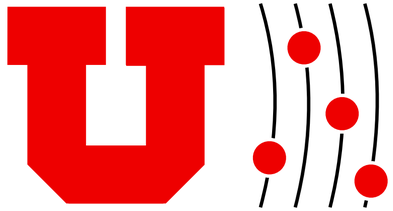
Mixing Condition Numbers and Oracles for Accurate Floating-point Debugging

Bhargav Kulkarni, Pavel Panchekha

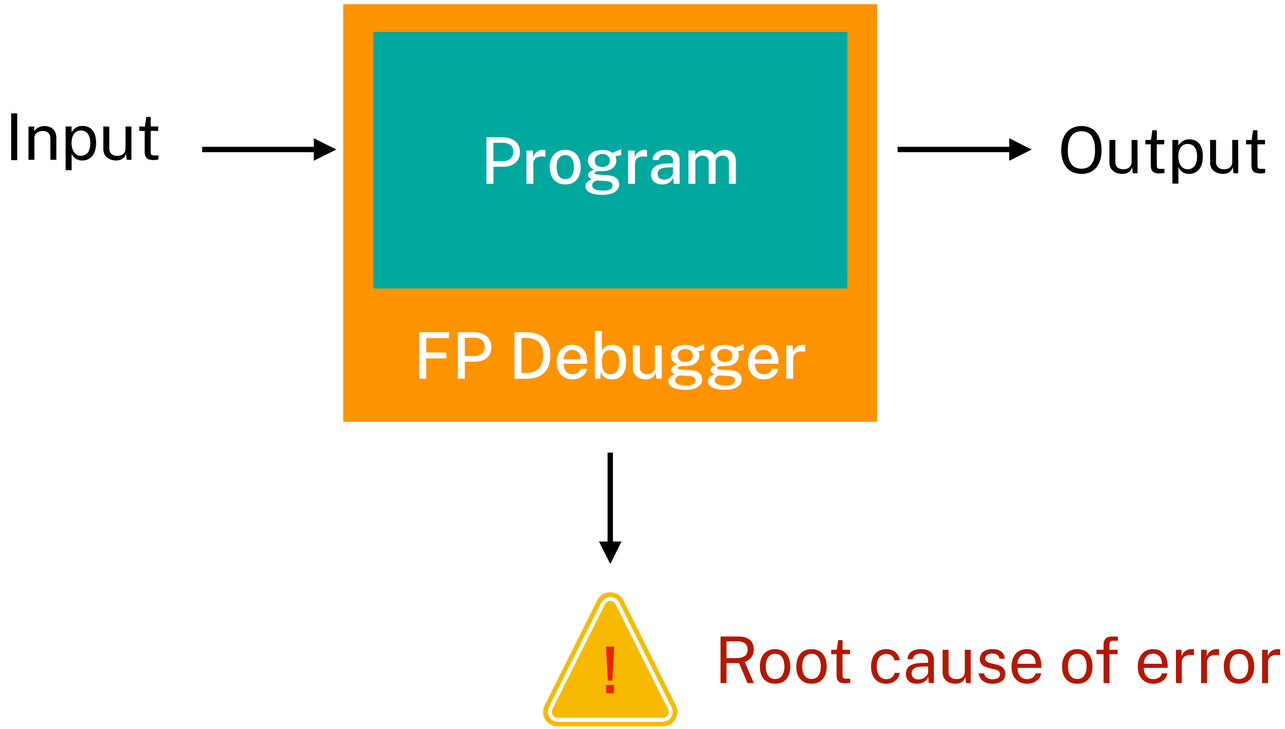
University of Utah



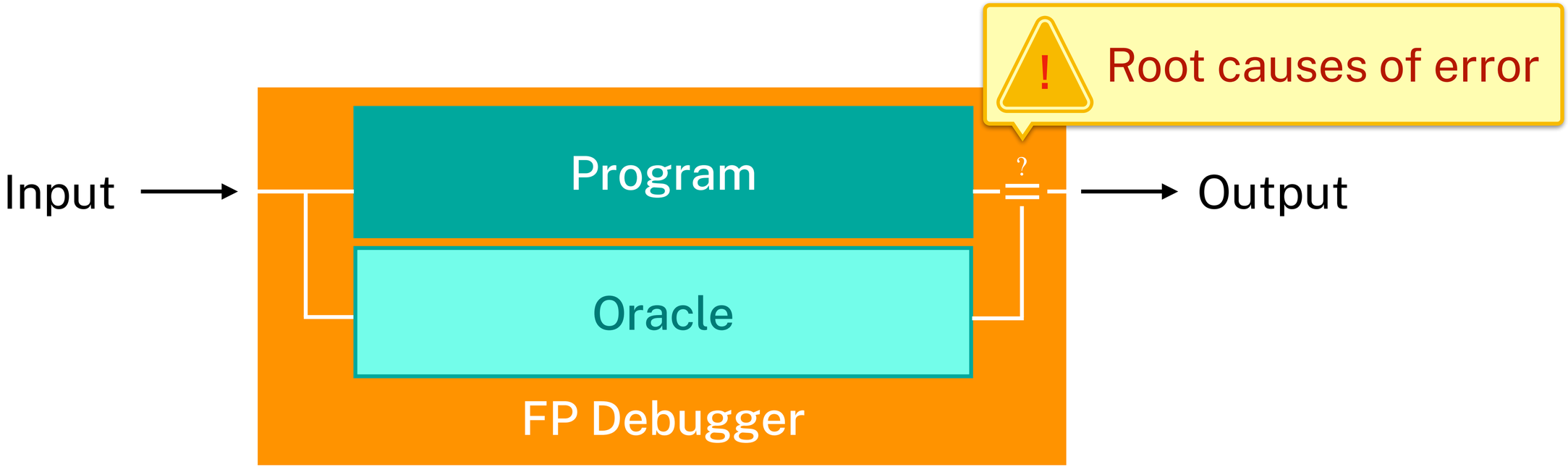
Programming Languages at Utah

Floating-Point Programming is Hard

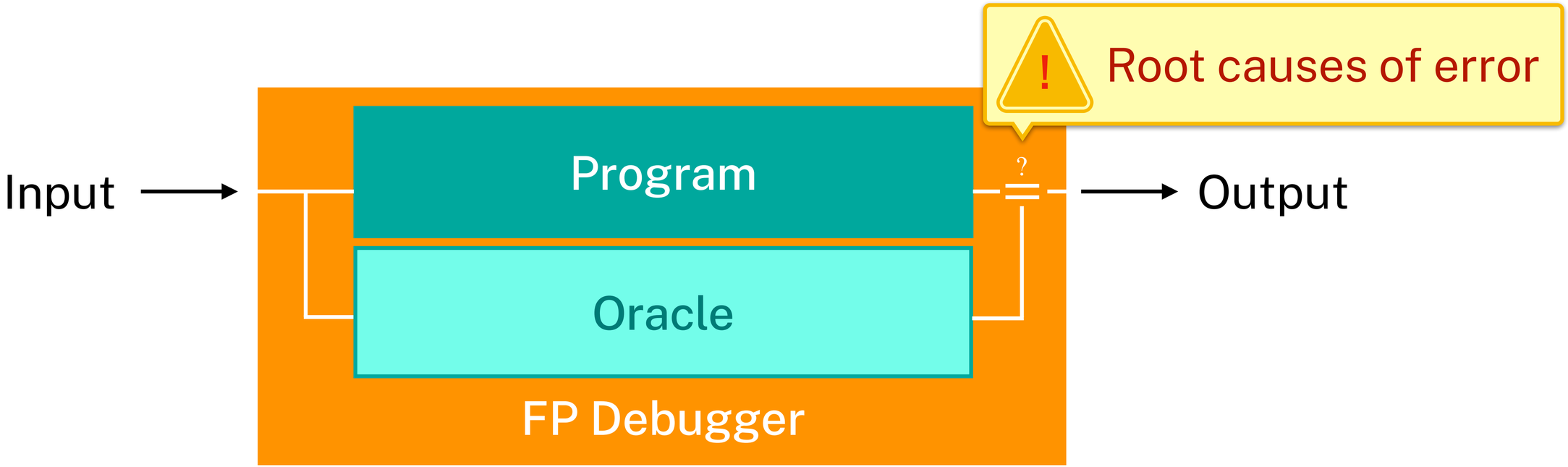
FP Debuggers



FP Debuggers



FP Debuggers



What is the **best oracle** to use in a debugger?

Choosing an Oracle

Choosing an Oracle

Arbitrary-Precision Oracles

FpDebug [PLDI'12]

Herbgrind [PLDI'17]

FPSanitizer [PLDI'20]

Accuracy: High

Overhead: 1000x

Choosing an Oracle

Arbitrary-Precision Oracles

FpDebug [PLDI'12]

Herbgrind [PLDI'17]

FPSanitizer [PLDI'20]

Accuracy: High

Overhead: 1000x

Machine-Float Oracles

EFTSanitizer [OOPSLA'22]

Verrou [SCAN'16]

Shaman [TOMS'23]

Accuracy: Low

Overhead: 10x

Choosing an Oracle

Arbitrary-Precision Oracles

FpDebug [PLDI'12]

Herbgrind [PLDI'17]

FPSanitizer [PLDI'20]

Accuracy: High

Overhead: 1000x

Machine-Float Oracles

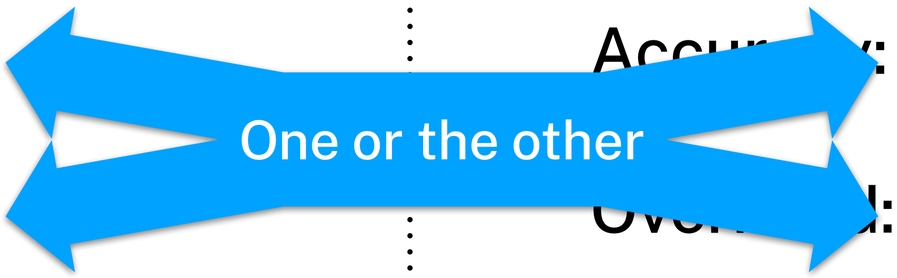
EFTSanitizer [OOPSLA'22]

Verrou [SCAN'16]

Shaman [TOMS'23]

Accu[illegible]: Low

Over[illegible]: 10x



Choosing an Oracle

Arbit

FpDe

Herb

FPSa

Accu

Over

Do we need to choose?

This talk:

Fast and Accurate

floating-point debugger

What a better debugger needs

- Accurately quantify FP error
- Use machine float operations

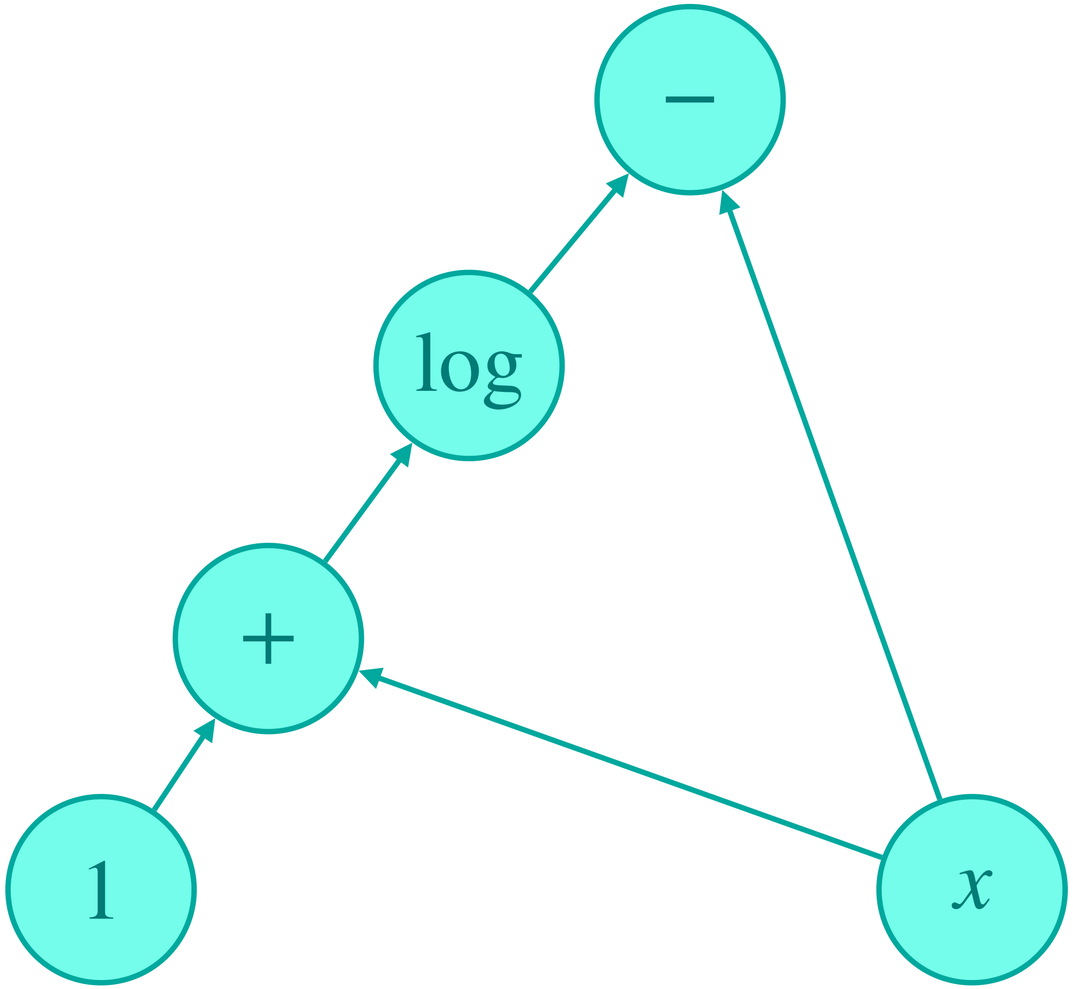
Quantifying Error Statically

A static analysis tool **quantifies error** in a program like:

