Now by associative law we can write (gf)a = a for all a in A, which means gf = 1A. Similarly, fg = 1B.

Now we have all the important ingredients to test whether the composite of two isomorphisms is an isomorphism. Let f: A → B and g: B → C be both isomorphisms, we have to show that gf: A → C is an isomorphism. Given f: A → B is an isomorphism, there's an fi: B → A such that fif = 1A and ffi = 1B. Similarly, given g: B → C is an isomorphism, there's an gi: C → B such that gig = 1B and ggi = 1C. Given all these, we have to show gf: A → C is an isomorphism. Let's say gf is an isomorphism in which case there is figi: C → A such that figigf: A → A = 1A and gffigi: C → C = 1C. Let's now show that that's the case: Case (i). figigf = fi1bf = fif = 1A. Case (ii). gffigi = g1Bgi = ggi = 1C.

Arithmetic of Composition of Functions

Now let's go back to our good old business of the correspondence between functions with domain and codomain and numbers with places and values. Let's, for the fun of it, tabulate the numerical representations of functions and their composites.

1. 0 o 0 = 0
2. 0 o 1 = 3
3. 0 o 2 = 0
4. 0 o 3 = 3
5. 1 o 0 = 0
6. 1 o 1 = 2

7. 1 o 2 = 2

8. 1 o 3 = 3

9. 2 o 0 = 0

10. 2 o 1 = 1

11. 2 o 2 = 2

12. 2 o 3 = 3

13. 3 o 0 = 0

14. 3 o 1 = 1

15. 3 o 2 = 3

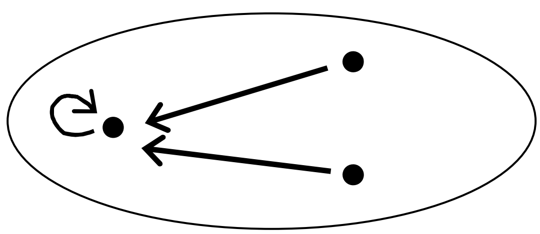
16. 3 o 3 = 3

Looking at the above numerical equalities, I couldn't imagine what the operation 'o' might be that satisfies all the equations (and many more). I am not even sure about the legitimacy of the correspondence between numbers (with their places and values) and functions (with their domains and codomains), notwithstanding the fact that given a function one could find the corresponding number following a systematic procedure. Similarly, given a number one could find the corresponding function. Of course, this all needs adopting a convention. In addition to the question of the nature of the numerical operation 'o' corresponding to the operation of composition of functions, how would one go about formally capturing the transformation of functions into numbers and numbers back into functions. At the risk of revealing my ignorance and making a fool of me, can we construct a functor between the category of functions and (say) monoid so as to capture the above arithmetization of algebra.

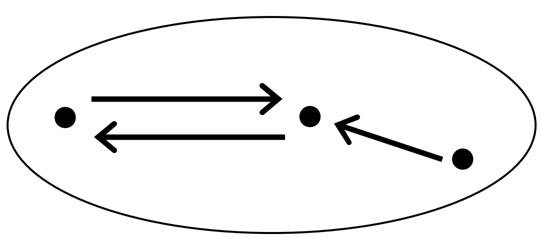
Fixed-points and Cycles

Let's do the exercise where f: A → A, with |A| = 3. In this case we have a total number of functions, #f = |codomain|^|domain| = 3^3 = 27. We would like to find out how many of these 27 functions satisfy ff = f. Since these f's are endomaps, let's try to figure out whether ff = f from the endomaps using the understanding gained from working out the case of |A|= 2 (see above). (Note that the numbering starts with 0 or 0 0 0.)

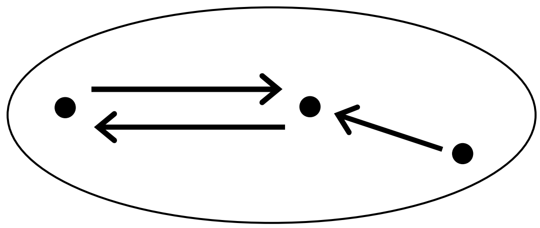
Case: 0 0 0 or 0



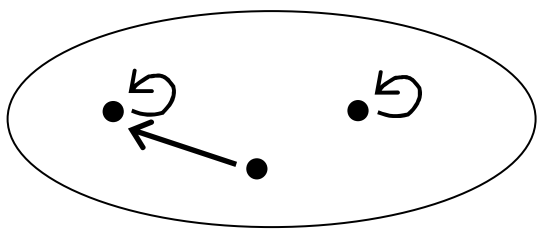
Case: 0 0 1 or 1



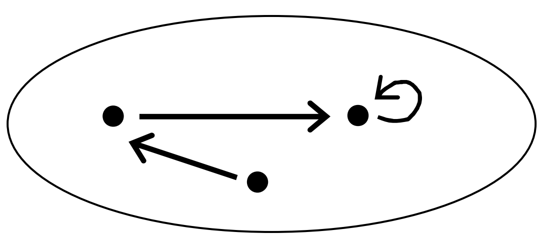
Case: 0 0 2 or 2



Case: 0 1 0 or 3



Case: 0 1 1 or 4



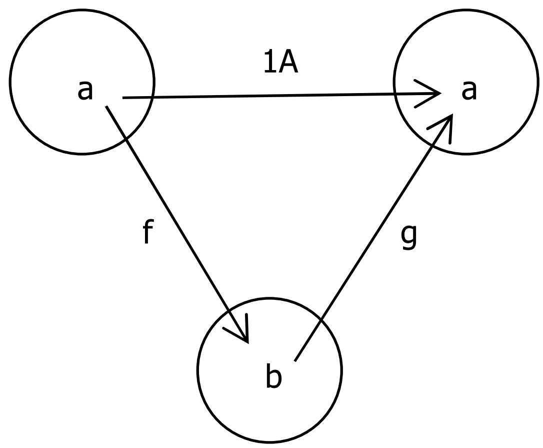
From the above endomaps we find that endomaps with fixed-points with path-length of at most 1 (e.g. Case 0 and Case 3) satisfy ff = f, while endomaps with cycles (e.g. Case 1 and Case2) and endomaps with fixed-points with path-length greater than 1 (Case 4) do not satisfy ff = f. Drawing the endomaps for the remaining cases we find that out of the 27 endomaps only Case 0, Case 3, Case 12, Case 13, Case 18, Case 21, Case 22, Case 23, and Case 24 satisfy ff = f.

Splitting Identity Functions

Now we have to address the question of given two sets A and B, what are all the pairs f: A → B and g: B → A satisfying gf = 1A (identity function). Since an identity function is both onto and 1-1, the pair of functions can both be identity functions 1A1A = 1A. Alternatively, a wild guess, one of the pair could contribute 1-1 character and the other could contribute onto character such that the identity function can exhibit both 1-1 and onto characteristics.

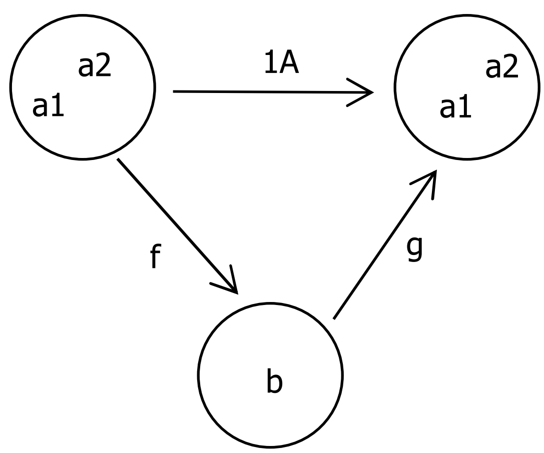
Taking |A| = |B| = 1 i.e. A = {a} and B = {b}. There is only one function f: A → B, with f(a) = b, and one function g: B → A with g(b) = a composite gf: A → A is given by g(f(a)) = g(b) = a. Thus the composite gf = 1A. Considering a bit more complicated case of A = {a1, a2} and B = {b}. Again there is only one function f: A → B, which assigns the only element b of B to both a1 and a2 of A, whereas there are two functions g1: B → A assigning b to a1 and g2: B → A assigning b to a2. In terms of internal diagrams:

Case 1. |A| = |B| = 1



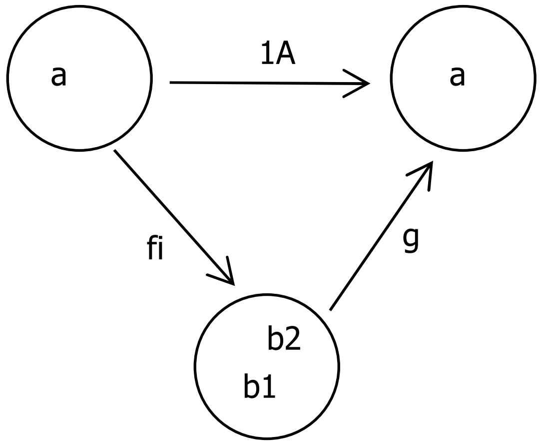
gf = 1A

Case 2. |A| = 2, |B| = 1



gf ≠ 1A

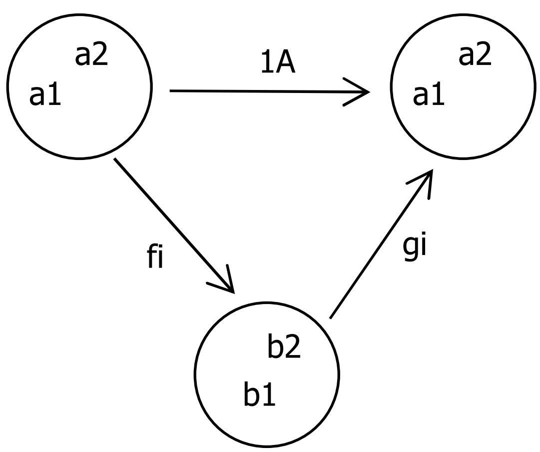
Case 3. |A| = 1, |B| = 2



gf1 = 1A
gf2 = 1A

In Case 3, f1: A → B assigns ‘a’ to ‘b1’, while f2: A → B assigns ‘a’ to ‘b2’. The two composites gf1 and gf2 are both equal to 1A.

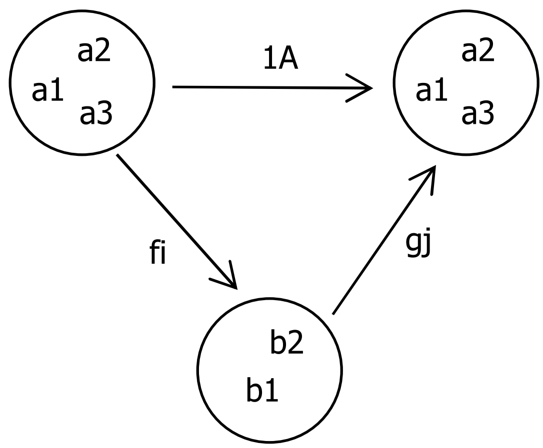
Case 4. |A| = 2, |B| = 2



g1f1 = 1A
g1f2 = 1A
g2f1 = 1A
g2f2 = 1A

In Case 4, there are two fi: A → B, i = 1, 2, 3, 4and two gi: B → A, i = 1, 2, 3, 4. Of the four fi's and gi's there are only two isomorphisms (onto and 1-1 function), which satisfy gjfj = 1A, j = 1, 2.

Case 5. |A| = 3, |B| = 2



There are 2^3 = 8 fi: A → B and 3^2 = 9 gj: A → B but there is no gjfi: A → A, which satisfies gf = 1A. Here we note that all fi's are not 1-1 (but some are onto), and that all gj's are not onto (but some are 1-1). This exercise suggests that in order for gf = 1A, g and f have to be isomorphisms or at least f must be 1-1 (denoted by m) and g must be onto (denoted by e). (In other words, gf can equal 1A if B is at least as big as A.) Thus we can say em = 1A. I am not sure how one would go about proving, but it sure would be interesting to see a proof of 1A = em. It would be interesting to compare and contrast with the factorization of a function into 1-1 followed by onto functions i.e. me = f.

Opposite: Intrinsic or Linguistic Artifact

Let's consider onto f: A → B and 1-1 g: B → A functions. The element-wise definition of onto function is as follows. f: A → B is onto if every 'b' of B is f(a) for some 'a' of A, while that of 1-1 function g: B → A is: if g(b1) = g(b2), then b1 = b2. From these definitions it is hard to see any opposition or duality between onto and 1-1 functions. However when we translate these definitions into arrow language we readily see the opposition or duality between onto and 1-1 functions.

f is onto if

$$A \xrightarrow{f} B \underset{k}{\overset{h}{\rightrightarrows}} C \quad \text{implies } h = k.$$

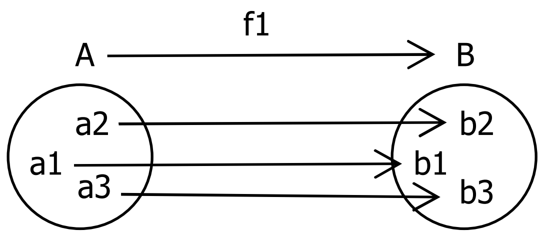
f is 1-1 if

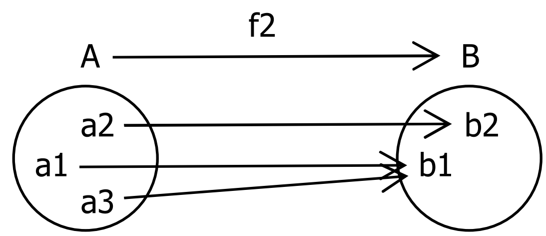
$$A \xleftarrow{f} B \underset{k}{\overset{h}{\leftleftarrows}} C$$

implies h = k.

However one can not help but wonder if the opposition of onto and 1-1 functions is intrinsic to the pair of onto and 1-1 functions or is it something gained in the translation into arrow-language. This question can be answered by drawing the corresponding internal diagrams.

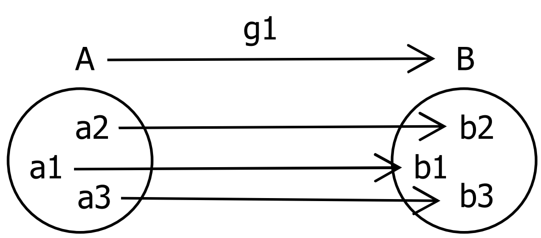
Onto functions

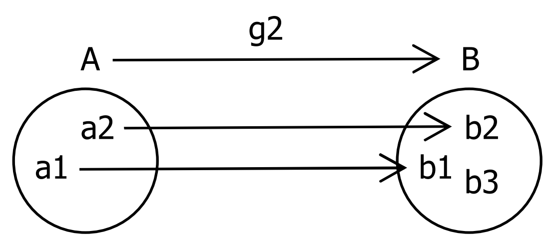




From the above two onto functions it is clear that if f: A → B is onto, then |A| ≥ |B|. Now let's look at 1-1 functions.

1-1 functions





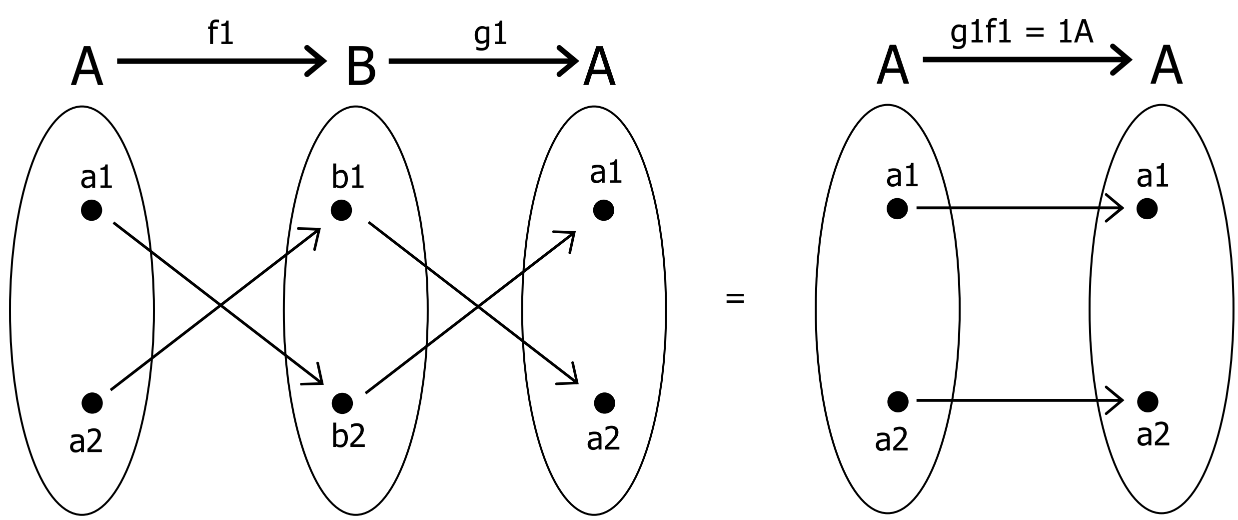
From the above two 1-1 functions it is clear that if g: A → B is 1-1, then |A| ≤ |B|. Looking at the above internal diagrams it is clear that onto and 1-1 functions are intrinsically opposite and the duality is not an artifact of articulation in a particular dialect. At this point, given that sum and product are dual (looking at the arrows), does it make sense to ask what exactly the intrinsic opposition between sum and product is.

Summing up, doing the exercises helped us understand [going from particulars to universals] beyond knowing the number of functions. We found a correspondence between numbers and functions, which lends itself to the suggestion that function can be thought of as a generalization of number or an abstract version of number. We also found that an identity function can be split into a 1-1 function followed by an onto function. In calculating the number of functions satisfying ff = f, we found that endomaps with cycles and fixed-points with path-length greater than 1 do not satisfy ff = f. Finally we realized that the opposition between onto and 1-1 functions is intrinsic to the functions and that the arrow language simply brings the duality into figural salience.

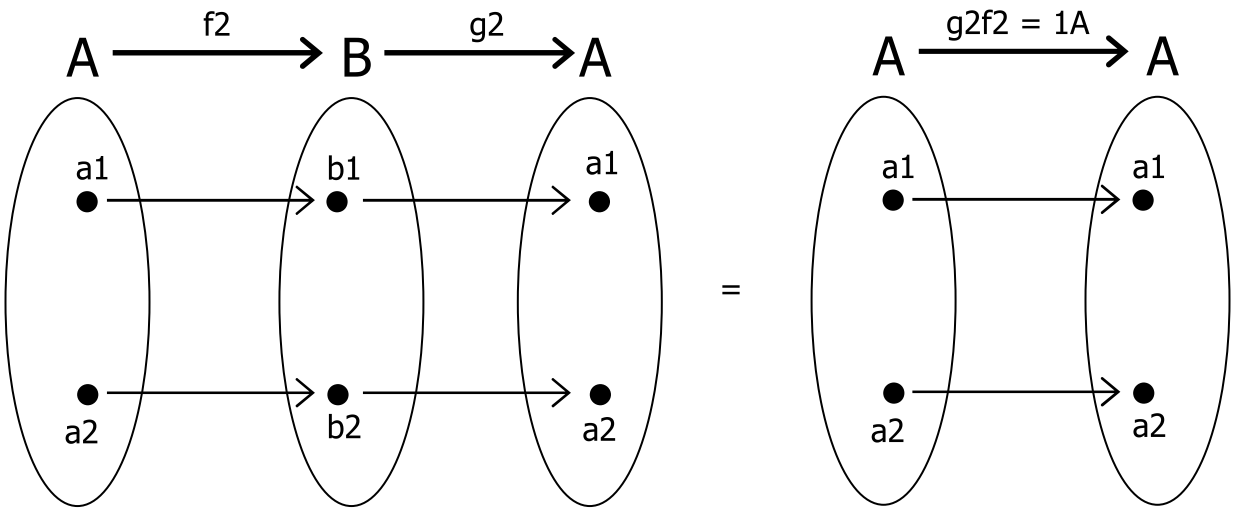
Composite of Functions and Identities

I made a mistake in the section 'Splitting Identity Functions' [more specifically Case 4] in the attachment 'UniversalsInParticulars' of my previous mail (Sub: Exercises & Understanding). I am really sorry about my carelessness. The problem was to find all pairs of maps A –f-> B –g-> A for which gf = 1A. Given |A| = 2 and |B| = 2, there are two pairs g1f1 and g2f2 which equal 1A as depicted below.

Case: g1f1

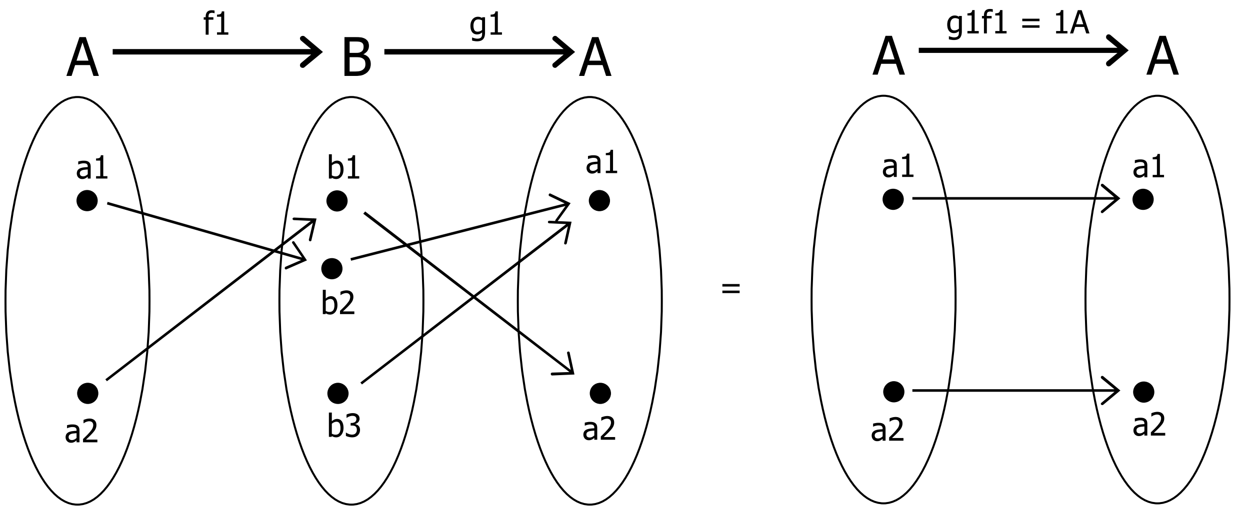


Case: g2f2

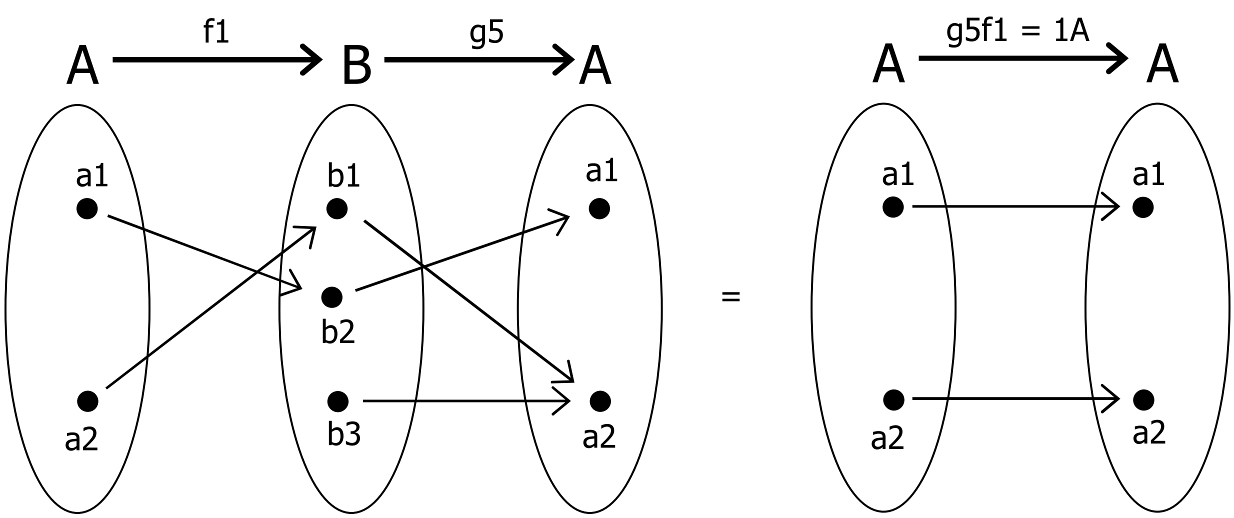


Now let's consider the case where |A| = 2 and |B| = 3. There are nine functions f: A → B of which six are 1-1 functions, and eight functions g: B → A of which six are onto. For each one of the six 1-1 functions f: A → B there are 2 onto functions g: B → A for which gf = 1A. There are 12 pairs of A –f-> B –g-> A which equal to 1A = A → A as depicted below.

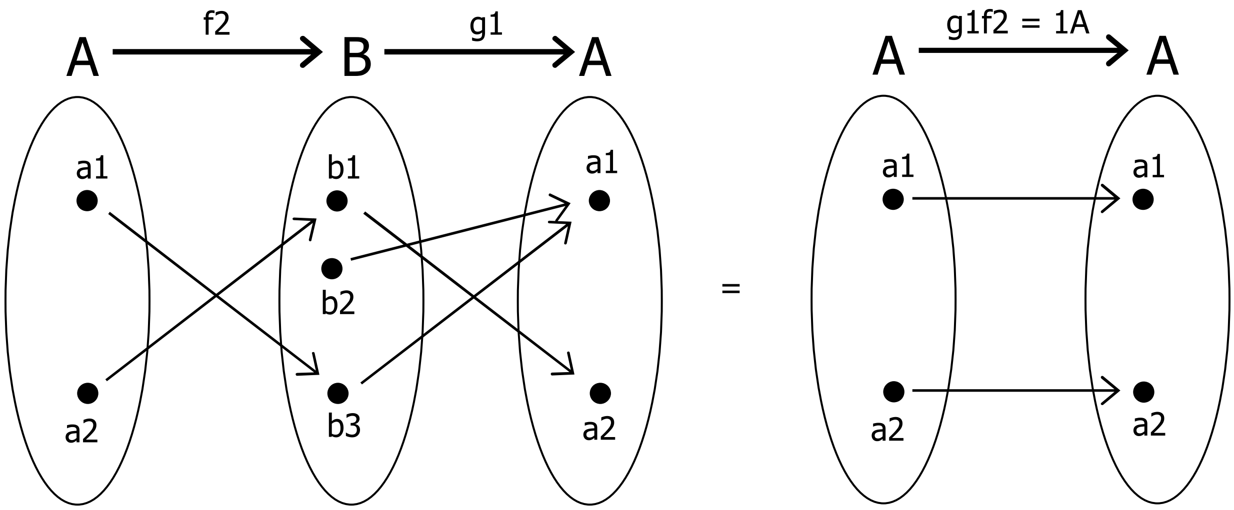
Case: 1a



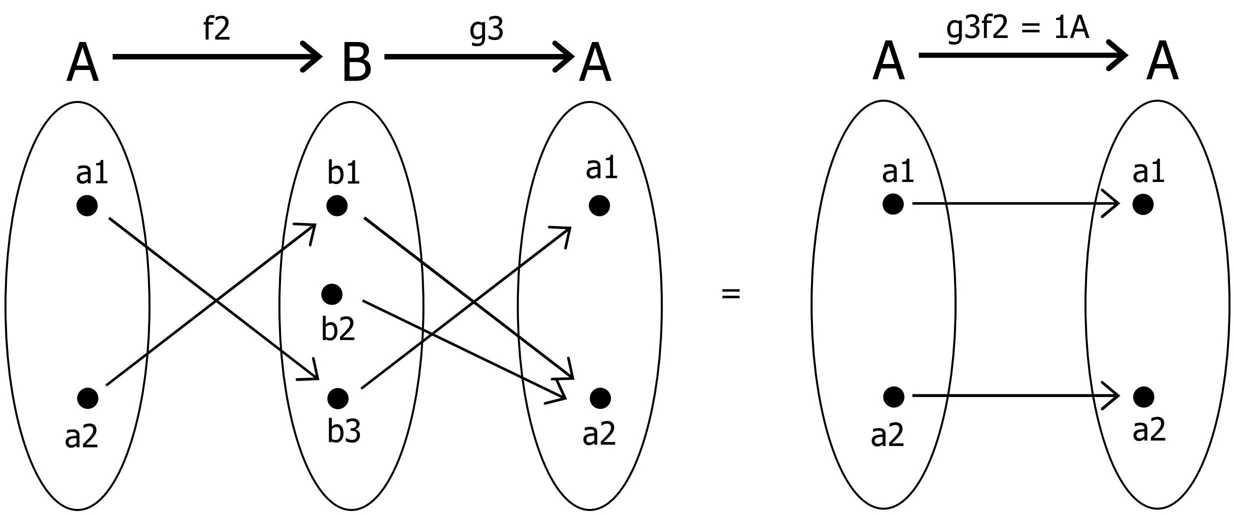
Case: 1b



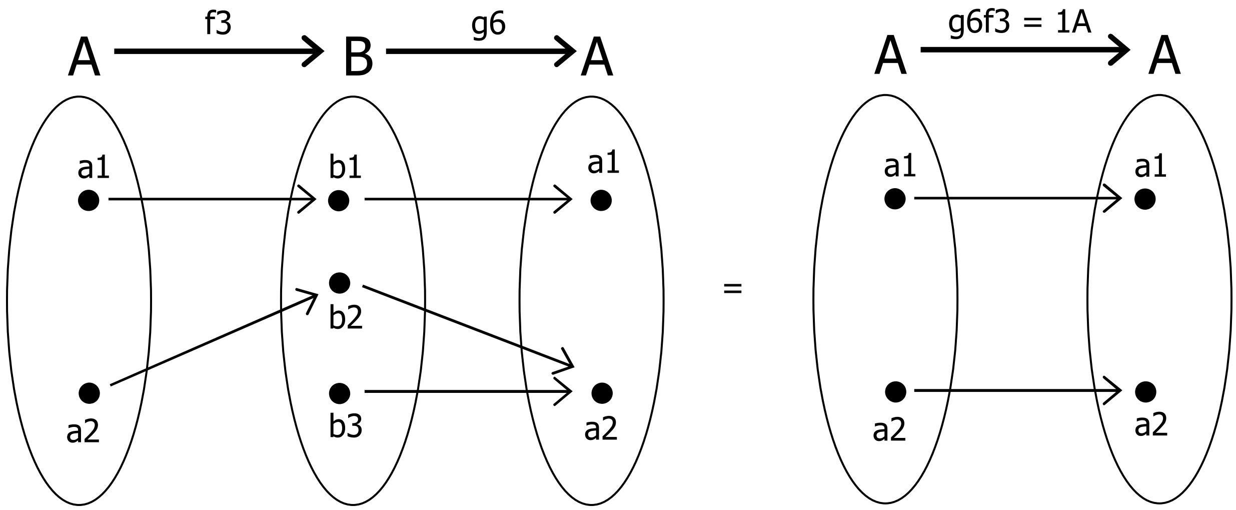
Case: 2a



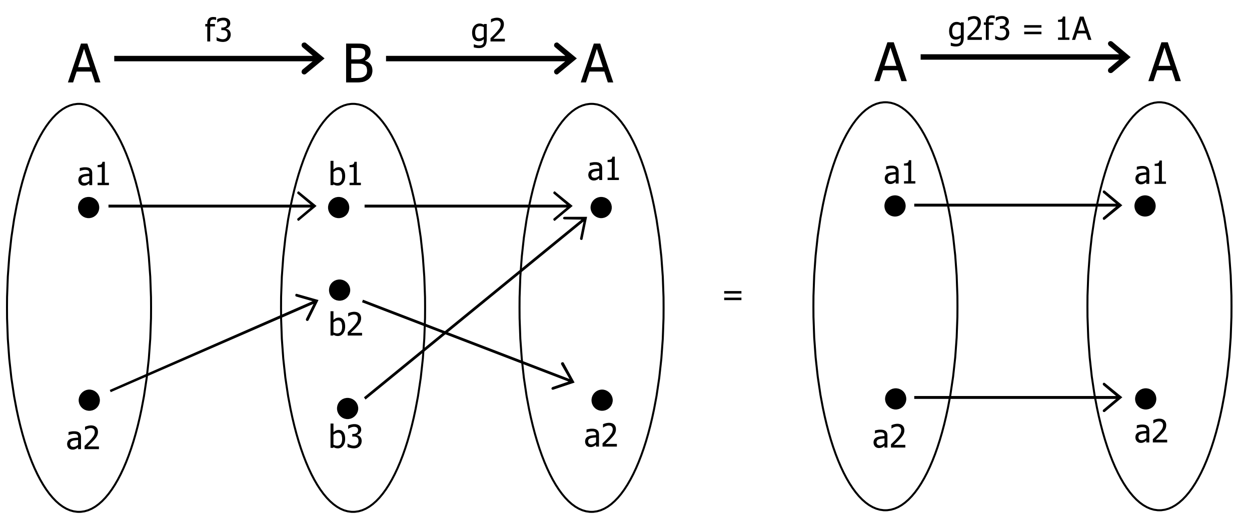
Case: 2b



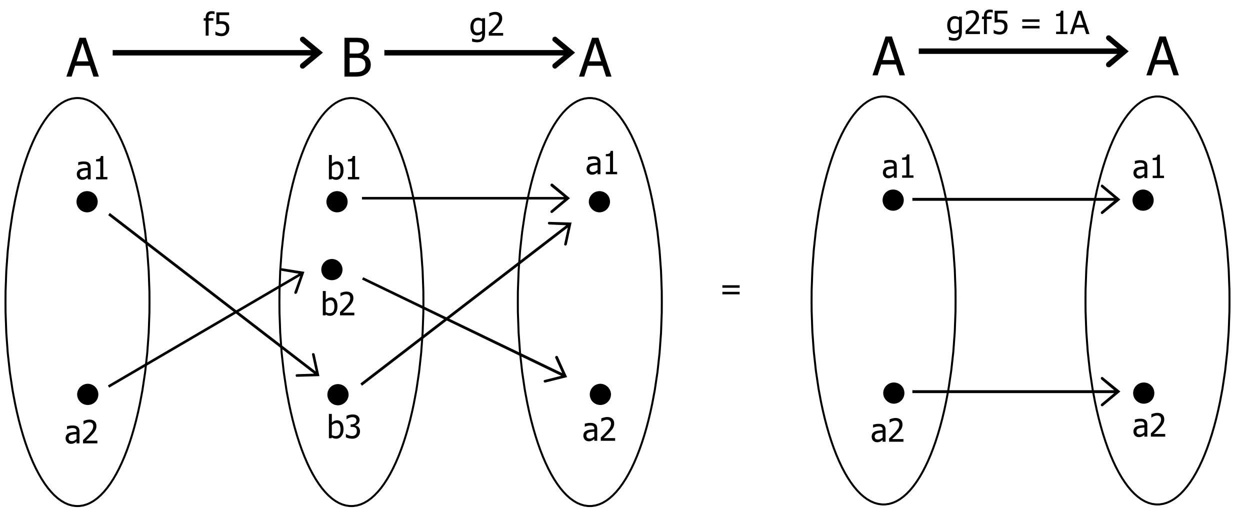
Case 3a



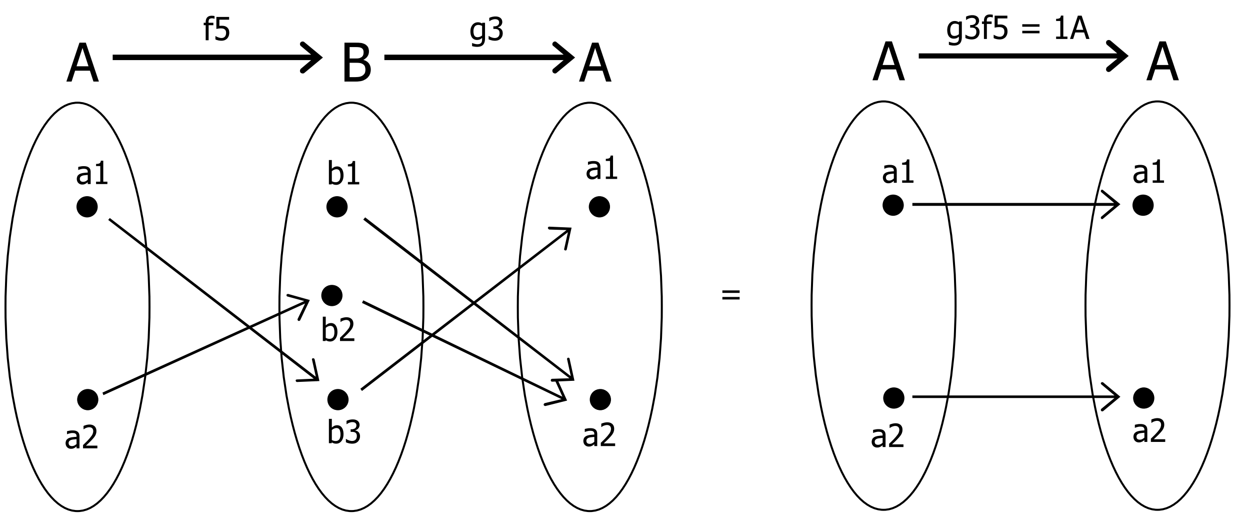
Case 3b



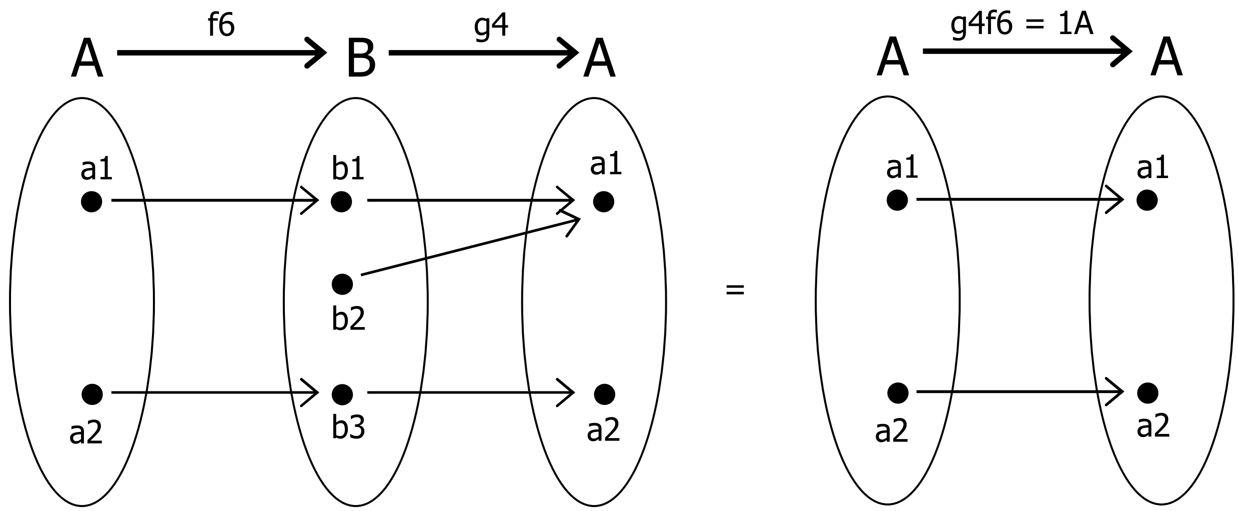
Case: 4a



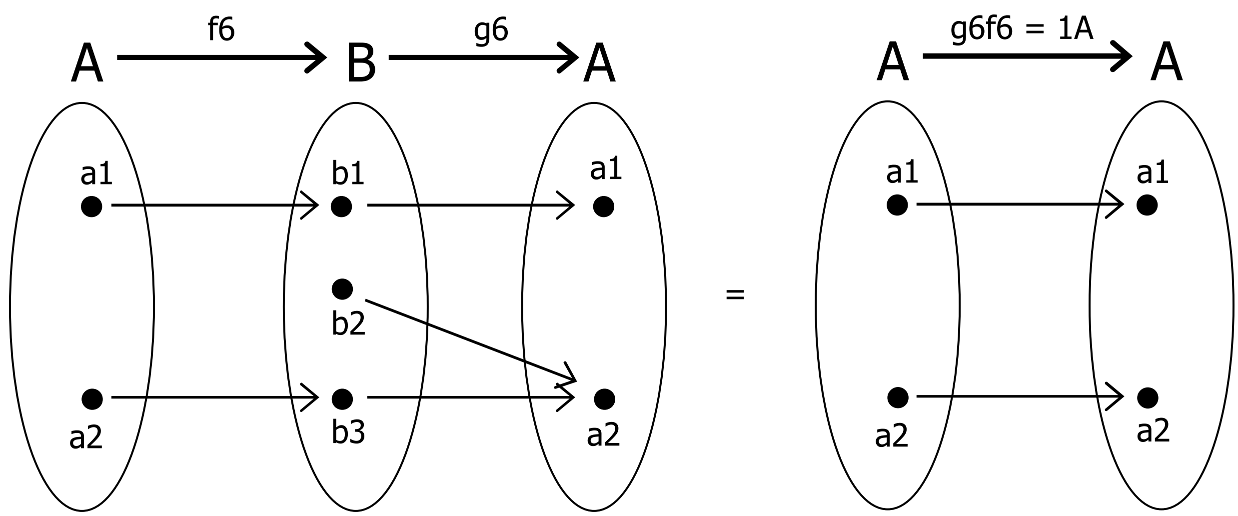
Case: 4b



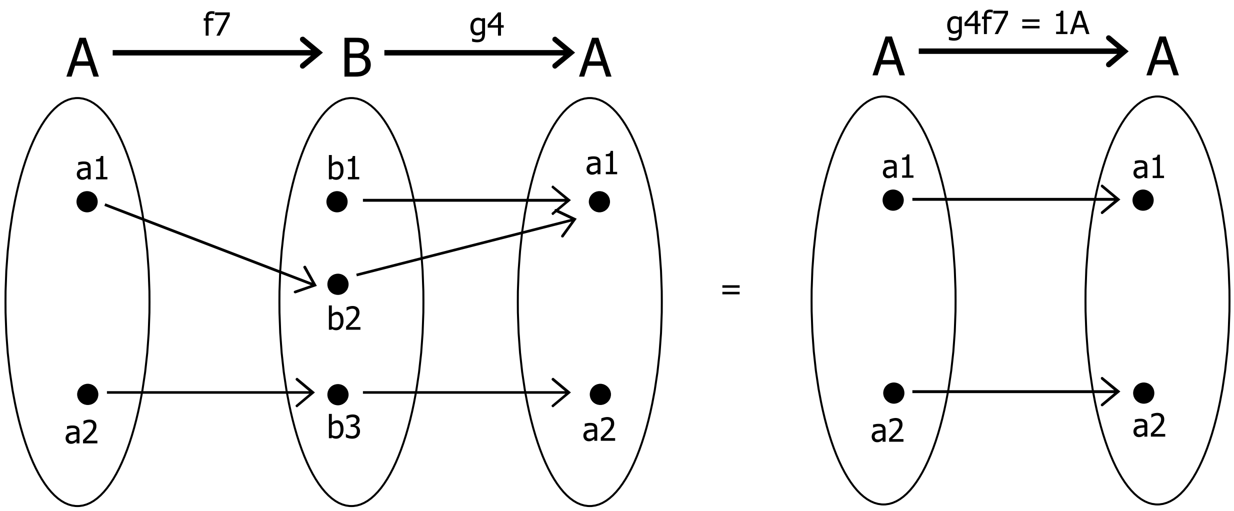
Case 5a



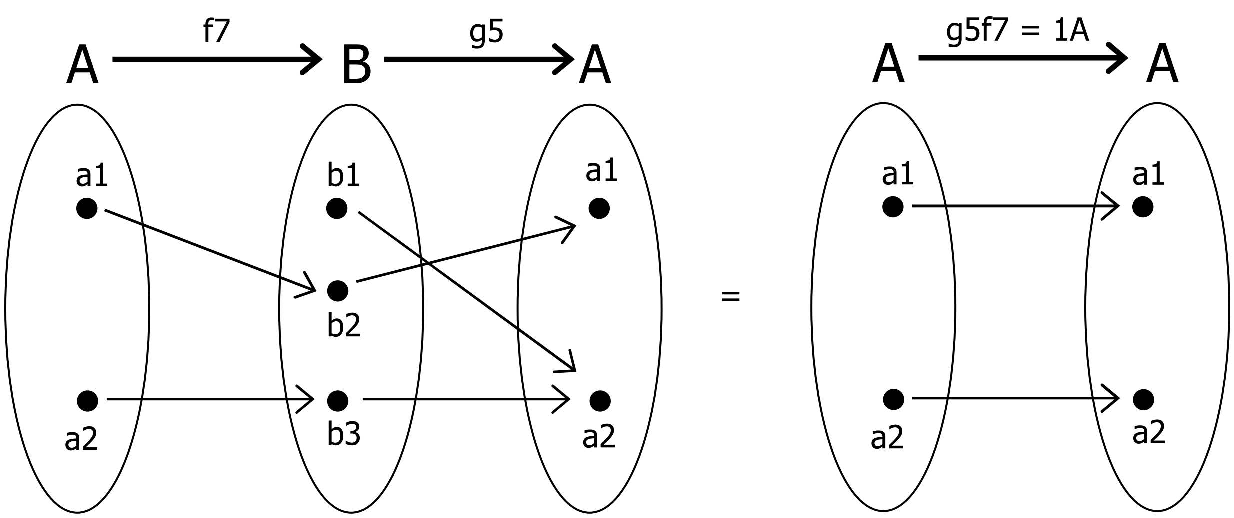
Case 5b



Case 6a



Case 6b



In all of the above cases wherein gf = 1A, f: A → B is 1-1 and g: B → A is onto, which makes intuitive sense: if the elements a1 and a2 of A are to be mapped to a1 and a2 respectively, then, to begin with, a1 and a2 should not be mapped to the same element i.e. f: A → B must be 1-1, and both a1 and a2 (all of A) should be values of g(f) i.e. g: B → A must be onto. Though now it seems "obvious" that if gf = 1A, then f: A → B must be 1-1 and g: B → A must be onto, there wasn't an inkling of this general property of identity maps when I began to do the exercises (assuming the generalization is correct). It would be interesting to see how this generalization relates to categories other than the category of functions.

P.S. In the above internal diagrams of maps, maps are labeled with numbers to the right of letters 'f' and 'g' such as f7, g4, etc. If you figure out the method behind the madness (numbering) that's attached to f and g to label maps, I'll buy you a Yellow Tail… oh no… I'll go broke… how about a glass of wine… now that I think about it... if man were given a choice between drunken stupor vs. radiant glow, I am sure mankind would have chosen radiant glow. So, instead of wine I'll get Shea butter straight from Africa to all those who crack the code behind the labeling of functions.

Concepts, Comprehension, Duality, and Composition

This week I'll briefly touch upon a bunch of stuff, which we will study in depth down the line, to give a feel for the breath of the subject. Let's begin. Given that our subject matter is Conceptual Mathematics, it is only natural that we address the question: "What is concept?" One simple answer, if you ask a philosopher, is 'concepts are what we think with' or 'concepts are the stuff of our thoughts'. This answer is not of much help. A little bit more, but not much more, informative answer can be obtained by contrasting concepts with intuition. Intuition is direct, unmediated, whereas a concept is apprehended only in terms of other concepts i.e. indirectly. Since we don't seem to be getting anywhere near a firm grasp on CONCEPT, let's take a slightly different approach. Let's, taking a cue from the practice of mathematics, define concept in terms of not what it is, but in terms of 'what it does.' Now the question we have to address is 'what does a concept do?' When we look at the R. C. James' image below:

we see black and white blobs of various shapes and sizes. Now if we bring the concept of DALMATIAN to bear on the image, our percept immediately transforms from the planar black and white blobs to a 3-dimensional percept of a Dalmatian with figural salience against a pastoral background. This example suggests a definition of concept: concept as a process that transforms one percept into another.

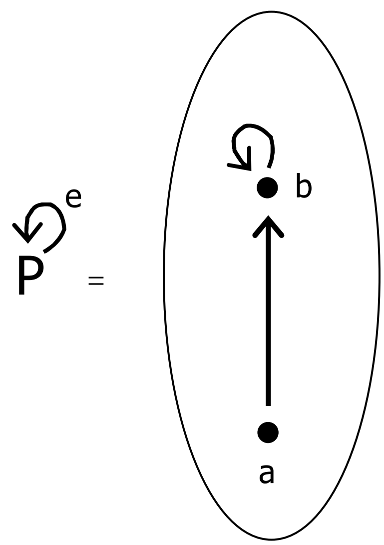
CONCEPT: Percept1 → Percept2

In other words, concepts enable us to see structures that are otherwise invisible. Unfortunately concepts, which are often presented as words, are rarely recognized for the instruments that they are—no different from a microscope or a telescope, to give couple of examples—enabling us to see "things" that are hidden in plain sight.

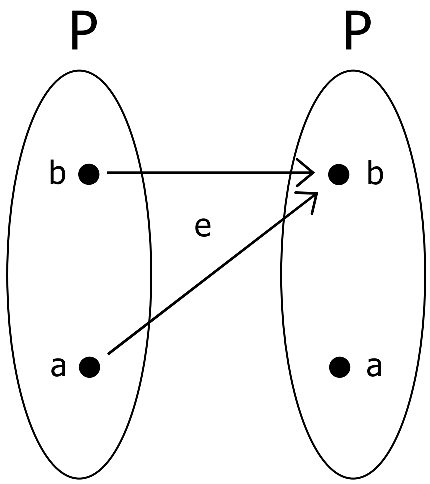
While we are around here, we might as well ask a related question: 'what is comprehension?' One way to approach understanding or comprehension is in terms of 'how we test' to find out whether someone has understood something, say, the concept of open-channel block in biophysics. Reproducing the textbook description in terms of forward and backward rate constants, binding sites, current, voltage, etc… is clearly not tantamount to understanding. One can claim to have understood if one can translate or present a given concept using terminology different from the one in which it was presented while preserving the structure of the concept. If one can express open-channel block as, for example, a traffic jam (i.e. traffic on a highway coming to a standstill when there is a herd of cows walking slowly in the midst of speeding cars) in addition to the formal terms (ionic flux decreases in the presence of a molecule unbinding from the ion channel pore at a very slow rate) in which it was presented to him or her, then he or she can claim to have understood the concept. One's comprehension of a concept can be "measured" in terms of how many different "dialects" one can re-present and how much of the structure is preserved is these transformations or translations. Here, there seems to be a dialectic: change vs.

invariance. One has to preserve the structure of the concept while varying the terminology (presentation) in which one expresses the structure. When teachers ask students to 'write in your own words,' it seems as though they are working with the above model of understanding.

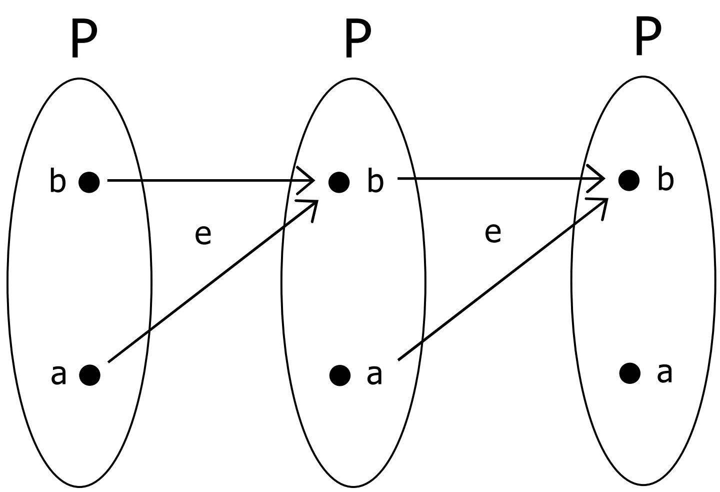
One could also express comprehension in terms of the properties of conceptualization that are preserved or reflected in understanding. For concreteness sake, let's model conception as an idempotent endomap e: P → P. First let's get a handle on idempotent endomap. An endomap e: P → P is called idempotent if e satisfies e*e = e, where * denotes composition, as shown below:



In familiar words, the above idempotent endomap can be interpreted as: particular 'a' (say, lily) is conceived in terms of a general 'b' (say, flower). The above idempotent endomap can be made more explicitly with domain P and codomain P of the idempotent endomap drawn separately:

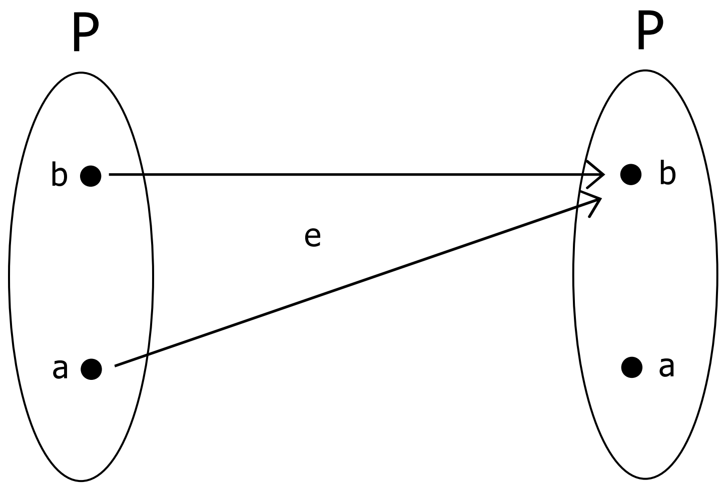


Now composing e: P → P with e: P → P

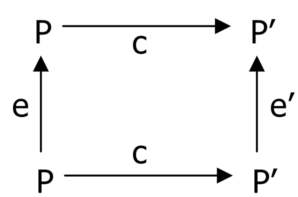


we get the composite e: P → P i.e. e*e = e as can be seen from the following

internal diagram:

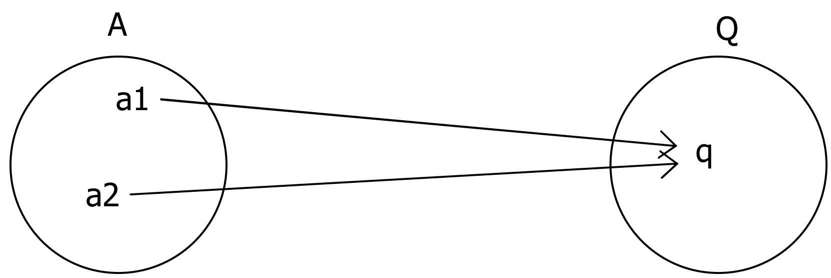


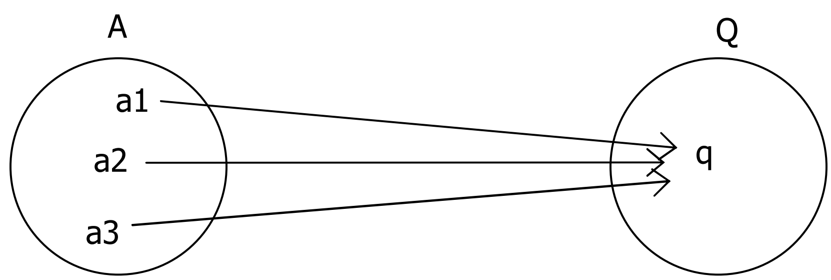
Going back to comprehension, we can model it as a map from a conception P to a conception P', both of which are modeled as idempotent endomaps. So, comprehension is given by c: e → e. Since e is an endomap, we can depict comprehension as follows:



Early on, in speaking of comprehension, we noted that comprehension c involves mirroring the structure of conception e in e'. The nature of the map c gives a "measure" of the quality of comprehension. If c were a mere function, then all of the properties of the conception may not be comprehended. If comprehension c is a structure-preserving map, i.e. if $c^*e = e'^*c$, then positive properties (presence of something such as cat has tail, or formally, fixed-point such as e (b) = b) are preserved. If c is reflective, then both positive and negative (absence of something such as cats don't have horns; 'a' is not a fixed-point, e (a) ≠ a) are reflected by comprehension c in conception e'. If comprehension c is an isomorphism, satisfying $c' \circ c = 1_P$ and $c \circ c' = 1_{P'}$, then we have a thorough understanding.

