objects
counter
represent
(black)board
semantic
syntactic

Chapter 1

Accounting For *Basic* Collections Of Money *On* A Counter

In this chapter we will be dealing with **objects** sitting on a **counter**/desk/table/etc and we will **represent** these *objects* on a **(black)board**/notebook/etc. We will then design *procedures* to be carried on the *board* to arrive at a representation of the result of what we did on the *counter*.

But, if the distinction between what sits on the *counter* and what we write on the *board* is quite clear in the classroom, it is not as easy to make in a *book* and here we will have to resort to various devices.

- Inasmuch as possible, we shall use *pictures* to stand for *objects* on the counter but, as this is not always possible, we shall also use their *usual name* but with a particular typeface so as to distinguish them from what we will write to represent them on the *board*.

  For instance, we will use ***dollar*** as an alternate for  to stand for a dollar-bill sitting on the counter while we will write **Dollar** to represent it on the *board*.
- Similarly, we shall use ONE, TWO, . . . , TEN, ELEVEN, etc, to stand for the *numbers* of objects sitting on the *counter*, with the firm understanding that, on the *board*, we can write only 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- We shall use the symbol ♠ to signal that we are in the **semantic** mode, that is, *working on the counter* and the symbol ❖ to signal that we are in the **syntactic** mode, that is, *writing on the board*.

represent
denominator
collection
(counting) number-phrase
numerator
slash
/
number

1.1 Representing *Basic* Collections with (Counting) Number-Phrases

1. We begin with the issue of **representing** on the board money sitting on a *counter*. (Note by the way that banks used to be called "*counting* houses".)

For instance, given ***dollars*** and ***dimes*** on the counter, we use the words **Dollars** and **Dimes** as **denominators**, that is as names/symbols/denominations/etc to represent them on the *board*.

♠ On the *counter*.	❖ On the *board*.
We *have*  	We *write* **Dollar**, **Dollar**, **Dollar**, **Dollar**, **Dollar**, **Dime**, **Dime**

2. The first breakthrough in the development of arithmetic was the realization that objects of *different* kinds have to be accounted for *separately* because, when all the objects are of the *same* kind, we can *then* refer to them *collectively*, that is as a **collection**. For instance, we shall refer to ***dollar, dollar, dollar, dollar, dollar, dollar, dollar*** on the counter as a *collection* of ***dollars***. On the other hand, according to this agreement, ***dollar, dollar, dime, dime, dime, dime, dime*** will *not* be a *collection*. *What* would it be a collection of?

What this does is to allow us to represent a *collection* on the counter by *writing* on the board a **(counting) number-phrase**, that is a phrase consisting of:

- a **numerator** to indicate *how many* objects there are in the collection, which we do for the moment by writing on the board a **slash**, /, for each object in the collection on the counter, and,
- a *denominator* to indicate the *kind* of objects the collection is made of.

For instance,

♠ On the *counter*, we *have*:	❖ On the *board*, we *write*:
	/////// **Dollars**

We will say that the **number** of objects in a collection is *that* which is represented by the *numerator* in the number-phrase that represents the collection on the board. Thus, while the *denominator* represents the *kind* of objects in a collection, the *numerator* represents the *number* of objects in a collection.

quantity
quality
digit
1, 2, 3, ... , 9
succession
procedure
count

Observe that, even though a number-phrase is much more economical a way to represent on the board a collection of objects on the counter than writing one denominator for each object, there is *no* loss of information. Essentially, what we have done was merely to *separate* **quantity** *from* **quality** but, as it will turn out, this is a very powerful idea.

In particular, given, say, ***dollar***, ***dollar***, ***dollar*** on the counter, we can ask two very different questions:

- "*What* is on the counter?" whose answer on the board is the *number-phrase* /// **Dollars**
- "*How many* ***dollars*** are on the counter?" whose answer on the board is the *numerator* ///.

Note. We will need to make a distinction somewhat analogous to our use in English of "one dime" versus "a dime". We will distinguish between a *collection* consisting of one ***dime***, which we represent on the board by the *number-phrase* / **Dime**, and the *object* that a ***dime*** is, which we represent on the board by the *denominator* **Dime**. While this will surely appear as beyond nitpicking, not making the distinction would turn the development of board procedures into a nightmare.

3. The second breakthrough in the development of arithmetic occurred when Indian scribes introduced as *numerators* the **digits 1, 2, 3, ... , 9** to be used instead of /, //, ///, . . . , ///////// so that we now write, say, 3 **Dollars** instead of /// **Dollars**.

a. Once we have *memorized* the **succession** 1, 2, 3, . . . , 9, what this does is to give us a **procedure** to find the *numerator* of the (counting) number-phrase that represents a given *collection* of objects on the board: we **count** the collection, that is we point in turn at each object in the collection, while reciting the succession of digits. The *numerator* we write on the board is the *last* digit recited in the count.

For instance,

basic collection
default rule

♠ On the *counter*.	❖ On the *board*.
We *have* 	We *count* $\xrightarrow{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7}$ We *write* 7 **Dollars**

b. At this point, we can count only up to 9 **Dollars** because we cannot recite TEN as we have no symbol to represent TEN on the board. So, by a **basic collection**, we shall mean a collection with fewer than TEN objects which we can therefore count with just the above digits. At some fundamental level, *basic* collections are thus the only ones we can really *represent*! Reaching "TEN" will be the signal for "bundling" as we shall see in Section **??**.

Note. There is nothing sacred about TEN: it is simply how many fingers ("digit" is just a fancy word for finger) are on our two hands and we could have used just about any number of digits instead of TEN. For example, deep down, computers use a *machine language* based on TWO digits, 0 and 1, because any electronic device is either ***off*** or ***on***. At intermediate levels, computer software may use EIGHT (0, 1, 2, 3, 4, 5, 6, 7) or SIXTEEN digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f). The Babylonians used SIXTY digits, a historical remnant of which is the fact that there are SIXTY seconds to a minute and SIXTY minutes to an hour. The point here is that all that we will do with TEN could easily be done with *any* number of digits.

c. A small complication is that the numerator 1 often "goes without saying" which has the unfortunate effect of obliterating the difference between *denominator* and *number-phrase*. This is often expressed as a **default rule**:

When no numerator is given, the numerator 1 is intended and goes without saying.