$$\log(x+1) - x$$

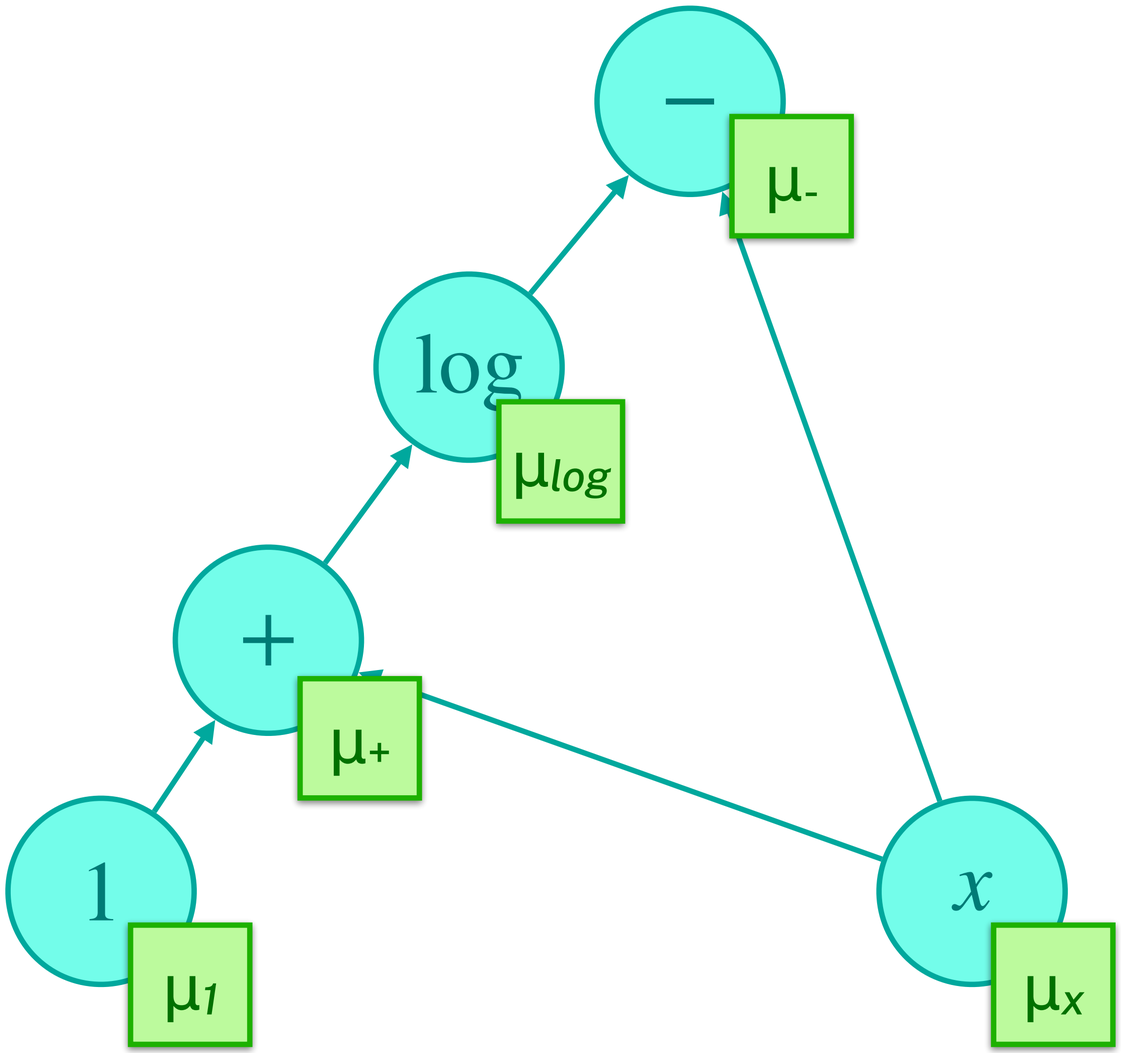
over a range of inputs like $x \in [0,1]$