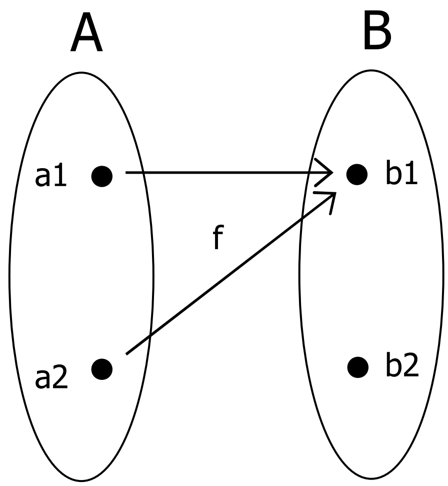
Speaking of understanding, the notion of duality is a particularly powerful tool to organize knowledge, which might otherwise seem disparate, into coherent structures that readily lend themselves to comprehension. So, let's look at a simple example of duality. Before we go on to find the dual of whatever we are trying to find, let's note one of the overarching theme or take-home message, in a somewhat rhetorical tone, of our subject of study: 'a thing is not the thing-in-itself, but is its relations to all the things that it is related.' In light of this understanding of "things," let's ask 'what is the dual or opposite of singleton or single-element set?' Taking the take-home message in all seriousness that it deserves, we ask 'how do we express singleton set in terms of its relations to all other sets?' Is there anything about the relations of singleton set to all sets worth noting or anything that lends to simple summing up? Fortunately, there is. There is only 1 function from any set to the singleton set. Now we can speak of singleton set without specifying that it has only 1 element (thing-in-itself), but by saying that it's a set (or object) to which there is only 1 function from any set (its relations). We illustrate couple of cases as shown below (reproduced from Week1):



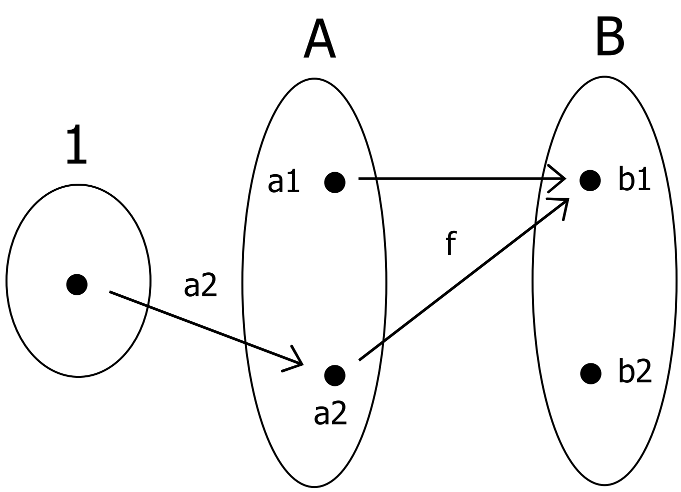


Now let's go back to our original question: 'what's the dual or opposite of singleton set (also known as terminal object)?' Since the terminal object is one to which there is only one map (another name for function), we can guess the opposite to be the initial object from which there is only one map to any other object (or set). The initial object happens to be empty set from which there is only one function to any set.

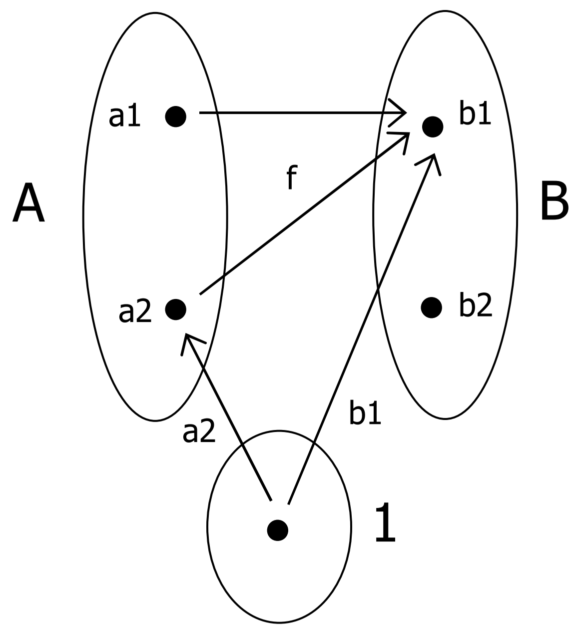
Lastly, and my favorite is to see 'evaluation of a function' as a special case of 'composition of functions'. Let's consider a function f: A → B, the internal diagram of which is depicted below:



Now if we look at the evaluation of the function f: A → B at a2 of its domain A, we find that f (a2) = b1 of its codomain B. Recollecting from Week1 that an element such as a2 of a set such as A can be identified with a function from singleton set to the set A i.e. a2: 1 → A, with a2 assigned to the single element of the singleton set. Diagrammatically,



To see that evaluation of a function is a special case of composition of functions, first note that the codomain of the function a2 is A, which is the same as the domain of f: A → B. So we can compose the two functions f and a2, and the composite is a function with the domain of the function a2 as domain and with the codomain of the function f as codomain. In terms of internal diagram,



Looking at the two paths: one from singleton set 1 to A and then to B, and the other from 1 to B, we find that f*a2 = b1, which is reminiscent of our familiar commutative diagram of composition of functions, except that here the domain of the first function and the domain of the composite function are one and the same singleton 1, whereas for composition in general the domain of the first is same as the domain of the composite, but it does not have to be singleton set 1.

Category of Endomaps

Rememorying Category of Maps

Before we look at the category of endomaps, please allow me to make a remark on the category of maps that we looked at in our previous note. First, what are the objects of category of maps? This is easy, especially given that categories are [often] named after their objects; maps are objects of the category of maps. Here's a map x: Z → Y i.e. an object in the category of maps.

$$Z \xrightarrow{x} Y$$

From the above external diagram we can readily note that an object of the category of maps has 2 component objects Z and Y and 1 component map x (of the category of sets). Let's place this small observation in our content-addressable memory and go ahead and look at maps in the category of maps. A map, whatever devil may lie in the details, has an object of the category as domain and an object as codomain of the map. So, given that the objects of the category of maps are maps such as x: Z → Y, a map in the category of maps has a map (of the category of sets) as domain and a map as codomain. More specifically we defined a map in the category of maps as a pair of maps <h, k>: f => g satisfying an equation kf = gh. To hasten to the point that I am trying to get at, a map in the category of maps has 2 component maps h and k and [constrained by] 1 equation. Now compare the definition of object and map in the category of maps (in terms of their components) and see if you notice anything interesting. Please don't hesitate to write to me if you don't. It's pure pleasure to perceive the obvious!

Now let's rewrite the definition of CATEGORY; rewriting can help in thinking with the text somewhat like Dreaming by the Book. As usual we'll go slow (to everyone one's very own peccadillo); rewriting A, B, C as X, Y, Z, which, to me, is good enough for now.

Definition of CATEGORY

A category consists of the data (with the corresponding notation):

(1) OBJECTS X, Y, Z, …

(2) MAPS p, q, r, …

(3) For each map p, one object X as DOMAIN of p and one object Y as CODOMAIN of p, with the notation p: X → Y.

(4) For each object X an IDENTITY MAP with domain X and codomain X, denoted as 1_X: X → X.

(5) For each pair of maps p: X → Y and q: Y → Z, a COMPOSITE MAP qp: X → Z. The composite map qp is only defined if the domain of q, which is object Y, is the codomain of p (object Y). The domain of qp (object X) is the domain of p (object X) and the codomain of qp (object Z) is the codomain of q (object Z).

The data of category satisfy the following 2 RULES:

(i) IDENTITY LAWS: If p: X → Y, then $1_Y p = p$ and $p1_X = p$.

(ii) ASSOCIATIVE LAW: If p: X → Y, q: Y → Z, and r: Z → W, then (rq)p = r(qp) = rqp.

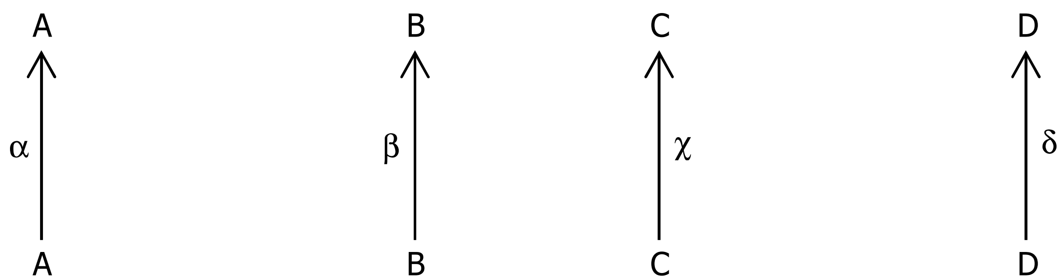
All ways of building composite by composition of pairs of maps give the same result irrespective of the number of maps the composite is composed of.

Now that we have the definition of CATEGORY in sight for ready reference, let's look at the category of endomaps.

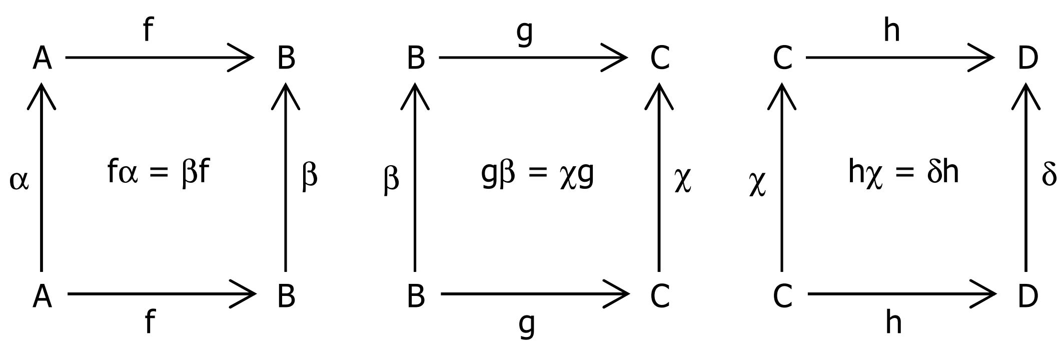
Definition of CATEGORY of ENDOMAPS

A category of endomaps consists of the data (with the corresponding notation):

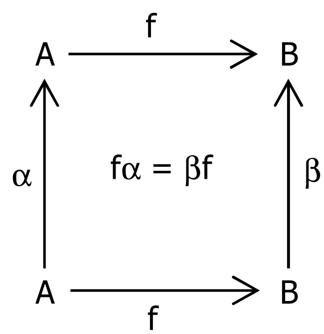
(1) OBJECTS $\alpha: A \to A$, $\beta: B \to B$, $\chi: C \to C$, $\delta: D \to D$, … are endomaps (of Category of Sets, with A, B, C, D, … objects of the category of sets: in other words A, B, C, D, … are sets) as shown below in terms of external diagrams.



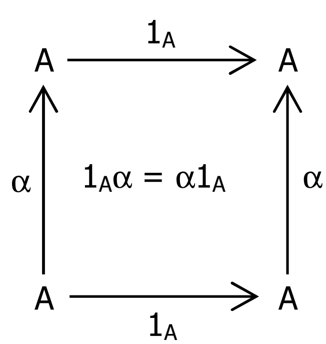
(2) MAPS <f, f>: (A, α) => (B, β), <g, g>: (B, β) => (C, χ), <h, h>: (C, χ) => (D, δ), … are commutative squares (of category of sets, with A, B, C, D, … objects of the category of sets and with α, f, β, g, χ, h, δ, … maps of the category of sets; in other words A, B, C, D, … are sets, and α, f, β, g, χ, h, δ, … are maps between sets) with the following notation.



(3) For each map <f, f>, one object α: A → A as DOMAIN of <f, f> and one object β: B → B as CODOMAIN of <f, f>, with the notation <f, f>: (A, α) => (B, β), and such that $f\alpha = \beta f$ expressing the commutativity of the following external diagram.



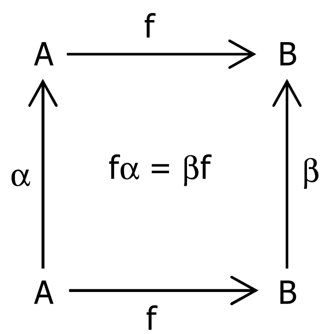
(4) For each object α: A → A, an IDENTITY MAP with domain α: A → A and codomain α: A → A with the notation <1_A, 1_A>: (A, α) => (A, α) denoting the commutative square depicted below.



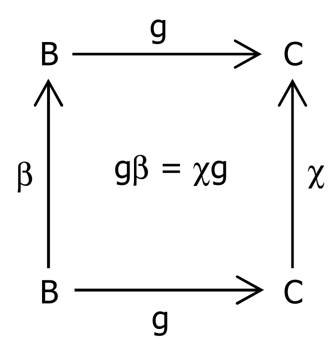
(5) For each pair of maps <f, f>: (A, α) => (B, β) and <g, g>: (B, β) => (C, χ), a COMPOSITE MAP <gf, gf>: (A, α) => (C, χ). The composite map <gf, gf> is only defined if the domain of the map <g, g>, which is object β: B → B, is same as the codomain of the map <f, f>

(object β: B → B). The domain of <gf, gf> (object α: A → A) is the domain of <f, f> (object α: A → A) and the codomain of <gf, gf> (object χ: C → C) is the codomain of <g, g> (object χ: C → C).

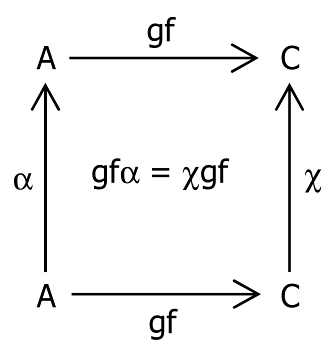
In terms of external diagrams, for each pair of commutative squares [maps <f, f>: (A, α) => (B, β) and <g, g>: (B, β) => (C, χ)]



and



a commutative composite square



Given two composable commutative squares satisfying $f\alpha = \beta f$ and $g\beta = \chi g$, we can see that the composite square is commutative i.e. $gf\alpha = \chi gf$.

$$gf\alpha = g\beta f = \chi gf$$

Just in case you are feeling little relieved that I am not taking you, for the n^{th} time, down the rabbit hole of meaning; sorry to disappoint you, but it's a trip we need to take every now and then. So what's the meaning of $gf\alpha$, $g\beta f$, and χgf in

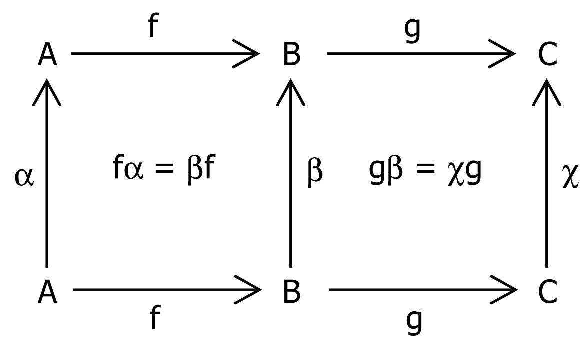
$$gf\alpha = g\beta f = \chi gf$$

First let's look at the composite of the two maps

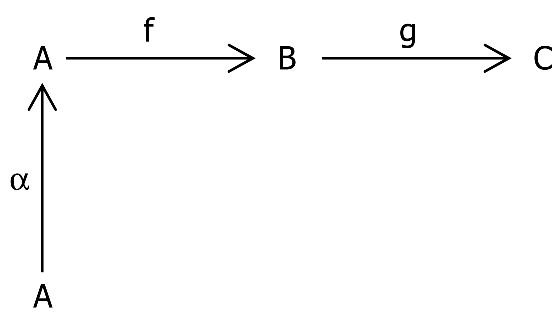
<f, f>: (A, α) => (B, β) and <g, g>: (B, β) => (C, χ)

Didn't we just look at it? Yes, but let's look at it again; there's an eternity in each moment. Here's the external diagram of the composite

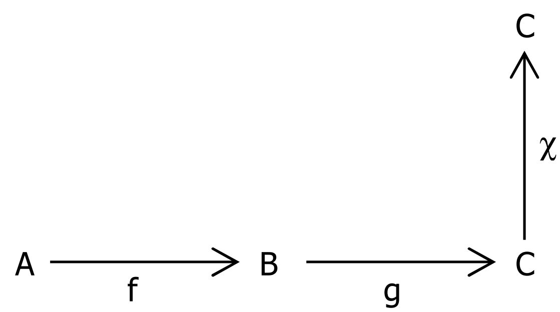
<gf, gf>: (A, α) => (C, χ)



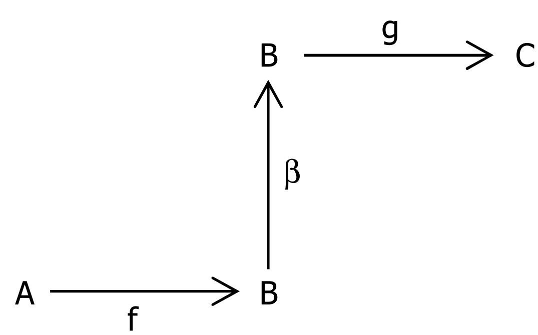
In showing that the big diagram made up of the two small diagrams with the common edge β: B → B is commutative, we are asserting that the path $gf\alpha$ from left-bottom A to top-right C:



is the same as the other path χgf from the very same A to the very same C:



by way of showing that each one of the above two paths is same as the middle-way (please note that this middle-way, notwithstanding all its significance, as far as I can see, is in no way related to the Middle Way of Nagarjuna) $g\beta f$:



I don't know about you but I do find this fascinating (I am not going to say absolutely fascinating because superlatives should be used sparsely, according to The Grammarian, for when not taken in moderation they tend to loose their superlativeness). All these years when I read that the composite of commutative diagrams is a commutative diagram, it didn't really occur to me to ask how much credit I should give to COMPOSITION for the COMMUTATIVITY of the composite of commutative diagrams. If any, I might have thought: 'of course, the composite is commutative; after all it's the composite of commutative diagrams.' Now after going back-and-forth between symbol substitution and arrow-tracing, I cannot help but credit COMPOSITION, which demands that there be a common edge β: B → B for the composite to be defined, with the credit that it deserves for the commutativity of the composite of commutative diagrams. Since I am getting a kick out of this, I cannot help but Praise the COMPOSITION with yet another example (a simpler one).

<u>Composition and Commutativity</u>

When I first came across, and even in the many subsequent reading and re-readings of, the, by now, all too familiar definition of COMPOSITION:

Given two maps

f: domain(f) → codomain(f)

and

g: domain(g) → codomain(g)

composite map (of maps f and g)

gf: domain(f) → codomain(g)

is defined if and only if the

domain(g) = codomain(f)

all I thought of it is that it's a natural thing to do, especially when one thinks of maps f and g as flights from one place to another and domain and codomain as places. If I want to go from the place where flight f originates i.e. domain(f) to the place where flight g ends i.e. codomain(g), then there's only one way this can happen, and that way is if the flight f ends where flight g originates or in a fancy equation:

domain(g) = codomain(f)

So as you can see it didn't seem like something to think about or even wonder what it might have in store: all of its umpteen consequences!

Now here's the simpler example I promised.

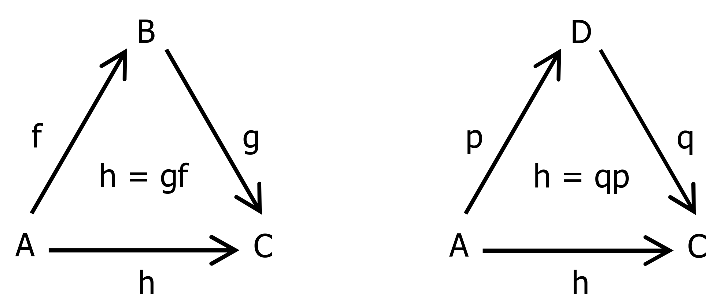
Consider two commutative triangles

h = gf with f: A → B, g: B → C, and h: A → C

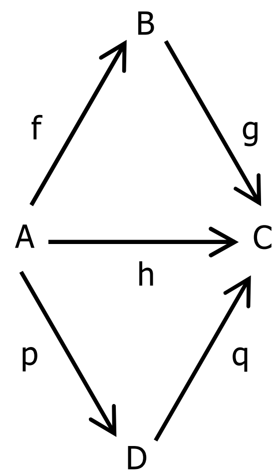
and

h = qp with p: A → D, q: D → C, and h: A → C

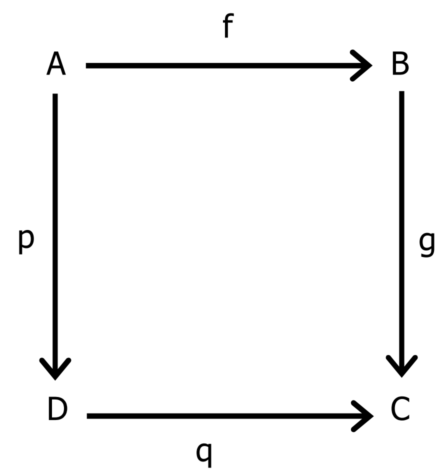
as shown below:



Can we compose these two commutative triangles? We can if they have a common edge (arrow/map). Since h: A → C is common to both commutative triangles, we can glue them together at the common edge as shown below:



We can turn the diagram around to look like a square that you would get when you put together two triangles as in:



Now that we have a square as the composite of two commutative triangles, we would like to know if the composite square is commutative i.e. we would like to know if the right-down path from A to C i.e. gf is equal to the down-right path from the same A to the same C i.e. qp. Is gf = qp? Since gf = h and qp = h, the common edge of the two triangles at which we fused the triangles to form the square,

$$gf = qp$$

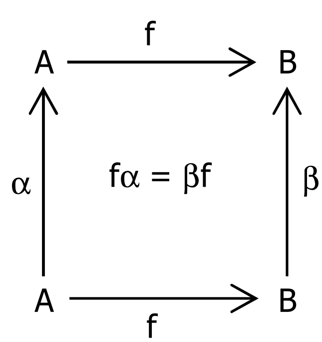
Here again we see that the composite of two commutative diagrams (triangles) is commutative by virtue of the definition of COMPOSITION, which demands that the things to be composed have something in common, which in this case, happens to be the common path $h: A \to C$.

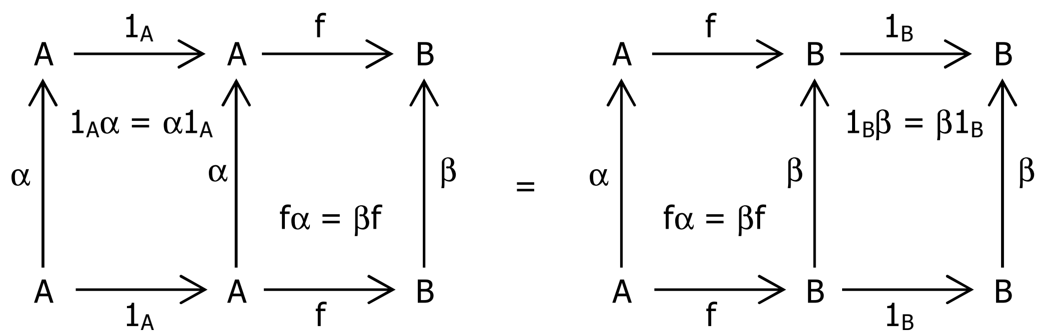
Now let's check the identity laws and associative law in the category of endomaps.

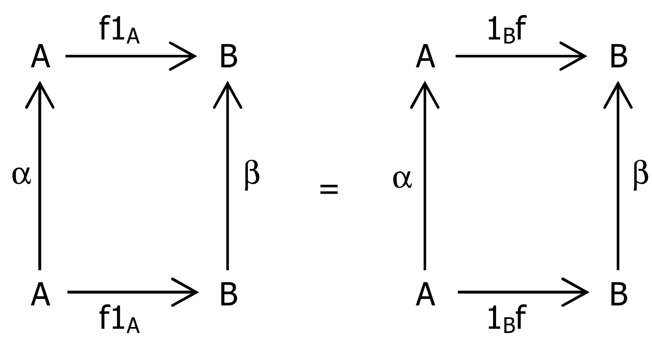
(i) IDENTITY LAWS: Given a map $f: A \to B$,

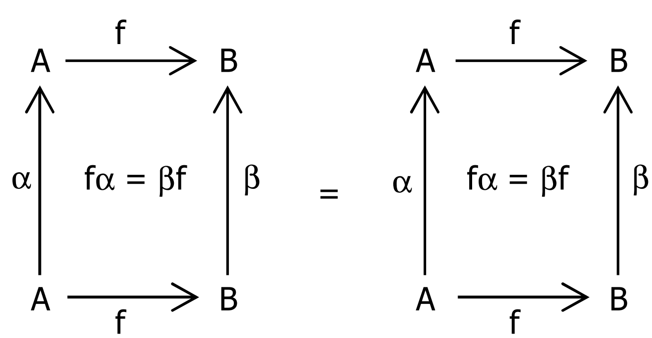
$$f1_A = 1_Bf = f$$

in any category. Given a map (commutative square) in the category of endomaps




A
1A
A
f
B
1Aα = α1A
α
α
β
fα = βf
A
1A
A
f
B
=
A
f
B
1B
B
α
fα = βf
β
1Bβ = β1B
β
A
f
B
1B
B

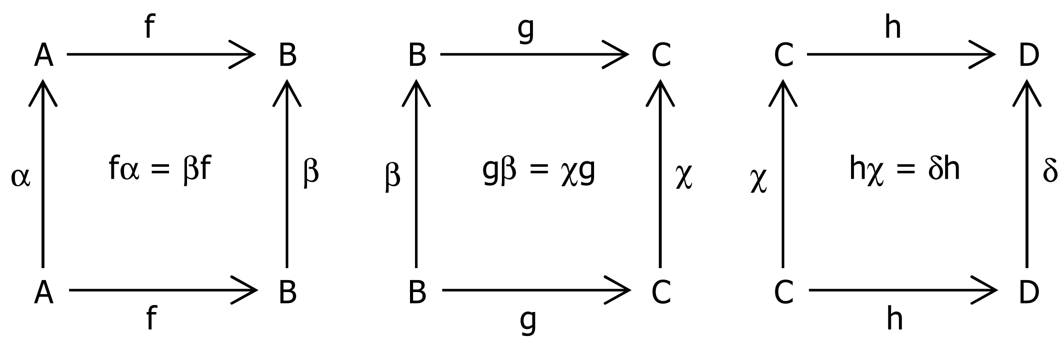

A
f1A
B
α
β
A
f1A
B
=
A
1Bf
B
α
β
A
1Bf
B

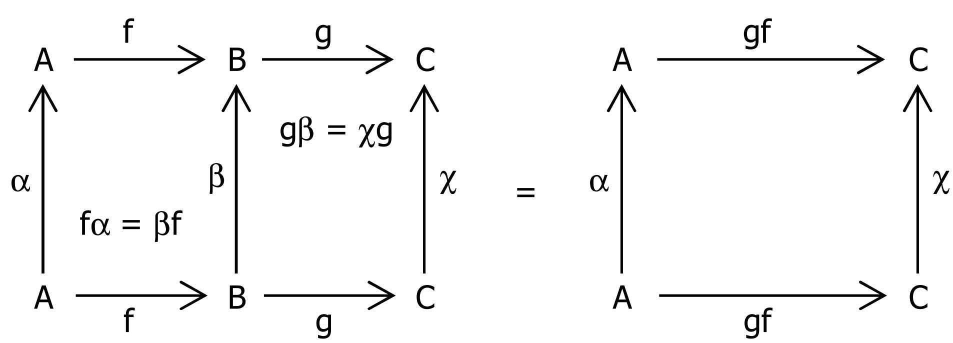

A
f
B
α
fα = βf
β
A
f
B
=
A
f
B
α
fα = βf
β
A
f
B

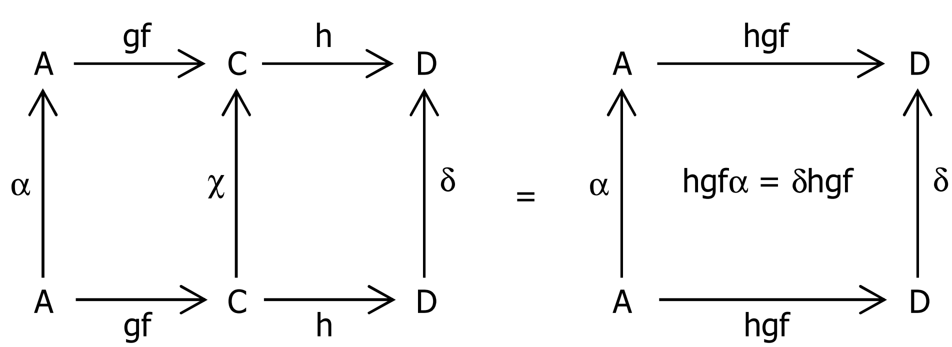
(ii) ASSOCIATIVE LAW: Given three maps f: A → B, g: B → C, and h: C → D,

$$h(gf) = (hg)f = hgf$$

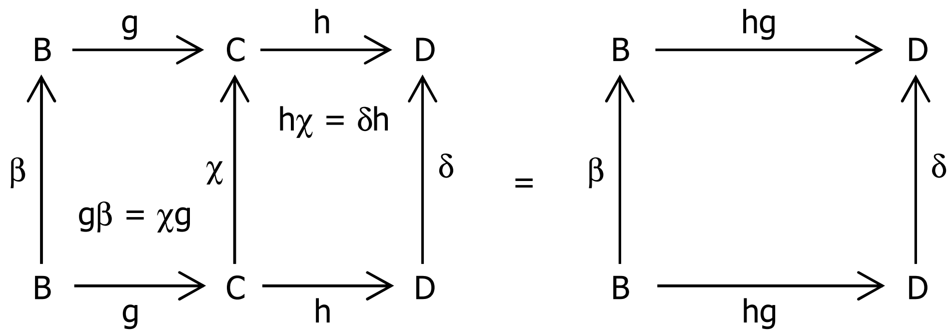
in any category. Given three maps (commutative squares) in the category of endomaps

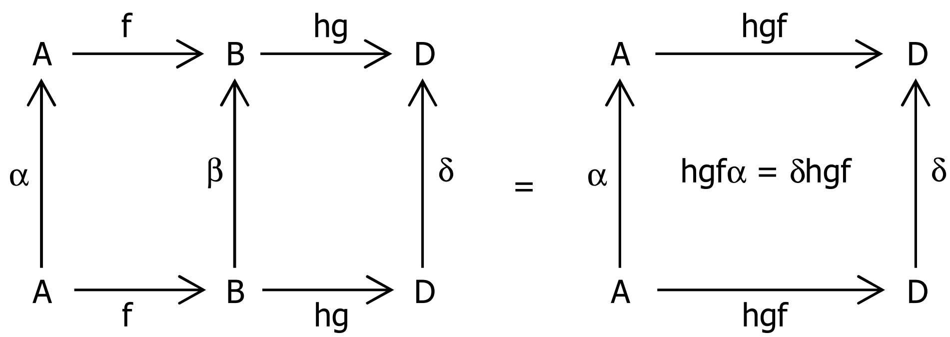






Similarly,





Voila! We have a category—category of endomaps.

Commutative Diagrams and Structure-Preserving Maps

Hidden in Plain Sight: Element

Oftentimes it's hard to see, leave alone state or assert, the obvious. Not so much so as to exemplify the general statement in the preceding sentence, but to warm up for the marathon exercise of seeing that which is all too clearly visible though not necessarily highlighted, let's begin with a function f: A → B and recollect that the function 'f' has (i). a domain set, A, (ii). a codomain set, B, and (iii). a rule:

for each element 'a' of set 'A'

there is exactly one element 'b' of set 'B'

such that 'b' is the value of the function 'f' at 'a'

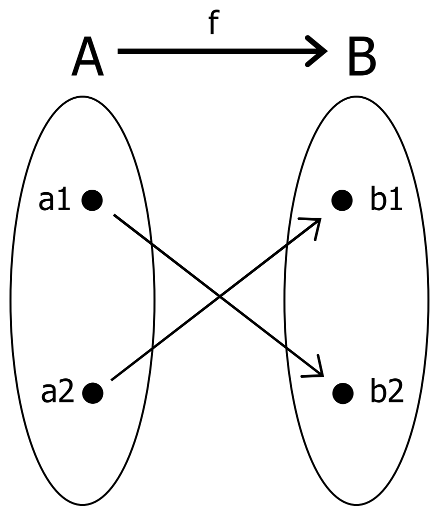
$$f(a) = b$$

Let's get little bit more concrete and consider the function 'f' for a given domain A = {a1, a2}, and codomain B = {b1, b2}, and the rule given by:

f(a1) = b2

f(a2) = b1

as illustrated in the following internal diagram:

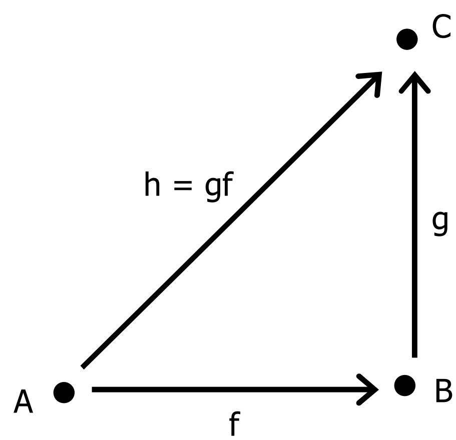


Now, we see that the function or process 'f' transforms a1 into b2, a2 into b1. We might also say that the operation 'f' takes a1 as input and gives b2 as output and when a2 is input gives b1 as output. What else is there to say about a function? All of the above descriptions of a function are views of a function from the perspective of 'change' as in when we say function is a process that transforms A into B. Yes, indeed it does change A into B. But is 'change' the only perspective to view a function from; how about looking from the other side or opposite perspective of 'invariance'. Is there something that the function, as it changes object A into object B, leaves alone or constant or doesn't change or preserves.

Let's look at the definition one more time: a function assigns [an] element to [each] element: element |→ element! The input to the function 'f' is a1, which is an element and the output f(a1) = b2 is also an element, so is input a2 and output f(a2) = b1. Thus we may say that the function which transforms one element into another preserves the element-hood of the elements of domain A in the elements of codomain B. Let's also note here what would constitute not preserving element-hood: if a "function" assigned lines to elements, then, given the continuous structure of line and the structure-less structure of points, we would have to say that the "function" does not preserve the structure-less element-hood or discreteness in the transformation. Going back to our main theme, we say that functions between sets preserve the structure of [structure-less distinct-and-yet-indistinguishable element-hood (to those of us who encountered redness of red and the like often enough element-hood may not seem odd, but to others this verbal construct may be little unsettling)] elements.

Composition of Functions and of Commutative Diagrams

Now, without making it anymore melodramatic than is pedagogically necessary, consider composition of two functions; f: A → B and g: B → C. First, we note that the composite gf: A → C is defined since the domain of the second function g is B, which is the same as the codomain B of the first function f, a necessary and sufficient condition to form a composite of two functions. The composite function gf has as domain the domain A of the first function f, and as codomain the codomain C of the second function g as depicted in the commutative diagram below:

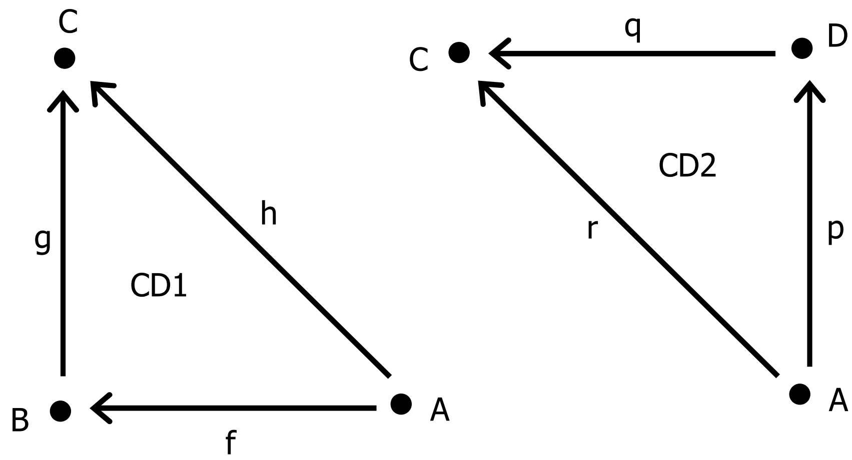


In saying that the above diagram commutes, we are asserting that evaluating function f followed by function g is same as evaluating the composite function h. In equations,

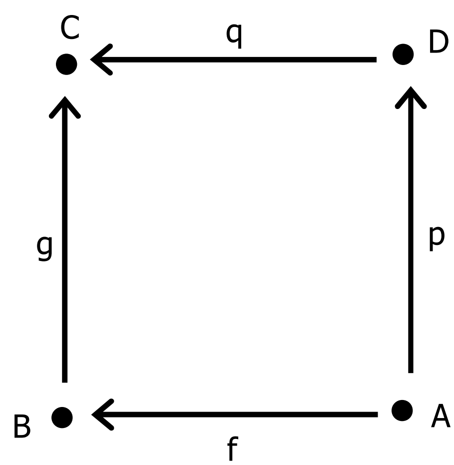
$$g(f(a)) = h(a) = c$$

In other words, the above equation says that transforming A directly by way of the process h into C is same as transforming A by way of f into B and then transforming B by way of g into C. Here also playing the same game we played with function and element in our heads silently, we note that composite of two functions is a function (paralleling the more familiar mating of people with people gives people as offspring). Composition of functions preserves function-hood. Lest we forget, we also found that function preserves element-hood.

Looking at the commutative diagram above we remember that just as we can form the composite of two functions we can also form the composite of two commutative diagrams if the necessary conditions are satisfied. Let's consider two commutative diagrams



with h = gf and r = qp. We can form the composite of the 2 commutative diagrams if h: A → C is equal to or coincides with r: A → C. Let's say it does i.e. h = r, so that we can form

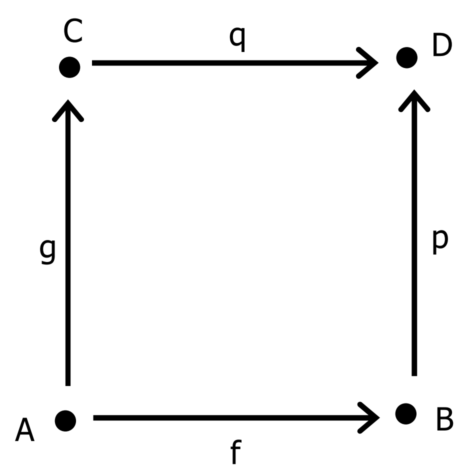


and we note that there are 2 paths from A to C: 1. from A to B to C and 2. from A to D to C. The first path is the composite 'gf' which is equal to 'h' which in turn is equal to 'r' which in a final turn is equal to 'qp'. gf = h = r = qp

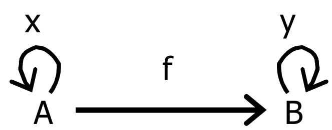
Thus we find that a big diagram (square) formed of smaller commutative diagrams (triangles) is also commutative, or the composite of commutative diagrams is commutative. Now to the obvious, composition preserves commutativity.

Understanding Commutative Diagrams

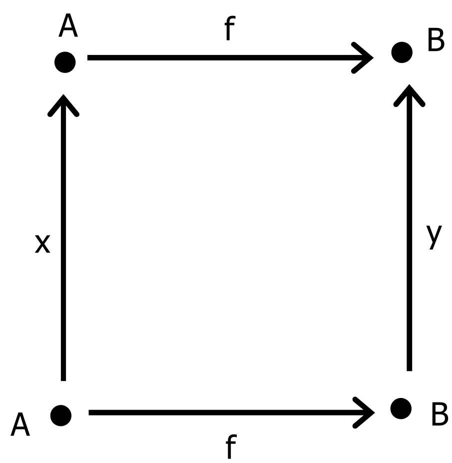
All of the above seeing the all too visible is but a warm-up for the marathon that starts now. What does commutativity mean? Is 'taking either one of the two paths gives the same result' all that there is to commutativity? Well, let's find out? Consider the following diagram



If, in the above diagram, pf = qg, then we say the diagram commutes. In order to get a better feel for what it means to say 'diagram commutes', let's consider a simple diagram made concrete by considering various sets as domains and codomains of the maps making the diagram.



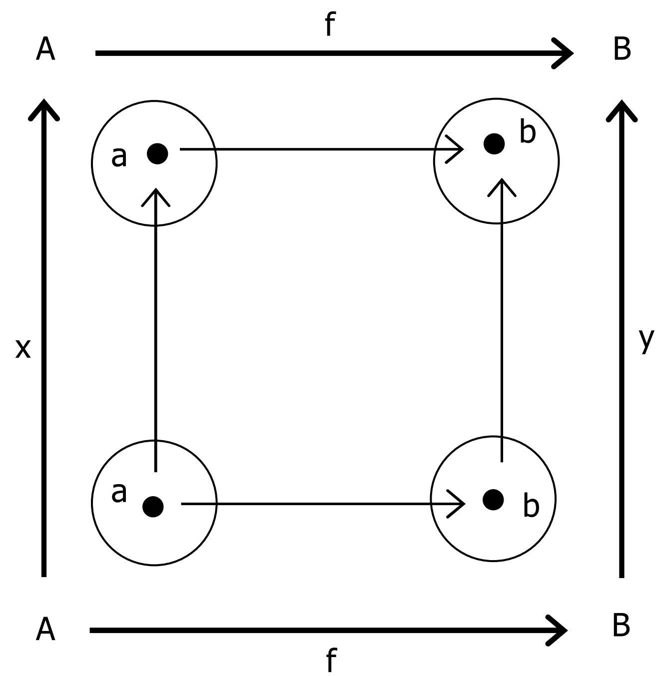
What we have here is two sets A and B and two endomaps x: A → A, y: B → B one on each set and a map from set A to set B, f: A → B. We can unfurl the above diagram into the square below:



If fx = yf, then the above diagram is said to commute. Now let's begin with singleton sets and go as far as we have to go until we find what it means to say fx = yf or to say the diagram commutes.

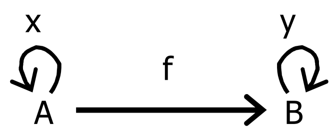
Many Composites to Evaluate

Let A = {a} and B = {b}. Given that both A and B are singleton sets, we have only one map x: A → A, the identity map, which assigns 'a' to 'a', and again only one map y: B → B, the identity map, which assigns 'b' to 'b' and one map from A to B, f: A → B, which assigns 'b' of B to 'a' of A. Shown below is the internal diagram inside the outer external diagram with thick arrows:

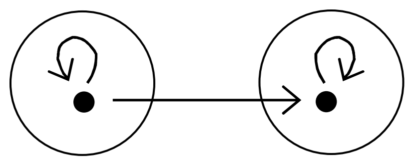


Looking at the above diagram we notice that there are two paths from bottom-left set A to top-right set B: (i). f: A → B after x: A → A, and (ii). y: B → B after f: A → B. The composites fx and yf give the same result i.e. taking either one of the two paths fx and yf takes to the same 'b' of B from 'a' of A. More explicitly, f(x(a)) = f(a) = b and y(f(a)) = y(b) = b; thus fx = yf. In other words, the above diagram, depicted below in a different format, commutes.

The above external diagram reproduced:



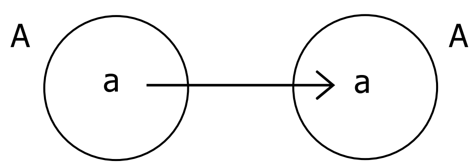
D.0 commutes i.e. fx = yf



Let's make a mental note of the above internal diagram and label it commutes, and before rushing to judgment as to what commutativity means with one simple internal diagram (above) in sight, let's wait and consider little bit more less-simple cases.

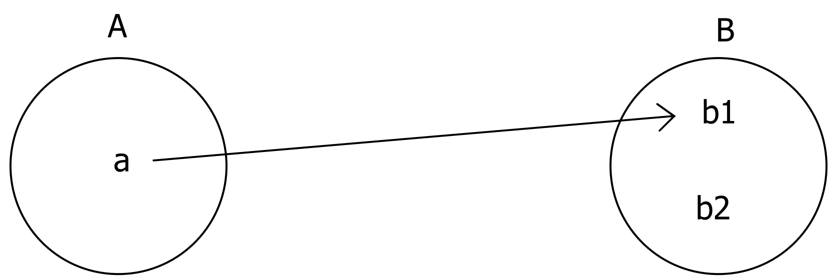
Let's consider A = {a} and B = {b1, b2}. First let's count the numbers of functions: (i). there is one function x: A → A, (ii). there are 2 functions fi: A → B, where i = 1, 2, and (iii). there are 4 functions yj: B → B, where j = 1, 2, 3, 4.

First let's look at x: A → A

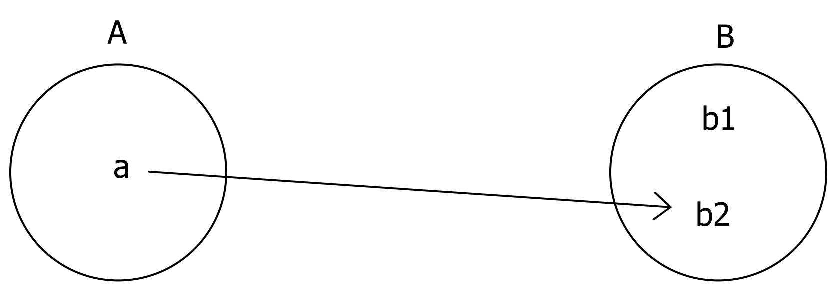


Next let's look at fi: A → B, i = 1, 2

f1: A → B

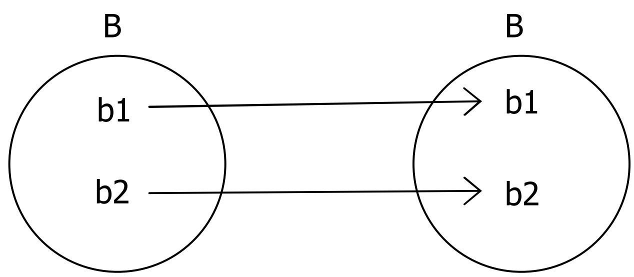


f2: A → B

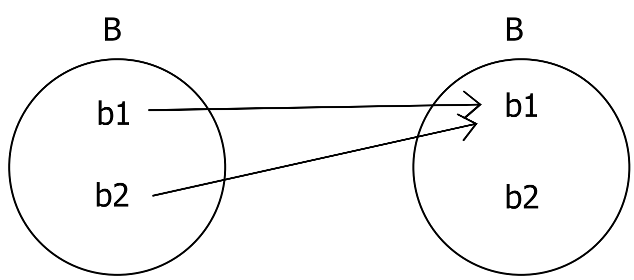


Now let's look at the endomaps on B, yj: B → B, j = 1, 2, 3, 4

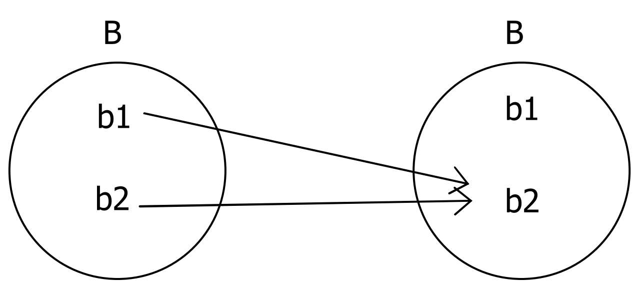
y1: B → B



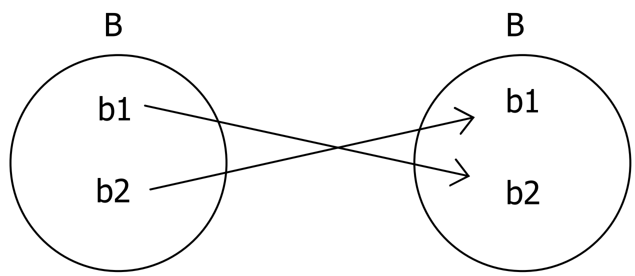
y2: B → B



y3: B → B

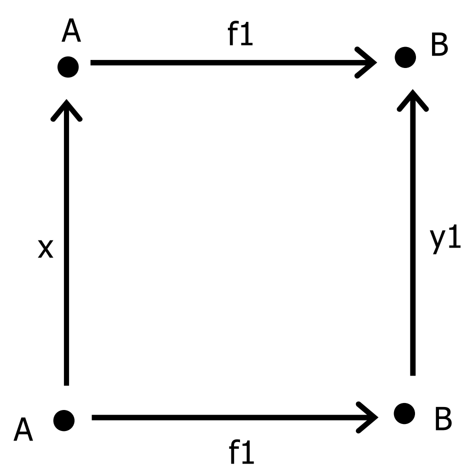


y4: B → B



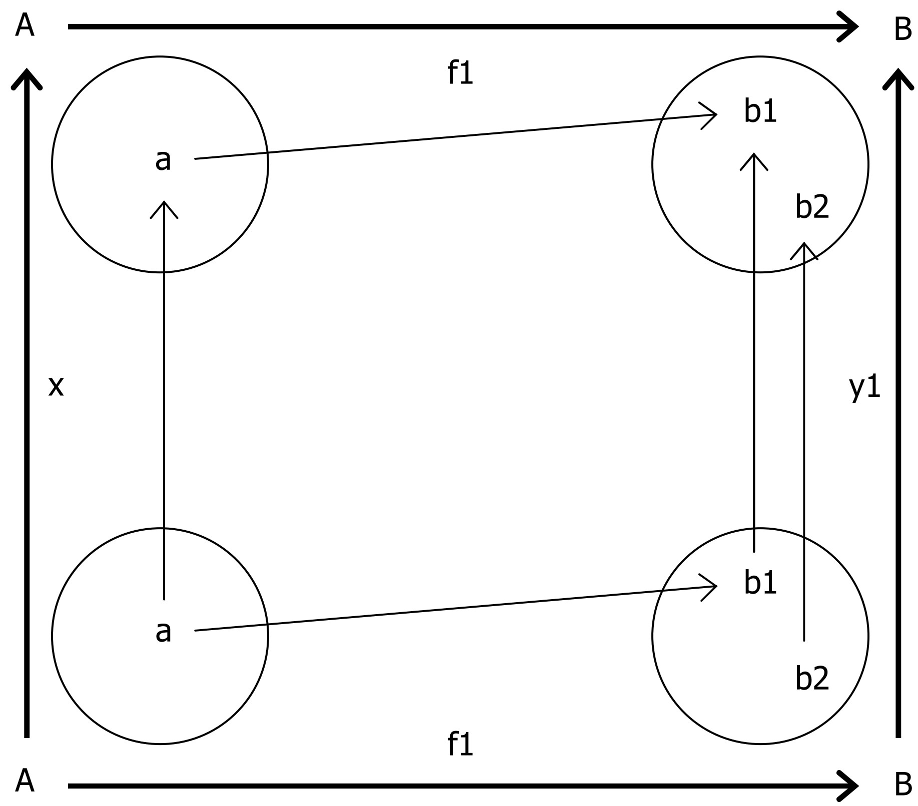
Lest we get lost in the Amazon of arrows, let's first enumerate all possible diagrams we have with A = {a} and B = {b1, b2}.

D.1.1 (the first index is for fi: A → B and the second is for yj: B → B)



First we want to see if f1x = y1f1 and label the corresponding internal diagram accordingly i.e. commutes (doesn't commute if f1x ≠ y1f1). For each pair of endomaps (x, yj; j = 1, 2, 3, 4) we examine whether fix = yjfi i.e. whether the diagram commutes as we switch from f1 to f2. Thus we have 4 pairs: (D.1.1

and D.2.1), (D.1.2 and D.2.2), (D.1.3 and D.2.3), and (D.1.4 and D.2.4) to examine.



