Math Bugs

Mike Naylor
Matematikkbølgen – Center for Mathematics Creativity
7125 Vanvikan Norway
Email: abacaba@gmail.com

Abstract

Math Bugs are cool little creatures that are made from sets of curves that represent factorial multiplication within modulo bases. The bugs have interesting properties and connections, and surprising number patterns. By representing modulo multiplication visually, the geometric patterns which result can be used for artwork which can draw attention to aspects of matemathematics which may otherwise be hidden.

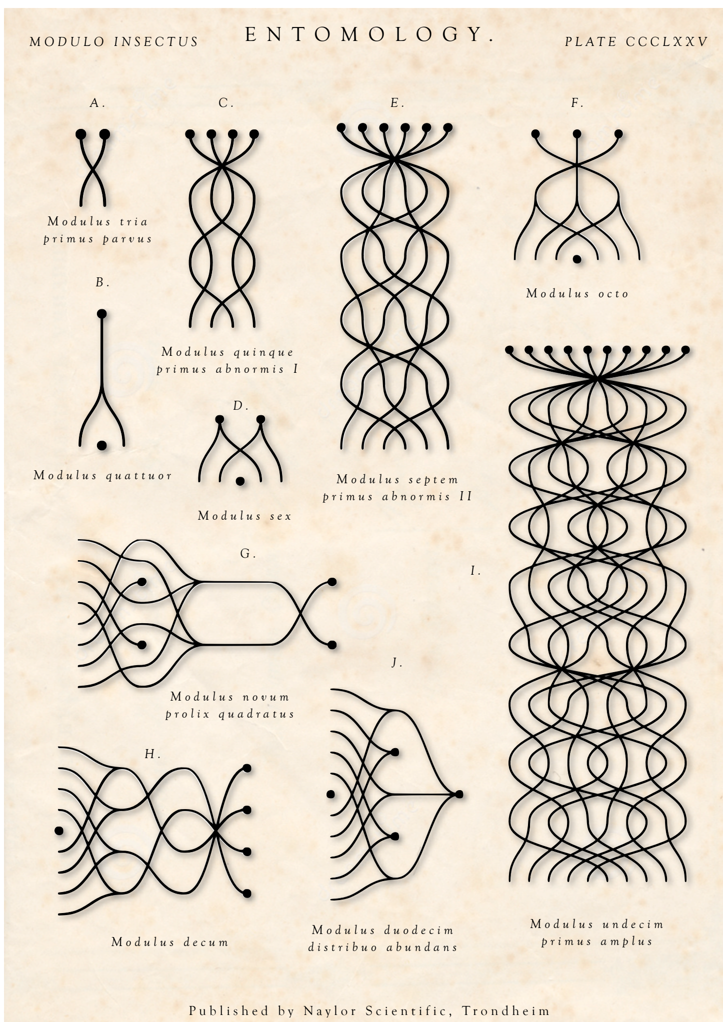

Figure 1: Math Bugs entomology

Introduction

Figure 1 shows a page from a fictional entomology text. These unusual bugs are found out in the wilds of moduluar multiplication and we will explore their structure and behaviors here.

In the 2013 Bridges Conference, we wrote the digits 0-9 around a circle and then connected the numbers with arcs to show the results of various operations such as ‘multiply by 3’, to create surprising patterns called “Math Runes” (see Figure 2) [1]. In 2014, the idea was extended to different bases as described in “Java Runes” [2], with an accompanying web app [3] that animates transitions between operations.