Quantifying Error Statically



Floating-point operation

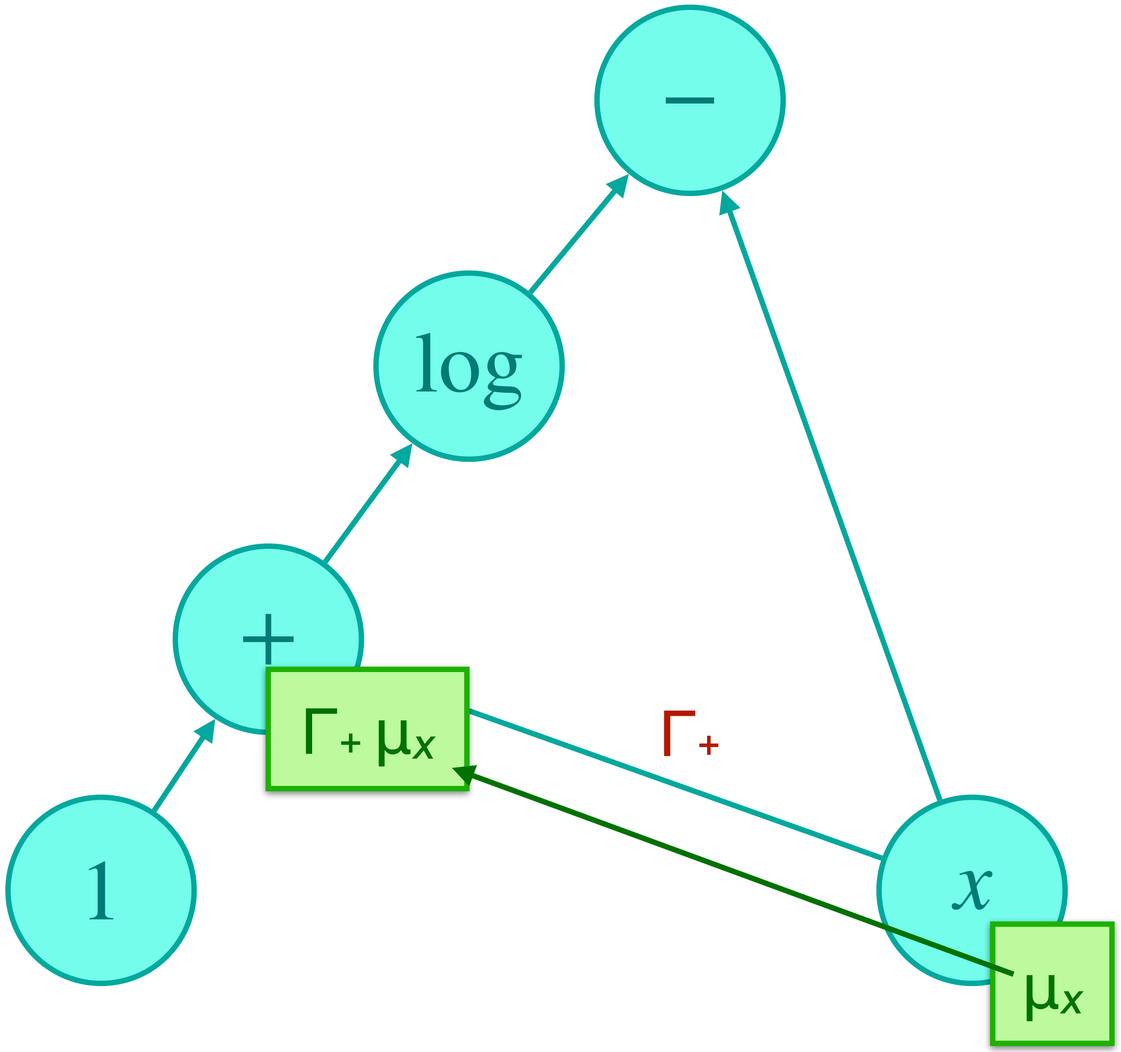
Quantifying Error Statically



Floating-point operation

μ Introduced rounding error

Quantifying Error Statically

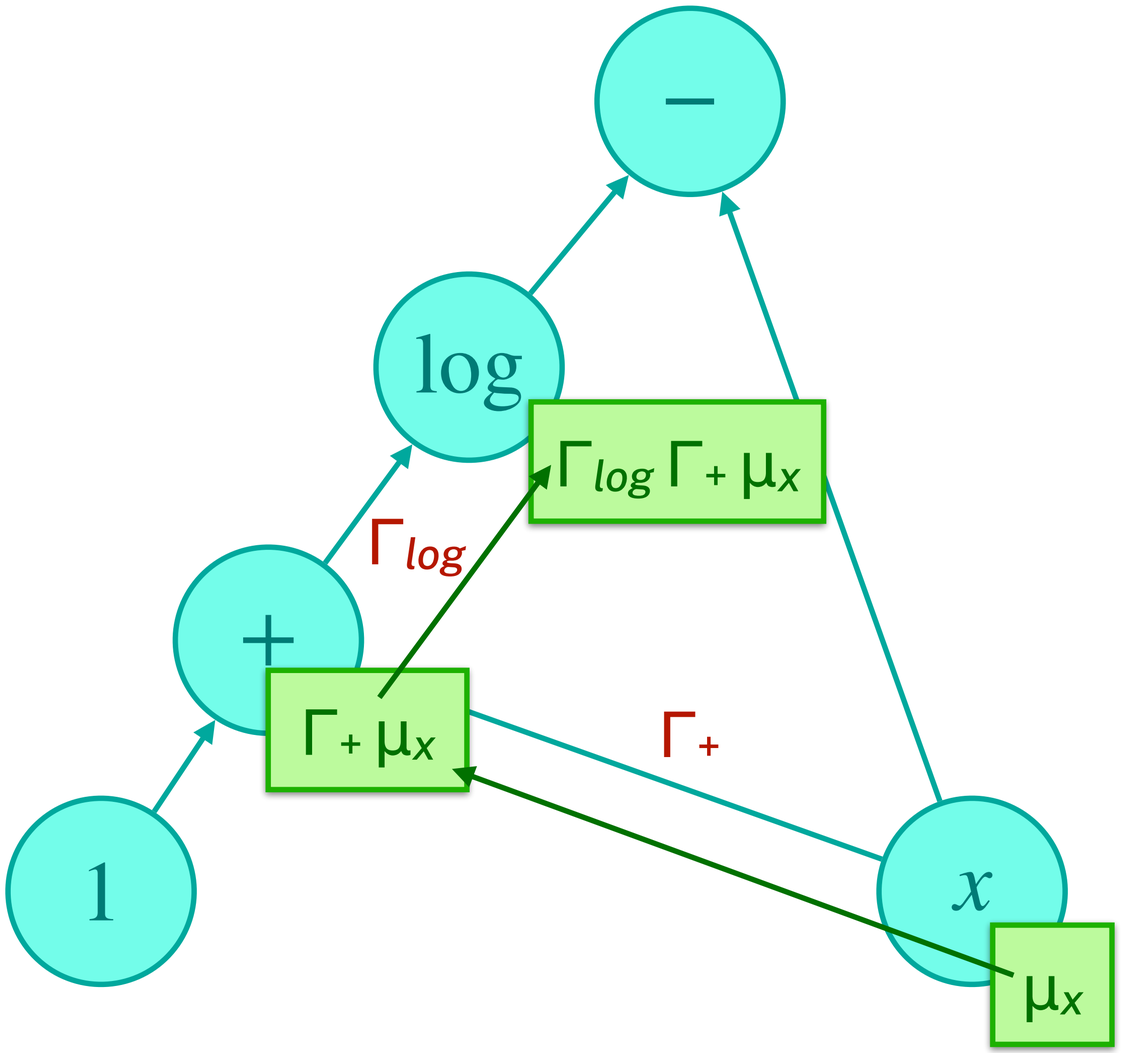


Floating-point operation

μ Introduced rounding error

Γ Condition numbers

Quantifying Error Statically

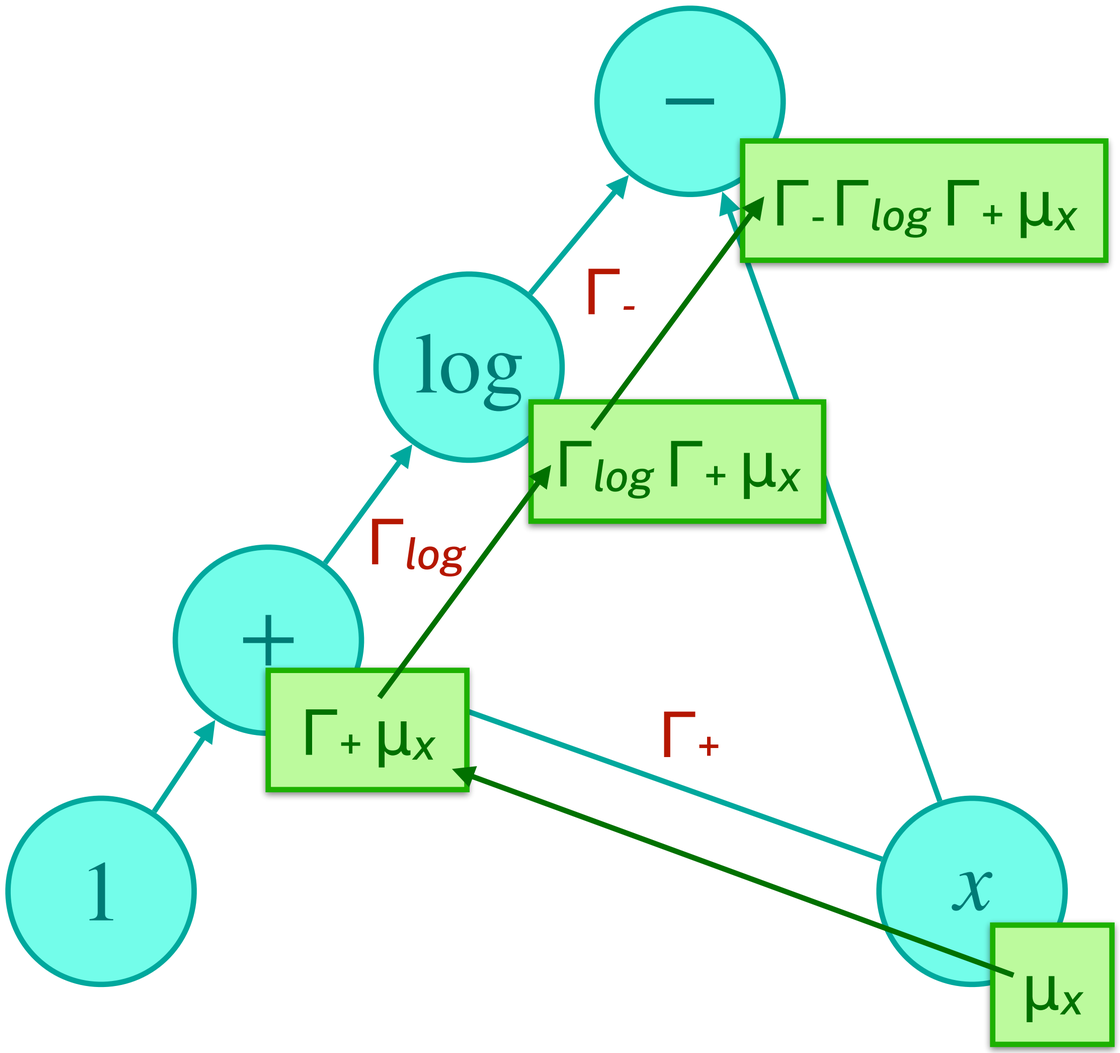


Floating-point operation

μ Introduced rounding error

Γ Condition numbers

Quantifying Error Statically

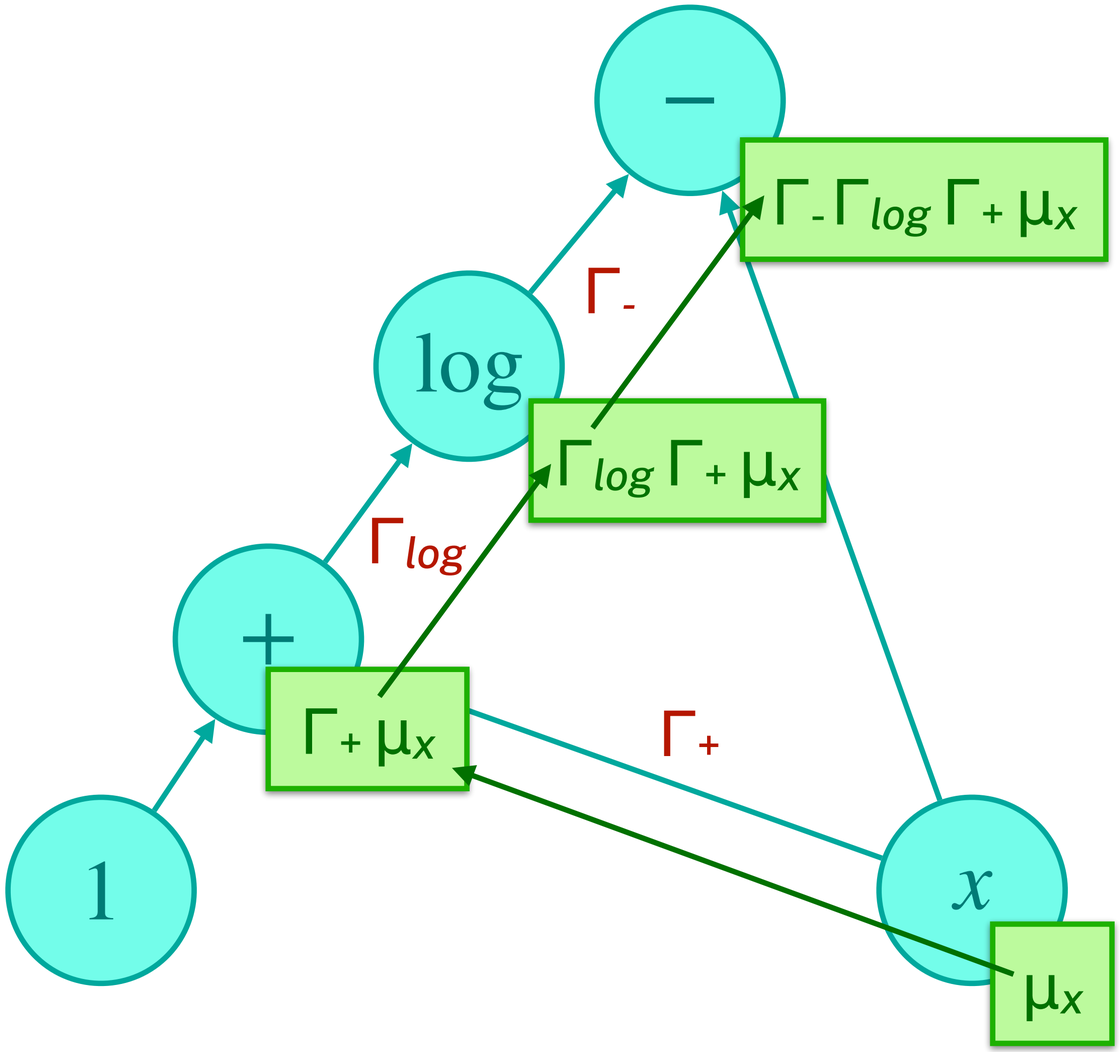


Floating-point operation

μ Introduced rounding error

Γ Condition numbers

Quantifying Error Statically



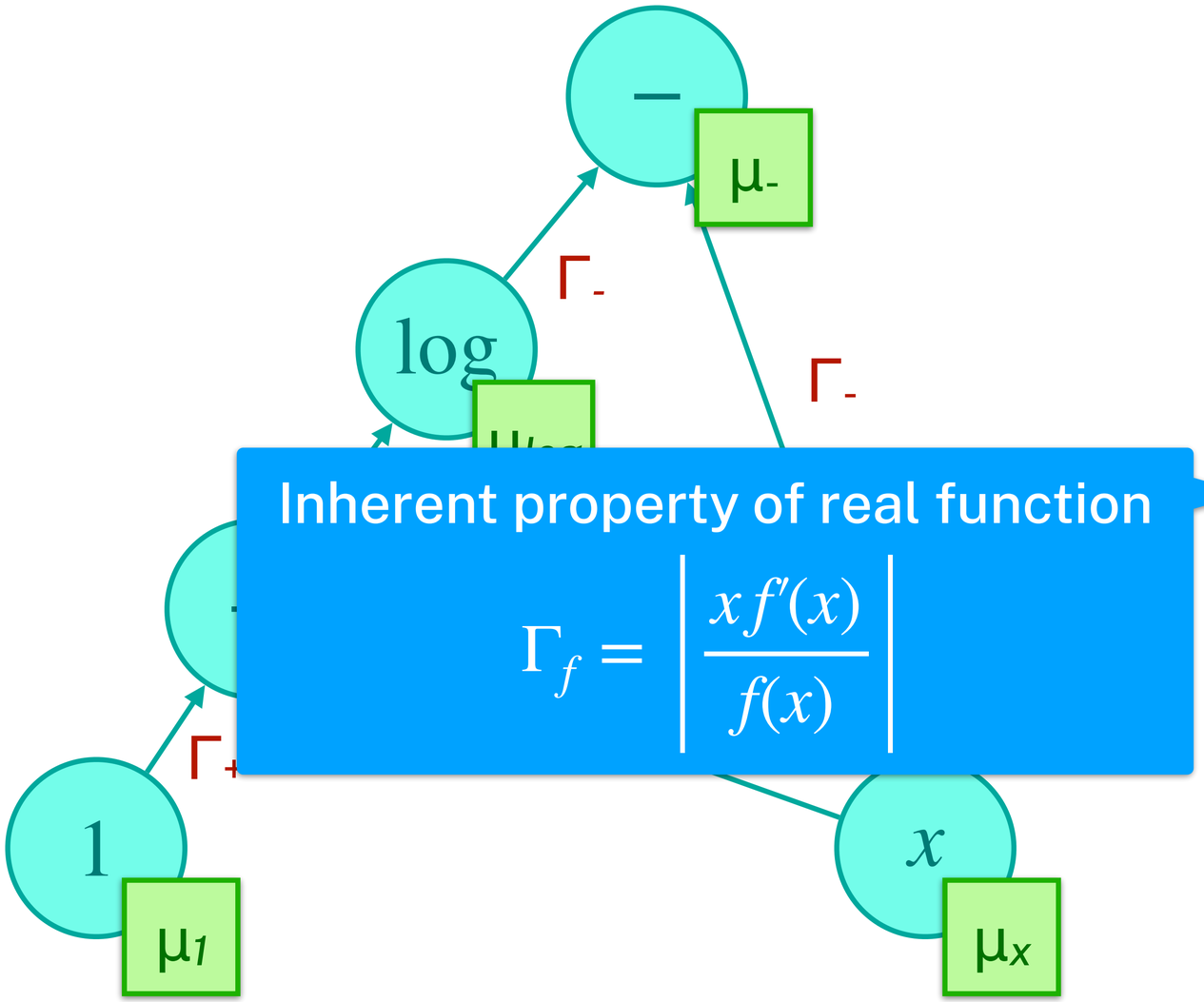