From the above, noting that f1: A → B, x: A → A, and y1: B → B, we find that

f1(x(a)) = f1(a) = b1

y1(f1(a)) = y1(b1) = b1

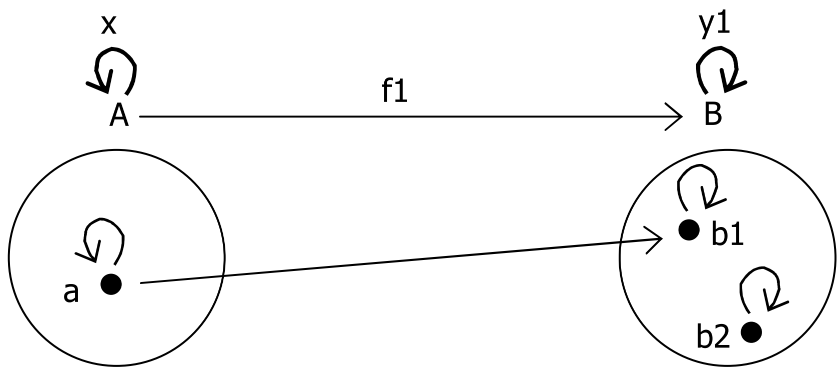
Since the functions f1x and y1f take equal values (b1) for equal arguments (a), they are equal:

f1x = y1f1

Thus the diagram D1.1 commutes

Since x: A → A and yj: B → B are endomaps, we can redraw the above internal diagram (along with external diagram) in a more condensed form:

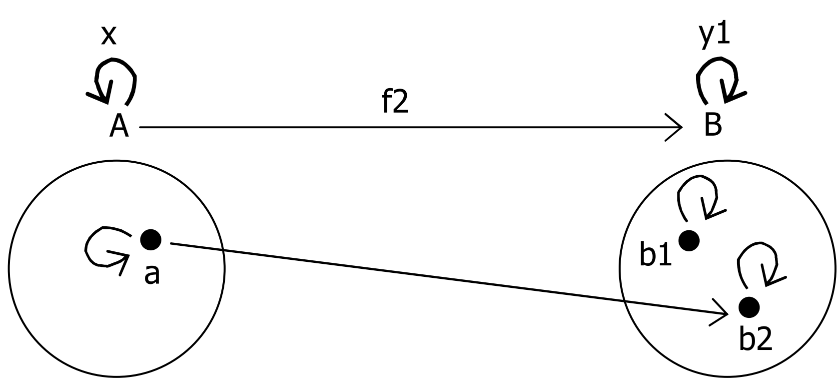
<u>D.1.1 commutes i.e. f1x = y1f1</u>



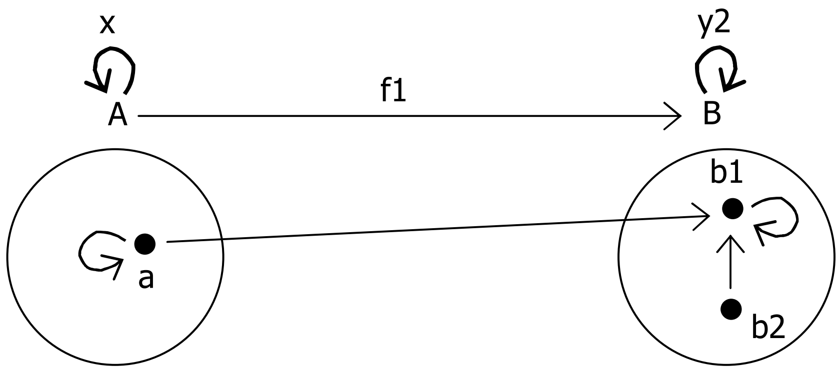
D.2.1 which differs from D1.1 simply in terms of f2: A → B, which assigns 'b2' of B to 'a' of A instead of 'b1' like f1: A → B above.

<u>D.2.1 commutes i.e. f2x = y1f2</u>

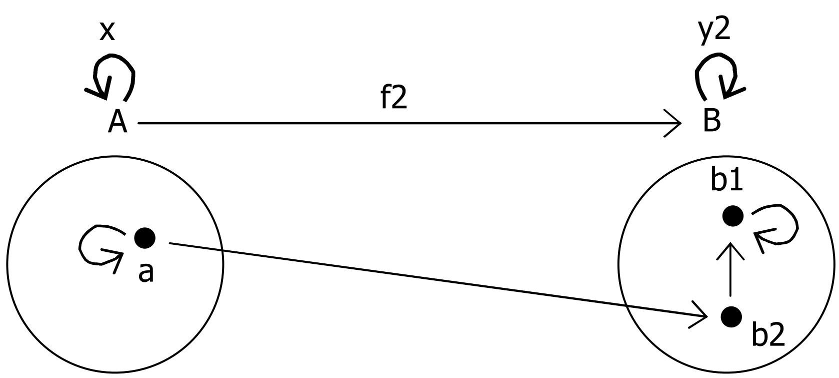
(From now onwards I'll draw diagrams as above and simply state commutes or not commutes; please feel free to ask me to show explicitly in the class.)



D.1.2 commutes i.e. f1x = y2f1



D.2.2 doesn't commute i.e. f2x ≠ y2f2



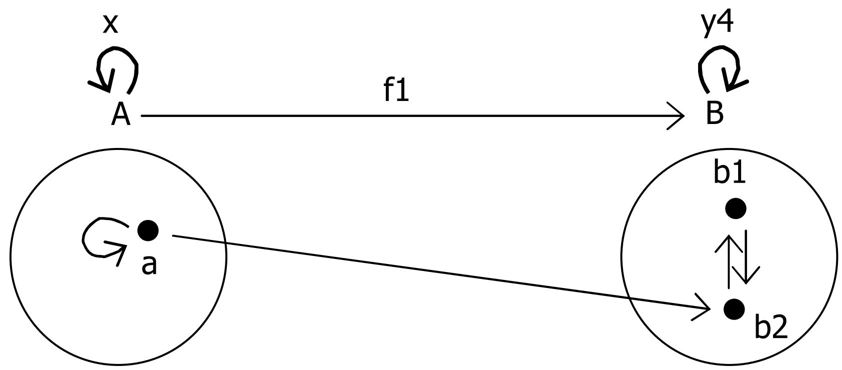
Now, looking at the above two diagrams, we notice that assigning 'b2' instead of 'b1' of B to 'a' of A changed the equation from commutes to doesn't commute. Let's now figure out by comparing the two diagrams what commutativity of a diagram tells us. First let's begin with noting that 'a' is both source and target of 'x'. Similarly, 'b1' is both source and target of 'y2', whereas 'b2' is only a source (and not target). Thus we can say, at least in the above two cases, diagram commutes when properties such as 'a' is both source and target are preserved by way of being assigned to something ('b1'), which is also both source and target. We also notice that this holds true of all three cases preceding the above two cases in that the diagrams commute and the property of being both source and target is preserved. Now let's see some more examples to see if there's more to the story. First, let's formulate some questions, so that we don't fail to recognize answers when we run into them. Based on the above, we found that a diagram commutes when a dot which is both source and target of an arrow is assigned to a dot which is both source and target of an arrow, and does not commute if a dot which is both source and target of an arrow is assigned to a dot which is merely a source of an arrow. Now to questions: (i) Does a diagram commute if a dot which is both source and target is assigned to a dot which is merely a source (or target) of an arrow? (ii). Does a diagram commute if a dot which is both source and target of an arrow is assigned to a dot which is source of one arrow and target of another arrow? (iii). Does a diagram commute if a dot which is source (or target) is assigned to a dot which is both source and target? (iv). Does a diagram commute if a dot which is source (or target) is assigned to a dot which is target (or source)? Let's find out.

We already have the answer to question (i) in diagram D.2.2, which is 'does not commute.'

D.1.3 does not commute (diagrammatically similar to D.2.2 above)

D.2.3 commutes (diagrammatically similar to D.1.2 above)

D.1.4 does not commute f1x ≠ y4f1



D.2.4 does not commute f2x ≠ y4f2 (diagrammatically similar to D.1.4 above)

Diagrams D.1.4. and D.2.4 answer one of the questions we wrote down earlier i.e. (ii). Does a diagram commute if a dot which is both source and target of an arrow is assigned to a dot which is source of one arrow and target of another arrow? No, the diagram doesn't commute if a dot which is both source and target of an arrow is assigned to a dot which is source of one arrow and target of another arrow as in the above diagram.

Now let's look at diagrams wherein A = {a1, a2} and B = {b}. There are 4 endomaps xi: A → B, i = 1, 2, 3, 4 on A (and are diagrammatically similar to yi above), while there is only one endomap on B. There's also only one f: A → B assigning every element 'a1' and 'a2' of domain A to the only element 'b' of codomain B.

Diagram D.1 commutes fx1 = yf (it is similar to D.1.1 and D.2.1 above in that loops are mapped to loops)

Diagram D.2 commutes fx2 = yf

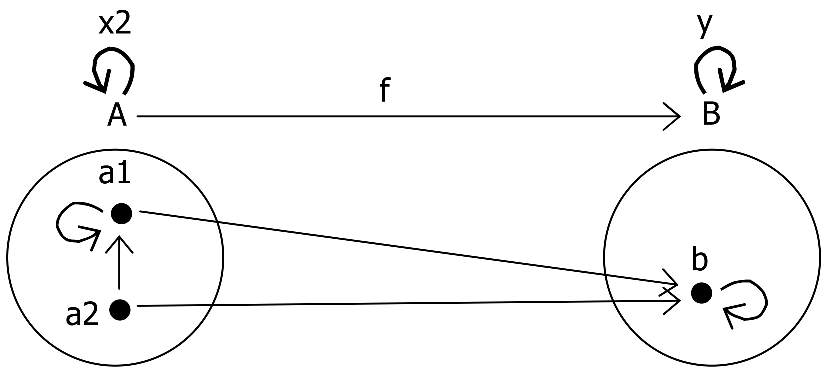
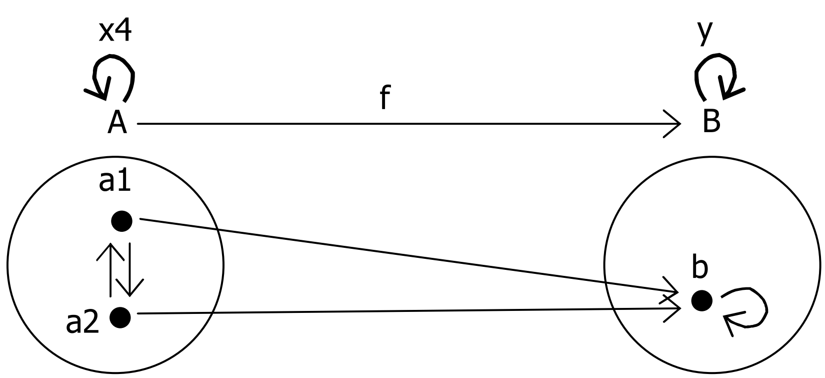


Diagram D.3 commutes fx3 = yf (diagrammatically similar to D.2 above)

The above two diagrams Diagram D.2 and D.3 answer one of our questions: (iii). Does a diagram commute if a dot which is source is assigned to a dot which is both source and target? Yes, the diagram commutes when a dot which is source of an arrow but not a target is assigned to a dot which is both source and target of an arrow.

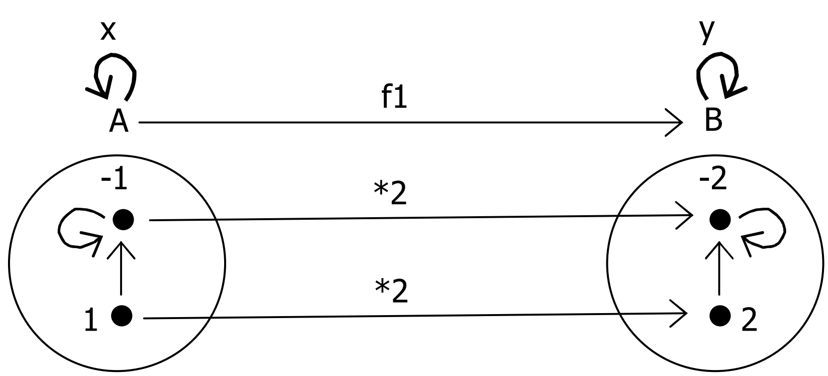
Diagram D.4 commutes fx4 = yf



The Diagram D.4 answers a question which we haven't formulated, which is that a dot which is source of one arrow and target of another arrow when mapped to a dot which is both a source and target gives a commutative diagram. Summing up we may note that commutative diagrams preserve positive properties such as source, target, 'source and target' structures as in 'a1 is source and target', but need not preserve negative properties such as 'a2 is not target' as can be seen from the commutative diagram D.2 above.

Commutativity Indicates Preserving Structure (e.g. Order)

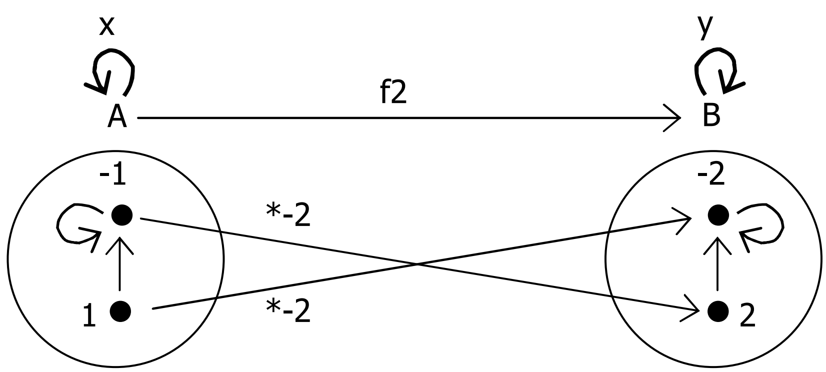
Consider a set A = {-1, 1} and a set B = {-2, 2}. Let's add the structure of '≥', which can be thought of as a map or arrow, on both A and B. In other words, we have (A, ≥) = {1 ≥ -1, -1 ≥ -1} and (B, ≥) = {2 ≥ -2, -2 ≥ -2}. Now consider two maps fi: A → B, i = 1, 2 defined as f1(a) = 2a and f2(a) = -2a.



In the above diagram, we represented the endomaps x, y on A and B, respectively, say, p ≥ q as p --> q. The above diagram commutes i.e. f1x = yf1.

More explicitly, taking either of the two paths from 1 gives the same result -2. Most importantly, we notice that the above commutative diagram preserves the order {1 ≥ -1, -1 ≥ -1} in mapping 1 to 2 and -1 to -2. In more explicit terms the order 1 ≥ -1 is preserved by 'f' in f(1) ≥ f(-1) since 2 ≥ -2.

Now consider f2(a) = -2a depicted below

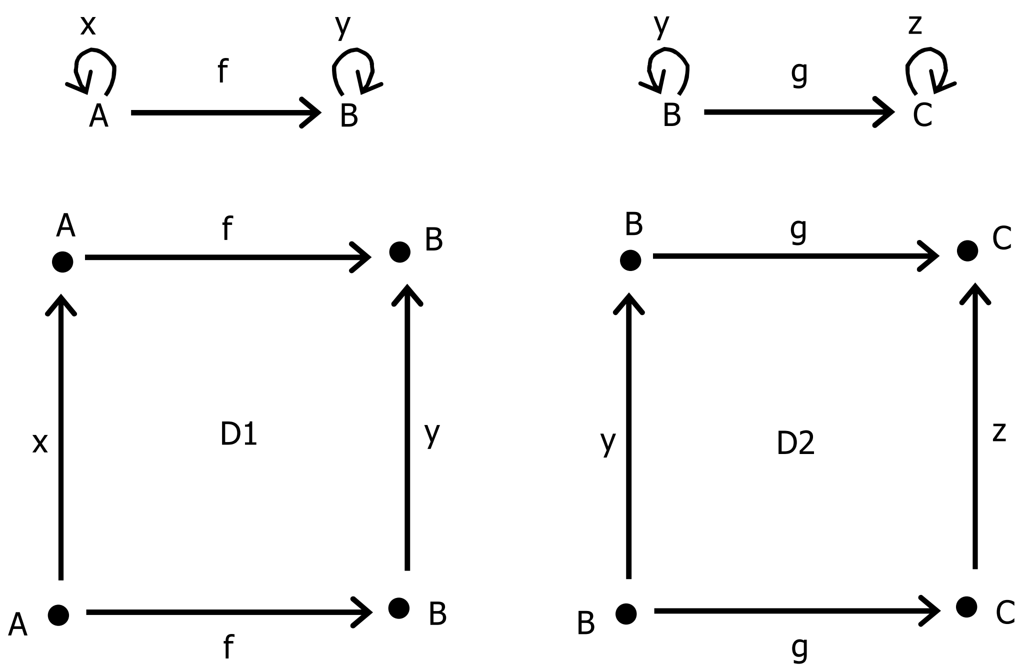


The above diagram with f2(a) = -2a is not commutative and doesn't preserve order in that 1 is mapped to -2 which is not greater than 2 to which is -1 mapped. In other words 1 ≥ -1, but f(1) is not ≥ f(-1). This example answers our question (iv). Does a diagram commute if a dot which is source (or target) is assigned to a dot which is target (or source)?, which is does not commute. Thus, from these two cases, we note that commutativity indicates preservation of order.

Composition of Structure-Preserving Maps

Let's close with my favorite notion: composition. Given 2 structure-preserving maps f and g, is the composite gf structure-preserving? This question boils down to forming a bigger diagram out of two smaller commutative diagrams, and the answer is yes, the bigger diagram is commutative and the composite map gf is structure-preserving as shown below.

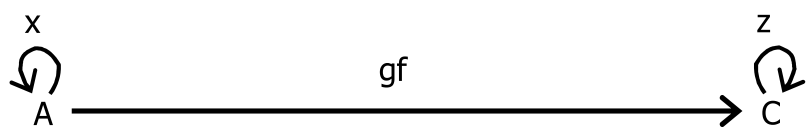
Given that maps f and g

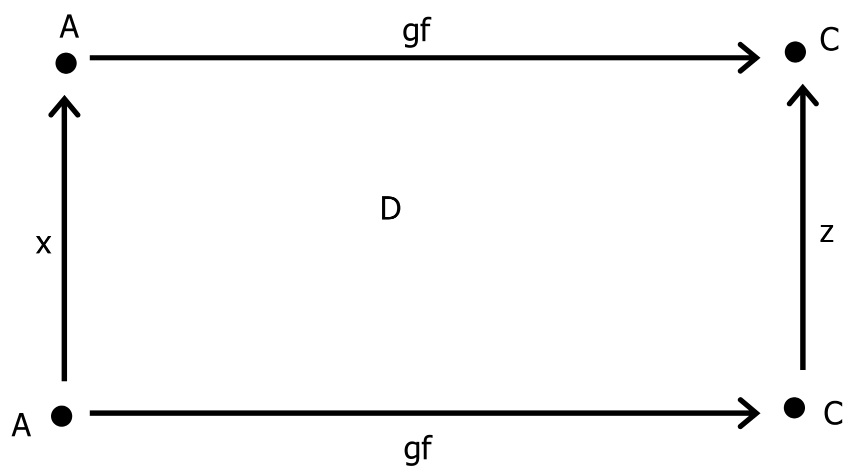


are structure-preserving or the above diagrams (squares D1 and D2) are commutative i.e.

$$fx = yf \text{ and } gy = zg,$$

is the composite gf (depicted below) structure-preserving or is the bigger diagram D (below) commutative?





Now let's see if the above bigger diagram D commutes.

gfx = gyf (since fx = yf)

= zgf (since gy = zg)

Thus the composite diagram D of two commutative diagrams D1 and D2 is commutative, or the composite of structure-preserving maps is structure-preserving.

The World is A World

of, by virtue of being, is—as is; just a simple assertion. It asks us of nothing—not a single question. So, I am happy to have a textbook that asks questions in addition to helping me appreciate the beauty beyond the paradise. Oh, how much work it is to formulate questions and then to have to answer them too once I venture out of this stone plaza [of Luis Barragan]. Within the confines of these concrete walls [of Kahn] with the text in front of me I know exactly what to prove and I am even told how to prove; before long I have to figure out what [to prove], and, then how [to prove], and then [prove]. Here's another take just in case this is not reason enough for you to jump with joy at the sight of a question. There are two kinds of [visual] objects: (1). Objects that we see and (2). Objects that let us see (of course, we can also see them). Much of the text is like objects we see; the more we study the more we learn, but it's the question that enables comprehension by way of illuminating the text. Of course, you can answer questions just as you would look at illuminants (e.g. light bulb, sun, candle light), but it's much more fun to watch the objects illuminated by way of discounting the illuminant. So, let's try to look at last week's exercise and let it shine to see what it illuminates. As it happens sometimes, what we are going to see in light of last week's exercise is somewhat like what you would see if you turn the lights on in my studio: 6 surfaces; nothing that has the immediacy of headline news. Simply put, you may find this note boring.

Here's the plan: first let's state last week's exercise, and then we will work on it, and finally close with the statement of the exercise for next Tuesday.

Exercise 1 (page 161):

Suppose that $x' = a^3(x)$ and that $f: (X, a: X \rightarrow X) \rightarrow (Y, b: Y \rightarrow Y)$ is a map in the category of endomaps. Let $y = f(x)$ and $y' = b^3(y)$. Prove that $f(x') = y'$.

Needless to note, we are not going to prove $f(x') = y'$ at lightening speed, which is not to trivialize fast, but looking at license plates and telling whether it is a number that can be expressed as a sum of squares or cubes in a jiffy is not my cup of tea (sorry Ram). So let's go slow starting with what we are given. We are given 3 equations:

$x' = a^3(x)$

$y = f(x)$

$y' = b^3(y)$

and asked for 1 equation in return (not a bad deal if equations were Benjamins)

$f(x') = y'$

First let's make sure we are given 3 equations and only three equations. If equation were just a symbol string with the symbol '=' somewhere (other than the very beginning or end) in the string, then, yes, we are given exactly 3 equations: one in the first sentence and two in the second, and of course, there is the one equation that we are asked to prove. But, wait, 'what is an equation?'

Didn't we just say an equation is a symbol string with the symbol of equality (=) as one of the symbols concatenated. Ah, yes, that's what I am trying to get at! Get at what? If a symbol (=) is a symbol of something (equality), then, surely, a bunch of symbols (equation) is symbolic of some bunch of some things (we may have to add some qualifications here). Or, how about this, an equation is an appearance of something, some state-of-affairs, or a facet of some situation. I thought of saying 'an equation is a presentation…' or 'an equation is a representation…' interchangeably. But, then, I didn't want to use 'presentation' and 'representation' in the sense of everyday usage since, at some point, I want to discuss the distinction between PRESENTATION and REPRESENTATION (I'll post a note summing up the discussion of the 'Presentations of dynamical systems (page 182) that we had sometime ago).

Returning back to equations, we can think of the three equations that we are given as saying something about some state of some being, while being cognizant of the fact that there are other modes of saying—sayings that can be translated into the format of equations. So, now, we want to make sure all that's been said is at hand before we begin to put together (I love this phrase) a profile. If we look back at the exercise we can readily see that there's something that we are told (given) in addition to the three equations. First, what's that thing that we are told? Second, can we translate that thing (description) into an equation? Third, what do we get when we put together all of the given equations? Last step, can we recognize anything in the thing that we got (by putting together the given equations) that can be expressed as the equation that we are asked to prove?

Let's begin at the beginning i.e. first find if we are told anything in addition to the 3 equations that we are given. Looking back at the exercise, we can readily see that we are told

f: (X, a: X → X) → (Y, b: Y → Y) is a map in the category of endomaps.

Ok, done with the first. Moving on to the second thing on our to-do list, can we translate the statement

'f: (X, a: X → X) → (Y, b: Y → Y) is a map in the category of endomaps' into an equation?

Not so fast, my dear friend. Given my love of the notion of SELF-CONTAINED (maybe, in part, because, I think, if I am allowed to say just one thing about the universe, I'd probably say: 'universe is self-contained'); so, I can't help but make this note self-contained (I also try to make our weekly discussions self-contained; of course, sometimes it works, sometimes it doesn't). More importantly, recently a few exceptionally brilliant high-school students joined our mailing list; so, I thought they might find it a convenient read if I made the note self-contained. Now, on to the real reason: I thought this slow walk through the basics might help me avoid saying things like 'function p from a set A to another set B, p: A → B is an idempotent when you can't even compose the function p with itself (p) leave alone the composite (pp) being equal to the function ($pp = p$), which is what is required of any function that's itching to be an idempotent endomap.

Familiar things first! What's a map? When we read 'a map in the category of endomaps,' we realize that (going by the fact that we are told of a map in some specific category) different categories might have different maps. Without belaboring anymore category of endomaps is a category which has endomaps as Objects (X, $a: X \to X$) of the category. An endomap is a map f which has same object (set A) as both domain and codomain, $f: A \to A$. A map f in the category of sets has a set (A) as domain and a set (B) as codomain and assigns an element of the set B (codomain) to each element of the set A (domain), and is denoted as (external diagram) $f: A \to B$.

Returning to the exercise, we are told '$f: (X, a: X \to X) \to (Y, b: Y \to Y)$ is a map in the category of endomaps.' Can we translate the statement into an equation? Would it (the translation) be 1 equation, 2 equations … how many, how? Since f is a map in the category of endomaps, f preserves the structure of endomaps, which translates into the equation

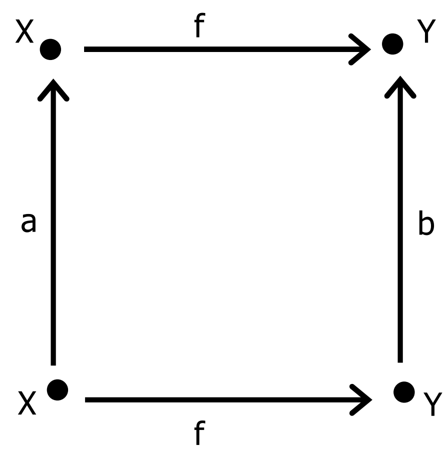
$$fa = bf$$

(Please see the attached CommutativeDiagram.pdf for an extensive discussion of preserving-structure and commutativity.)

To see how we got the above equation from the given

'f: (X, a: X → X) → (Y, b: Y → Y) is a map in the category of endomaps'

let's unpack f:



One thing that's asked of all maps is that they preserve structure, which is guaranteed by commutativity condition i.e. by the equality of the two paths between two objects: one that goes from bottom-left X to X on top and then to Y on the right, and the other that goes from bottom-left X to Y on the right and then to Y on top, which can be succinctly summed up as

$$fa = bf$$

We now have 4 equations:

1. $x' = a^3(x)$
2. $y = f(x)$
3. $y' = b^3(y)$
4. $fa = bf$

and we have to prove 1 equation:

$$f(x') = y'$$

When we are asked to prove f(x′) = y′, we are in essence told (not asked) that the equation f(x′) = y′ is a statement saying something (about the situation about which the [now] 4 equations speak of) that's been stated already by the 4 equations, albeit differently (as in different words or different tongues), but, most importantly, collectively. In other words, when we look at the image formed by putting together the equations (somewhat like putting together circular top-view and rectangular front-view into a cylinder, and, then, recognizing that the cylinder has the property of being able to hold water), we should recognize an aspect of the image thus formed that can be stated as the equation that we are asked to prove.

Let's carefully examine the given—the given 4 equations. It is helpful to have an image—not necessarily in sharp-focus or high-def—an image of what is that we are looking for; it might help us recognize if there happens to be some such thing in the 4 equations we are given:

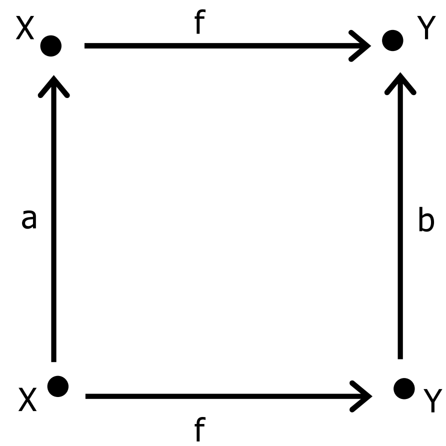
1. $x' = a^3(x)$

2. $fa = bf$

3. $y = f(x)$

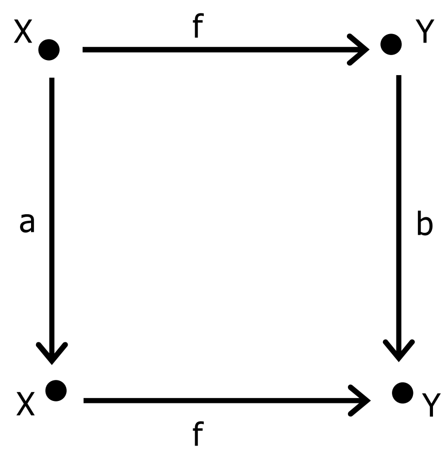
4. $y' = b^3(y)$

Let's also draw the diagram corresponding to equation 2:

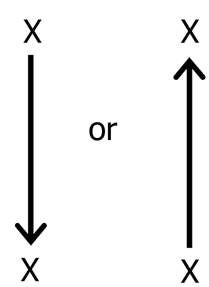


What do we have here? We have 4 arrows denoting the 3 functions (a, f, and b), and we have 2 sets (X and Y). One immediately notices that the number of arrows (4) is equal to the number of equations (4) that we are given. Let's find out how the given equations are related to the arrows of the external diagram. But before we do that let's get clear about the directions of arrows in the diagram. One simple-minded question do they have to point the way they do (a and b upwards; f rightwards). For instance, can the arrow depicting f point leftwards. No, given that the domain of the function f: X → Y is X and the codomain is Y, the arrow depicting the function f in the external diagram must point rightward as long as the domain X is depicted to the left of the codomain Y. Ok, now that we cleared that, how about the directions of arrows depicting the two endomaps a: X → X and b: Y → Y; can they point downwards instead of upwards?

Let’s see. The arrow depicting the endomaps a: X → X and b: Y → Y can point downwards since the domain and codomain of endomaps are the same sets:

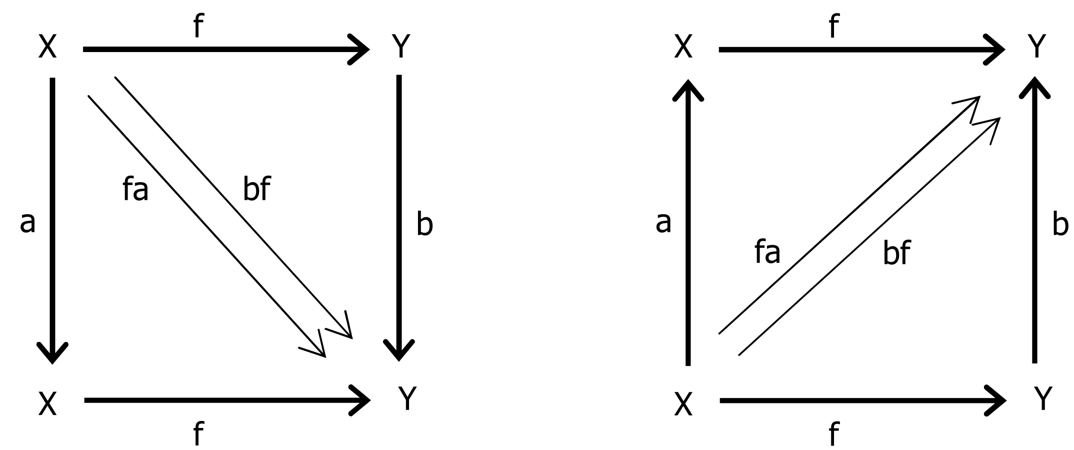


Yes, of course! What we found is just that at the scale of individual arrows flipping the direction of arrows depicting endomaps is fine. Or more explicitly we can depict both endomaps up or down as illustrated below:



Now we have to check to see whether we can flip the direction of arrows depicting endomaps a and b in the external diagram of the map f: (X, a) → (Y, b) in the category of endomaps.

In other words, we have to check if both of the following two external diagrams depict the same commutativity condition fa = bf that's required of the map f: (X, a) → (Y, b) in order for f to be a map in the category of endomaps:



Yes, they do: both diagrams are acceptable depictions of fa = bf. Or, in other words, when we say the external diagram on the left (with downward pointing arrows depicting endomaps a: X → X and b: Y → Y) commutes, we mean fa = bf. So is the case with the external diagram on the right (with upward pointing arrows depicting the same two endomaps). How about a diagram with 1 arrow up and the other down? I'd love to, but really don't want to loose the last person reading (assuming there's one left; an artist, I was told by Camus, is not worried about getting mauled by critics, but is scared to death about not having an audience).

Ok, that's enough chit-chat. Let's get back Leela—our Leela. Where are we? We are trying to find out how the given 4 equations relate to the 4 arrows of the given external diagram.

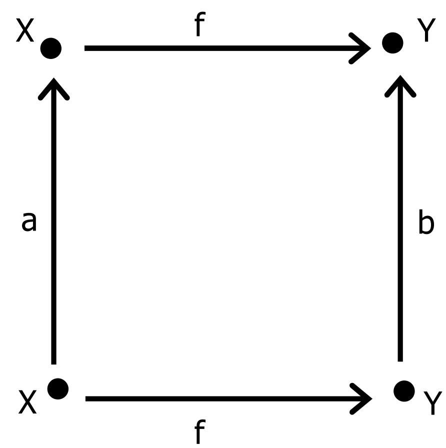
Let's line them up and take a look.

1. $x' = a^3(x)$

2. $fa = bf$

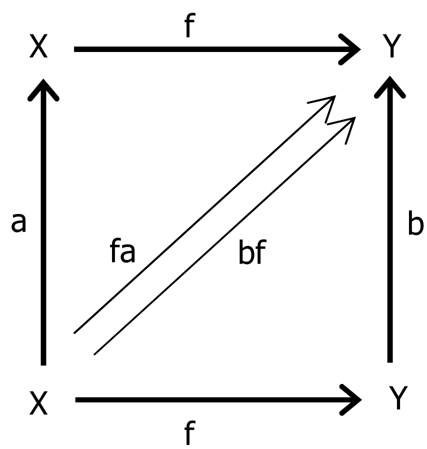
3. $y = f(x)$

4. $y' = b^3(y)$

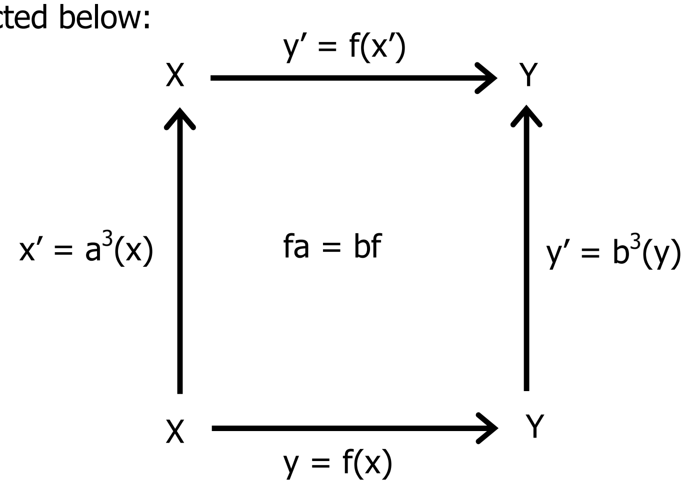


Before we go about finding how the given equations relate to the external diagram we should try to find out how the given 4 equations relate to one another. Let's start with the first and go one-by-one. In the equation $x' = a^3(x)$, a is an endomap, the upward pointing arrow on the left going from X to X. x and x' are elements of the set X and 3 tells us that a: X → X takes the element x of set X to element x' of X in 3 steps. Let's move on to the second, fa = bf. In this equation, unlike the equation 1, which had symbols denoting both maps (a) and elements (x' and x), we have 3 distinct maps (a, f, b). There are no symbols denoting elements in the equation fa = bf, which is to say that the equation holds true for all elements of the set X.

Unlike the case of the equation 1, which corresponds to 1 arrow, equation 2 corresponds to the entire diagram saying that for every element of the set X, we get the same element of the set Y whether we first transform by a and then by f or first by f and then by b as shown in the diagram with composite arrows fa and bf added:



Now let's go to the third equation y = f(x), which corresponds to the top and bottom arrows which go from set X to set Y. Speeding along we get to the fourth equation $y' = b^3(y)$, which corresponds to the right upward arrow that goes from set Y to Y. Did we find something? A little: the 3 equations $x' = a^3(x)$, y = f(x), and $y' = b^3(y)$ are of a kind in the sense they each correspond to one arrow in the external diagram, while the equation fa = bf corresponds to all 4 arrows—the external diagram as a whole—as a commutative square. How about the equation we are asked to prove f(x') = y'. Doesn't it look like the third (given) equation y = f(x). If it's going to help we could write the equation we are asked to prove as y' = f(x'), and we can go ahead and identify the (given) equation y = f(x) with the bottom arrow and the equation to be proved y′ = f(x′) with the top arrow as depicted below:



Before we move on, here's a word of caution. Even though in the present context we can switch symbols to the right of '=' with symbols on the left, this may not be legit in computer science. I vaguely remember Ruadhan telling me (something like) when we say x = y it means x is assigned the value of y which is not the same as y = x (you may want to check with Ruadhan on this one).

Back to play. We have a total of 5 equations out of which one equation (fa = bf) corresponds to the whole diagram made up of 4 arrows, and the other 4 correspond to each one of the 4 arrows of which we are given 3, and asked for 1 equation. This problem is somewhat analogous to being given a total of five numbers (say, 2, 4, 6, u, and 20) out of which one number (20) is the sum of the remaining four numbers (2, 4, 6, and u) of which one number (u) is unknown.

$$2 + 4 + 6 + u = 20$$

$$u = 8$$

With that warm-up, let's go ahead and do the Exercise 1:

Suppose that $x' = a^3(x)$ and that $f: (X, a: X \to X) \to (Y, b: Y \to Y)$ is a map in the category of endomaps. Let $y = f(x)$ and $y' = b^3(y)$. Prove that $f(x') = y'$.

First let's list the givens:

1. $x' = a^3(x)$

2. $fa = bf$

3. $y = f(x)$

4. $y' = b^3(y)$

and let's start with $f(x')$ and show that it's equal to y'.

$f(x') = f(a^3(x)) = bf(a^2(x)) = b^2f(a(x)) = b^3f(x) = b^3(y) = y'$.

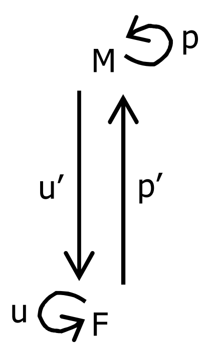
Now let's start at the other end

$y' = b^3(y) = b^3f(x) = b^2f(a(x)) = bf(a^2(x)) = f(a^3(x)) = f(x')$.

These "two" proofs: one starting with $f(x')$ and the other with y' appear different from what I had in mind when I thought it would be helpful to examine various ways of getting to equality; it must have been some other exercise which when I did reminded me of hysteresis. Simply put, let's look at the above two proofs as two paths: one going from A to B and the other from B to A. In the present case, it is clearly the case that we can retrace our steps (substitutions) that took us from A to B in order to get from B to A. Sometimes, for example, we could go from, say, San Diego to Buffalo to Hyderabad, but may not be able to return from Hyderabad to Buffalo to San Diego; we may have to take a different route to return, say, we may have to go from Hyderabad to Bangor to Amiens to San Diego. One of these days we will, surely, come across some such cases of proofs and then we can revisit. I thought of discussing the relation between structure-preserving and commutativity in terms of a skeletal version of our exercise, but I realized I have nothing interesting to say in addition to the discussion in the Commutative Diagram note; so, I am attaching it for your ready perusal.

Here's the exercise (page 145) for next Tuesday:

Consider a structure involving two sets and four maps as in



(no equations required)

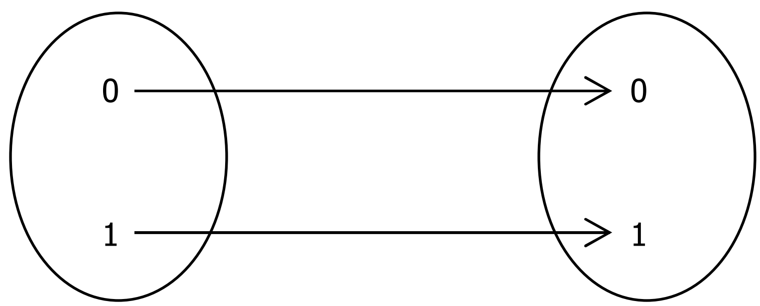
(for example M = males, F = females, p and p' are father, and u and u' are mother). Devise a rational definition of map between such structures in order to make them into a category.

I'll have to discuss this exercise, which might take some 25 pages, in a subsequent note. Got to go; my first-love (Match-to-Meaning) is calling.

Internal Diagrams: Similarity of Shapes

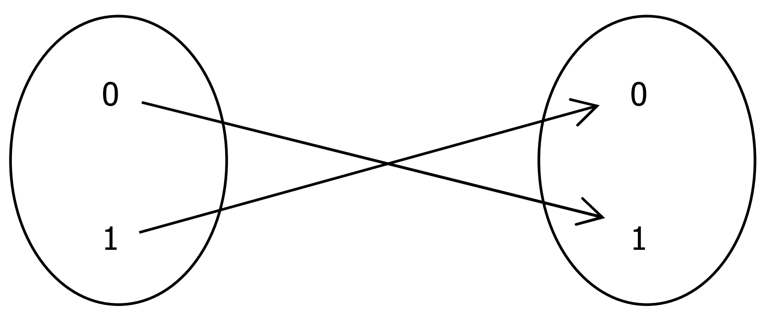
Internal Diagrams of Maps

For lack of better things to do, I was starring at a blank page hoping that somewhere in the thickets of my cortex I'll make a decision, sooner than later, to stop starring and to do something like, say, drawing—drawing a diagram—an internal diagram—an internal diagram of a map—a map from a 2-element set to a 2-element set. Thank God! Before long I drew the internal diagram of an identity map 1_A: A → A, with A = {0, 1} as shown below (I like to think of identity map as a blanket that stays put wherever you place it just like me):

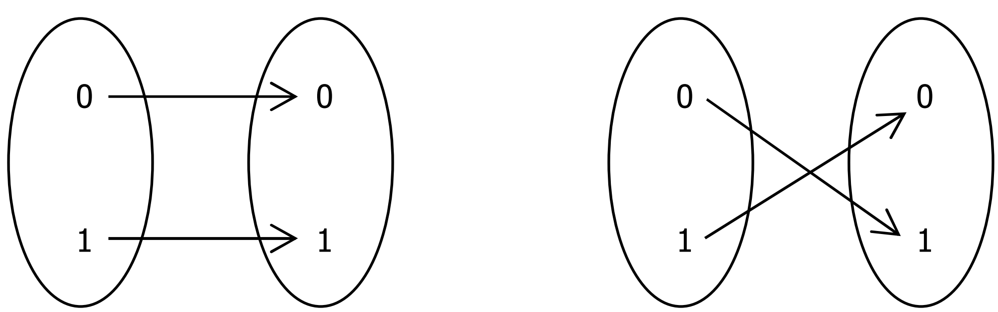


If you are wondering whether I couldn't find a more boring internal diagram, then I have to admit that I tried and this is the most skeletal diagram that I could readily think of that's meaty enough to make the point that I intended to make in this short note. Starring at the above internal diagram, I asked myself, 'do I really understand the internal diagram?', which struck me as both natural and odd; natural because these days I am going around asking, 'do I really understand this or that?'; odd because the other me inside my head said, "you mean, ‘do I really understand identity map?’” Then I said, “no, I really mean ‘do I really understand internal diagram?’”

When I look at the above internal diagram, ‘what do I see?’ Not much really—couple of arrows, couple of sets, and a couple of elements. Since mere starring at the above internal diagram doesn’t seem to reward us with much additional insight into internal diagrams, let’s look at another internal diagram of another map, say, involution i: A → A, A = {0, 1}; (think of involution as a switch, which when pressed turns the light ON (1) if the light is OFF (0) and turns it OFF (0) if it is ON (1)).

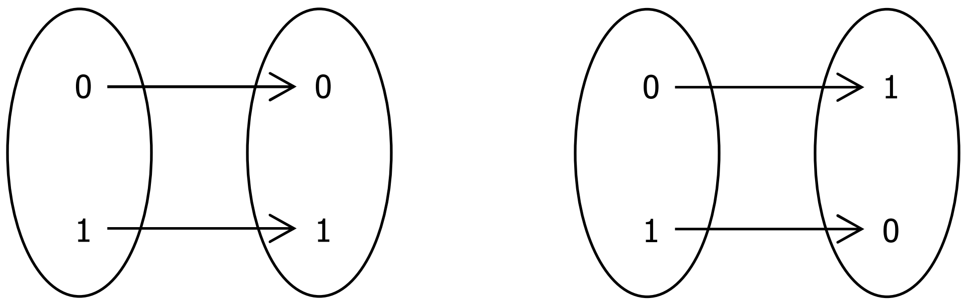


I sense something—yes, I see now. Let’s draw the two diagrams next to one another as shown below:

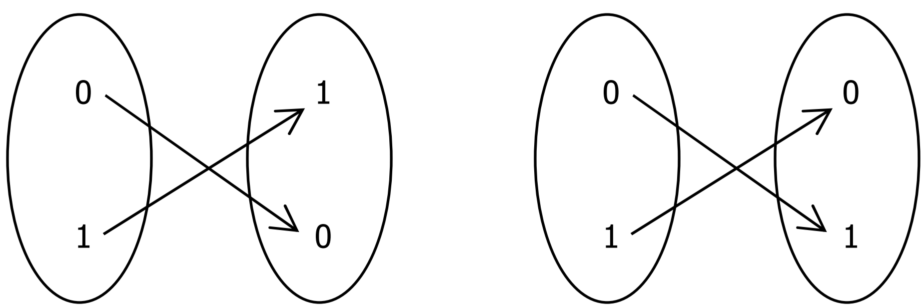


We can readily see that the two diagrams look different; the one on the left, going by its looks, may be labeled 'parallel arrows', and the one on the right 'crossed arrows'. Of course, they look different—they are diagrams of two different maps. However, looks can be deceptive. The two diagrams clearly look different, but 'do they really have different shapes?' Well, let's see if we can find out—back to starring again.

Holy cow! The two internal diagrams look different, but they don't have different shapes. I hear you (FYI: I not a run-of-the-mill muggle) saying something like: "OK, this is word-salad to me; you lost me." No, look at the two internal diagrams: 'what's the difference in shapes that you see?' One diagram has parallel arrows and the other has crossed arrows, right? Now let's ask, 'what determines, in the diagrams, the relative placement of arrows?' Simplifying further: 'what determines the placement of an arrow in an internal diagram?' The placement of an arrow is determined by the positions of 2 elements: position of an element in the domain set which is mapped by the arrow and position of the element in the codomain set to which the element in the domain set is mapped by the arrow. But, note that the placement of individual elements in the internal diagram—in the domain and codomain circles—is completely extraneous; it tells nothing about a map. What do I mean by that? Take a look at the two internal diagrams below:



The above two internal diagrams look similar, right? But they are the internal diagrams of the same identity (on the left) and involution (on the right) maps. Now the internal diagram of the involution looks similar to the internal diagram of the identity map (both have 'parallel arrows' shape). All I did is place the element '1' on top of element '0' in the codomain circle of the internal diagram of the involution map. Now look at the two internal diagrams below:



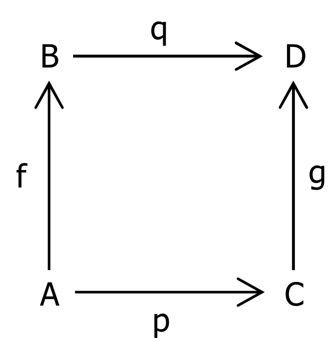
They look similar (both have 'crossed arrows' shape), but, again, they are internal diagrams of the same identity map (on the left) and involution (on the right). For this, I switched element '1' to the top in the codomain circle of the internal diagram of the identity map. The differences in the appearances of the shapes of internal diagrams of identity and involution maps (parallel vs. crossed) is, then, an artifact of the placement of elements in the depiction of sets. From this we are led to conclude that two internal diagrams of two different maps can have similar shape. This immediately raises a question: 'is the shape-similarity of the internal diagrams of identity and involution maps suggestive or better yet indicative of or depicting some similarity between the two different maps: identity and involution?' If two different maps have same-shaped internal diagrams, then 'does it mean that the two different maps are of same shape or isomorphic, formally speaking?' Well, there's only one exit: let's find out.

Before we go any further, I have a confession to make. When I first saw, on page 1 of Lawvere & Rosebrugh's book Sets For Mathematics (see attached SetsForMathExcerpt), a set depicted in two different shapes, I didn't get the take home message: the relative placement of different elements in a depiction of a set speaks nothing of the set that's depicted somewhat in a vein similar to the fact that the ink in which the elements are printed don't say anything about the set of elements. It's only when I was preparing for this class that image on page 1 came back to my mind helping me realize how important it is to recognize the need to discount the relative positions of elements of a set in a depiction of the set (as illustrated above). Of course, one has to place an element here, there, or somewhere in picturing a set, but, just as we discount the illuminant [that enables us to see objects] to see the objects illuminated, we need to discount the positional information [of elements] to see the similarity in the shapes of different internal diagrams.

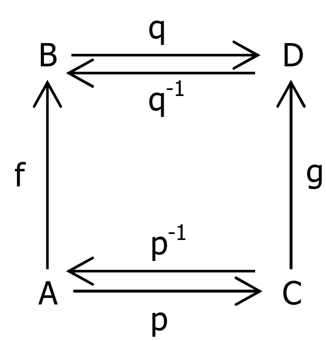
Category of Functions (of sets)

First, let's get clear about what's that we are looking for before we go searching for it. We began with two different maps: 1. identity and 2. involution and found that the internal diagrams of the two different maps have same shape, which raised the question: 'does similarity of shapes of internal diagrams of maps imply shape-similarity or isomorphism of the corresponding maps?' This question takes us to another question: 'how does one go about checking to see if two given maps are isomorphic?' In the spirit of a Saturday night stroll through the streets of downtown, let's walk slow taking in all the sights and sounds. In less exciting terms, we will ask ourselves what do we know about isomorphism; do we know how to check for isomorphism; did we ever check for isomorphism, etc. You might remember that we did check for isomorphism in previous classes; we know how to check to see if two given sets are isomorphic. What are the conditions that have to be satisfied in order for two sets A and B to be isomorphic? Two set A and B are said to be isomorphic if there exits an isomorphism between the two sets. (Pardon me if this comes across palming off a rephrasing of the question as an answer to the question.) Now we are led to the question, 'what is an isomorphism?' A function $f: A \rightarrow B$ is an isomorphism if there exists a function $g: B \rightarrow A$ such that $gf = 1_A$ and $fg = 1_B$. If there exists one such isomorphism between the two sets, then the two sets A and B are said to be isomorphic. (In passing you may note that there can be more than one isomorphism between two isomorphic sets.) Remember all of this recollection is to get to the question of how we are going to check to see if two functions are isomorphic. Going back to isomorphism between sets, and abstracting a little to get to the category of sets and functions, we find that two objects (sets A and B) in a category (of sets) are isomorphic if there exists a map (function f: A → B) in the category that's an isomorphism. From this we surmise that given a category of objects and maps, we can check to see if two objects of the category are isomorphic using conditions on maps between the objects. What does this tell us? One reading is that if we have a category with functions as objects, then we can check whether two functions are isomorphic using conditions on the maps in the category which has functions as its objects. The initial part of this exercise is simple: we need a category with functions (between sets) as objects. This is easy. Let's make one and call it category of functions. The category of functions has functions f: A → B as objects. Now to the interesting question: what are the maps in the category of functions? Fortunately we have a criterion to guide us here. Maps between objects in a category preserve the structure of objects. Translated to the present context it means maps between functions in the category of functions must preserve the structure of functions. This takes us to the question: how does one find out if something is structure-preserving and hence a map. Fortunately, we know the answer: commutativity of diagrams guarantees preservation of structure. I have to defer addressing the relation between 'structure-preserving' and 'commutative diagrams' in all its generality for a later note, but I'll briefly note that in one of the earlier notes, we saw that every time we had commutativity structure [of objects] was preserved and every time commutativity conditions were not satisfied structure was not preserved.

Let's recap the problem we are addressing. We have a category of functions with functions as objects. We are looking for a reasonable notion of map from one function to another in the category of functions. A reasonable definition of a map from one object to another in the category of functions is a pair of functions from one object (function) to another (function) such that the four functions form a commutative square as shown below:



In the above diagram, f: A → B and g: C → D are two objects (functions) in the category of functions. A map from object f to object g is a pair of functions p: A → C and q: B → D such that the above diagram commutes i.e. qf = gp. Slowly but surely we are inching our way to figuring out how to go about finding if two functions are isomorphic. Let's take stock of things as they stand now. We have category of functions, whose objects are functions and whose maps are commutative squares. Now all we need to check to see if two functions in the category of functions are isomorphic is a way to find out if the map between two functions in the category of functions is an isomorphism. Since a map in the category of functions is a pair of functions <p, q>, we extend the definition of isomorphism in the case of functions (in the category of sets) to the pair of functions i.e. to the maps in the category of functions. Looking (I am tired of starring) at the above commutative square (map in the category of functions) reproduced below, we say that the two objects f: A → B and g: C → D are isomorphic if both p: A → C and q: B → D are isomorphisms. In other words, assuming f is isomorphic to g, given a map in the category of functions i.e. a pair of functions p: A → C and q: B → D, there is a function p^{-1}: C → A such that $p^{-1}p$: A → A = 1_A and pp^{-1}: C → C = 1_C, and there is another function q^{-1}: D → A such that $q^{-1}q$: B → B = 1_B and qq^{-1}: D → D = 1_D as shown below:



Let's recap what we are doing i.e. where we are and where we are going. We are trying to find out under what conditions two objects in the category of functions are isomorphic. Assuming f: A → B and g: C → D are isomorphic in the category of functions, we came up with the following conditions: 1. the above diagram commutes i.e. $qf = gp$, and 2. the functions p, q are isomorphisms i.e. $p^{-1}p = 1_A$, $pp^{-1} = 1_C$, and $q^{-1}q = 1_B$, $qq^{-1} = 1_D$. Reflecting on these conditions, one question that comes up is whether the inverses of p and q i.e. p^{-1} and q^{-1}, respectively constitute a map in the category of functions. In other words, we have to check to see if $fp^{-1} = q^{-1}g$.

Let's start in a fresh paragraph. We are given

1. $qf = gp$
2. $p^{-1}p = 1_A$
3. $pp^{-1} = 1_C$
4. $q^{-1}q = 1_B$
5. $qq^{-1} = 1_D$

and we have to show

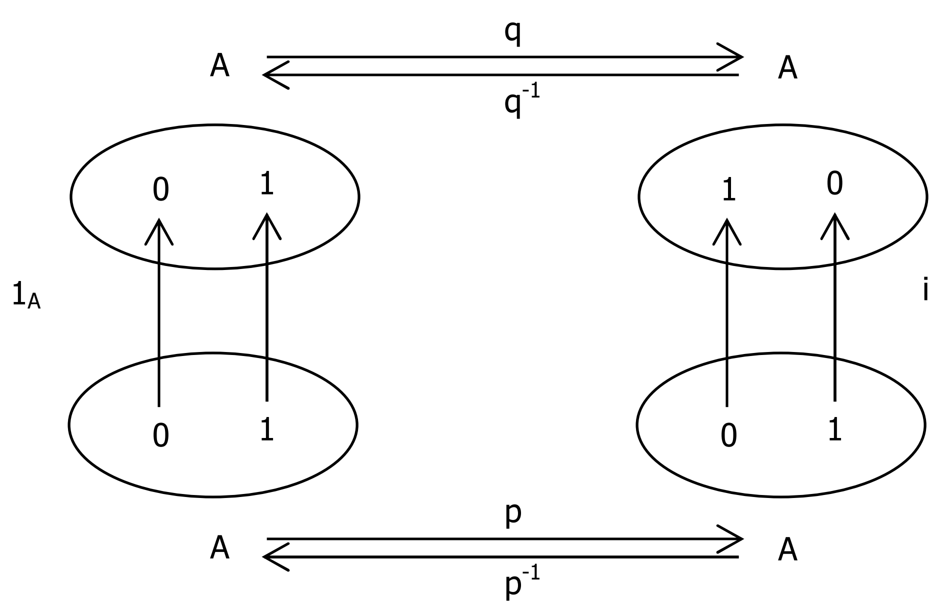
$$1^*.\ fp^{-1} = q^{-1}g$$

Let's start with left-hand side

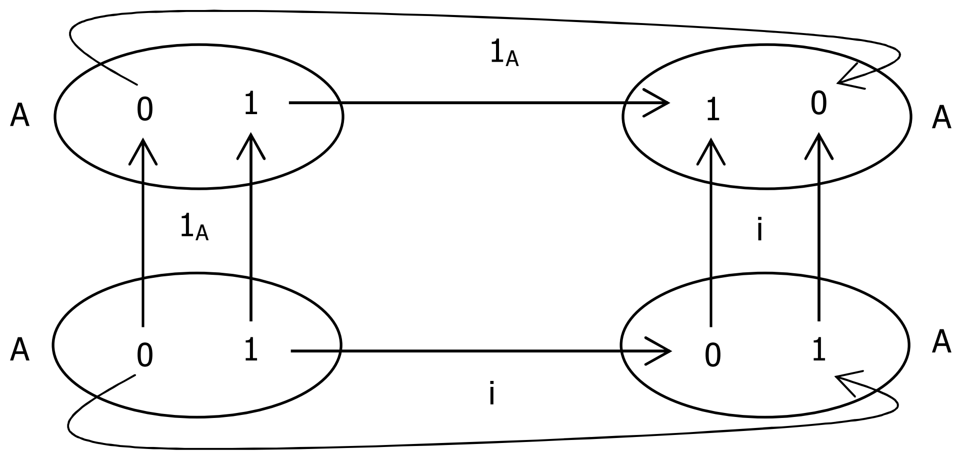
$fp^{-1} = 1_B fp^{-1} = q^{-1}qfp^{-1} = q^{-1}qfp^{-1} = q^{-1}gpp^{-1} = q^{-1}g1_C = q^{-1}g$ (which is right-hand side of the equation 1*). Thus the inverses p^{-1} and q^{-1} constitute a map in the category of functions.

