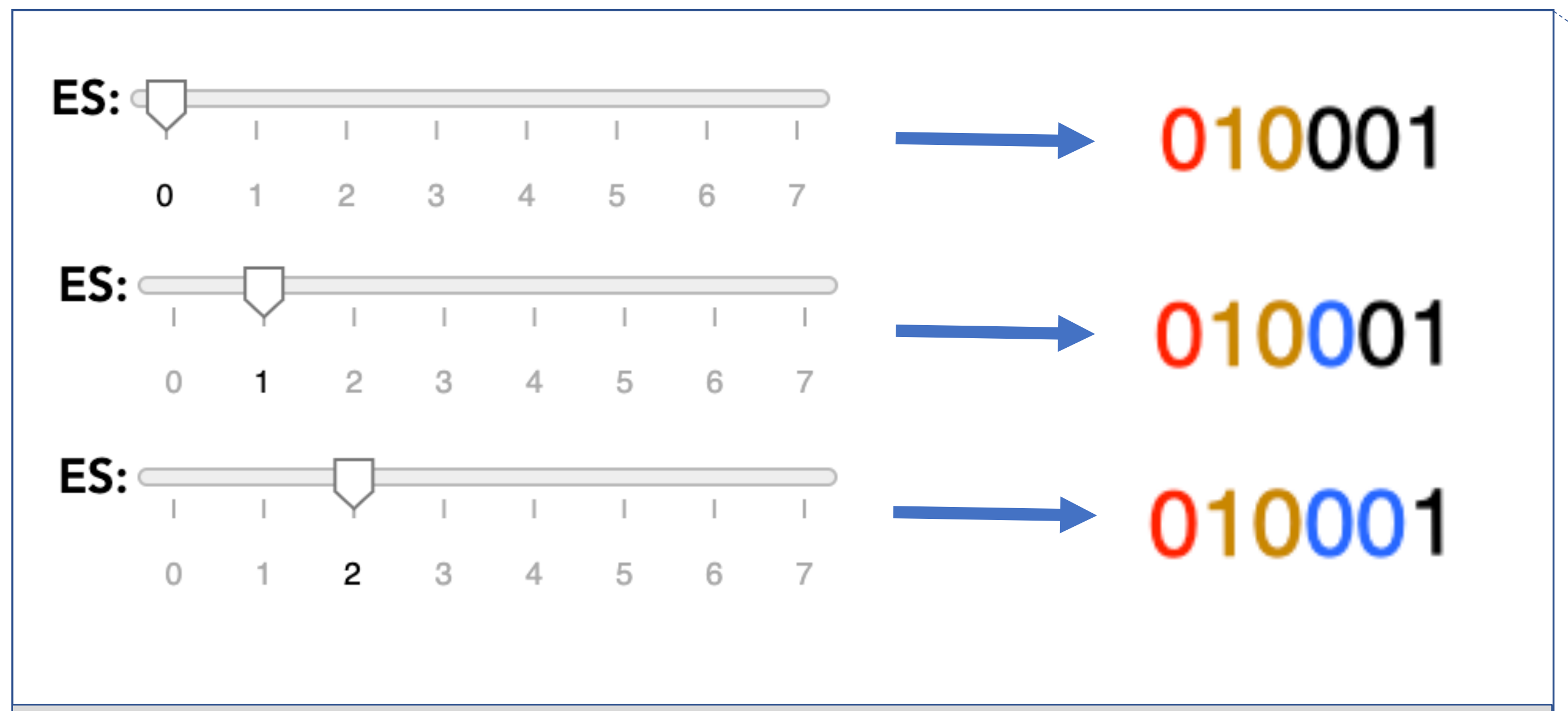
Well-Rounded: Visualizing Floating Point Representations

