# Binary Modifiers for Waldorf Synthesizers

Achim Gratz

April 10, 2010

The binary modifiers can be used to achieve spectacular modulation effects, however most people find them difficult to use because they can't easily imagine what the effect of applying a modifier to some modulation source is. Simply probing around might yield interesting results, but for some things it would really be nice to know beforehand what's going on. This document provides some visualizations of what the result of binary modifier operations will look like and nothing else, for the operation of your synthesizer please consult the fine manual.

## Modulation Sources

There are two types of modulation sources: bipolar sources like LFO can be thought of having a range of $[-1, 1]$ and unipolar sources like the envelopes, which can be thought of having a range of $[0, 1]$. For the binary modifiers the resulting values are interpreted as 2's-complement binary numbers and then subjected to various binary operations.

## Binary Operations

The binary operations are applied to each bit of the sources and have the following truth table:

| Source | | Operation | | |
|---|---|---|---|---|
| A | B | AND | OR | XOR |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |

## Using a Constant as the second Operator

Since the modulation signals are really 24 bit numbers, they have to be scaled to match the range of the constant. You can think of this operation as the modulation source being multiplied by 64. That just uses six bits for the integer part, the remaining 18 bit will be used as a fractional number. The constant itself (if it is used by the modifier) is an integer and hence needs to be extended to cover the fractional bits (those to the right of the dyadic point). For AND, the fractional bits are always set to zero, which means that the resulting modulation signal "jumps" from one integer value to the next, making the modulation noticeably steppy, especially at high modulation amounts. For both OR and XOR, the last bit in the mask is extended to all fractional bits; if it is one, all fractional bits are also set to one and if it is zero, then all fractional bits will be zero, too. This means that for OR odd values will create steppy modulations, while for even values and for XOR this does not happen. It is hence possible to, for instance, invert a unipolar modulation source without introducing steppiness. Should you want to make these modulations steppy as well, you need to filter the modulation source through AND -1 first.

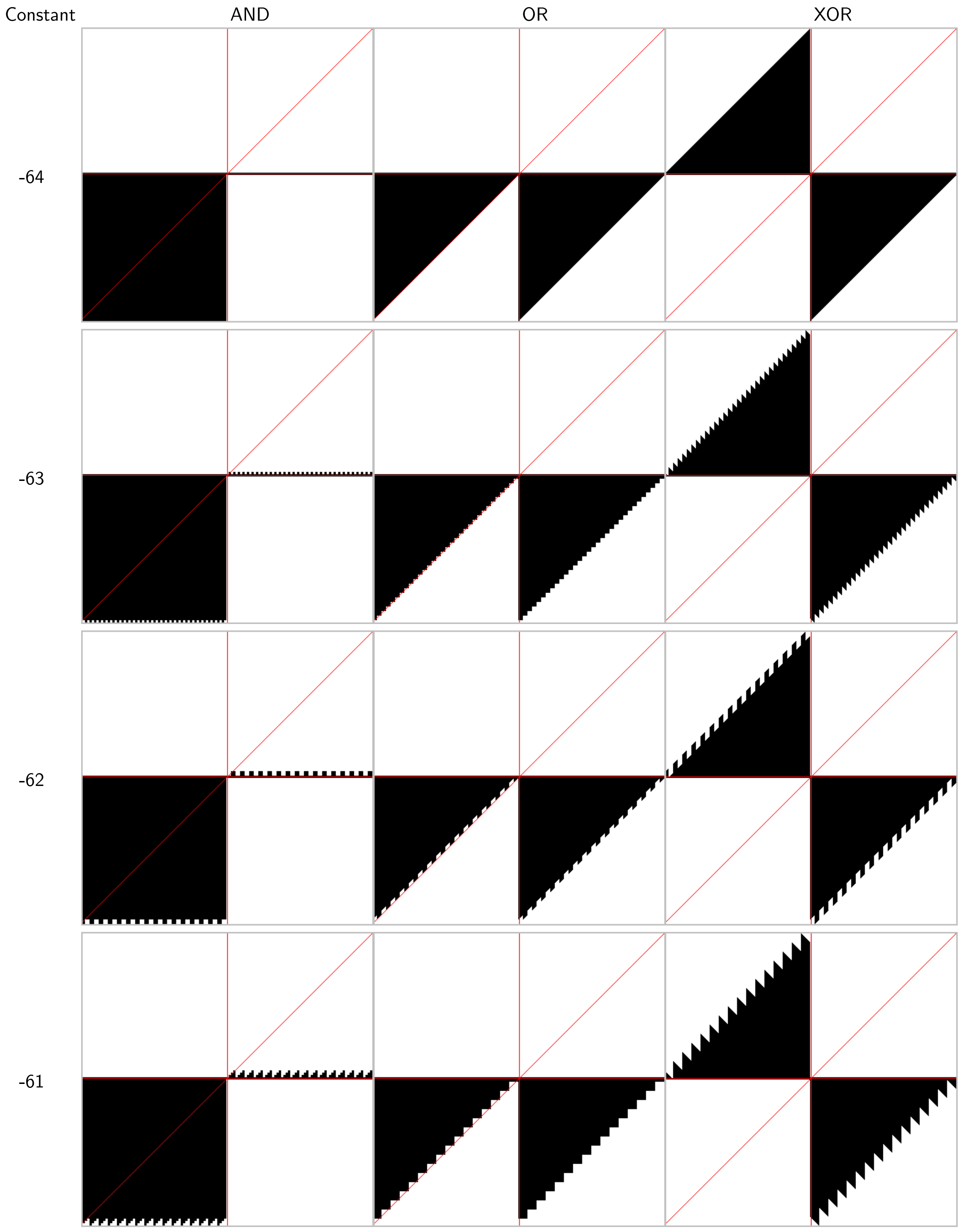
## Specials

A few particularly noteworthy cases:

- OR +0 and XOR +0 provide a copy of their source;
- AND -1 provides a steppy copy,
- AND -2, AND -4, AND -8, AND -16, AND -32, AND -64 increase the steppiness until only two modulation values are left;
- XOR -1 inverts a modulation source;
- AND 0 produces a flatline.

## Graphics

The graphics on the following 32 pages show the result of applying the binary modifiers to a linear ramp from $[-64, 63]$. The linear ramp is the scaled Source A, while the constant value shown on the left is the integer part of the scaled Source B or the Constant value. A red crosshair that shows zero both on the X and Y axis serves as a visual guide. The red diagonal is where the an exact copy of the source woud fall onto. Please note that a zero value in the result is shown as a single black pixel and not as a missing one. You may have to zoom in to see the finer details. If you zoom in really deep you will find that an integer step really consists of three black sub-pixels, so if you are not sure if the result will be steppy or not, simply check it out.