Note. Unfortunately, this default rule is often abbreviated as "when there is no numerator, the numerator is 1" which is dangerous because, when there is *no numerator*, it is tempting to think that there is *no object* either! To be on the safe side, *we* will avoid letting the numerator 1 go "without saying".

d. Finally, we note that we have not yet introduced the digit 0. This is only because, so far, we have had no *need* for it. In fact, historically, the digit 0 was a much later invention. It will be introduced in Section 1.5.

Note. Since we refer to, say, ***dollar, dollar, dollar, dollar, dollar, dollar, dollar*** as a collection of ***dollars***, it is tempting to "improve" a bit and write "a collection of 7 ***dollars***" but we should resist the temptation because ***dollars*** are *objects* that sit on the *counter* while 7 is something we write on the *board* and we don't want to mix what is written on the board with what sits on the counter. On the other hand, we can speak of a collection of SEVEN ***dollars***.

compare
match one-to-one
relationship
leftover
count from ... to ...
is less numerous than
count forward
succeed

1.2 Comparing Collections: Equalities and Inequalities

We now want to **compare** collections—involving the *same* kind of objects. (We will compare collections involving *different* kinds of objects in Section **??**.)

1. We begin with the *comparison* of two collections on the *counter* and with the board *procedure* for *getting* the result of the comparison. We will deal with the issue of how to *represent* this result on the board in sub-section **2.** below.

♠ On the *counter*, what we do is to **match one-to-one** the *objects* in the two collections; the particular **relationship** that stands between the two collections will depend on which of the two collections the **leftover** objects are in.

❖ On the *board*, we count each one of the two collections and then we **count from** the numerator of the first number-phrase **to** the numerator of the second number-phrase, that is, starting *after* the numerator of the *first* number-phrase, we count to the numerator of the *second* number-phrase.

Either way, we then have three possibilities:

a. In the first case, that is

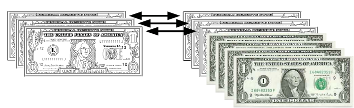
♠ When the leftover objects are in the *second* collection, we will say that the first collection **is less numerous than** the second collection.

❖ To count from the first numerator to the second one, *starting* with the digit *after* the first numerator, we must **count forward**, that is, we must call the digits that **succeed** it in the *succession* 1, 2, 3, 4, 5, 6, 7, 8, 9 and *end* with the second numerator.

For instance, $\xrightarrow{4,\ 5,\ 6,\ 7}$ is a *forward* count that starts *after* 3 and ends with 7.

For instance,

is more numerous than
count backward
precede
precession

♠ On the *counter*.	❖ On the *board*.
Jack has 	We *count* Jack's collection: $\xrightarrow{1,\ 2,\ 3}$
Jill has 	We *count* Jill's collection: $\xrightarrow{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7}$
We match ***Jack***'s collection one-to-one with ***Jill***'s collection: 	We count from ***Jack***'s collection to ***Jill***'s collection: $\xrightarrow{4,\ 5,\ 6,\ 7}$
Jack's collection is *less numerous than* ***Jill***'s collection	We must count *forward*.

b. In the second case, that is

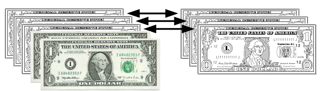
♠ When the leftover objects are in the *first* collection, we will say that the first collection **is more numerous than** the second collection.

❖ To count from the first numerator to the second one, *starting* with the digit *before* the first numerator, we must **count backward**, that is, we must call the digits that **precede** it in the *succession* 1, 2, 3, 4, 5, 6, 7, 8, 9 and *end* with the second numerator. For instance, $\xrightarrow{3,\ 4}$ is a *backward* count that starts *before* 5 and ends with 3.

Note. Thus, the **precession** 9, 8, 7, 6, 5, 4, 3, 2, 1 should be memorized as well as the *succession* 1, 2, 3, 4, 5, 6, 7, 8, 9.

For instance,

is as numerous as

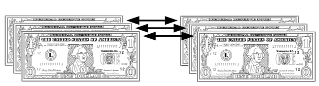
♠ On the *counter*.	❖ On the *board*.
Jack has 	We *count* Jack's collection: $\xrightarrow{1,\,2,\,3,\,4,\,5}$
Jill has 	We *count* Jill's collection: $\xrightarrow{1,\,2,\,3}$
We match ***Jack***'s collection one-to-one with ***Jill***'s collection: 	We count from ***Jack***'s collection to ***Jill***'s collection: $\xleftarrow{3,\,4}$
Jack's collection is *more numerous than* ***Jill***'s collection.	We must count *backward.*

c. In the third case, that is

♠ When there are *no* leftover objects, we will say that the first collection **is as numerous as** the second collection.

❖ The two numerators are the same and we must count neither forward nor backward.

For instance,

♠ On the *counter*.	❖ On the *board*.
Jack has 	We *count* Jack's collection: $\xrightarrow{1,\,2,\,3}$
Jill has 	We *count* Jill's collection: $\xrightarrow{1,\,2,\,3}$
We match ***Jack***'s collection one-to-one with ***Jill***'s collection: 	We count from ***Jack***'s collection to ***Jill***'s collection:
Jack's collection is *equal* to ***Jill***'s collection.	We must count neither forward nor backward.