Floating-point operation

μ Introduced rounding error

Γ Condition numbers

☑ Quantifies error

Quantifying Error Statically



Inherent property of real function

$$\Gamma_f = \left| \frac{x f'(x)}{f(x)} \right|$$

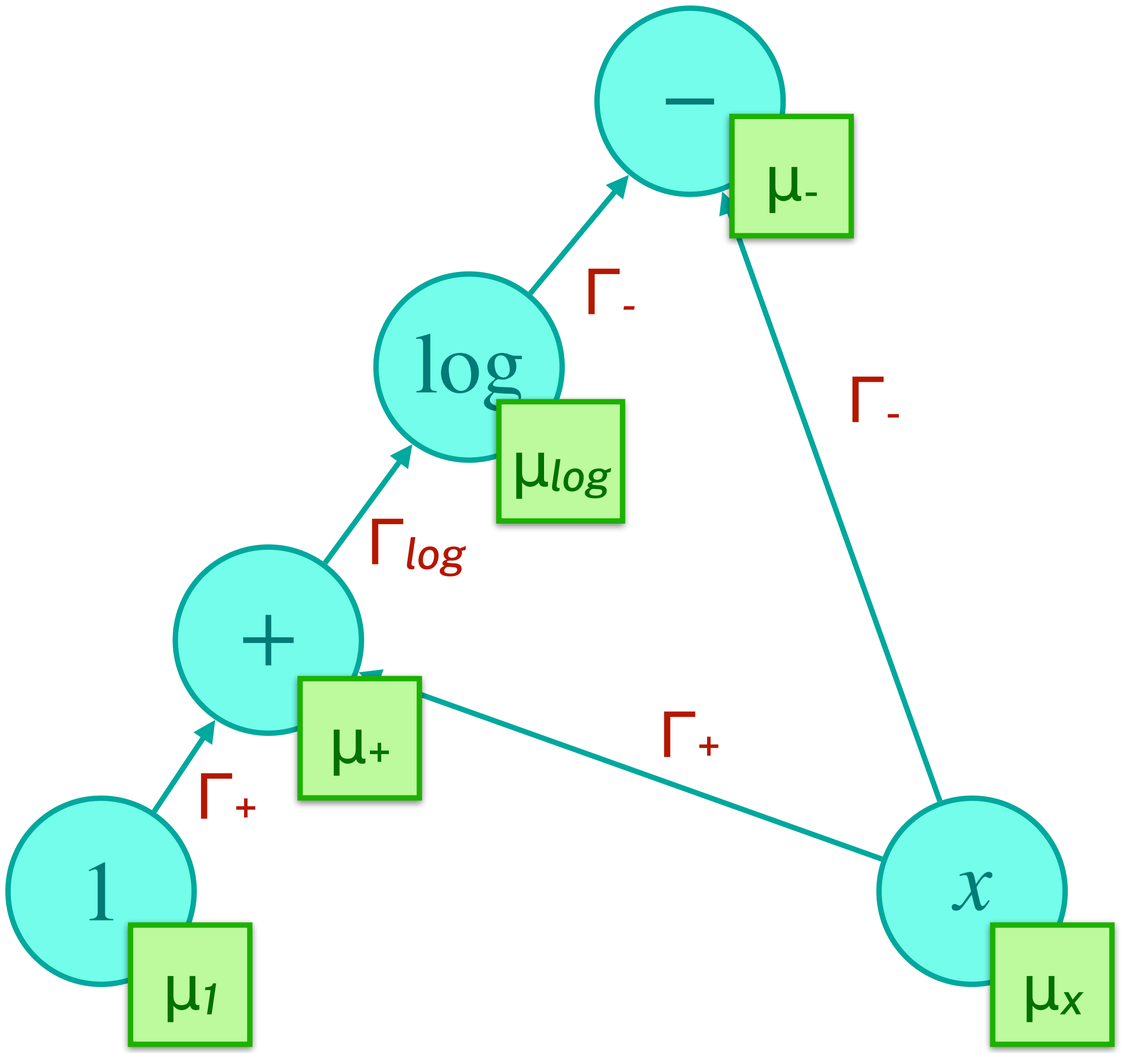
Floating-point operation

μ Introduced rounding error

Γ Condition numbers

☑ Quantifies error

Quantifying Error Statically



Floating-point operation

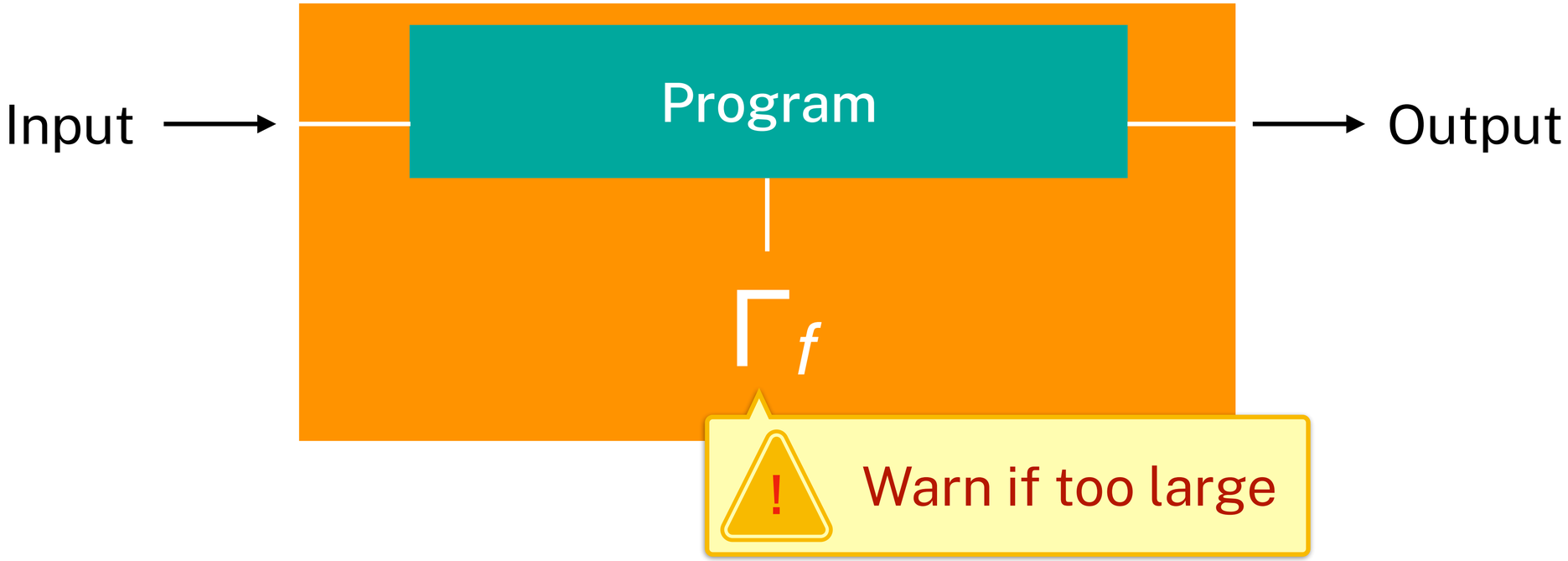
μ Introduced rounding error

Γ Condition numbers

☑ Quantifies error

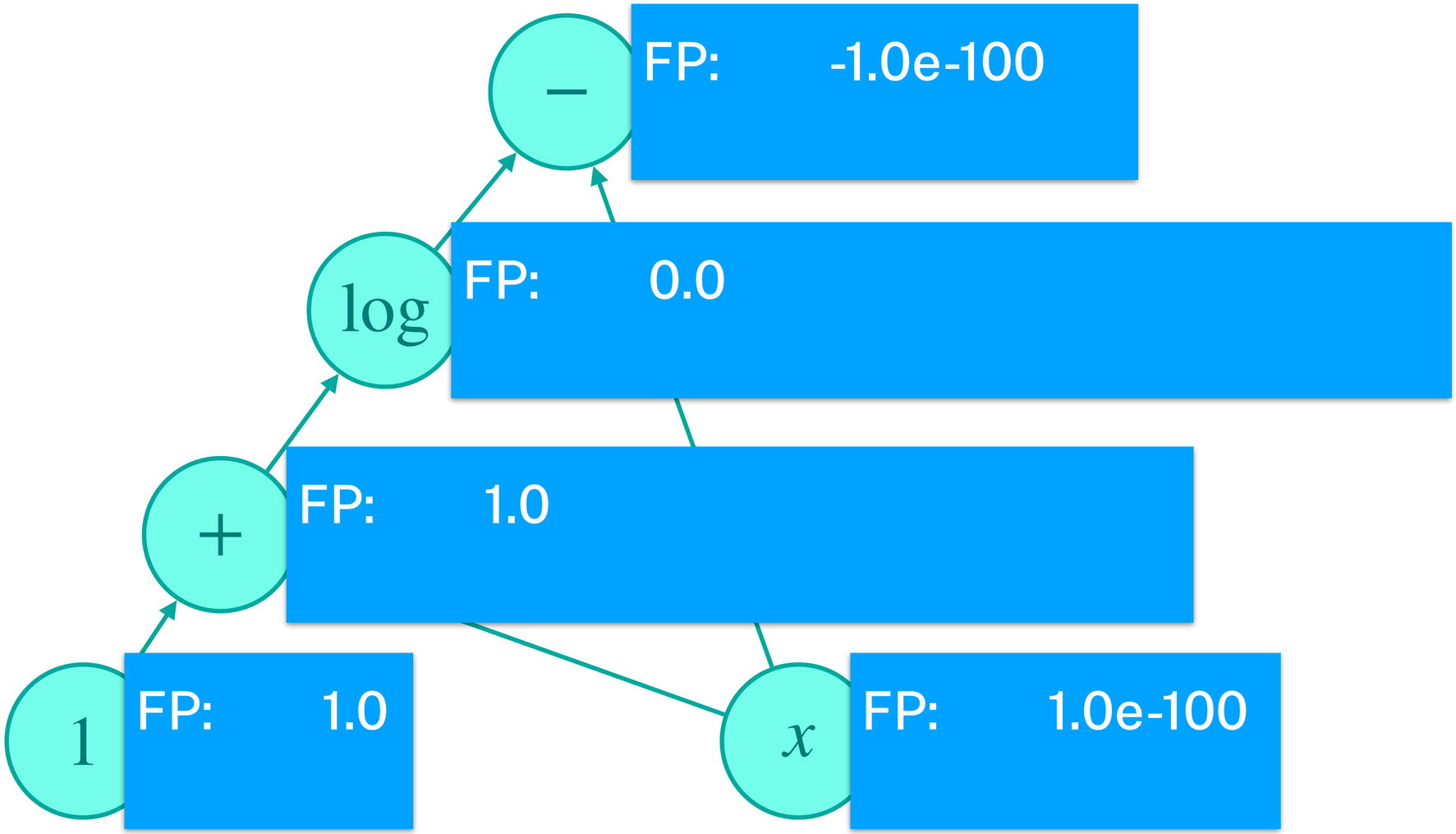
☑ Machine float

Γ-based Debugger

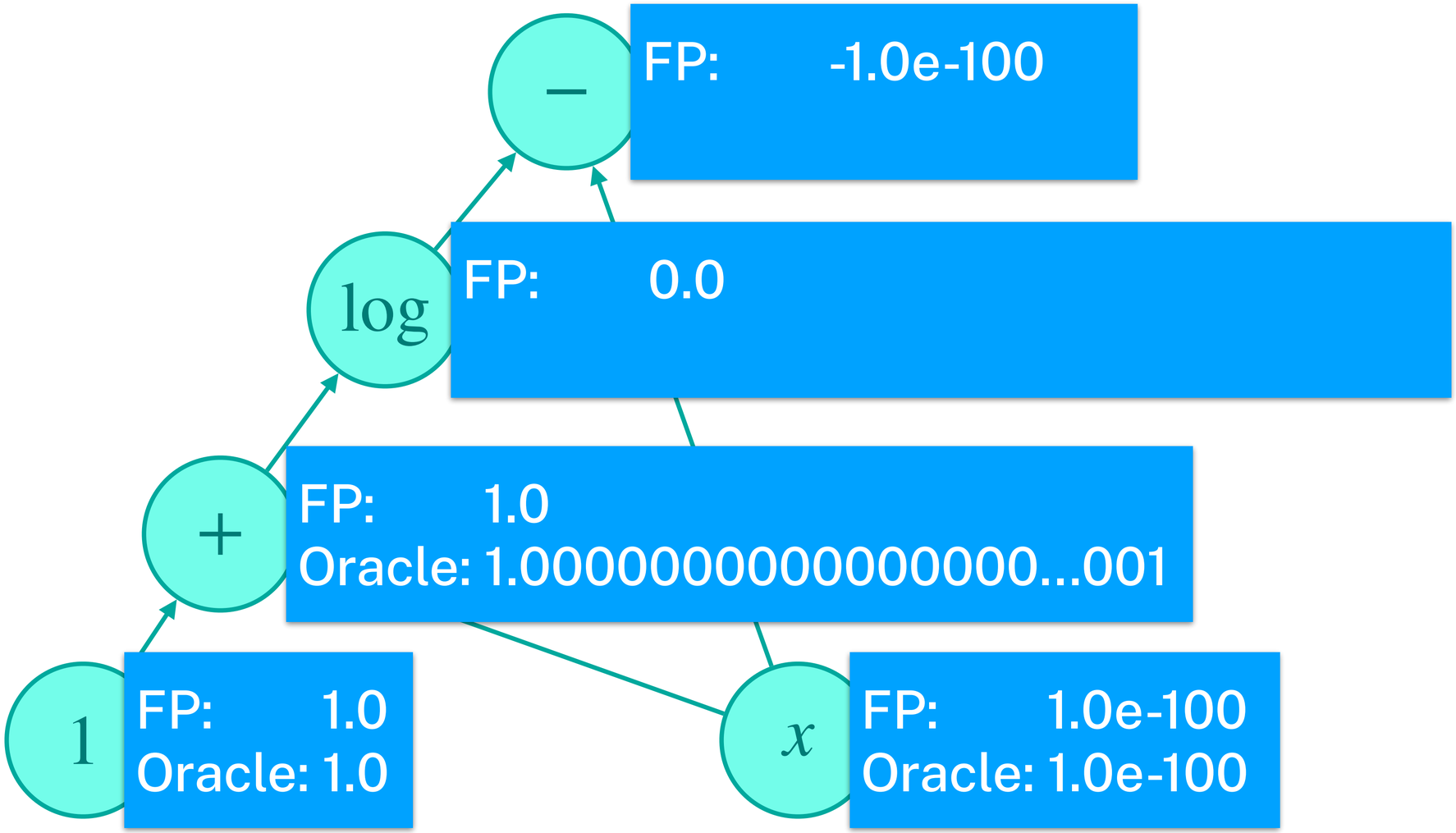


In practice, use a threshold of Γ > 64.

