Abstract Semantics Used in `fp_ai.py`

Evelyn Ma

April 2025

1 Concrete Values

I track all IEEE-754 single-precision numbers plus the specials $\{+\infty, -\infty, \text{NaN}\}$. Sub-normal (denormal) numbers are treated as ordinary finite numbers, so they live safely inside our *finite interval*. All arithmetic uses the C round-to-nearest-even (RNE) rule.

2 Abstract Value

Element

$$x = \langle lo, hi, \ \mathsf{InfFlag}, \ \mathsf{NaNFlag} \rangle$$

- lo, hi – smallest and largest *finite* values still possible
- $\mathsf{InfFlag}$ – 1 iff $\pm\infty$ is still possible
- $\mathsf{NaNFlag}$ – 1 iff NaN is still possible

Special elements

$$\top = \langle -\text{FLT\_MAX}, \ \text{FLT\_MAX}, \ 0, 0 \rangle, \qquad \bot = \langle 0, 0, 0, 0 \rangle_{\text{empty}}$$

($\bot$ denotes the empty set, not "exactly 0".)

Order

$$x \sqsubseteq y \iff x.lo \geq y.lo \ \wedge \ x.hi \leq y.hi \ \wedge \ (x.\mathsf{InfFlag} \Rightarrow y.\mathsf{InfFlag}) \ \wedge \ (x.\mathsf{NaNFlag} \Rightarrow y.\mathsf{NaNFlag}).$$

Hence every element lies below $\top$, and $\bot$ lies below every element.

Join

$$x \sqcup y = \langle \min(x.lo, y.lo), \ \max(x.hi, y.hi), \ x.\mathsf{InfFlag} \vee y.\mathsf{InfFlag}, \ x.\mathsf{NaNFlag} \vee y.\mathsf{NaNFlag} \rangle.$$

3 Abstract Arithmetic

Op.	Abstract transfer (new element)
$+$	$lo = a.lo + b.lo, \quad hi = a.hi + b.hi$; $\mathsf{InfFlag} = a.\mathsf{InfFlag} \vee b.\mathsf{InfFlag} \vee$ overflow of either bound; $\mathsf{NaNFlag} = a.\mathsf{NaNFlag} \vee b.\mathsf{NaNFlag}$.
$-$	treat $a - b$ as $a + (-b)$; flags as above.
$\times$	take the four endpoint products for new bounds; $\mathsf{InfFlag} = a.\mathsf{InfFlag} \vee b.\mathsf{InfFlag} \vee$ overflow of any product; $\mathsf{NaNFlag} = a.\mathsf{NaNFlag} \vee b.\mathsf{NaNFlag} \vee (a.\mathsf{InfFlag} \wedge 0 \in b) \vee (b.\mathsf{InfFlag} \wedge 0 \in a)$.
$\div$	if $0 \in b$ set interval to $[-\mathrm{FLT\_MAX}, \mathrm{FLT\_MAX}]$ and $\mathsf{InfFlag} = 1$; if $b.\mathsf{InfFlag} = 1$ and $0 \notin b$, include 0 in the interval, but *do not* add new ∞; $\mathsf{NaNFlag} = a.\mathsf{NaNFlag} \vee b.\mathsf{NaNFlag} \vee (0/0) \vee (\infty/\infty)$.

Whenever a bound $< -\mathrm{FLT\_MAX}$ or $> +\mathrm{FLT\_MAX}$, we set $\mathsf{InfFlag} = 1$ and clip that bound independently to the nearest limit.

4 Condition Narrowing

- Atom `x >0`: true $\Rightarrow$ $x.lo := \max(x.lo, \varepsilon)$ (where ε is a small positive constant); false $\Rightarrow$ $x.hi := \min(x.hi, 0)$.
- Atom `x == 0`: true $\Rightarrow$ $x.lo := x.hi := 0$; false narrows away 0 on the appropriate side.
- For `&&` and `||`, each feasible branch keeps its own copy of the environment; environments are merged variable-wise using the lattice join after the `if`.
- Only *shallow* refinement is performed; variables are narrowed independently (no cross-variable constraints).

5 Soundness

For every expression e

$$\alpha\big(e_{\text{concrete}}\big) \ \sqsubseteq \ e_{\text{abstract}}\big(\alpha(\cdot)\big).$$

Proof sketch. Structural induction on the syntax tree:

1. *Base cases*: constants map exactly.
2. *Inductive step*: Table 3 gives, for every operator, an abstract element that is an upper bound of the concretisation of the exact result, hence monotone.
3. *Control flow*: branch environments are joined with the lattice join, which is the least upper bound, so the property is preserved.

Why Inf/NaN are safe. Finite results are covered by the interval. Every concrete $\pm\infty$ maps into an element with $\mathsf{InfFlag} = 1$, and every concrete NaN maps into an element with $\mathsf{NaNFlag} = 1$. Both flags are joined using Boolean $\vee$, the least upper bound in the Boolean lattice, so a "possible Inf/NaN" fact can never be lost through merging or looping. Therefore the abstraction is sound for all concrete outcomes.