Constant
AND
OR
XOR
-64
-63
-62
-61

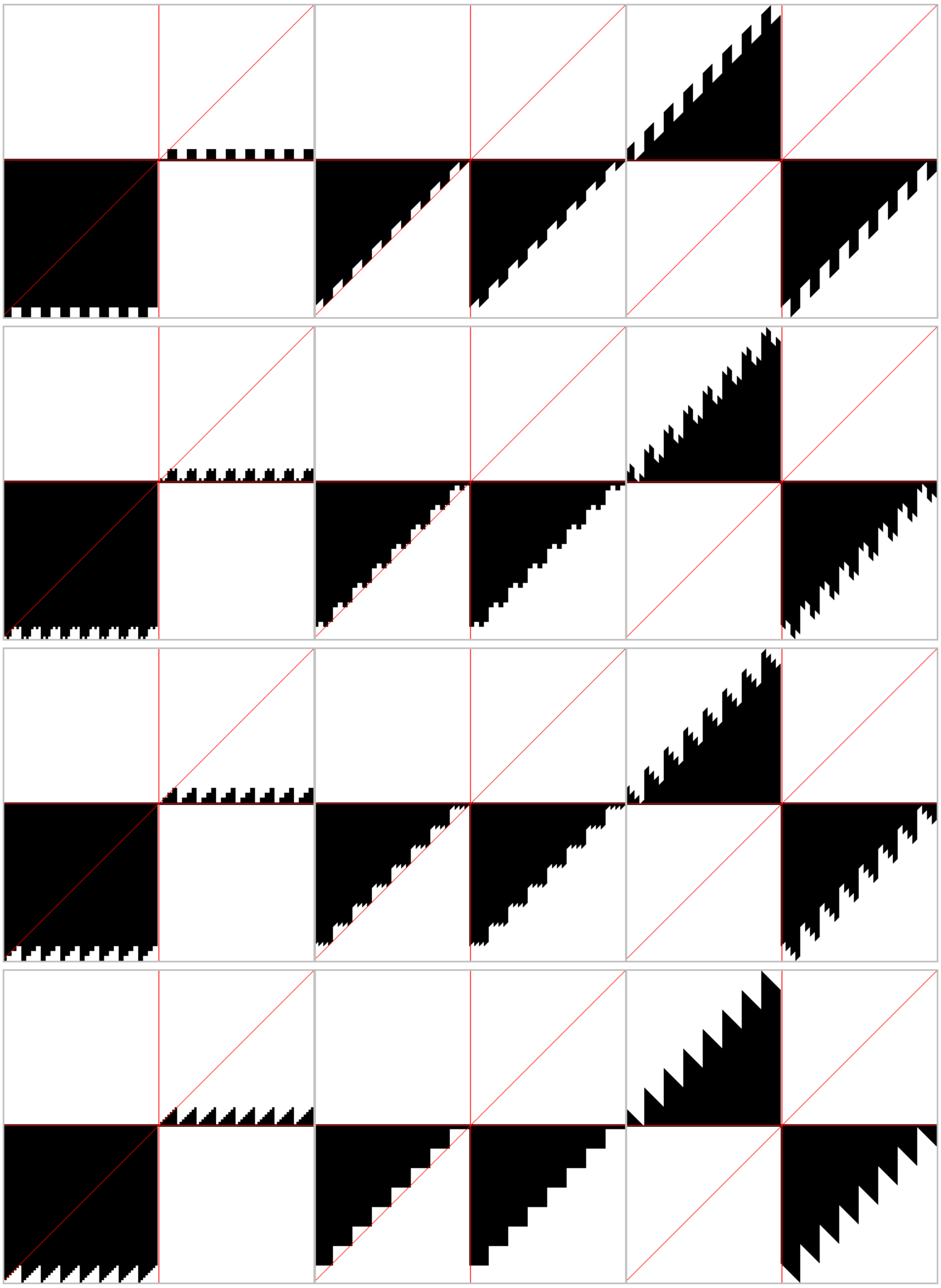
Constant AND OR XOR

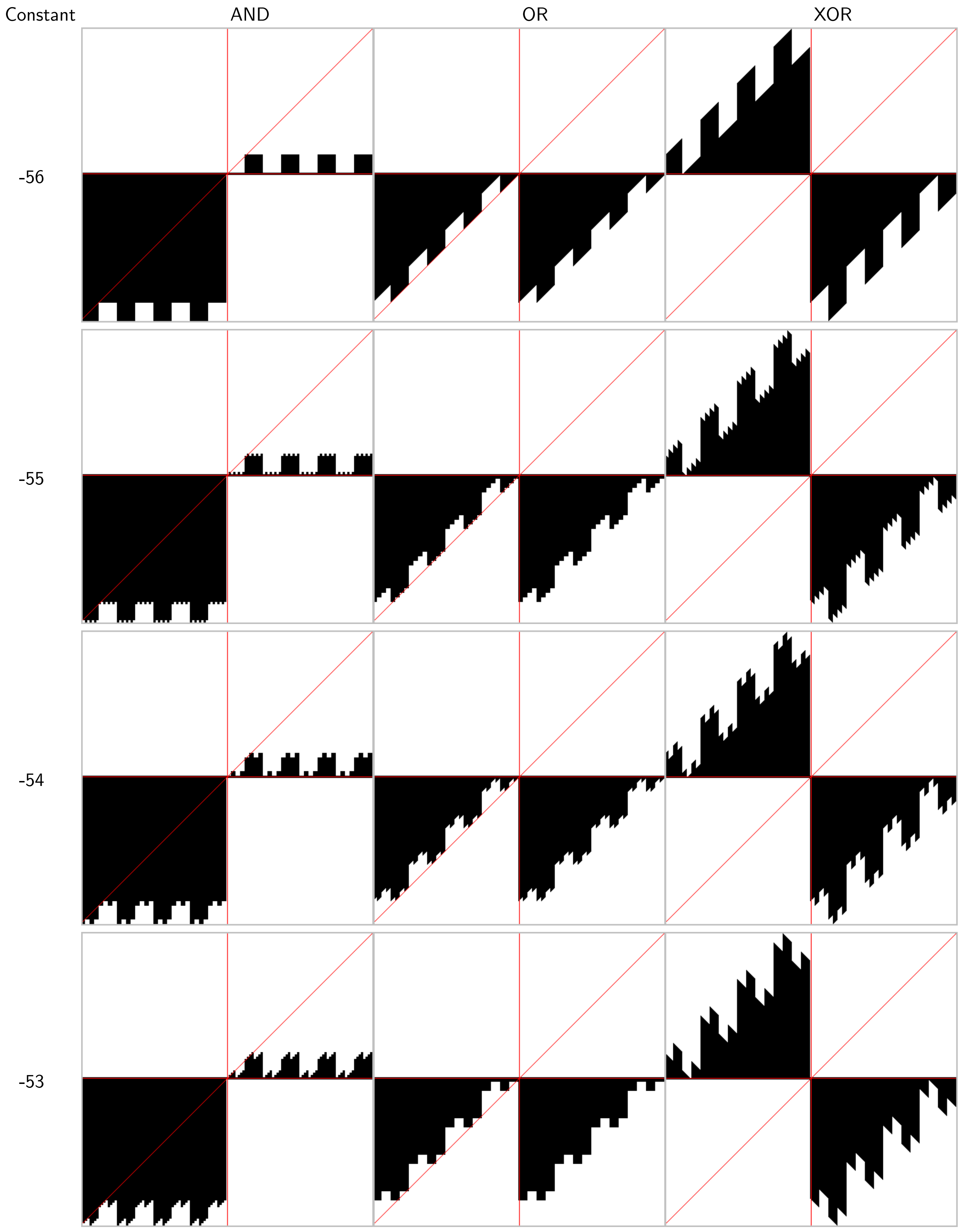
-60

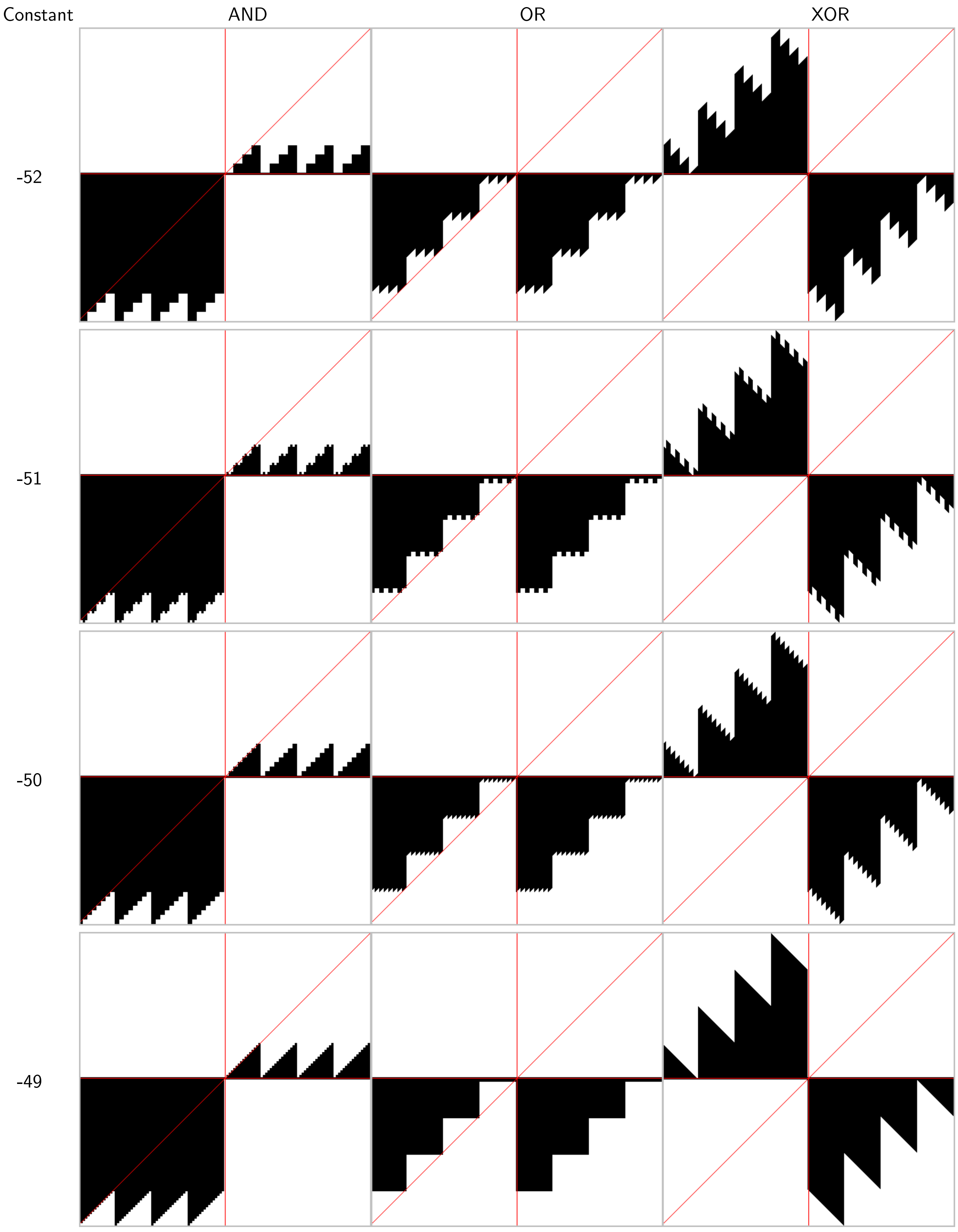
-59

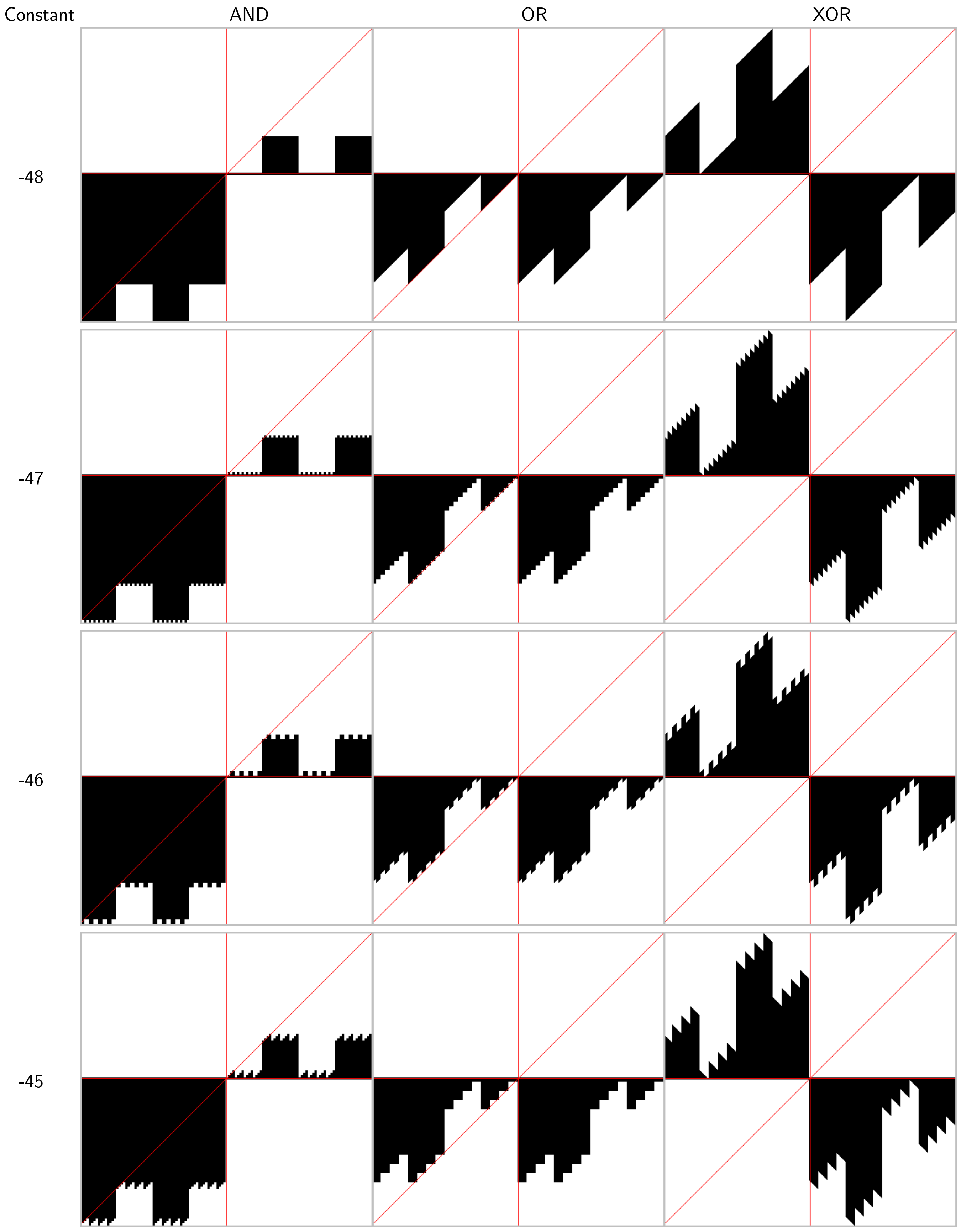
-58

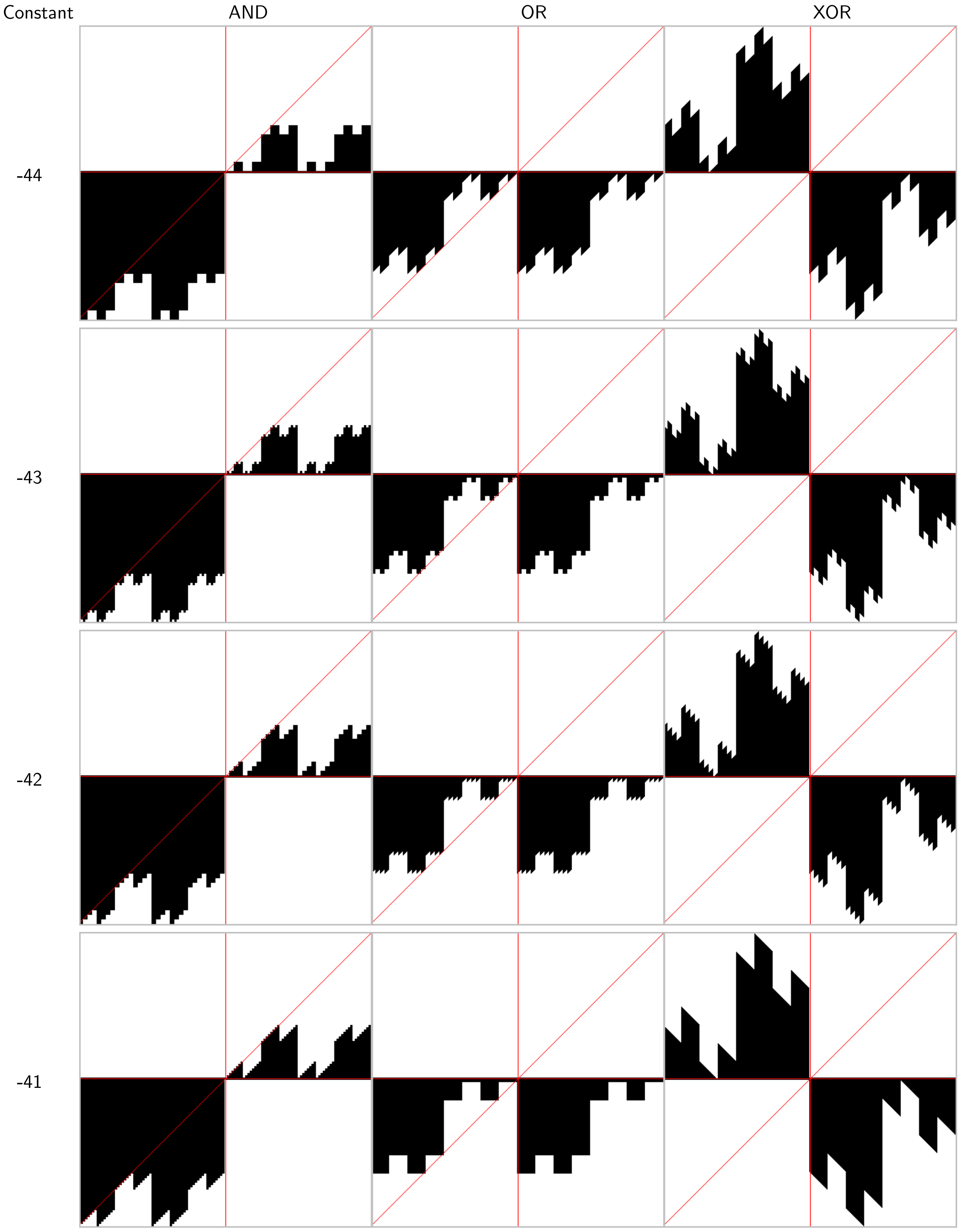
-57

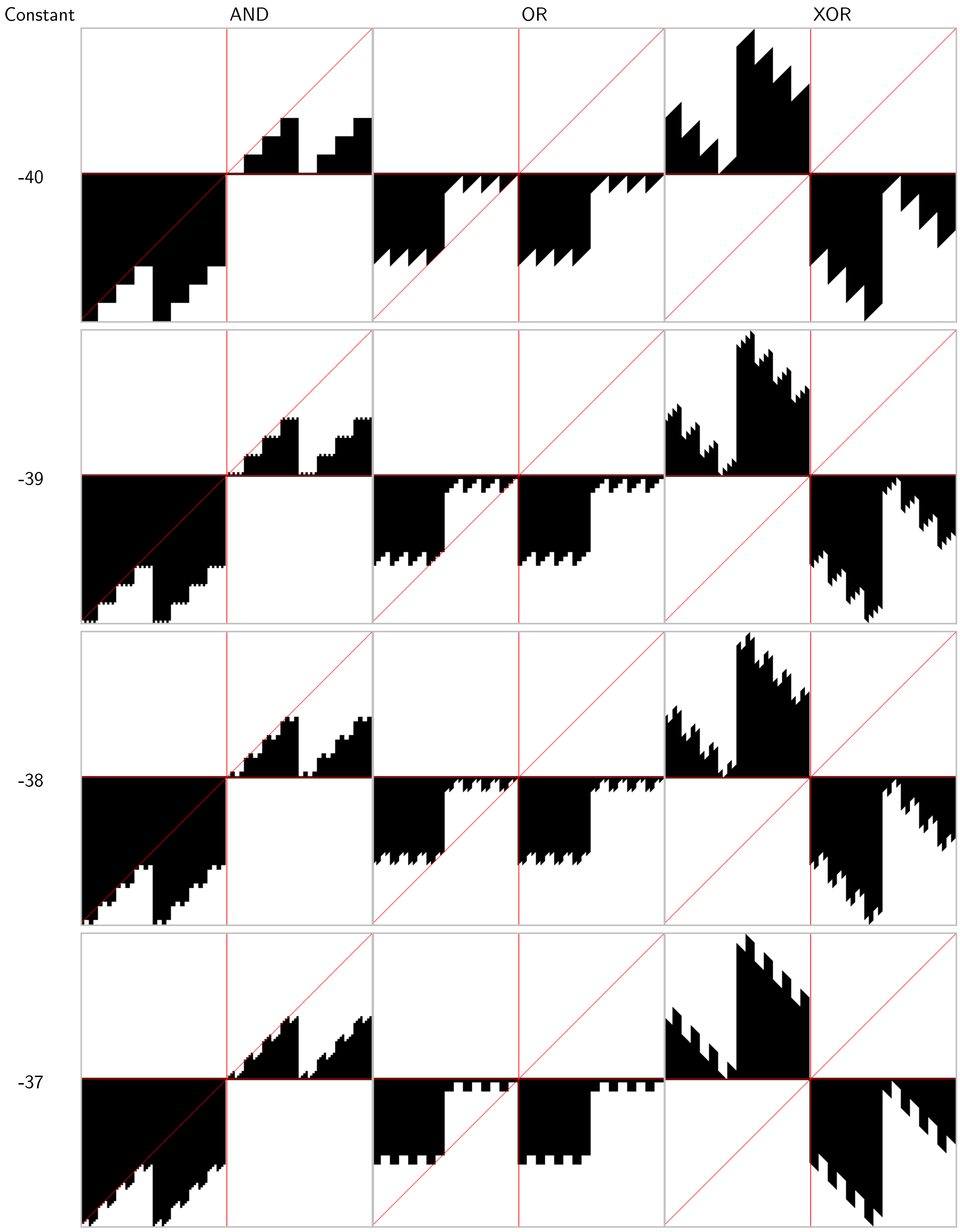


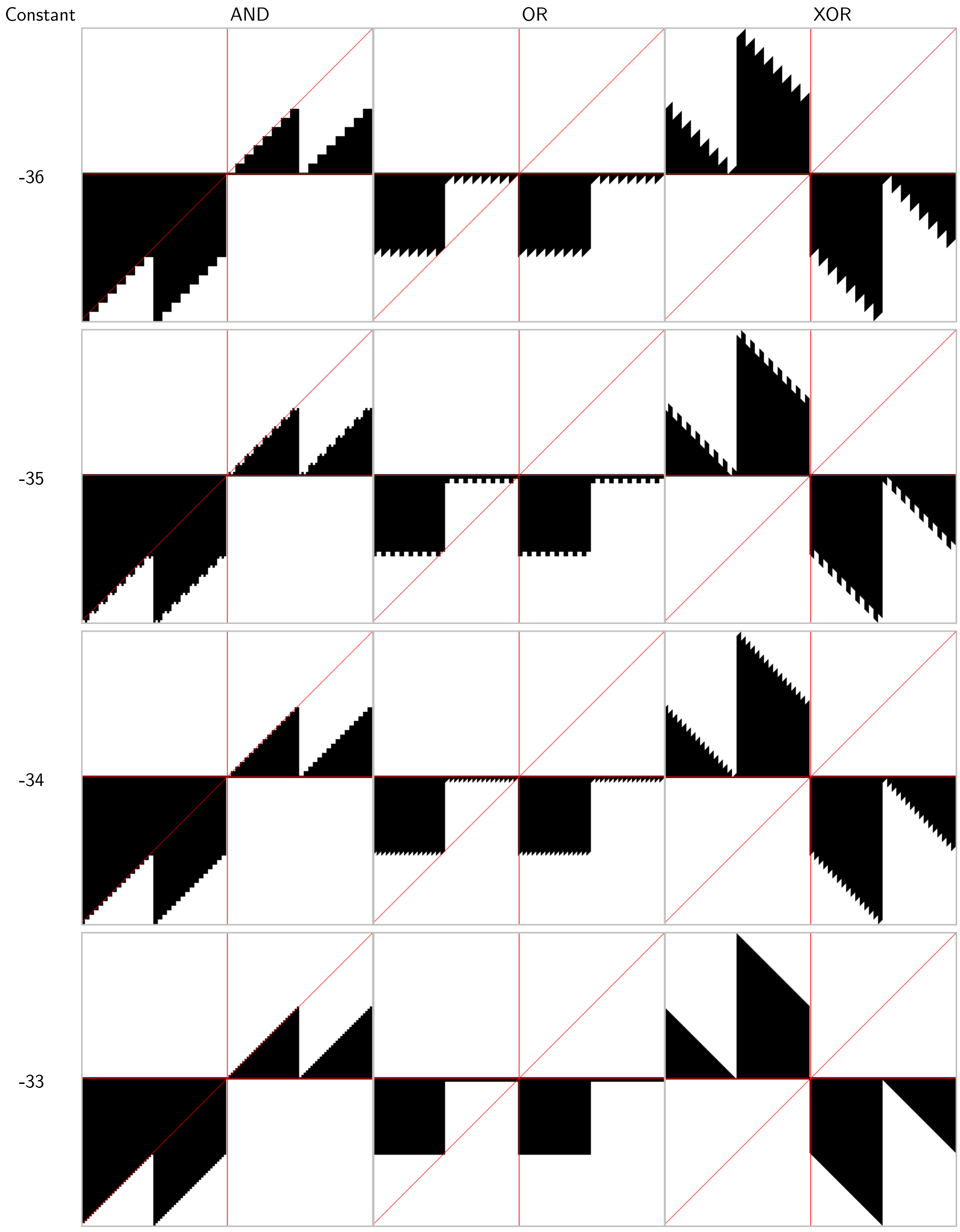

Constant
AND
OR
XOR
-56
-55
-54
-53