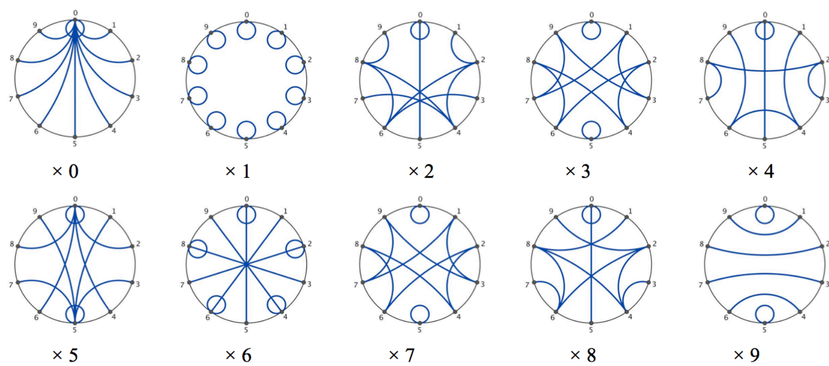

Figure 2: Math runes mod 10

Now we’ll make a tile, a linear version that allows us to string operations together in a way that is difficult to do with circular runes. Our first tile has 10 digits down both sides, and we connect one side of the tile to the other with arc to show a function. For example, Figure 3 shows a “+2” tile. Each number on the left connects to the number on the right which is 2 greater. We’re operating in mod 10, so the numbers “wrap-around” and 8+2 = 10 wraps past the 9 to the 0, and 9+2 = 1 wraps up to the 1. Mod systems have surprising properties, and it is these properties of mod systems that give our forms interesting shapes.

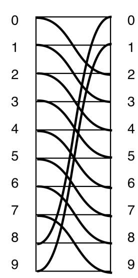

Figure 3: The +2 tile

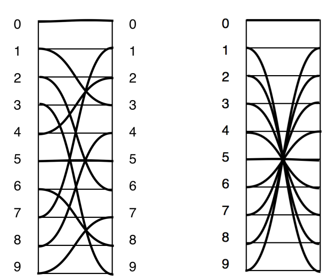

Figure 4: The ×3 and the ×9 tiles

Figure 4 shows the ×3 and the ×9 tiles. They’re strange shapes with symmetry like the math runes. We can string multiple copies together of each to make weaving patterns seen below (Figures 5-8).

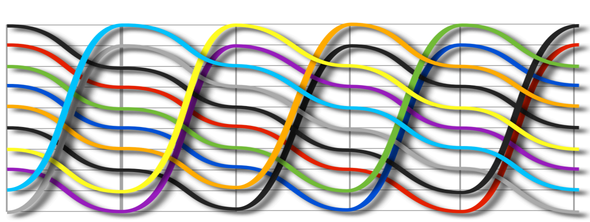

Figure 5: Weaving with +2

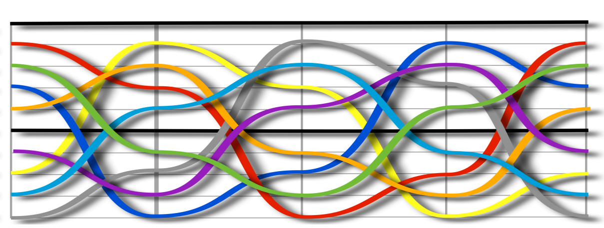

Figure 6: Weaving with ×3

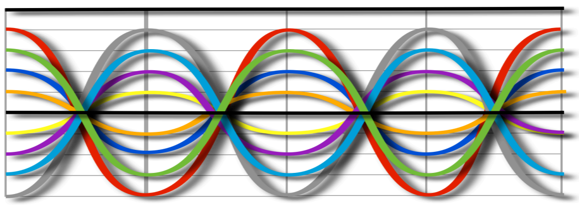

Figure 7: Weaving with ×9

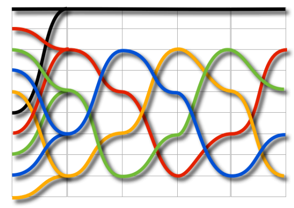

Figure 8: ×2 tiles – the strands combine

In the ×3 weaving (Figure 6), each line returns to its starting position after 4 steps. What does that represent computationally? With the operation ×9 (Figure 7) the strands switch place pairwise. What is happening here? With ×2 tiles, the strands combine after the first operation and there are only 5 strands to continue the pattern. What happens with other multipliers?

Creating Math Bugs with Factorials

We can create compositions of functions to see what happens with the strands. One scheme that makes interesting shapes is the class of figures created by the factorial function. Figure 9 shows the set of digits 0-9 on a mod 10 system where we first multiply by 1, then 2, then 3, and so on. Only the first two tiles have 10 inputs, after multiplying by 2, the strands combine in pairs and there are only 5 inputs and outputs until the fifth step, ×5, where all of the strands go to 0.

This design is *almost* pretty, but the strands all end up at 0, making the shape look lopsided. Also the first step is boring: ×1... do we really need that? Let's clean this up. We'll get rid of the 0 row and instead put a big black dot at the end of a strand that goes to 0. We'll also get rid of the ×1 column.