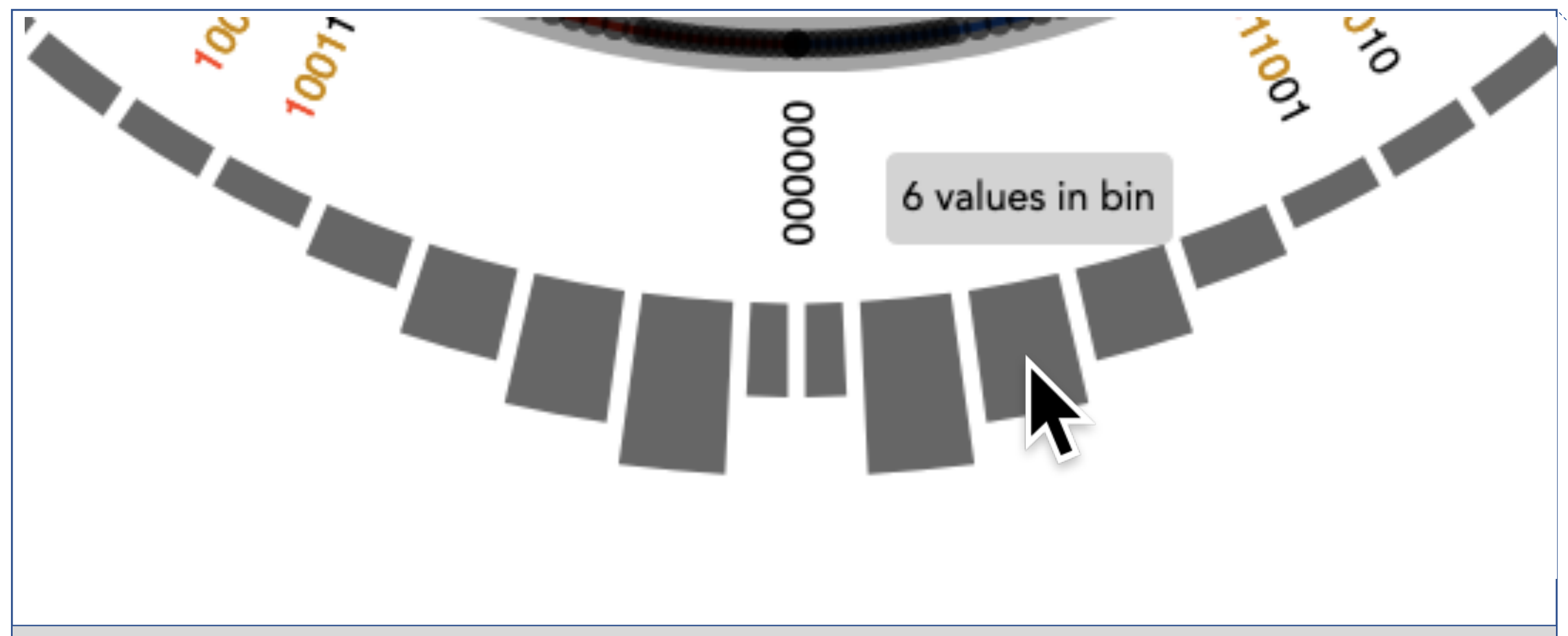
Neil Ryan, Katie Lim, Gus Smith, Dan Petrisko

Motivation

Machine learning and AI have brought about revolutions in countless fields—revolutions that require incredibly high power and memory bandwidth. Responding to increased hardware demand, computer architects are rethinking even the most basic design decisions, such as how real numbers are represented in hardware. As a result, **the age-old IEEE 754 floating point types ("floats") have several modern competitors—such as the posit—which are smaller, more energy-efficient, and have better numerical behavior.**



Posit parameters ***n*** and ***es*** are controlled via sliders. The effects of the parameters are visualized in a number of ways, the simplest being the changing colors of the bitfields in the bitstring.