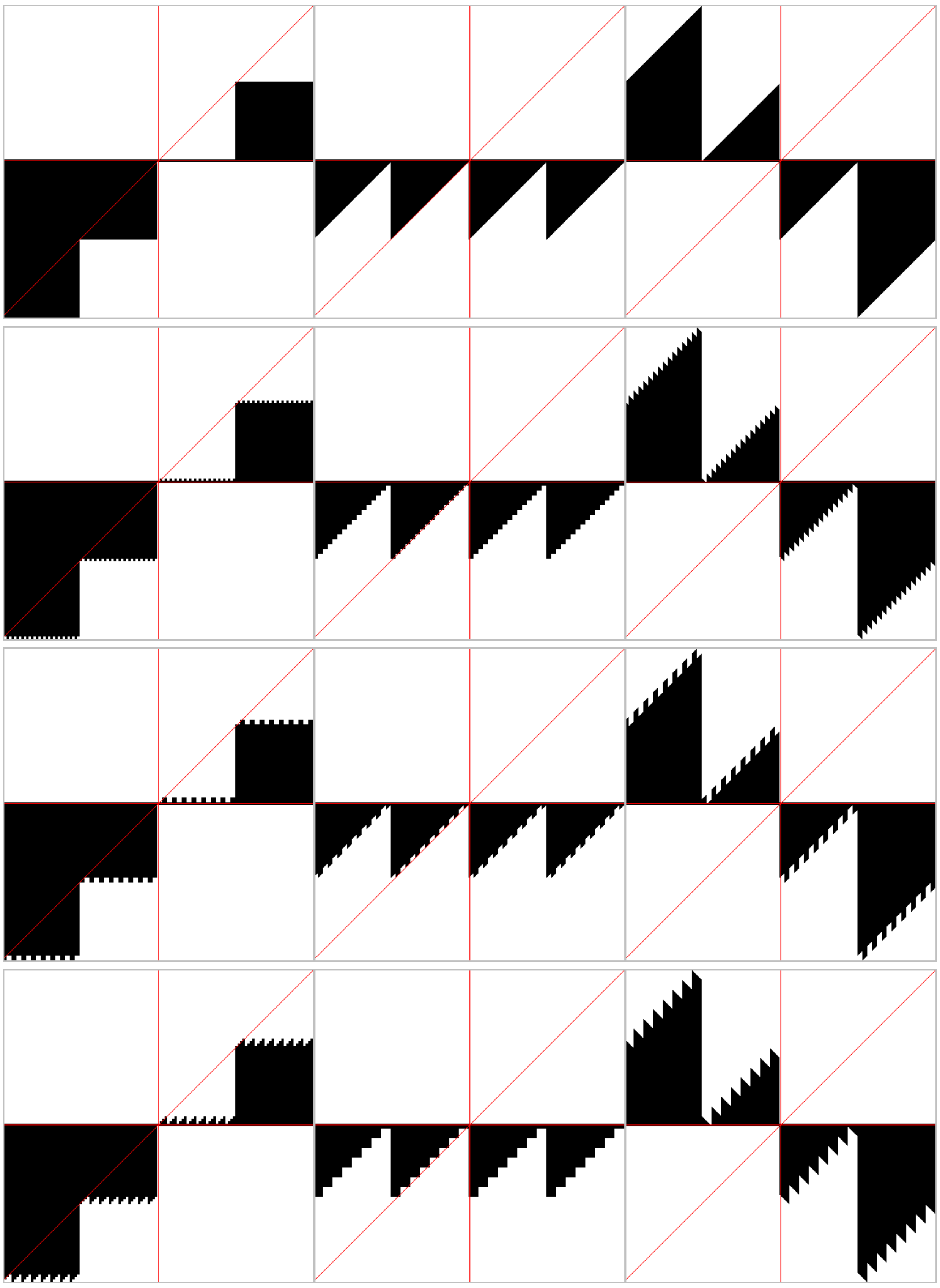

Constant
AND
OR
XOR
-52
-51
-50
-49

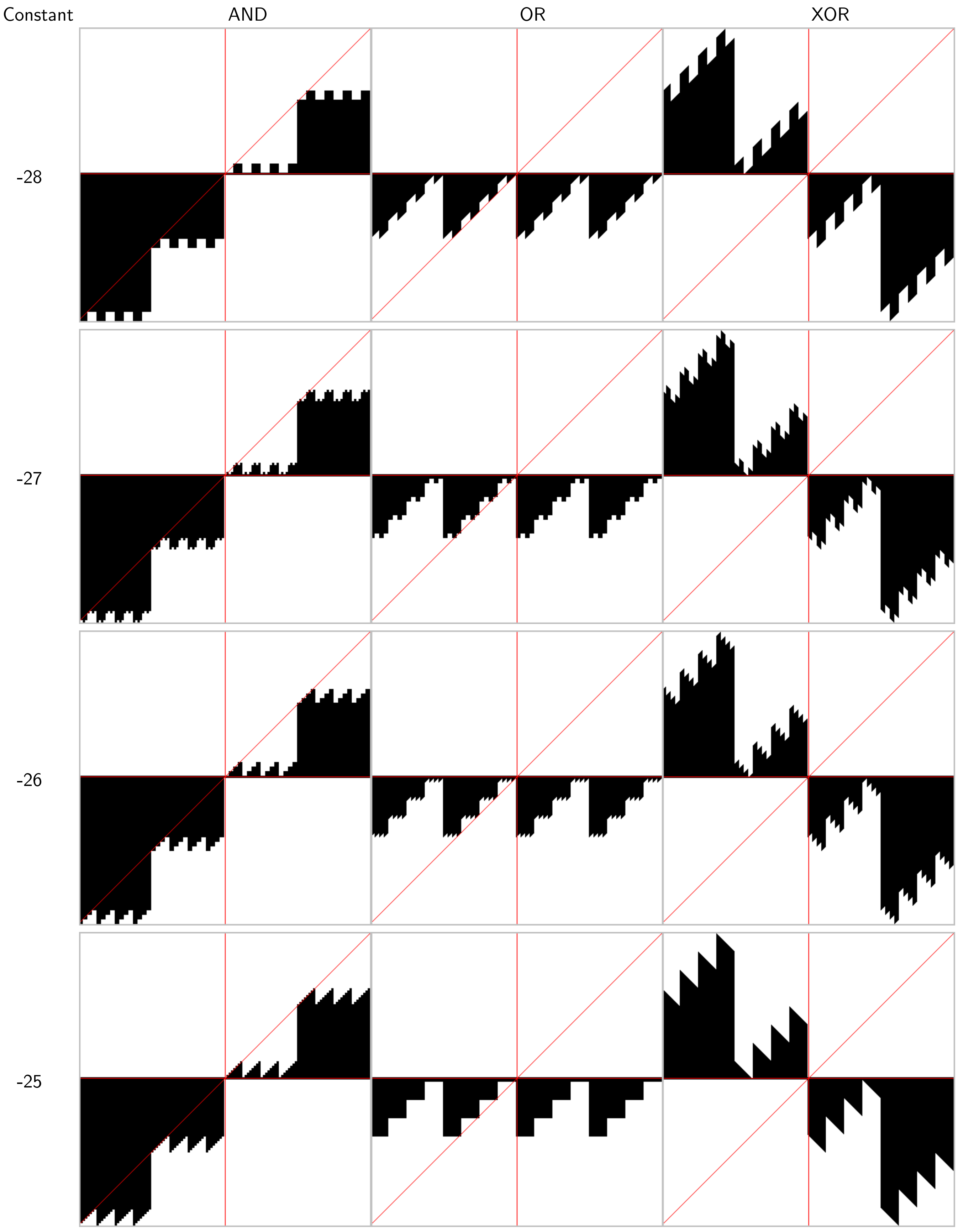

Constant
AND
OR
XOR
-48
-47
-46
-45


Constant
AND
OR
XOR
-44
-43
-42
-41

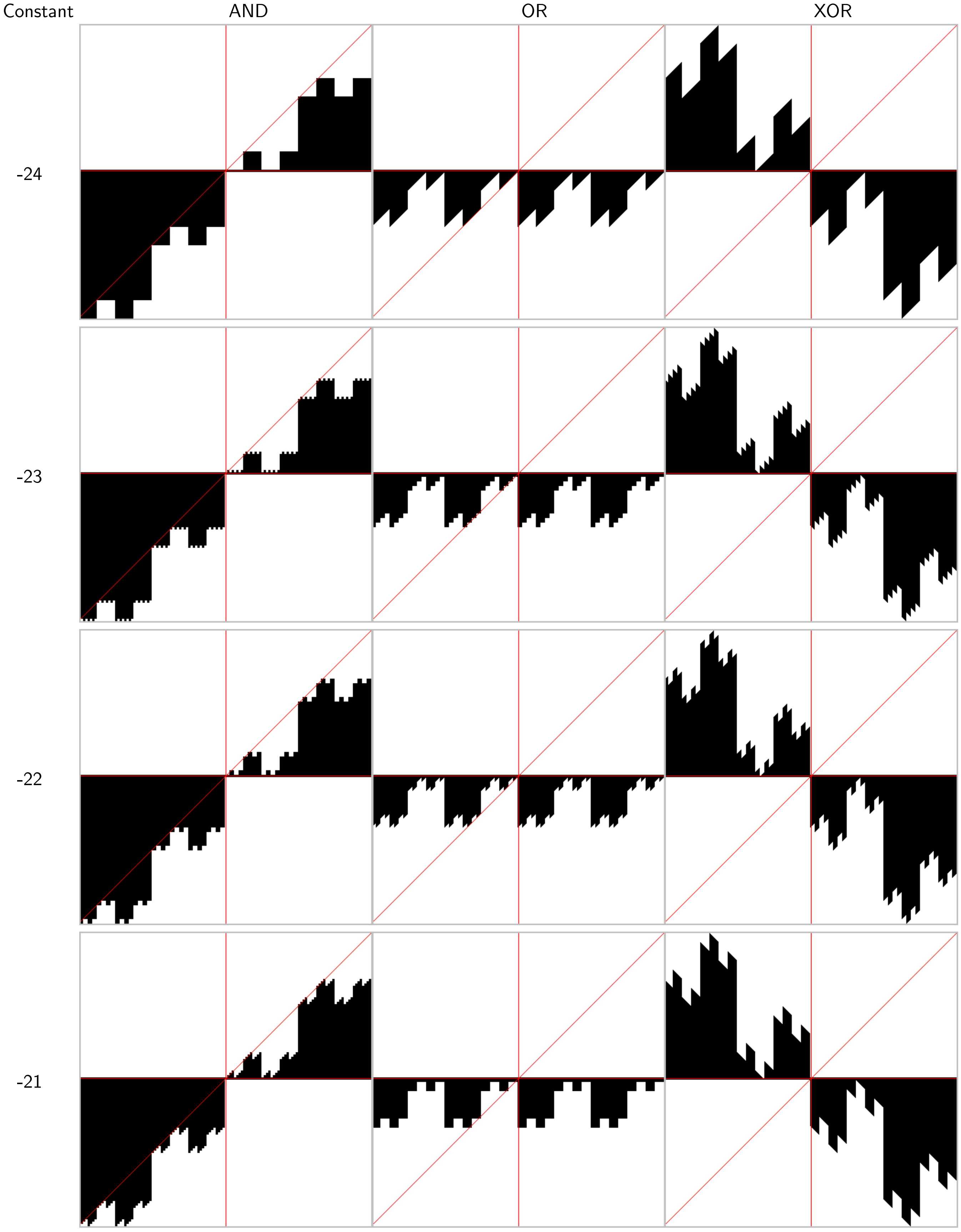

Constant
AND
OR
XOR
-40
-39
-38
-37

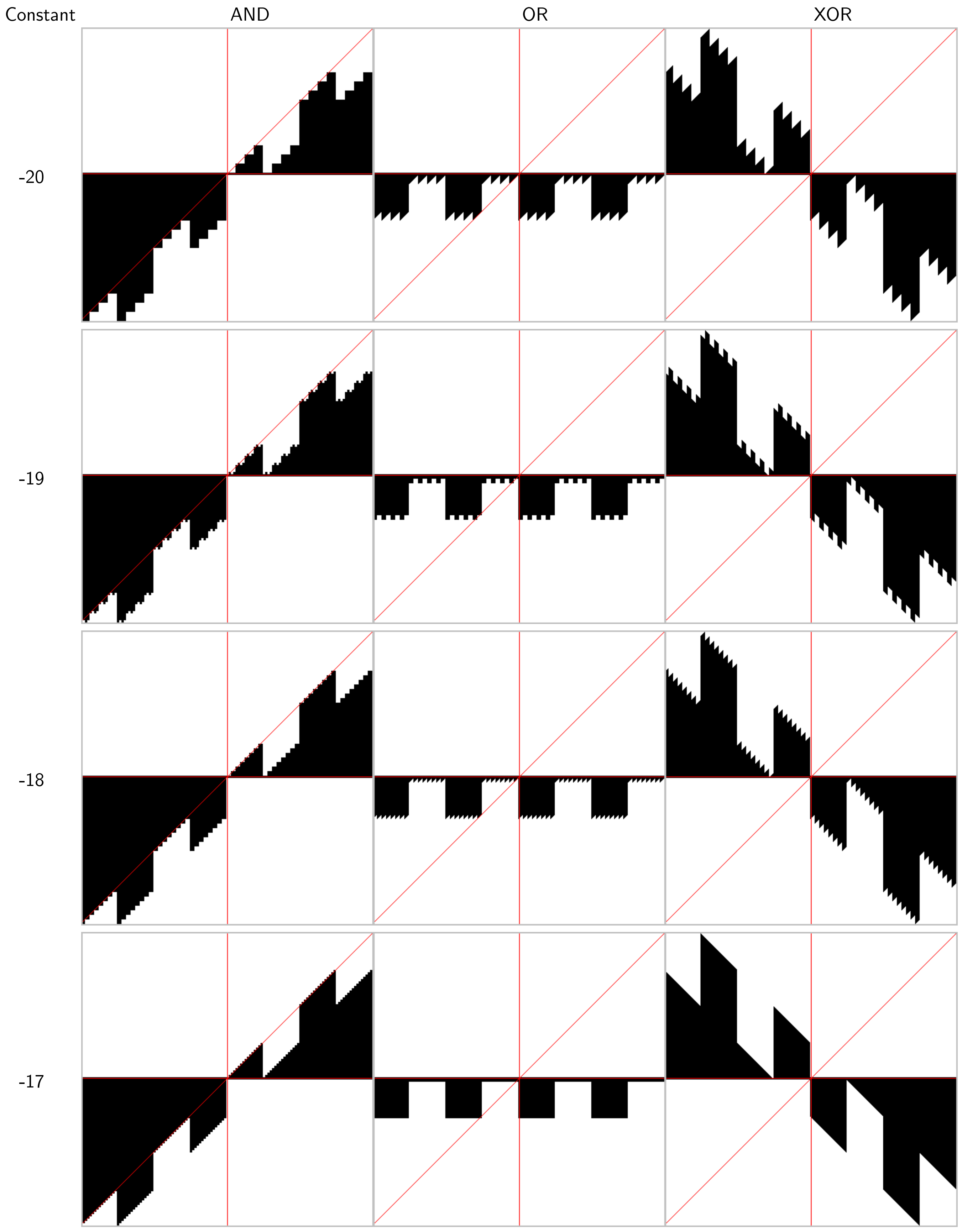

Constant
AND
OR
XOR
-36
-35
-34
-33

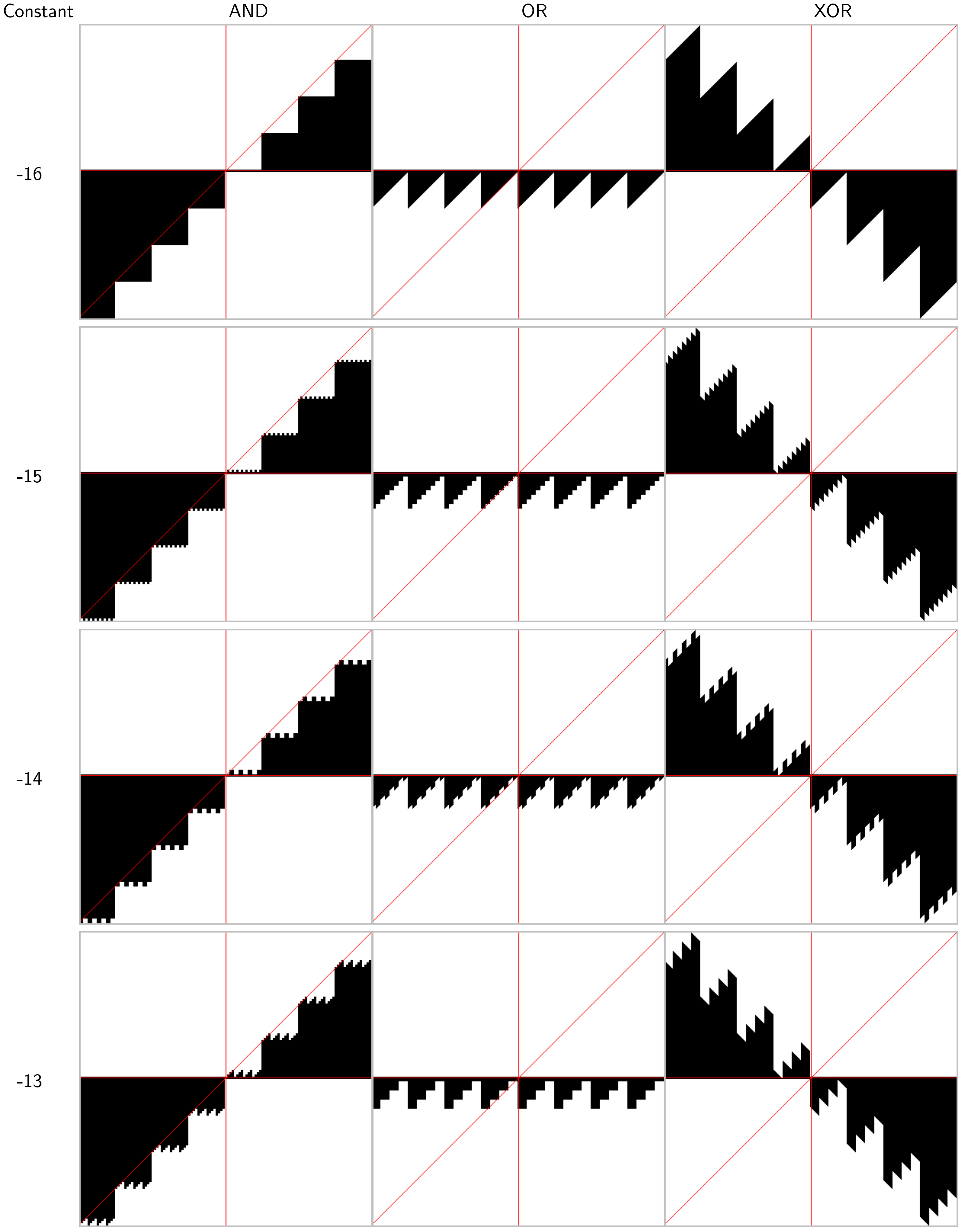
Constant
AND
OR
XOR
-32
-31
-30
-29