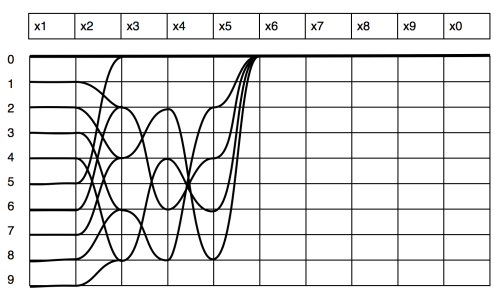

Figure 9: Factorial process mod 10

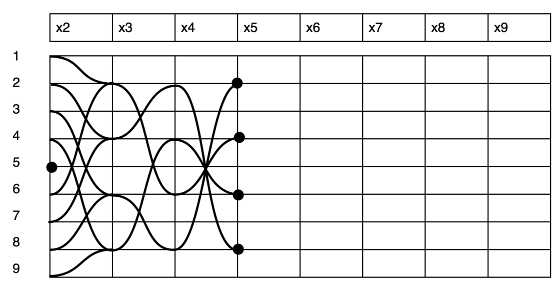

Figure 10: Factorial process mod 10 streamlined

Figure 10 shows the design redrawn, nice and symmetric. It looks like a bug. We'll call it "bug 10." Bug 10 only makes it halfway across the board. After we multiply by 5, all of the numbers will have been multiplied by 2 and 5 and thus be a multiple of 10 (which is 0 mod 10).

Each mod number has its own bug. Figure 11 shows the bug for mod 5. All of the strands in bug 5 make it all the way across, so bug 5 is maximal length. Bug 6 (shown in Figure 12) is very short. Once we multiply all the numbers by 2 and 3 every number is a multiple of 6 which is 0 mod 6. The strands end.

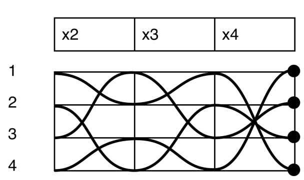

Figure 11: Bug 5

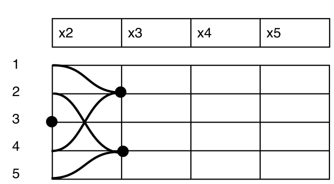

Figure 12: Bug 6

Figure 13 shows the bugs for mod 2 through mod 25. These images were made with Processing.