Same Shape of Identity and Involution Maps

Lest we forgot, this all started with two bare-bone internal diagrams—internal diagrams of identity function and involution function reproduced below for ready reference:



In order to show that the identity function (1_A; on the left in the above diagram) is isomorphic to involution (i; on the right) in the category of functions, we have to find two isomorphisms (of sets; p and q). Let's try p = i (involution, which is an isomorphism) and q = 1_A (identity; another isomorphism) and check if the diagram commutes.

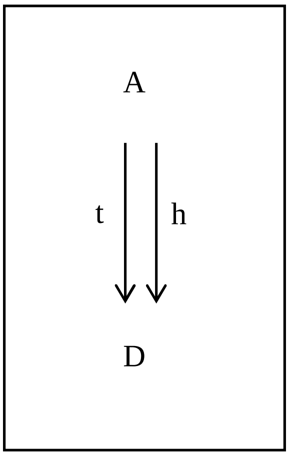


First in the above diagram we find that identity function 1_A: A → A is isomorphic to the involution i: A → A in the category of functions since both identity and involution are isomorphisms. We also find that the above diagram commutes i.e. 1_A1_A = ii (since $1_A1_A = 1_A$ and ii = 1_A) and 1_Ai = 1_Ai (commutativity of the above diagram with inverses; note that the inverse of identity is identity and that of involution is involution), which in layman terms means beginning at the left-bottom A both paths (1_A1_A and ii) to top-right A are equal. Or little bit more explicitly, start, say, at '0' of left-bottom A and following the up-right path we end up at '0' of the top-right A. Starting at the same '0' and taking the other path i.e. right-up path we end up at the same '0' to which we were led by the up-right path. The same holds true of the other element '1'. Thus we find that all conditions that are required to show that identity function and involution are isomorphic in the category of functions are satisfied. To be continued…

A Map in the Category of Graphs

Let's look at the category of graphs, which has graphs such as O shown below as objects.

O

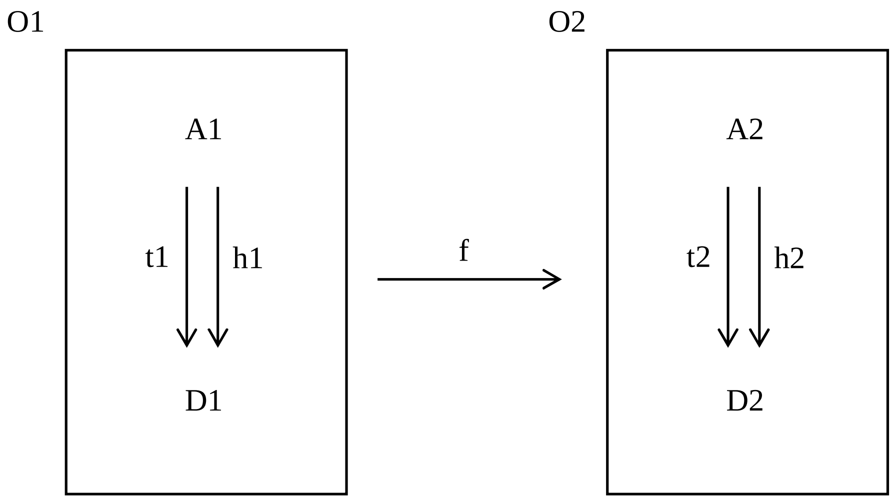


Object O is a pair of sets: set A of arrows, set D of dots; and a parallel pair of functions: function t assigns to each arrow (in A) its source dot (tail; in D); function h assigns to each arrow (in A) its target dot (head; in D).

A map f from an object O1 to an object O2

f: O1 --> O2

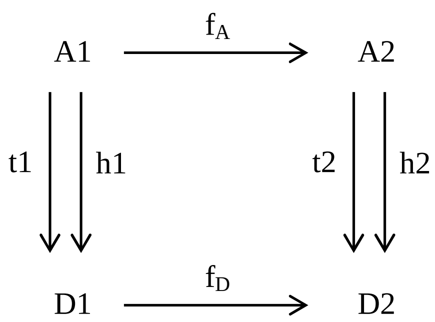
depicted as



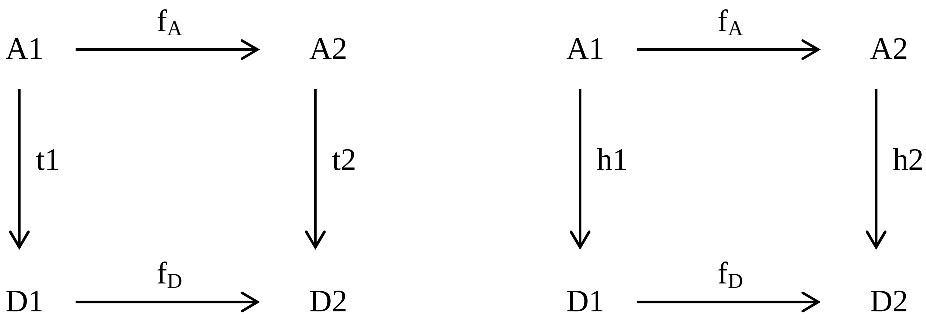
is a pair of functions

f = <f_A, f_D>

depicted as



or as a pair of commutative squares



satisfying

$t_2 f_A = f_D t_1$

corresponding to the square on the left and

$h_2 f_A = f_D h_1$

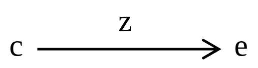
corresponding to the square on the right (in the above).

Let's consider a map

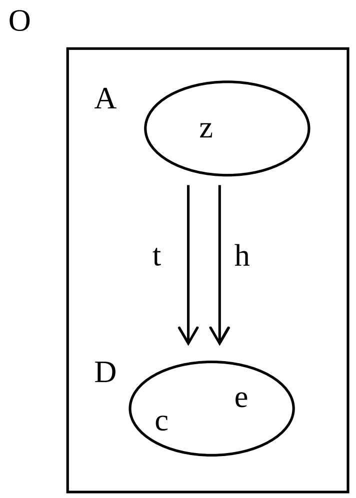
f: O --> O

from domain object O to codomain object O to illustrate the idea of map in the category of graphs in some more detail.

Let’s take a graph



as our object O



A = {z} and D = {c, e} are the pair of sets of arrows and dots, respectively of object O.

t: A --> D, with t(z) = c

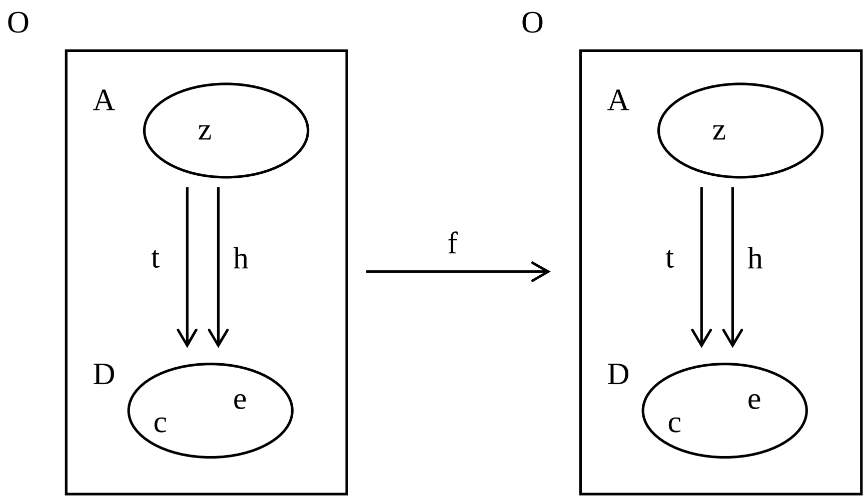
and

h: A --> D, with h(z) = e

are the parallel pair of functions of tail (source) and head (target), respectively of the object O.

Now, the map

f: O --> O



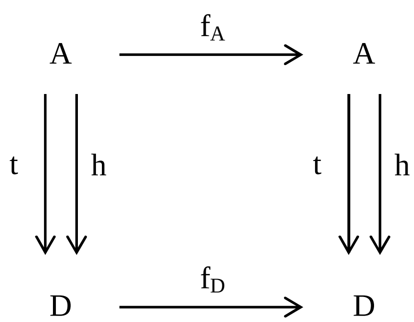
which we recollect as

f = <f_A, f_D>

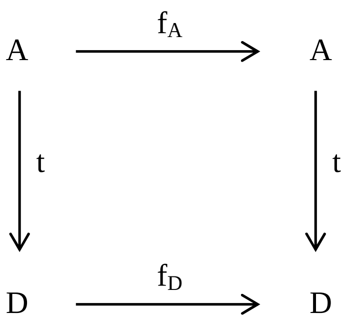
with

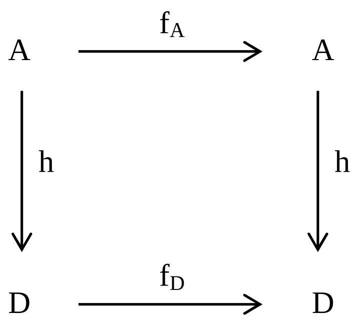
f_A: A --> A and f_D: D --> D

depicted as



and after separating heads from tails





satisfies

$tf_A = f_Dt$ and $hf_A = f_Dh$

Now, we have a question!

What does ‘a map f: O --> O is a pair of functions $f = <f_A, f_D>$ satisfying $tf_A = f_Dt$ and $hf_A = f_Dh$’ mean?

What do we have here? We have 4 functions:

t: A --> D

h: A --> D

f_A: A --> A

f_D: D --> D

of which we already know, clearly, what the functions tail t and head h are. But first, let’s write the domain and codomain sets of the functions.

$A = \{z\}$

$D = \{c, e\}$

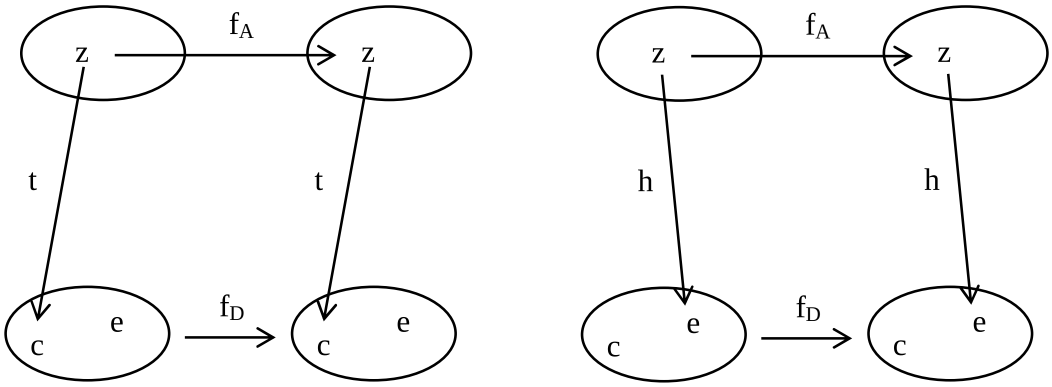
The function t: A --> D is given by t(z) = c, and the function h: A --> D is given by h(z) = e.

How about the functions f_A and f_D?

Let's start with f_A: A --> A.

Since A = {z}, there is only one possibility for f_A; the function f_A assigns the only element z of the codomain set A to the one element z of the domain set A; $f_A(z) = z$.

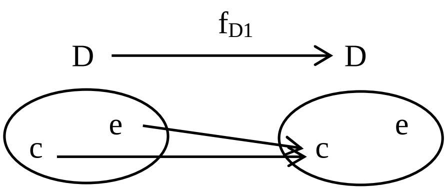
Before we go on to f_D: D --> D, let's depict diagrammatically all that we stated above as

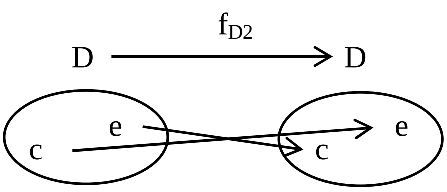


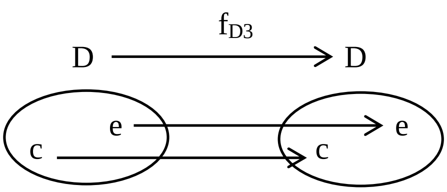
Now in order for the f: O --> O to be a map, function f_D: D --> D, D = {c, e} must satisfy $tf_A = f_Dt$ and $hf_A = f_Dh$.

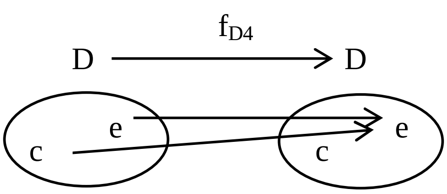
What is function f_D? f_D is a function f_D: D --> D from domain set D = {c, e} to codomain set D = {c, e}.

Given that there are 2 elements in the domain set D and 2 elements in the codomain set D, we have a total of 4 (2^2) functions from D to D as shown below:









Now in order to find out how many maps there are from the object O to O, we have to see how many of the following 4 pairs of equations hold true.

1. $tf_A = f_{D1}t$ and $hf_A = f_{D1}h$

2. $tf_A = f_{D2}t$ and $hf_A = f_{D2}h$

3. $tf_A = f_{D3}t$ and $hf_A = f_{D3}h$

4. $tf_A = f_{D4}t$ and $hf_A = f_{D4}h$

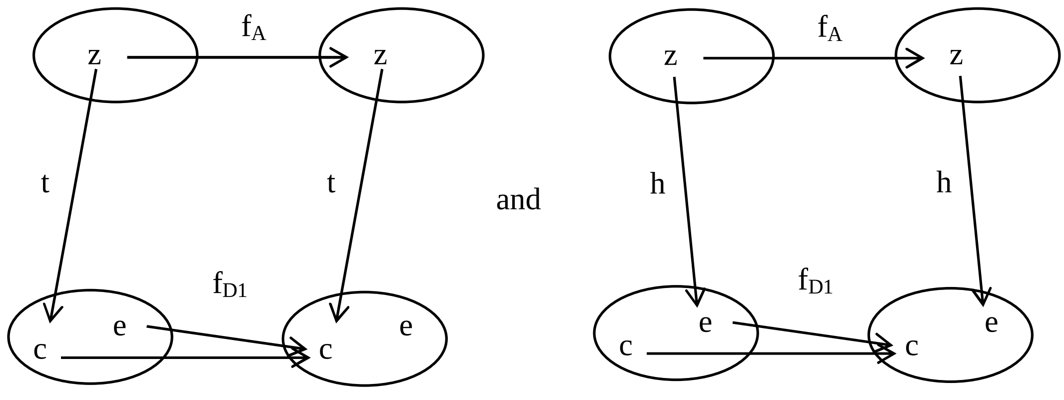
Restating what we just said, we say

$f_1 = \langle f_A, f_{D1} \rangle : O \dashrightarrow O$

is a map from domain object O to codomain object O if

$tf_A = f_{D1}t$ and $hf_A = f_{D1}h$

or pictorially, if



commute.

We say a diagram, for example, the square on the left commutes if $tf_A = f_{D1}t$.

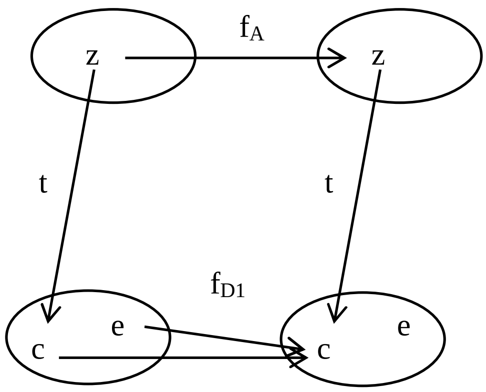
OK, fine, but first let's list out all 3 functions in the equation $tf_A = f_{D1}t$ to be satisfied:

$t(z) = c$

$f_A(z) = z$

$f_{D1}(c) = c$ and $f_{D1}(e) = c$

which is what is depicted in the diagram



Let's take off at the top-left z; we can take f_A and go to z, and from z take t to land at c. Or, we can take t, from the very same top-left z, and go to c, and from c take f_{D1} to land at c. Both itineraries take us from z at the top-left to the very same down-right c. Speaking less verbally, we evaluate both the left-hand side and the left-hand side of the equation

$tf_A = f_{D1}t$

at z to see if the equation

$tf_A = f_{D1}t$

holds true.

Left-hand side

$tf_A(z) = t(z) = c$

Right-hand side

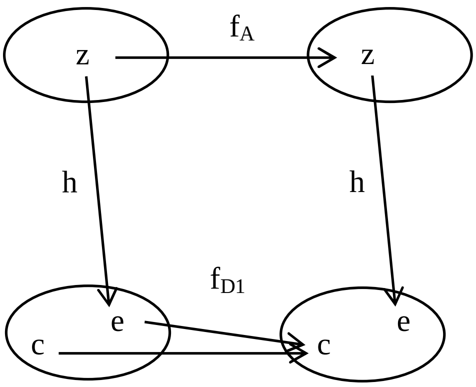
$f_{D1}t(z) = f_{D1}(c) = c$

Therefore

$tf_A = f_{D1}t$

which is not surprising given that we already saw that the corresponding diagram commutes.

Now let's see if our diagram on the right (above) corresponding to heads



commutes, for which we check if $hf_A = f_{D1}h$.

Evaluating at z

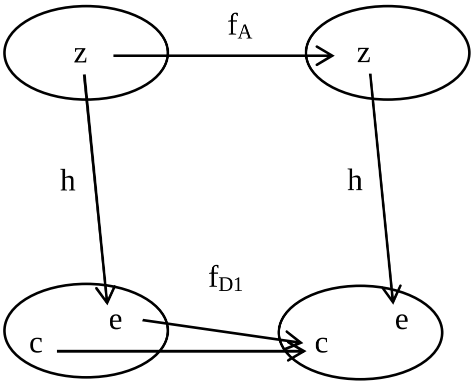
$hf_A(z) = h(z) = e$

$f_{D1}h(z) = f_{D1}(e) = c$

we find that

$hf_A \neq f_{D1}h$

i.e.



doesn't commute.

Let's remind ourselves what we are doing now. We started out saying

$f_1 = <f_A, f_{D1}>: O \dashrightarrow O$

is a map if

$tf_A = f_{D1}t$ and $hf_A = f_{D1}h$.

We found out that

$tf_A = f_{D1}t$

but

$hf_A \neq f_{D1}h$.

So $f_1 = <f_A, f_{D1}>$ is not a map.

How about

$f_2 = <f_A, f_{D2}>$

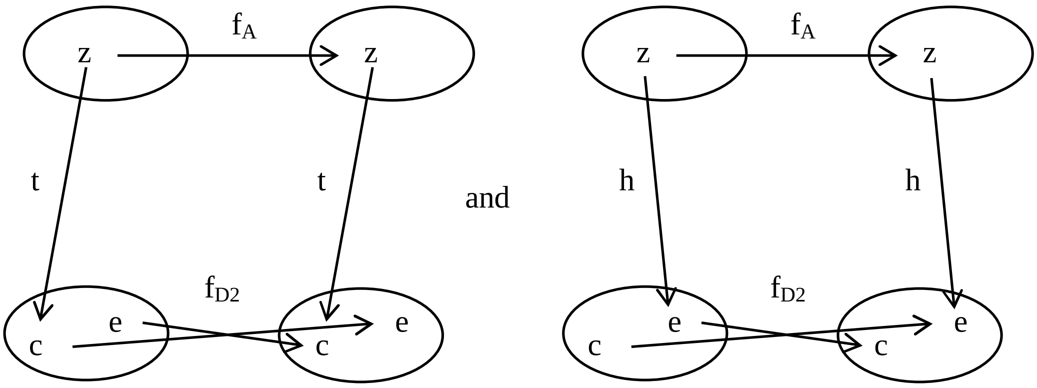
$f_3 = <f_A, f_{D3}>$

$f_4 = <f_A, f_{D4}>$

Let’s look at

$f_2 = <f_A, f_{D2}>$

f_2 is a map if



commute.

In terms of equations,

if $tf_A = f_{D2}t$ and $hf_A = f_{D2}h$,

then f2 = <fA, fD2> is a map.

Let's first look at the equation on the left

$tf_A = f_{D2}t$

and evaluate both sides of the equation at z.

$tf_A(z) = t(z) = c$

$f_{D2}t(z) = f_{D2}(c) = e$

Therefore, $tf_A \neq f_{D2}t$.

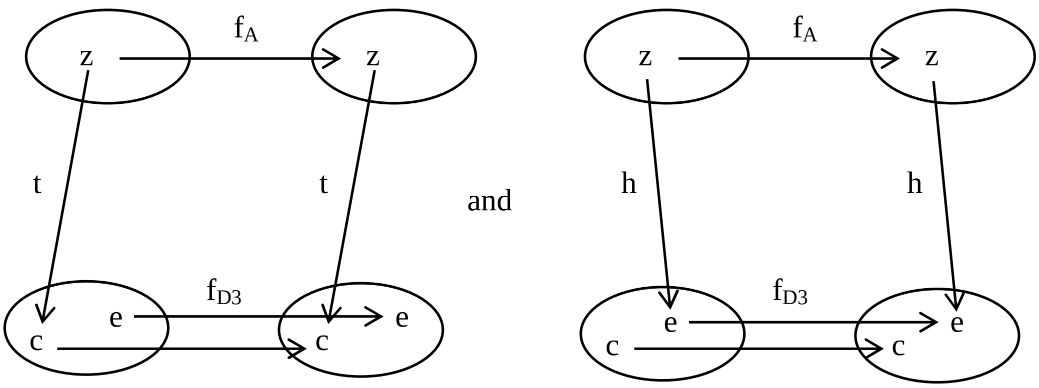
Since we need both equations

$tf_A = f_{D2}t$ and $hf_A = f_{D2}h$

to hold true for $f_2 = <f_A, f_{D2}>$ to be a map, and since we found $tf_A \neq f_{D2}t$, we won't bother checking the other equation, and conclude $f_2 = <f_A, f_{D2}>$ is not a map.

How about $f_3 = <f_A, f_{D3}>$?

Does the pair of diagrams



commute?

We have to check if

$tf_A = f_{D3}t$ and $hf_A = f_{D3}h$

which we can also do by following the arrows in the diagram in addition to substituting symbols in the equations.

Evaluating both sides of the equation on the left at z

$tf_A(z) = t(z) = c$

$f_{D3}t(z) = f_{D3}(c) = c$

Therefore, the equation $tf_A = f_{D3}t$ holds true i.e. the corresponding diagram on the left commutes.

Next, evaluating $hf_A = f_{D3}h$ at z, we find that

$hf_A(z) = h(z) = e$

$f_{D3}h(z) = f_{D3}(e) = e$

Therefore, the equation $hf_A = f_{D3}h$ holds true i.e. the corresponding diagram on the right commutes.

Since

$tf_A = f_{D3}t$ and $hf_A = f_{D3}h$

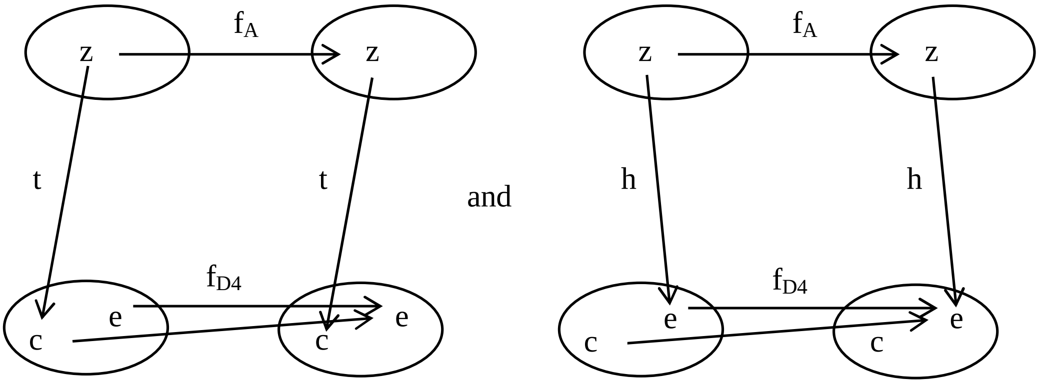
we say

$f_3 = \langle f_A, f_{D3} \rangle$

is a map f_3: O --> O from domain object O to codomain object O.

How about f_4 = <f_A, f_{D4}>?

Does the pair of diagrams



commute?

Is $tf_A = f_{D4}t$ and $hf_A = f_{D4}h$?

Evaluating both sides of the equation on the left at z, we find that

$tf_A(z) = t(z) = c$

$f_{D4}t(z) = f_{D4}(c) = e$

Therefore, $tf_A \neq f_{D4}t$. Thus, f_4 = <f_A, f_{D4}> is not a map.

To sum up, of all the 4 possibilities

f_1 = <f_A, f_{D1}>

$f_2 = <f_A, f_{D2}>$

$f_3 = <f_A, f_{D3}>$

$f_4 = <f_A, f_{D4}>$

we found that only

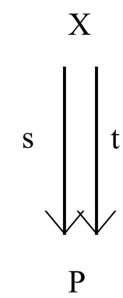
$f_3 = <f_A, f_{D3}>$ is a map f_3: O --> O from the domain object O to the codomain object O.

Well, what does all this mean? Where's the big-picture? Here, it might help to note that f_{D1} mapped both head and tail to tail, f_{D2} mapped tail to head and head to tail, and f_{D4} mapped both tail and head to head, while f_{D3} mapped head to head and tail to tail.

So?

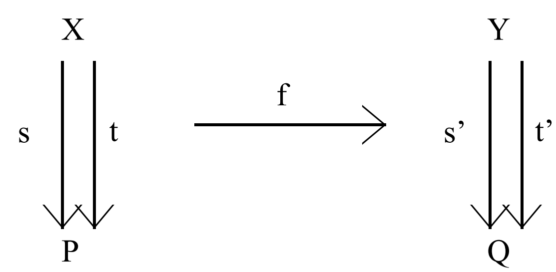
Composition of Maps in the Category of Graphs

An object of the category of graphs is a parallel pair of functions

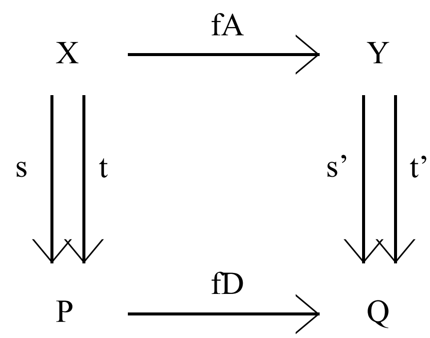


where X is called the set of arrows and P the set of dots of the graph. If x is an arrow (element of X), then s(x) is called the source of x, and t(x) is called the target of x.

A map

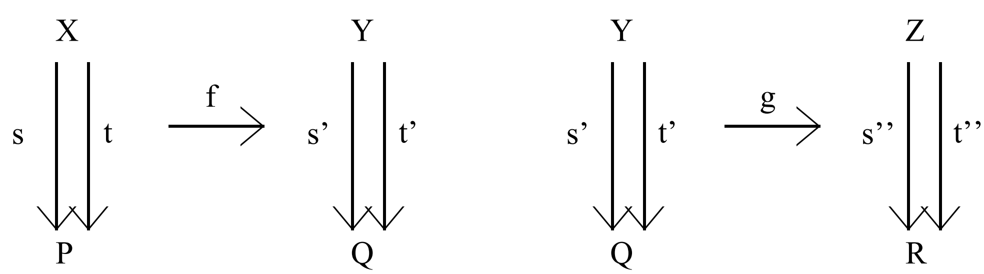


in the category of graphs is defined to be any pair of functions f_A: X --> Y, f_D: P --> Q for which the diagram

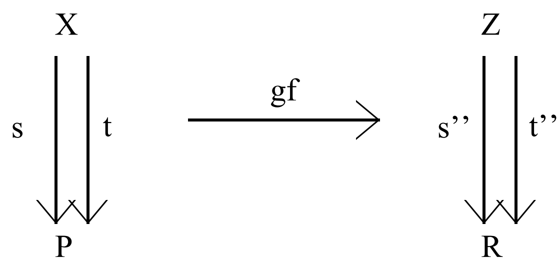


commutes satisfying $f_D s = s' f_A$ and $f_D t = t' f_A$.

What is the composite map gf of the maps f and g depicted below



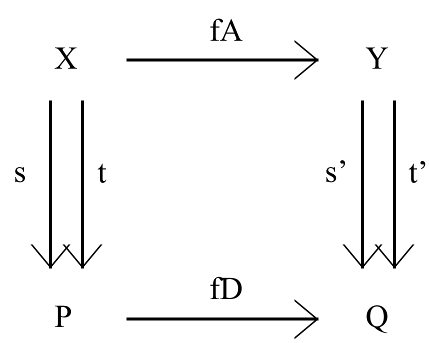
The composite map gf of maps f and g is



Is the above composite map gf a map in the category of graphs?

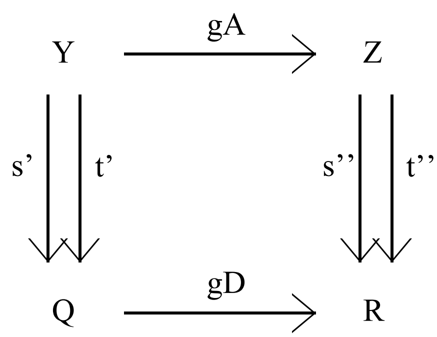
First, let's look at the maps f, g in the category of graphs of which gf is composite.

The map f, when spelled-out, is



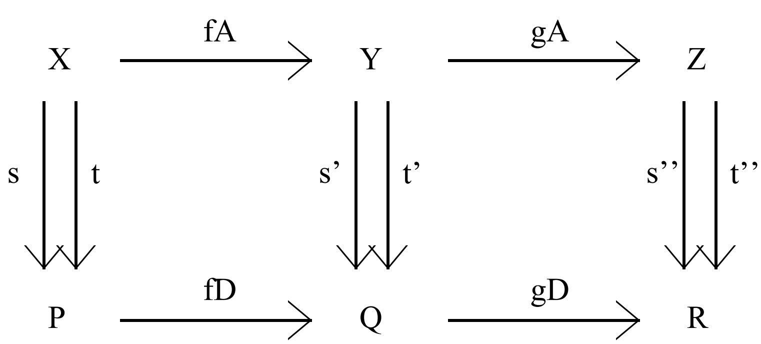
satisfying $f_D s = s' f_A$ and $f_D t = t' f_A$

The map g, when spelled-out, is

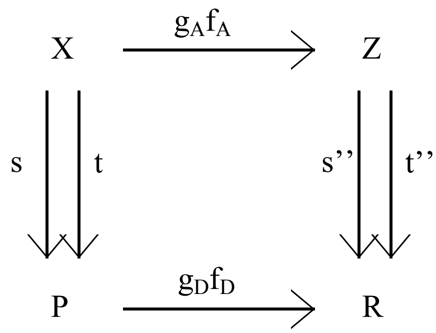


satisfying $g_D s' = s'' g_A$ and $g_D t' = t'' g_A$

The composite gf of maps g after f



which is equal to



which must satisfy

$s''g_Af_A = g_Df_Ds$ and $t''g_Af_A = g_Df_Dt$

for the composite gf to be a map in the category of graphs.

We know, going by the fact that f and g are maps in the category of graphs, that

$f_Ds = s'f_A$ and $f_Dt = t'f_A$

and

$g_Ds' = s''g_A$ and $g_Dt' = t''g_A$

and that we have to check to see if $s''g_Af_A = g_Df_Ds$ and $t''g_Af_A = g_Df_Dt$

$s''g_Af_A = g_Ds'f_A = g_Df_Ds$ and $t''g_Af_A = g_Dt'f_A = g_Df_Dt$

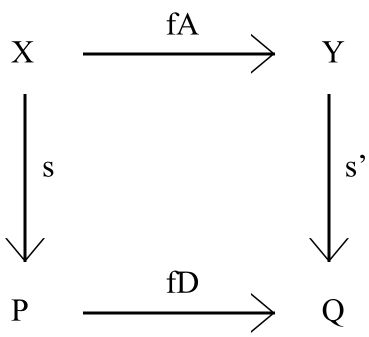
Therefore…; I’ll let you conclude, but given that we, often, look at one thing and see something (plz don’t press that panic button; I am saving my symbolic conscious experience for sometime later), what do we see when we look at symbol substitution in, say,

$s''g_Af_A = g_Ds'f_A = g_Df_Ds$

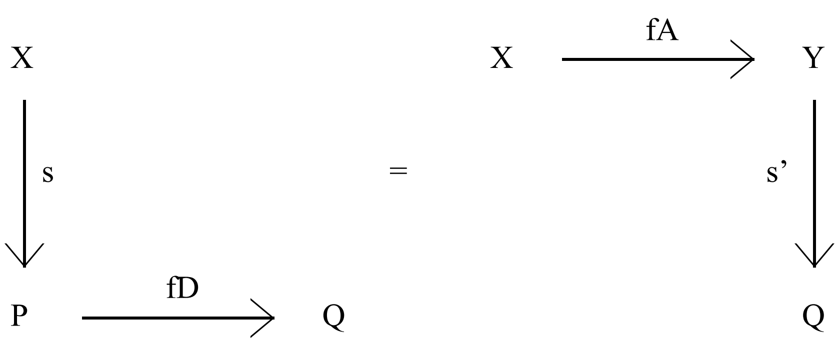
Given

$f_Ds = s'f_A$

we look at



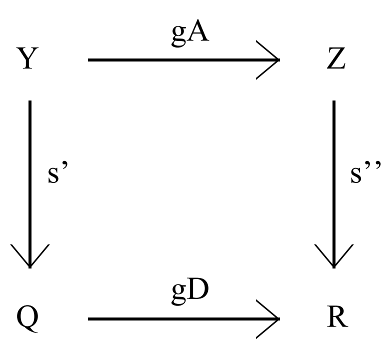
and see



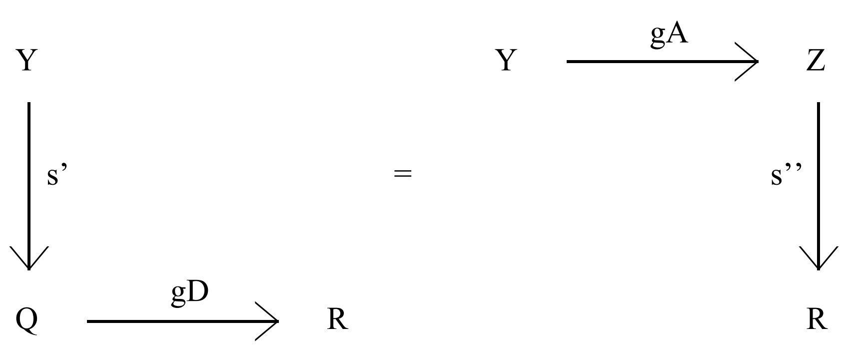
Again, given

$g_D s' = s'' g_A$

we look at



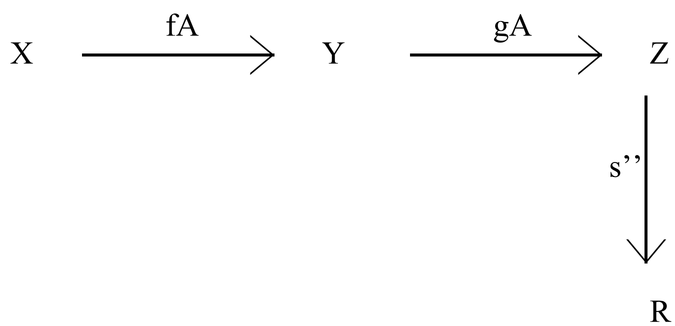
and, from our vantage point, see



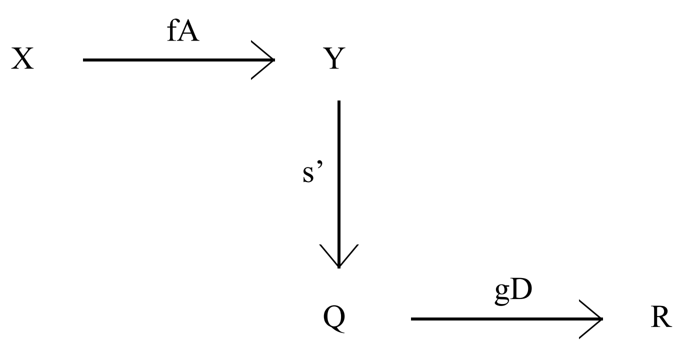
Now, from this perspective, when we look at

$s''g_A f_A = g_D s' f_A = g_D f_D s$

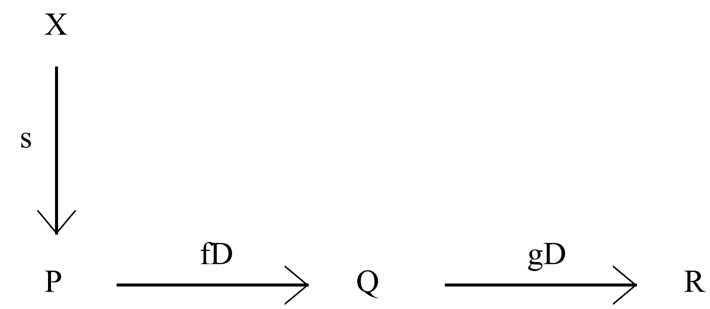
we see



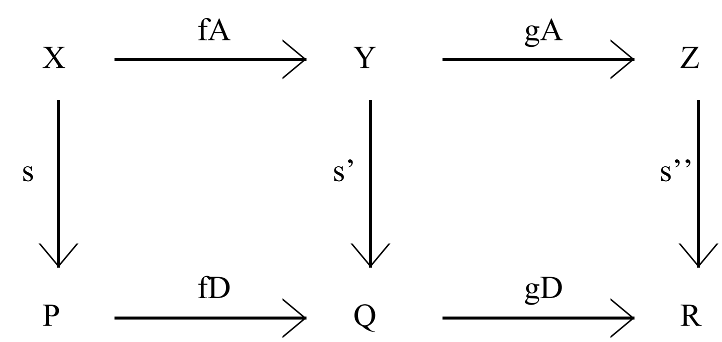
equal to



equal to



in



If you say so.

Now, what do we see when we look at

$t''g_Af_A = g_Dt'f_A = g_Df_Dt$

OK, fine. No more drops dripping on to forehead; for now let’s just friend—facebook—

substitution and composition.

See you soon, alligator!

Identity Maps in the Category of Graphs

First, let’s look at the definition of CATEGORY.

A category consists of the data:

(1) Objects A, B, C, …

(2) Maps f, g, h, …

(3) For each map f, one object A as domain of f and one object B as codomain of f as in f: A --> B.

(4) For each object A, an identity map with object A as both domain and codomain of the identity map as in 1_A: A --> A.

(5) For each composable pair of maps f: A --> B, g: B --> C with domain of g, B equal to codomain of f, B, a composite map gf with the domain of f, A as domain and the codomain of g, C as codomain as in gf: A --> C.

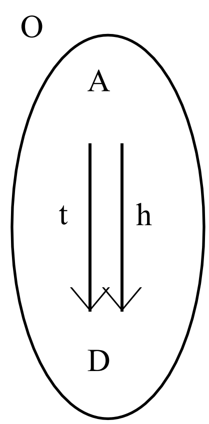
The above data of category satisfy the following rules:

(1) Identity laws: If f: A --> B, then $1_B f = f$ and $f 1_A = f$.

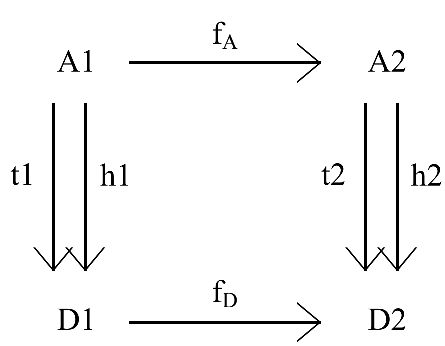
(2) Associative law: If f: A --> B, g: B --> C, and h: C --> D, then (hg)f = h(gf) = hgf.

Now that we have seen Category, let’s look at Category of Graphs.

An object O of the category of graphs is a parallel pair of functions called tail, head with a set called arrows as domain and a set called dots as codomain of the pair of functions as in t: A --> D, h: A --> D shown below:



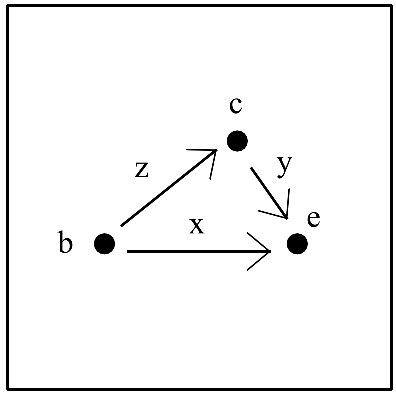
A map f: O1 --> O2 from a domain object O1 (t1: A1 --> D1, h1: A1 -->D1) to a codomain object O2 (t2: A2 --> D2, h2: A2 --> D2) is a pair of functions f_A: A --> A, f_D: D --> D as in



satisfying $t_2f_A = f_Dt_1$ and $h_2f_A = f_Dh_1$.

Before we go any further, let's look at an object O in the category of graphs

O

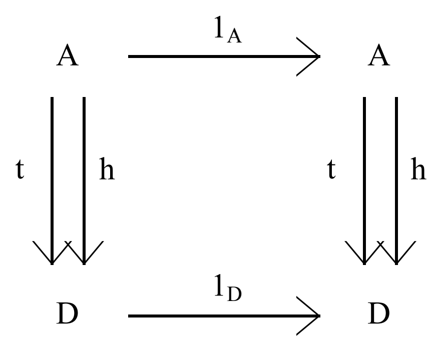


and save it to monkey later.

Now, if we look back at the definition of category, it looks like we recognized (1), (2), and (3) of the data of a category in the case of our category of graphs. Now we have to look for (4), i.e., identity map.

What's an identity map in the category of graphs? Thanks to the definition, we need not get lost in thought.

An identity map is a map. Before we unwrap this goodie, let's parrot the definition. For each object O (t: A --> D, h: A --> D), there is an identity map with object O as both domain and codomain of the identity map as in 1_O: O --> O. Now let's bite into the chocolate before it melts away. Let's recollect that a map (which is what an identity map is first and foremost) in the category of graphs is a pair of functions satisfying a pair of equations (our life couldn't have been easier), which when translated to the case of identity map 1_O: O --> O translates to a pair of identity functions 1_A: A --> A, 1_D: D --> D as in



satisfying a pair of equations $t1_A = 1_Dt$ and $h1_A = 1_Dh$.

Do they? Don't we have to check to see if $t1_A = 1_Dt$ and $h1_A = 1_Dh$? Aren't we defining? If so, by definition, isn't $t1_A = 1_Dt$ and $h1_A = 1_Dh$. Well, don't we want our definition of Category of Graphs to be consistent with our, again, definition of Category? Definition, in delimiting, description, changes—changes in practice—in the practice of describing. Holy cow! For now, as an exit-strategy, let's just say we aren't modern enough—enough to go post-modern, go [all-out] postal. Jeez!

Let's now check if $1_O = (1_A, 1_D)$ is a map in the category of graphs. In other words, let's check if $t1_A = 1_Dt$ and $h1_A = 1_Dh$, noting that t: A --> D, h: A --> D.

Looking back at the definition of the category, we see:

If f: A --> B, then $1_B f = f$ and $f1_A = f$.

So, given t: A --> D, $1_D t = t$ and $t1_A = t$. With another so in tow, we have $t1_A = 1_D t$.

In a similar vein, given h: A --> D, $1_D h = h$ and $h1_A = h$. Therefore, as earlier, $h1_A = 1_D h$.

Thus the identity map 1_O: O --> O defined as a pair of identity functions 1_A: A --> A,

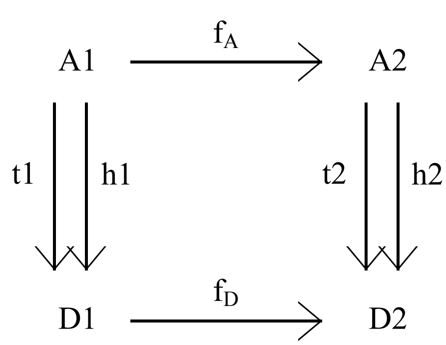
1_D: D --> D is indeed a map in the category of graphs.

Now, is the map 1_O: O --> O in the category of graphs an identity map in the category of graphs?

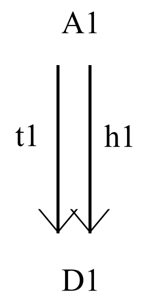
Wut!

Well, when in doubt, we study the definition—definition of category.

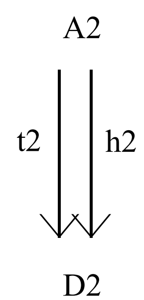
We have, in the category of graphs, a map



with a domain object

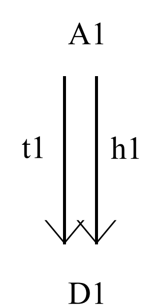


and a codomain object

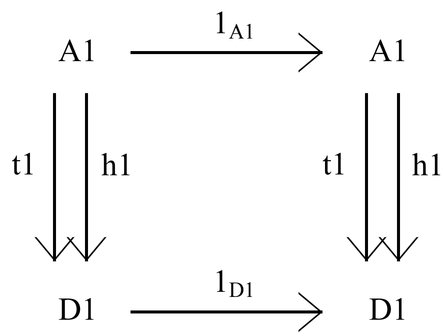


satisfying $t_2f_A = f_Dt_1$ and $h_2f_A = f_Dh_1$.

For each object of the category we have an identity map with the very object as both domain and codomain. So, corresponding to the object

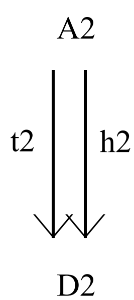


we have the identity map

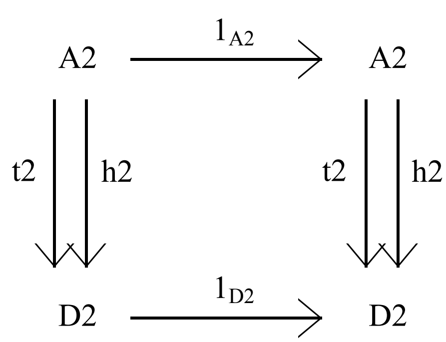


satisfying $t_1 1_{A1} = 1_{D1} t_1$ and $h_1 1_{A1} = 1_{D1} h_1$.

And corresponding to

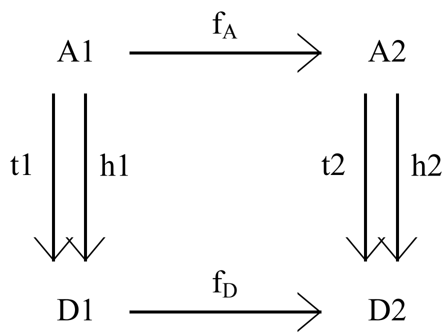


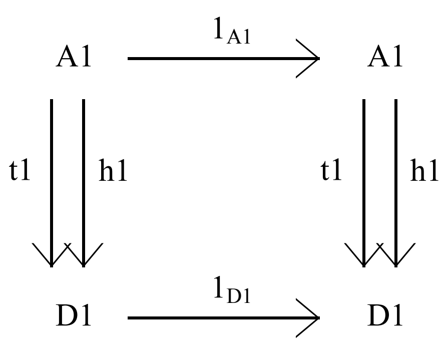
we have the identity map

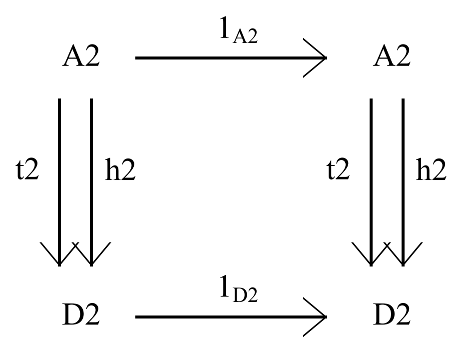


satisfying $t_2 1_{A2} = 1_{D2} t_2$ and $h_2 1_{A2} = 1_{D2} h_2$.

To sum up, we have three maps







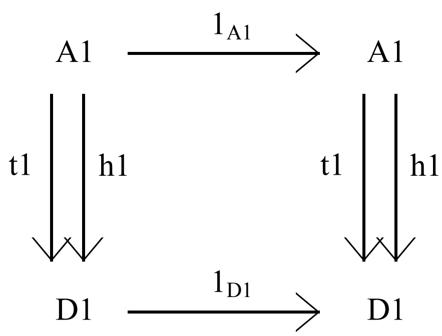
What are we going to do with this trinity?

Looking ahead, in the rear-view mirror, at the definition of category, we see the identity laws

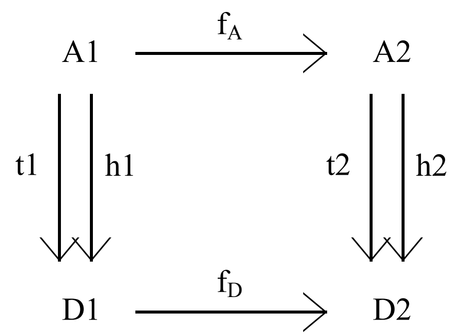
If f: A --> B, then $1_B f = f$ and $f 1_A = f$

that maps in a category must satisfy.

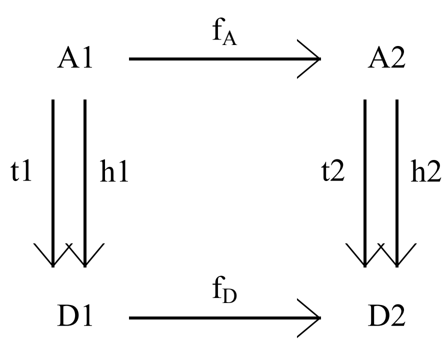
Importing these beautiful laws into our category of graphs, we see that we have to, first, see if the composite of



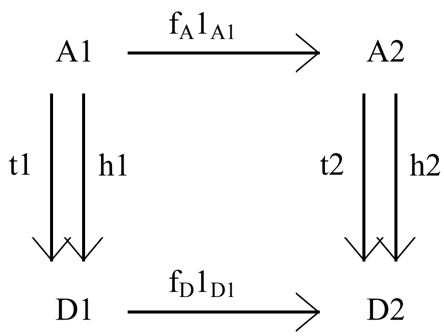
and



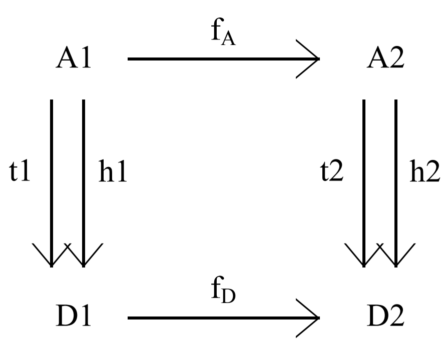
is equal to



The composite map of (1_{A1}, 1_{D1}) and (f_A, f_D) shown above can be drawn as

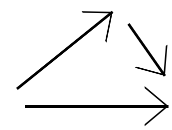


which is equal to



So is the case with the other identity law.

Now, I feel like, in explaining something, I said something like, ‘that’s what it means’ to which I can hear you say something like ‘what is that that that that it is supposed to mean?’ Or, in more politically-correct terminology, there’s always room for clarification, which I’ll provide in terms of the example



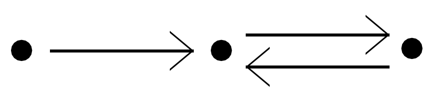
of an object in the category of graphs we saw earlier, but didn't get a chance to look at.

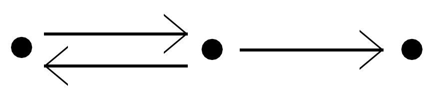
Isomorphisms in the Category of Graphs

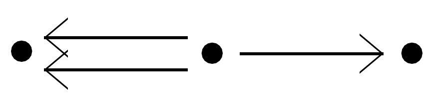
Let's do Exercise 6 (Conceptual Mathematics, page 159).

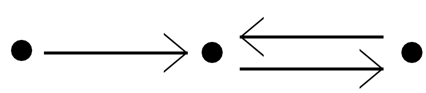
Exercise: Each of the following graphs is isomorphic to exactly one of the others.

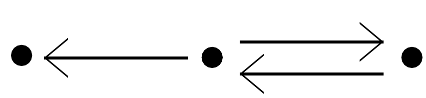
Which?