2. In order to *represent* on the board the *result* of comparing two collections, we first need to expand our *mathematical* language beyond *number-*

verb
sentence
$<$
is smaller than
$>$
is larger than
$=$
is equal to
strict inequality
equality
endpoint

phrases.

a. Given a *relationship* between two collections, we need a **verb** that represents the *relationship* between the two collections. Then we can write a **sentence** involving the two *number-phrases* that represent the collections and the *verb* that represents the *relationship* between the two collections:

- We will use the *verb* $<$ to represent the relationship *is less numerous than* and we will read it **is smaller than**. For instance, for the first of the above three examples, we will write the *sentence* 3 **Dollars** $<$ 7 **Dollars** which we will read "THREE ***dollars*** is smaller than FIVE ***dollars***".
- We will use the *verb* $>$ to represent the relationship *is more numerous than* and we will read it **is larger than**. For instance, for the second of the above three examples, we will write the *sentence* 5 **Dollars** $>$ 3 **Dollars** which we will read "FIVE ***dollars*** is larger than THREE ***dollars***".
- We will use the *verb* $=$ to represent the relationship *is as numerous as* and we will read it **is equal to**. For instance, for the third of the above three examples, we will write the *sentence* 3 **Dollars** $=$ 3 **Dollars** which we will read "THREE ***dollars*** is equal to THREE ***dollars***".

In other words,

When we must count *forward* $\xrightarrow{\ ,\ \cdots\ ,\ }$	we write $<$	which is read as "is *smaller* than"
When we must count *backward* $\xleftarrow{\ ,\ \cdots\ ,\ }$	we write $>$	which is read as "is *larger* than"
When we must *not* count either way	we write $=$	which is read "is *equal* to"

Note. Beware that the symbols $<$ and $>$ go in directions *opposite* to that of the arrowheads when we count from the first numerator to the second numerator. (If need be, one can think of $<$ as $\cdot$: with $\cdot$ being "smaller" than : and of $>$ as : $\cdot$ with : being "larger" than $\cdot$.)

b. *Sentences* involving the *verbs* $>$ or $<$ are called **strict inequalities** while sentences involving the *verb* $=$ are called **equalities**. For example,

3 **Dollars** $<$ 7 **Dollars** and 8 **Dollars** $>$ 2 **Dollars** are *strict inequalities*
3 **Dollars** $=$ 3 **Dollars** is an *equality*

c. In English, when we say that we allow "up to" 5 **Dollars**, we mean that we allow 1 **Dollar**, 2 **Dollars**, 3 **Dollars**, 4 **Dollars** but that we do *not* allow the **endpoint** itself, 5 **Dollars**. If we do want also to allow the *endpoint*, 5 **Dollars**, we say "up to and including" 5 **Dollars**.

In mathematics we shall also need to make this distinction, that is, to allow or not to allow the *endpoint*, and, when we do allow it, we will say that the inequality is a **bounded inequality**:

- We will use the verb $\leqq$ to represent the relationship *is less numerous than or as numerous as* and we will read it **less than or equal to**.
- We will use the verb $\geqq$ to represent the relationship *is more numerous than or as numerous as* and we will read it **more than or equal to**.

d. Inasmuch as the sentences that we wrote above represented *actual* relationships between collections on the counter, they were **true** but there is of course nothing to prevent us from writing sentences that are **false** in the sense that there is no way that we could come up with *situations* that these sentences would represent. For example, the sentences

$$5 \textsf{ Dollars} = 3 \textsf{ Dollars} \quad \text{and} \quad 5 \textsf{ Dollars} < 3 \textsf{ Dollars},$$

are *false* because there is no way that we could **realize** them on the counter, that is come up with actual collections with these *relationships*.
Observe, by the way, that the sentence

$$5 \textsf{ Dollars} \leqq 3 \textsf{ Dollars}$$

is *true*.

e. However, while occasionally useful, it is usually not very convenient to write sentences that are *false* because then we must not forget to write that they are false when we write them and we may miss that it says somewhere that they are false when we read them. So, inasmuch as possible, we shall write only sentences that are *true* and we will use the *default rule*:

When no indication of truth or falsehood is given, mathematical sentences will be understood to be true and this will go without saying.

When a sentence is *false*, rather than writing *it* and say that it is *false*, what we shall usually do is to write *its* **negation**—which is *true* and therefore which "goes without saying". We can do this either in either one of two manners:

- We can place the *false* sentence within the symbol $\neg[\quad]$,
- We can just **slash** the *verb* which is what we shall usually do.

For instance, instead of writing that

the sentence $5 \textsf{ Dollars} = 3 \textsf{ Dollars}$ is *false*

we can either write the (true) sentence

$$\neg[5 \textsf{ Dollars} = 3 \textsf{ Dollars}]$$

or the (true) sentence

bounded inequality
$\leqq$
less than or equal to
$\geqq$
more than or equal to
true
false
realize
negation
$\neg[\quad]$
slash

(linguistic) duality
(linguistic) symmetry
opposite
dual

5 **Dollars** $\neq$ 3 **Dollars**

3. The **(linguistic) duality** that exists between $<$ and $>$ must not be confused with **(linguistic) symmetry**, a concept which we tend to be more familiar with.

a. Examples of linguistic *symmetry* include pairs of sentences—which may be *true* or *false*—such as the following:

- **Jack** is a child of **Jill** versus **Jill** is a child of **Jack**
- **Jill** beats **Jack** at poker versus **Jack** beats **Jill** at poker
- **Jack** loves **Jill** versus **Jill** loves **Jack**
- 9 **Dimes** $>$ 2 **Dimes** versus 2 **Dimes** $>$ 9 **Dimes**

In each example, the two sentences represent **opposite** relationships between the two people/collections because, even though the verbs are *the same*, the two people/collections are mentioned in *opposite* order.

Observe that just because one of the two sentences is *true* (or *false*) does *not*, by itself, automatically force the other to be either *true* or *false* and that whether or not it does depends on the *nature* of the relationship.

b. Examples of linguistic *duality* include:

- **Jack** is a *child* of **Jill** versus **Jill** is a *parent* of **Jack**
- **Jill** *beats* **Jack** at poker versus **Jack** *is beaten by* **Jill** at poker
- **Jack** *loves* **Jill** versus **Jill** *is loved by* **Jack**
- 9 **Dimes** $>$ 2 **Dimes** versus 2 **Dimes** $<$ 9 **Dimes**

In each example, the two sentences represent the *same* relationship between the two people/collections because, even though the people/collections are *mentioned in opposite order*, the two *verbs* are **dual** of each other which "undoes" the effect of the order so that only the *emphasis* is different.