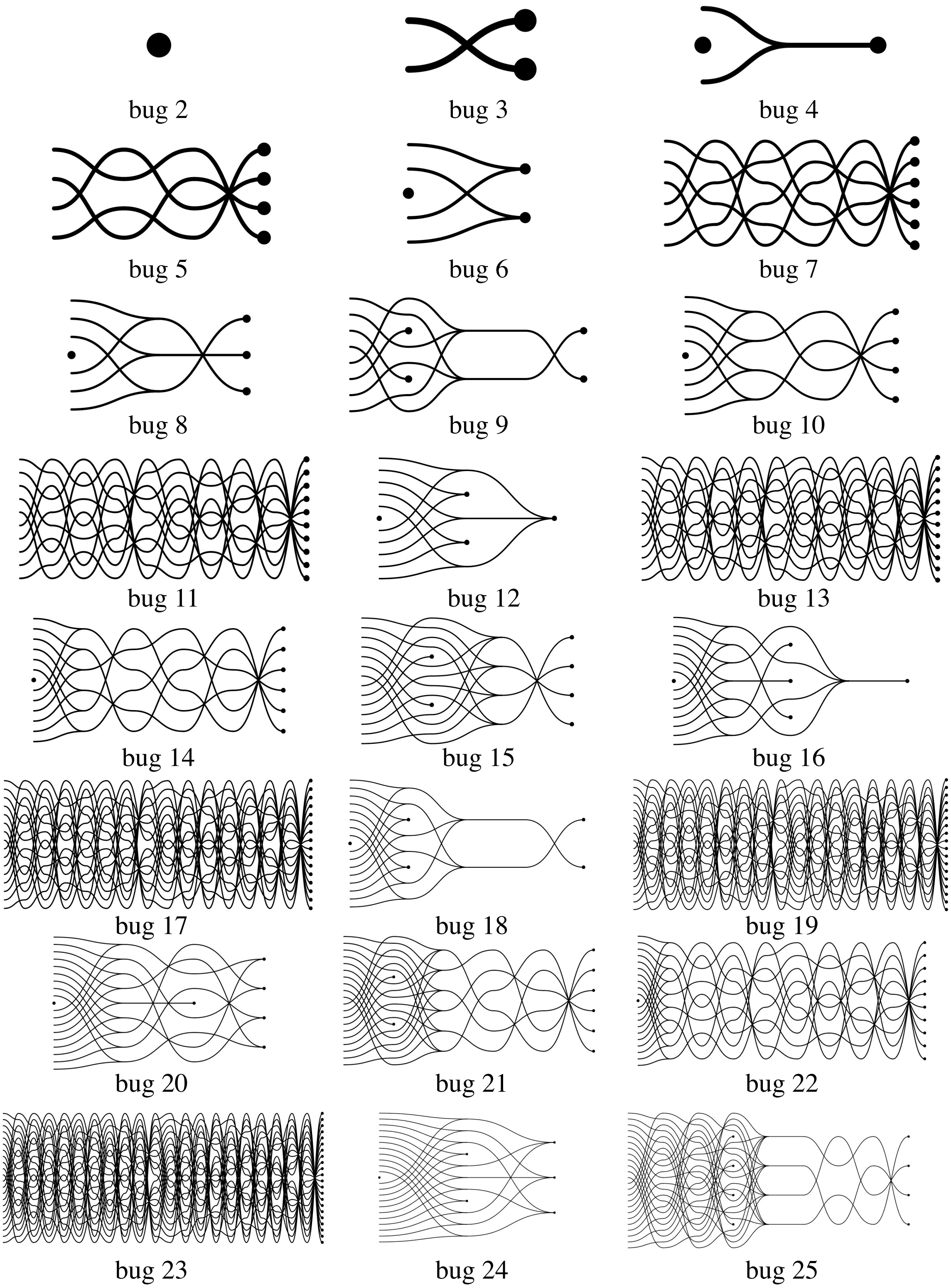

Figure 13: Bugs 2-25

Properties and behaviors of Math Bugs

How can the bugs be sorted? What visual properties do the bugs have and what mathematical properties do they represent? Here are a few ideas:

• Most apparent are the bugs that are very long, like millipedes. Prime number bugs are of maximum length, displaying the entire operation *p!*. We call these “prime bugs.”

• What properties are shared by even/odd number bugs? Square number bugs? Cubes and other powers?

• The bugs have antennas. What does the number of antennas have to do with the bug number? What about antenna placement? Interior antennas?

• Many bugs have antennas on the right – strands that swap positions in pairs from top to bottom to form a twist. Prime bugs have a full head of twisted antennas, but a few bugs, 6 and 12 for example, do not have these twists at all.

• Prime bugs have strands that return to their starting position just before twisting into antennas. Take a look at bug 5 for example. Just before the final twist, 1 has returned to 1, 2 has returned to 2, and so on. This property holds for all primes, demonstrating that $(p-2)! \equiv 1 \bmod n$. This can be derived from Wilson’s theorem (see [4]) which shows that $(p-1)! \equiv -1 \bmod n$. It is this property of modulo systems which is responsible for the twist – input k connecting to output $n-k$.

• What determines the number of antennae on the bugs? Prime bugs have the maximum number of antennae. Some bugs have just one antenna, for example bugs 4, 12, 16, 32, 48, 80 and 96. Why? On some of these, like on bugs 12 and 80, the single antenna is short. On others, like 4 and 16, it is long. Why?

An App to Explore the Bugs

I began drawing these bugs by hand, and then with Apple Pages, before writing code in Processing. An app is available to explore the bugs at mike-naylor.com. Figures 14-16 show several larger bugs created with this app.

