

Abstraction, and Kaldewaij's “mod” exercises (revised)

When we are faced with a problem to solve, it can be useful and sometimes even crucial to generalize the problem, to abstract away from its details. In doing so, we give ourselves the opportunity to solve a whole class of problems in one go, including the original problem. But also, by suppressing distracting or irrelevant details, the problem may become easier to solve, despite being more general.

In this note, I will use abstraction in various ways to aid in solving some problems about the binary **mod** operator on integers, as given in Anne Kaldewaij's book *Programming*. Some additional results will be included, for completeness's sake.

* *

 *

In what follows, all variables are of type integer, and in particular, $m \neq 0$, unless otherwise mentioned.

Kaldewaij defines the binary infix operator **mod** together with **div**, by:

$$(0) \quad a \mathbf{div} m = q \quad \wedge \quad a \mathbf{mod} m = r$$

$$\equiv$$

$$a = m * q + r \quad \wedge \quad 0 \leq r < |m| \quad ,$$

where, recall, we have $m \neq 0$. So that **mod** and **div** are properly defined by (0), Kaldewaij invites us to note that equation:

$$(1) \quad (q, r) : a = m * q + r \quad \wedge \quad 0 \leq r < |m|$$

has, for $m \neq 0$, precisely one solution. A simple program will settle the existence of a solution; as for the uniqueness, suppose (q_0, r_0) and (q_1, r_1) are solutions to (1), so that we have:

$$(2) \quad a = m * q_0 + r_0$$

$$(3) \quad 0 \leq r_0 < |m|$$

$$(4) \quad a = m * q_1 + r_1$$

$$(5) \quad 0 \leq r_1 < |m| \quad .$$

We will show $r_0 = r_1$ and $q_0 = q_1$. The shapes of (3) and (5) suggest an appeal to the chunking lemma:

$$|x - y| < |m| \quad \Rightarrow \quad (x \overset{m}{\approx} y \quad \equiv \quad x = y) \quad ,$$

where $\overset{m}{\approx}$ is congruence modulo m . (See Appendix.) Indeed, the reader can verify that (3) and (5) imply $|r_0 - r_1| < |m|$, which has the form of the antecedent to the chunking lemma. Thus we have:

JAW45a-1

$$\begin{aligned}
& r_0 = r_1 \\
\equiv & \quad \{ \text{chunking lemma, using (3) and (5)} \} \\
& r_0 \stackrel{m}{\approx} r_1 \\
\equiv & \quad \{ (2) \text{ and } (4) \} \\
& a - m * q_0 \stackrel{m}{\approx} a - m * q_1 \\
\Leftarrow & \quad \{ \text{property of } \stackrel{m}{\approx} \} \\
& a \stackrel{m}{\approx} a \quad \wedge \quad m * q_0 \stackrel{m}{\approx} m * q_1 \\
\equiv & \quad \{ \text{properties of } \stackrel{m}{\approx} \} \\
& \mathbf{true} \quad ,
\end{aligned}$$

which is half our proof obligation. The reader can check that with $r_0 = r_1$, equations (3) and (5) imply $q_0 = q_1$, thus settling the uniqueness of the solution to (1).

* *

Because in this note we will not be concerned with **div**, we use predicate calculus to help us “eliminate” it from definition (0):

$$\begin{aligned}
& a \mathbf{mod} m = r \\
\equiv & \quad \{ \text{one-point rule} \} \\
& \langle \exists q : a \mathbf{div} m = q : a \mathbf{mod} m = r \rangle \\
\equiv & \quad \{ \text{shunting} \} \\
& \langle \exists q :: a \mathbf{div} m = q \wedge a \mathbf{mod} m = r \rangle \\
\equiv & \quad \{ (0) \} \\
& \langle \exists q :: a = m * q + r \quad \wedge \quad 0 \leq r < |m| \rangle \\
\equiv & \quad \{ \wedge \text{ over } \exists \} \\
& \langle \exists q :: a = m * q + r \rangle \quad \wedge \quad 0 \leq r < |m| \\
\equiv & \quad \{ \text{property of } \stackrel{m}{\approx} \} \\
& a \stackrel{m}{\approx} r \quad \wedge \quad 0 \leq r < |m| \quad ;
\end{aligned}$$

summarizing:

$$\begin{aligned}
(6) \quad & a \mathbf{mod} m = r \\
& \equiv \\
& a \stackrel{m}{\approx} r \quad \wedge \quad 0 \leq r < |m| \quad .
\end{aligned}$$