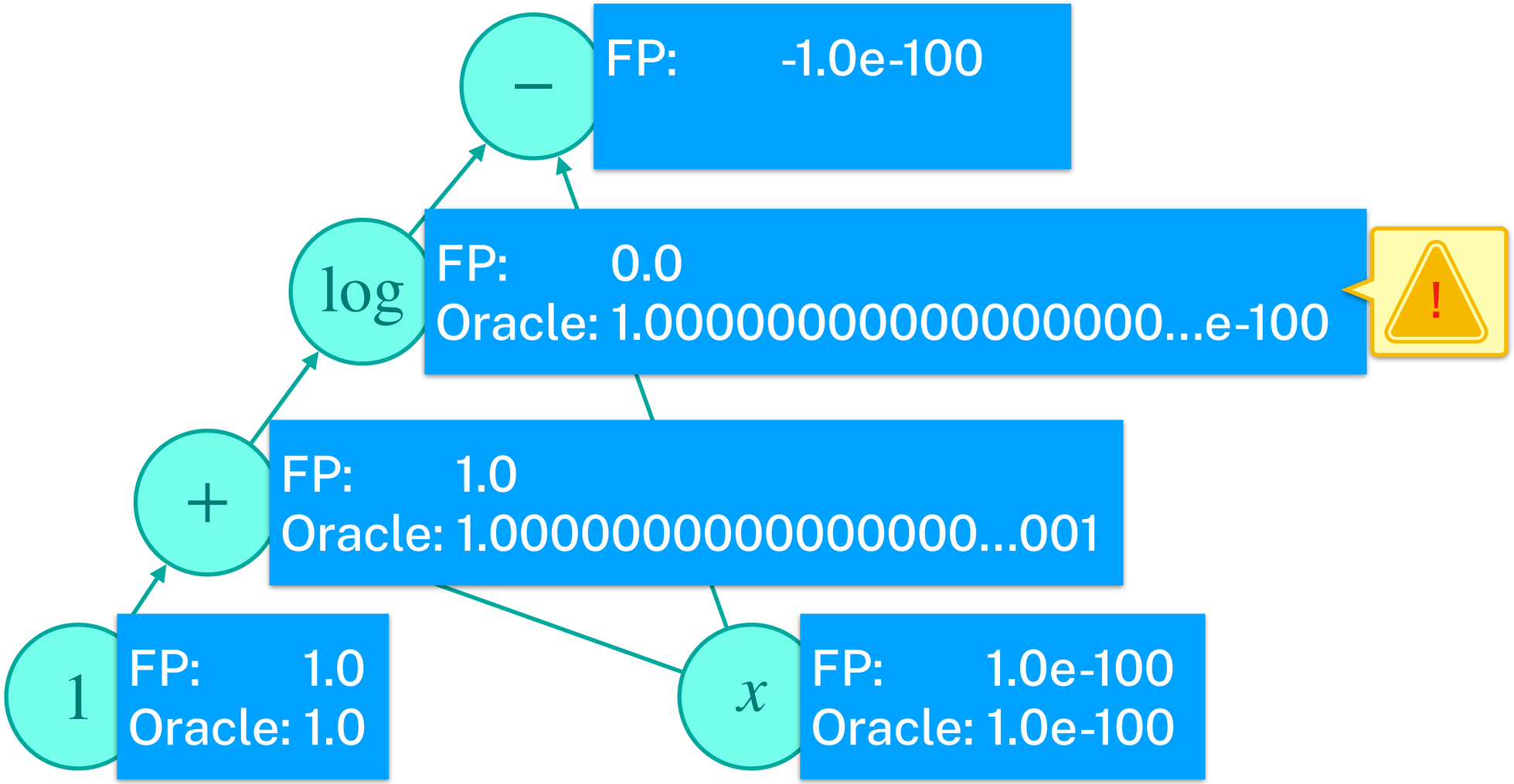
Example, x = 1e-100



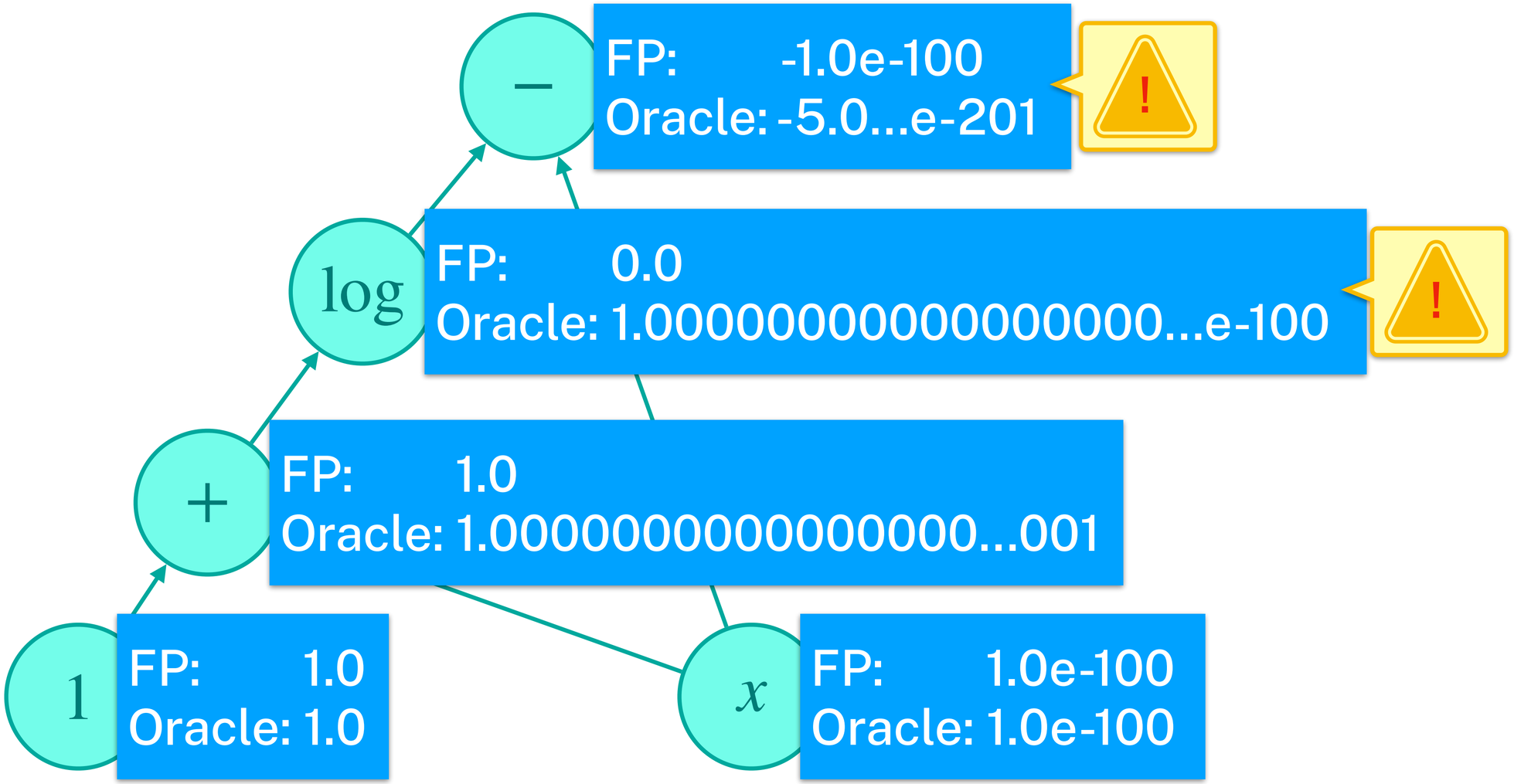
Example, x = 1e-100



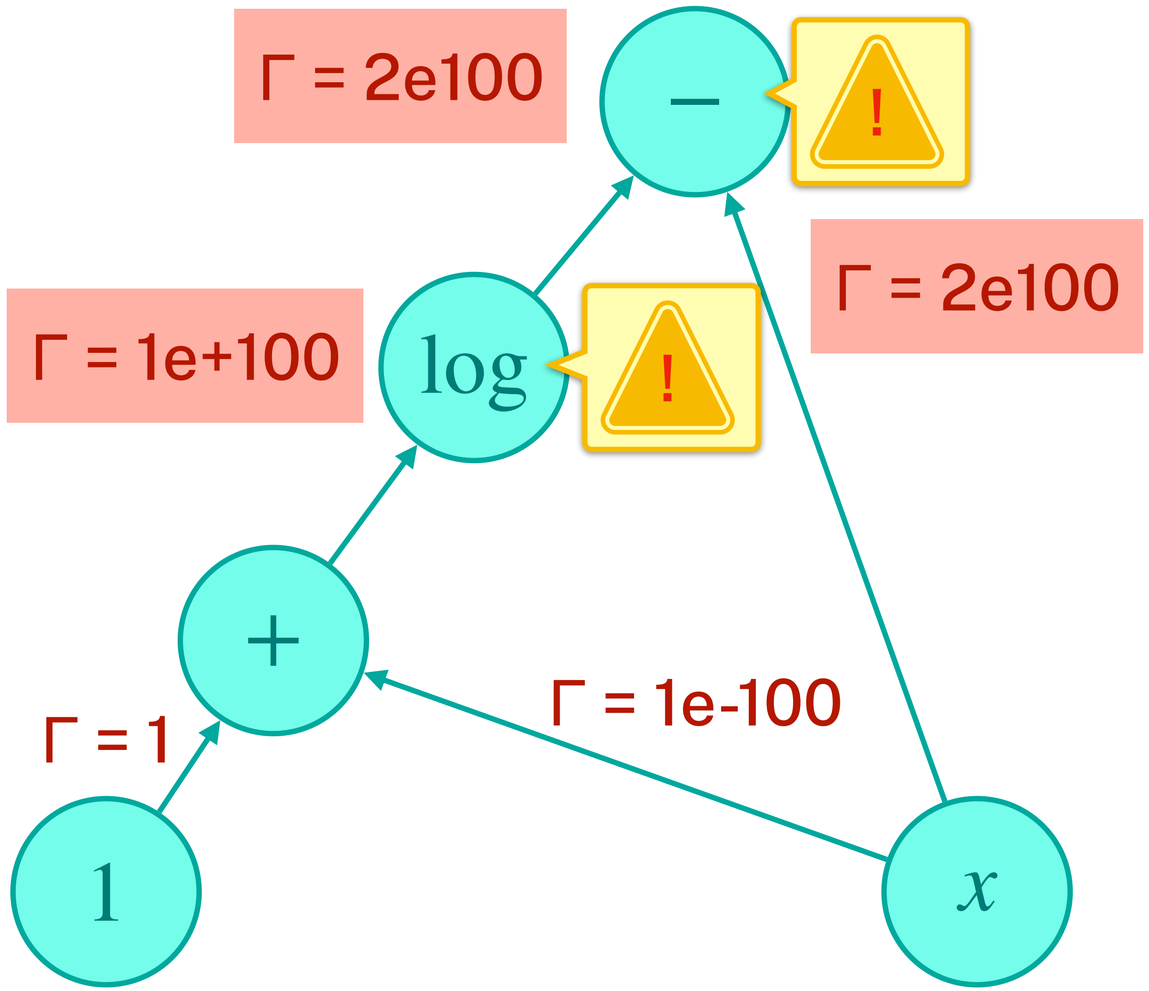
Example, x = 1e-100



Example, x = 1e-100



Example, x = 1e−100



☑ Quantifies error

Identifies **same root cases** as arbitrary-precision oracle

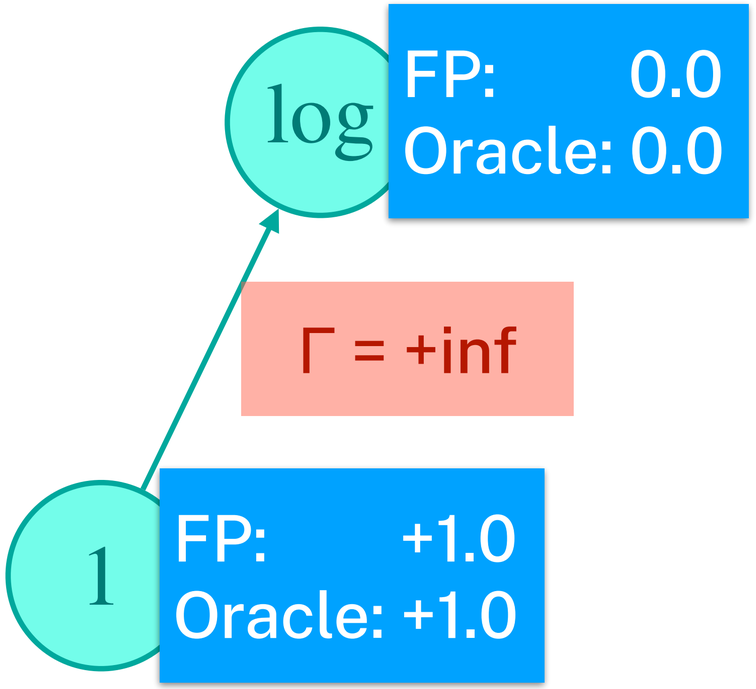
☑ Machine float