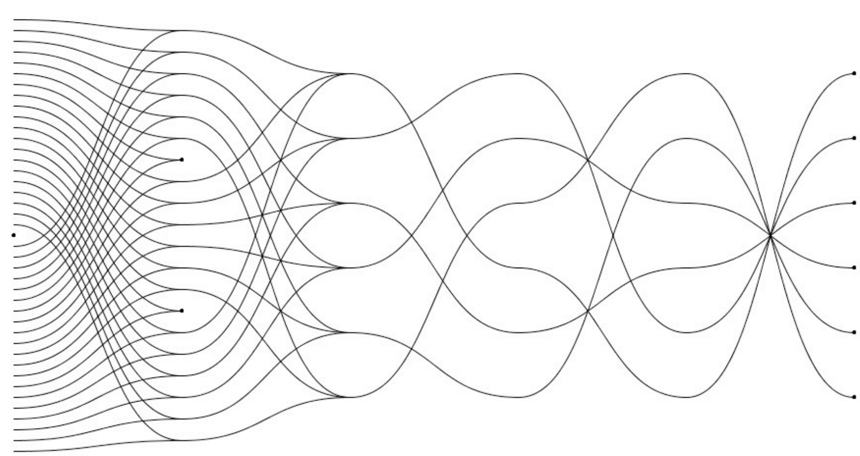

Figure 14: Bug 42

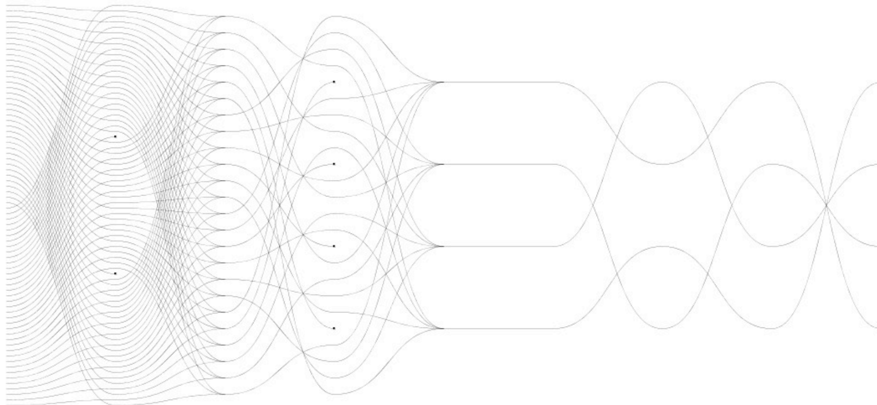

Figure 15: Bug 75

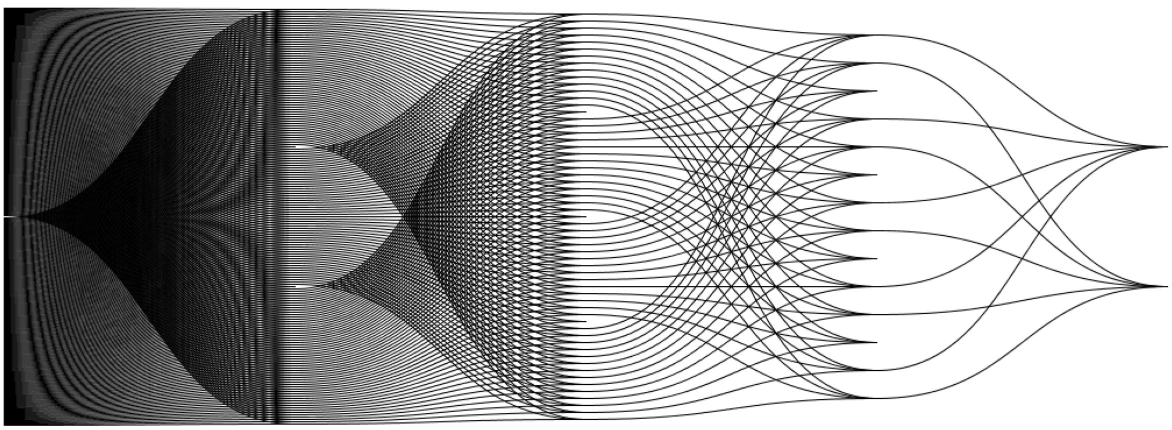

Figure 16: Bug 360

Math bugs and Math art

Math bugs are a visual representation of factorial multiplication. The variety of shapes is quite surprising and hint at properties of number and operations. By representing mathematical ideas in different ways, we can gain insight into properties of numbers and operations that would be difficult to see in other ways.

To me, this is the essence of mathematical art – finding ways to represent mathematical ideas in aesthetically pleasing forms that invite exploration and lead to a deeper understanding of mathematical principles.

References

[1] Naylor, M. "Math Runes." 2013 Bridges Conference Proceedings, pp. 191-198, July 2013.

[2] Naylor, M. "Java Runes." 2014 Bridges Conference Proceedings, pp. 191-198, July 2014.

[3] Naylor, M. "Math Runes web app." http://mike-naylor.com/runes, July 2014.

[4] Elston, F. G. "A Generalization of Wilson's Theorem." Mathematics Magazine, Vol. 30, No. 3, Jan. - Feb., 1957.