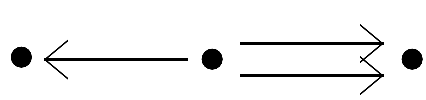
(1) 

(2) 

(3) 

(4) 

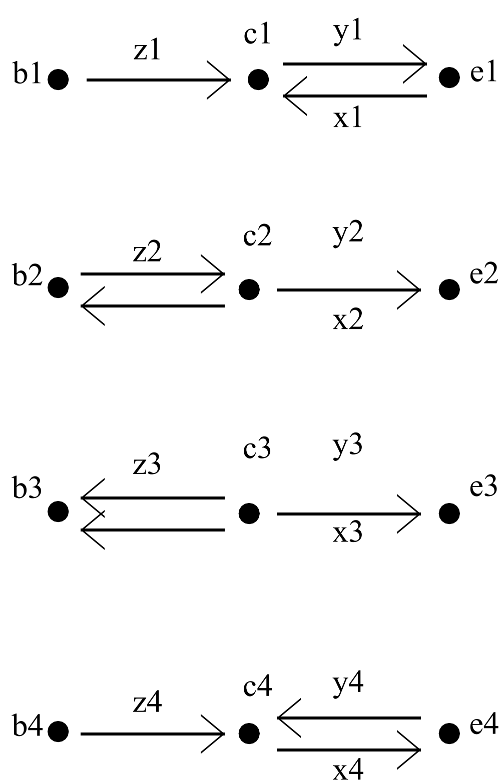
(5) 

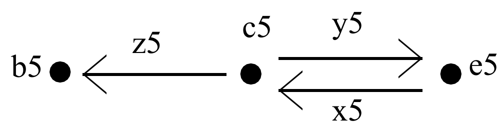
(6) 

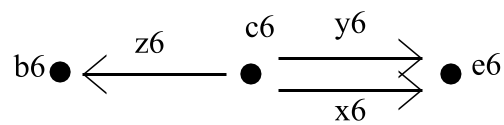
Earlier on (page 158) we learn that two graphs are isomorphic if we can exactly match arrows of one graph to arrows of the other and dots of one to dots of the other; in such a way that if two arrows are matched, then so are their source-dots and so are their target-dots. Listening to what we are just told we find it, comforting, notwithstanding the demanding exactness, to learn that math is our making—in our hands.

Whatever.

Let's first label the arrows and dots of the given six graphs.

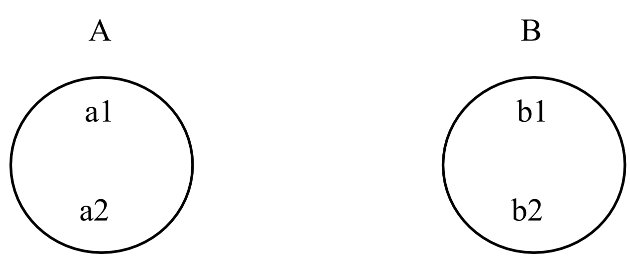






Now let's see how isomorphism looks like in the case of something much more familiar, say, sets.

Consider two sets A = {a1, a2} and B = {b1, b2} shown below:



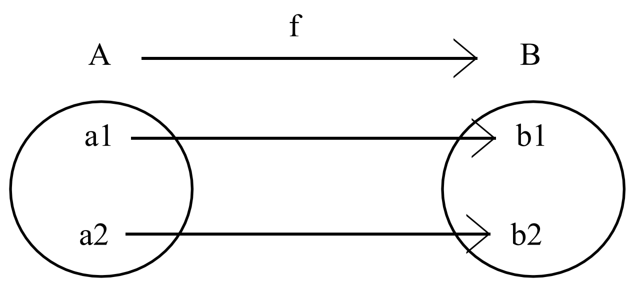
Clearly, A and B are two different sets. So, we can ask, 'are there any similarities between the two sets A, B?' In asking this question, we are rather boldly, but with good reason, asserting that two different things can be similar in more than one respect. (On a not so tangential note, when we are around kids, we often mistake their statements for questions—for not so well-formulated questions failing to recognize them as what they indeed are: answers we all knew full-well, but have forgotten during the course of our schooling by the society, which is what my sister's daughter Bhavana has been teaching me recently.). Well, this is not as high-funda as it sounds. After all, we can be similar in just one dimension, say, living, or in exactly two dimensions, say, living and feeling, so on and so forth.

Returning to our sets A and B, we say that A and B are isomorphic (same shape, which in the case of sets happens to be size) if there exists an isomorphism between A and B. Does this sound somewhat like: two different things are similar if there exits a similarity between the different things. It better; welcome to the wonderful wizard of obvious.

Now let's ask, 'what is isomorphism?' An isomorphism is a map. A function f: A --> B from domain set A to codomain set B is an isomorphism if there exits a function g: B --> A such that the composite function of f and g, gf: A --> B --> A = 1_A, the identity function on A and the composite function of g and f, fg: B --> A --> B = 1_B, the identity function on B.

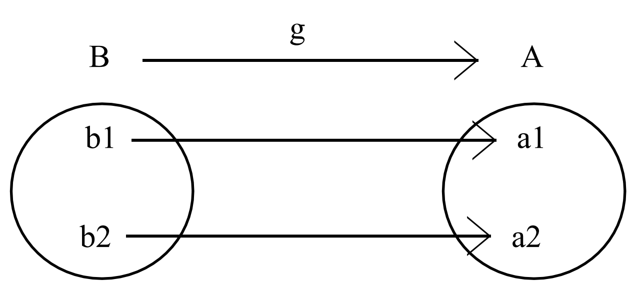
Given that we already know that the given two sets A = {a1, a2} and B = {b1, b2} are of the same size, i.e. |A| = |B| = 2, let's see if A and B are isomorphic. All we need is one isomorphism between A and B.

Consider a function f: A --> B, whose internal diagram is shown below:



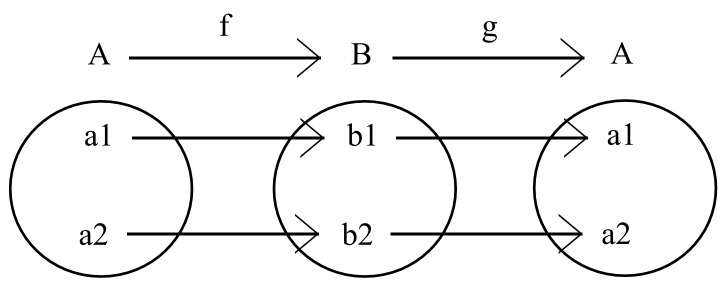
and in terms of equations f(a1) = b1 and f(a2) = b2

and a function g: B --> A, whose internal diagram is shown below:

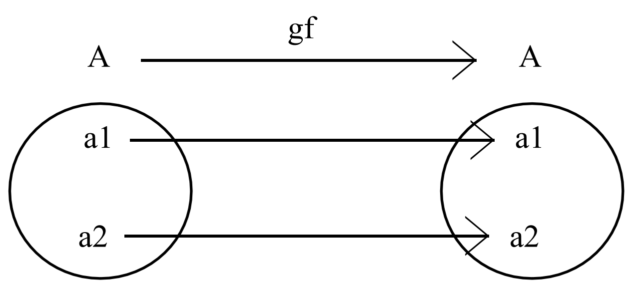


and in terms of equations g(b1) = a1 and g(b2) = a2.

The composite function gf: A --> B --> A is

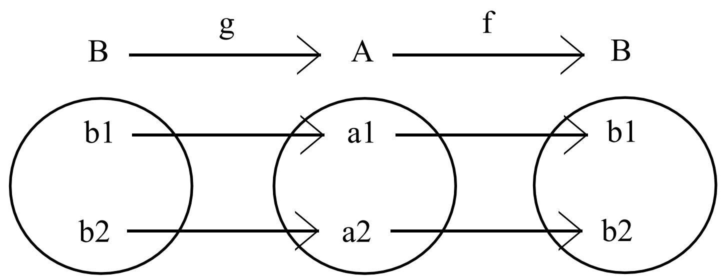


which is equal to

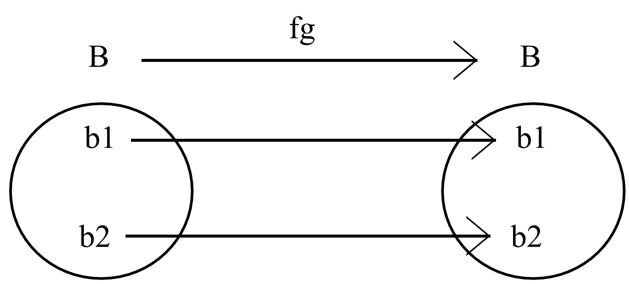


Looking above we see gf: A --> A = 1_A, i.e. 1_A(a1) = a1 and 1_A(a2) = a2.

The composite function fg: B --> A --> B is



which is equal to



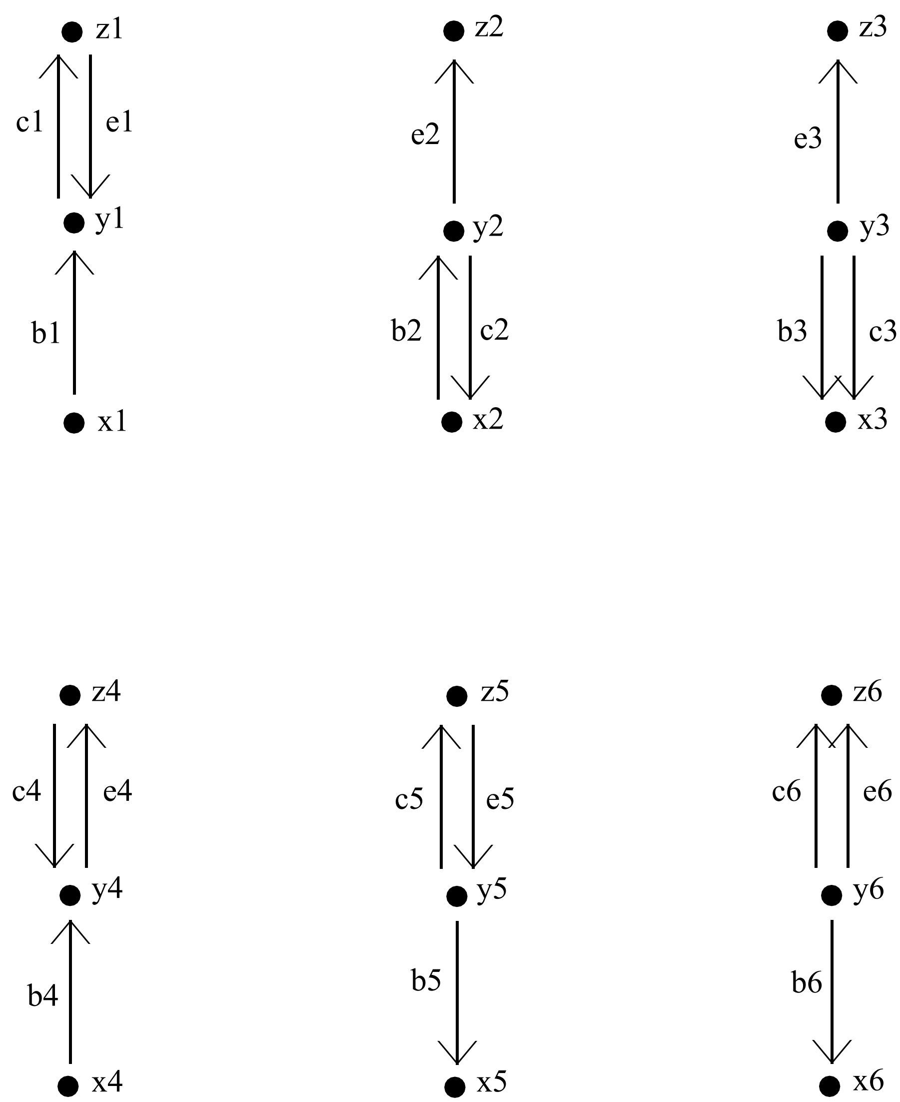
Looking above, one more time, we see that fg: B --> B = 1_B, i.e. 1_B(b1) = b1 and 1_B(b2) = b2.

So we say A and B are isomorphic; are of the same size without even counting the number of elements of either set A or B. I guess this is what it means to participate in the practice of plain-sight, of stating the obvious.

To be continued…

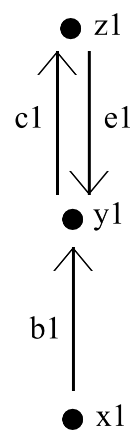
Exercise 6 (Conceptual Mathematics, page 159)

Let’s complete Exercise 6; we have a long ways to separating in the category of graphs (Conceptual Mathematics, page 215).

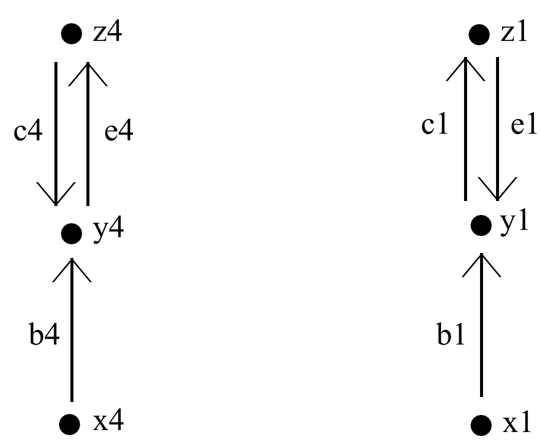


Each one of the above six graphs is isomorphic to exactly one of the other five graphs.

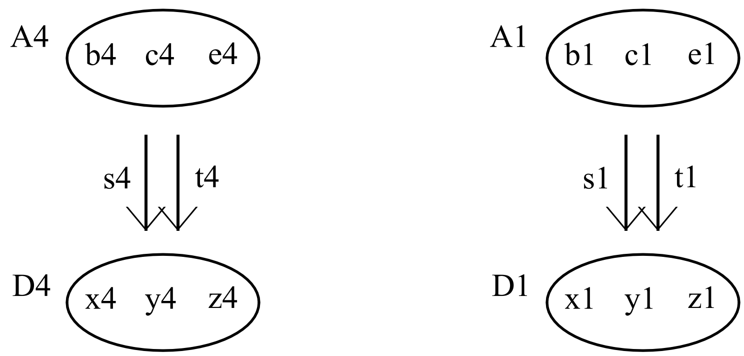
Let’s start with graph 1.



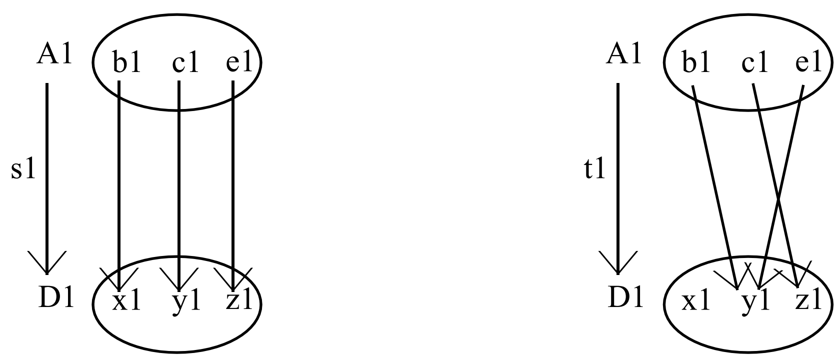
Looking at the other five graphs, it appears as though graph 4 is like graph 1. Let’s place them next to one another.



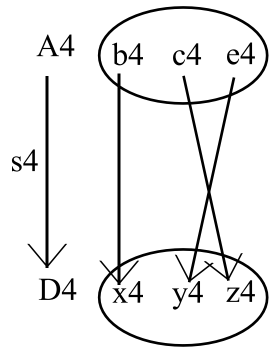
Let's now see if there is an isomorphism between the above two graphs depicted below.

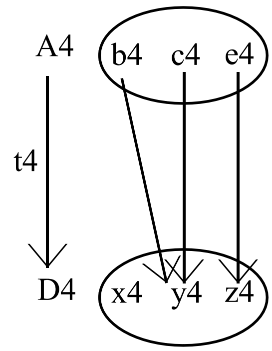


Let's first note that s_1: A1 --> D1 and t_1: A1 --> D1 are given as shown below.

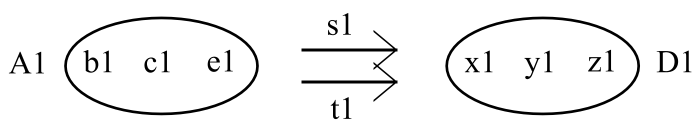


So are s_4: A4 --> D4 and t_4: A4 --> D4 as shown below.

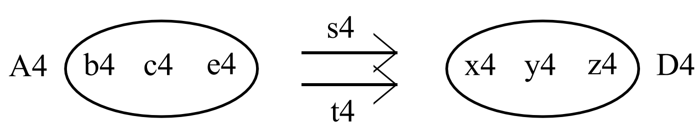




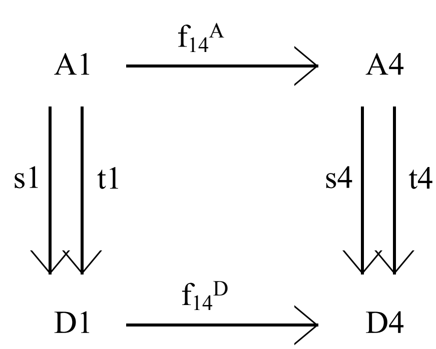
Now to show that the graph



is isomorphic to the graph



we need to find an isomorphism f_{14}: (s1, t1) --> (s4, t4)

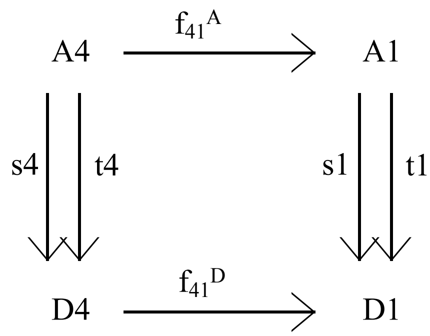


In order for $f_{14} = \langle f_{14}^A, f_{14}^D \rangle$ to be an isomorphism, first, it has to be a map in the category of graphs satisfying $s_4 f_{14}^A = f_{14}^D s_1$ and $t_4 f_{14}^A = f_{14}^D t_1$

(don't we love subscripts and superscripts; oops, no venting)

Next up, in order for the map $f_{14} = \langle f_{14}^A, f_{14}^D \rangle$ to be an isomorphism, we need a map

$f_{41} = \langle f_{41}^A, f_{41}^D \rangle$

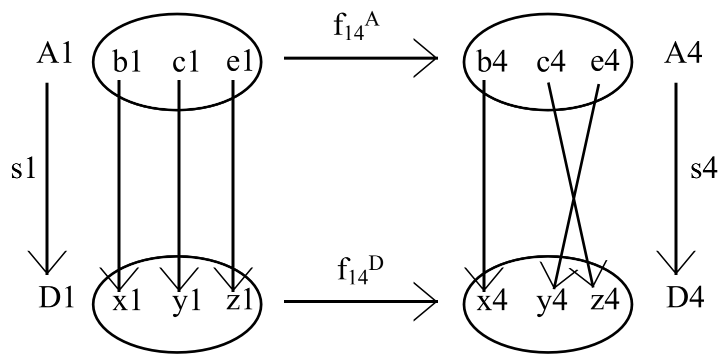


which, by virtue of being a map, satisfies $s_1 f_{41}^A = f_{41}^D s_4$ and $t_1 f_{41}^A = f_{41}^D t_4$ and along with $f_{14} = \langle f_{14}^A, f_{14}^D \rangle$ satisfying

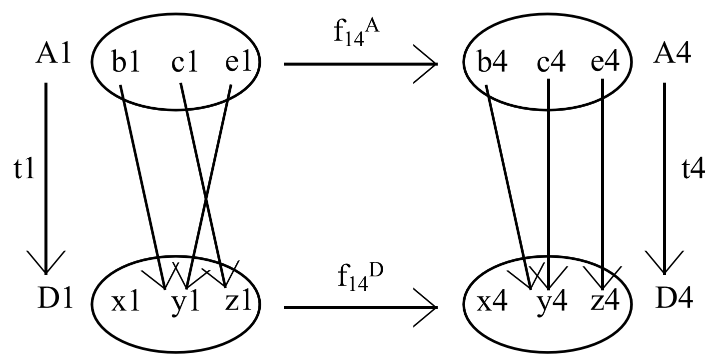
$f_{41}^A f_{14}^A = 1_{A1}$ and $f_{41}^D f_{14}^D = 1_{D1}$

$f_{14}^A f_{41}^A = 1_{A4}$ and $f_{14}^D f_{41}^D = 1_{D4}$

Let's start with $f_{14} = <f_{14}^A, f_{14}^D>$ depicted as



and



satisfying $s_4 f_{14}^A = f_{14}^D s_1$ and $t_4 f_{14}^A = f_{14}^D t_1$

which is short-hand for a more verbose statement saying for $f_{14} = <f_{14}^A, f_{14}^D>$ to be a map in the category of graphs, the functions f_{14}^A and f_{14}^D should be such that if the function f_{14}^A: A1 --> A4 assigns an arrow in the codomain set A4 = {b4, c4, e4} to an arrow in the domain set A1 = {b1, c1, e1}, then the function f_{14}^D: D1 --> D4 must assign a dot in the codomain set D4 = {x4, y4, z4} to each one of the dots in the domain set D1 = {x1, y1, z1} in such a way so as to preserve the source, target relations of arrows in the domain graph (A1, D1) in the codomain graph (A4, D4) (pardon me for being cryptic here; gettin lazy).

Once we find a pair of functions $<f_{14}^A, f_{14}^D>$ satisfying

$s_4f_{14}^A = f_{14}^Ds_1$ and $t_4f_{14}^A = f_{14}^Dt_1$

we, then, have to find another pair of functions $<f_{41}^A, f_{41}^D>$ satisfying

$s_1f_{41}^A = f_{41}^Ds_4$ and $t_1f_{41}^A = f_{41}^Dt_4$

Then we have to see if the maps f_{14} and f_{41} are inverses of one another satisfying

$f_{41}^Af_{14}^A = 1_{A1}$ and $f_{41}^Df_{14}^D = 1_{D1}$

$f_{14}^Af_{41}^A = 1_{A4}$ and $f_{14}^Df_{41}^D = 1_{D4}$

Once we have an isomorphism between graph 1 and graph 4, that is once we have seen that graph 1 is isomorphic to graph 4 (we also have seen that graph 4 is isomorphic to graph 1, which is reminiscent of saying *saying A = B is same as saying B = A*; here it may be of some interest to note cases wherein, going by some metric, for example, dog may be similar to animal without necessarily asserting that animal is similar to dog; think of arrow vs. loop also), we have to show that graph 1 is not isomorphic to the other four graphs (graph 2, graph 3, graph 5, and graph 6); while we are at it we might as well show that graph 4 is also not isomorphic to graph 2, graph 3, graph 5, and graph 6, which subliminally reads like we are too comfy in here and are somewhat little less than enthusiastic to face the unfamiliar universal properties of the familiar addition (1 + 1 = 2) as if afraid of something short of an excursion from

1 + 1 = 2

to

1 apple + 1 orange = 2 fruits

in thought.

Category of Idempotents

The category of idempotents figures prominently in the axiomatic study of QUALITY: "the fundamental quality type consists of the actions of just one idempotent" (Lawvere, 2007, p. 48). Idempotents are endomaps

$$e: A \to A$$

satisfying

$$ee = e$$

(see Conceptual Mathematics, p. 54). A simple example of idempotent endomap arises when sorting a collection A of people based on their gender G (= {female, male}) i.e. a retract

$$r: A \to G$$

and then giving an example of female, male person in A i.e. a section

$$s: A \leftarrow G$$

Composing the section *s* with the retract *r* gives an idempotent endomap

$$sr: A \to G \to A = e: A \to A$$

which maps each person in A to the 'prototype' in A of that person's gender (in G; see Child's problem in Conceptual Mathematics, p. 106). With idempotents (such as the above *e*) as objects and maps (between these objects) preserving the idempotent structure as morphisms, we obtain a category of idempotents (Conceptual Mathematics, pp. 138-9). The idempotent endomaps need not necessarily be maps of sets (as in above example). In the following we will see how actions of just one idempotent (in a category) result in a category of idempotents.

Given a space

$$S$$

(an object in a category) and a distinguished point

$$s\colon \mathbf{1} \to S$$

in S (where **1** is the terminal object of the category), we obtain a generic idempotent

$$e\colon S \to S = S - r \to \mathbf{1} - s \to S$$

(mapping every point in the space S to the distinguished point in S) satisfying

$$ee = [srsr = s1_1r = sr =]\ e$$

since

$$rs\colon \mathbf{1} \to S \to \mathbf{1} = 1_1$$

Given an idempotent

$$e\colon S \to S$$

(in a category), any object

$$A$$

in the category gives rise to an idempotent structure on A and any map

$$f\colon A \to B$$

gives rise to a structure-preserving morphism of the corresponding idempotents as follows (Conceptual Mathematics, pp. 150-1). Any S-shaped figure in A

$$a\text{: S –> A}$$

can be pre-composed with the given idempotent

$$e\text{: S –> S}$$

to get an S-shaped figure in A

$$\text{S – } e \text{ –> S – } a \text{ –> A = } ae\text{: S –> A}$$

Thus the idempotent

$$e\text{: S –> S}$$

induces an endomap

$$e^*_A\text{: } A^S \text{ –> } A^S$$

assigning an S-shaped figure in A (in the set A^S) to each S-shaped figure in A (in A^S) i.e.

$$e^*_A\ (a\text{: S –> A}) = ae\text{: S –> A}$$

Now we check to see if the induced endomap

$$e^*_A\text{: } A^S \text{ –> } A^S$$

is an idempotent i.e. satisfies

$$e^*_A\ e^*_A = e^*_A$$

e^*_A (a: S –> A) = ae

$e^*_A\, e^*_A\, (a\colon S \to A) = e^*_A\, (ae) = aee = ae$

since $ee = e$ and hence $e^*_A\, e^*_A = e^*_A$

Thus the given idempotent endomap

$$e\colon S \to S$$

induces on every object

$$A$$

(of the category) a corresponding idempotent structure

$$e^*_A\colon A^S \to A^S$$

Next, we see how every map

$$f\colon A \to B$$

in the category gives rise to an idempotence-preserving morphism

$$e^*_B\, f = f\, e^*_A$$

by the associative law (where $e^*_B\colon B^S \to B^S$ satisfying $e^*_B\, e^*_B = e^*_B$ is the idempotent structure corresponding to the object B).

Any S-shaped figure in A

$$a\colon S \to A$$

can be first post-composed with the map

$$f\colon A \to B$$

to obtain a S-shaped figure in B

$$fa\colon S \to B$$

which can then be pre-composed with the given idempotent

$$e\colon S \to S$$

to obtain another S-shaped figure in B

$$(fa)e\colon S \to B$$

Alternatively, any S-shaped figure in A

$$a\colon S \to A$$

can be first pre-composed with

$$e\colon S \to S$$

to obtain another S-shaped figure in A

$$ae\colon S \to A$$

which can then be post-composed with

$$f\colon A \to B$$

to obtain a S-shaped figure in B

$$f(ae)\colon S \to B$$

These two sequences of composing maps can be depicted as two paths to go from the set A^S of S-shaped figures in A to the set B^S of S-shaped figures in B as in

A^S —*f*—> B^S

^ ^

| |

e^*_A e^*_B

| |

A^S —*f*—> B^S

with Right-Up path giving

$$e^*_B\, f\,(a) = e^*_B\,(fa) = (fa)e$$

and Up-Right path giving

$$f\, e^*_A\,(a) = f(ae)$$

Since

$$(fa)e = f(ae)$$

by the associative law, we have

$$e^*_B\, f = f\, e^*_A$$

and the commutativity of the above diagram preserving the idempotent structure.

Unlike this case of actions of single idempotent (resulting in a category of idempotents), actions of two idempotents give rise to a different structure: category of reflexive graphs. Comparing and contrasting these two categories

Category of Reflexive Graphs vs. Category of Idempotents

in terms of their truth value objects and the behavior of functors to and from these categories to the category of sets can be helpful in understanding the contrast

COHESION vs. QUALITY

(or that’s a plan :)

Life after death

Speaking of 'life after death', there are four possibilities:

1. Life does exist after death.
2. Life does not exist after death.
3. Life does and does not exist after death.
4. Life neither does nor does not exist after death.

The first two are enough, tells me my nephew Dheeraj (see also http://aeon.co/magazine/philosophy/logic-of-buddhist-philosophy/). Putting aside, for a moment, the content (life, death, time, existence, etc.), we find, upon looking at the form of that which is stated:

1. A
2. not A
3. A and not A
4. not A and not not A

To make sense of

A and not A

think of boundary (Sets for Mathematics, p. 201). To find scenarios where

$$\text{not not } A \neq A$$

all we need to do is imagine a bunch of dots and arrows (Conceptual Mathematics, p. 355).

But… still… all this doesn't really sound real ;) Well, then; what's more real than perception:

$$\text{Physical stimuli} \to \text{Brain} \to \text{Conscious Experience}$$

or communication:

X – coding –> Y – decoding –> Z

These 2-step (3-stage) processes can be objectified as

$$A - f \to B - g \to C$$

an object made up of three sets and two [structural] functions. The logic (of life and death) that got us into this is, as we'll see, about parts of objects such as

$$A - f \to B - g \to C$$

Parts of an object X are monomorphisms with the object X as codomain (Conceptual Mathematics, pp. 335-57). But first, we need to define maps. A map from one object

$$A - f \to B - g \to C$$

to another object

$$A' - f' \to B' - g' \to C'$$

in this category of 3-stage variable sets is a triple of functions

$$p: A \to A',\ q: B \to B',\ r: C \to C'$$

satisfying two equations

$$q\,f = f'\,p,\ r\,g = g'\,q$$

making the two squares in the diagram

C – *r* –> C’

^

|

g *g*’

|

B – *q* –> B’

^

|

f *f*’

|

A – *p* –> A’

commute (Conceptual Mathematics, pp. 149-150).

Next we need to calculate the truth value object of this category of 3-stage variable sets. It is this truth value object that’ll, hopefully, help us make sense of the four possibilities with which we started. Note that in the case of the category of sets, the set

$$\Omega = \{\text{false}, \text{true}\}$$

is the truth value object. If we limit our discourses to these categories of sets, then all that there is

'A' or 'not A'

and

'A and not A'

makes no sense and

not not A = A

How are we going to calculate the truth value object of the category of 3-stage variable sets? Let's start with, say, the terminal object

$$\mathbf{1} - 1_1 \to \mathbf{1} - 1_1 \to \mathbf{1}$$

(where **1** = {•}) and look at its parts. In some categories (such as the category of sets and 2-stage variable sets; see Sets for Mathematics, pp. 114-9) points of the truth value object are in 1-1 correspondence with parts of the terminal object. Parts of

$$\mathbf{1} - 1_1 \to \mathbf{1} - 1_1 \to \mathbf{1}$$

are the following:

Part 1.

$$\begin{array}{ccccc} \mathbf{1} & \to & \mathbf{1} & \to & \mathbf{1} \\ \\ \mathbf{0} & & \mathbf{0} & & \mathbf{0} \end{array}$$

Part 2.

$$\begin{array}{ccccc} \mathbf{1} & \to & \mathbf{1} & \to & \mathbf{1} \\ & & & & \uparrow \\ \mathbf{0} & & \mathbf{0} & & \mathbf{1} \end{array}$$

Part 3.

$$\begin{array}{ccccc} \mathbf{1} & \to & \mathbf{1} & \to & \mathbf{1} \\ & & \uparrow & & \uparrow \\ \mathbf{0} & & \mathbf{1} & \to & \mathbf{1} \end{array}$$

Part 4.

$$\begin{array}{ccccc} \mathbf{1} & \to & \mathbf{1} & \to & \mathbf{1} \\ \uparrow & & \uparrow & & \uparrow \\ \mathbf{1} & \to & \mathbf{1} & \to & \mathbf{1} \end{array}$$

These four parts correspond to the four truth values (about forms of life after death ;)

1. true
2. not true
3. true and not true
4. not true and not not true

but which one to which? It all is not as clear as I wish it were (see http://tlvp.net/~fej.math.wes/SIPR_AMS-IndiaDoc-MSIE.htm). In any case, here's the truth value object (of 3-stage variable sets)

$$\mathbf{4} -j\to \mathbf{3} -k\to \mathbf{2}$$

where $\mathbf{2} = \{0, 1\}$, $\mathbf{3} = \{0, 0_1, 1\}$, $\mathbf{4} = \{0, 0_1, 0_2, 1\}$

with $k(0) = 0$, $k(0_1) = 1$, $k(1) = 1$

$j(0) = 0$, $j(0_1) = 0_1$, $j(0_2) = 1$, $j(1) = 1$

Let's now look at the correspondence between parts of the terminal object and maps from the terminal object to the truth value object:

Part 1.

1 → 1 → 1

0 0 0

4	– *j* –>	**3**	– *k* –>	**2**
^		^		^
|		|		|
w		*v*		*u*
|		|		|
1	– 1_1 –>	**1**	– 1_1 –>	**1**

u (•) = 0, *v* (•) = 0, *w* (•) = 0

Part 2.

$$\mathbf{1} \to \mathbf{1} \to \mathbf{1}$$

$$\uparrow$$

$$\mathbf{0} \quad \mathbf{0} \quad \mathbf{1}$$

$$\begin{array}{ccccc}
\mathbf{4} & -j \to & \mathbf{3} & -k \to & \mathbf{2} \\
\wedge & & \wedge & & \wedge \\
| & & | & & | \\
w & & v & & u \\
| & & | & & | \\
\mathbf{1} & -1_1 \to & \mathbf{1} & -1_1 \to & \mathbf{1}
\end{array}$$

$u(\bullet) = 1$, $v(\bullet) = 0_1$, $w(\bullet) = 0_1$

Part 3.

$$\mathbf{1} \rightarrow \mathbf{1} \rightarrow \mathbf{1}$$

$$\uparrow \quad \uparrow$$

$$\mathbf{0} \quad \mathbf{1} \rightarrow \mathbf{1}$$

4	– *j* –>	**3**	– *k* –>	**2**
^		^		^
|		|		|
w		*v*		*u*
|		|		|
1	– 1_1 –>	**1**	– 1_1 –>	**1**

$u\ (\bullet) = 1$, $v\ (\bullet) = 1$, $w\ (\bullet) = 0_2$

Part 4.

$$\mathbf{1} \rightarrow \mathbf{1} \rightarrow \mathbf{1}$$

$$\uparrow \quad \uparrow \quad \uparrow$$

$$\mathbf{1} \rightarrow \mathbf{1} \rightarrow \mathbf{1}$$

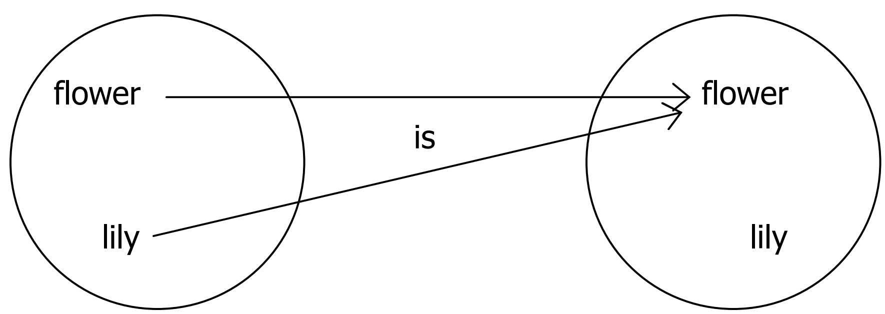
4	– *j* –>	**3**	– *k* –>	**2**
^		^		^
|		|		|
w		*v*		*u*
|		|		|
1	– 1_1 –>	**1**	– 1_1 –>	**1**

u (•) = 1, v (•) = 1, w (•) = 1

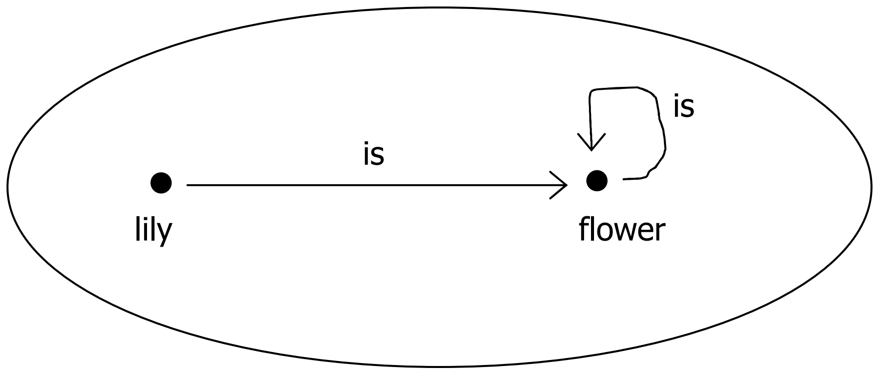
Categorical Perception and Part-Whole Relations

Let's begin with function f: A → B, where A is a set called domain and B is a set called codomain. It is very important to note that both the domain and the codomain are integral to the function just as the two end-points of a line segment are integral to the line. We may also want to note that the domain and codomain need not be different sets. When the domain and codomain are the same set A, then the function f: A → A is called endofunction (also known as endomap).

Let's look at a simple endomap, f: A → A, with A = {flower, lily}



In the above internal diagram the arrows can be interpreted as 'is', so that we can read the diagram as 'flower is flower' and 'lily is flower'. We can also draw the above internal diagram, in view of the fact that both domain and codomain are one and the same set, as follows:

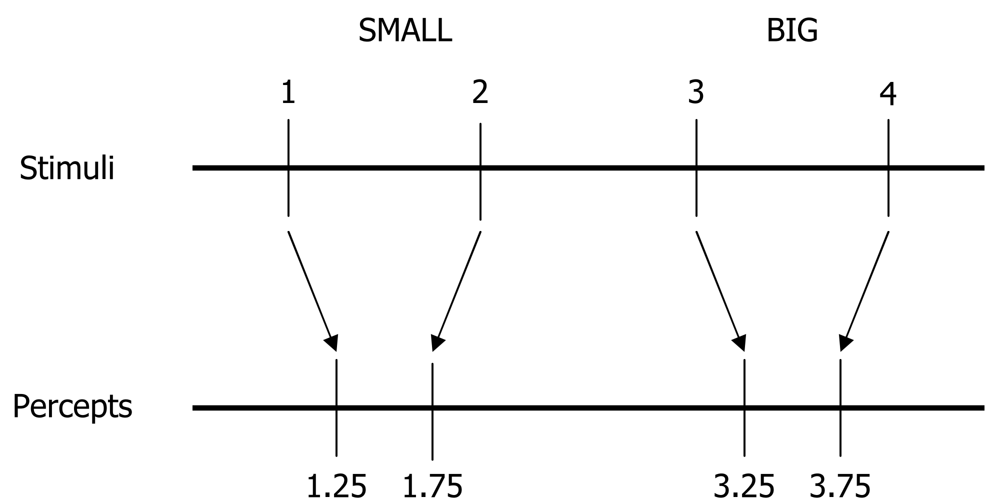


where dots stand for ‘lily’ and ‘flower’, and arrows denote ‘is’.

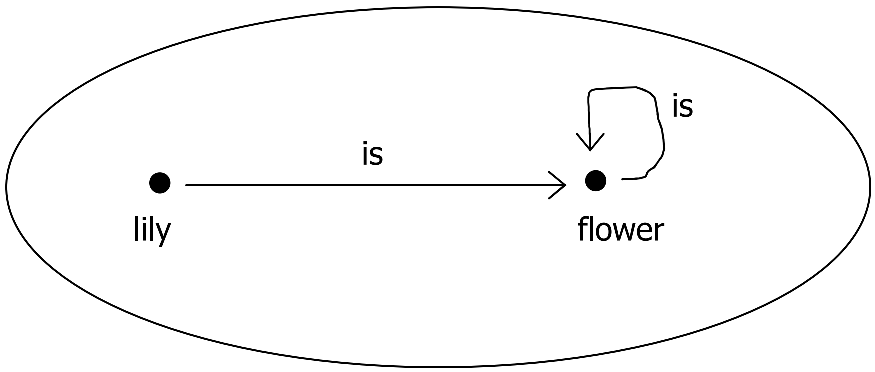
Switching gears slightly, the above diagram happens to be a model of perception, wherein particulars such as ‘lily’ are perceived as exemplars of the corresponding category such as ‘flower’. Let’s make the model little bit more concrete by considering a numerical example. Consider the concrete particulars or physical stimuli, S = {1, 2, 3, 4}, where the elements of S can be thought of as intensities of light, or heights of people, etc. Now consider two abstract generals, say, ‘SMALL’ and ‘BIG’, which we can treat as the names of two categories. To each of these abstract generals there corresponds a concrete general, which can be thought of as the prototype of the corresponding category. The prototypes corresponding to the categories ‘SMALL’ and ‘BIG’ are 1.5 and 3.5, respectively (which are simply the averages of concrete particulars falling under the corresponding categories). Now, let’s depict in tabular form the process of going from physical stimulus to perceptual experience, as follows:

Stimuli (concrete particulars)	1	2	3	4
Categories (abstract generals)	SMALL		BIG	
Prototypes (concrete generals)	1.5 (1 + 2)/2		3.5 (3 + 4)/2	
Percepts	1.25 (1 + 1.5)/2	1.75 (2 + 1.5)/2	3.25 (3 + 3.5)/2	3.75 (4 + 3.5)/2

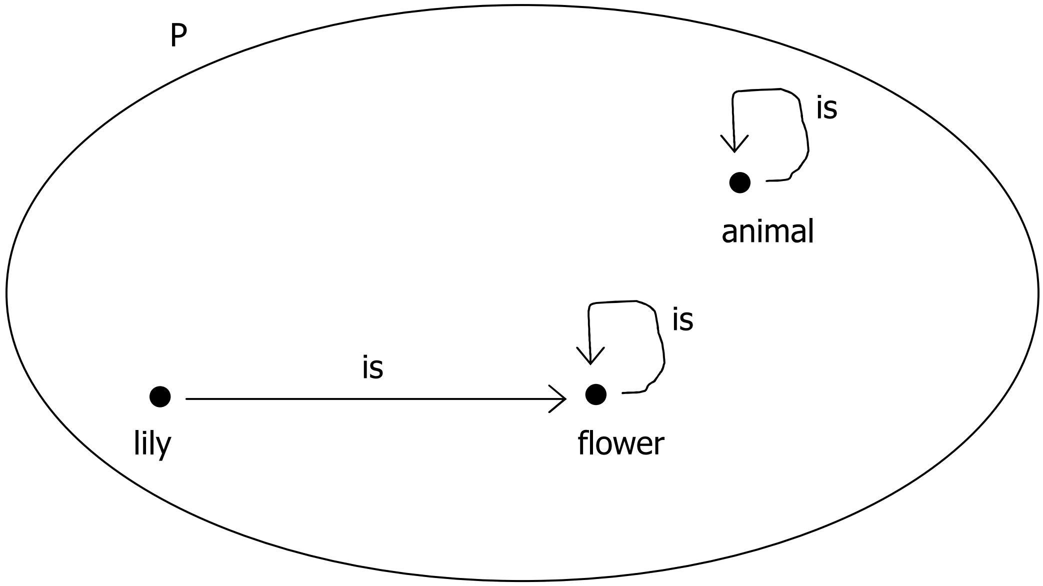
To see more clearly the transformation from physical stimuli to percepts:



From the above stimulus-percept transformation we can see why all the bananas look alike and unlike apples. Summing up, we perceive particular physical stimuli in terms of mental categories (such as SMALL and BIG) to which they belong. All of the involved processes can be abstracted in the following simplified internal diagram (reproduced from earlier):

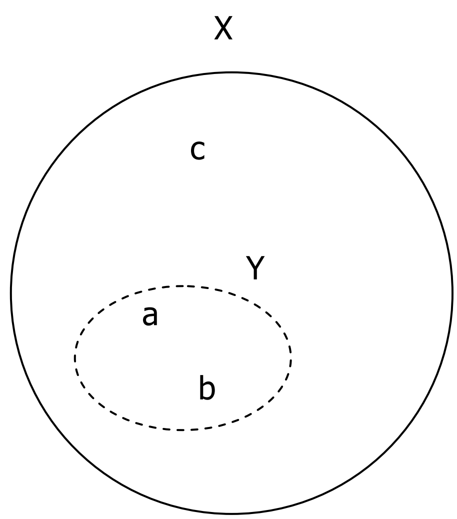


Now to make things little bit more interesting, consider the following perceptual universe:



Let's try to find out its logical structure. Before we go any further, let's set our words straight: 'what do we mean by 'logic'?' We can think of logic as the algebra of parts or even more plainly as part-whole relations.

Let's begin simple. Consider a set X = {a, b, c}, and a part of X, say, Y = {a, b} as depicted below:



Now if we ask 'is, say, 'a' in Y?'

We get the answer: YES

If we ask 'is 'c' in Y?'

We get the answer: NO

These are the only two possible relations that a structure-less element such as 'a' can have with respect to a discrete set such as Y. This is our familiar Boolean logic with its truth-value object of two elements:

Ω_B = {true, false}.

If we note

$$true = not\ (false)$$

$$false = not\ (true)$$

we find that

$$not\ (not\ (A)) = A$$

where A is an element of the truth-value object Ω_B.

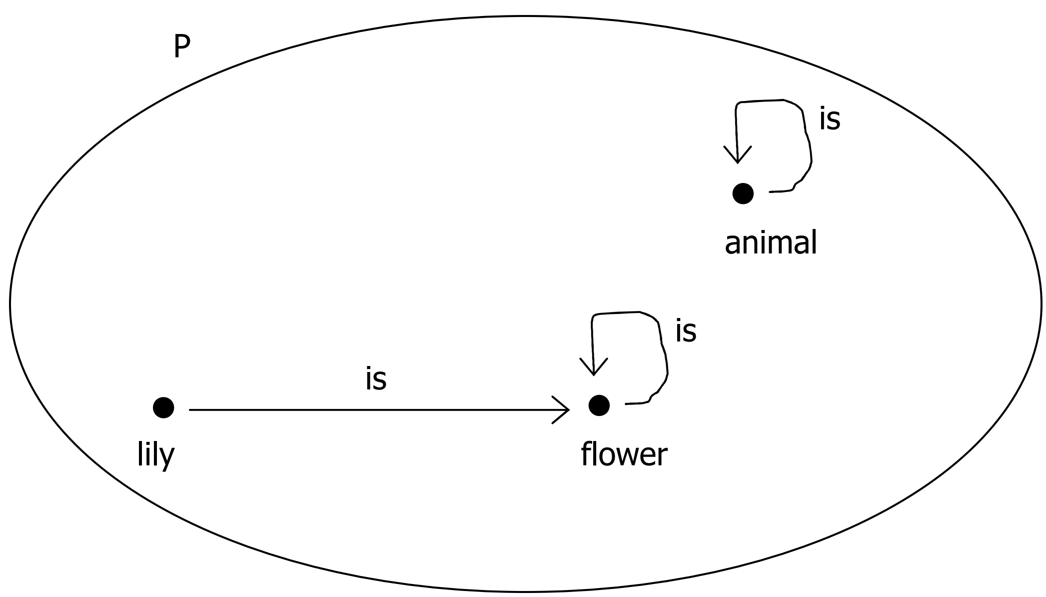
For example,

$$not\ (not\ (true)) = not\ (false) = true$$

similarly,

$$not\ (not\ (false)) = not\ (true) = false$$

Now let's go back to our perceptual universe P (shown below) and try to characterize its logical structure.



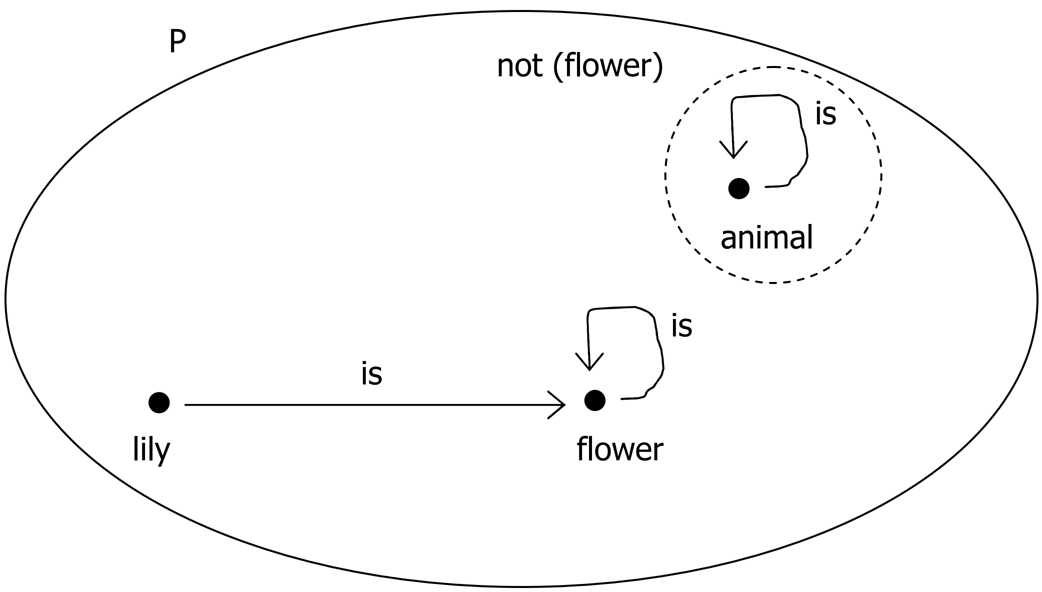
We have just noted that in Boolean algebra

not (not (A)) = A

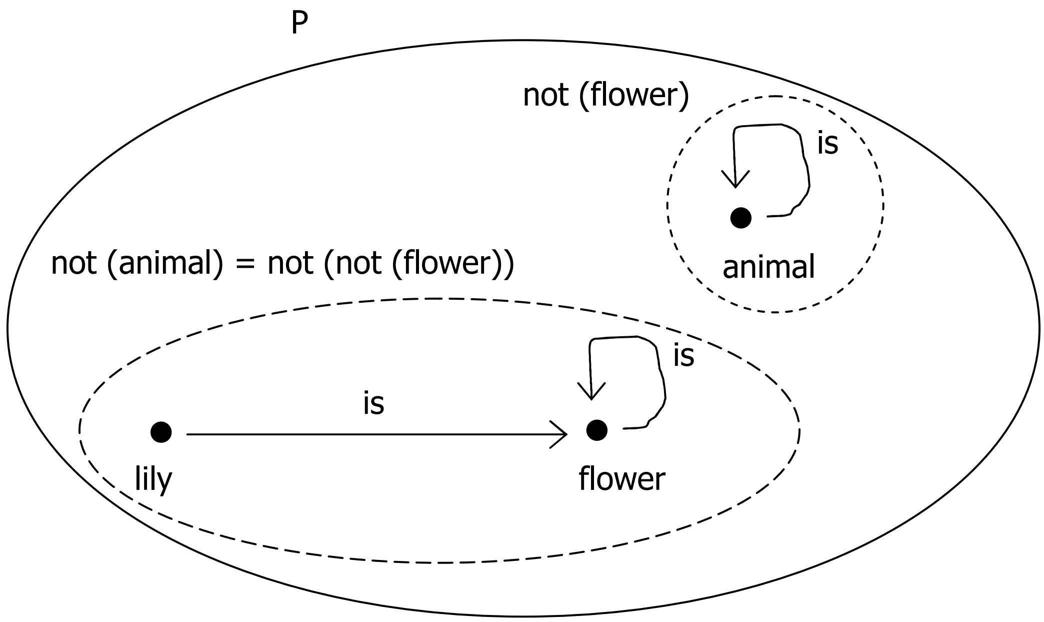
Now let's see if this identity or the law of excluded-middle (either YES or NO; nothing in between) holds water in our perceptual universe P.

Let's begin with 'flower' and ask 'what is 'not (flower)'?' in our perceptual universe P. Looking at the above diagram

not (flower) = animal



Now if we ask: 'what is 'not (animal)'?' we find that 'not (animal)' or 'not (not (flower))' is not just 'flower', but also 'lily' as illustrated in the following internal diagram:



Thus we find that

not (not (flower)) ≠ flower

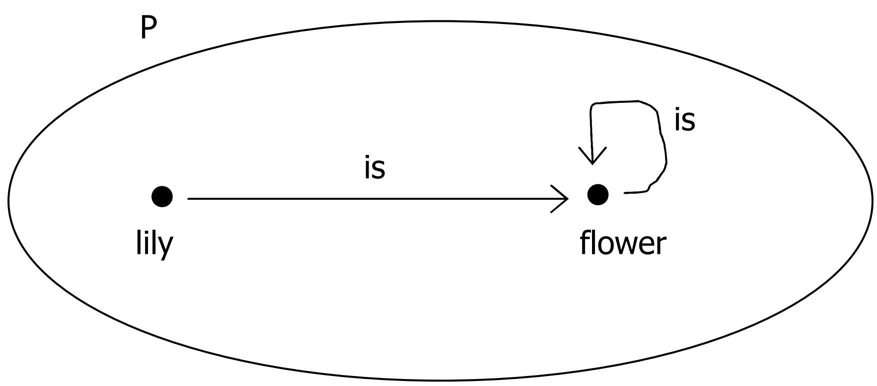
The above case of not (not (A)) ≠ A is just an illustration of Heyting algebra.

Now let's ask, 'how about truth-value object?'