Observe that here, as a result, if one of the two sentences is *true*(or *false*) this *automatically* forces the other to be *true* (or *false*) and this regardless of the *nature* of the relationship.

c. The following are examples of simultaneous (*linguistic*) *symmetry* and (*linguistic*) *duality* because the *verbs* are *the same* and the order does *not* matter.

- **Jack** is a sibling of **Jill** versus **Jill** is a sibling of **Jack**
- 2 **Nickels** $=$ 1 **Dime** versus 1 **Dime** $=$ 2 **Nickels**

Observe that, here again, as soon as one sentence is *true* (or *false*), by itself this *automatically* forces the other to be *true* (or *false*) and that it does not depend on the *nature* of the relationship.

1.3 Specifying Collections: Equations and “Inequations”

specify
requirement
satisfy
efficient
data

In real life, we often have to **specify** things we want by stating some **requirement(s)** that these things must **satisfy**.

Here, we will *specify* collection(s) by the *requirement* that they stand in a given *relationship* with a given *collection*, namely one or the other of the following,

- *is more numerous than* the *given* collection,
- *is less numerous than* the *given* collection,
- *is as numerous as* the *given* collection.

For instance, say that

> ***Jack*** has THREE ***dollars***,
> ***Jill*** has SEVEN ***dollars***,
> ***Dick*** has THREE ***dollars***,
> ***Jane*** has FOUR ***dollars***.

and that we specify the collection(s) that satisfy the *requirement* that they be *more numerous than* ***Jack***'s collection.

1. We could of course proceed as we did in Section 1.2:

♠ On the counter, matching ***Jack***'s collection one-to-one with each one of the collections of ***Jill***, ***Dick*** and ***Jane*** shows that this specifies the collections of ***Jill*** and ***Jane***.

❖ On the board, counting *from* ***Jack***'s collection each one of the collections of ***Jill***, ***Dick*** and ***Jane*** specifies the same collections.

This approach, though, is somewhat short of ideal if only because it would become very time-consuming with large numbers of collections to compare. So, what we want is to develop a board procedure that is more **efficient** in that the time it requires will not go up appreciably as the number of collections and of objects in the collections goes up.

2. Before we do that, though, we need a way to phrase *requirements* that lends itself to *procedural manipulations*.

a. Essentially, what we will do is to introduce the *mathematical* version of something common in everyday life, namely *forms* such as

$\boxed{\qquad\qquad\qquad}$ was President of the United States.

which, when we *fill* it it with some **data**, say,

Kissinger

produces a *sentence*, namely

solution
non-solution
unspecified numerator
equations
strict inequation
bounded inequation
replace
instruction
:=

$\boxed{\text{Kissinger}}$ was President of the United States.

which happens to be *false* while, when we fill it with the *data*

Bill Clinton

it produces the *sentence*

$\boxed{\text{Bill Clinton}}$ was President of the United States.

which happens to be *true*.

b. In the case of the above example,

♠ On the counter, we want the collections of ***dollars*** that *satisfy* the *requirement* that they be more numerous than THREE ***dollars***.

❖ On the board, we want the **solutions** of the form

$$\boxed{}\ \textbf{Dollars} > 3\ \textbf{Dollars}$$

Thus, from what we did above, we have that

- the *data* 7 produces the *sentence* $\boxed{7}$ **Dollars** > 3 **Dollars** which is *true*,
- the *data* 4 produces the *sentence* $\boxed{4}$ **Dollars** > 3 **Dollars** which is *true*,
- the *data* 3 produces the *sentence* $\boxed{3}$ **Dollars** > 3 **Dollars** which is *false*.

so that 7, 4 are *solutions* of the *form* $\boxed{}$ **Dollars** > 3 **Dollars** while 3 is a **non-solution**.

c. Boxes, though, would soon turn out to be impossibly difficult to use and, instead, we will use **unspecified numerators**, such as for instance the letter x, as in

$$x\ \textbf{Dollars}$$

and, instead of the form $\boxed{}$ **Dollars** > 3 **Dollars** we shall write

$$x\ \textbf{Dollars} > 3\ \textbf{Dollars}$$

We shall call:

- **equations** those forms whose *verb* is $=$,
- **strict inequations** those forms whose *verb* is either $<$ or $>$,
- **bounded inequations** those forms whose *verb* is either $\leqq$ or $\geqq$.

d. Instead of *filling the box* with the data, say, 3, we **replace** x by 3 and the **instruction** to do so will be

$$\big|_{\text{where } x:=3}$$

in which the symbol **:=**, borrowed from a computer language called PASCAL, is read as "is to be replaced by". Thus

$$x\ \textbf{Dollars}\big|_{\text{where } x:=3}$$

specifying-phrase
identity
identify
general
solution set

is a **specifying-phrase** in that it *specifies*

$$3 \textsf{ Dollars}$$

The following sentence

$$x \textsf{ Dollars}\big|_{\text{where } x:=3} = 3 \textsf{ Dollars}$$

is therefore "trivially" *true*; it is an example of a type of sentence called **identity** because it **identifies** the numerator specified by the *specifying-phrase*.
We also have

- $x \textsf{ Dollars}\big|_{\text{where } x:=7} > 3 \textsf{ Dollars}$,
- $x \textsf{ Dollars}\big|_{\text{where } x:=4} > 3 \textsf{ Dollars}$,
- $x \textsf{ Dollars}\big|_{\text{where } x:=3} \not> 3 \textsf{ Dollars}$.