At low precision, Γ = +∞

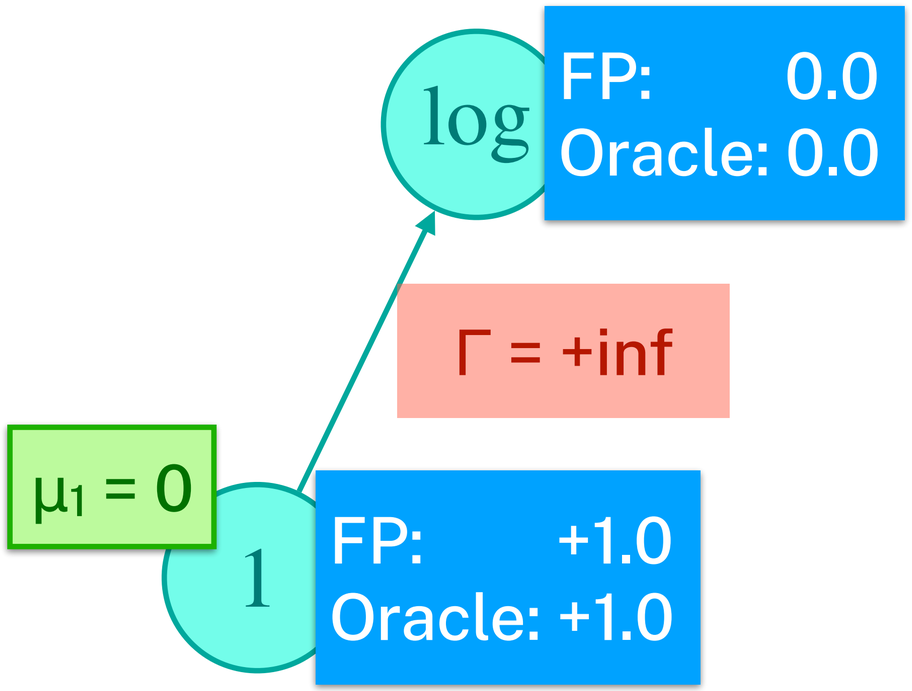
Exact Values

$$\log 1$$

Exact Values

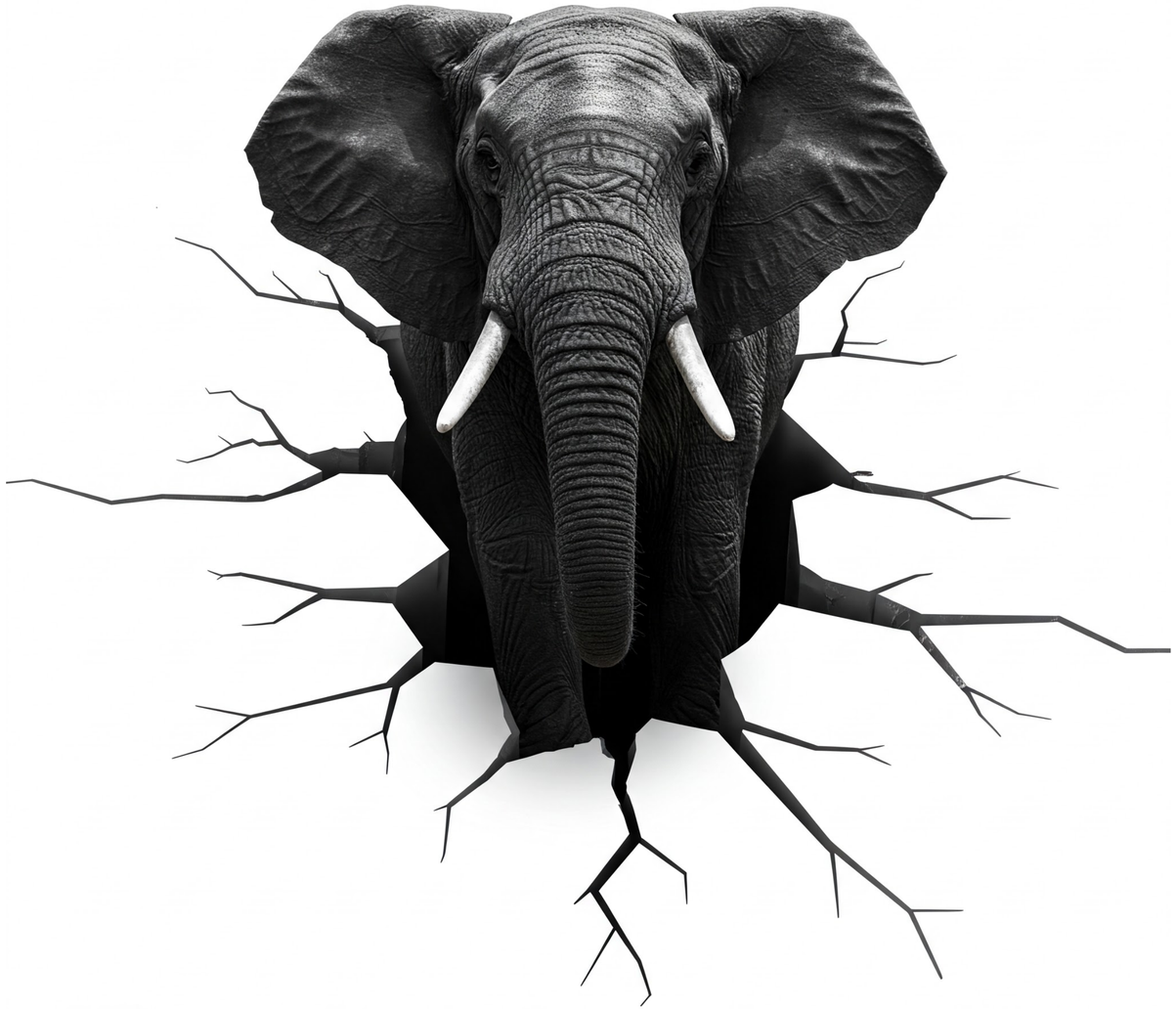


Exact Values



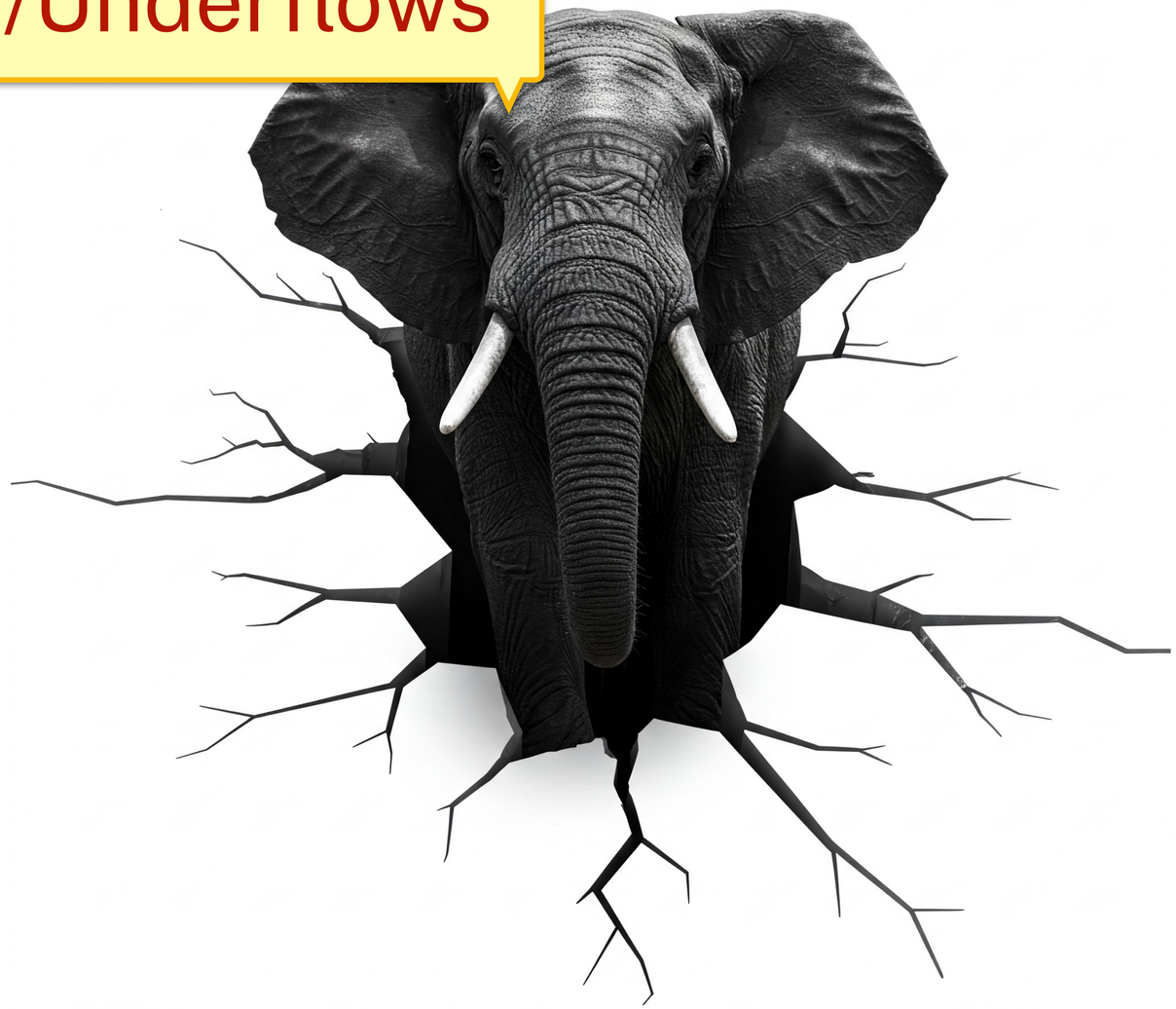
We built a **fast** and **accurate** floating-point debugger!

We built a **fast** and **accurate** floating-point debugger!



We built a **fast** and **accurate** floating-point debugger!