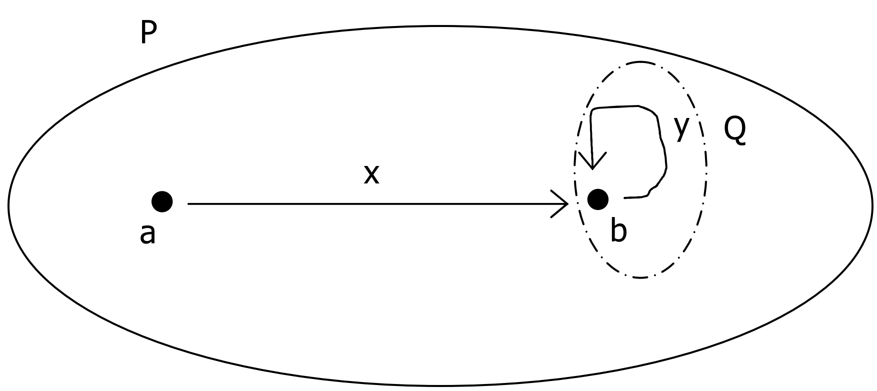
In the case of structure-less elements, a truth-value object:

Ω_B = {true, false}

of two elements suffices to capture all possible relations a part may have with respect to a whole. Since we are dealing with more structured objects (arrows with a source and a target) in the case of our perceptual universe, we can guess that the truth-value object in all likelihood needs more than 2 elements to capture the part-whole relations in the perceptual universe P. Let's redraw a simple perceptual universe as below:



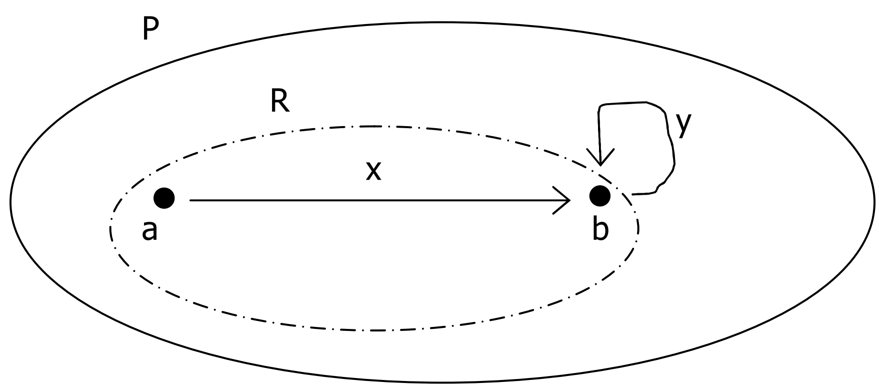
In our universe P, there are dots, and then there are arrows with a source and a target. First let's consider the case of dots in a simplified (in terms of labeling) version of P:



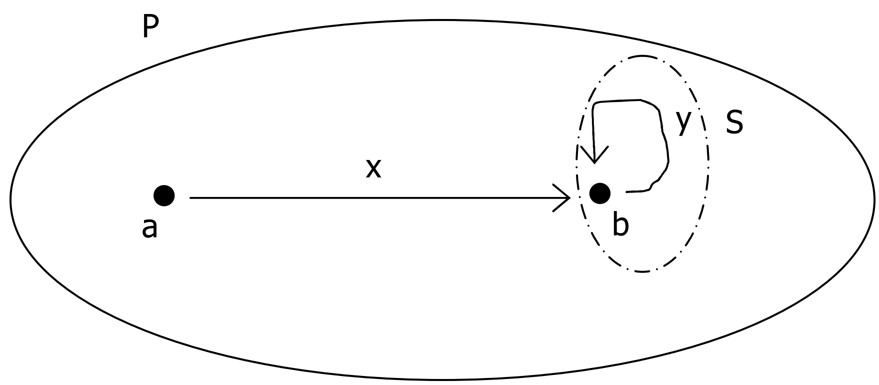
Is dot ‘b’ in ‘Q’? YES

Is dot ‘a’ in ‘Q’? NO

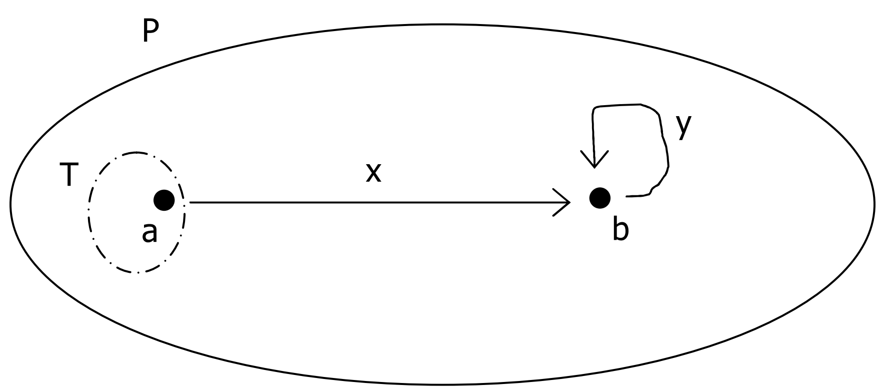
So in the case of dots, the part-whole relations can be captured with two elements: YES, NO. Now let’s look at the case of arrows:



1. The arrow and its source and its target are in part R; e.g. arrow x, its source a, and its target b are in R.
2. The arrow is not in R, but its source and its target are in R; e.g. arrow y is not in R, but its source b and its target b are in R.

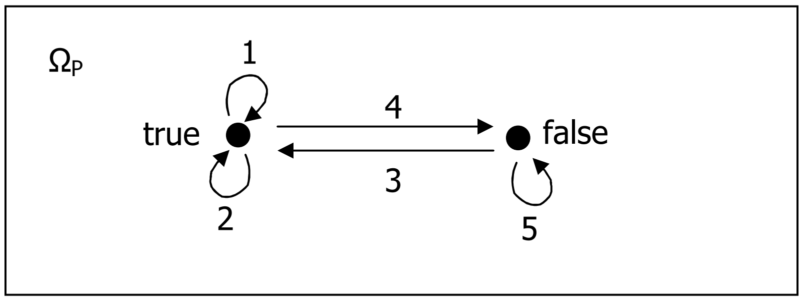


3. The arrow and its source are not in S, but its target is in S; e.g. arrow x and its source a are not in S, but its target b is in S.

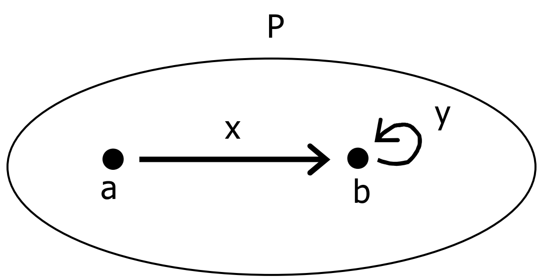


4. The arrow and its target are not in part T, but its source is in T; e.g. arrow x and its target b are not in T, but its source a is in T.
5. The arrow, its source, and its target are not in T; e.g. arrow y, its source b, and its target b are not in T.

Summing up, the truth-value object of our perceptual universe P has a total of 7 elements: 2 for dots, and 5 for arrows, which can graphed as follows:



Question: What would be the truth-value object of percept P (the following diagram) taken as a unitary whole (the way we took the arrow along with its source and target as a unitary whole)?



The other day I was leafing through a newspaper where I found, of all the things that could possibly find me, the following.

Find the subsets of

animal = {cat, dog, elephant}

I thought, somewhat reluctant to do the exercise, it must be same as the number of functions; but from where to where I couldn't think of. Halfway through, as I was reading, I thought there must be a formula for the number of subsets of a set; but, as usual, I couldn't remember. So, grudgingly, I went through the motions.

1. {}
2. {cat}
3. {dog}
4. {elephant}
5. {cat, dog}
6. {dog, elephant}
7. {elephant, cat}
8. {cat, dog, elephant}

There are 8 subsets. Noting that the set we started with

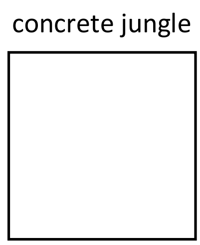
animal = {cat, dog, elephant}

has 3 elements, the writer of that kids column, after one or two more exercises, concluded with the formula for the number of subsets of a set. A set of n elements has 2^n subsets. So, in our case, it is $2^3 = 8$ subsets. No sooner had I said, to myself, fine—now I am free to find the classifieds section, which was the reason I bought the newspaper in the first place, it occurred to me that the '2' in $2^3 = 8$ is not as familiar as the '3' and '8'. I know where that '3' came from; it is the number of elements in the set animal = {cat, dog, elephant}. I also know what that '8' is: it is the number of subsets. But, what is 2? I couldn't help, for the simple reason that it gave the correct answer (speak of *no good deed goes un-suspect*), but question: wherefrom did '2' come from?

Truth about being

I don't know about you, but I like stories—silly stories—stories light on morals. Here's one.

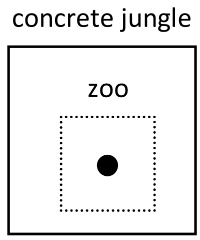
Once upon a time there was a concrete jungle.



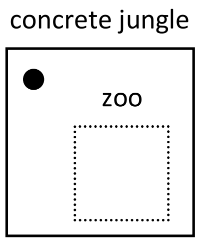
Within the concrete jungle there was an animal and a zoo. The animal, by virtue of being a being, can be and not (or is it 'be or not'; c.f. $2^{1+2} = 2^1 \times 2^2$, Conceptual

Mathematics, Exercise 7 on page 356) in the zoo, which allows us to ask: is the animal in the zoo?

If the animal, depicted, to save some ink, as a dot, is in the zoo, then we answer: YES.



If the animal is not in the zoo, then we answer: NO.



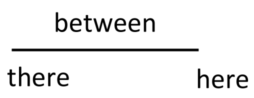
The 2 states, of being and not, in which a being can be are together known as the truth value object

2 = {true, false}

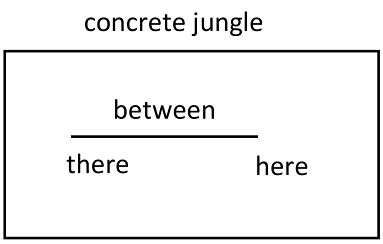
of the category of sets. Now we see where the '2' in $2^{|X|}$ subsets of set X came from.

Truth about between

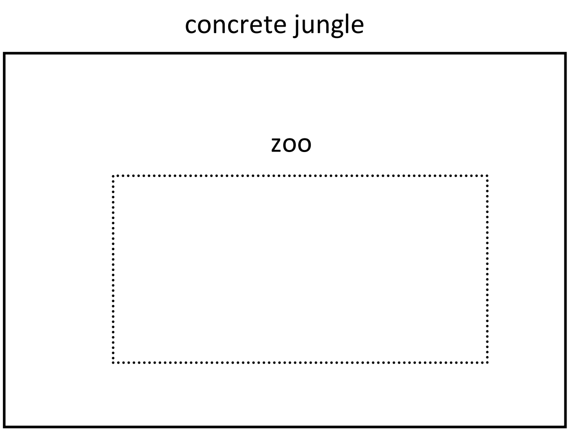
Let us say we own a property from there to here.



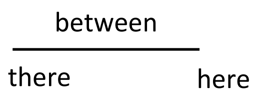
To be exact, we own the line 'between' from a point 'there' to a point 'here.' Let us say, since everything has to be somewhere—to be sensible—in some sense, that our property is in, being not that creative, the concrete jungle.



Let's say, again, that we have a zoo, housing some scientists, to spice-up our story, in the concrete jungle.

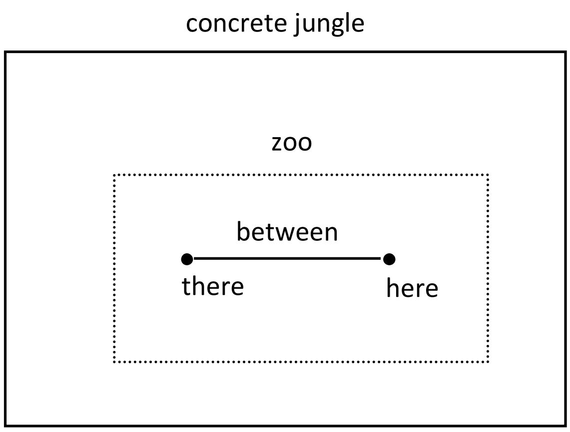


Now that there is a zoo in the concrete jungle and given that our property

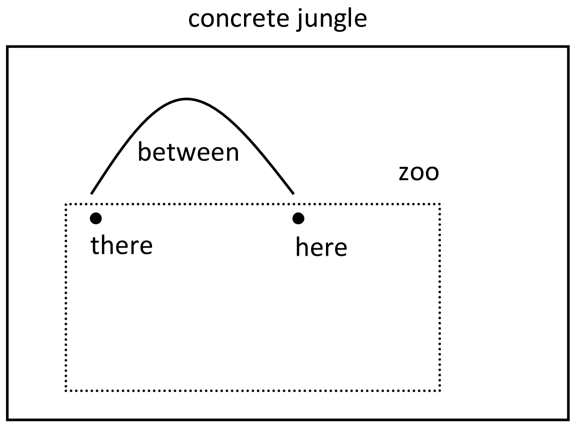


is also in the concrete jungle, it is only natural that we are concerned if our property is in the zoo. Well, there's nothing much we can do about it; if it's in, then it is in. If not, then not. Or, is it; is that all really? Let's see. Let us start with the worst-case scenario.

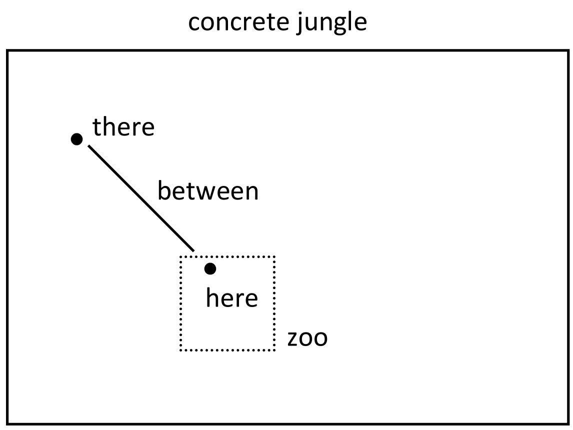
Case 1. Our line *between* along with its two end points *there* and *here* is in the zoo.



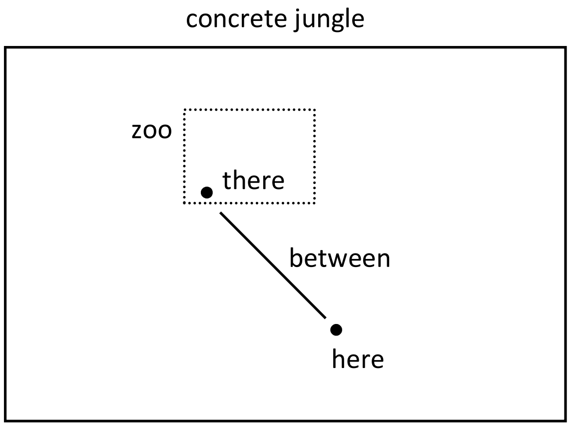
Case 2. Our line *between* is not in, but its end points *there* and *here* are in the zoo.



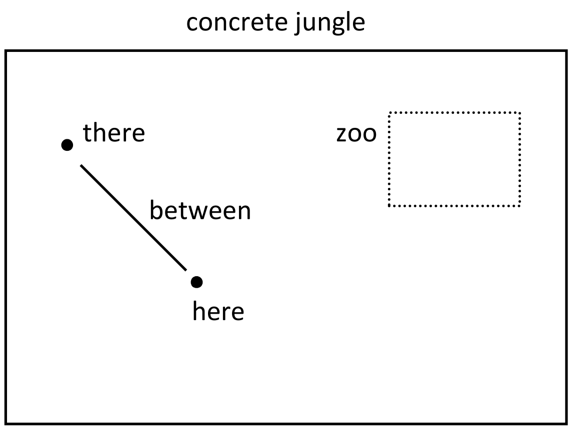
Case 3. Our line *between* and one of its end points *there* are not in, but the other end point *here* is in the zoo.



Case 4. Our line *between* and one of its end points *here* is not in, but the other end point *there* is in the zoo.



Case 5. Luckily our line *between* along with both of its end points *there* and *here* are not in the zoo.

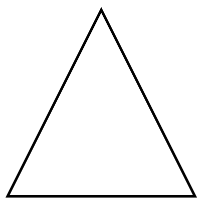


Summing up for now, we say, that in the case of lines there are 5 levels of truth in addition to the **2** = {true, false} of points. The truth value object of the category of graphs, based on the reasoning sketched above, has 7 (5 + 2) elements (Conceptual Mathematics, page 345).

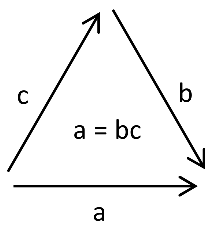
Truth about triangles

Given our geometrical ancestry, it is not only natural but also healthy to ask, especially after having looked at the truth about points and about lines, 'what about the truth about surfaces?'

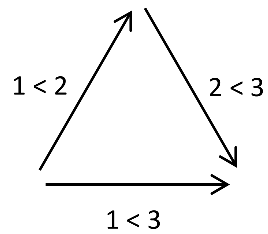
Let's, having realized simple is a mantra more powerful than any tantra, think of a simple surface—simpler than a square—a triangle with just the bare minimum of 3 edges needed to be a surface.



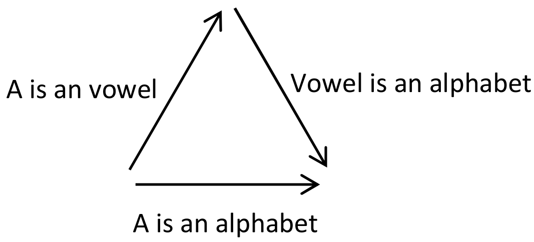
Before we get carried away with the truth about a triangle, let's take a moment and ask 'what can we do with it?' We can use it to depict composition



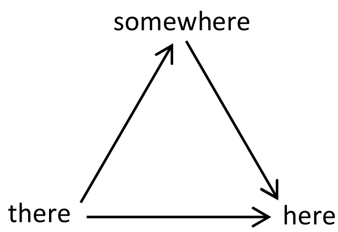
which we can interpret in a somewhat more familiar terms such as



or as



or as



This commutative triangle is something; if I am not mistaken it subsumes PRODUCT, SUM, and PERCEPTION (names of shapes of its parts) and I think it’ll be a while before we can get to the truth value of object of the commutative triangle.

Calculating Products

Structures such as sets have properties, whose values can be expressed as numbers. A property of a set A is its size |A|, which is a number. For example, the size of a set A = $\{a_1, a_2\}$ is |A| = 2. Analogous to the multiplication of numbers such as

$$2 \times 2 = 4$$

we can define product of structures which have these numbers as properties and in a way the size of product set is the product of sizes of the factor sets (Conceptual Mathematics, pp. 238-9):

$$|A \times B| = |A| \times |B|$$

In the present note, we calculate the product of two sets

$$A = \{a_1, a_2\} \text{ and } B = \{b_1, b_2\}$$

using the definition of product (Conceptual Mathematics, p. 217):

An object P along with a pair of maps

$$p_A: P \to A,\ p_B: P \to B$$

is called a product of the objects A and B if for every object Q and every pair of maps

$$q_A: Q \to A,\ q_B: Q \to B$$

there is exactly one map

$$r\colon Q \rightarrow P$$

satisfying the following two equations:

$$q_A = p_A \circ r$$

$$q_B = p_B \circ r$$

where ‘∘’ denotes composition of maps.

Let’s now carefully look at what the definition of product says. The product of two objects

A, B

is an object P along with a pair of projection maps

$$p_A\colon P \rightarrow A,\ p_B\colon P \rightarrow B$$

(satisfying certain conditions, which we will look at in a moment). Now, if we consider a map from an object Q to the product object P

$$r\colon Q \rightarrow P$$

we find that the map *r* can be composed with the pair of projection maps p_A, p_B to obtain a pair of maps from the object Q to the two factor objects A, B

$$Q - p_A \circ r \rightarrow A,\ Q - p_B \circ r \rightarrow B$$

Thus we find that each map from any object Q to the product object P

Q –> P

gives rise, by way of composition with the pair of projection maps, to a pair of maps from the object Q to the two factor objects

Q –> A, Q –> B

Next, since the object P is product, for any pair of maps to the two factors A, B

q_A: Q –> A, q_B: Q –> B

there is exactly one map

$<q_A, q_B>$: Q –> P

from the common domain object Q to the product object (satisfying the two equations given in the definition of product). Thus for each pair of maps (to the two factor objects A, B)

Q –> A, Q –> B

there is a map to the product object

Q –> P

Summing these together, we say that there is a 1-1 correspondence:

Q –> A, Q –> B

Q –> P

between maps to the product object P and pairs of maps to the factor objects A, B. This 1-1 correspondence can be used to calculate products as follows.

First, let's see what the definition of product means in the category of sets. The product of two sets

$$A = \{a_1, a_2\} \text{ and } B = \{b_1, b_2\}$$

is a set P along with a pair of functions

$$p_A: P \rightarrow A,\ p_B: P \rightarrow B$$

such that for every set Q and every pair of functions

$$q_A: Q \rightarrow A,\ q_B: Q \rightarrow B$$

there is exactly one function

$$r: Q \rightarrow P$$

satisfying the following two equations:

$$q_A = p_A \circ r$$

$$q_B = p_B \circ r$$

Let's now calculate the product of two sets:

$$A = \{a_1, a_2\} \text{ and } B = \{b_1, b_2\}$$

Calculation of the product involves calculating the product set P along with a pair of functions

$$p_A: P \rightarrow A,\ p_B: P \rightarrow B$$

Let's see how we can calculate the product set P, given factor sets A and B, using the above discussed 1-1 correspondence:

$$Q \to A,\ Q \to B$$

$$Q \to P$$

between pairs of maps to the factor sets A, B and maps to the product set P from any set Q.

First, recollect that any set is completely determined its points (Conceptual Mathematics, p. 245). So, the product set P, just as any other set, is completely determined by its points

$$p\colon \mathbf{1} \to P$$

Since points of the product set P are functions from the terminal set **1** (= {•}) to P, points are in 1-1 correspondence with pairs of points of the factors A, B:

$$\mathbf{1} \to A,\ \mathbf{1} \to B$$

$$\mathbf{1} \to P$$

Since we are given the factor sets:

$$A = \{a_1, a_2\} \text{ and } B = \{b_1, b_2\}$$

we first list out all four pairs of points of the two sets A, B:

$$a_1\colon \mathbf{1} \to A,\ b_1\colon \mathbf{1} \to B$$

$$a_1\colon \mathbf{1} \to A,\ b_2\colon \mathbf{1} \to B$$

$$a_2\colon \mathbf{1} \to A,\ b_1\colon \mathbf{1} \to B$$

$$a_2: \mathbf{1} \to A,\ b_2: \mathbf{1} \to B$$

Corresponding to these four pairs of points of the two sets A, B, we have the four points of the product set P:

$$(a_1, b_1): \mathbf{1} \to P$$

$$(a_1, b_2): \mathbf{1} \to P$$

$$(a_2, b_1): \mathbf{1} \to P$$

$$(a_2, b_2): \mathbf{1} \to P$$

Thus the product set P is a four-element set

$$A \times B = \{(a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2)\}$$

Next, since the product of two sets A and B is a set A x B along with two projection functions

$$p_A: A \times B \to A,\ p_B: A \times B \to B$$

we still have to find out the two functions p_A and p_B. In other words, we have to find out the values of the two functions at each one of the elements of the product set A x B:

$$p_A (a_1, b_1) = ?,\ p_A (a_1, b_2) = ?,\ p_A (a_2, b_1) = ?,\ p_A (a_2, b_2) = ?$$

$$p_B (a_1, b_1) = ?,\ p_B (a_1, b_2) = ?,\ p_B (a_2, b_1) = ?,\ p_B (a_2, b_2) = ?$$

Since elements of a set are points of the set, evaluation of a function at an element is pre-composition of the function with the point corresponding to the element (Conceptual Mathematics, pp. 230-1). For example,

$$p_A\ (\mathrm{a}_1, \mathrm{b}_1) = p_A \circ (a_1, b_1)$$

where $(\mathrm{a}_1, \mathrm{b}_1)$ is an element of the set A x B, while (a_1, b_1) is a function

$$(a_1, b_1)\text{: } \mathbf{1} \text{ –> A x B}$$

Since the point

$$(a_1, b_1)\text{: } \mathbf{1} \text{ –> P}$$

corresponds to a pair of points

$$a_1\text{: } \mathbf{1} \text{ –> A, } b_1\text{: } \mathbf{1} \text{ –> B}$$

and since the functions

$$p_A\text{: A x B –> A, } p_B\text{: A x B –> B}$$

are product projections, we have, by the definition of product,

$$a_1 = p_A \circ (a_1, b_1)$$

$$b_1 = p_B \circ (a_1, b_1)$$

Along similar lines, we find the values of the two functions

$$p_A\text{: A x B –> A, } p_B\text{: A x B –> B}$$

at every element of the product set

$$A \times B = \{(a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2)\}$$

as

$$p_A (a_1, b_1) = a_1, p_A (a_1, b_2) = a_1, p_A (a_2, b_1) = a_2, p_A (a_2, b_2) = a_2$$

$$p_B (a_1, b_1) = b_1, p_B (a_1, b_2) = b_2, p_B (a_2, b_1) = b_1, p_B (a_2, b_2) = b_2$$

Thus, using the definition of product, we calculated the product of two sets

$$A = \{a_1, a_2\} \text{ and } B = \{b_1, b_2\}$$

as a product set

$$A \times B = \{(a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2)\}$$

together with two projection functions

$$p_A: A \times B \to A, p_B: A \times B \to B$$

with values as

$$p_A (a_1, b_1) = a_1, p_A (a_1, b_2) = a_1, p_A (a_2, b_1) = a_2, p_A (a_2, b_2) = a_2$$

$$p_B (a_1, b_1) = b_1, p_B (a_1, b_2) = b_2, p_B (a_2, b_1) = b_1, p_B (a_2, b_2) = b_2$$

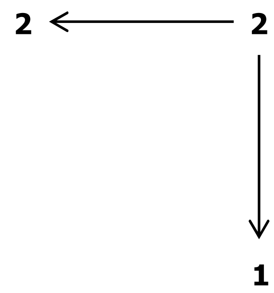
A category of pairs of maps (Conceptual Mathematics, pp. 255-6)

The title, as is often the case, doesn't give away all that's in store. It's not just about any pairs of maps that we are going to talk about. The lead characters are two objects—two sets: **1** = {.} and **2** = {., .}; and pairs of maps with the objects **1** and **2** as codomain objects are the objects of our category. Yes, of course we have to make sure we have everything we need to have a category (please see page 21 of the textbook). We will cross the bridge when we get to it (remembering not to burn the bridge after crossing; there sure will be visitations long after we are gone); we are, as we write, stepping towards it with a pair of maps X -> **1**, X -> **2** in hand, with the common domain set X left to imagination. However [if you want], we can name our category (we can always change if the desire ever arises). How about $\boldsymbol{C}_{1x2}$? I know it's somewhat cryptic… Ok, let's go with $\boldsymbol{C}_{1x2}$ for now.

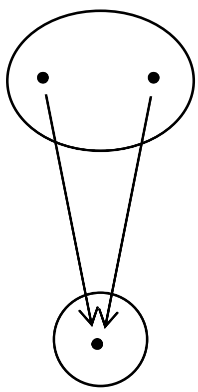
We have name: $\boldsymbol{C}_{1x2}$

We have objects: (X -> **1**, X -> **2**), (Y -> **1**, Y -> **2**), (Z -> **1**, Z -> **2**)…

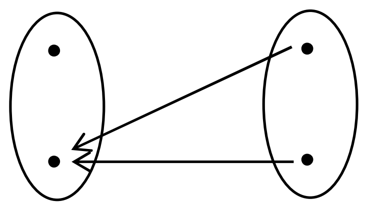
How many objects do we need in order to have a category? I don't like this question… it takes me places I really can't afford to go now (e.g. EMPTY SET *vs*. EMPTY CATEGORY; http://conceptualmathematics.wordpress.com/2012/09/18/empty-category/). Since you seem to be on a budget, let's content ourselves with pairs of maps from just one object (X = **2**) to the two codomain objects (**1** = {.}, **2** = {., .}).



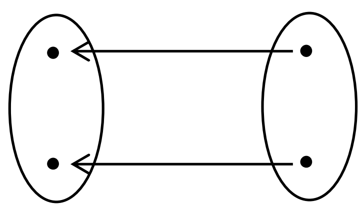
You mean just one object (one pair of maps)? That is an outstanding question: as of now we only settled on the common domain (**2**) and the codomains (**1**, **2**) of the pairs of maps. How many functions are there from the domain set **2** to the codomain set **1**? Just one: *f*: **2** -> **1**



How many functions are there from the domain set **2** to the codomain set **2**?



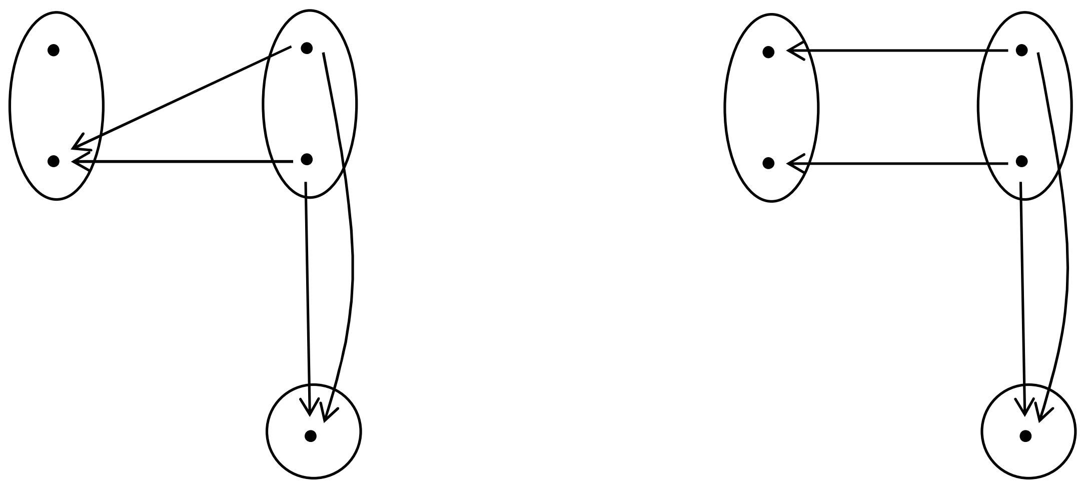
and



Exactly two! *g*: **2** -> **2** and *h*: **2** -> **2**

So we have 1 x 2 = 2 pairs (of maps) i.e. 2 objects in our category. Is this why we called our category $\boldsymbol{C}_{1x2}$? Yes, no, kinda sort of…

Of immediate importance are the two objects of our category $\boldsymbol{C}_{1x2}$ illustrated below:

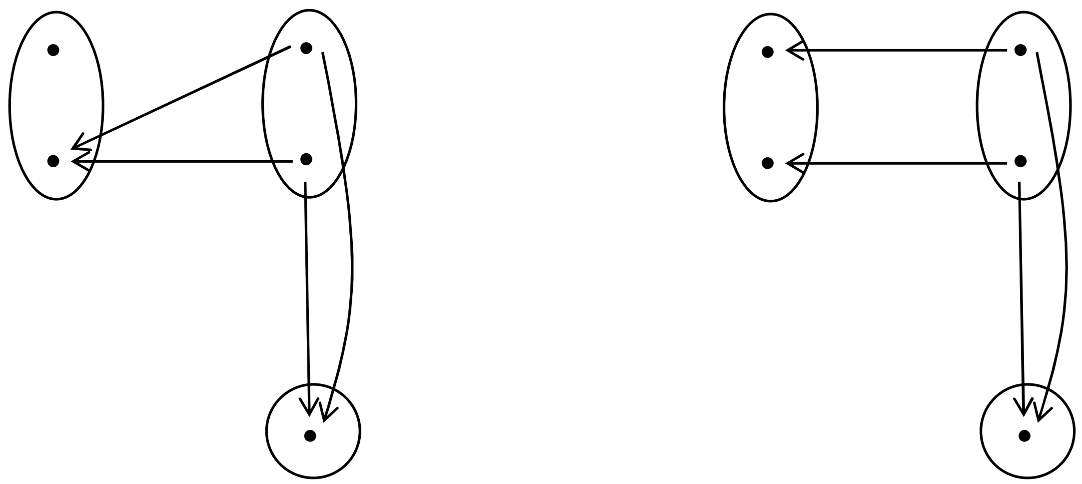


OK, we have two objects; where to next? Back in the day, when we recognized day and night as different—2 objects held in thought (though not like held-in-prison; more like holding carefully like a child holding a bubble)—we wondered how day becomes night and how night turns into day. That's all the intuition we need. We want none of that post-modern mumbo-jumbo in here.

In thinking about transforming our two objects—one into another—we think of maps *f* from object A to object B. Maps are loaded. We can think of *f*: A -> B as a transformation of A into B, a picture of A in B, a B-valued property on A (didn't I just say loaded)!

How many maps are there between our two objects?

A   B

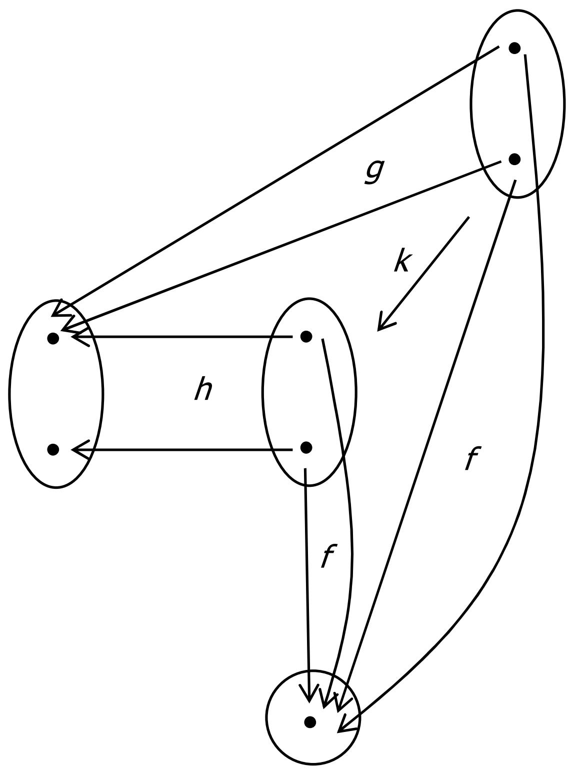


Aah; that incessant urge to count may in part be an expression of my eagerness to conclude and go home.

Before I can count, I need to know what is it that I am dying to count: maps in the category $\boldsymbol{C}_{1x2}$

What does a map *x*: A -> B in the category $\boldsymbol{C}_{1x2}$ look like? Well, it's a map of sets

k: **2** -> **2**



satisfying

f = *fk* and *g* = *hk*

There is exactly one map of sets *k* (= *g*: **2** -> **2**) satisfying the above two equations.

Sounds familiar—yes, 'there is exactly one map' from the definition of terminal object, and of product—our buddy B i.e. 1 x 2 = 2.

A Study of Function

Recently I came across a paper, according to which retrieving the material that's been read once is equivalent to re-reading the material 5 times or so. It seemed to make some sense in the sense retrieval inevitably highlights what I forgot, which in turn forces me to formulate questions such as 'what's that condition that has to be satisfied in order for the composite of two functions to be defined?', which in turn focuses my attention and help structure and glue the material to be learned into a coherent and cohesive unity.

Be that as it may, I thought of retrieving what I have been discussing for the past few weeks. Here I go. We have been talking a lot about functions such as

$$f: A \to B$$

also depicted as

$$A \xrightarrow{f} B$$

where A is the domain set and B is the codomain set. Even though 'A' and 'B' are depicted as disconnected from the arrow representing function, they i.e. domain A and codomain B are integral to the function f just as the end-points of a line-segment are integral to the line-segment. Yet another useful metaphor to keep in mind when thinking about functions is to think of a function as a journey 'j' with domain and codomain of the function corresponding to beginning (e.g. La Jolla) and destination (e.g. Amsterdam) of the journey,

$$j\text{: La Jolla} \rightarrow \text{Amsterdam}$$

also depicted as

$$\text{La Jolla} \xrightarrow{\quad j \quad} \text{Amsterdam}$$

We also noted that domain and codomain sets, and sets in general can be identified with identity functions such as $1_A: A \rightarrow A$, which when translated to our line-segment metaphor says that the end-points of a line-segment can be thought of as line-segments of zero length. In terms of our journey metaphor, the beginning and destination can be thought of as journeys that go nowhere (or stay wherever they are; $1_{\text{La Jolla}}$: La Jolla → La Jolla). When we put down or formalized our thought of thinking of a set as an identity function, we found ourselves on the one hand simplifying the conceptual repertoire needed to speak of functions; on being able to speak of functions in terms of functions alone, albeit special functions, in the sense we can now say that a function has an identify function as domain and an identity function as codomain as in,

$$f: 1_A \rightarrow 1_B$$

and on the other hand confronting a problem as depicted below:

$$A \xrightarrow{\quad 1_A \quad} A \xrightarrow{\quad f \quad} B \xrightarrow{\quad 1_B \quad} B$$

Looking at the above diagram, in an effort to make sense of it, one immediate question we had was 'how do we put-together or compose two functions?' Here again we found that our journey metaphor is instructive. To elaborate, consider two journeys

j: La Jolla → Amsterdam

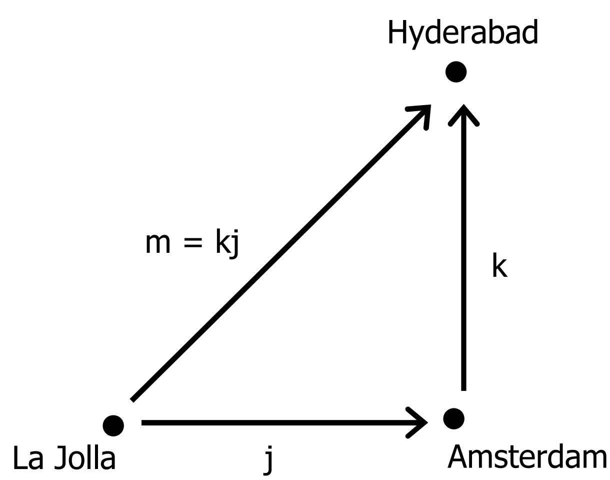
and

k: Amsterdam → Hyderabad

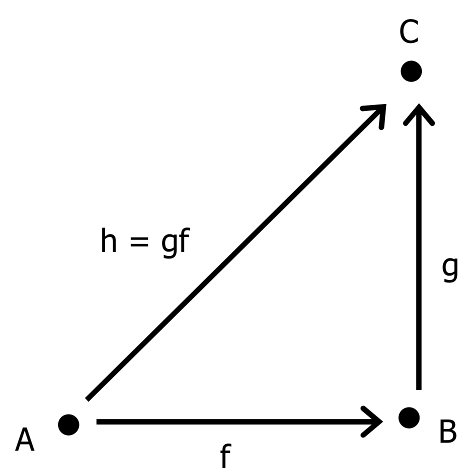
the composite kj (read as journey k after journey j) is, taking the most obvious take on journeys, the journey from La Jolla to Hyderabad. We also noted that the composite journey

kj: La Jolla → Hyderabad

of two journeys such as j and k is possible if and only if the destination of the first journey j, Amsterdam, is the same as the beginning of the second journey k, Amsterdam. Pictorially we can depict as follows:



Finally we noted that taking the journey j from La Jolla to Amsterdam and journey k from Amsterdam to Hyderabad is same as taking the composite journey m from La Jolla to Hyderabad. Now let's translate these everyday intuitions into the terminology of functions. Drawing on the above metaphor, we say that the composite of two functions f: A → B and g: C → D is defined if and only if the codomain set of the first function is same as the domain of the second function i.e. B = C, and that the domain of the composite is same as the domain of the first function and the codomain of the composite is same as the codomain of the second function. More explicitly the composite of f: A → B and g: B → C is gf: A → C.



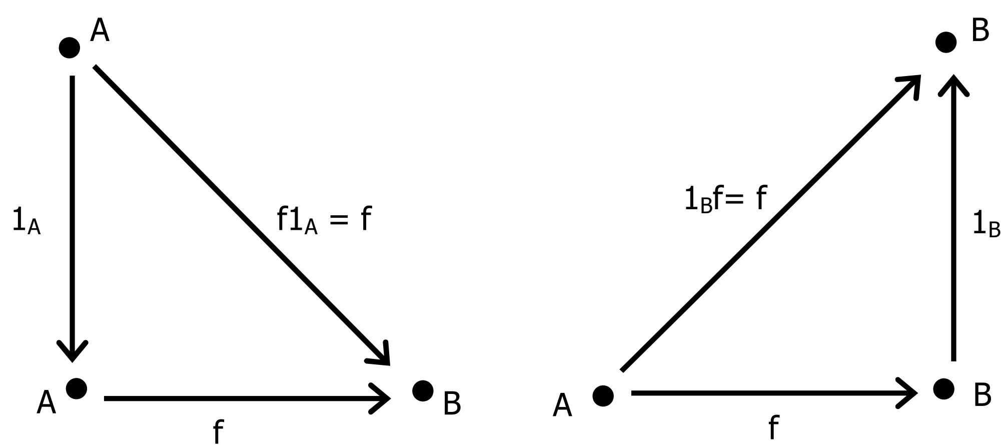
Most importantly, the composite function h is equal to the 'function g after function f'. Now we find ourselves ready to answer the question raised by our representation of function f: A → B

$$A \xrightarrow{f} B$$

as

$$A \xrightarrow{1_A} A \xrightarrow{f} B \xrightarrow{1_B} B$$

under the pretext of terminological austerity. We calculated the composites $f1_A$: A → B and 1_Bf: A → B and found that $f1_A = f$ and $1_Bf = f$ as depicted below:



Now given that $f1_A$: A → B and 1_Bf: A → B are defined, we found that the composite 1_Bf1_A: A → B can be defined. Given that the following two pair-wise composites

$$A \xrightarrow{1_A} A \xrightarrow{f} B \qquad A \xrightarrow{f} B \xrightarrow{1_B} B$$

are defined, it is easy to see that the composite of all three functions is defined by way of imagining f: A → B segments of the above two pair-wise composites overlap (which is somewhat analogous to the condition that codomain of the first function f must coincide with the domain of the second function g in order for the composite gf to be defined) so that we get

$$A \xrightarrow{1_A} A \xrightarrow{f} B \xrightarrow{1_B} B$$

We can also be more specific and state that the composite $1_B f 1_A$ can be evaluated either by first evaluating $f1_A$, which is f which when composed with 1_B gives f as the composite, which is exactly what we get when we first evaluate $1_B f$ and then compose the composite f with 1_A. Or even more explicitly the composite $1_B f 1_A$ can be calculated either as a composite of

$f1_A: A \rightarrow B$ and $1_B: B \rightarrow B$

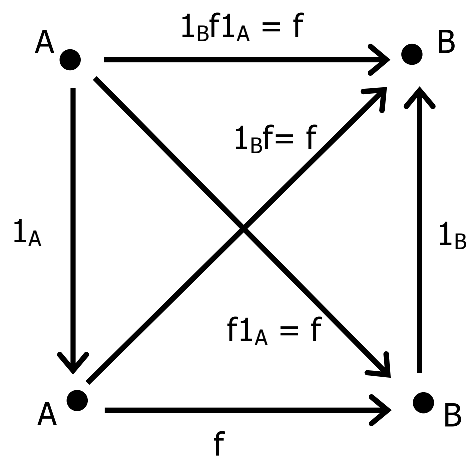
or as a composite of

$1_A: A \rightarrow A$ and $1_B f: A \rightarrow B$

and both ways of calculating $1_B f 1_A$ give the same result i.e.

$$1_B(f1_A) = (1_B f)1_A = 1_B f 1_A = f$$

as shown below:



Generalizing from identity functions to functions in general, we note that whenever two composites gf and hg are defined, then the composite hgf is defined, which can be thought of as a generalization of given 'B = C' the composite gf: A → C of functions f: A → B and g: C → D is defined, and can be calculated as the composite of gf and h i.e. h(gf) or as the composite of f and hg i.e. (hg)f is as illustrated below in terms of our favorite journeys.

j: La Jolla → Amsterdam

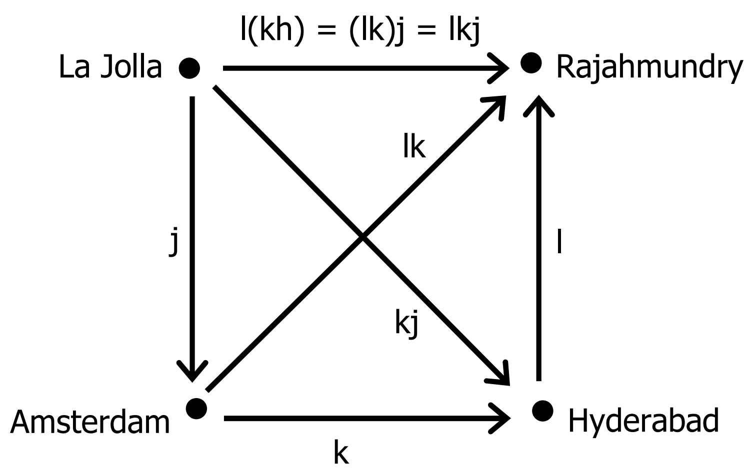
and

k: Amsterdam → Hyderabad

and

l: Hyderabad → Rajahmundry

Since the journey l's beginning is Hyderabad, which is the same as the destination of journey k, whose beginning is Amsterdam, which is the same as the destination of journey j, we can clearly form pair-wise composites (lk)j and l(kj) to obtain lkj, with, of course, (lk)j = l(kj) = lkj, when the l, k, and j are interpreted as functions.



Now let's collate our recollections of the properties of function—properties that are true of all functions—each and every function.

1. Function f: A → B has a domain A and codomain B, which are identity functions 1_A: A → A and 1_B: B → B, respectively
2. Given two functions f: A → B and g: C → D composition of f and g is defined if B = C and the composite h: A → C is given by h = gf
3. Composite of a function f: A → B with identities 1_A: A → A and 1_B: B → B satisfies: $f1_A = f = 1_Bf$

4. Given three functions f: A → B, g: C → D, and h: E → F, the triple composite hgf: A → F is defined if the pair-wise composites gf: A → D and hg: C → F are defined, or in other words hgf is defined if B = C and D = E and is given as h(gf) = (hg)f = hgf

Now let's give a name to the collection of the above list of properties; since they are dealing with functions and only functions, let's call the structure formed of this list a category of functions (in naming 'category of functions' instead of 'category of sets', I seem to think I am following Ehresmann's naming convention, which is more revealing of the category we are dealing with). If we replace function with arrow to denote anything that satisfies the above 4 conditions, we have an arbitrary category or a generic category. With little rewriting we have the general notion of CATEGORY defined:

1. Arrow f: A → B has a domain A and a codomain B, which are identity arrows 1_A: A → A and 1_B: B → B, respectively
2. Given two arrows f: A → B and g: C → D composition of f and g is defined if B = C and the composite h: A → C is given by h = gf
3. Composite of an arrow f: A → B with identities 1_A: A → A and 1_B: B → B satisfies: $f1_A = f = 1_Bf$
4. Given three arrows f: A → B, g: C → D, and h: E → F, the triple composite hgf: A → F is defined if the pair-wise composites gf: A → D and hg: C → F are defined, or in other words hgf is defined if B = C and D = E and is given as h(gf) = (hg)f = hgf

In passing we may note that, with isomorphisms, a subset of arbitrary functions, as arrows we obtain the notion of groupoid, and with automorphisms, a subset of isomorphisms, as arrows we obtain the notion of group.

It might be helpful to state what we mean by a CATEGORY in plain English. A CATEGORY, in plain English, is a mathematical universe or a domain of mathematical discourse. For example, the category of functions that we were talking about in this session is the mathematical universe inhabited by sets, functions, and composition of functions. Alternatively, the category of functions is a mathematical discourse about sets, functions, and composite of functions. In a sense the mathematical notion of CATEGORY is not inconsistent with its everyday usage.

In the spirit of complete disclosure, since I am not so sure about the legitimacy of the way we arrived at the notion of CATEGORY as a collection of properties of functions, I'll go over the textbook definition of CATEGORY, which on the surface does not seem to be much different, but may differ in matters that matter most.

Before we close let's look at a concrete illustration of the notion of CATEGORY, especially one in which arrows are not functions (Arbib & Manes). Before we get to the category, we need to have a definition in place.

A poset (or partially ordered set) is a set A with a structure of ≥, which is

Reflexive: $a \geq a$ for all a in A

Antisymmetric: $a \geq a'$ and $a' \geq a \Rightarrow a = a'$ for all a, a' in A

Transitive: $a \geq a'$ and $a' \geq a'' \Rightarrow a \geq a''$ for all a, a′, a″ in A

Consider the set A = {1, 2, 3, 4} along with the structure '≥', so that we have as arrows $2 \geq 1$, $3 \geq 2$, etc., where 1, 2, 3, and 4 are considered objects or identity arrows. The identity arrows such as $4 \geq 4$ are given by the reflexivity of the structure of ≥. The composite of two composable arrows $3 \geq 2$ and $2 \geq 1$ is $3 \geq 1$ by virtue of transitivity of ≥, and is in accord with the definition of composition of arrows of a category. We can also note that the composite of an arrow with its identities is the arrow as in the composite of $3 \geq 3$ and $3 \geq 2$ is $3 \geq 2$, and the composite of $3 \geq 2$ and $2 \geq 2$ is $3 \geq 2$. Having checked the identity laws, let's check to see if associativity holds true. The composite of three composable arrows: $4 \geq 3$, $3 \geq 2$, and $2 \geq 1$ can be evaluated by first evaluating the composite of $3 \geq 2$, and $2 \geq 1$, which is $3 \geq 1$, and then evaluating the composite of $4 \geq 3$ and $3 \geq 1$, which is $4 \geq 1$. Alternatively, we could first evaluate the composite of $4 \geq 3$ and $3 \geq 2$, which is $4 \geq 2$, and then evaluate the composite of $4 \geq 2$ and $2 \geq 1$, which is $4 \geq 1$; thereby upholding associativity. Thus we have a category (a poset) in which $a \geq b$ is an arrow (and not a function) and $a \geq a$ is the identity arrow on a in A. This example clearly shows that arrows of category need domain and codomain, which could be identities, and as long as there is composition of arrows defined satisfying identity and associative laws, we have a category.

From Function to Functor via Category

Me: We have been talking a lot about functions.

You: Yes. Given the conceptual and computational reach (that is "deeper than the seas and wider than the skies") of the notion and the notation of function it is useful to thoroughly internalize the simple mathematical construct of function so that the intuitions engendered [by working with functions] can be brought to bear on the challenging task of developing sophisticated mathematical constructs needed to refine our understanding of the universe within and without. In more concrete terms, for example, abstracting the properties of functions (e.g. every function has a domain and a codomain) leads to the notion of category, which is indispensable in organizing the subject matter of mathematics among others. There are quite a number of categories such as the category of sets and functions that we have been studying for a while and the category of dynamical systems that we briefly discussed, to name a couple.

Me: What role did (does) the notation f: A → B that we use to denote a function play in distilling and defining mathematical constructs?

You: Very important foundational role. We denote a function as f: A → B, where A and B are domain and codomain sets, respectively. Abstracting a directed arrow f with A as source and B as target from the above notation used to denote function facilitated translation and application of calculations such as composition to universes of mathematical discourse other than that of sets and functions. More pointedly, the A and B in f: A → B need not necessarily have to be sets. A and B can be the origin and the destination, respectively of a journey f. A and B can be numbers and f a relation (such as 'less than or equal to') between them. A and B can be states with f as a transformation of one state into another. In other words, we can construct well-defined operations that behave like functions even when A and B are not sets. Depending on the interpretation of source A and target B of arrow f: A → B, we can define a number of mathematical constructs as arrow f. A mathematical construct f: A → B with a particular interpretation of A and B gives rise to a definite concrete category. For example, category of matrices has matrices as arrows f: A → B, with A and B interpreted as numbers.

Me: What, if any mathematical construct, would arrow f: A → B be, if A and B in arrow f: A → B are categories?

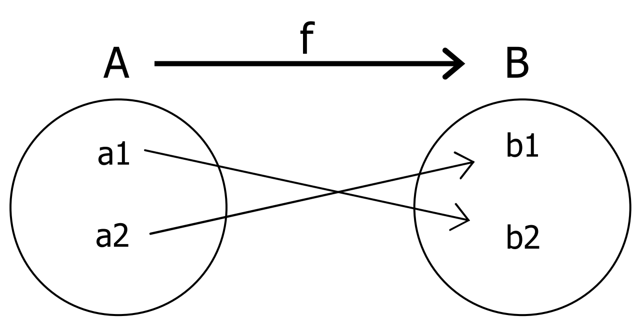
You: Analogous to the function f: A → B from domain set A to codomain set B, we have functor f: A → B from domain category A to codomain category B.

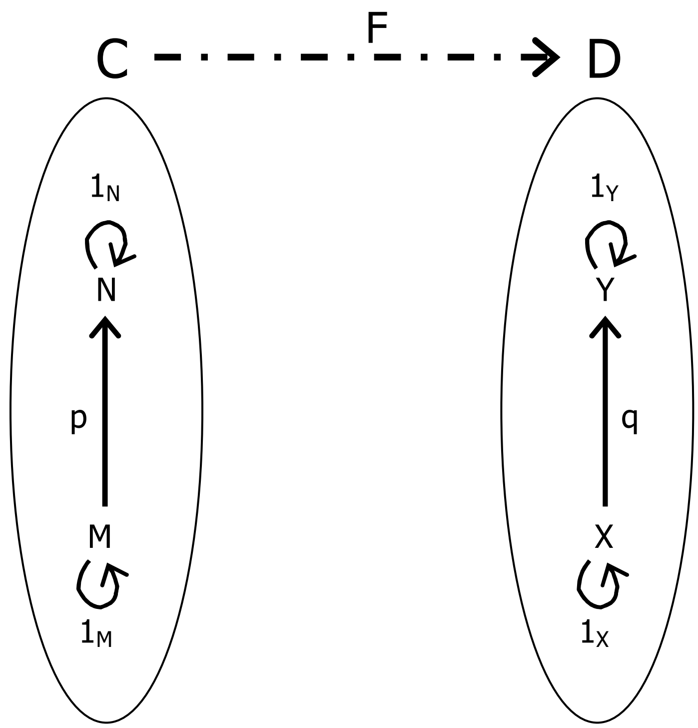
Me: So, both function and functor use the same arrow notation f: A → B.

You: Implicit in using the arrow f: A → B notation, which is used to denote function, to denote functor is the assertion that functor behaves like function in very many ways.

Me: Looking at the arrow notation f: A → B used to denote both function and functor, I can see that both function and functor have a domain object A (set/category) and a codomain object B (set/category).

You: The kinship runs deep. A function f: A → B assigns an element of codomain set B to each element of domain set A. Functor also does something similar. However, unlike sets, which have only elements, categories have objects and arrows. So functor F: C → D assigns an object of codomain category D to each object of domain category C and an arrow of codomain category D to each arrow of domain category C as shown below. (Note: From now onwards we use capital letters F to label functors and lower-case letters f to label functions. Also note that functor F: C → D is drawn as dot dash arrow to remind that functor is a pair of functions: an object function and an arrow function.) To facilitate comparison internal diagram of a function f: A → B with A = {a1, a2} and B = {b1, b2} is depicted below along with that of a functor F: C → D.