Instantiating (6) with $r := a \bmod m$, we conclude:

$$(7) \quad a \stackrel{m}{\approx} a \bmod m \quad \text{and}$$

$$(8) \quad 0 \leq a \bmod m < |m| \quad ,$$

which we might call the “defining properties” of **mod**.

So much for the definition of **mod** and its defining properties.

* *
 * *

We turn now to a “warm-up” exercise, one that will not require any abstraction, but will allow us to practice with our definition of **mod**. The theorem to be proved is:

$$(9) \quad a \bmod m = a \bmod (-m) \quad .$$

A calculational proof of (9) is straightforward:

$$\begin{aligned} & a \bmod (-m) = r \\ \equiv & \{ (6) \} \\ & a \stackrel{-m}{\approx} r \wedge 0 \leq r < |-m| \\ \equiv & \{ \text{property of } \approx ; \text{ property of absolute value} \} \\ & a \stackrel{m}{\approx} r \wedge 0 \leq r < |m| \\ \equiv & \{ (6) \} \\ & a \bmod m = r \quad , \end{aligned}$$

from which we can conclude (9) on account of “indirect equality”.

* *
 * *

Well, that was simple enough, but already the next exercise is much more challenging with its embedded **mod**’s:

$$(10) \quad (a \bmod m) \bmod m = a \bmod m \quad .$$

With **mod**’s all over the place, application of (6) is uninviting, because we have many choices for where to apply it, and we may have to apply it multiple times.

In such an instance, abstraction becomes invaluable. Notice that (10) is of the form:

$$(11) \quad r \bmod m = r \quad ,$$

which is much more inviting to manipulate. We calculate with (11) to discover when it holds:

$$\begin{aligned}
& r \mathbf{mod} m = r \\
\equiv & \{ (6) \} \\
& r \stackrel{m}{\approx} r \quad \wedge \quad 0 \leq r < |m| \\
\equiv & \{ \text{left conjunct: property of } \approx \} \\
& 0 \leq r < |m| \quad .
\end{aligned}$$

So (11) holds precisely when $0 \leq r < |m|$. The relevant instantiation for problem (10) is $r := a \mathbf{mod} m$, so the question now is whether we have:

$$0 \leq a \mathbf{mod} m < |m| \quad ,$$

and by defining property (8) of \mathbf{mod} , the answer is a resounding yes.

Problem (10) was a mess, but solving it via the more general (11) couldn't have been easier. Three cheers for abstraction.

* *

 *

Finally, we turn to the last problem from Kaldewaij:

$$(12) \quad (a \mathbf{mod} m + b \mathbf{mod} m) \mathbf{mod} m = (a + b) \mathbf{mod} m \quad .$$

If (10) was a mess, (12) is a nightmare. And unfortunately, abstraction (11) is of no help, because (12) is not of that form. That is all right: we just make a new abstraction! Formula (12) is of the shape:

$$(13) \quad c \mathbf{mod} m = d \mathbf{mod} m \quad ,$$

so let us investigate when (13) holds:

$$\begin{aligned}
& c \mathbf{mod} m = d \mathbf{mod} m \\
\equiv & \{ (6) \text{ with } a := c \text{ and } r := d \mathbf{mod} m \} \\
& c \stackrel{m}{\approx} d \mathbf{mod} m \quad \wedge \quad 0 \leq d \mathbf{mod} m < |m| \\
\equiv & \{ \text{right conjunct: (8) with } a := d \} \\
& c \stackrel{m}{\approx} d \mathbf{mod} m \\
\equiv & \{ \text{property of } \stackrel{m}{\approx}, \text{ using } d \stackrel{m}{\approx} d \mathbf{mod} m \text{ from (7)} \} \\
& c \stackrel{m}{\approx} d \quad ,
\end{aligned}$$

or in other words:

$$(14) \quad c \stackrel{m}{\approx} d \equiv (c \bmod m = d \bmod m) \quad .$$

Terrific! So making the appropriate instantiations to (14), we see that (12) holds precisely when:

$$\begin{aligned} & a \bmod m + b \bmod m \stackrel{m}{\approx} a + b \\ \Leftarrow & \quad \{ \text{property of } \stackrel{m}{\approx} \} \\ & a \bmod m \stackrel{m}{\approx} a \quad \wedge \quad b \bmod m \stackrel{m}{\approx} b \\ \equiv & \quad \{ (7) \} \\ & \mathbf{true} \quad , \end{aligned}$$

which settles (12) .

Generalization (13) is quite general, and hence quite powerful. Note that our earlier problem (11) is also of form (13), with $c := a \bmod m$ and $d := a$. By (14), we just need to check whether $a \bmod m \stackrel{m}{\approx} a$ holds, and it does, thanks to (7) .

We can also use (14) to prove a generalization of another theorem given in *Programming* , namely:

$$(a + k * m) \bmod m = a \bmod m$$

by observing:

$$\begin{aligned} & a + k * m \stackrel{m}{\approx} a \\ \equiv & \quad \{ \text{property of } \stackrel{m}{\approx} \} \\ & k * m \stackrel{m}{\approx} 0 \\ \equiv & \quad \{ \text{property of } \stackrel{m}{\approx} \} \\ & \mathbf{true} \quad . \end{aligned}$$

Et cetera!

* * *

To reiterate in conclusion: three cheers for abstraction!

Appendix

Integer relation \sqsubseteq (“divides”) can be defined by:

$$m \sqsubseteq x \equiv \langle \exists q :: x = m * q \rangle \quad .$$

From this definition it follows that \sqsubseteq is a partial order, as the reader can verify. Also we have the following properties of \sqsubseteq , which are easily verified:

$$\begin{aligned} m &\sqsubseteq 0 \\ m &\sqsubseteq m * x \\ m \sqsubseteq x &\equiv -m \sqsubseteq x \\ m \sqsubseteq x &\equiv m \sqsubseteq -x \\ m \sqsubseteq x &\Rightarrow (m \sqsubseteq y \equiv m \sqsubseteq (x + y)) \quad . \end{aligned}$$

In terms of \sqsubseteq , $\overset{m}{\approx}$ can be defined by:

$$x \overset{m}{\approx} y \equiv m \sqsubseteq x - y \quad .$$

Using the above properties of \sqsubseteq , we can show that $\overset{m}{\approx}$ is reflexive, symmetric, and transitive (that is, an equivalence relation), and satisfies the following properties:

$$\begin{aligned} m * x &\overset{m}{\approx} 0 \\ x \overset{m}{\approx} y &\equiv x \overset{-m}{\approx} y \\ x \overset{m}{\approx} y &\Rightarrow x + a \overset{m}{\approx} y + a \\ x \overset{m}{\approx} y &\Rightarrow (x \overset{m}{\approx} a \equiv y \overset{m}{\approx} a) \quad . \end{aligned}$$

The original version of this note carried the entire argument in terms of \sqsubseteq , rather than $\overset{m}{\approx}$, which made many of the arguments clumsier than they needed to be. (I can't be blamed, as I hadn't yet met Dan Grundy.)

(**End** of Appendix.)

Written: Zurich (Starbucks Cafe & Train to Salzburg), 21 December 2005

Typeset: Westward o'er the Atlantic, 4 January 2006

Revised: Santa Cruz, 22 May 2008