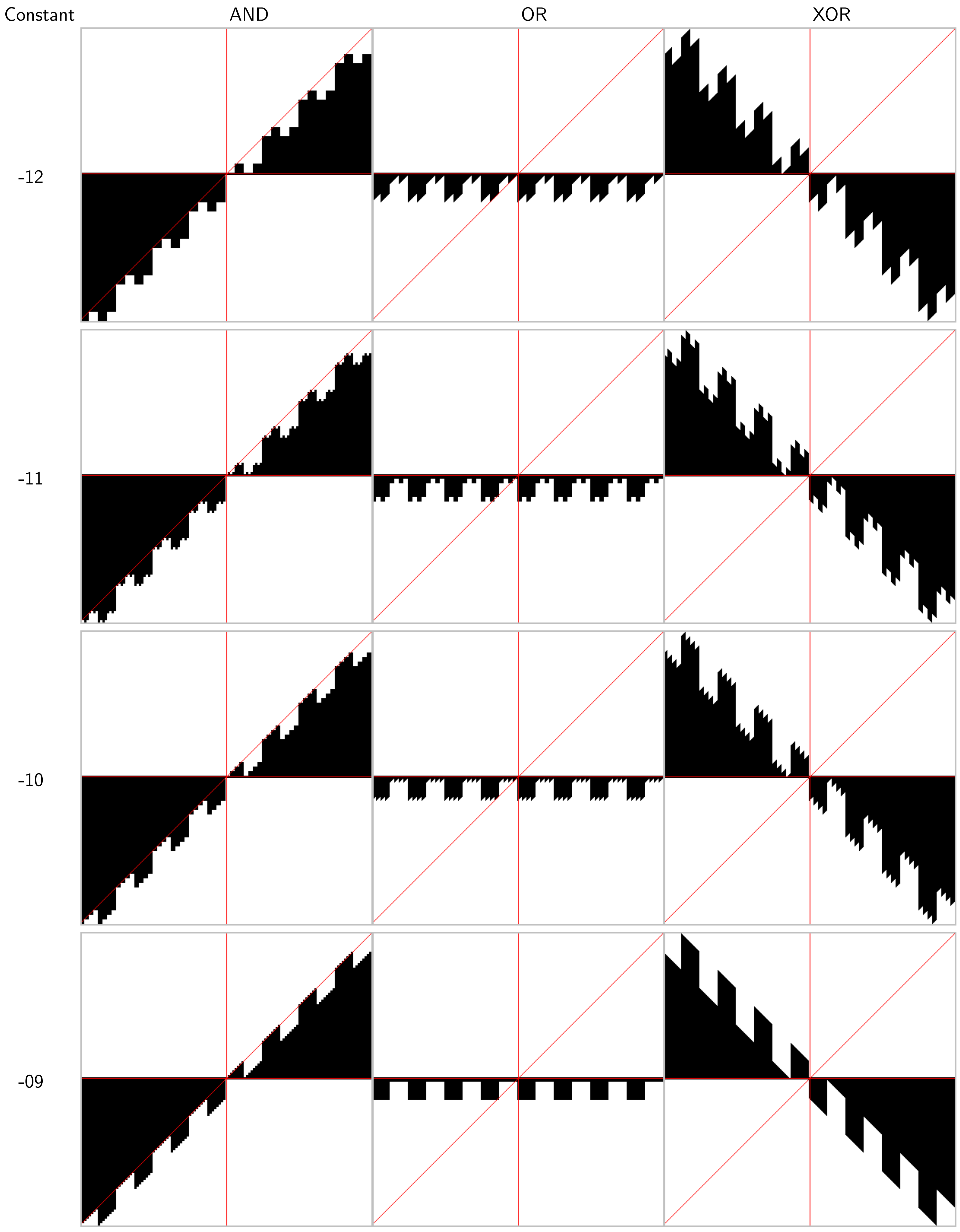

Constant
AND
OR
XOR
-28
-27
-26
-25

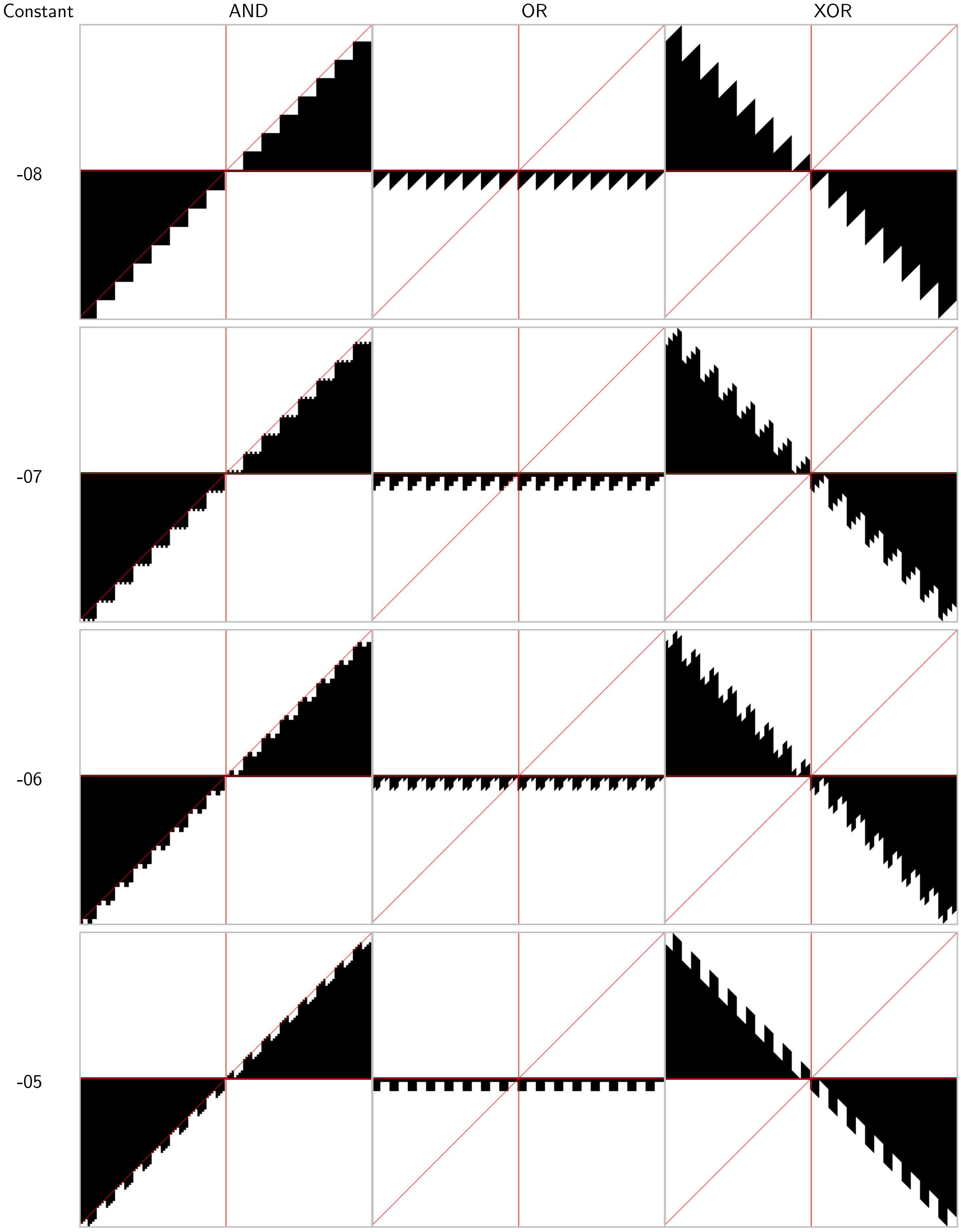
Constant AND OR XOR

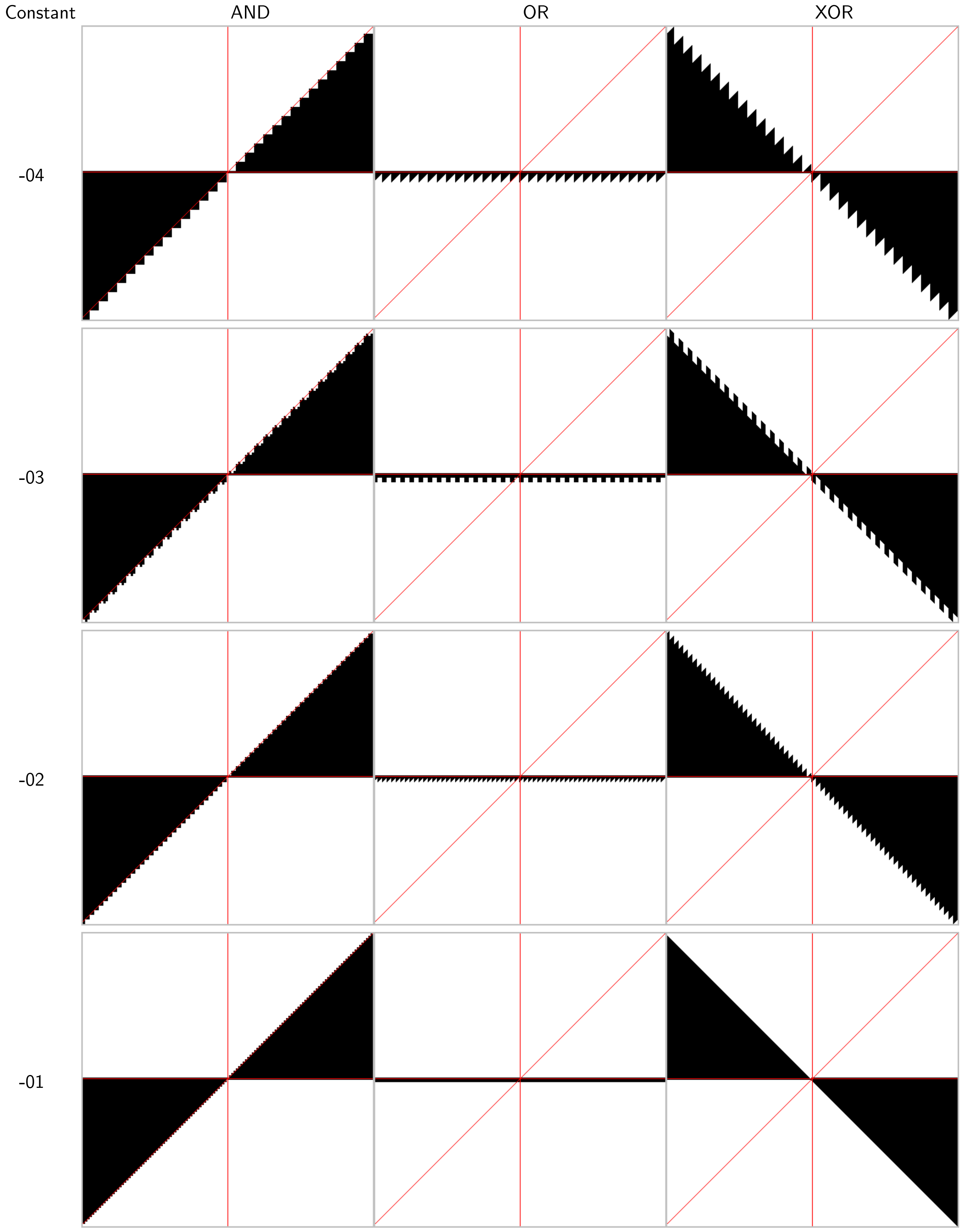


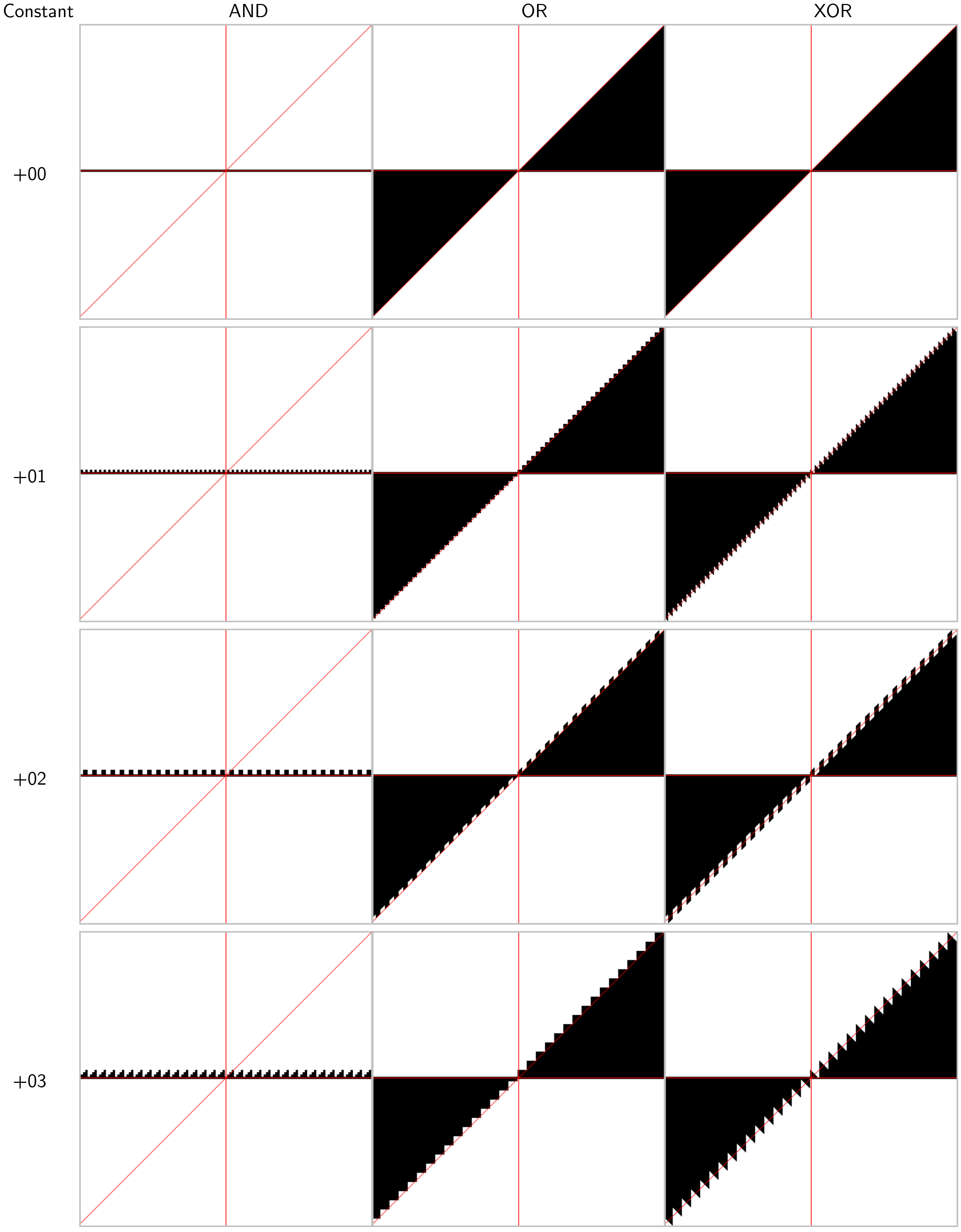

Constant
AND
OR
XOR
-20
-19
-18
-17

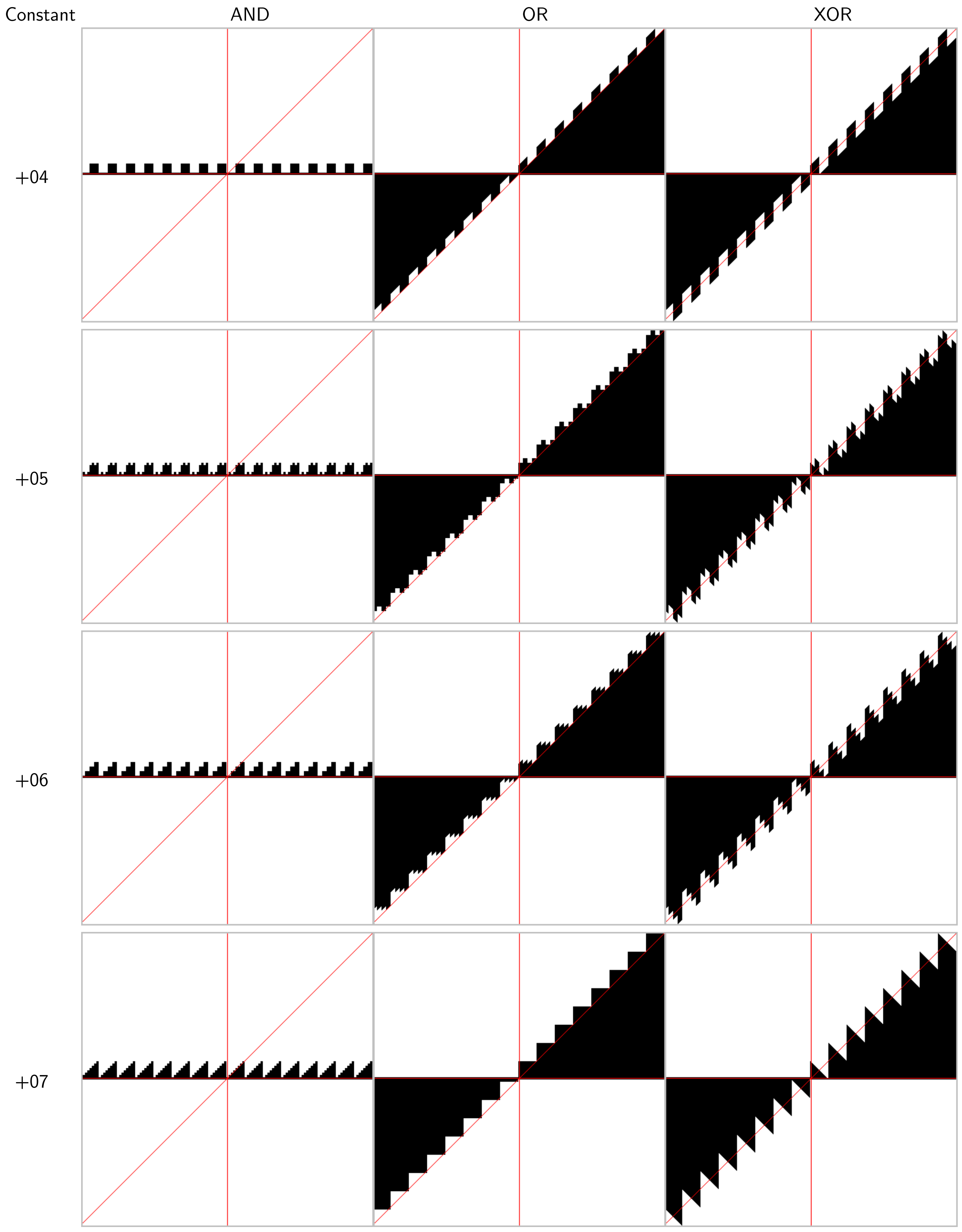

Constant
AND
OR
XOR
-16
-15
-14
-13

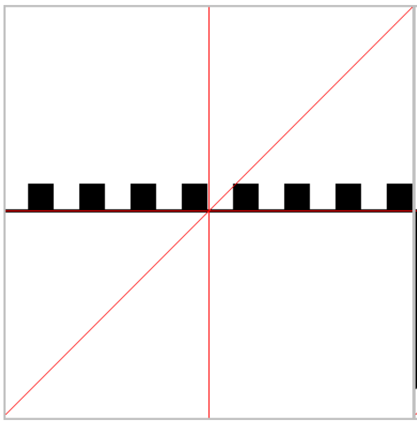
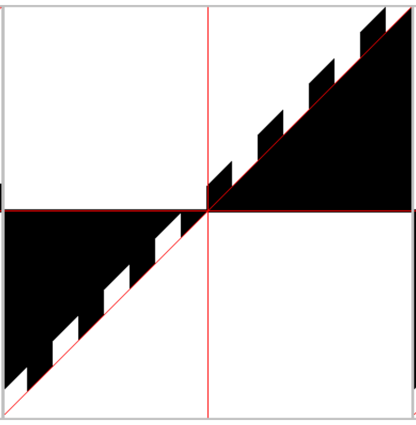