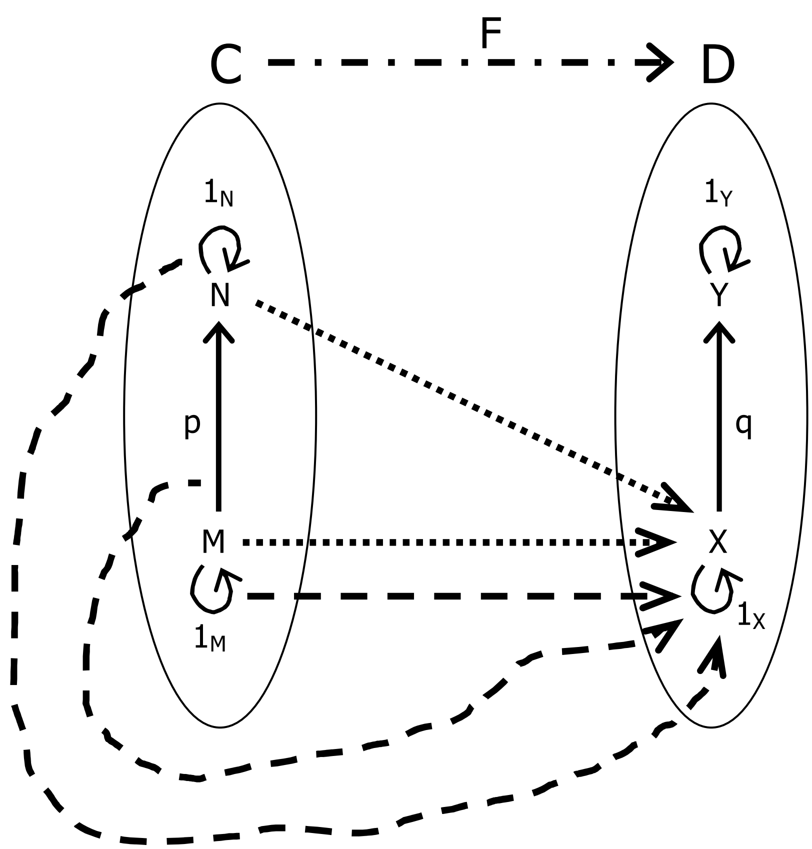
Me: Category C has 2 objects M and N, and 2 identity arrows 1_M: M → M and 1_N: N → N and 1 non-identity arrow p: M → N, a total of 3 arrows. Category D also has 2 objects X and Y, and 2 identity arrows 1_X: X → X and 1_Y: Y → Y and 1 non-identity arrow q: X → Y, a total of 3 arrows. If all we ask of a functor is assigning objects to objects and arrows to arrows, then there are 4 object functions (1. F(M) = X, F(N) = X; 2. F(M) = X, F(N) = Y; 3. F(M) = Y, F(N) = X; 4. F(M) = Y, F(N) = Y) and 27 (3^3) arrows functions giving rise to a total of 108 (4 x 27) functors from category C to category D; is this the case?

You: No. Functors between categories are required to preserve the structure that is generic to all categories. More specifically, functors preserve source/target structure [if functor F maps object A to F(A) and object B to F(B), then it should map arrow f: A → B to F(f): F(A) → F(B)], identities [if functor F maps object A to F(A), then it should map identity 1_A: A → A to $1_{F(A)}$: F(A) → F(A) i.e. $F(1_A) = 1_{F(A)}$], and composition [if functor F maps arrow f to F(f) and arrow g to F(g), then the composite gf, when defined, should be mapped to F(g)F(f) i.e. F(gf) = F(g)F(f)]. Not all 108 combinations of object and arrow functions constitute functors (i.e. satisfy the above 3 conditions) as illustrated below with couple of cases.

Case (i).

Object function 1 (dot arrow): F(M) = X, F(N) = X

Arrow function 1 (dash arrow): $F(1_M) = 1_X$, $F(p) = 1_X$, $F(1_N) = 1_X$



Now we have to check if this object function and arrow function constitute a functor i.e. satisfy the three conditions: 1. source/target, 2. identities, and 3. composition.

1. Source/target condition:

Given 1_M: M → M, the condition to be satisfied is $F(1_M)$: F(M) → F(M). From the above object function and arrow function, we find that F(M) = X and $F(1_M) = 1_X$, i.e. 1_X: X → X thereby satisfying the source/target condition. Similarly, for arrow p: M → N, F(p): F(M) → F(N) is satisfied since F(p) = 1_X, and F(M) = F(N) = X i.e. 1_X: X → X. For the third arrow 1_N: N → N, $F(1_N)$: F(N) → F(N) is satisfied since F(N) = X and $F(1_N) = 1_X$, i.e. 1_X: X → X.

2. Identity condition:

$F(1_M) = 1_{F(M)}$ [condition]

$1_X = 1_X$ [satisfied]

$F(1_N) = 1_{F(N)}$ [condition]

$1_X = 1_X$ [satisfied]

3. Composition condition:

$F(p1_M) = F(p)F(1_M)$ [condition]

$F(p) = F(p)F(1_M)$

$1_X = 1_X\, 1_X = 1_X$ [satisfied]

$F(1_N p) = F(1_N)F(p)$ [condition]

$F(p) = F(1_N)F(p)$

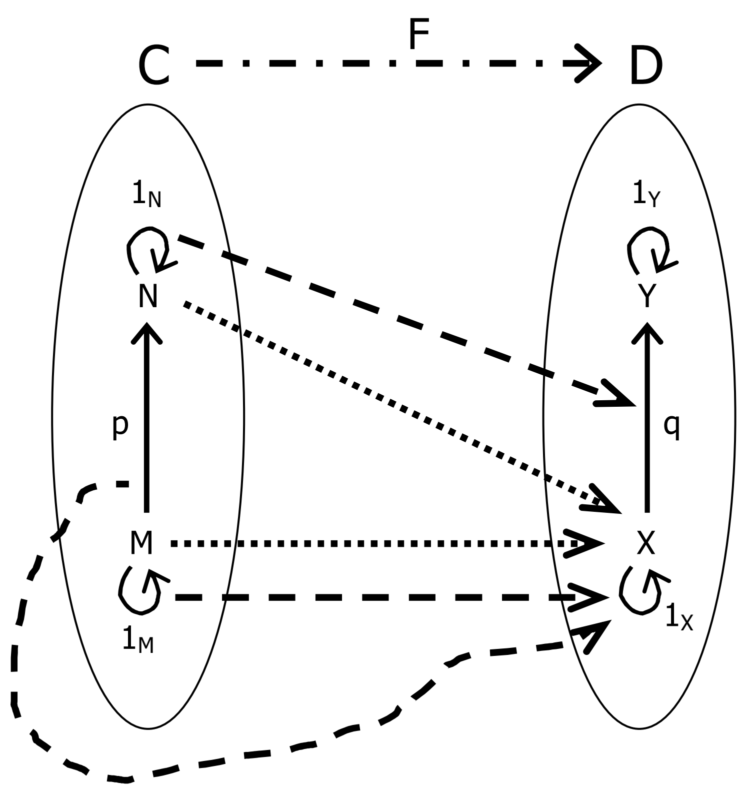
$1_X = 1_X\, 1_X = 1_X$ [satisfied]

Thus $F: C \rightarrow D$ with $F(M) = X$, $F(N) = X$, and $F(1_M) = 1_X$, $F(p) = 1_X$, $F(1_N) = 1_X$ is a functor from category C to category D.

Case (ii).

Object function 1 (dot arrow): $F(M) = X$, $F(N) = X$

Arrow function 2 (dash arrow): $F(1_M) = 1_X$, $F(p) = 1_X$, $F(1_N) = q$



Now we have to check if this object function and arrow function constitute a functor.

1. Source/target condition:

1_M: M → M [given]

$F(1_M)$: F(M) → F(M) [condition]

1_X: X → X [satisfied]

p: M → N [given]

F(p): F(M) → F(N) [condition]

1_X: X → X [satisfied]

1_N: N → N [given]

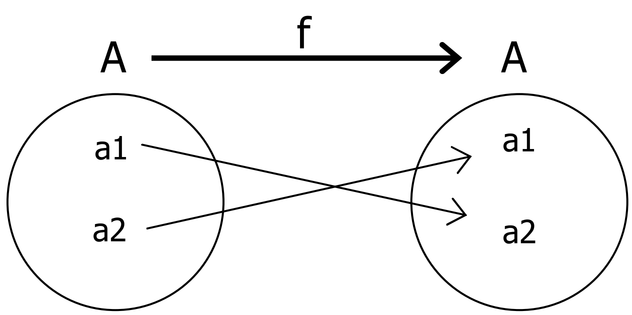
$F(1_N)$: F(N) → F(N) [condition]

q: X → X [not satisfied since q: X → Y]

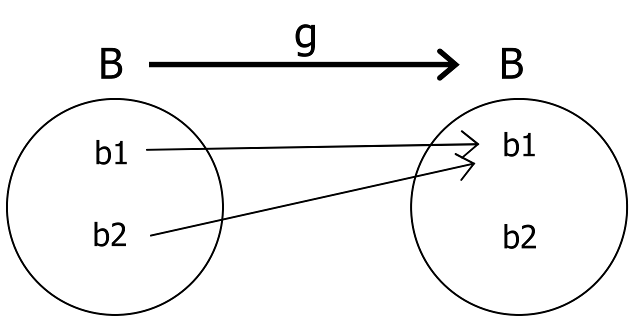
Thus F: C → D with F(M) = X, F(N) = X, and $F(1_M) = 1_X$, $F(p) = 1_X$, $F(1_N) = q$ is not a functor from category C to category D.

Me: Maps in a category are required to preserve the structure of objects of the category and are not required to reflect structure. Drawing parallels to functors, do functors between categories preserve (and not necessarily reflect) any positive property (such as isomorphism) that an arrow (in the category) may have?

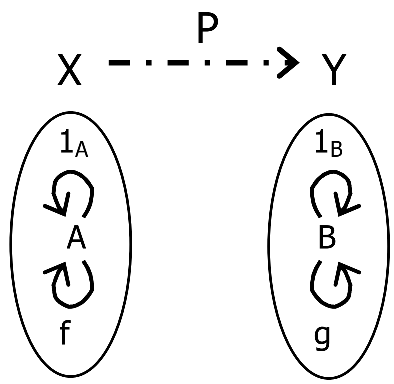
You: Yes. Functors preserve (and may not reflect) positive properties as shown in the following examples. First, let's consider two sets A = {a1, a2} and B = {b1, b2} and two endomaps f: A → A [f(a1) = a2 and f(a2) = a1] and g: B → B [g(b1) = b1 and g(b2) = b1]. With these we can form two categories: one category X with object A, and arrow f and the identity arrow 1_A: A → A, and the second category Y with object B, and arrow g and identity arrow 1_B: B → B. The internal diagrams of functions f and g and functor P: X → Y are shown below.



Note that f is an isomorphism and in particular an involution i.e. ff = 1_A.



Note that g is an idempotent i.e. gg = g.

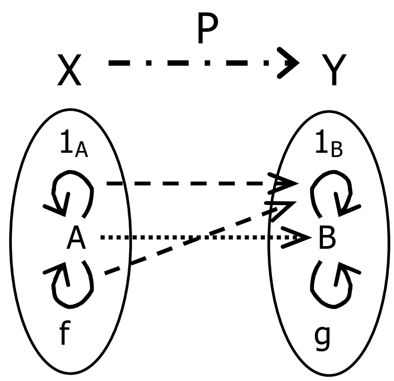


There is one object A in the domain category X and one object B in the codomain category Y. So, there is only 1 object function which takes the only object A of category X to the only object B of category Y i.e. P(A) = B. There are two arrows 1_A and f in category X and two arrows 1_B and g in category Y. So there are a total of 4 (2^2) arrow functions: 1. $P(1_A) = 1_B$ and $P(f) = 1_B$, 2. $P(1_A) = 1_B$ and P(f) = g, 3. $P(1_A) = g$ and $P(f) = 1_B$, and 4. $P(1_A) = g$ and P(f) = g. Since we have 1 object function and 4 arrow functions, we have a total of 4 (1 x 4) candidate functors. Let's see how many of these satisfy the definition of functor.

Case (i).

Object function 1: $P(A) = B$

Arrow function 1: $P(1_A) = 1_B$ and $P(f) = 1_B$



1. Source/target condition:

$1_A: A \rightarrow A$ [given]

$P(1_A): P(A) \rightarrow P(A)$ [condition]

$1_B: B \rightarrow B$ [satisfied]

$f: A \rightarrow A$ [given]

$P(f): P(A) \rightarrow P(A)$ [condition]

$1_B: B \rightarrow B$ [satisfied]

2. Identity condition:

$P(1_A) = 1_{P(A)}$ [condition]

$1_B = 1_B$ [satisfied]

3. Composition condition:

$P(f1_A) = P(f)P(1_A)$ [condition]

$P(f) = P(f)P(1_A)$

$1_B = 1_B\ 1_B = 1_B$ [satisfied]

$P(1_Af) = P(1_A)P(f)$ [condition]

$P(f) = P(1_A)P(f)$

$1_B = 1_B\ 1_B = 1_B$ [satisfied]

$P(ff) = P(f)P(f)$ [condition]

$P(1_A) = P(f)P(f)$

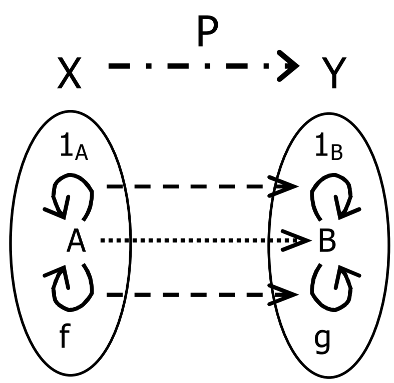
$1_B = 1_B\ 1_B = 1_B$ [satisfied]

Thus $P: X \rightarrow Y$ with object function $P(A) = B$, and arrow function $P(1_A) = 1_B$ and $P(f) = 1_B$ is a functor from category X to category Y. Note that the functor P mapped identity 1_A to identity 1_B and involution f to involution (identity 1_B is an involution since $1_B\ 1_B = 1_B$) i.e. preserved identity and involution. However non-identity f got mapped to identity 1_B i.e. did not reflect identity.

Case (ii).

Object function 1: P(A) = B

Arrow function 2: $P(1_A) = 1_B$ and P(f) = g



1. Source/target condition:

1_A: A → A [given]

$P(1_A)$: P(A) → P(A) [condition]

1_B: B → B [satisfied]

f: A → A [given]

P(f): P(A) → P(A) [condition]

g: B → B [satisfied]

2. Identity condition:

$P(1_A) = 1_{P(A)}$ [condition]

$1_B = 1_B$ [satisfied]

3. Composition condition:

$P(f1_A) = P(f)P(1_A)$ [condition]

$P(f) = P(f)P(1_A)$

$g = g\ 1_B = g$ [satisfied]

$P(1_A f) = P(1_A)P(f)$ [condition]

$P(f) = P(1_A)P(f)$

$g = 1_B\ g = g$ [satisfied]

$P(ff) = P(f)P(f)$ [condition]

$P(1_A) = P(f)P(f)$

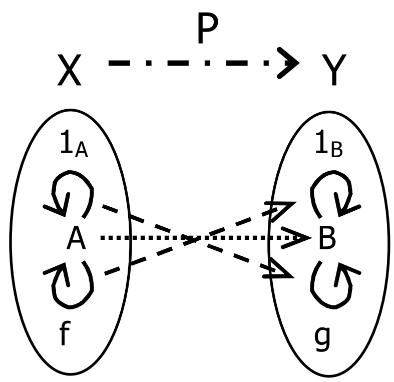
$1_B = gg = g$ [not satisfied since $g \neq 1_B$]

Thus $P: X \rightarrow Y$ with object function $P(A) = B$, and arrow function $P(1_A) = 1_B$ and $P(f) = g$ is not a functor. Note that P mapped involution (isomorphism) f to idempotent g (non-involution, non-isomorphism) i.e. did not preserve involution and as a result of which failed to satisfy a composition condition.

Case (iii).

Object function 1: P(A) = B

Arrow function 3: $P(1_A) = g$ and $P(f) = 1_B$



1. Source/target condition:

1_A: A → A [given]

$P(1_A)$: P(A) → P(A) [condition]

g: B → B [satisfied]

f: A → A [given]

P(f): P(A) → P(A) [condition]

1_B: B → B [satisfied]

2. Identity condition:

$P(1_A) = 1_{P(A)}$ [condition]

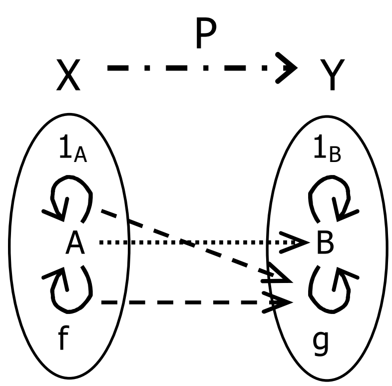
$g = 1_B$ [not satisfied since $g \neq 1_B$]

Thus P: X → Y with object function P(A) = B, and arrow function $P(1_A) = g$ and $P(f) = 1_B$ is not a functor. Note that P mapped identity 1_A to non-identity g i.e. did not preserve identity and as a result of which failed to satisfy the identity condition.

Case (iv).

Object function 1: P(A) = B

Arrow function 4: $P(1_A) = g$ and P(f) = g



1. Source/target condition:

1_A: A → A [given]

$P(1_A)$: P(A) → P(A) [condition]

g: B → B [satisfied]

f: A → A [given]

P(f): P(A) → P(A) [condition]

g: B → B [satisfied]

2. Identity condition:

$P(1_A) = 1_{P(A)}$ [condition]

$g = 1_B$ [not satisfied since $g \neq 1_B$]

Thus P: X → Y with object function P(A) = B, and arrow function $P(1_A) = g$ and P(f) = g is not a functor. Note that P mapped identity 1_A to non-identity g i.e. did not preserve identity and as a result of which failed to satisfy the identity condition.

Me: In the previous particular examples functors preserved positive properties such as isomorphism. Can we show that that is the case in general i.e. can we prove that functors preserve, say, isomorphisms?

You: Yes, we can. Consider a functor P: X → Y. Let category X have 2 objects A and B, and 4 maps f: A → B, g: B → A, 1_A: A → A, and 1_B: B → B. Let f / g be an isomorphism i.e. $gf = 1_A$ and $fg = 1_B$. Furthermore let the functor P map objects A and B in X to objects P(A) and P(B) in Y. Then by the definition of functor we have P(f): P(A) → P(B) and P(g): P(B) → P(A); $P(1_A) = 1_{P(A)}$ and $P(1_B) = 1_{P(B)}$; P(gf) = P(g)P(f) and P(fg) = P(f)P(g). Functor P preserves isomorphism f / g if $P(g)P(f) = 1_{P(A)}$ and $P(f)P(g) = 1_{P(B)}$. Going by the definition of functor, we have $P(g)P(f) = P(gf) = P(1_A) = 1_{P(A)}$ and

$P(f)P(g) = P(fg) = P(1_B) = 1_{P(B)}$.

Thus by definition functor preserves isomorphisms.

Me: In the case of functions, given two functions f: A → B and g: B → C, with the codomain of f same as the domain of g, we can compose them to get a composite function gf: A → C. Is it the case with functors also? In other words, given two functors U: R → S and V: S → T, with the codomain category of functor U same as the domain category of functor V, can we compose the two functors to get a composite functor VU: R → T?

You: Yes. The composite of two composable functors is a functor.

Me: In the case of functions, we have various types of functions such as identity functions, 1-1 functions, and onto functions, to name a few. Do we have any analogous functors?

You: Yes. We have an identity functor for each category taking every object and every arrow to itself. We also have faithful and full functors, which are analogous to the 1-1 and onto functions, respectively.

Me: Sets and functions along with composition form a category with sets as objects and functions as arrows. Given that composition of functors is defined and given identity functors, does it make sense to ask whether categories and functors form a category with categories as objects and functors as arrows?

You: Yes. Categories and functors form a category.

Forgetful functor

In seeing the states (when looking at a dynamical system visualized as a collection of dots with an arrow leaving each dot and pointing to a dot) of a dynamical system, the dots depicting the states of the dynamical system are brought to figural salience while the arrows denoting state-transitions are relegated to a distant background. It is then not surprising to find that the set of thus attended naked-dots has as many elements as the number of states of the dynamical system we are looking at.

In seeing, when looking at a set of elements, a natural number sequence corresponding to each one of the elements in the set, we are imagining a dynamical system with elements of the set as initial states. It is then not surprising to find that the thus imagined dynamical system has as many sequences of state-transitions as the number of elements of the set we are looking at.

Stating the obvious or stating in a way that the statement reads as though that which is stated is all too obvious is one of my long-term objectives in the study of Conceptual Mathematics. Obviously I'm nowhere near—especially in making the definition of Adjoint Functor seem like stating something obvious in a cursory glance at dynamical systems.

Given a dynamical system, we can think of a process which extracts the states of the given dynamical (while forgetting state-transitions). Given a set of elements we can think of a process of interpreting the elements of the given set as initial states of an imagined dynamical system. The process of free creation of dynamical systems from sets (with elements thought of as initial states of imagined dynamical systems) is, in a sense, undoing the process of distilling sets (of states; while blissfully forgetting state-transitions) from dynamical systems.

There’s more: dynamical systems are objects of one category (of dynamical systems) and sets are objects of another category (of sets). So, the process of extracting sets (of states; of forgetting state-transitions) from dynamical systems is a functor from the category of dynamical systems to the category of sets; while the process of freely imagining a dynamical system when looking at a set (with elements of the set thought of as initial states of the imaginary dynamical system) is a functor from the category of sets to the category of dynamical systems. The domain category of the forgetful functor is the codomain category of the free functor and the codomain category of the forgetful functor is the domain category of the free functor; hence the pair of functors is an opposite-pair.

There’s more that can be said about the ‘opposition.’ Functors between categories not only map objects to objects (dynamical systems to sets; sets to dynamical systems), but also assign a map in the codomain category to each map in the domain category. Well, then, which one?

One approach to finding something we don’t know is, instead of engaging in orgies of wishful thinking, consciously copying something we know. What we know: one (forgetful) functor assigns to each dynamical system in the category of dynamical systems a set (with as many elements as the number of states of the dynamical system) in the category of sets, while the other (free) functor assigns to each set (in the category of sets) a dynamical system (with as many natural number sequences as the number of elements in the set). Given a set, we get a dynamical system (which is the value of the [free] functor at the given set) and given a dynamical system, we get a set (which is the value of the [forgetful] functor at the given dynamical system). We now have two sets (the given set and the set of states of the given dynamical system) and two dynamical systems (the given dynamical system and the free dynamical system with elements of the given set as initial states). It is, by now, abundantly clear to me that that which I'm talking about will be clearer if I introduce few symbols ;)

Let ***A*** denote the category of sets and ***B*** denote the category of dynamical systems.

Let

$$P: \boldsymbol{A} \rightarrow \boldsymbol{B}$$

denote the [free] functor assigning to each set A (in the category of sets ***A***) its free dynamical system P (A) (in the category of dynamical systems ***B***; with the elements of the set A as initial states). Let's say

$$A = \{a, a'\}$$

then

$$P\ (A) = [a(0) \rightarrow a(1) \rightarrow a(2)\ldots$$

$$a'(0) \rightarrow a'\ (1) \rightarrow a'\ (2)\ldots]$$

with a(0) = a, a'(0) = a' while a(i) and a'(i) for i = 1, 2… take values in A. In other words, the dynamical system has two initial states a, a' and with each state-transition the system states transition into these states. For example, a(i) = a and a'(i) = a for all i. However we need not worry about specific states into which each one of the states transition; all we care about, for now, is that we have a dynamical system P (A) with elements of the set A as initial states.

Let

$$Q: \boldsymbol{A} \leftarrow \boldsymbol{B}$$

denote the [state-space] functor assigning to each dynamical system B (in the category of dynamical systems ***B***) its set of states (while forgetting the state-transitions) Q (B) in the category of sets ***A***.

Let’s consider a set A with m elements and a dynamical system B with n states. Since Q (B) is the set of states of the dynamical system B, the set Q (B) has n elements. So the number of functions (in the category of sets ***A***)

$$A \rightarrow Q\ (B)$$

from the set A to the set of states Q (B) of the dynamical system B is equal to n^m (since |A| = m and |Q (B)| = n). What worth noting is that the number of maps (in the category of dynamical systems ***B***)

$$P(A) \rightarrow B$$

is also equal to n^m since the free dynamical system P (A) has m [natural number] sequences: one for each one of the m elements in the set A. Note that the natural number object

$$N = 0 \rightarrow 1 \rightarrow 2 \rightarrow \ldots$$

in the category of dynamical systems ***B*** plays the role that a singleton set **1** plays in the category of sets ***A***. More specifically, the number of maps (in the category of dynamical systems ***B***) from N to a dynamical system B is equal to the number of states of the dynamical system. And the number of maps from a dynamical system P (A) with m [natural number] sequences to a dynamical system B with n states is equal to n^m.

So for the given opposite-pair of functors

$$P: \boldsymbol{A} \rightarrow \boldsymbol{B}$$

$$Q: \boldsymbol{A} \leftarrow \boldsymbol{B}$$

we find that the number of maps

$$P(A) \rightarrow B$$

in the category of dynamical systems ***B*** is equal to the number of maps

$$A \to Q(B)$$

in the category of sets ***A***. And this is how close I got to the 1-1 correspondence

$$P(A) \to B$$

$$A \to Q(B)$$

between maps in the category of dynamical systems and in the category of sets. This 1-1 correspondence is what we need to show that the [forgetful] functor

$$Q: \boldsymbol{A} \leftarrow \boldsymbol{B}$$

assigning to each dynamical system B (in ***B***) its set Q (B) of states in ***A*** (while forgetting state-transitions) is right adjoint to the [free] functor

$$P: \boldsymbol{A} \to \boldsymbol{B}$$

assigning to each set A (in ***A***) its free dynamical system P (A) in ***B*** (with elements of the set as initial states).

I'll get back to adjoint functors (Conceptual Mathematics, pp. 372 – 7) in a min. but first what's the difference between:

two sets are in 1-1 correspondence

vs.

two sets have same number of elements

It is one thing to know that the number of chairs is equal to the number of students in a classroom, and is something different [i.e. less information] from seeing a classroom with all students seated with each student occupying a chair and with no chair empty. In the latter case, we know not only that the number of chairs is equal to the number of students but also know which chair is occupied by which student (Conceptual Mathematics, pp. 96 – 7).

Returning to our beloved adjoints, there's another related right adjoint (Exercise 9.25 on page 166 of Sets for Mathematics). The [fixed-point] functor

$$Y: \boldsymbol{A} \leftarrow \boldsymbol{B}$$

assigning to each dynamical system B its set of equilibrium states Y (B) is right adjoint to the [static] functor

$$X: \boldsymbol{A} \rightarrow \boldsymbol{B}$$

assigning to each set A its static dynamical system X (A) (i.e. the identity function 1_A: A → A).

Given a set A of m elements, the dynamical system X (A) to which A is mapped to by the [static] functor

$$X: \boldsymbol{A} \rightarrow \boldsymbol{B}$$

has m equilibrium states.

Given a dynamical system B with n equilibrium states, the set Y (B) to which B is mapped to by the [fixed-point] functor

$$\text{Y}: \boldsymbol{A} \leftarrow \boldsymbol{B}$$

has n elements.

Once again the number of functions

$$\text{A} \rightarrow \text{Y (B)}$$

is equal to the number of maps

$$\text{X (A)} \rightarrow \text{B}$$

but, once again what we need is little more i.e. the 1-1 correspondence

$$\frac{\text{X (A)} \rightarrow \text{B}}{\text{A} \rightarrow \text{Y (B)}}$$

Exercise 5.17c (Sets for Mathematics, p. 109)

Show that the assignments $(-)^T$ (A) = A^T for any set A and $(-)^T$ (*f*) = f^T for any function *f* define a functor $(-)^T$: ***S*** --> ***S***.

First, ***S*** is the category of sets, whose objects and morphisms are sets and functions, respectively. Next, a functor

P: ***V*** --> ***W***

is an assignment of objects and morphisms of the codomain category ***W*** to objects and morphisms, respectively, of the domain category ***V*** in a way respectful of the domain, codomain, identity, and composition structure of the category.

Let's look at the case of the functor of our present concern, which has the category ***S*** of sets as both domain and codomain category i.e. a functor

P: ***S*** --> ***S***

The functor P assigns sets to sets, which we denote as

P_{Ob}: $\boldsymbol{S}_{Ob}$ --> $\boldsymbol{S}_{Ob}$

and functions to functions, which we denote as

P_{Mp}: $\boldsymbol{S}_{Mp}$ --> $\boldsymbol{S}_{Mp}$

These two functions

P_{Ob}: $\boldsymbol{S}_{Ob}$ --> $\boldsymbol{S}_{Ob}$

P_{Mp}: $\boldsymbol{S}_{Mp}$ --> $\boldsymbol{S}_{Mp}$

together are required to satisfy, in order to constitute a functor

$$P: \boldsymbol{S} \dashrightarrow \boldsymbol{S}$$

the following four conditions corresponding to preserving (i) domain, (ii) codomain, (iii) identity, and (iv) composition.

(i) Preserving Domain

$$\begin{array}{ccc} \boldsymbol{S}_{Ob} & \text{--}P_{Ob}\text{-->} & \boldsymbol{S}_{Ob} \\ \wedge & & \wedge \\ | & & | \\ \textit{domain} & & \textit{domain} \\ | & & | \\ \boldsymbol{S}_{Mp} & \text{--}P_{Mp}\text{-->} & \boldsymbol{S}_{Mp} \end{array}$$

The commutativity of the above diagram stated as

$$\textit{domain} \text{ o } P_{Mp} = P_{Ob} \text{ o } \textit{domain}$$

guarantees the preserving of domain (where ‘o’ denotes composition). This commutativity equation is read as: the domain set (in top-right $\boldsymbol{S}_{Ob}$) of the function (in bottom-right $\boldsymbol{S}_{Mp}$) to which a function (in bottom-left $\boldsymbol{S}_{Mp}$) is assigned to by P_{Mp} is same as the set (in top-right $\boldsymbol{S}_{Ob}$) to which the domain set (in top-left $\boldsymbol{S}_{Ob}$) of the function (in bottom-left $\boldsymbol{S}_{Mp}$) is assigned to by P_{Ob}.

(ii) Preserving Codomain

$\boldsymbol{S}_{Ob}$ --P_{Ob}--> $\boldsymbol{S}_{Ob}$

^ ^

| |

codomain *codomain*

| |

$\boldsymbol{S}_{Mp}$ --P_{Mp}--> $\boldsymbol{S}_{Mp}$

The commutativity of the above diagram stated as

$$codomain \circ P_{Mp} = P_{Ob} \circ codomain$$

guarantees the preserving of codomain. This commutativity equation is read as: the codomain set (in top-right $\boldsymbol{S}_{Ob}$) of the function (in bottom-right $\boldsymbol{S}_{Mp}$) to which a function (in bottom-left $\boldsymbol{S}_{Mp}$) is assigned to by P_{Mp} is same as the set (in top-right $\boldsymbol{S}_{Ob}$) to which the codomain set (in top-left $\boldsymbol{S}_{Ob}$), of the function (in bottom-left $\boldsymbol{S}_{Mp}$), is assigned to by P_{Ob}.

(iii) Preserving Identity

$\boldsymbol{S}_{Mp}$ --P_{Mp}--> $\boldsymbol{S}_{Mp}$

^ ^

| |

identity *identity*

| |

$$\boldsymbol{S}_{Ob} \quad \text{--}P_{Ob}\text{-->} \quad \boldsymbol{S}_{Ob}$$

The commutativity of the above diagram stated as

$$\textit{identity} \text{ o } P_{Ob} = P_{Mp} \text{ o } \textit{identity}$$

guarantees the preserving of identity. This commutativity equation is read as: the identity function (in top-right $\boldsymbol{S}_{Mp}$) of the set (in bottom-right $\boldsymbol{S}_{Ob}$) to which a set (in bottom-left $\boldsymbol{S}_{Ob}$) is assigned to by P_{Ob} is same as the function (in top-right $\boldsymbol{S}_{Mp}$) to which the identity function (in top-left $\boldsymbol{S}_{Mp}$), of the set (in bottom-left $\boldsymbol{S}_{Ob}$), is assigned to by P_{Mp}.

(iv) Preserving Composition

$$P_{Mp}\ (g \text{ o } f) = P_{Mp}\ (g) \text{ o } P_{Mp}\ (f)$$

The function (P_{Mp} (g o f)) to which the composite function (g o f) is assigned to by P_{Mp} is same as the composite of the functions (P_{Mp} (g), P_{Mp} (f)) to which the functions (g, f, respectively) are assigned to by P_{Mp}.

Now that we know what it takes to be a functor, let's see if what we are given i.e. the functions

$$(\text{-})^T(A) = A^T$$

$$(\text{-})^T(f) = f^T$$

together constitute a functor

$$(\text{-})^T\text{: } \boldsymbol{S} \text{ --> } \boldsymbol{S}$$

The function

$$(-)^{T}_{Ob}: \boldsymbol{S}_{Ob} \dashrightarrow \boldsymbol{S}_{Ob}$$

assigns to each set

$$A$$

(in the domain set $\boldsymbol{S}_{Ob}$) the set

$$A^T$$

of T-shaped figures in A i.e. the functions

$$a: T \text{-->} A$$

from [a fixed set] T to A.

The function

$$(-)^{T}_{Mp}: \boldsymbol{S}_{Mp} \text{-->} \boldsymbol{S}_{Mp}$$

assigns to each function

$$f: A \text{-->} B$$

(in the domain set $\boldsymbol{S}_{Mp}$) the induced function

$$f^T: A^T \text{-->} B^T$$

where A^T and B^T are map sets whose elements are T-shaped figures in A and B, respectively.

The function

$$f^T: A^T \text{-->} B^T$$

assigns to each element (a T-shaped figure in A)

$$a: T \dashrightarrow A$$

in the domain map set

$$A^T$$

the element

$$T \text{ --}a\text{--> } A \text{ --}f\text{--> } B$$

i.e. a T-shaped figure in B

$$fa: T \text{ --> } B$$

in the codomain set

$$B^T$$

Thus

$$f^{T}(a) = fa$$

for all

$$a: T \text{ --> } A$$

in the domain map set

$$A^T$$

of the function

$$f^T: A^T \dashrightarrow B^T$$

Now we have to check to see if the object function

$$(-)^T{}_{Ob}(A) = A^T$$

and the morphism function

$$(-)^T{}_{Mp}(f: A \dashrightarrow B) = f^T: A^T \dashrightarrow B^T$$

together constitute a functor

$$(-)^T: \boldsymbol{S} \dashrightarrow \boldsymbol{S}$$

i.e. preserve (i) domain, (ii) codomain, (iii) identity, and (iv) composition.

(i) Preserving Domain

$$\begin{array}{ccc} \boldsymbol{S}_{Ob} & \text{--}(-)^T{}_{Ob}\text{-->} & \boldsymbol{S}_{Ob} \\ \wedge & & \wedge \\ | & & | \\ \textit{domain} & & \textit{domain} \\ | & & | \\ \boldsymbol{S}_{Mp} & \text{--}(-)^T{}_{Mp}\text{-->} & \boldsymbol{S}_{Mp} \end{array}$$

$$\textit{domain} \circ (-)^T{}_{Mp} = (-)^T{}_{Ob} \circ \textit{domain}$$

LHS

$domain \circ (-)^{T}_{Mp}$ (f: A --> B) = $domain$ (f^{T}: A^{T} --> B^{T}) = A^{T}

RHS

$(-)^{T}_{Ob} \circ domain$ (f: A --> B) = $(-)^{T}_{Ob}$ (A) = A^{T}

(ii) Preserving Codomain

$$\mathbf{S}_{Ob} \quad \text{--}(-)^{T}_{Ob}\text{-->} \quad \mathbf{S}_{Ob}$$

^ ^

| |

codomain *codomain*

| |

$$\mathbf{S}_{Mp} \quad \text{--}(-)^{T}_{Mp}\text{-->} \quad \mathbf{S}_{Mp}$$

$$codomain \circ (-)^{T}_{Mp} = (-)^{T}_{Ob} \circ codomain$$

LHS

$codomain \circ (-)^{T}_{Mp}$ (f: A --> B) = $codomain$ (f^{T}: A^{T} --> B^{T}) = B^{T}

RHS

$(-)^{T}_{Ob} \circ codomain$ (f: A --> B) = $(-)^{T}_{Ob}$ (B) = B^{T}

(iii) Preserving Identity

$$\mathbf{S}_{Mp} \quad \text{--}(-)^{T}_{Mp}\text{-->} \quad \mathbf{S}_{Mp}$$

^ ^

| |

identity *identity*

| |

$\boldsymbol{S}_{\text{Ob}}$ --$(-)^{T}_{\text{Ob}}$--> $\boldsymbol{S}_{\text{Ob}}$

$$\textit{identity} \circ (-)^{T}_{\text{Ob}} = (-)^{T}_{\text{Mp}} \circ \textit{identity}$$

LHS

$\textit{identity} \circ (-)^{T}_{\text{Ob}} (A) = \textit{identity} (A^{T}) = 1_{AT}: A^{T} \dashrightarrow A^{T}$

RHS

$(-)^{T}_{\text{Mp}} \circ \textit{identity} (A) = (-)^{T}_{\text{Mp}} (1_{A}: A \dashrightarrow A) = 1_{A}{}^{T}: A^{T} \dashrightarrow A^{T}$

where

$1_{A}{}^{T} (a: T \dashrightarrow A) = T \text{ --}a\text{--> } A \text{ --}1_{A}\text{--> } A = 1_{A} \circ a = a: T \dashrightarrow A$

(iv) Preserving Composition

$$(-)^{T}_{\text{Mp}} (g \circ f) = (-)^{T}_{\text{Mp}} (g) \circ (-)^{T}_{\text{Mp}} (f)$$

where

$$f: A \dashrightarrow B,\ g: B \dashrightarrow C, \text{ and } g \circ f: A \dashrightarrow C$$

LHS

$(-)^{T}_{Mp}$ (g o f: A --> C) = $(gf)^{T}$: A^{T} --> C^{T}

where

$(gf)^{T}$(a: T --> A) = T --a--> A --gf--> C = (gf)a: T --> C

RHS

$(-)^{T}_{Mp}$ (g) o $(-)^{T}_{Mp}$ (f)

= g^{T}: B^{T} --> C^{T} o f^{T}: A^{T} --> B^{T}

= A^{T} --f^{T}--> B^{T} --g^{T}--> C^{T}

= g^{T} o f^{T}: A^{T} --> C^{T}

where

g^{T} o f^{T} (a: T --> A) = g^{T} (fa: T --> B) = g(fa): T --> C

Thanks to the associative law

$$(gf)a = g(fa)$$

of composition

$$T \text{ --}a\text{--> } A \text{ --}f\text{--> } B \text{ --}g\text{--> } C$$

the morphism component

$$(-)^{T}_{Mp}: \boldsymbol{S}_{Mp} \text{ --> } \boldsymbol{S}_{Mp}$$

of the functor

$$(-)^T: \boldsymbol{S} \dashrightarrow \boldsymbol{S}$$

preserves composition. Thus with (i) domain, (ii) codomain, (iii) identity, (iv) composition preserved by the assignments

$$(-)^T(A) = A^T$$

$$(-)^T(f) = f^T$$

we do have a functor

$$(-)^T: \boldsymbol{S} \dashrightarrow \boldsymbol{S}$$

which assigns to each set

$$A$$

the map set

$$A^T$$

of T-shaped figures in A and to each function

$$f: A \dashrightarrow B$$

the induced function

$$f^T: A^T \dashrightarrow B^T$$

with

$$f^T(a: T \dashrightarrow A) = fa: T \dashrightarrow B$$

for all elements

$$a\colon T \dashrightarrow A$$

in the domain set

$$A^T$$

Exercise 29 (Conceptual Mathematics, p.150): Every morphism *f*: X –> Y in a category ***C*** gives rise to a morphism in the category of *Z*-structures, by the associative law.

Let's start with what we are given: associative law. This seemingly simple law from elementary school

$$(1 + 2) + 3 = 1 + (2 + 3)$$

is all that we need to do the exercise (all exercises, for that matter; see Conceptual Mathematics, p.136 and p.371).

The composite of three composable morphisms such as

A – *p* –> B – *x* –> X – *f* –> Y

can be calculated in two ways:

1. first calculate the composite of morphisms *p* and *x*, and then the composite of *xp* and *f*

2. first calculate the composite of morphisms *x* and *f*, and then the composite of *p* and *fx*

The associative law states that these two calculations give the same result, which is expressed as

$$f\,(xp) = (fx)\,p$$

All we have to do, in order to do the exercise, is turn the associate law

$$f\,(xp) = (fx)\,p$$

into the commutative diagram

```
xp    –>    fxp
^           ^

|           |

x     –>    fx
```

Let’s start at the bottom-left corner

$$x\text{: B –> X}$$

which when pre-composed with

$$p\text{: A –> B}$$

takes us to the top-left corner

$$xp\text{: A –> X}$$

which when post-composed with

$$f\text{: X –> Y}$$

takes us to the top-right corner

$$fxp: A \to Y$$

Of course, we could have taken the other route i.e. start (as earlier) at the bottom-left corner

$$x: B \to X$$

which when post-composed with

$$f: X \to Y$$

takes us to the bottom-right corner

$$fx: B \to Y$$

which when pre-composed with

$$p: A \to B$$

takes us to the top-right corner

$$fxp: A \to Y$$

So, to sum up, reading the commutative diagram *xp* –*post*–> *fxp*

^ ^

| |

pre *pre*

| |

x –*post*–> *fx*

satisfying

post o *pre* = *pre* o *post*

(where ‘o’ denotes composition) into the associative law

f (*xp*) = (*fx*) *p*

satisfied by

A – *p* –> B – *x* –> X – *f* –> Y

is all that’s needed to do the exercise.

What we need to deliver is a category of *Z*-structures, and that's starters. But, what on earth is *Z*-structure? First, let's make explicit something that's implicit. *Z* is a **small family** of objects and morphisms of the category ***C***. The only condition on the small family *Z* is that if a morphism of ***C*** is in the small family, then both domain and codomain of the morphism are in the family.

Let's take

A – *p* –> B

as our small family *Z* i.e. our family *Z* consists of two objects (A, B) and one morphism (*p*). Every object X of the category ***C*** gives rise to a *Z*-structure, which has as many component sets as the number of objects in *Z* and as many structural functions (between these component sets) as the number of morphisms in *Z*. Since there are two objects (A, B) in our family *Z*, we have two component sets:

1. A-th component set A(X) is the set of all A-shaped figures in X (i.e. A –> X)
2. B-th component set B(X) is the set of all B-shaped figures in X (i.e. B –> X)

corresponding to the two objects (A, B). Since there is one morphism

p: A –> B

in *Z*, we have one structural function

p(X): A(X) <– B(X)

(note the opposite direction of *p*(X) compared to *p*) assigning to each element

$$x: B \rightarrow X$$

in the domain set B(X) an element

$$p(X)(x) = xp$$

in the codomain set A(X).

Summing up, what we have so far is the following diagram

A(X)

^

|

p(X)

|

B(X)

of two component sets and one structural function constituting a *Z*-structure denoted *Z*(X), which the object X (of the category ***C***) gave rise to. Along similar lines, object Y (of ***C***) gives rise to another *Z*-structure *Z*(Y) i.e.

A(Y)

^

|

p(Y)

|

B(Y)

Now we have to show that a morphism

f: X –> Y

in the category ***C*** gives rise to a morphism

Z(*f*): *Z*(X) –> *Z*(Y)

in the category of *Z*-structures i.e. a morphism from the object *Z*(X)

A(X)

^

|

p(X)

|

B(X)

to the object *Z*(Y)

A(Y)

^

|

p(Y)

|

B(Y)

Putting it all together, we need a pair of functions

$$f(A): A(X) \to A(Y)$$

$$f(B): B(X) \to B(Y)$$

making the diagram

A(X) –*f*(A)–> A(Y)

^ ^

| |

p(X) *p*(Y)

| |

B(X) –*f*(B)–> B(Y)

commute i.e. satisfy

$$p(Y) \circ f(B) = f(A) \circ p(X)$$

If that's not enough headache we need to show that the morphism

$$f\text{: A} \rightarrow \text{B}$$

gives rise to these functions

$$f\text{(A): A(X)} \rightarrow \text{A(Y)}$$

$$f\text{(B): B(X)} \rightarrow B\text{(Y)}$$

This added headache turns out to be the solution in the following sense. Take the case of

$$f\text{(A): A(X)} \rightarrow \text{A(Y)}$$

which has to assign to each element

$$xp\text{: A} \rightarrow \text{X}$$

of the domain set A(X) an element of the codomain set A(Y), but which one? The solution is to post-compose with

$$f\text{: A} \rightarrow \text{B}$$

i.e. define the function

$$f\text{(A): A(X)} \rightarrow \text{A(Y)}$$

as

$$f\text{(A)}(xp) = f\,(xp)$$

Similarly, we define the function

f(B): B(X) –> B(Y)

as

f(B)(*x*) = *fx*

With

f(A): A(X) –> A(Y)

f(B): B(X) –> B(Y)

defined as above, the diagram

A(X) –*f*(A)–> A(Y)

^ ^

| |

p(X) *p*(Y)

| |

B(X) –*f*(B)–> B(Y)

does commute i.e. satisfies

$$p(\mathrm{Y}) \circ f(\mathrm{B}) = f(\mathrm{A}) \circ p(\mathrm{X})$$

To be more explicit, let's start at the bottom-left corner with a B-shaped figure in X

$$x: \mathrm{B} \to \mathrm{X}$$

i.e. an element of B(X):

LHS: $p(\mathrm{Y}) \circ f(\mathrm{B})$

$p(\mathrm{Y}) \circ f(\mathrm{B})\ (x) = p(\mathrm{Y})\ (fx) = (fx)\ p$

RHS: $f(\mathrm{A}) \circ p(\mathrm{X})$

$f(\mathrm{A}) \circ p(\mathrm{X})\ (x) = f(\mathrm{A})\ (xp) = f\ (xp)$

If

$$(fx)\ p = f\ (xp)$$

then the above diagram commutes. Fortunately the associative law

$$(fx)\,p = f\,(xp)$$

says just that.

In other words, the commutativity of the diagram (morphism of *Z*-structures)

A(X) –*f*(A)–> A(Y)

^ ^

| |

p(X) *p*(Y)

| |

B(X) –*f*(B)–> B(Y)

induced by the morphism

$$f\colon X \to Y$$

(of the category ***C***) follows from the associativity of composition of morphisms

A – *p* –> B – *x* –> X – *f* –> Y

What if the family *Z* has more (or less) than the one morphism

$$p\text{: A} \rightarrow \text{B}$$

that we considered?

In general the *Z*-structure has a component set for each object and a structural function for each morphism in *Z*. These component sets and structural functions (that an object X of ***C*** gave rise to) together constitute an object of the category of *Z*-structures. A morphism in the category of *Z*-structures has as many component functions as the number of component sets (which is same as the number of objects in *Z*). All these component functions together are required to satisfy, in order to constitute a morphism, as many equations as the number of structural functions (which is same as the number of morphisms in *Z*). Morphisms (in so satisfying the commutativity equations) preserve, while transforming, all the structure of objects of the category of *Z*-structures.

Category of Reflexive Graphs

The category of reflexive graphs is a cohesive category (Lawvere, 2007), and hence serves as a model system to understand the cohesiveness of conscious experience. The cohesion or unity of conscious experience is a question of binding different modalities of vision, audition, emotions, thoughts, etc., each with its own mode of cohesion and variation into the cohesiveness of conscious experience that we experience (Roskies, 1999). In mathematics, different modes or kinds of cohesion are modeled as different categories. All these categories have (at one level of generalization) the structure of reflexive graphs. (I will elaborate on this in a separate note.) Since the category of reflexive graphs is cohesive, the category of categories is cohesive. If we think of the different modalities that go into the making of conscious experience as different categories (Category of Thoughts, Category of Feelings, etc.) and conscious experience as a category of these categories, we find the cohesion of conscious experience reflected in the cohesion of the category of categories. I must hasten to add that this no more than a suggestive analogy (at this point), which can serve as a thinking device in conceptualizing conscious experience.

In addition to the above category of categories, physical fields on a space (e.g. temperature in a room) can also be thought of as having reflexive graph structure (Lawvere and Schanuel, 2009) as follows.

Consider a space S representing a room and a map

$$t: S \to T$$

(where T is the temperature line) specifying the temperature at each point in the room. This 'temperature field' has reflexive graph structure when two points

$$d: 1 \to S$$

$$c: 1 \to S$$

in the space S are distinguished (1 is a one-point space). Since the domain set of the function

$$t: S \to T$$

is same as the codomain set of

$$d: 1 \to S$$

we can compose them to get a point on the temperature line T

$$1 \xrightarrow{d} S \xrightarrow{t} T = 1 \xrightarrow{td} T$$

which is the temperature at the point

$$d: 1 \to S$$

in the room S. So, corresponding to each temperature field in the room S i.e. corresponding to each function

$$t: S \to T$$

we have a temperature

$$td: 1 \to T$$

at the point

$$d: 1 \rightarrow S$$

in S. In other words, the function

$$d: 1 \rightarrow S$$

induces a function

$$d^*: T^S \rightarrow T^1$$

(where T^S is the set of all possible temperature fields on S and T^1 is the set of all temperature values on T) with

$$d^* (t: S \rightarrow T) = td: 1 \rightarrow T$$

giving the temperature td at the point d in S for a given temperature field t on S. Along the same lines, the second distinguished point

$$c: 1 \rightarrow S$$

induces another function

$$c^*: T^S \rightarrow T^1$$

with

$$c^* (t) = tc$$

giving the temperature tc at point c in S. Furthermore, the unique function

$$s: S \rightarrow 1$$

(mapping all points in S to the only point in 1) induces another function

$$s^*: T^1 \to T^S$$

mapping each point on the temperature line to a temperature field which has that temperature at every point in S.

The above temperature-field or T-field consisting of two sets (T^S and T^1) and three functions

$$d^*: T^S \to T^1$$

$$c^*: T^S \to T^1$$

$$s^*: T^1 \to T^S$$

has the structure of reflexive graphs as shown below.

A reflexive graph consists of two sets and three functions

$$p: A \to B$$

$$q: A \to B$$

$$r: B \to A$$

such that the function r is the common section of p and q. In other words,

$$pr = qr = 1_B$$

Now we have to show that the three functions

$$d^*: T^S \to T^1$$

$$c^*: T^S \to T^1$$

$$s^*: T^1 \text{ –> } T^S$$

corresponding to the T-field satisfy

$$d^*s^* = c^*s^* = 1_{T^1}$$

$d^*s^* (u) = d^* (us) = usd = u1_1 = u$

since $sd = 1 – d$ –> $S – s$ –> $1 = 1_1$ and $d^*s^* = 1_{T^1}$

$c^*s^* (u) = c^* (us) = usc = u1_1 = u$

since $sc = 1 – c$ –> $S – s$ –> $1 = 1_1$ and $c^*s^* = 1_{T^1}$

Thus for each physical quantity, such as temperature T, we obtain a reflexive graph. Transformations from one physical quantity into another induce transformations of the corresponding reflexive graphs. For example, given a transformation

$$f: T \text{ –> } V$$

we can pre-compose it with the T-field

$$t: S \text{ –> } T$$

to obtain a V-field

$$ft: S \text{ –> } V$$

on the space S. Pre-composing the V-field with a point

$$d: 1 \text{ –> } S$$

gives the value of the V-field

$$(ft)d: 1 \to V$$

at that point in space S. Alternatively, we could first find the value of the T-field

$$t: S \to T$$

at the point

$$d: 1 \to S$$

i.e.

$$td: 1 \to T$$

and post-compose it with the given transformation of fields

$$f: T \to V$$

to obtain the value

$$f(td): 1 \to V$$

of the V-field at the point

$$d: 1 \to S$$

in the space S. The associativity of composition of functions i.e.

$$(ft)d = f(td)$$

preserves the reflexive graph structure. In addition to the above equation corresponding to

$$d: 1 \to S$$

we have two more equations

$$(ft)c = f(tc)$$

$$f(us) = (fu)s$$

corresponding to

$$c: 1 \to S$$

$$s: S \to 1$$

all of which together constitute a structure-preserving transformation of reflexive graphs. Thus fields (such as temperature T on a space S with two distinguished points and a retraction of the space to a one-point space) along with their transformations (such as from T to V) can be construed as a category of reflexive graphs i.e. as a cohesive category.

References

Lawvere, F. W. (2007) Axiomatic Cohesion, Theory and Applications of Categories 19(3):41-49.

Lawvere, F. W. and Schanuel, S. H. (2009) Conceptual Mathematics, New York: Cambridge University Press, p. 151.

Roskies, A. L. (1999) The Binding Problem, Neuron 24(1):7-9.

Exercise 7 (Conceptual Mathematics, p. 293)

If

$$f = m \cdot e$$

$$f = f \cdot f$$

and

$$e \cdot m = 1_P$$

(where '·' denotes composition), then

$$m\text{: P} \rightarrow \text{A}$$

is an equalizer of the parallel pair of maps

$$f, 1_A\text{: A} \rightarrow \text{A}$$

(Conceptual Mathematics, p. 293).

An equalizer of a parallel pair of maps

$$f, g\text{: A} \rightarrow \text{B}$$

is a map

$$i\text{: X} \rightarrow \text{A}$$

satisfying

$$f \cdot i = g \cdot i$$

and every map

$$j\colon \mathrm{Y} \to \mathrm{A}$$

satisfying

$$f \cdot j = g \cdot j$$

is uniquely included in

$$i\colon \mathrm{X} \to \mathrm{A}$$

i.e. there is exactly one map

$$k\colon \mathrm{Y} \to \mathrm{X}$$

such that

$$j = i \cdot k$$

To show that

$$m\colon \mathrm{P} \to \mathrm{A}$$

is an equalizer of

$$f,\ 1_{\mathrm{A}}\colon \mathrm{A} \to \mathrm{A},$$

given

$$f = m \cdot e$$

$$f = f \cdot f$$

$$e \cdot m = 1_P,$$

we have to first show that

$$f \cdot m = 1_A \cdot m$$

i.e.

$$f \cdot m = m$$

$f \cdot m = (m \cdot e) \cdot m = m \cdot (e \cdot m) = m \cdot 1_P = m$

Next, we have to show that if

$$j\text{: Y –> A}$$

satisfies

$$f \cdot j = j$$

then there is exactly one map

$$k\text{: Y –> X}$$

satisfying

$$j = m \cdot k$$

First, we need a map from Y to P. Given

$$j\text{: Y –> A}$$

and

$$e: A \rightarrow P$$

we can take the composite map

$$e \cdot j: Y \rightarrow A \rightarrow P$$

as the map k from Y to P i.e.

$$k = e \cdot j$$

and see if

$$m \cdot k = j$$

i.e.

$$m \cdot (e \cdot j) = j$$

$m \cdot (e \cdot j) = (m \cdot e) \cdot j = f \cdot j = j$

(since we are given $f \cdot j = j$).

Finally, we have to show that

$$k: Y \rightarrow X$$

satisfying

$$m \cdot k = j$$

is unique i.e. if there is another map

$$k': Y \rightarrow X$$

satisfying

$$m \cdot k' = j$$

then

$$k' = k$$

Given

$$m \cdot k' = j = m \cdot k$$

post-composing with

$$e\text{: A –> P}$$

on both sides of

$$m \cdot k' = m \cdot k$$

we get

$e \cdot (m \cdot k') = e \cdot (m \cdot k)$

$(e \cdot m) \cdot k' = (e \cdot m) \cdot k$

$1_P \cdot k' = 1_P \cdot k$

$k' = k$

Thus the section

$$m\text{: P –> A}$$

of the splitting of an idempotent

$$f = m \cdot e\text{: A –> P –> A}$$

is an equalizer of

$$f, 1_A\text{: A –> A}$$

Tailpiece: An equalizer of two idempotents

$$f, g\text{: A –> A}$$

with a common section i.e.

$$f = m \cdot e$$

$$f = m \cdot e'$$

is the common section

$$m\text{: P –> A}$$

True or false?