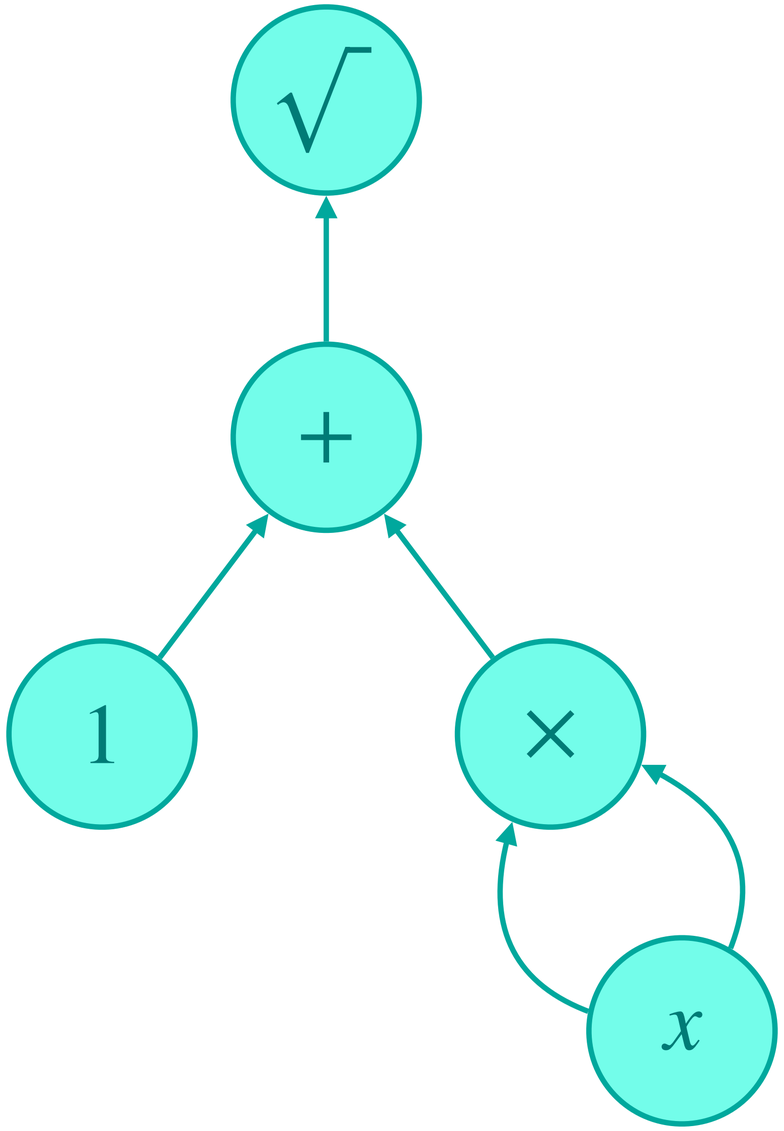
Over/Underflows



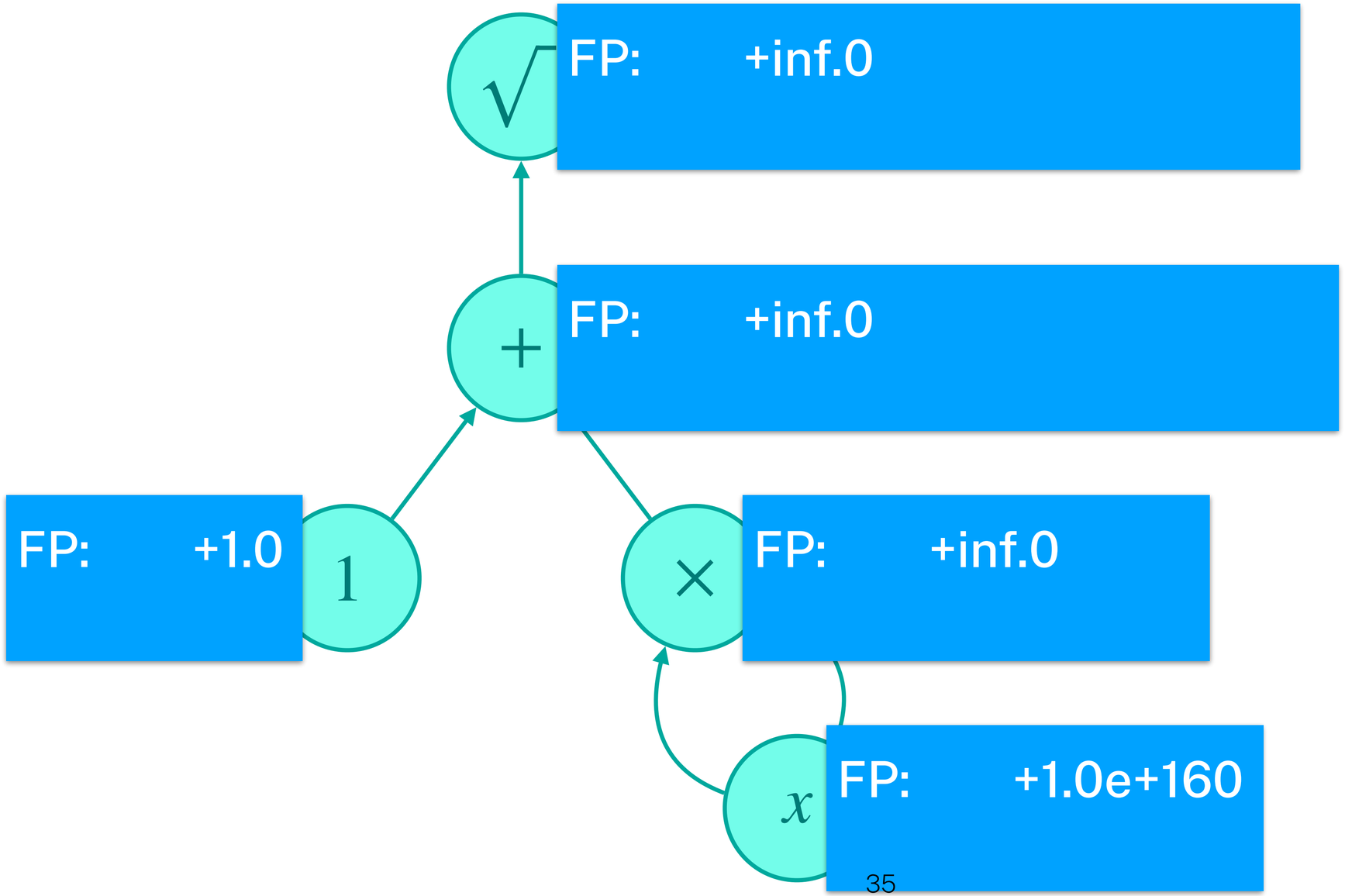
Example, x = 1e+160

$$\sqrt{1+x^2}$$

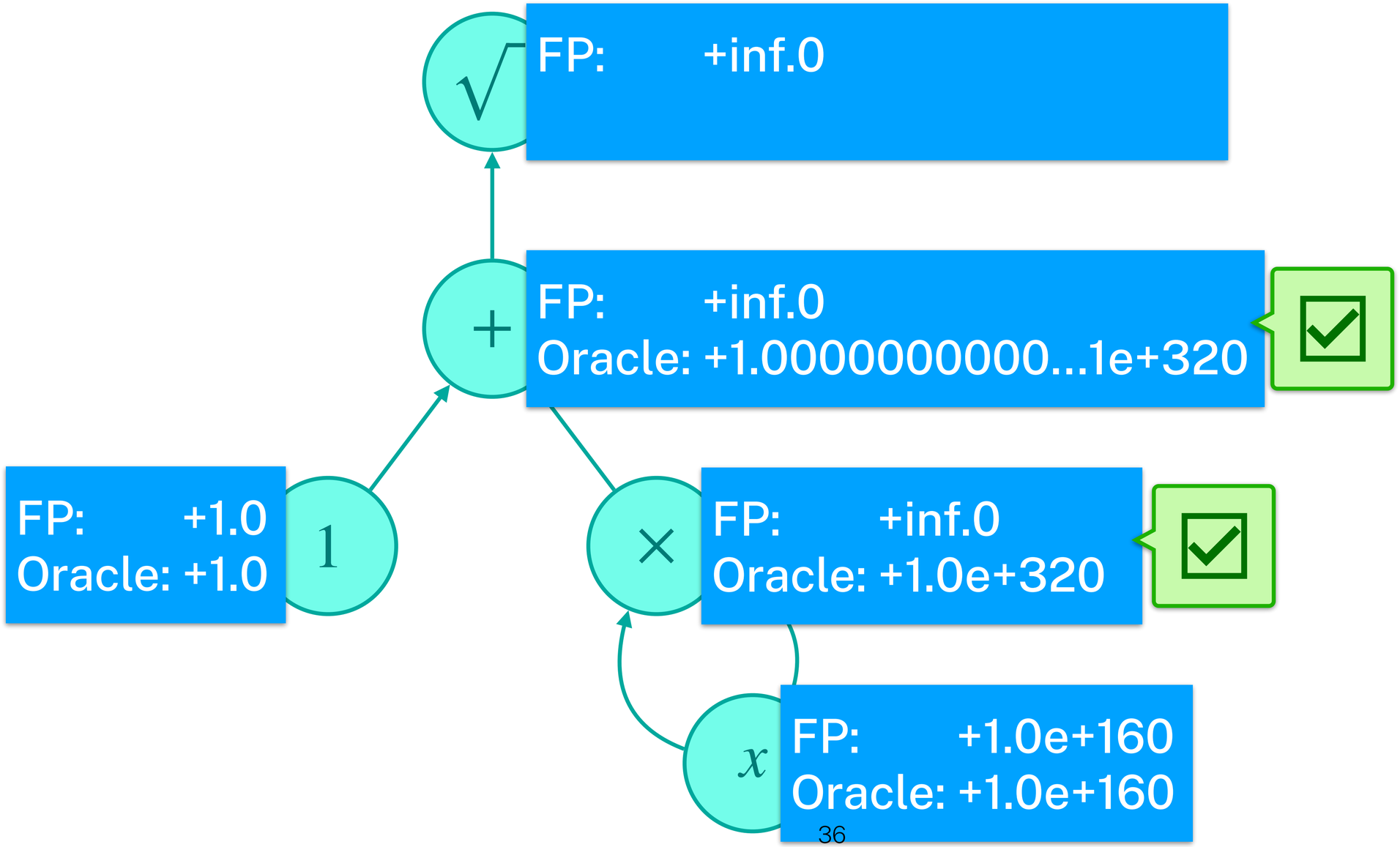
Example, x = 1e+160



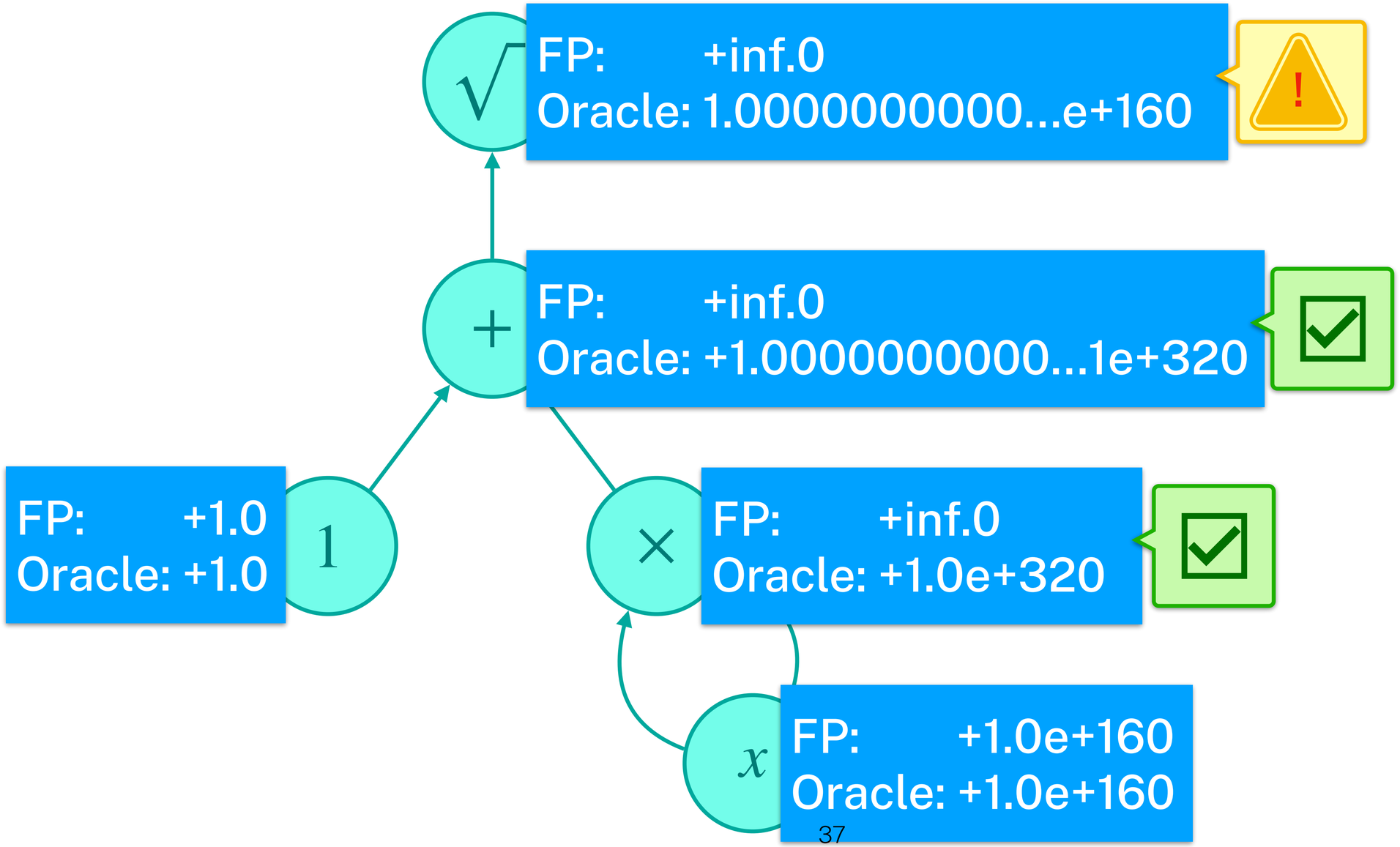
Example, x = 1e+160



Example, x = 1e+160



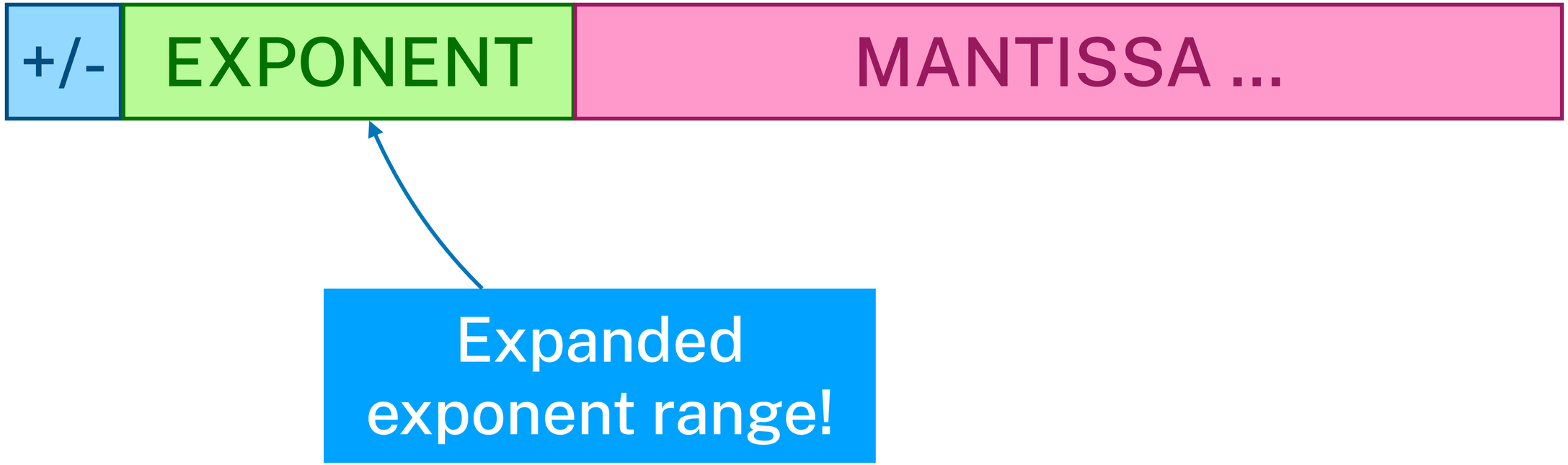
Example, x = 1e+160



Can Condition Numbers deal with Over/Underflow?

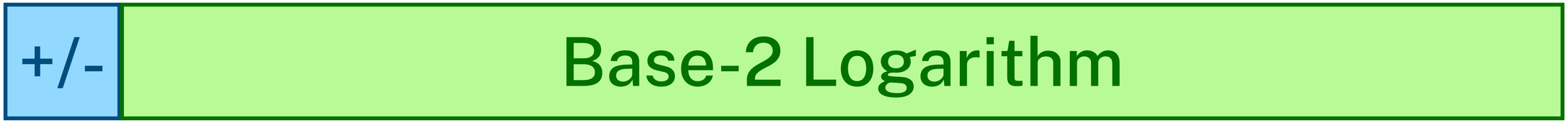
No, over/underflows are out of model!