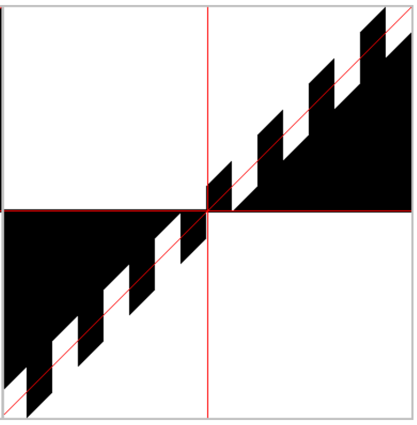
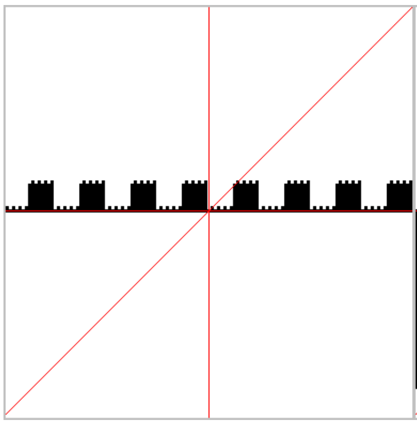
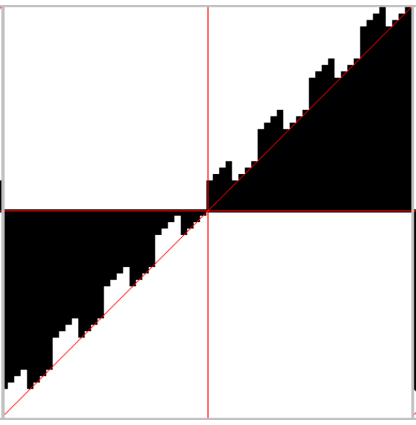
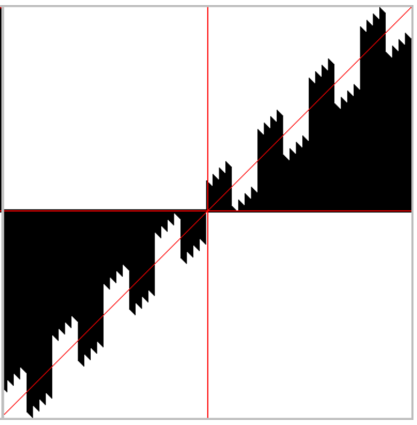
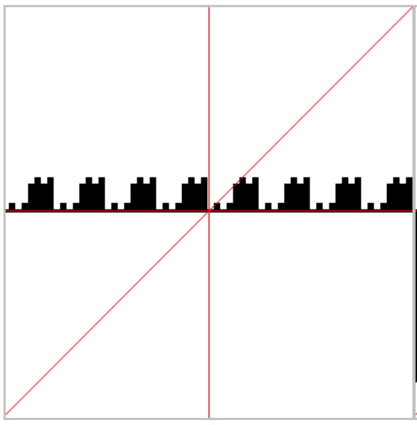
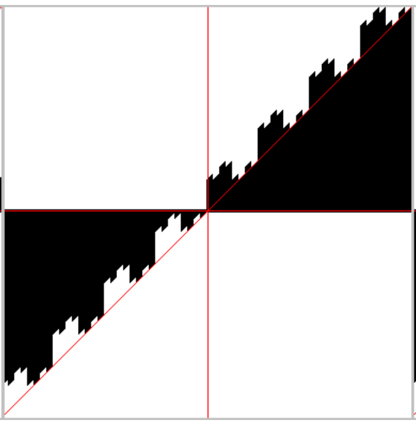
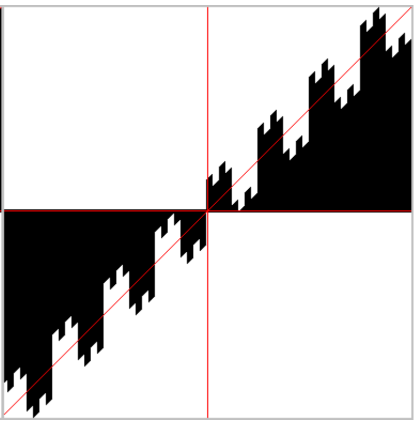
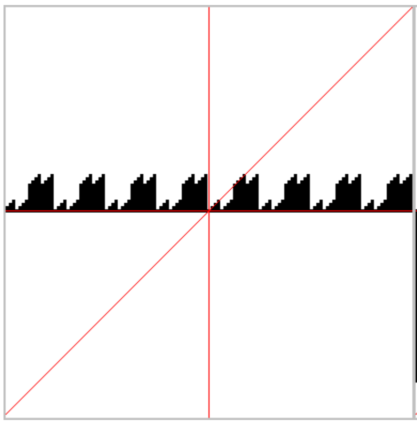
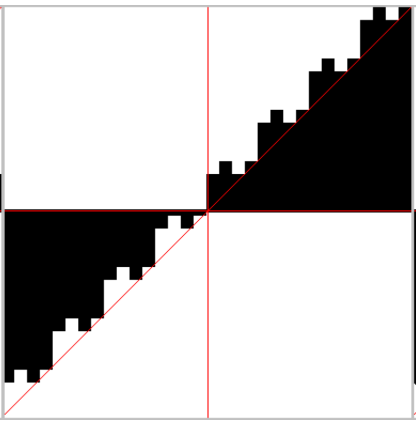
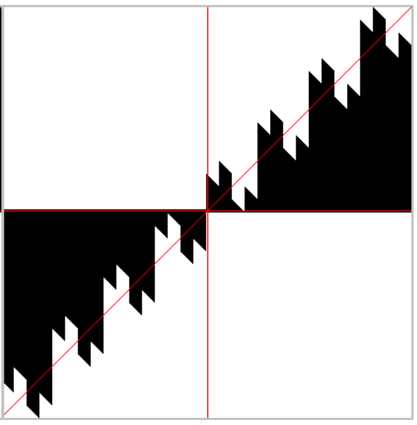