Groundhog Day

I wish I can remember how it ends… until then, it's yet another glitch in the matrix: is a given adjoint of a functor

discrete: ***S*** –> ***F***

left or right adjoint? As if to compound my confusion, left adjoint coincides with right:

pieces = **points**

in the definition of quality type (see Axiomatic Cohesion) that I'm trying to understand.

Let's start with the functor

discrete: ***S*** –> ***F***

from the category ***S*** of sets to the category ***F*** of functions (simply because I have a vague feeling that this is how I went about clearing my confusion). The category ***S*** of sets has sets and functions as its objects and morphisms, respectively, while the category ***F*** of functions has functions and commutative squares as its objects and morphisms, respectively (Conceptual Mathematics, pp. 144-5). The functor

discrete: ***S*** –> ***F***

assigns to each object (set)

A

(in ***S***) its identity function (an object)

1_A: A –> A

(in ***F***) i.e.

$$\textbf{discrete}\ (A) = 1_A\colon A \to A$$

and to each morphism (function)

$$f\colon A \to B$$

(in ***S***) a morphism

$$\textbf{discrete}\ (f\colon A \to B) = \textbf{discrete}\ (A) \to \textbf{discrete}\ (B)$$

$$= \langle f, f \rangle\colon 1_A \to 1_B$$

(in ***F***) which is a commutative square

$$\begin{array}{ccc} A & \xrightarrow{\ f\ } & B \\ \wedge & & \wedge \\ | & & | \\ 1_A & & 1_B \\ | & & | \\ A & \xrightarrow{\ f\ } & B \end{array}$$

(satisfying $1_B \cdot f = f \cdot 1_A$, where '·' denotes composition).

Now the functors

$$\textbf{points}\colon \boldsymbol{F} \to \boldsymbol{S}$$

pieces: ***F*** –> ***S***

are adjoint functors of the functor

discrete: ***S*** –> ***F***

(see [or is it more like do] Exercise 6.14 on page 119 of Sets for Mathematics). And our job is to figure out which one is left adjoint and which one is right adjoint and, of course, why?

Let's start with the functor

points: ***F*** –> ***S***

which assigns to each object (function)

f: A –> B

(in ***F***) its domain set

A

(in ***S***) i.e.

points (*f*: A –> B) = A

and to each morphism from an object

f: A –> B

to an object

f': A' –> B'

i.e. to each commutative square

$$\begin{array}{ccc} B & \xrightarrow{\;w\;} & B' \\ \uparrow & & \uparrow \\ | & & | \\ f & & f' \\ | & & | \\ A & \xrightarrow{\;v\;} & A' \end{array}$$

(satisfying $f' \cdot v = w \cdot f$) a function

$$v: A \to A'$$

(in ***S***) i.e.

points ($f'\ v = w\ f$) = **points** (f) –> **points** (f')

= v: A –> A'

Looking at the definition of adjoint functor (Conceptual Mathematics, pp. 374-5), we realize that we could call

discrete: ***S*** –> ***F***

left adjoint to

points: ***F*** –> ***S***

if we can find a natural correspondence

d: **discrete** (X) –> *f*

$$p\colon X \to \textbf{points}\ (f)$$

for every object X in ***S*** and *f* in ***F***.

In other words, every function

$$p\colon X \to \textbf{points}\ (f)$$

(in ***S***; whose type is given as a value of the **points** functor) is determined by the function

$$n_X\colon X \to \textbf{points}\ (\textbf{discrete}\ (X))$$

and the determination

$$\textbf{points}\ (d\colon \textbf{discrete}\ (X) \to f)$$

is unique i.e. for every

$$p\colon X \to \textbf{points}\ (f)$$

(in ***S***) there is a unique

$$d\colon \textbf{discrete}\ (X) \to f$$

(in ***F***) such that

$$p = \textbf{points}\ (d) \cdot n_X$$

In (yet) other words, there is a natural transformation

$$n\colon 1_{S} \to \textbf{points} \cdot \textbf{discrete}$$

from the identity functor

$$1_{\boldsymbol{S}}: \boldsymbol{S} \to \boldsymbol{S}$$

(on ***S***) to the composite functor

$$\mathbf{points} \cdot \mathbf{discrete}: \boldsymbol{S} \to \boldsymbol{S}$$

(an endofunctor on ***S***), whose components are

$$n_{\mathrm{X}}: \mathrm{X} \to \mathbf{points}\ (\mathbf{discrete}\ (\mathrm{X}))$$

Summing up, so far, we say

$$\mathbf{discrete}: \boldsymbol{S} \to \boldsymbol{F}$$

functor is left adjoint to

$$\mathbf{points}: \boldsymbol{F} \to \boldsymbol{S}$$

functor if there is a natural transformation

$$n: 1_{\boldsymbol{S}} \to \mathbf{points} \cdot \mathbf{discrete}$$

What if, instead, the functor

$$\mathbf{discrete}: \boldsymbol{S} \to \boldsymbol{F}$$

is right adjoint to

$$\mathbf{points}: \boldsymbol{F} \to \boldsymbol{S}$$

functor? Well, then we would expect to see a natural correspondence

$$p': \textbf{points}\ (f) \to X$$

$$d': f \to \textbf{discrete}\ (X)$$

for every object X in ***S*** and *f* in ***F***.

In other words, every figure

$$p': \textbf{points}\ (f) \to X$$

(in ***S***; whose shape is given as a value of the **points** functor) is included in the figure

$$n'_X: \textbf{points}\ (\textbf{discrete}\ (X)) \to X$$

and the inclusion

$$\textbf{points}\ (d': f \to \textbf{discrete}\ (X))$$

is unique i.e. for every

$$p': \textbf{points}\ (f) \to X$$

(in ***S***) there is a unique

$$d': f \to \textbf{discrete}\ (X)$$

(in ***F***) such that

$$p' = \textbf{points}\ (d') \cdot n'_X$$

In (yet) other words, there is a natural transformation

n': **points** · **discrete** –> 1_S

from the composite functor

points · **discrete**: ***S*** –> ***S***

(an endofunctor on ***S***) to the identity functor

1_S: ***S*** –> ***S***

(on ***S***), whose components are

n'_X: **points** (**discrete** (X)) –> X

Summing up, so far, we say

discrete: ***S*** –> ***F***

functor is right adjoint to

points: ***F*** –> ***S***

functor if there is a natural transformation

n': **points** · **discrete** –> 1_S

Summing it all, if there's a natural transformation

n: 1_S –> **points** · **discrete**

we say

discrete is left adjoint to **points**

and if there is a natural transformation

$$n': \textbf{points} \cdot \textbf{discrete} \to 1_S$$

we say

discrete is right adjoint to **points**

Let's see: since

$$\textbf{points} \cdot \textbf{discrete}\ (X) = X$$

and, of course,

$$1_S\ (X) = X$$

and since we can take the identity function

$$1_X: X \to X$$

as components of both the natural transformations i.e. with

$$n_X: X \to \textbf{points}\ (\textbf{discrete}\ (X)) = 1_X: X \to X$$

$$n'_X: \textbf{points}\ (\textbf{discrete}\ (X)) \to X = 1_X: X \to X$$

we have both the natural transformations

$$n: 1_S \to \textbf{points} \cdot \textbf{discrete}$$

$$n': \textbf{points} \cdot \textbf{discrete} \to 1_S$$

which means

discrete is both left and right adjoint of **pieces**

But it’s clearly not the case:

discrete is left adjoint to **pieces**

(do Exercise 6.14 on page 119 of Sets for Mathematics). Are we doomed? No, it’s intermission and everything that could possibly go wrong goes wrong half-way through the movie…

Quality type

DEFINITION: A functor

$$q^*: \boldsymbol{S} \to \boldsymbol{F}$$

(between extensive categories) which is full and faithful and which is both reflective and coreflective by a single functor

$$q_! = q_*$$

makes ***F*** a quality type over ***S*** (Axiomatic Cohesion, p. 43; see also Exercise 5 on page 367 of Conceptual Mathematics).

EXAMPLE: The functor

discrete: ***S*** –> ***F***

(from the category ***S*** of sets to the category ***F*** of idempotents) has a right adjoint

points: ***F*** –> ***S***

which is also left adjoint to the **discrete** functor, and hence makes the category ***F*** of idempotents a quality type over the category ***S*** of sets.

First, we show that

discrete is left adjoint to **points**

and then show that

points is [also] left adjoint to **discrete**

(see Conceptual Mathematics, pp. 372-7).

To show that

discrete: ***S*** –> ***F***

is left adjoint to

points: ***F*** –> ***S***

we have to show that there is a natural transformation

n: **discrete • points** –> 1_F

(where '•' denotes composition) from the composite functor

discrete • points: ***F*** –> ***S*** –> ***F***

to the identity functor

1_F: ***F*** –> ***F***

The functor

points: ***F*** –> ***S***

assigns to each idempotent

e: X –> X, *ee* = *e*

(in the category ***F*** of idempotents) its set of fixed-points

Y

(in the category ***S*** of sets), which can be obtained by splitting the idempotent

e: X –> X

into its retract-section pair

X – *r* –> Y – *s* –> X

satisfying

sr = *e*

(Conceptual Mathematics, p. 102 & 117), i.e.

points (*e*: X –> Y –> X) = Y

and to each morphism

<*f*,*f*>: *e* –> *e*’

of idempotents i.e. to each commutative diagram

```
X        – – f – –>        X’

 ^                          ^

 |                          |

 e                          e’

 |                          |

X        – – f – –>        X’
```

(satisfying *e’ f* = *f e*) a function

g: Y –> Y’

(from the set Y of fixed-points of the idempotent *e*: X –> Y –> X to the set Y’ of fixed-points of the idempotent *e’*: X’ –> Y’ –> X’) satisfying

g r = *r’ f*

s’ g = *f s*

(where

X’ – *r’* –> Y’ – *s’* –> X’

is the splitting of the idempotent *e’*: X’ –> X’).

Next, the functor

discrete: ***S*** –> ***F***

assigns to each set

A

(in ***S***) its identity function

1_A: A –> A

(in ***F***) and to each function

υ: A –> B

a commutative square

$$\begin{array}{ccc} A & \xrightarrow{\upsilon} & B \\ \uparrow 1_A & & \uparrow 1_B \\ A & \xrightarrow{\upsilon} & B \end{array}$$

(satisfying $1_B\, \upsilon = \upsilon\, 1_A$).

Returning to the natural transformation

n: **discrete • points** –> 1_F

we need, for each idempotent

e: X –> X, *ee* = *e*

(in ***F***) a map

n_e: **discrete • points** (e) –> $1_{\boldsymbol{F}}$ (e)

(in ***F***). Since

discrete • points (e: X –> Y –> X) = **discrete** (Y) = 1_Y

$1_{\boldsymbol{F}}$ (e: X –> Y –> X) = e

we need a map from

1_Y: Y –> Y

to

e: X –> X

making the diagram

Y –– h ––> X

^ ^

| |

1_Y e

| |

Y –– h ––> X

commutative i.e. satisfying

$e\, h = h$

Since Y is the set of fixed-points of the idempotent

e: X –> X

obtained by splitting e, i.e.

$$X - e \to X = X - r \to Y - s \to X$$

we take

$$h = s: Y \to X$$

and find that

$$e\, h = s\, r\, s = s\, 1_Y = s = h$$

since $r\, s = 1_Y$ (Conceptual Mathematics, pp. 108-13). So we can take sections

$$s: Y \to X$$

of the splitting

$$e = s\, r$$

as components

$$n_e: 1_Y \to e$$

of the natural transformation

$$n: \textbf{discrete} \cdot \textbf{points} \to 1_{\boldsymbol{F}}$$

Next, for each morphism (in the category ***F*** of idempotents)

$$<f, f>: e \to e'$$

(from e: $X \to X$ to e': $X' \to X'$) we need a commutative square

discrete • points (e')	$\to$	1_F (e')
^		^
|		|
discrete • points ($<f, f>$)		1_F ($<f, f>$)

| |

discrete • points (*e*)   ––> $1_{\boldsymbol{F}}$ (*e*)

(in ***F***). Since

points (<*f*, *f*>: *e* –> *e'*) = **points** (*e*) –> **points** (*e'*) = *g*: Y –> Y'

(satisfying

$$g\ r = r'\,f$$

$$s'\,g = f\,s$$

where *s r* = *e* and *s' r'* = *e'*) and

discrete • points(<*f*, *f*>) = **discrete**(*g*: Y–>Y') = <*g*, *g*>: 1_Y –> $1_{Y'}$

and

$1_{\boldsymbol{F}}$ (<*f*, *f*>: *e* –> *e'*) = <*f*, *f*>: *e* –> *e'*

we find that we need the diagram

$1_{Y'}$ –– $n_{e'}$ ––>   *e'*

^   ^

| |

<*g*, *g*>   <*f*, *f*>

| |

1_Y –– n_e ––>   *e*

to commute i.e. satisfy

$$f\,n_e = n_{e'}\,g$$

With sections as components we have the diagram

```
Y'        -- s' -->        X'

^                          ^

|                          |

g                          f

|                          |

Y         -- s -->         X
```

commuting i.e. satisfying

$$f\,s = s'\,g$$

and in turn a natural transformation

n: **discrete • points** –> $1_{\boldsymbol{F}}$

which in turn tells that

discrete: ***S*** –> ***F***

is left adjoint to

points: ***F*** –> ***S***

Next, in order to show that the

points: ***F*** –> ***S***

functor is left adjoint to

discrete: ***S*** –> ***F***

we need a natural transformation

u: 1_{F} –> **discrete • points**

with components

u_{e}: 1_{F} (e: X –> X) –> **discrete • points** (e: X –> X)

making the diagram

```
e’          -- u_e’ -->          1_Y’

 ^                                ^
 |                                |
<f,f>                           <g, g>
 |                                |

e           -- u_e -->           1_Y
```

commute. Taking the retract

r: X –> Y

of the splitting

X – r –> Y – s –> X

of an idempotent

X – e –> X

as the component corresponding to the idempotent i.e. with

u_{e} = r: X –> Y

we find that the diagram

```
X’          -- r’ -->          Y’
```

^ ^

| |

f *g*

| |

X –– *r* –– Y

commutes i.e. satisfies

$$g\, r = r' f$$

i.e. we do have a natural transformation from the identity functor

$1_{\boldsymbol{F}}$: ***F*** –> ***F***

to the composite functor

discrete • points: ***F*** –> ***F***

Since we have both natural transformations

discrete • points –> $1_{\boldsymbol{F}}$

$1_{\boldsymbol{F}}$ –> **discrete • points**

we can say that the functor

points: ***F*** –> ***S***

is both right and left adjoint of the functor

discrete: ***S*** –> ***F***

all of which makes the category ***F*** of idempotents a quality type over the category ***S*** of sets.

Summary: The discrete functor from the category Sets to the category Idempotents, with points functor as its right and left adjoint, makes Idempotents a quality type over Sets.

Natural Structure

For any set S, the map set

$$S^C$$

(where C is the set of constants of a monoid M) has the **natural structure** of a right M-set (Conceptual Mathematics, p. 366).

Given a set

$$A = \{a_1, a_2\}$$

we can define four endomaps:

$$c_1 \colon A \dashrightarrow A$$

with $c_1(a_1) = a_1$ and $c_1(a_2) = a_1$

$$i \colon A \dashrightarrow A$$

with $i(a_1) = a_2$ and $i(a_2) = a_1$

$$1 \colon A \dashrightarrow A$$

with $1(a_1) = a_1$ and $1(a_2) = a_2$

$$c_2 \colon A \dashrightarrow A$$

with $c_2(a_1) = a_2$ and $c_2(a_2) = a_2$

Now let’s look at the set of these four endomaps

$$M = \{c_1, i, 1, c_2\}$$

What do we see? First note that since the elements of M are endomaps they can be composed with one another to get an element of the set M. Next

$$1 \text{ o } m = m = m \text{ o } 1$$

for all elements of M and ‘o’ denotes composition. The elements of M also satisfy the associative law i.e.

$$m \text{ o } (m' \text{ o } m'') = (m \text{ o } m') \text{ o } m''$$

where *m*, *m’*, and *m’’* are any three elements of M. Thanks to the identity element *1* and the associativity of composition ‘o’ of elements of M, we say that the set

$$M = \{c_1, i, 1, c_2\}$$

(of endomaps on the set A) is a **monoid** (Sets for Mathematics, p. 77).

What else do we see? Pre-composing the elements of the monoid M with one of its elements c_1, we notice that

$$c_1 \text{ o } c_1 = c_1$$

$$c_1 \text{ o } i = c_1$$

$$c_1 \text{ o } 1 = c_1$$

$$c_1 \text{ o } c_2 = c_1$$

Let us call an element *c* of M satisfying

$$c \circ m = c$$

a constant of the monoid (Conceptual Mathematics, p. 366). Since

$$c_1 \circ m = c_1$$

for all elements *m* of M, c_1 is a constant of the monoid M.

We also notice that

$$c_2 \circ c_1 = c_1$$

$$c_2 \circ i = c_2$$

$$c_2 \circ 1 = c_2$$

$$c_2 \circ c_2 = c_2$$

i.e. for all elements *m* of the monoid M

$$c_2 \circ m = c_2$$

So c_2 is another constant of the monoid M. Thus we have two constants in the set C of constants

$$C = \{c_1, c_2\}$$

of the monoid M.

YOU: What more there is to see?

Post-composing the elements of the monoid M with an element of the set of constants C i.e. with a constant, say, c_1, we find that

$$c_1 \circ c_1 = c_1$$

$$i \text{ o } c_1 = c_2$$

$$1 \text{ o } c_1 = c_1$$

$$c_2 \text{ o } c_1 = c_2$$

That is, we find that the composite is an element of C i.e. a constant. We also find that

$$c_1 \text{ o } c_2 = c_1$$

$$i \text{ o } c_2 = c_1$$

$$1 \text{ o } c_2 = c_2$$

$$c_2 \text{ o } c_2 = c_2$$

i.e. once again we find that the composite is a constant. Summing up, we say that the composite

$$m \text{ o } c$$

(where *m* is an element of the monoid M and *c* is an element of the set C of constants of M) is a constant (an element of C).

YOU: Are we done slaying the dead?

Nope, we just inhaled—life ;)

Consider a set, say,

$$S = \{s_1, s_2\}$$

Let's now look at functions

$$y\text{: C --> S}$$

There are a total of four functions

$$y_1: C \dashrightarrow S$$

with $y_1(c_1) = s_1$ and $y_1(c_2) = s_1$

$$y_2: C \dashrightarrow S$$

with $y_2(c_1) = s_2$ and $y_2(c_2) = s_1$

$$y_3: C \dashrightarrow S$$

with $y_3(c_1) = s_1$ and $y_3(c_2) = s_2$

$$y_4: C \dashrightarrow S$$

with $y_4(c_1) = s_2$ and $y_4(c_2) = s_2$

The set of all functions from C to S is the map set

$$S^C = \{y_1, y_2, y_3, y_4\}$$

YOU: What’s there, besides its four elements, to look in this map set S^C?

Natural Structure

For each one of the elements in the map set S^C i.e. for each one the functions

$$y: C \dashrightarrow S$$

we can define a new function

$$y{*}m: C \dashrightarrow S$$

(where m is an element of the monoid M) defined as

$$y^*m\ (c) = y\ (m \circ c)$$

This definition works because of the fact that post-composing elements m of M with a constant c gives a constant.

Take for example, the function

$$y_1\text{: C --> S}$$

with $y_1(c_1) = s_1$ and $y_1(c_2) = s_1$

We can define a new function

$$y_1{}^*m\text{: C --> S}$$

for each one of the elements m in the monoid M.

Taking $m = c_1$

$$y_1{}^*c_1\ (c) = y_1\ (c_1 \circ c)$$

for all c in the set of constants C.

More explicitly

$y_1{}^*c_1\ (c_1) = y_1\ (c_1 \circ c_1) = y_1\ (c_1) = s_1$

$y_1{}^*c_1\ (c_2) = y_1\ (c_1 \circ c_2) = y_1\ (c_1) = s_1$

Thus the new function, for $m = c_1$, is

$$y_1{}^*c_1\text{: C --> S}$$

with

$y_1 * c_1 (c_1) = s_1$

$y_1 * c_1 (c_2) = s_1$

Representable Functor

A set-valued function

$$Q: \boldsymbol{A} \dashrightarrow \boldsymbol{S}$$

is called **representable** if there is an object A in the domain category ***A*** and an element q in the set Q(A) such that for any object B of the category ***A***, the function

$$\boldsymbol{A}(A, B) \dashrightarrow Q(B)$$

(from the set ***A***(A, B) of A-shaped figures in B

$$b: A \dashrightarrow B$$

to the set Q(B), which is the value of the functor Q at B) assigning to each map

$$b: A \dashrightarrow B$$

the element

$$Q(b)\ (q)$$

is an isomorphism of sets (Sets for Mathematics, p. 248).

Consider a set

$$W = \{you, me\}$$

There are two elements in W.   Next consider another set

$$I = \{i\}$$

There are two functions from I to W. They are

$$you: I \dashrightarrow W$$

with

$$you(i) = \text{you}$$

and

$$me: I \dashrightarrow W$$

with

$$me(i) = \text{me}$$

Let

$$W^I = \{you, me\}$$

be the set of functions from I to W. Next, note down the fact that both sets

$$W \text{ and } W^I$$

have the same number of elements i.e.

$$|W| = |W^I| = 2$$

which means that there is an isomorphism

$$W = W^{I}$$

which is an opposed-pair of functions

$$f: W \dashrightarrow W^{I}$$

with

$$f(\text{you}) = \textit{you}: I \dashrightarrow W$$

$$f(\text{me}) = \textit{me}: I \dashrightarrow W$$

and

$$g: W^{I} \dashrightarrow W$$

with

$$g(\textit{you}: I \dashrightarrow W) = \textit{you}(\text{i})$$

$$g(\textit{me}: I \dashrightarrow W) = \textit{me}(\text{i})$$

satisfying

$$gf = 1_{W}$$

$$fg = 1_{W^{I}}$$

All of this may seem like an excessively elaborate discourse on a rather trivial matter: the number of elements of a set W is equal to the number of functions from the singleton set **1** = {•} to the set W.

A simple example of **representable functor**, as you might have guessed, is our "you, me" story, little generalized. More specifically, the identity functor

1: ***S*** --> ***S***

(from the category ***S*** of sets to the category ***S***) is a representable functor. What do I need to do in order to convince you that

1: ***S*** --> ***S***

is indeed a representable functor?

We need an object

A

of the domain category ***S*** of sets and an element

q

of the value of the set-valued functor 1: ***S*** --> ***S*** at A i.e. of the set

1(A)

Simply put, we need a set

A

and an element

q: **1** --> A

where **1** = {•} is a single-element set.

What does this set

$$A$$

and this element

$$q\colon \mathbf{1} \dashrightarrow A$$

have to do with representable functor?

For any set

$$B$$

the function

$$\boldsymbol{S}(A, B) \dashrightarrow 1(B)$$

from the set of functions (from A to B)

$$\boldsymbol{S}(A, B) = B^A$$

(to the value of our identity functor at the set B i.e.) to the set

$$1(B) = B$$

assigning to each function

$$b\colon A \dashrightarrow B$$

(in the set B^A of functions from A to B) the element

$$1(b)\,(q)$$

i.e. (since $1(b\colon A \dashrightarrow B) = b\colon A \dashrightarrow B$) the element

$$\mathbf{1} \xrightarrow{q} A \xrightarrow{b} B$$

is an isomorphism of sets. In other words, we need a set

$$A$$

and an element

$$q\colon \mathbf{1} \dashrightarrow A$$

such that for any set B

$$B^A = B$$

Thanks to our "you, me" story, we can readily guess that

$$A = \mathbf{1} = \{\bullet\}$$

and

$$q\colon \mathbf{1} \dashrightarrow \mathbf{1}$$

will make the assignment of each function

$$b\colon \mathbf{1} \dashrightarrow B$$

in the set

$$\boldsymbol{S}(\mathbf{1}, B) = B^{\mathbf{1}}$$

to the element

1 –*q*–> A –*b*–> B

in the set

$$1(B) = B$$

an isomorphism

$$B^{\mathbf{1}} = B$$

Right Actions and Codiscrete Inclusions

For any set S, the map set

$$S^C$$

(of all functions to S from C, where C is the set of all constants of a monoid M) has the natural structure of a right M-set. If C is not empty, then the resultant graph is codiscrete (Conceptual Mathematics, pp. 366-7 and 372-7).

Let's begin with dynamical systems (Conceptual Mathematics, 136). A dynamical system is a set of states (of, say, a light bulb):

$$A = \{on, off\}$$

equipped with a dynamic, say, a button switching the states, which we represent as an endomap

$$\alpha: A \rightarrow A$$

with α(on) = off and α(off) = on. Now, imagine another button which when pressed turns off the light if it's on and does nothing if the light is off, which we represent as another endomap

$$\alpha': A \rightarrow A$$

with α'(on) = off and α'(off) = off. We can combine these two endomaps into one map

$$\beta: B \times A \rightarrow A$$

with B = {α, α'}, β(α, a) = α(a) and β(α', a) = α'(a) for all elements 'a' of A. A set

A

equipped with a map

$$\beta: B \times A \rightarrow A$$

is called an action of B on A. If the set B = **1** (e.g. {α}), then actions

$$\beta: B \times A \rightarrow A$$

i.e.

$$\beta: \mathbf{1} \times A \rightarrow A$$

reduce to dynamical systems

$$\alpha: A \rightarrow A$$

since $\mathbf{1} \times A = A$. Thus the notion of action is a generalization of dynamical system so as to accommodate any number of dynamics on a given set of states (Conceptual Mathematics, pp. 218-9 and 303). There is more though: if the elements of B are endomaps such as the above α and α', then, since any two endomaps can be composed to get an endomap, say

$$\alpha \cdot \alpha': A \rightarrow A$$

(where '·' denotes composition), and since, by the associative law,

$$\alpha \cdot \alpha' (a) = \alpha(\alpha'(a))$$

we can require actions

$$\beta: B \times A \rightarrow A$$

to respect the composition of endomaps, which means, among other things, that the set B of endomaps should include, for every ordered pair (α, α') of endomaps in B, the composite endomap α · α' also. Let us now look at one such set of endomaps.

Given a two-element set

$$A = \{a_1, a_2\}$$

we find that there are a total of four endomaps:

$$c_1: A \rightarrow A$$

(with $c_1(a_1) = a_1$ and $c_1(a_2) = a_1$)

$$i: A \rightarrow A$$

(with $i(a_1) = a_2$ and $i(a_2) = a_1$)

$$1: A \rightarrow A$$

(with $1(a_1) = a_1$ and $1(a_2) = a_2$)

$$c_2: A \rightarrow A$$

(with $c_2(a_1) = a_2$ and $c_2(a_2) = a_2$). Now let's look at the set of these four endomaps

$$M = \{c_1, i, 1, c_2\}$$

What do we see?

First, note that since the elements of M are endomaps they can be composed with one another; since M is the set of all endomaps on A, the composite of any two elements of M is an element of M i.e. there is a function

$$w: M \times M \rightarrow M$$

with $w(m, m') = m \cdot m'$. Next, there is an element of M (which is the identity function above)

$$1: \mathbf{1} \rightarrow M$$

satisfying

$$1 \cdot m = m = m \cdot 1$$

for all elements m of M. The elements of M also satisfy the associative law i.e.

$$m'' \cdot (m' \cdot m) = (m'' \cdot m') \cdot m$$

where m, m', and m'' are any three elements of M. Thanks to the identity element 1 and the associativity of composition of elements of M, we say that the set

$$M = \{c_1, i, 1, c_2\}$$

(of endomaps of the set A) is a monoid (Sets for Mathematics, p. 77).

What else do we see?

Pre-composing the elements of the monoid M with one of its elements c_1, we notice that

$$c_1 \cdot c_1 = c_1$$

$$c_1 \cdot i = c_1$$

$$c_1 \cdot 1 = c_1$$

$$c_1 \cdot c_2 = c_1$$

Let us call an element c of a monoid M satisfying

$$c \cdot m = c$$

(for every element m of M) a constant of the monoid (Conceptual Mathematics, p. 366). Since

$$c_1 \cdot m = c_1$$

for all elements m of M, c_1 is a constant of the monoid M. We also notice that

$$c_2 \cdot c_1 = c_2$$

$$c_2 \cdot i = c_2$$

$$c_2 \cdot 1 = c_2$$

$$c_2 \cdot c_2 = c_2$$

i.e. for all elements m of the monoid M

$$c_2 \cdot m = c_2$$

So c_2 is another constant of the monoid M. Thus we have two elements in the set C of all constants

$$C = \{c_1, c_2\}$$

of the monoid M. (The other two elements (i, 1) of M do not satisfy $c \cdot m = c$ for all m in M; for example: $i \cdot c_1 \neq i$, $1 \cdot c_1 \neq 1$.)

YOU: What else is there to see?

Post-composing the elements of the monoid M with an element of the set C of constants i.e. with a constant, say, c_1, we find that the composite

$$c_1 \cdot c_1 = c_1$$

$$i \cdot c_1 = c_2$$

$$1 \cdot c_1 = c_1$$

$$c_2 \cdot c_1 = c_2$$

is an element of C i.e. a constant. We also find that the composite

$$c_1 \cdot c_2 = c_1$$

$$i \cdot c_2 = c_1$$

$$1 \cdot c_2 = c_2$$

$$c_2 \cdot c_2 = c_2$$

is a constant. Summing up, we say that the composite

$$m \cdot c$$

(where m is an element of the monoid M and c is an element of the set C of all constants of M) is a constant (an element of C).

YOU: Are we done slaying the dead?

Nope, we just inhaled—life ;)

Now looking back at the four endomaps

$$c_1: A \to A$$

$$i: A \to A$$

$$1: A \to A$$

$$c_2: A \to A$$

we realize that we can combine the four maps into a single map

$$l: M \times A \to A$$

with

$$l(m, a) = m(a)$$

Since m(a) is an element of A, we can look at the value of l at points of M x A, which have m(a) as their second-component i.e.

$$l(m', m(a)) = m'(m(a))$$

Since an element (a) of set (A) is a function from the terminal set **1** to A i.e.

$$a: \mathbf{1} \to A$$

we write the RHS of the above equation

$$m'(m(a))$$

as a composite of three functions

$$\mathbf{1} \xrightarrow{a} A \xrightarrow{m} A \xrightarrow{m'} A$$

and the associative law tells us that

$$m' \cdot (m \cdot a) = (m' \cdot m) \cdot a$$

which says that the value of an endomap m' at the value of another endomap m at 'a' is same as the value of the composite endomap $m' \cdot m$ at 'a', which can be summed up as

$$l\colon \mathrm{M} \times \mathrm{A} \to \mathrm{A}$$

satisfies

$$l(m' \cdot m, \mathrm{a}) = l(m', l(m, \mathrm{a}))$$

(for all m, m' in M and all ‘a’ in A) and depicted as a commutative diagram

$$\begin{array}{ccc} \mathrm{A} & \xleftarrow{\ l\ } & \mathrm{M} \times \mathrm{A} \\ \uparrow & & \uparrow \\ | & & | \\ l & & 1_{\mathrm{M}} \times l \\ | & & | \\ \mathrm{M} \times \mathrm{A} & \xleftarrow{\ w \times 1_{\mathrm{A}}\ } & \mathrm{M} \times \mathrm{M} \times \mathrm{A} \end{array}$$

satisfying

$$l \cdot (w \times 1_{\mathrm{A}}) = l \cdot (1_{\mathrm{M}} \times l)$$

Also note that

$$l(1, \mathrm{a}) = 1(\mathrm{a}) = \mathrm{a}$$

for all ‘a’ in A, where 1 is the unit of the monoid M. Now, there can be any number of maps from a product (such as M x A) to one of its factors (A), among these we single out those maps

$$l\colon \mathrm{M} \times \mathrm{A} \to \mathrm{A}$$

satisfying

$$l(m' \cdot m, \mathrm{a}) = l(m', l(m, \mathrm{a}))$$

$$l(1, \mathrm{a}) = \mathrm{a}$$

and call these maps

$$l\colon M \times A \to A$$

actions of a monoid M on a set A (Conceptual Mathematics, pp. 218-9 and p. 303). Now instead of the above

$$l\colon M \times A \to A$$

we can consider

$$r\colon A \times M \to A$$

(with M on the right instead of left) making the diagram

A   <– r –   A x M

^   ^

| |

r $r \times 1_M$

| |

A x M   <– $1_A \times w$ –   A x M x M

commute i.e. satisfying

$$r \cdot (1_A \times w) = r \cdot (r \times 1_M)$$

Any set A equipped with a map

$$r\colon A \times M \to A$$

satisfying

$$r(a, m' \cdot m) = r(r(a, m), m')$$

$$r(a, 1) = a$$

is called a right M-set (Conceptual Mathematics, pp. 360-7; see also Exercise 3.51 on p. 77 and pp. 167-92 of Sets for Mathematics). Note that a right M-set

$$r: A \times M \rightarrow A$$

is any family of endomaps

$$A \rightarrow A$$

parameterized by a set M of endomaps (which need not be the endomaps of A) in a way compatible with the monoid structure of M, which is what the requirement that

$$r: A \times M \rightarrow A$$

satisfy

$$r(a, m' \cdot m) = r(r(a, m), m')$$

$$r(a, 1) = a$$

means.

Let's go back to what got us into all this:

For any set S, the map set

$$S^C$$

(of all functions to S from C, where C is the set of all constants of a monoid M) has the natural structure of a right M-set (Conceptual Mathematics, p. 366).

Now that we know what a right M-set looks like, let's see if the map set S^C has the structure of a right M-set. Consider a set, say,

$$S = \{s_1, s_2\}$$

Let's now look at functions

$$y: C \rightarrow S$$

to the set S from the set

$$C = \{c_1, c_2\}$$

of all constants of the monoid

$$M = \{c_1, i, 1, c_2\}$$

There are a total of four functions

$$y_1: C \to S$$

(with $y_1(c_1) = s_1$ and $y_1(c_2) = s_1$)

$$y_2: C \to S$$

(with $y_2(c_1) = s_2$ and $y_2(c_2) = s_1$)

$$y_3: C \to S$$

(with $y_3(c_1) = s_1$ and $y_3(c_2) = s_2$)

$$y_4: C \to S$$

(with $y_4(c_1) = s_2$ and $y_4(c_2) = s_2$).

The set of all functions from C to S is the map set

$$S^C = \{y_1, y_2, y_3, y_4\}$$

Now we have to show that this map set S^C has the natural structure of a right M-set. (I am not sure how the qualifier 'natural' restricts structure.) Let's see if S^C can be equipped with a map

$$r: S^C \times M \to S^C$$

satisfying

$$r(y, m' \cdot m) = r(r(y, m), m')$$

(for all y: C –> S in S^C and all m, m' in M) and

$$r(y, 1) = y$$

(where 1 is the unit of the monoid M). First, since

$$r: S^C \times M \to S^C$$

is a family of endomaps (of the map set S^C) parameterized by M, we need as many endomaps of S^C as the number of elements of the monoid M. Since

$$M = \{c_1, i, 1, c_2\}$$

has four elements, we need an endomap

$$S^C \to S^C$$

for each one of the four elements of the monoid M. Next, we need to spell-out what

$$r(y, m)$$

means in the map

$$r: S^C \times M \to S^C$$

(besides mapping to an element of the codomain S^C) i.e. we need to specify the interaction between the first-factor y and the second-factor m of the domain S^C x M that r implements (Conceptual Mathematics, p. 218 and 302). Let's define

$$r(y, m) = y{*}m: C \to S$$

with

$$y{*}m\ (c) = y\ (m \cdot c)$$

This definition works because post-composing an element of C i.e. a constant c with elements m of M gives a constant i.e. an element of C (Conceptual Mathematics, p. 366). We thus obtain a total of 16 elements (functions C –> S) of the domain set S^C x M of r. Now, we check to see if the map

$$r: S^C \text{ x } M \text{ –> } S^C$$

satisfies

$$r(y, m' \cdot m) = r(r(y, m), m')$$

(for all y: C –> S in S^C and all m, m' in M) and

$$r(y, 1) = y$$

(where 1 is the unit of the monoid M). Beginning with an element y of S^C and an element m of M, we obtain an element

$$y^*m: C \text{ –> } S$$

of S^C x M. For example, with an element

$$y_1: C \text{ –> } S$$

(with $y_1(c_1) = s_1$ and $y_1(c_2) = s_1$) of S^C and with an element c_1 of M, we obtain an element

$$y_1{}^*c_1: C \text{ –> } S$$

of S^C x M defined as

$$y_1{}^*c_1\,(c) = y_1\,(c_1 \cdot c)$$

(for all c in the set C of constants). More explicitly

$$y_1{}^*c_1\,(c_1) = y_1\,(c_1 \cdot c_1) = y_1\,(c_1) = s_1$$

$$y_1{}^*c_1\,(c_2) = y_1\,(c_1 \cdot c_2) = y_1\,(c_1) = s_1$$

Looking at the values of

$$y_1{}^*c_1: C \text{ –> } S$$

we notice that

$$y_1{}^*c_1 = y_1$$

Since each element of S^C x M i.e. each

$$y*m: C \rightarrow S$$

is a function from C to S and since S^C is the set of all functions from C to S, every element $y*m$ of S^C x M will be equal to an element of S^C, which is what we found when we noticed

$$y_1*c_1 = y_1$$

and this equality specifies a map from S^C x M to S^C. Calculating the remaining elements of S^C x M, as above, gives a complete specification of the map

$$r: S^C \times M \rightarrow S^C$$

as

$$r(y_1, c_2) = y_1 \quad r(y_2, c_2) = y_1 \quad r(y_3, c_2) = y_4 \quad r(y_4, c_2) = y_4$$

$$r(y_1, 1) = y_1 \quad r(y_2, 1) = y_2 \quad r(y_3, 1) = y_3 \quad r(y_4, 1) = y_4$$

$$r(y_1, i) = y_1 \quad r(y_2, i) = y_3 \quad r(y_3, i) = y_2 \quad r(y_4, i) = y_4$$

$$r(y_1, c_1) = y_1 \quad r(y_2, c_1) = y_4 \quad r(y_3, c_1) = y_1 \quad r(y_4, c_1) = y_4$$

Now we have to check to see if the above

$$r: S^C \times M \rightarrow S^C$$

satisfies

$$r(y, m' \cdot m) = r(r(y, m), m')$$

(for all y: C –> S in S^C and all m, m' in M) and

$$r(y, 1) = y$$

(where 1 is the unit of the monoid M). We have to check if

$$r(y, m' \cdot m) = r(r(y, m), m')$$

for each triple y, m, m', which would be 64 equations. Looking at the third row (from bottom), we see that

$$r(y, 1) = y$$

and since

$$1 \cdot m = m = m \cdot 1$$

(for all m of M), we have

$$r(y, m' \cdot 1) \; [= r(y, m')] = r(r(y, 1), m') \; [= r(y, m')]$$

$$r(y, 1 \cdot m) \; [= r(y, m)] = r(r(y, m), 1) \; [= r(y, m)]$$

Looking up at the first and fourth columns, we see that, for all elements m of M,

$$r(y_1, m) = y_1$$

$$r(y_4, m) = y_4$$

which means we have

$$r(y, m' \cdot m) \; [= y] = r(r(y, m), m') \; [= r(y, m') = y]$$

for $y = y_1, y_2$. So we have to check at the remaining two elements (y_2 and y_3) of S^C and three elements (c_1, i, and c_2) of M, which would be a total of 18 equations to check. Calculating

$$r(y_2, m' \cdot m) \text{ and } r(r(y_2, m), m')$$

at the elements c_1, i, and c_2 of the monoid M, we find that

$$r(y_2, m' \cdot m) = r(r(y_2, m), m')$$

We also find, upon calculating

$$r(y_3, m' \cdot m) \text{ and } r(r(y_3, m), m')$$

at the elements c_1, i, and c_2 of M, that

$$r(y_3, m' \cdot m) = r(r(y_3, m), m')$$

Putting it all together, we find that

$$r(y, m' \cdot m) = r(r(y, m), m')$$

for all elements y of the map set S^C and for all ordered pairs of elements m, m' of the monoid M. So we can say that the map set S^C equipped with

$$r: S^C \times M \rightarrow S^C$$

is a right M-set.

Finally, going back to where we started:

If the set C (of all constants of a monoid M) is not empty, then the resultant graph is codiscrete (Conceptual Mathematics, pp. 366-7 and 372-7).

The map

$$J(S) = S^C$$

is a functor (from the category of sets to the category of reflexive graphs) assigning to each set

$$S$$

(in the category of sets) a codiscrete graph (in the category of reflexive graphs) with the map set

$$S^C$$

as its set of arrows (and S dots) and with the constants in C acting as source, target structural maps (Conceptual Mathematics, pp. 372-3; Sets for Mathematics, p. 22 and pp. 176-85). One of these days we can verify that this codiscrete inclusion

J: Sets –> Reflexive Graphs

is indeed a functor (see Exercise 5.17c on p. 109 of Sets for Mathematics), and then go on to see that the codiscrete functor is right adjoint to points functor, and then go even further to see that the unity and identity of adjoint opposites (i.e. codiscrete inclusion is opposite to discrete inclusion and both have points functor as common retract).

More importantly, since I seem to have some trouble seeing what exactly that we did, let's recap: We showed that the set S^C of all maps to a set S from the set C of all constants of a monoid M is a right-M set i.e. the map set S^C is equipped with an M-parameterized family of endomaps of S^C

$$r: S^C \text{ x } M \text{ –> } S^C$$

(with the mapping $r(y, m)$ defined as $y^*m\ (c) = y\ (m \cdot c)$, where y is an element of S^C, m is an element of the monoid M, and c is an element of the set C of all constants of the monoid M) compatible with the monoid composition

$$w: M \text{ x } M \text{ –> } M$$

i.e. making the diagram

S^C <– r – S^C x M

^ ^

| |

r r x 1_M

| |

S^C x M <– 1_{S^c} x w – S^C x M x M

commutative i.e. satisfying

$$r \cdot (1_{S^c} \text{ x } w) = r \cdot (r \text{ x } 1_M)$$

YOU: But, what does this have to do with cognitive science or consciousness studies?

For starters: codiscrete inclusions coincide with discrete inclusions in quality types (study Axiomatic Cohesion and do Exercise 5 on p. 367 of Conceptual Mathematics); for closers: the two canonical qualities form and substance surely figure in any serious study of things and their descriptions

FORM

^

|

SUBSTANCE <– Things

Summary: The set of all maps from the set C (of all constants of a monoid M) to a set S has the natural structure of a right M-set.

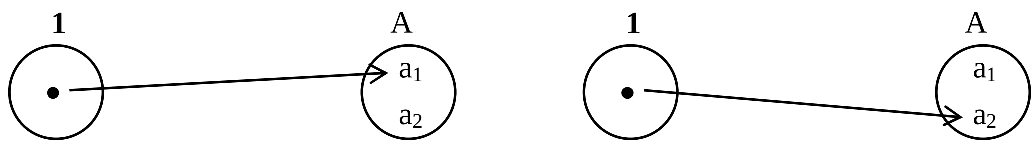
Describing Objects and Distinguishing Maps

Let's begin with a seemingly simple question: What is a single-element set **1** = {•} good for? First, note that the elements of any set A are in 1-1 correspondence with functions from the single-element set **1** to the set A. For example, with A = {a_1, a_2}, a two-element set, we have two functions from **1** to A. The internal diagrams of the two functions

$$a_1: \mathbf{1} \to A,\ a_1(\bullet) = a_1$$

$$a_2: \mathbf{1} \to A,\ a_2(\bullet) = a_2$$

are shown below:



So we can use the single-element set **1** to list all the elements of any set. In other words, any set is completely determined by the functions to the set from a one-element set **1**.

What else can we do with a single-element set **1**? A single-element set **1** can be used to test for the equality of functions. Notice that two functions

$$f: A \to B$$

$$g: A \to B$$

are equal

$$f = g$$

if and only if

$$f(\mathrm{a}) = g(\mathrm{a})$$

for every element ‘a’ of the domain set A. With elements of a set A as functions to the set A from the one-element set **1**, if there is a function

$$a: \mathbf{1} \rightarrow A$$

such that the composites

$$\mathbf{1} - a \rightarrow A - f \rightarrow B = \mathbf{1} - fa \rightarrow B$$

$$\mathbf{1} - a \rightarrow A - g \rightarrow B = \mathbf{1} - ga \rightarrow B$$

are not equal i.e. if

$$fa \neq ga$$

then the two functions are not equal i.e.

$$f \neq g.$$

Thus we find that we can use a one-element set **1** to completely characterize any set and also to tell apart functions.

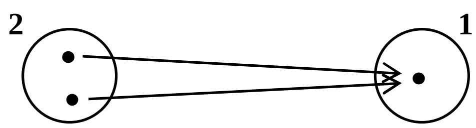
The ability to completely describe any object and to tell apart any two maps between objects is good thing to be able to do in any category. So, let us look for objects (in other categories) that play the role that the one-element set **1** plays in the category of sets. For this purpose, we need a more general description of the one-element set: a description which when interpreted in the category of sets gives the one-element set and when interpreted in other categories gives an object, which does what one-element set does in the category of sets. Is there any other way to describe the set **1** such that the description uniquely identifies the set **1** and only the set **1** in the category of sets? Does the set **1** have any other property (other than having exactly one element) based on which it can be distinguished from all other sets? In order to answer this question we need to consider the definition of function, which is a relation between sets. A function

f: A –> B

is an assignment of an element of the codomain set B to each element of the domain set A. Now, what can we say about the situation where the codomain set B has exactly one element. If B = **1**, then the function

c: A –> **1**

assigns all elements of the domain set A to the only element of the codomain set **1** (as shown below in the case of A = **2**).



More importantly, this assignment of all elements of the domain set A to the only element of the domain set **1** is the only function that there is from any set A to the set **1**. Even more importantly, this property of having 'exactly one function from any set' is a property unique to the set **1** i.e. no other set has this property. In other words, if a set T has exactly one function from any set A to the set T, then the set T is **1**. This description of **1** as the set to which there is exactly one function from every set (of the category of sets) is referred to as a universal mapping property; ‘mapping’ refers to fact that the description of the property is in terms of maps (of the category of sets, which are functions) as in: ‘there is exactly one function,’ while ‘universal’ refers to the fact that the description of the property is in terms of all objects (of the universe of discourse i.e. the category of sets, which are sets) as in: ‘from every set.’ Abstracting from the particular case of the category of sets to categories in general, we arrive at the definition of terminal object.

Definition: An object T of a category is called a terminal object of the category if there is exactly one map from every object of the category to the object T.

The terminal object of the category of sets is **1**. Maps from the terminal object of the category of sets can be used to completely determine every object and also to separate maps of the category (as we saw earlier). Now, is this true of terminal objects of other categories?

Let’s look at the category of functions. Objects of the category of functions are functions such as f: A –> B, g: C –> D. A map from an object f: A –> B to an object g: C –> D is a pair of functions

$$p: A \to C,\ q: B \to D$$

satisfying

$$qf = gp$$

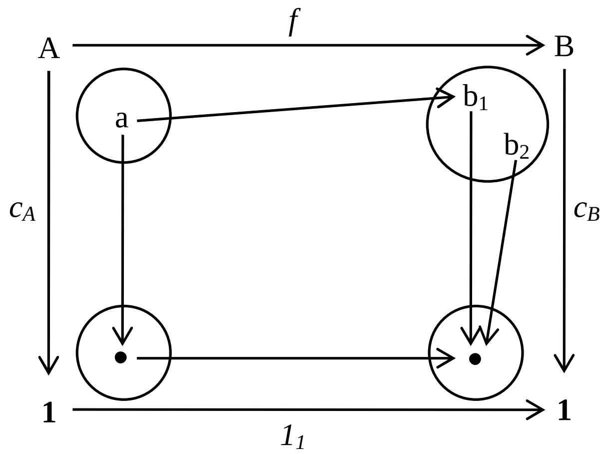
which makes the diagram

A –*f*–> B

p↓ *q*↓

C –*g*–> D

commute. Now we ask: what is the terminal object of the category of functions? The terminal object of the category of functions is a function to which there is exactly one map from every function. Since there is exactly one function from every function *f*: A –> B to the identity function 1_1: **1** –> **1** (of a single-element set), 1_1: **1** –> **1** is the terminal object of the category of functions. For example, with *f*: A –> B as displayed below



we do find that there is exactly one map from *f*: A –> B to the terminal object 1_1: **1** –> **1**, which is the pair of functions

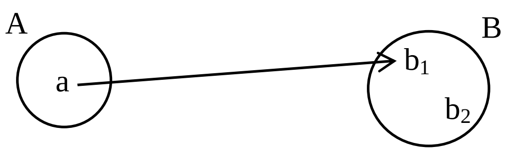
$$c_A: A \rightarrow \mathbf{1},\ c_B: B \rightarrow \mathbf{1}$$

satisfying

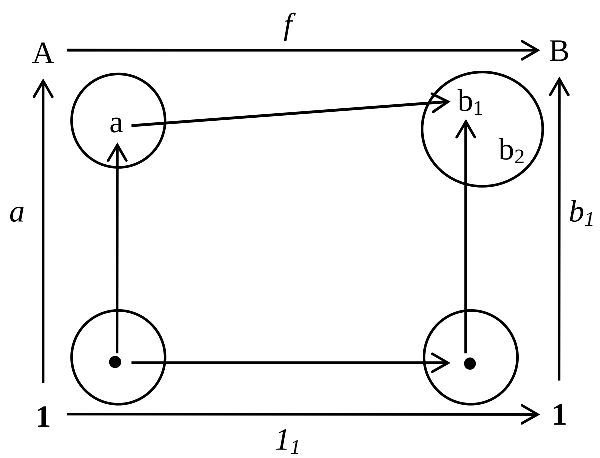
$$c_B\, f = 1_1\, c_A$$

and thereby making the above diagram commute.

Now we have to see if the terminal object 1_1: **1** –> **1** is sufficient to completely describe every object of the category of functions the way the terminal object **1** of the category of sets is sufficient to exhaustively characterize every set. More specifically, can we characterize every function using maps from the terminal object 1_1: **1** –> **1**? Let's consider the function *f*: A –> B displayed below:



There is an element 'a' in the domain set A and there are two elements b_1, b_2 in the codomain set B; one of which is the value of the function *f*: A –> B at the element 'a' i.e. *f*(a) = b_1. Now if we look at maps from the terminal object 1_1: **1** –> **1** to the function *f*: A –> B, we find that there is only one map from 1_1: **1** –> **1** to *f*: A –> B as shown below:

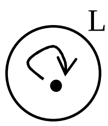


The above map, which is a pair of functions

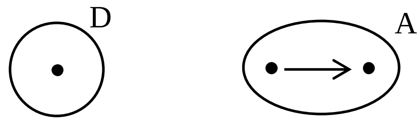
$$a\text{: } \mathbf{1} \text{ –> A, } b_1\text{: } \mathbf{1} \text{ –> B}$$

points to the domain element ‘a’ and the value of the function *f* at the element ‘a’ i.e. *f*(a) = b_1. Since there are no other maps from the terminal object 1_1: **1** –> **1** to *f*: A –> B, we find that there is no way to point to the element ‘b_2’ (of the codomain set B), which is not a value of the function *f* at any element of the domain set A. In order to point to codomain elements (of a function) which are not values of the function, we need maps from another object *U*: **0** –> **1**, which is a part of the terminal object. Thus we find that maps from the terminal object of the category of functions are not sufficient to completely describe every object of the category of functions. However, every object of the category of functions can be completely described using maps from a family of objects consisting of the terminal object 1_1: **1** –> **1** and its non-empty part *U*: **0** –> **1**. Since the terminal object **1** (of the category of sets) has no non-empty parts, we can summarize these two particular situations of the category of sets and the category of functions as follows. Every object of a category is completely described by maps from a family of objects consisting of the terminal object and its non-empty parts. Maps from this family of objects can also be used to separate maps of the category.

Let’s now check to see if the above generalization holds true in the case of other categories such as graphs and dynamical systems. In the category of graphs, the terminal object is a loop L with one dot and one arrow which has the only dot both as its source and target, as shown below:

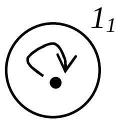


Since the graph maps *g*: L –> G from the terminal object L to a graph G are L-shaped figures in the graph G, we can only list loops of a graph using maps from the terminal object L. In order to exhaustively list the contents of a graph and to separate graph maps, we need a family of objects consisting of the non-empty part of the terminal object L i.e. a dot D and another object, which is the generic arrow A (both of which are depicted below):

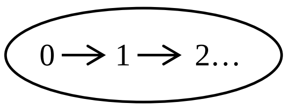


Maps from the above family of objects can also be used to separate graph maps.

Next, let's look at the situation in the category of dynamical systems. In the case of the category of dynamical systems the terminal object is a dynamical system 1_1: **1** –> **1**, which has one fixed state as shown below:



Just as in case of graphs, maps from the terminal object 1_1: **1** –> **1** can only point to fixed states of a dynamical system. We need a dynamical system *n*: N –> N, with *n*(n) = n + 1 and N = {0, 1, 2…} as shown below



in order to list all the states of any dynamical system and to separate maps of the category of dynamical systems. Notice that the above object that can serve as domain for maps used to completely describe any dynamical system is very much unlike the terminal object of the category of dynamical systems.

All of the above brings us to the main question: how are we going to generalize all of the above particular situations, wherein we have objects that serve as domains for maps capable of completely describing every object and also capable of separating maps in the category? It is clearly not the property of being a terminal object that allows the single-element set to completely list all the contents of any set since terminal objects in other categories are not capable of completely describing objects or distinguishing maps. Then, what is it about the single-element set that makes it do what it does? The answer is: single-element set is the essence which all sets partake in; it is the abstract general of all particular objects of the category of sets. That this is the right generalization of the particular situation with which we started can be seen by noting that the family of objects which are sufficient to describe all objects and also to separate maps in all of the above discussed categories also correspond to the essence or abstract general of their respective categories. More specifically, just as the set **1** = {•} is the essence of sets, the family of two functions 1_1: **1** –> **1**, *U*: **0** –> **1** is the essence of the category of functions. Similarly, the family of graphs consisting of a dot D and arrow A is the abstract general of graphs, while the dynamical system *n*: N –> N with *n*(n) = n + 1 and N = {0, 1, 2…} is the abstract general of dynamical systems. It is by virtue of being the essence(s) of their respective categories that these families of objects get to describe objects and distinguish maps in the corresponding categories. This brings us to the final question: given some particulars, how do we generalize or how do we abstract the essence(s) of the given particulars?