Viewing function composition as transformation of the domain

devblogs.microsoft.com/oldnewthing/20060207-08

February 7, 2006

Raymond Chen

A lot of formulas you encounter in computer science can be viewed as function composition. Let's start with the simple problem of rounding integers down to the nearest multiple of some positive constant. The formula for this should be relatively easy for you to produce:

> round_down(n, m) = floor_div(n, m) * m

where `floor_div` returns the largest integer less than or equal to n/m. If n≥0 and m>0, then `floor_div(n,m) = n/m` where `/` is the C integer division operator.

But what if you want to round up? Take a look at the difference between rounding up and rounding down, say, using multiples of four for concreteness.

	0	1	2	3	4	5	6	7	8	9	10	11	12
round_down	0	0	0	0	4	4	4	4	8	8	8	8	12
round_up	0	4	4	4	4	8	8	8	8	12	12	12	12

The `round_up` table is just the `round_down` table shifted left three places. The mathematical way of shifting the table heading is by manipulating the domain, in this case, by adding three. In other words, don't think of adding three as a vertical operation

0	1	2	3	4	5	6	7	8	9	10	11	12
+3	+3	+3	+3	+3	+3	+3	+3	+3	+3	+3	+3	+3
3	4	5	6	7	8	9	10	11	12	13	14	15

but rather as a horizontal one:

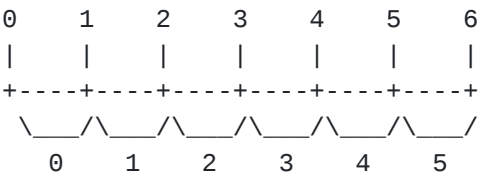
0	1	2	3	4	5	6	7	8	9	10	11	12
<- move left three spaces												
3	4	5	6	7	8	9	10	11	12	13	14	15

(Sorry, I'm too lazy to cook up the appropriate VML diagram. Use your imagination and pretend that there is an arrow from the "3" in the top row to the "3" in the bottom row, similarly from the "4" in the top row to the "4" in the bottom row, and so on.)

Now that you see that the answer is to "move the results" three spots to the left, you can read off that the desired formula is

> round_up(n, 4) = round_down(n + 3, 4)

Shifting the domain left and right can be done by addition. Multiplication and division let you stretch and shrink it. Consider the puzzle of rounding down to the nearest quarter. You already know how to round down to the nearest unit, namely by using the language's built-in truncation operator.



If only we could divide everything in the diagram by four. But we can! To do this, we transform the problem space into one in which everything is four times as big as normal, apply the operation, and then convert back to normal size.

Those of you who've played with a Rubik's Cube are well familiar this technique: If you have a move that, say, flips two adjacent edges, but you want to flip two edges that aren't adjacent, you can still accomplish this by manipulating the cube until the two edges are adjacent, perform the flip move, then undo the steps you performed to get the edges adjacent in the first place. (This is known as "conjugation" in group theory and is a very handy technique.)

We're just doing the same thing with this truncation operation: If we could shrink the truncator a factor of four, it would truncate by quarters. But we don't know how to shrink the truncator, so we do it from the other direction: Stretch the number line, apply the truncator, then shrink it back.

```
0                   1
|                   |
|   1/4  2/4  3/4   |   5/4  6/4
|    |    |    |    |    |    |
+----+----+----+----+----+----+
 \___/\___/\___/\___/\___/\___/
   0    1    2    3    4    5
   0   1/4  2/4  3/4  4/4  5/4
```

The top line shows the number line stretched by a factor of four. The truncator is still unchanged. And below it, we shrink the results by a factor of four, resulting in our desired rounding down to the nearest quarter.

Taking the above diagram and converting it back to a formula:

> round_down_to_quarter(v) = trunc(v * 4.0) / 4.0

This was probably old hat for most of you, but I think it's worthwhile seeing how the problem can be viewed geometrically. In particular, if you have a reversible operation "f", then the composition "$f^{-1} \circ g \circ f$" has the effect of "reinterpreting g through f-colored glasses". Here, the operation "f" was "multiple by four" and "g" was "truncate to nearest integer". Putting them together allowed us to take the truncation operator "g" and make it truncate according to a different set of rules.

Raymond Chen

Follow