Constant
AND
OR
XOR
-12
-11
-10
-09

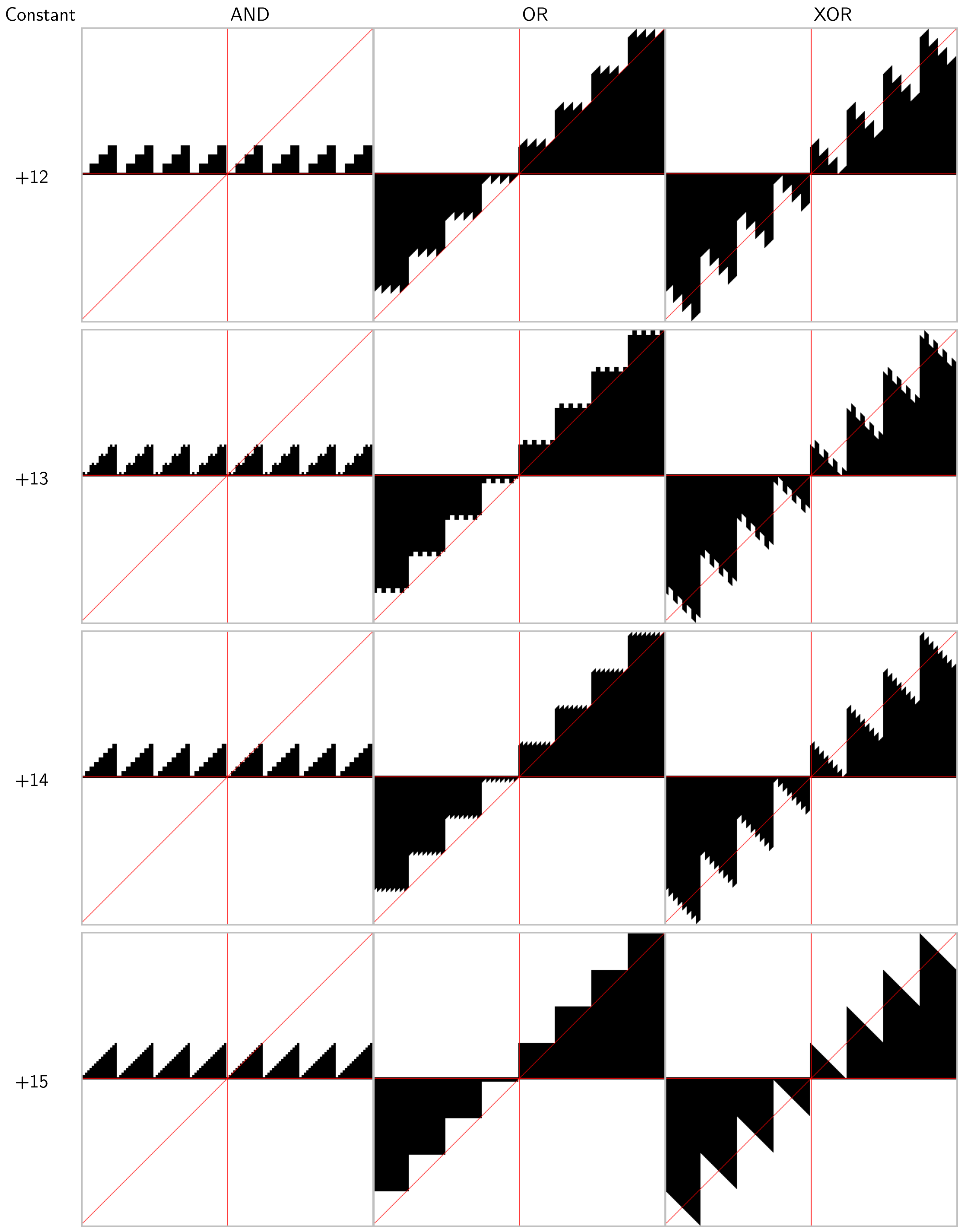

Constant
AND
OR
XOR
-08
-07
-06
-05

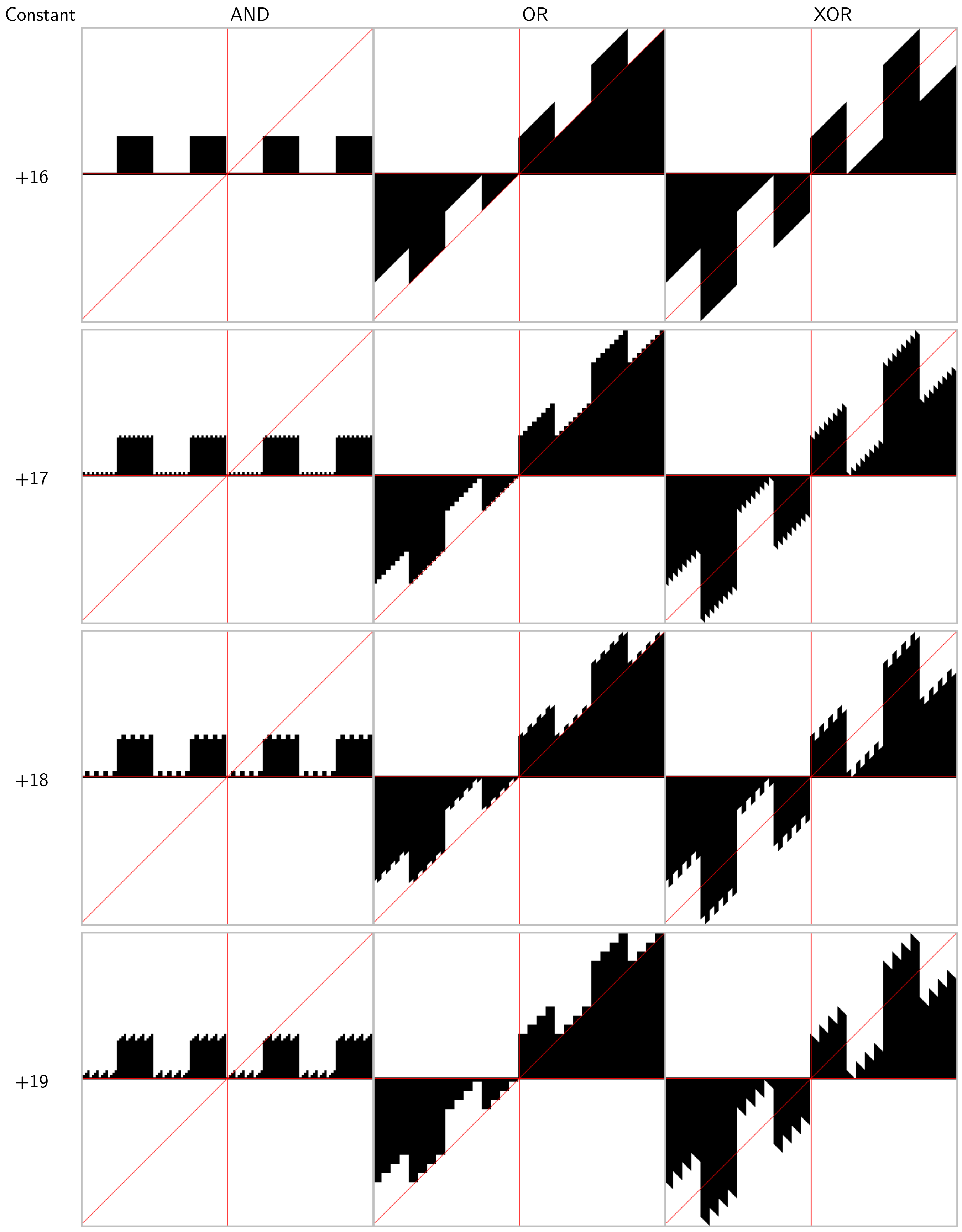

Constant
AND
OR
XOR
-04
-03
-02
-01

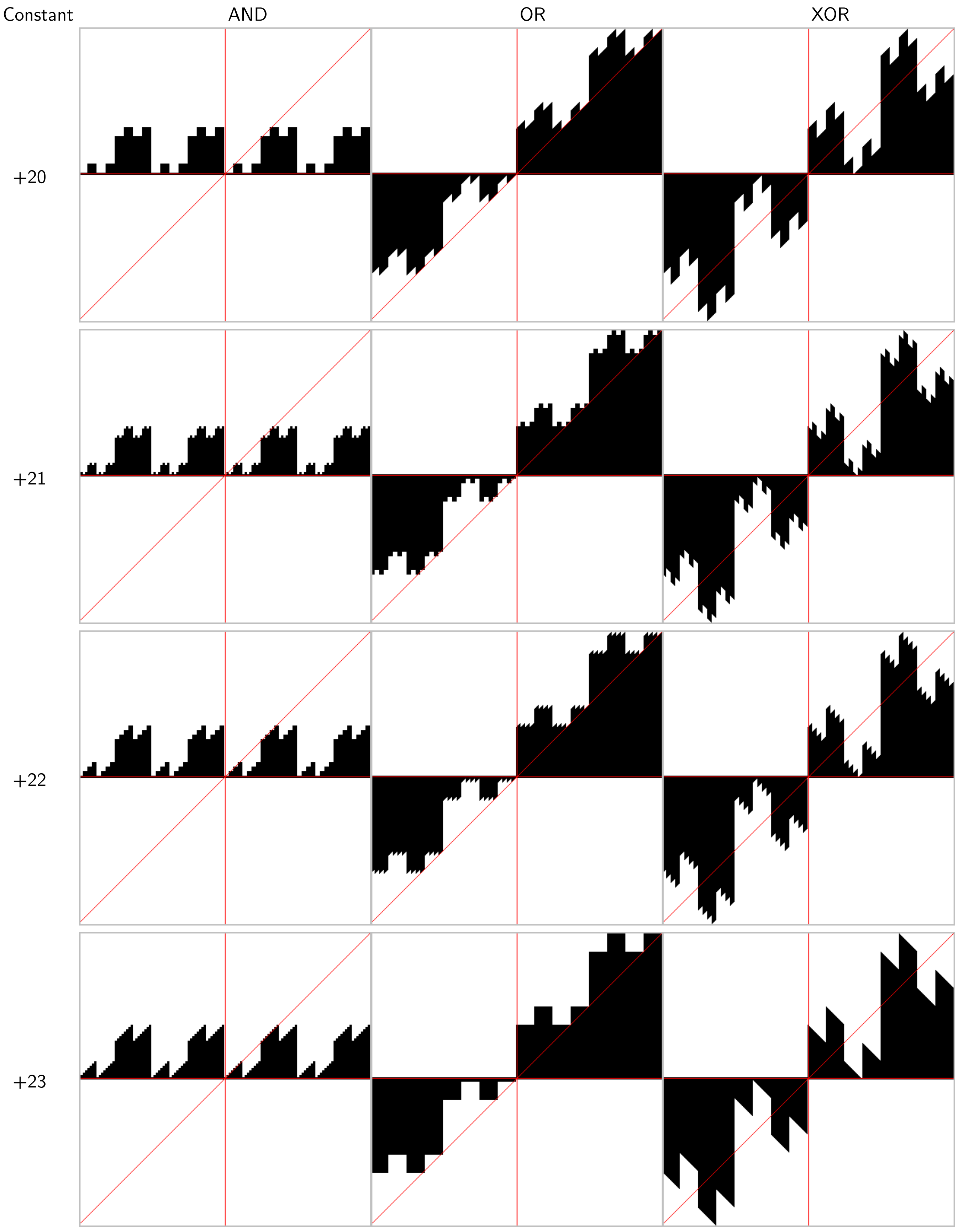

Constant
AND
OR
XOR
+00
+01
+02
+03


Constant
AND
OR
XOR
+04
+05
+06
+07

| Constant | AND | OR | XOR |
|---|---|---|---|
| +08 |  |  |  |
| +09 |  |  |  |
| +10 |  |  |  |
| +11 |  |  |  |