3. We now turn to the simplest possible instance of a more **general** problem which is that we shall now want *all* the collections, if any, that stand in a given relationship with a given collection.
For example,

♠ Say ***Jack*** has FIVE ***dollars*** on the *counter.* We then want to find *all* collections of ***dollars*** that satisfy a given one of the following three *requirements*:
  i. *is less numerous* than ***Jack***'s collection,
  ii. *is more numerous* than ***Jack***'s collection,
  iii. *is as numerous* as ***Jack***'s collection.
  (In other words, we are looking here at *three* distinct problems at once.)

❖ On the *board*, we are looking for the **solution set** of the corresponding *inequation/equation*:
  i. $x \textsf{ Dollars} < 5 \textsf{ Dollars}$
  ii. $x \textsf{ Dollars} > 5 \textsf{ Dollars}$
  iii. $x \textsf{ Dollars} = 5 \textsf{ Dollars}$

We now proceed to do just so.
Regardless of which one of the three requirements we are trying to satisfy, we begin by considering the *equation*

$$x \textsf{ Dollars} = 5 \textsf{ Dollars}$$

whose *solution set* contains of course one, and only one, numerator: 5.
Then,

a. If it was the *equation* we were trying to solve, we are of course done.

break-even point
associated equation
pick
test
test-point
curly brackets

b. If we were trying to solve either one of the *inequations*

$$x \text{ Dollars} < 5 \text{ Dollars} \quad \text{or} \quad x \text{ Dollars} > 5 \text{ Dollars},$$

it remains to determine which side of the **break-even point** the *solution set* of the *inequation* is. (The *break-even point* is the solution of their **associated equation**, x Dollars $= 5$ Dollars, that is, of the *equation* obtained from the *inequation* by replacing the verb, $<$ or $>$, by the verb $=$.)

That the solution set must be a *whole side* of the break-even point is because if the solution set was only *part* of a whole side, then there would have to be both a *solution* and a *non-solution* on the *same* side of the break-even point and then there would have to be *another* break-even point in-between the two. But that cannot be since a break-even point is a solution of the *associated equation* x Dollars $= 5$ Dollars and we just saw that it has one and *only* one solution, namely 5.

So, on each side of the *break-even* point, all we need to do is to **pick** *one* numerator and **test** it against the wanted requirement, that is to ask whether this **test-point** is a *solution* or a *non-solution*: Then, every numerator on the same side of the break-even point as the test-point will be the same.

For instance, say we are looking at the *inequation*

$$x \text{ Dollars} > 5 \text{ Dollars}$$

The *associated equation* is

$$x \text{ Dollars} = 5 \text{ Dollars}$$

so that the *break-even point* of the *inequation* is 5. Then, on each side of 5, we pick a *test-point*. Say we *pick* 3 and 7. Since to count from 3 to 5 we have to count *forward*, 3 is *not* a solution and *all* numerators on the same side of 5 as 3 will *not* be solutions either. Since to count from 7 to 5 we have to count *backward*, 7 *is* a solution and *all* numerators on the same side of 5 as 7 *will* also be solutions so that the solution set of the *inequation*

$$x \text{ Dollars} > 5 \text{ Dollars}$$

is 6, 7, 8, ...

Note. It is customary, though, to write solutions sets in-between **curly brackets** as in $\{6, 7, 8, \ldots\}$ and we shall follow the custom.

Observe that the time we spent with the above procedure does *not* depend anymore on the number of collections we are dealing with.

Observe that, here, the *break-even point* is also an **endpoint** in that *all* the numerators on the *one* side of the break-even point *are* solutions and *all* the numerators on the *other* side of the break-even point are *not* solutions. This, though, will *not* be always the case and we will encounter *break-even points* that will turn out *not* to be *endpoint*s.

endpoint
operation
initial situation
action
terminal situation
direct problem
aggregation
aggregate
aggregate collection

1.4 Aggregating To A Collection. Addition.

Comparing collections is static in that nothing gets *created* and we now turn to **operations** on collections which are dynamic in that:

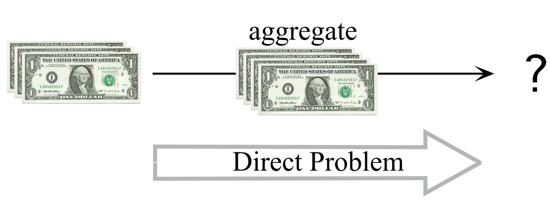
i. Starting from a given **initial situation**,
ii. We perform some **action** on the initial situation,
iii. Which results in some **terminal situation**.

Given an *operation*, we will be considering different types of *problems* that can be associated with the operation. In the simplest type, which we shall call **direct problems**, given an *initial* situation and an *action*, we want to find the *terminal* situation. (We call this a *direct* problem because it "goes in the *same* direction" as the *operation*.)

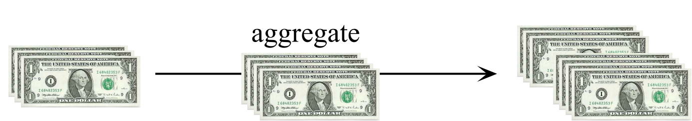
In this section, we consider *direct problems* associated with **aggregation**, an operation in which the *initial* situation involves a collection of objects, the *action* is to **aggregate** another collection of objects (of the same kind) and the *terminal* situation then involves the **aggregate collection**, namely the collection obtained by collecting all the objects in the two collections into one single collection.

For instance, a *direct problem* might be

♠ Aggregating FOUR ***dollars*** to THREE ***dollars***:



Performing the *action* of collecting all the objects in the two collections



+
addition
specifying-phrase
specify
identify
forward count
identifying sentence

gives the collection in the *terminal* situation:  .

❖ On the board,

i. In order to *state the problem* we use the symbol + to denote **addition**, the *procedure* that will give us the *numerator* of the number-phrase that represents the *aggregate* collection, and we write

$$3 \textbf{ Dollars} \xrightarrow{+\,4 \textbf{ Dollars}} 3 \textbf{ Dollars} + 4 \textbf{ Dollars}$$

where $3 \textbf{ Dollars} + 4 \textbf{ Dollars}$ is the **specifying-phrase** that represent on the board the *aggregate* collection *before we count it*. We shall call it a specifying-phrase because, while it is not a *number-phrase*, it does **specify** a *number-phrase* namely that which will be the *result* of the addition.

