Floating-point Ranges

Why are they so hard?

Can we do better?

Stefan Karpinski

https://JuliaHub.com

Why are float ranges hard?

Consider `0.1:0.2:1.7`

- Let's try generating the values naively

Naive generation?

```
julia> [0.1 + i*0.2 for i=0:8]
9-element Vector{Float64}:
 0.1
 0.30000000000000004
 0.5
 0.7000000000000001
 0.9
 1.1
 1.3000000000000003
 1.5000000000000002
 1.7000000000000002
```

Why this is happening

The source of 99% of confusion about floating-point:

- `0.1` actually means $\frac{3602879701896397}{2^{55}} > \frac{1}{10}$
- `0.2` actually means $\frac{3602879701896397}{2^{54}} > \frac{2}{10}$
- `1.7` actually means $\frac{7656119366529843}{2^{52}} < \frac{17}{10}$

Taken literally, we cannot have `start + n*step == stop`

Face value?

When evaluating `sin(0.1)` we compute

$$\sin \frac{3602879701896397}{2^{55}}$$

- And round back to `Float64`

We don't compute $\sin \frac{1}{10}$

Interpretation required

Sadly, we need to do some amount of guessing here

- Actual start, step and stop values are usually incoherent

Each value has some wiggle room if interpreted as an interval

- Interpret `x` as the set of values in $\mathbb{R}$ that round to `x`

Another example

```
julia> r = -3e25:1e25:4e25
-3.0e25:1.0e25:3.0000000000000001e25

julia> collect(r)
7-element Vector{Float64}:
 -3.0e25
 -1.9999999999999998e25
 -9.999999999999999e24
  4.294967296e9          # <= should be zero
  1.0000000000000003e25
  2.0e25
  3.0000000000000001e25
```

Another example

Range: `-3e25:1e25:4e25`

- `3e25` $= 30000000000000000570425344 \pm 2^{32}$
- `1e25` $= 10000000000000000905969664 \pm 2^{31}$
- `4e25` $= 40000000000000003623878656 \pm 2^{33}$

What we do now

Guessing what `a:s:b` means (today):

- Compute length `n = round((b-a)/s)`
- Rationalize `a` to `n_a//d_a`
- Rationalize `b` to `n_b//d_b`
- Compute `n_s//d_s = (n_b//d_b - n_a//d_a)/n`
- Check if `float(n_s//d_s) == s`

Otherwise fall back to literal (bad) interpretation

Let's formalize things

We'll view float inputs as intervals:

- `a:s:b` $\to (A, S, B) \subseteq \mathbb{R}^3$
  - where $A = [A^-, A^+] \subseteq \mathbb{R}$
  - where $S = [S^-, S^+] \subseteq \mathbb{R}$
  - where $B = [B^-, B^+] \subseteq \mathbb{R}$

Definitions

An *rational interpretation* of a range:

$$(\alpha, \beta, \sigma) \in (A \times B \times S) \cap \mathbb{Q}^3$$

$$\alpha + n\sigma = \beta \text{ for some } n \in \mathbb{Z}$$

Refer to n as the length

- even though ranges iterates $n + 1$ values
- nicer value: `length(-1e6:1e6) == 2e6 + 1`

Some key properties

The *grid unit*:

$$\gamma = \gcd(\alpha, \beta, \sigma) \in \mathbb{Q}^+$$

Define *grid ratios* as:

$$a = \frac{\alpha}{\gamma} \in \mathbb{Z} \qquad b = \frac{\beta}{\gamma} \in \mathbb{Z} \qquad s = \frac{\sigma}{\gamma} \in \mathbb{Z}$$

Note $\gcd(a, b, s) = 1$.

Picking interpretations

There are infinite interpretations for any feasible range

- How do we pick a good interpretation?
- This is the whole problem

Needs to be practical to compute

Simplest grid unit?

Each interpretation corresponds to a grid unit, γ

- Pick interpretation with simplest γ?
    - minimize numerator and denominator size

Two issues:

- Hard to compute
- Tends to violate invariants...

Transformations

Some transformations of range specifications:

- Scaling: $(A, B, S) \mapsto (cA, cB, cS)$ for $c \in \mathbb{R}$
- Translation: $(A, B, S) \mapsto (A + t, B + t, S)$ for $t \in \mathbb{R}$
- Binary refinement: $(A, B, S) \mapsto (A, B, S/2)$

Invariants

Ideally, range selection should have these invariants:

- Length: scaling & translation
  - `length(c*a:c*s:c*b) = length(a+t:s:b+t) = length(a:s:b)`
- Ratios: scaling & binary refinement
  - `ratios(c*a:c*s:c*b) = ratios(a:s/2:b) = ratios(a:s:b)`
- We also want this to hold:
  - `length(a:s/2:b) = 2*length(a:s:b)`

Invariants *vs* simplest rationals

It's well-known how to find the simplest rational in an interval

- This is how the `rationalize` function works
- Also how the current range interpretation works

Does not commute with scaling or translation

- `simplest(c*I) ≠ c*simplest(I)`
- `simplest(I+t) ≠ simplest(I)+t`

Invariants *vs* simplest grid unit

Finding simplest grid unit entails

- Finding simplest common denominator in several intervals
  - (least generator of numerical semigroup intersection)

I actually have an algorithm for this

- Very hard to compute efficiently
- Violates invariants badly

The Algorithm

1. If $0 \in S$, range is infeasible
2. If $S^+ < 0$ swap signs
3. Compute N:

$$N^- = \left\lceil \frac{B^- - A^+}{S^+} \right\rceil \qquad N^+ = \left\lfloor \frac{B^+ - A^-}{S^-} \right\rfloor$$

4. If N is empty, range is infeasible

The Algorithm

5. Find $n \in N$ with maximal trailing zeros
6. Let $p = z(n)$ and $(\bar{n}, \bar{A}, \bar{B}) = (n, A, B) \cdot / \ 2^p$
7. Compute $\bar{Q}$:

$$\bar{Q}^- = \max\{\frac{\bar{A}^-}{S^+}, \frac{\bar{B}^-}{S^+} - \bar{n}\}$$

$$\bar{Q}^+ = \min\{\frac{\bar{A}^+}{S^-}, \frac{\bar{B}^+}{S^-} - \bar{n}\}$$

The Algorithm

8. If $\lceil \bar{Q}^- \rceil \le \lfloor \bar{Q}^+ \rfloor$
   - Find $\bar{a} \in \bar{Q}$ with the maximal trailing zeros; let $\bar{s} = 1$
9. Otherwise
   - Let $F = \bar{Q} - \lfloor \bar{Q}^- \rfloor \subseteq (0, 1)$
     - Find $\bar{a}/\bar{s} \in F$ the simplest fraction
10. Let $\bar{b} = \bar{a} + \bar{n}\bar{s}$

The Algorithm

11. Compute $a/s = \bar{a}2^p/\bar{s}$ and $b/s = \bar{b}2^p/\bar{s}$ as reduced fractions:
    - $a = \bar{a} \ll \max(0, p - z(\bar{s}))$
    - $b = \bar{b} \ll \max(0, p - z(\bar{s}))$
    - $s = \bar{s} \gg \max(0, z(\bar{s}) - p)$
12. Let $G = (A/a) \cap (B/b) \cap (S/s)$
13. Find $\gamma \in G$ the simplest fraction