Constant
AND
OR
XOR
+12
+13
+14
+15


Constant
AND
OR
XOR
+16
+17
+18
+19


Constant
AND
OR
XOR
+20
+21
+22
+23

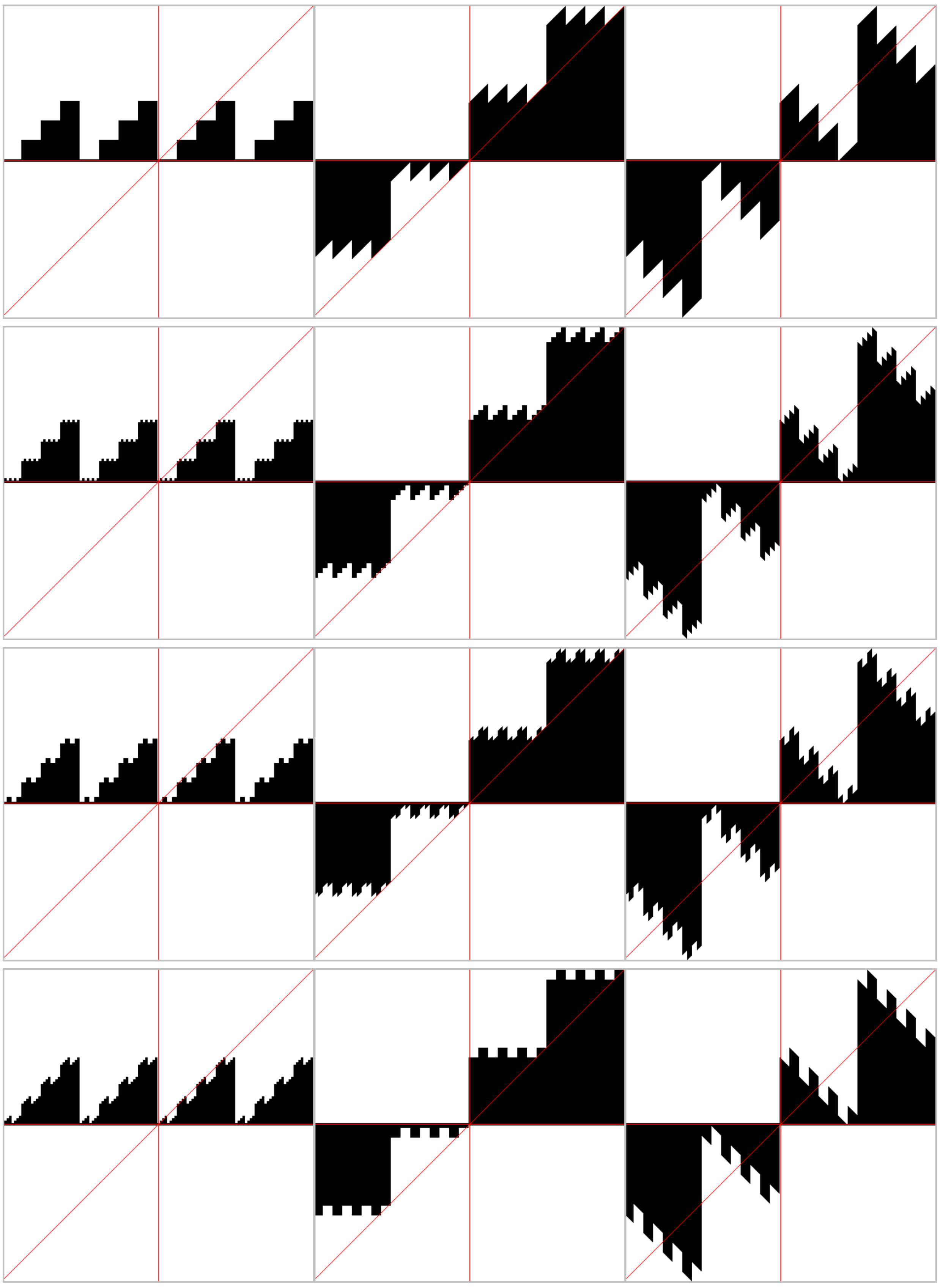
Constant AND OR XOR

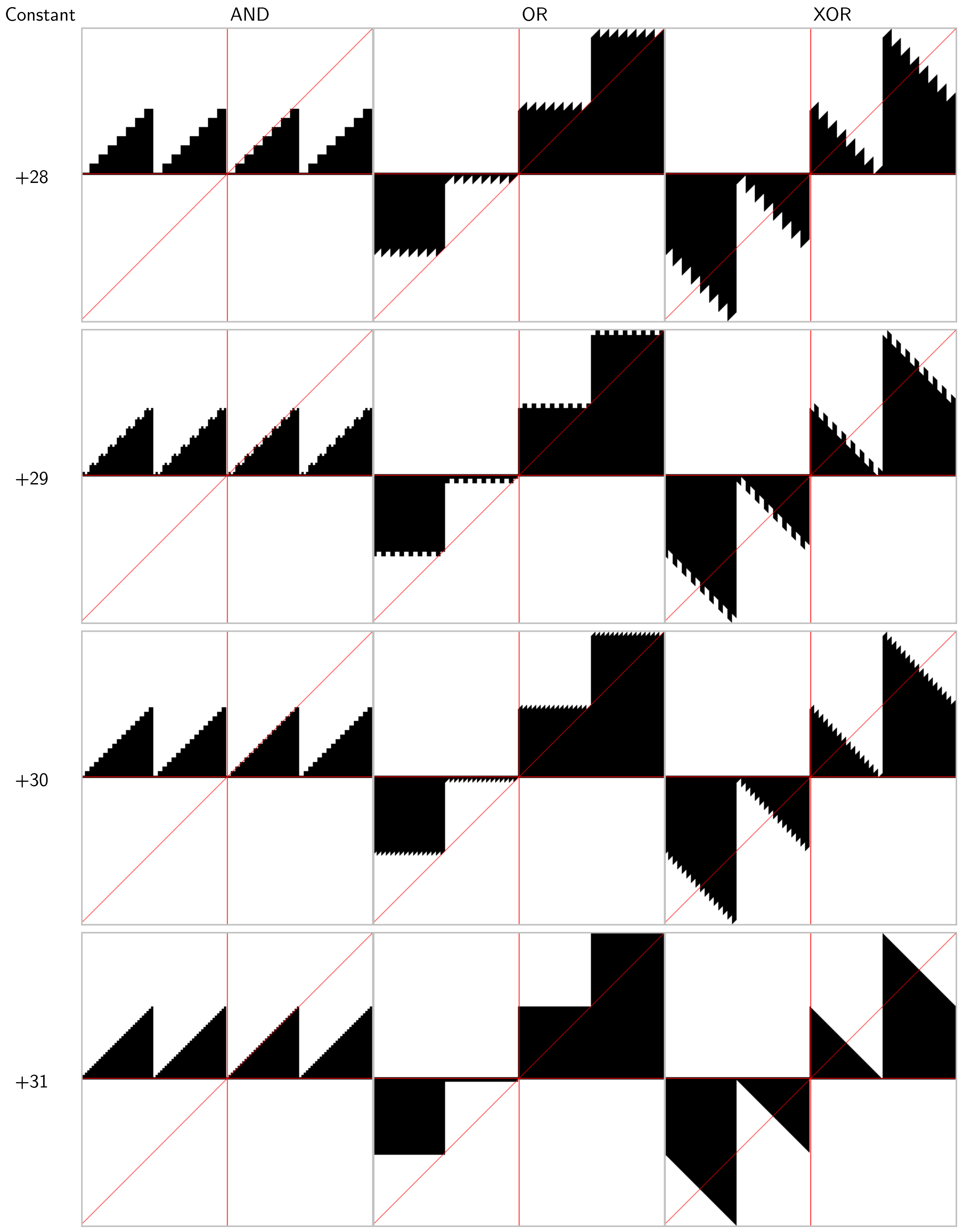
+24

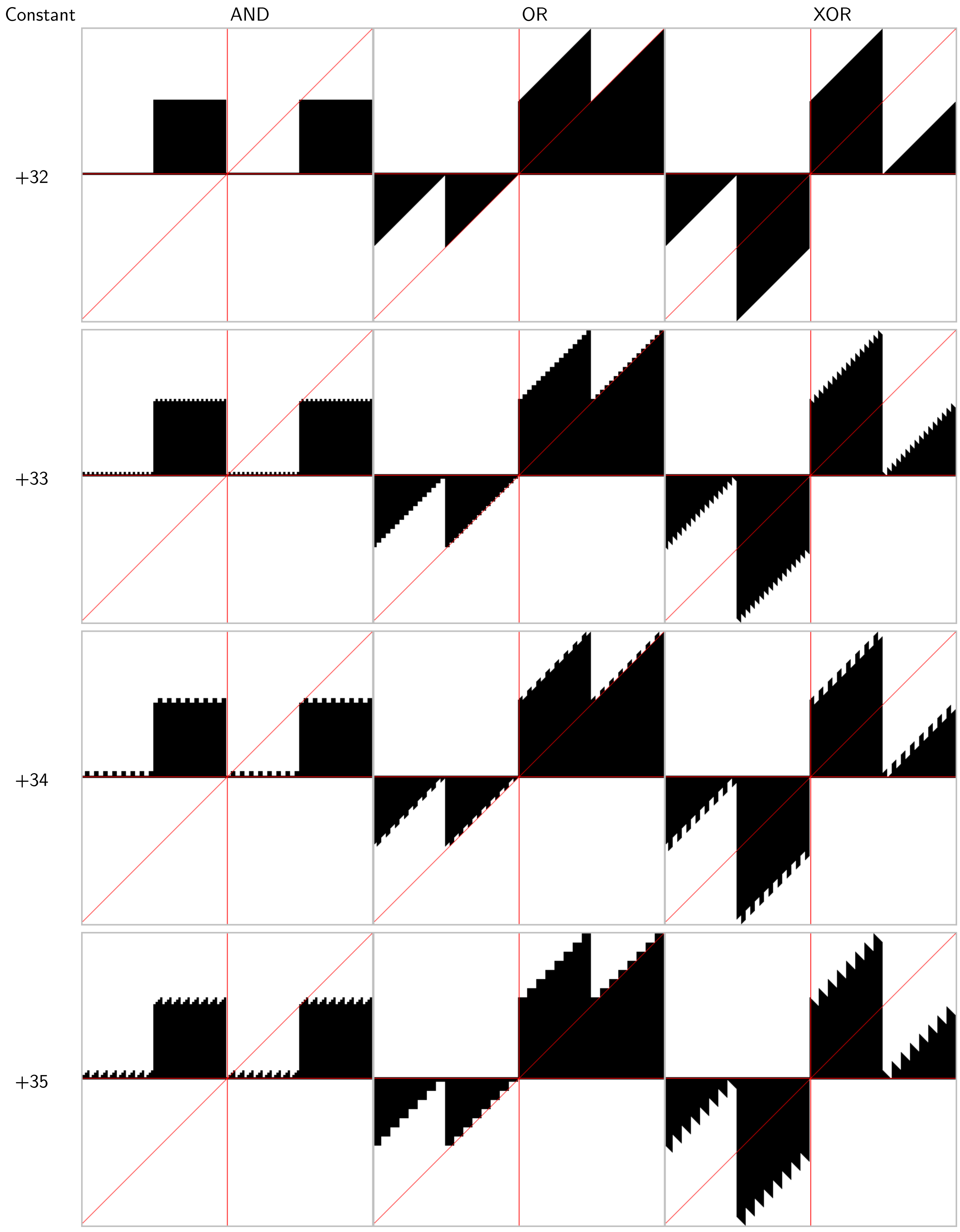
+25

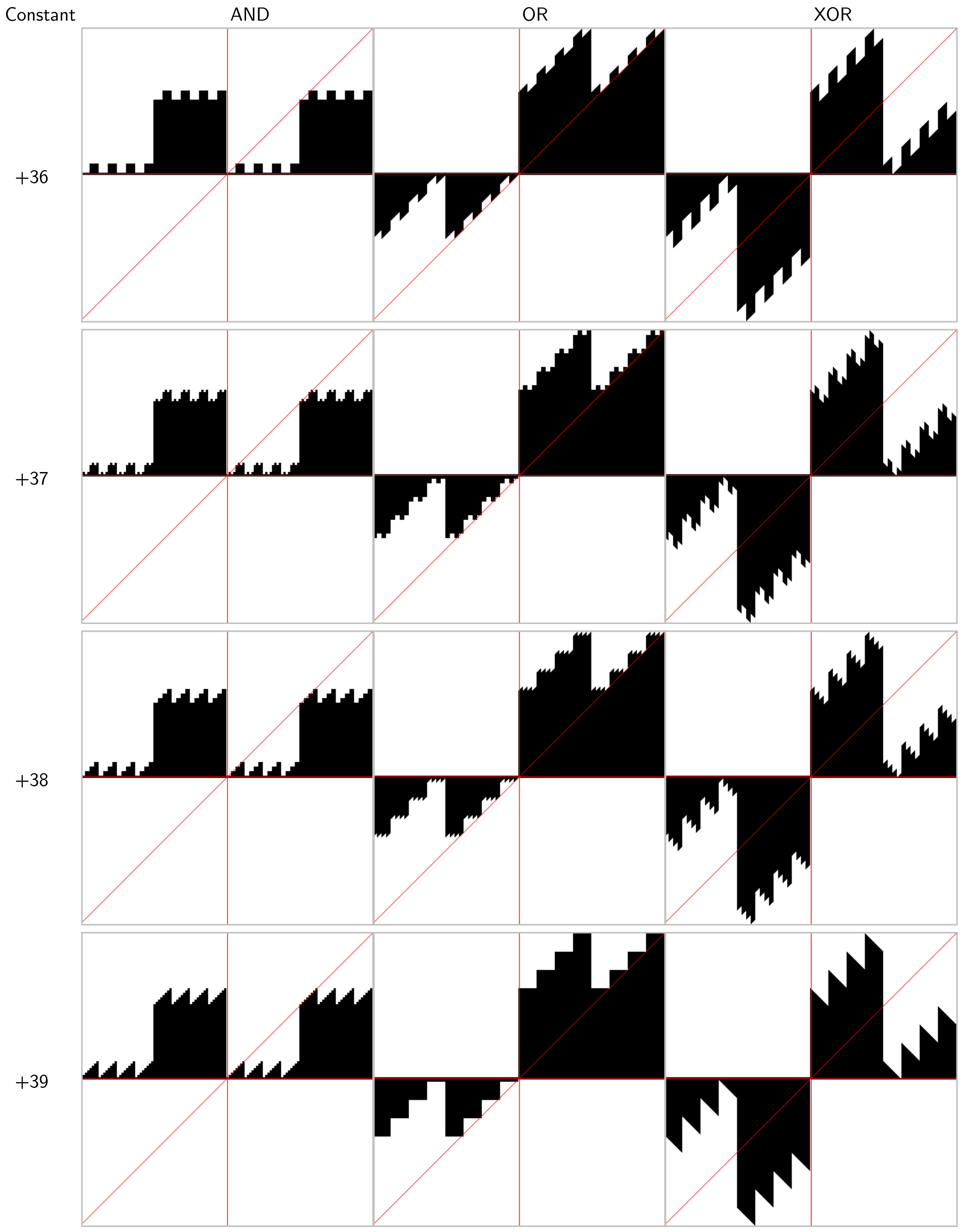
+26

+27




Constant
AND
OR
XOR
+28
+29
+30
+31


Constant
AND
OR
XOR
+32
+33
+34
+35


Constant
AND
OR
XOR
+36
+37
+38
+39

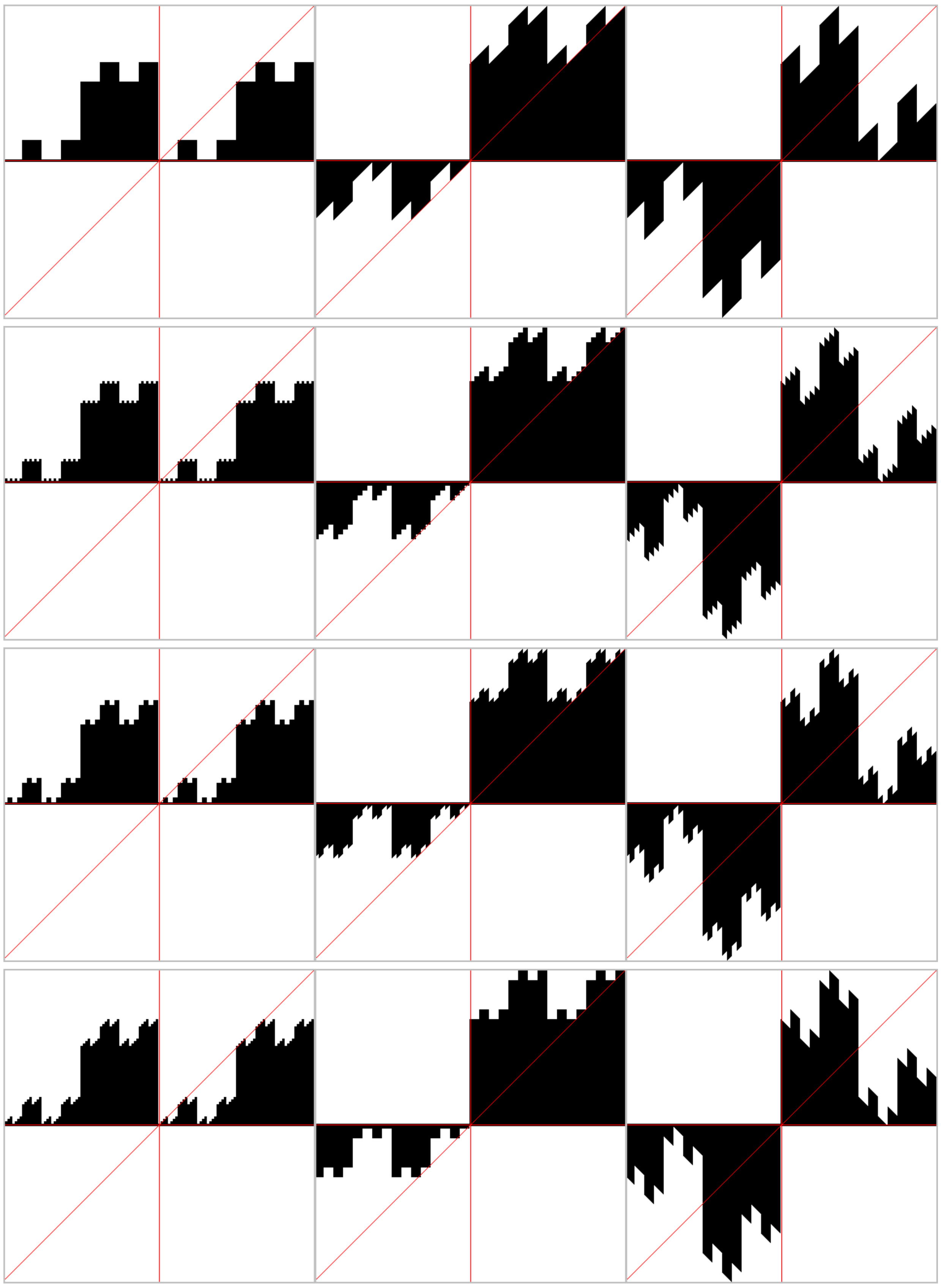
Constant AND OR XOR

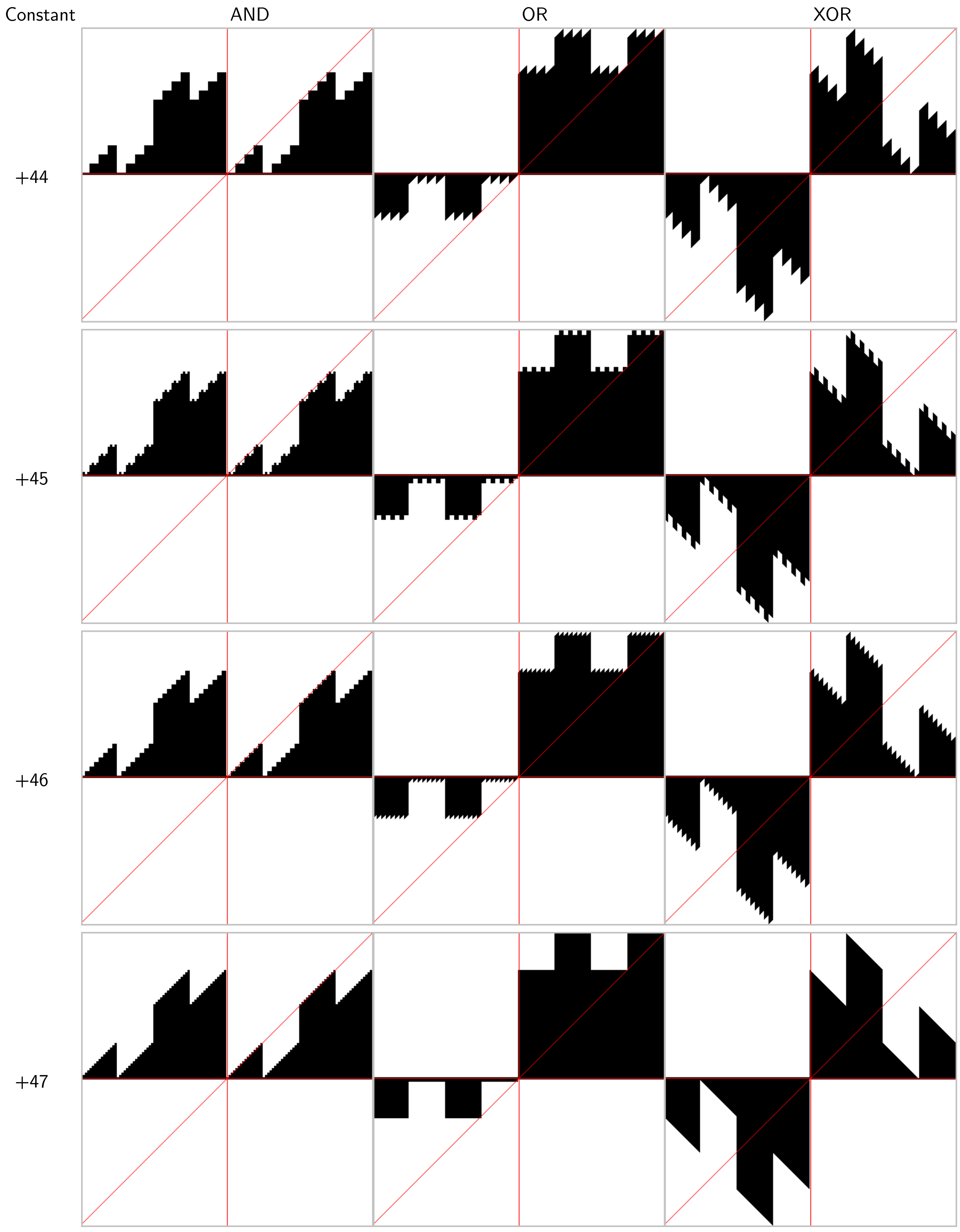