More succinctly, but less transparently, we shall usually write only the specifying phrase

$$3 \textbf{ Dollars} + 4 \textbf{ Dollars}$$

ii. In order to **identify** the collection specified by the specifying-phrase, we *count* the *initial* collection and then **forward count** the collection being aggregated, that is, starting *after* the count of the *initial* collection, we call the digits that *succeed* it in the succession 1, 2, 3, 4, 5, 6, 7, 8, 9 while pointing at the objects in the collection being *aggregated*. For instance, $\xrightarrow{4,\ 5,\ 6,\ 7}$ is a *forward* count that starts *after* 3 and ends with 7. The numerator of the number-phrase that represents the *aggregate* collection is the *end* of the forward count.

iii. In order to *represent* the solution of the direct problem, we write a *sentence* which we will call an **identifying sentence** because it *identifies* the number-phrase that was specified by the specifying-phrase:

$$3 \textbf{ Dollars} \xrightarrow{+\,4 \textbf{ Dollars}} 3 \textbf{ Dollars} + 4 \textbf{ Dollars} = 7 \textbf{ Dollars}$$

or, more succinctly,

$$3 \textbf{ Dollars} + 4 \textbf{ Dollars} = 7 \textbf{ Dollars}$$

Note. A *specifying-phrase* such as $3 \textbf{ Dollars} + 4 \textbf{ Dollars}$ is of course not to be confused with a *sentence* such as $3 \textbf{ Dollars} < 4 \textbf{ Dollars}$.

Altogether then,

general statement
reverse problem

♠ On the *counter*.	❖ On the *board*.
We have 	We write the *specifying phrase*[1] 3 **Dollars** + 4 **Dollars** We count the 1st collection: $\xrightarrow{1,\,2,\,3}$
to which we aggregate 	We count the 2nd collection *forward* starting *after* 3: $\xrightarrow{4,\,5,\,6,\,7}$
The *aggregate collection* is: 	The *numerator* of the result is 7. We write the *identifying sentence* 3 **Dollars** + 4 **Dollars** = 7 **Dollars**

So, we have the **general statement**:

When we aggregate on the counter, we add on the board by counting forward.

1.5 Removing From A Collection; Subtraction. (A *Reverse* Problem.)

We saw in Section 1.4 that, given an *operation*, a *direct* problem consists in performing a given action on a given initial situation and thus getting to some terminal situation—whatever that may turn out to be. In this section we consider a rather different kind of problem coming out of the fact that we are usually not ready to accept whatever terminal situation may happen to come up but, rather, that we usually have a *goal* in mind, namely a *specific* terminal situation that we *want*.

Generally speaking, we shall call **reverse problem** any problem involving a *wanted* terminal situation but there are two *types* of reverse problem depending on what else is *given* aside from the *wanted* terminal situation.

- If it is also the *initial situation* that is given, then what we must find is what *action* on this initial situation will get us the *wanted* terminal

[1] Educologists will surely appreciate this being "question oriented".

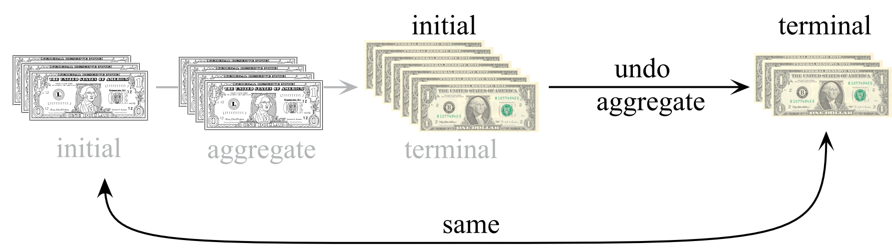
undo
remove

situation.

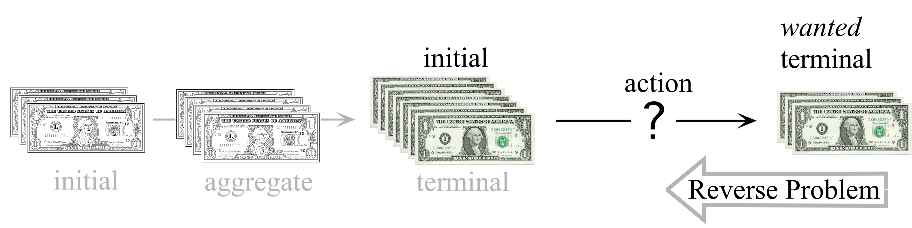
- If it is also the *action* that is given, then what we must find is for what *initial situation* will this action get us the *wanted* terminal situation.

In either case we call this a *reverse* problem because, since it starts from a wanted *terminal* situation, it "goes in the *opposite* direction" from the actual *operation*.

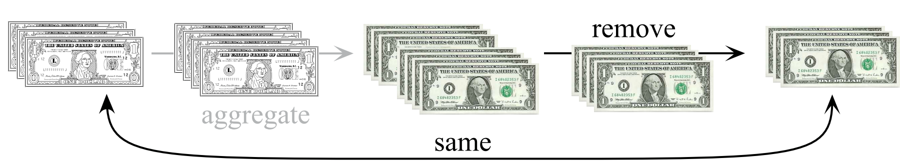
1. A special instance of a *reverse* problem arises when we want to **undo** the result of an *action*, that is, when we want to "return" from the *terminal* situation to the *initial* situation.
For instance, we might want to *undo* the *aggregation* of a collection to an *initial* collection.



The *reverse problem* associated with the "undo aggregate" operation then is to find what *action* on the *original* terminal situation will get us back to the *original* initial situation:



♠ On the *counter*, we must **remove** from the *original* terminal collection the collection that had been *aggregated* to the *original* initial collection.