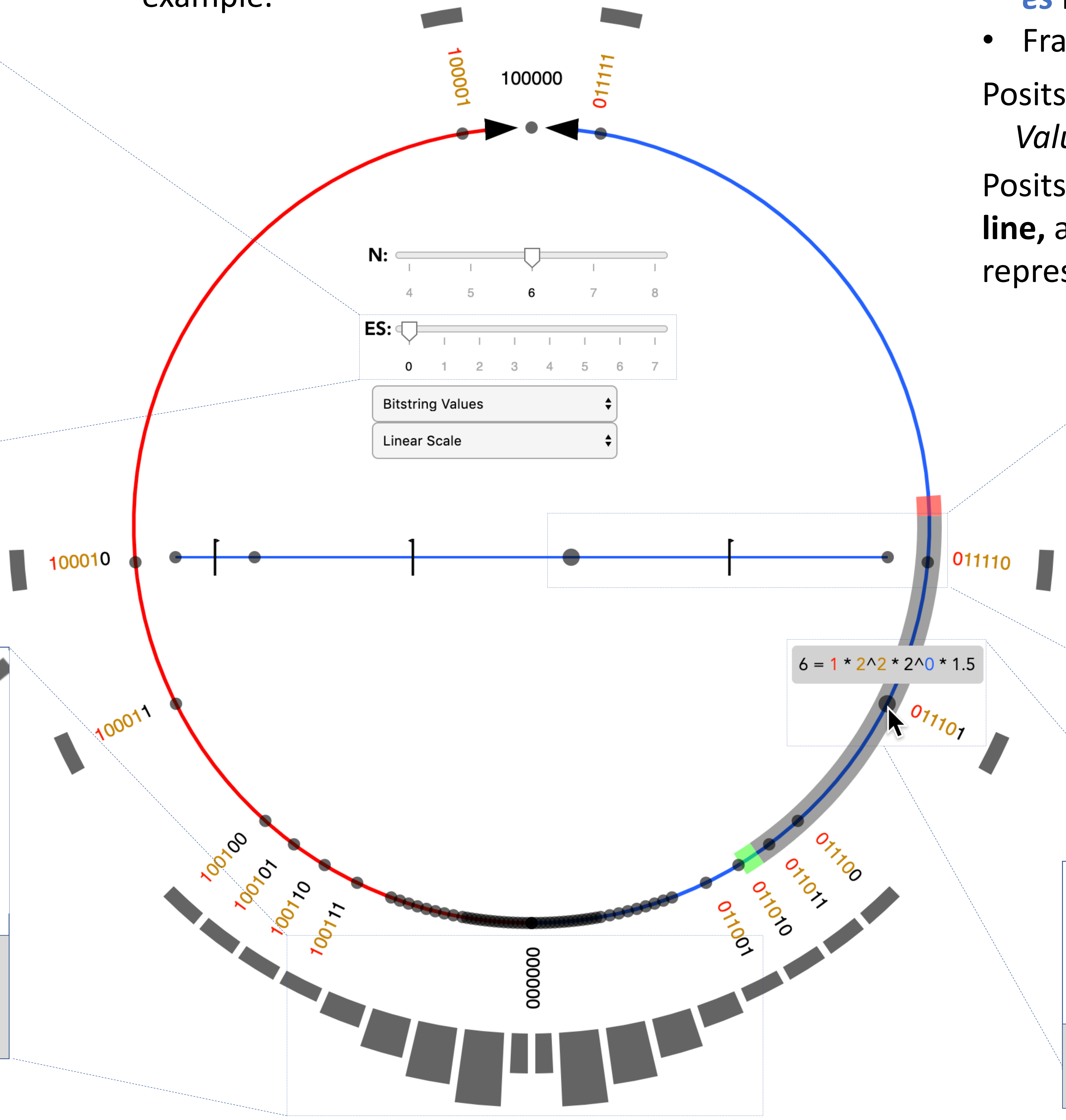
Histogram shows how the datatype's values are distributed, both in linear and logarithmic scales. A tooltip gives details-on-demand.

Problem

Programmers lack understanding of floating-point representations in general, let alone new representations like the posit. These representations have many common stumbling points, such as

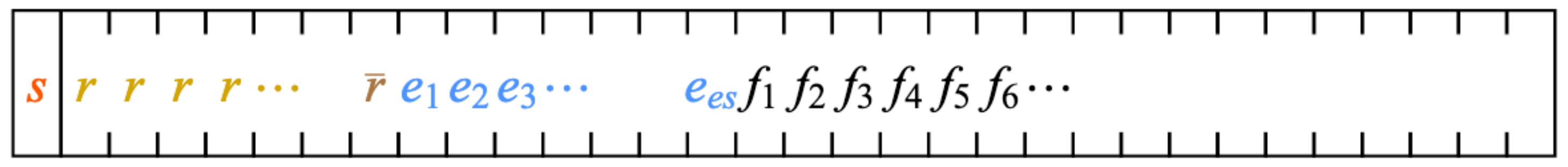
- what datatype parameters mean,
- how the datatype values are distributed, and
- how rounding is performed.

To address this problem, we have built a visualization which informs programmers about floating point representations, using the posit as our primary example.



Posits

The posit format has two parameters: ***n***, the total number of bits, and ***es***, the maximum number of exponent bits. The bitfields of a posit are as follows:

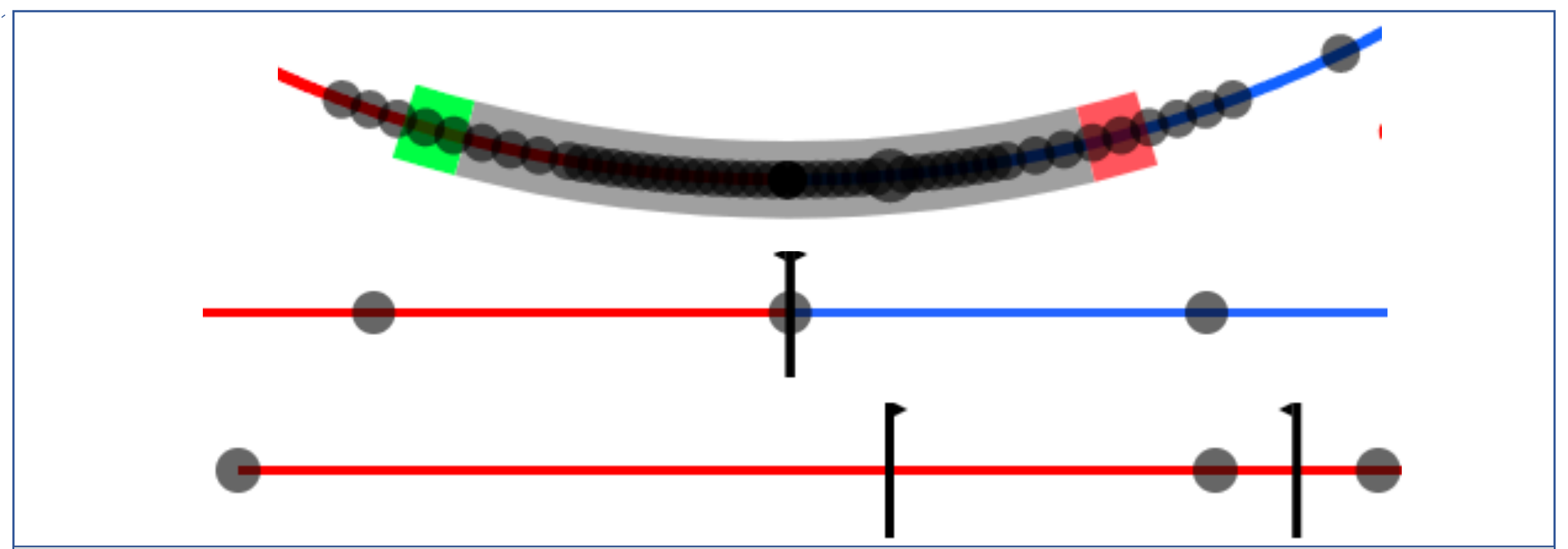

s
$r\ r\ r\ r \cdots$
$\bar{r}\ e_1 e_2 e_3 \cdots$
$e_{es} f_1 f_2 f_3 f_4 f_5 f_6 \cdots$

- Sign: Set to 1 for negative numbers, 0 for non-negative numbers
- Regime: A run of identical bits, potentially terminated by an opposing bit that indicates the end of the run. The run length encodes ***K***
- Exponent: Encode the unsigned integer ***E*****,** max of ***es*** bits
- Fraction: Encodes $F = \frac{frac\_bits}{len(frac\_bits)}$

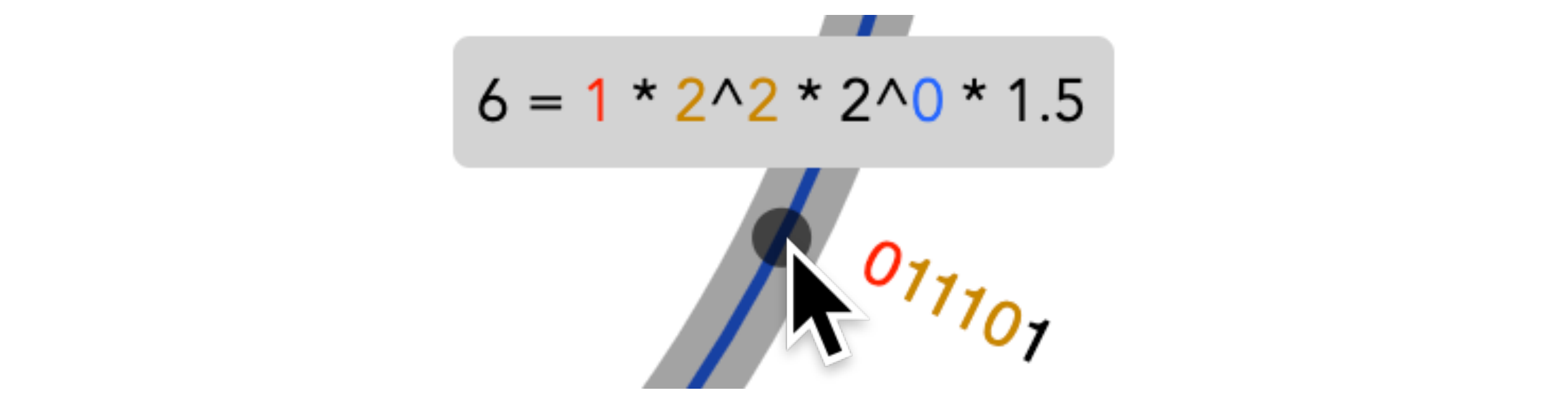
Posits are calculated by the formula below:

$$Value = sign \bullet useed^{K} \bullet 2^{E} \bullet F \qquad (useed = 2^{2^{es}})$$

Posits are **often visualized on a circle, rather than a line,** as negative and positive infinity have the same representation (100000, in this case).



Brush allows for zooming in on specific points. Zoomed-in number line shows the rounding tie-points between posits, and the arrow indicates the direction in which ties are broken.



Tooltip and colored bitstring explain how the posit's value is calculated.