+40

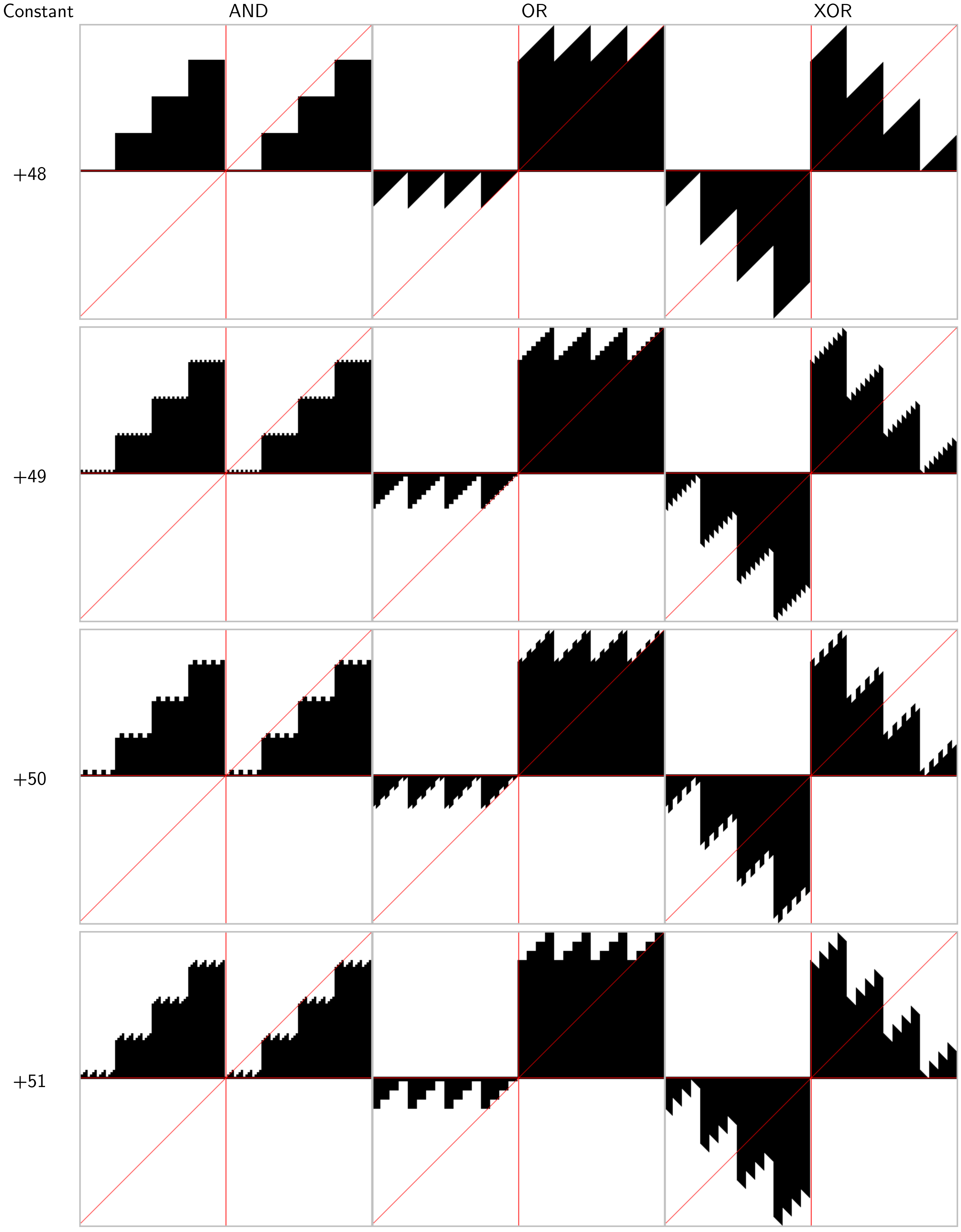
+41

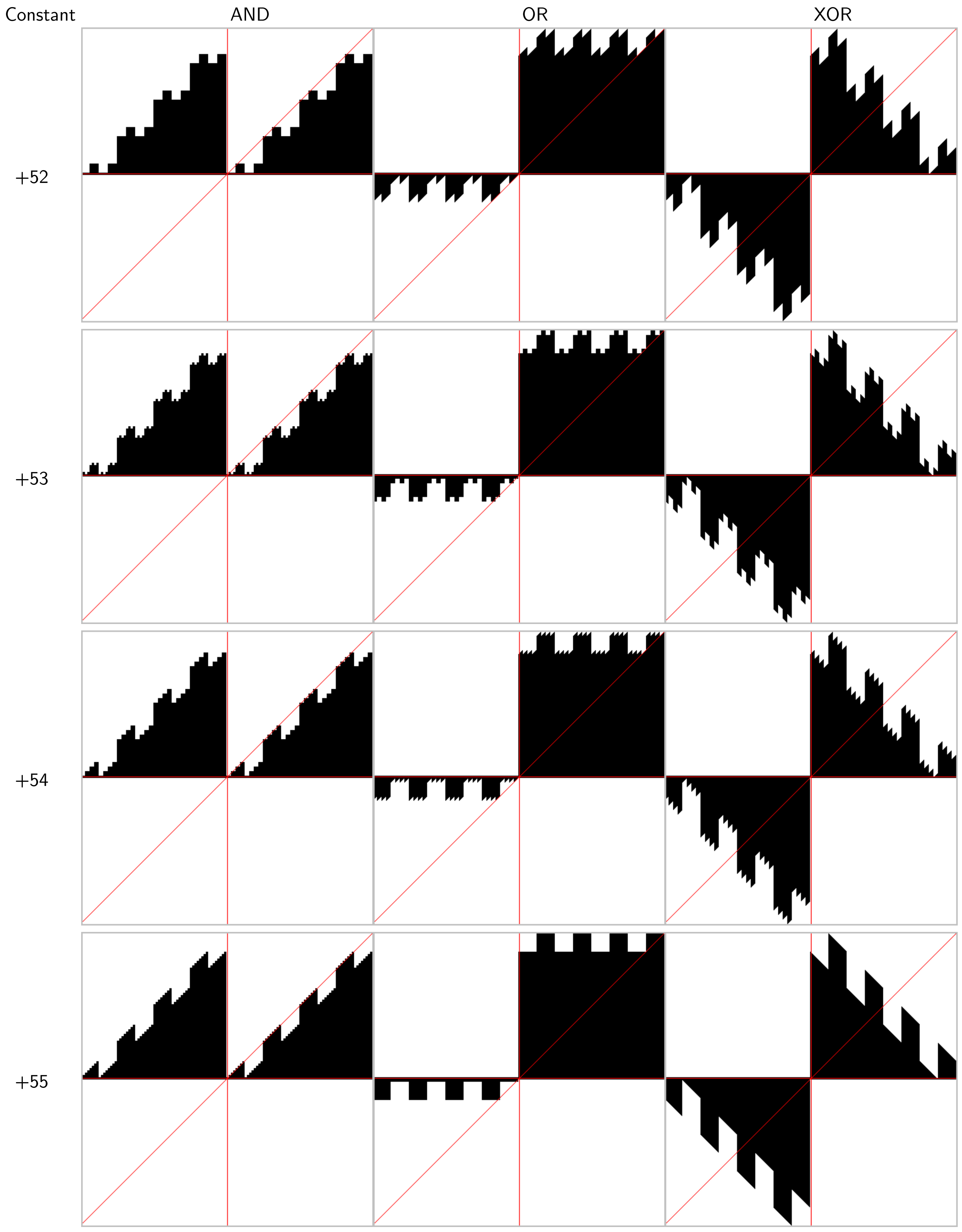
+42

+43



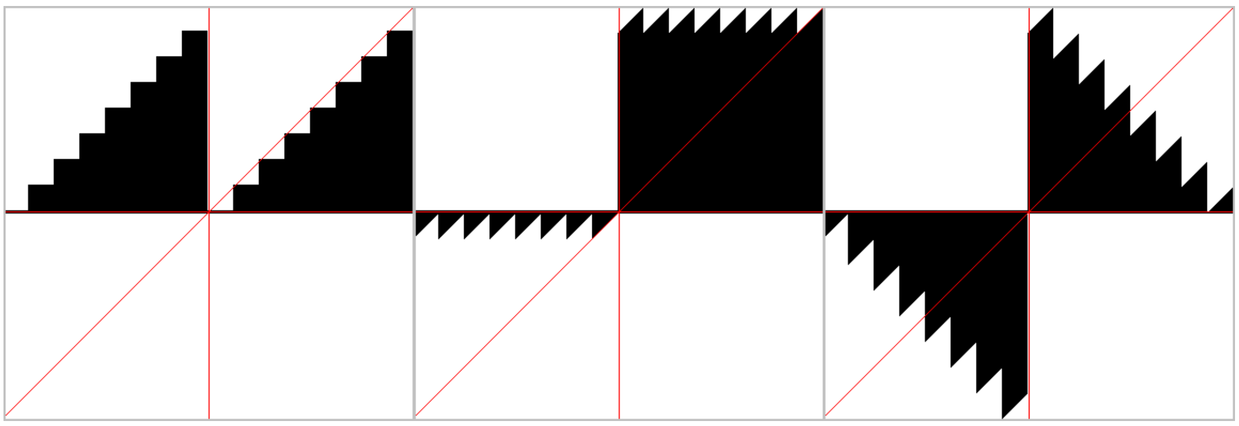

Constant
AND
OR
XOR
+44
+45
+46
+47

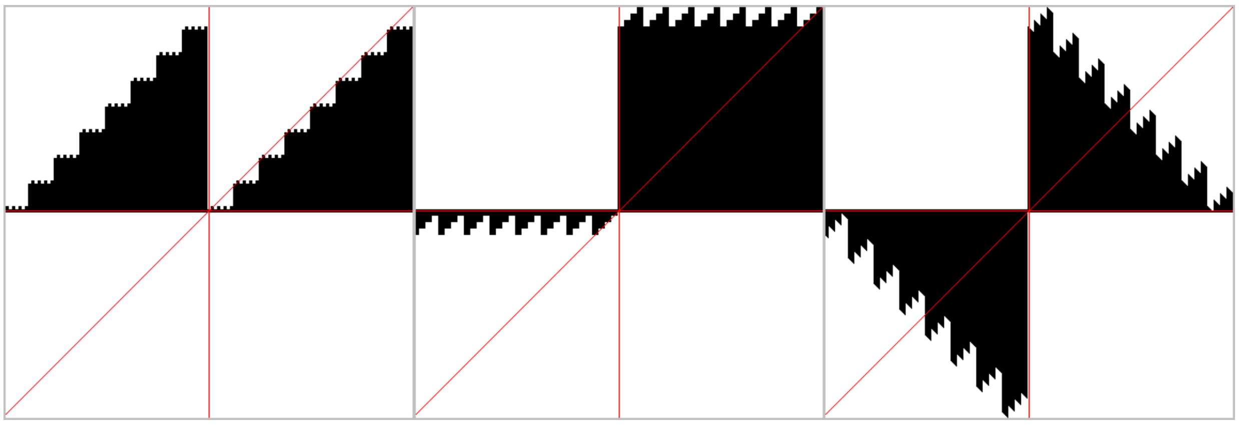

Constant
AND
OR
XOR
+48
+49
+50
+51

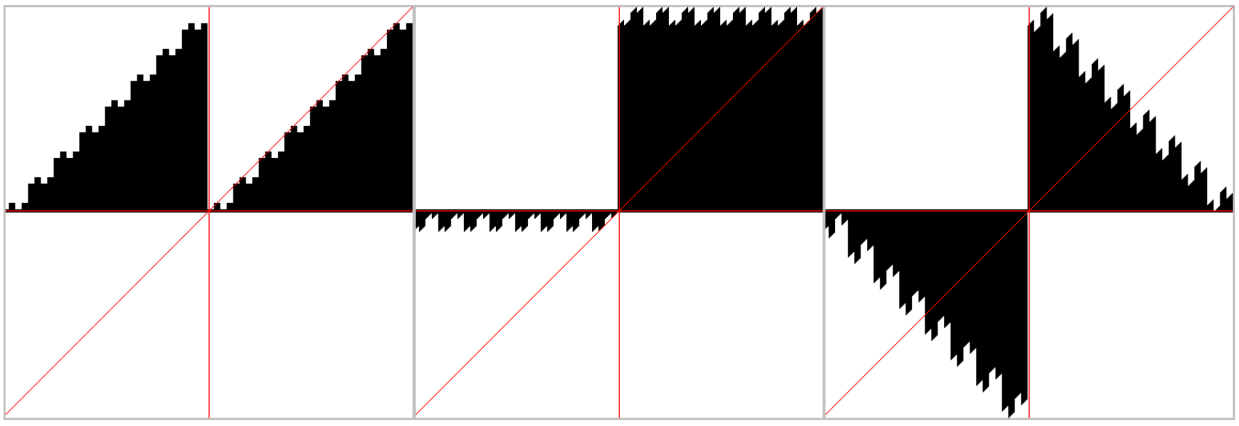


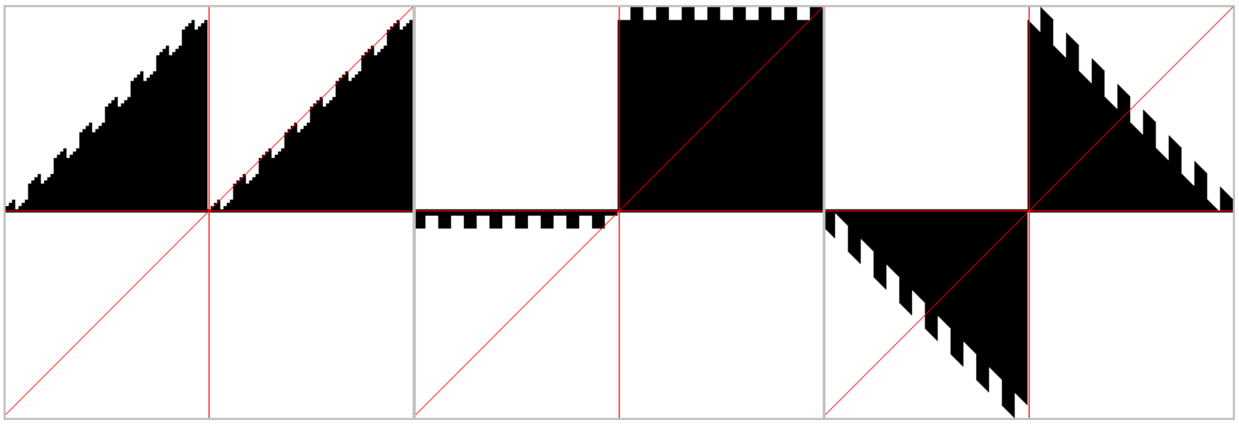
Constant
AND
OR
XOR
+52
+53
+54
+55

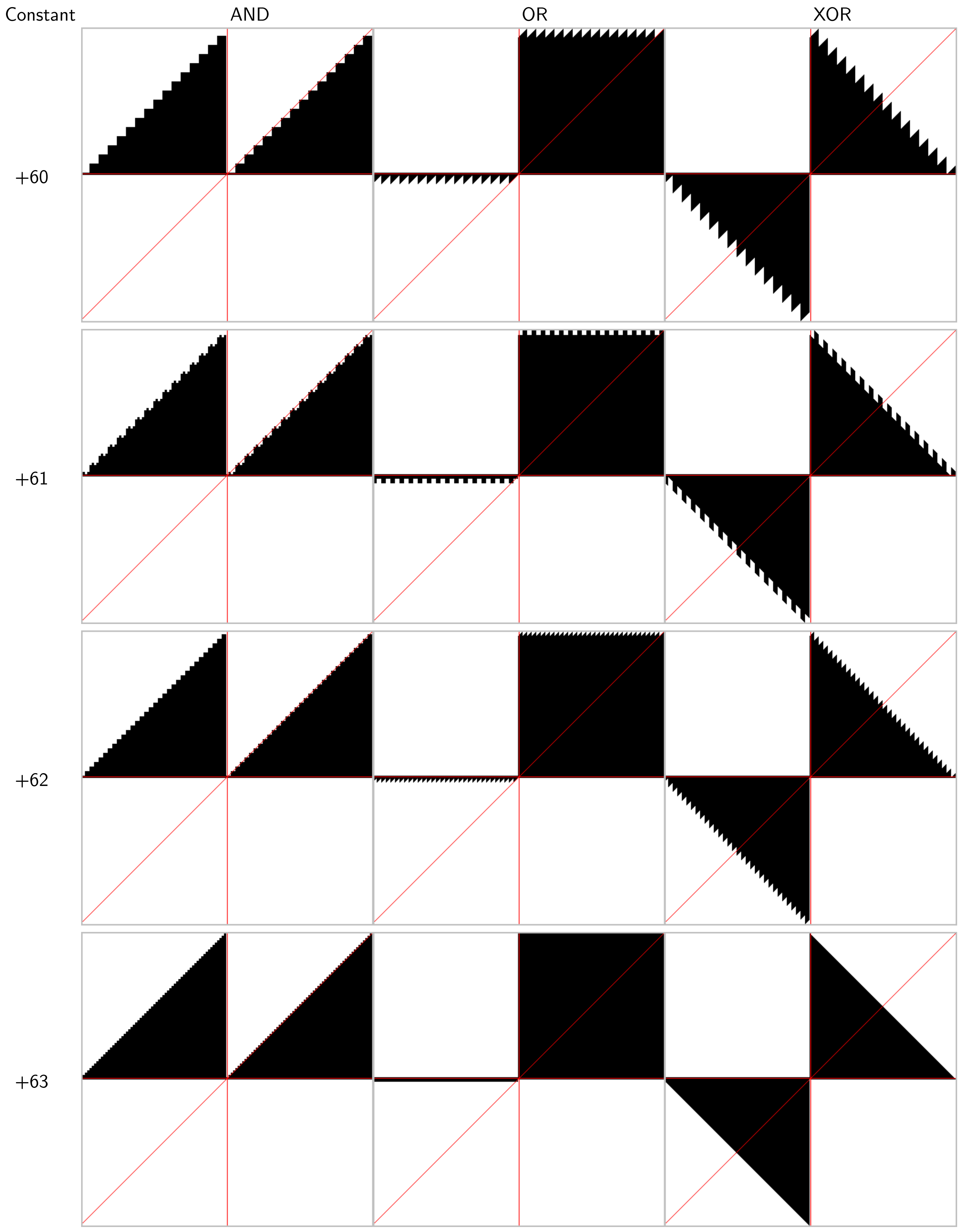
Constant AND OR XOR

+56


+57


+58


+59



Constant
AND
OR
XOR
+60
+61
+62
+63