❖ On the *board*, we count the terminal collection and then we count *backward* from the numerator of the *terminal* collection the collection that

had been *aggregated*. (This makes sense since, in the original operation, we obtained the numerator of the *terminal* collection by counting *forward* from the numerator of the *initial* collection the collection being *aggregated*.) subtraction $-$
We shall say that we **subtract** the numerator of the collection being *aggregated*—in the original operation—from the numerator of the *terminal* collection and we use the symbol $-$ to write, for instance,

$$7 \textsf{ Dollars} \xrightarrow{-\,4 \textsf{ Dollars}} 7 \textsf{ Dollars} - 4 \textsf{ Dollars}$$

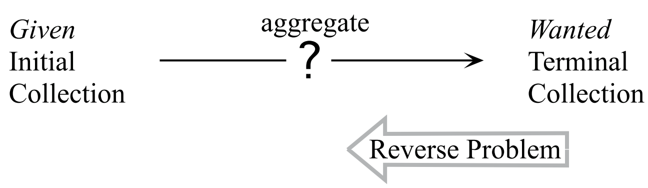
where 7 **Dollars** $-$ 4 **Dollars** is the *specifying-phrase* that represent on the board the *leftover* collection *before we count it*, namely that which will be the *result* of the subtraction.

In other words,

♠ On the *counter*.	❖ On the *board*.
From the *terminal* collection in the original operation, 	We write the *specifying phrase* 7 **Dollars** $-$ 4 **Dollars**
	We count the *terminal* collection: $\xrightarrow{1,\,2,\,3,\,4,\,5,\,6,\,7}$
we *remove* the collection that had been *aggregated* 	
	We count the *aggregated* collection *backward* starting *after* 7: $\xleftarrow{3,\,4,\,5,\,6}$
The *leftover* collection	The *numerator* of the *initial* collection is 3.
 is the *initial* collection.	We write the *sentence* 7 **Dollars** $-$ 4 **Dollars** $= 3$ **Dollars**

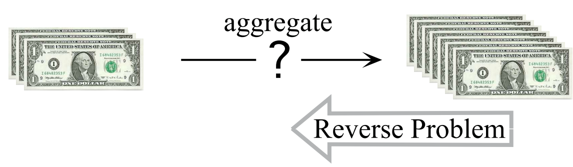
2. We now turn to a reverse problem that is more *general* in that, given *any* initial collection and *any* wanted terminal collection, we shall now want to find what collection (if possible) we have to *aggregate* to the *given* initial collection to get the *wanted* terminal collection.

empty



Here, though, because the *wanted* terminal collection need not have resulted from the aggregation of a collection, but can now be *any* collection, the reverse problem, as we shall see, may or may not have a *solution*. There are three cases.

a. The *wanted* terminal collection is *more numerous than* the *given* initial collection:



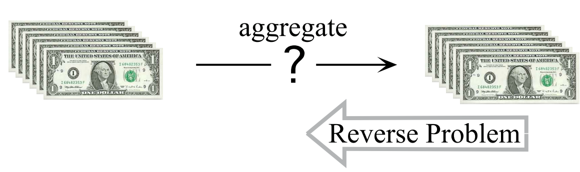
To find the collection to be aggragated, we proceed essentially as when we wanted to undo aggregation

♠ On the *counter*, we *remove* the *initial* collection from the wanted *terminal* collection.

❖ On the *board*, we *subtract* the numerator of the *initial* collection from the numerator of the wanted *terminal* collection, that is, we count the *initial* collection *backward* from the numerator of the wanted *terminal* collection.

partial undo

b. The wanted terminal collection is *as numerous as* the initial collection.



♠ There is *just* enough in the *wanted* terminal situation to remove the given first collection but, rather than to say that there *is* no *leftover* collection, we shall say that the leftover collection is **empty**.

❖ We subtract the *initial* numerator from the *terminal* numerator, that is, after we have counted the *wanted* terminal collection we count the initial collection *backward* but here we must introduce a new digit, **0**, to end the backward count. The digit 0 is thus the numerator in the number-phrase that represents any *empty* collection. 0

Note. As a matter of historical fact, 0 was invented much later than the other digits and *not* for this purpose. We shall see the historical purpose in Section **??**

c. The wanted terminal collection is *less numerous than* the initial collection.

♠ There is *not* enough in the *wanted* terminal situation to remove the given first collection. The reverse problem has no solution.

❖ On the board, we cannot *subtract* the *initial* numerator from the *terminal* numerator because we cannot count *backward* more than we counted in the first place!

Thus, say, specifying-phrases such as 3 **Dollars** − 5 **Dollars** make no sense whatsoever.

So, we have the *general statement*:

When we remove on the counter, we subtract on the board by counting backward.

3. We can now look at more complicated problems in which we would be looking for the *solution set* of one of the following

i. 3 **Dollars** $+ x$ **Dollars** < 7 **Dollars** or 3 **Dollars** $+ x$ **Dollars** $\leqq 7$ **Dollars**

ii. 3 **Dollars** $+ x$ **Dollars** > 7 **Dollars** or 3 **Dollars** $+ x$ **Dollars** $\geqq 7$ **Dollars**

iii. 3 **Dollars** $+ x$ **Dollars** $= 7$ **Dollars**

For instance, say the *initial* situation is that ***Jack*** has THREE ***dollars*** that he will donate to ***Jill*** but that the *wanted* terminal situation is that ***Jill*** should have a collection *more numerous than* SEVEN ***dollars***. The question thus is what collection should be aggregated to ***Jack***'s collection.

♠ On the counter, removing the THREE ***dollars*** in ***Jack***'s collection from SEVEN ***dollars***, we find that FOUR ***dollars*** are leftover. So, if we *aggregate* FOUR ***dollars*** to ***Jack***'s collection, then the aggregated collection will be *as numerous as* ***Jill***'s collection and aggregating a collection more numerous than FOUR ***dollars*** to ***Jack***'s collection will make ***Jill***'s collection *more numerous than* SEVEN.***dollars***

❖ On the board

- ***Jack***'s collection is represented by 3 **Dollars** and ***Jill***'s collection is represented by 7 **Dollars** and thus we are trying to find the solution(s), if any, of the *inequation*

$$3 \textbf{ Dollars} + x \textbf{ Dollars} > 7 \textbf{ Dollars}$$

bunch

– To obtain the *break-even point*, that is the solution of the *associated equation*,

$$3 \textsf{ Dollars} + x \textsf{ Dollars} = 7 \textsf{ Dollars}$$

we must *identify*

$$7 \textsf{ Dollars} - 3 \textsf{ Dollars}$$

that is we must count from 3 to 7:

$$3 \xrightarrow{4,\ 5,\ 6,\ 7} 7$$

which is a *forward* count of 4. Thus the *break-even point* is $7 \textsf{ Dollars} - 3 \textsf{ Dollars} = 4 \textsf{ Dollars}$.

– We pick a test-point on each side of the break-even point, say $2 \textsf{ Dollars}$ and $5 \textsf{ Dollars}$.
By counting from 3, we get:

$$3 \textsf{ Dollars} + 2 \textsf{ Dollars} \not> 7 \textsf{ Dollars}$$

and

$$3 \textsf{ Dollars} + 5 \textsf{ Dollars} > 7 \textsf{ Dollars}$$

So the *solution set* of $3 \textsf{ Dollars} + x \textsf{ Dollars} > 7 \textsf{ Dollars}$ is $\{5, 6, 7, \dots\}$.

1.6 Combinations.

Situations in the real world are rarely that simple that they only involve one single kind of objects. As it turns out, though, only a small but far-reaching adjustment needs to be made to what we have done so far.

1. When the objects are *not all of the same kind*, that is when we do *not* have a collection and therefore we cannot represent them by a (counting) number-phrase.
For instance, say we have ***dime***, ***dime***, ***nickel***, ***nickel***, ***nickel***, ***nickel***, ***nickel***, on the counter. Of course, we could write $7 \textsf{ Coins}$ but then we would be losing information, for instance, about how much *money* there is. Moreover, what could we write to represent, say, ***dollar***, ***dollar***, ***dime***, ***dime***, ***dime***, ***dime***, ***nickel***, ***nickel***, ***nickel***?

In the latter case, for instance, and in accordance with the "second breakthrough in the development of arithmetic" (Section 1.1), we begin by *separating* the objects into a **bunch** of collections:

- the collection ***dollar***, ***dollar***, which we can represent by the (counting) number-phrase $2 \textsf{ Dollars}$

- the collection ***dime***, ***dime dime***, ***dime***, which we can represent by the (counting) number-phrase 4 **Dimes**
- the collection ***nickel***, ***nickel***, ***nickel***, which we can represent by the (counting) number-phrases 3 **Nickels**

& combination attach

Then, we represent the *bunch* on the board by writing 2 **Dollars** & 4 **Dimes** & 3 **Nickels** where **&** means "and". We will call this a **combination** of **Dollars**, **Dimes** and **Nickels**. Thus *combinations* represent on the board *bunches* of collections on the counter.

Combinations are a very powerful concept that comes up again and again and that, in fact, is the subject of a whole part of mathematics called LINEAR ALGEBRA.

Note. Here again, it is usual to write, say, 2 **Dollars** & **Dime** but while we *see* of course that **Dollars** is the denominator in a number-phrase whose numerator is 2, we have to *remember* that **Dime** is not a *denominator* but really stands for a *number-phrase* whose numerator is 1 and whose denominator is **Dime** so that 2 **Dollars** & **Dime** stands for 2 **Dollars** & 1 **Dime**. *We* will always write, say, 2 **Dollars** & 1 **Dime** rather than 2 **Dollars** & **Dime**.

2. In the absence of any additional information, we cannot *compare* bunches of collections. We shall see in Section **??** what kind of information permits what kind of comparison.

3. When two collections consist of *different* kinds of objects, we cannot *aggregate* them since the result would not be a *collection* but a *bunch* of two collections.

However, we can **attach** bunches of collections and the result is still just another bunch represented by a combination. For instance,

$$\begin{aligned}
2\ \textsf{Dollars}\ \&\ 3\ \textsf{Nickels} + 4\ \textsf{Dollars}\ \&\ 5\ \textsf{Dimes} &= \textsf{Dollar},\ \textsf{Dollar},\\
&\quad\ \textsf{Nickel},\ \textsf{Nickel},\ \textsf{Nickel},\\
&\quad\ \textsf{Dollar},\ \textsf{Dollar},\ \textsf{Dollar},\ \textsf{Dollar},\\
&\quad\ \textsf{Dime},\ \textsf{Dime},\ \textsf{Dime},\ \textsf{Dime},\ \textsf{Dime},\\
&= 6\ \textsf{Dollars}\ \&\ 5\ \textsf{Dimes}\ \&\ 3\ \textsf{Nickels}
\end{aligned}$$

4. Neither, when two collections consist of *different* kinds of objects, can we cannot *remove* one from the other.

Occasionally, we can *detach* one bunch from another and the result being usually a bunch. For instance,

$$\begin{aligned} 7\ \textsf{Dollars}\ \&\ 5\ \textsf{Nickels}\ \&\ 9\ \textsf{Dimes}\ &- 4\ \textsf{Dollars}\ \&\ 1\ \textsf{Dime} \\ &= 7\ \textsf{Dollars} - 4\ \textsf{Dollars}\ \&\ 5\ \textsf{Nickels}\ \&\ 9\ \textsf{Dimes} - 1\ \textsf{Dime} \\ &= 3\ \textsf{Dollars}\ \&\ 5\ \textsf{Nickels}\ \&\ 8\ \textsf{Dimes} \end{aligned}$$

However, most of the time we cannot as, for instance, in

$$7\ \textsf{Dollars}\ \&\ 5\ \textsf{Nickels} - 4\ \textsf{Dollars}\ \&\ 3\ \textsf{Dimes}$$