Oracular Over/Underflow

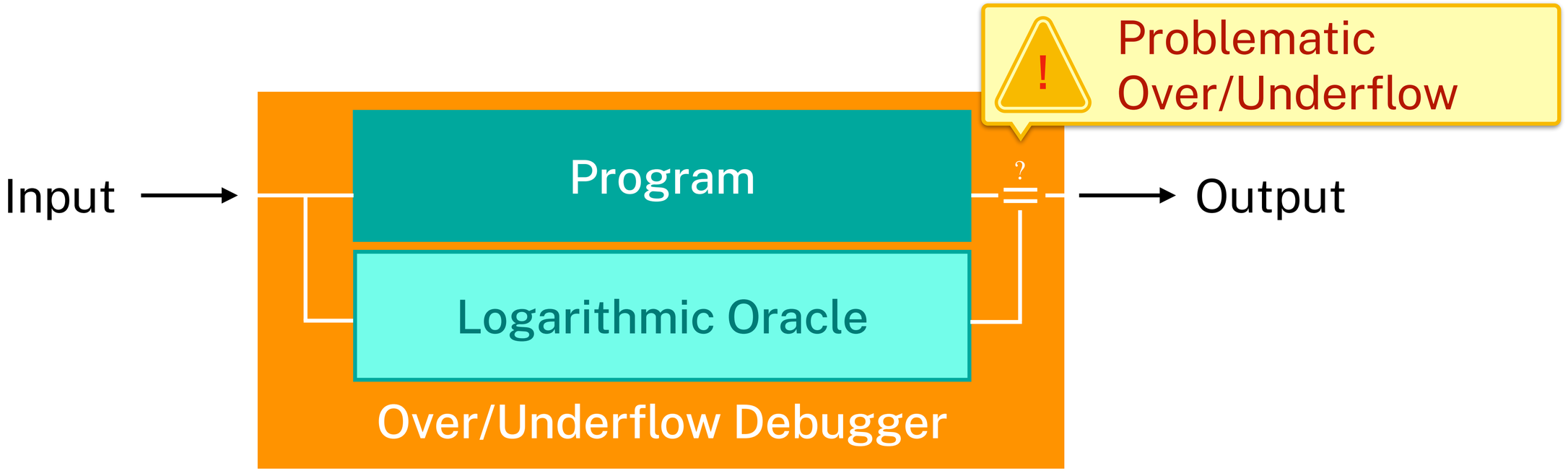


Oracular Over/Underflow

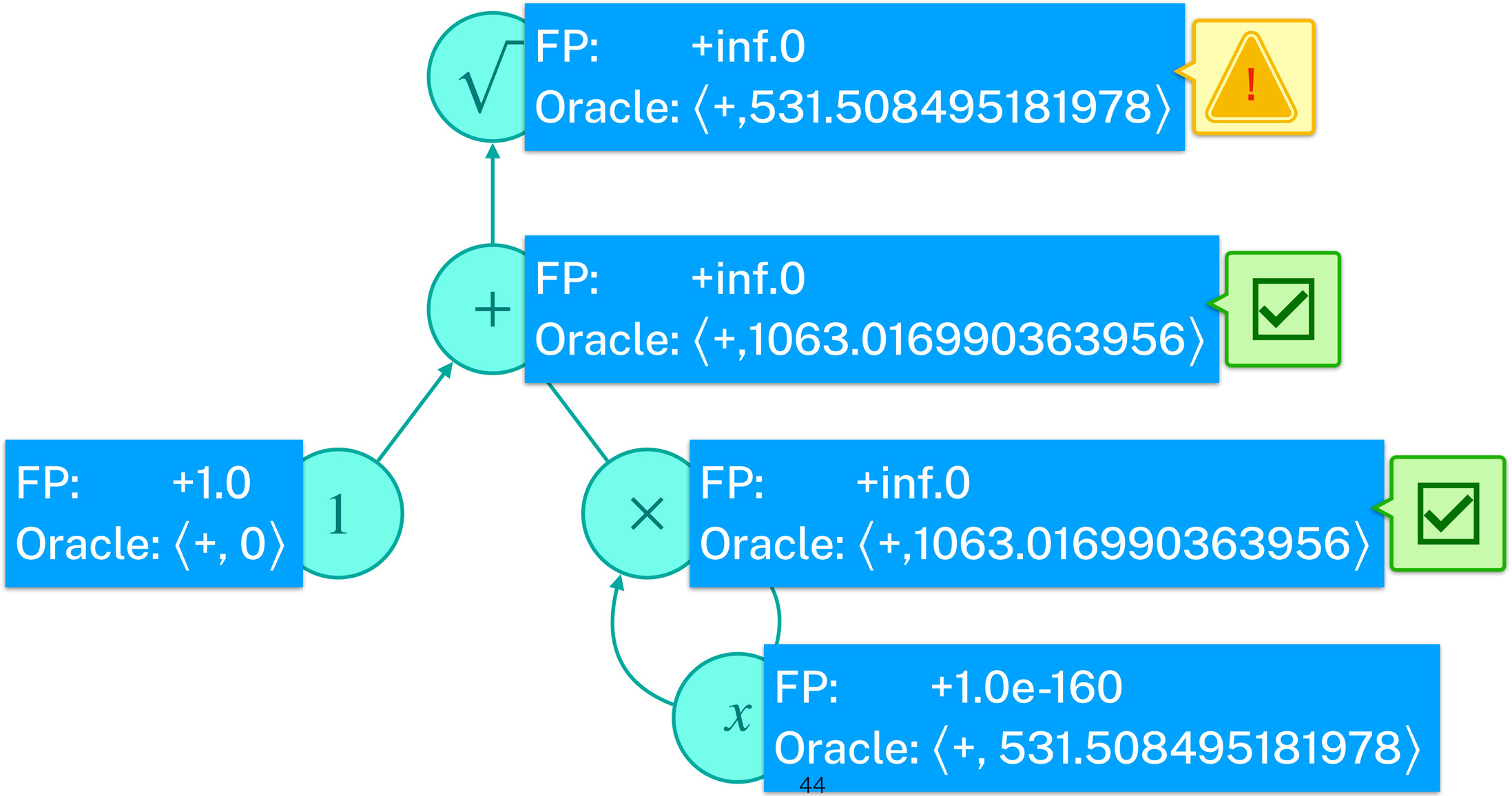
Logarithmic Number System



Over/Underflow Debugger



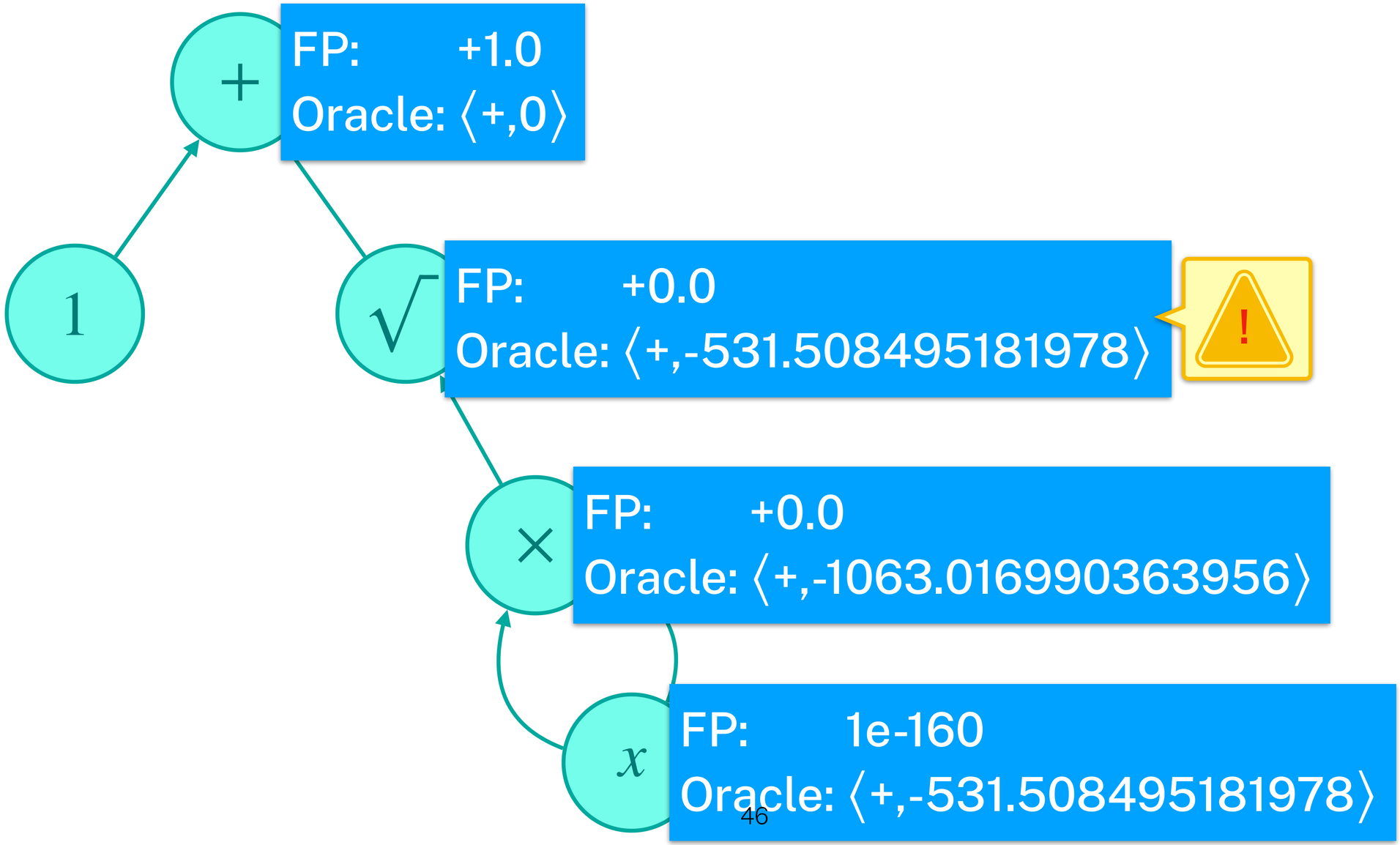
Example, x = 1e+160



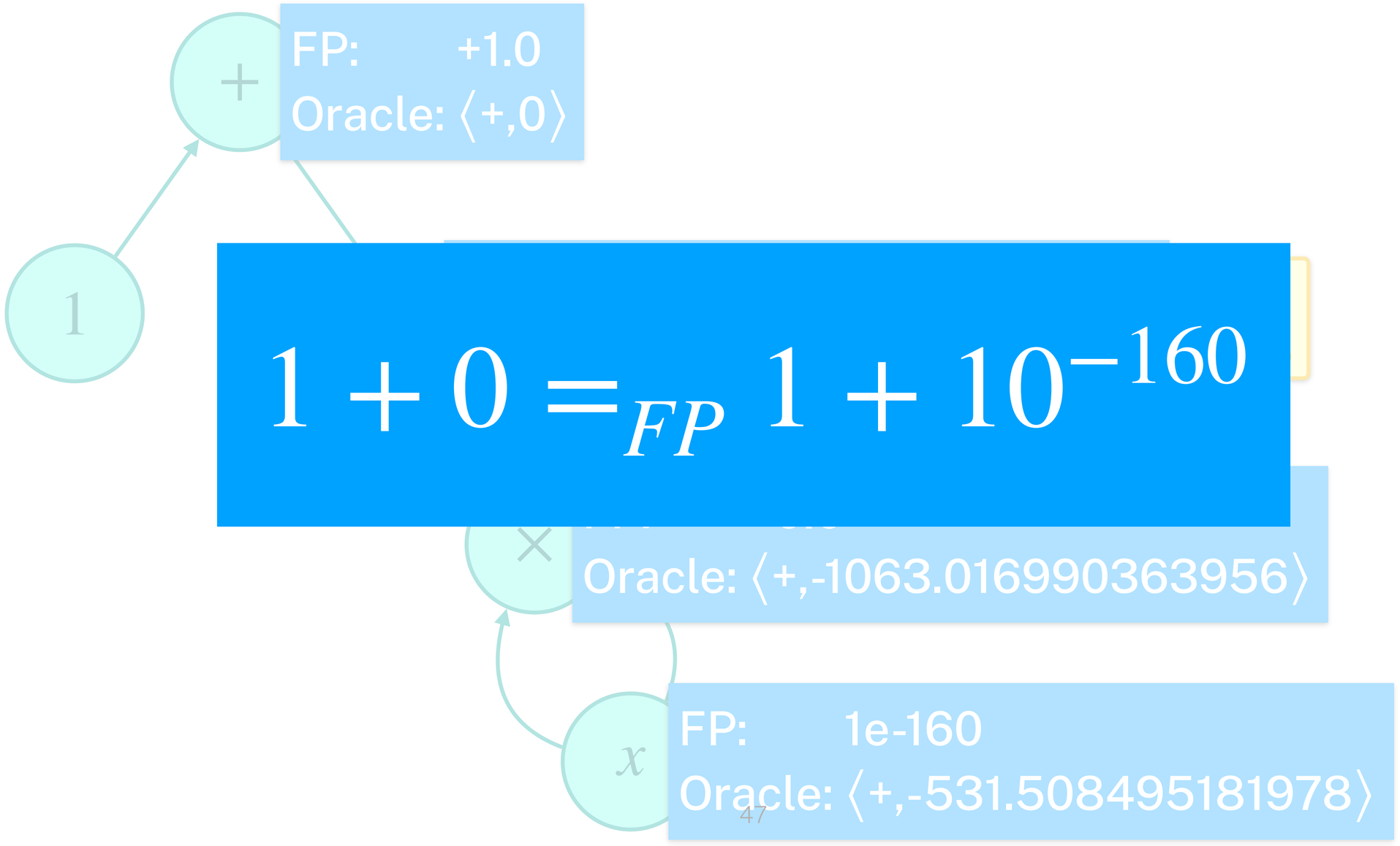
Suppressing silent underflows

$$1 + \sqrt{x^2}$$

Suppressing silent underflows

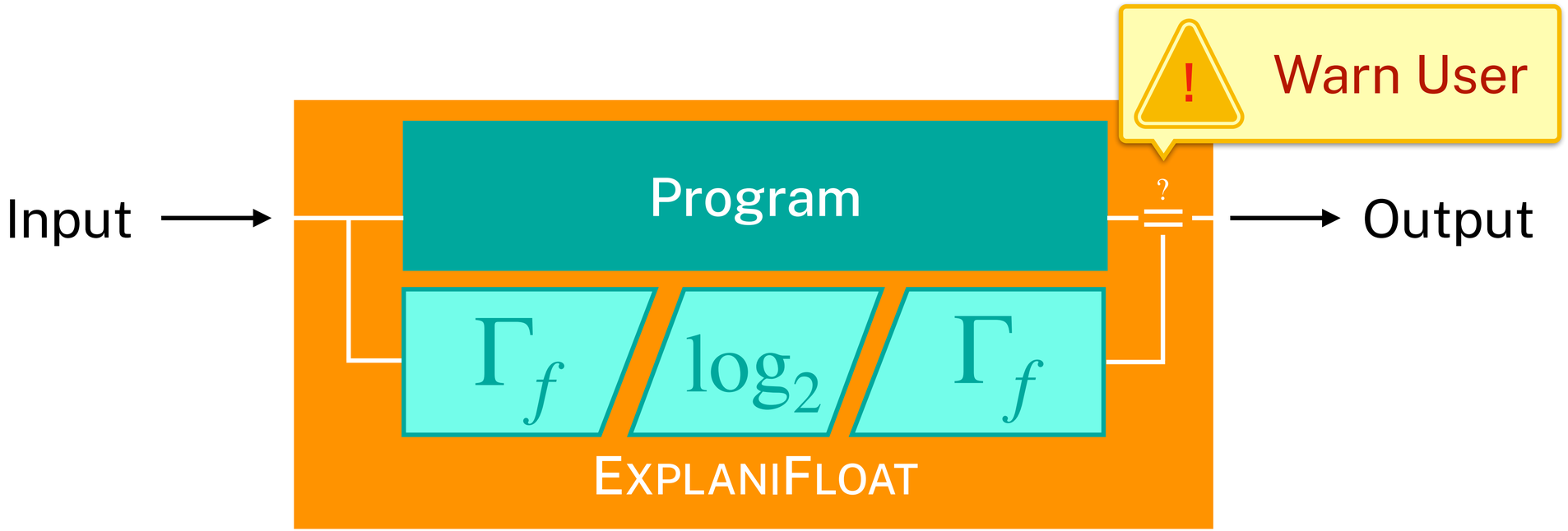


Suppressing silent underflows



Bringing it together

ExplaniFloat



Evaluating ExplaniFloat

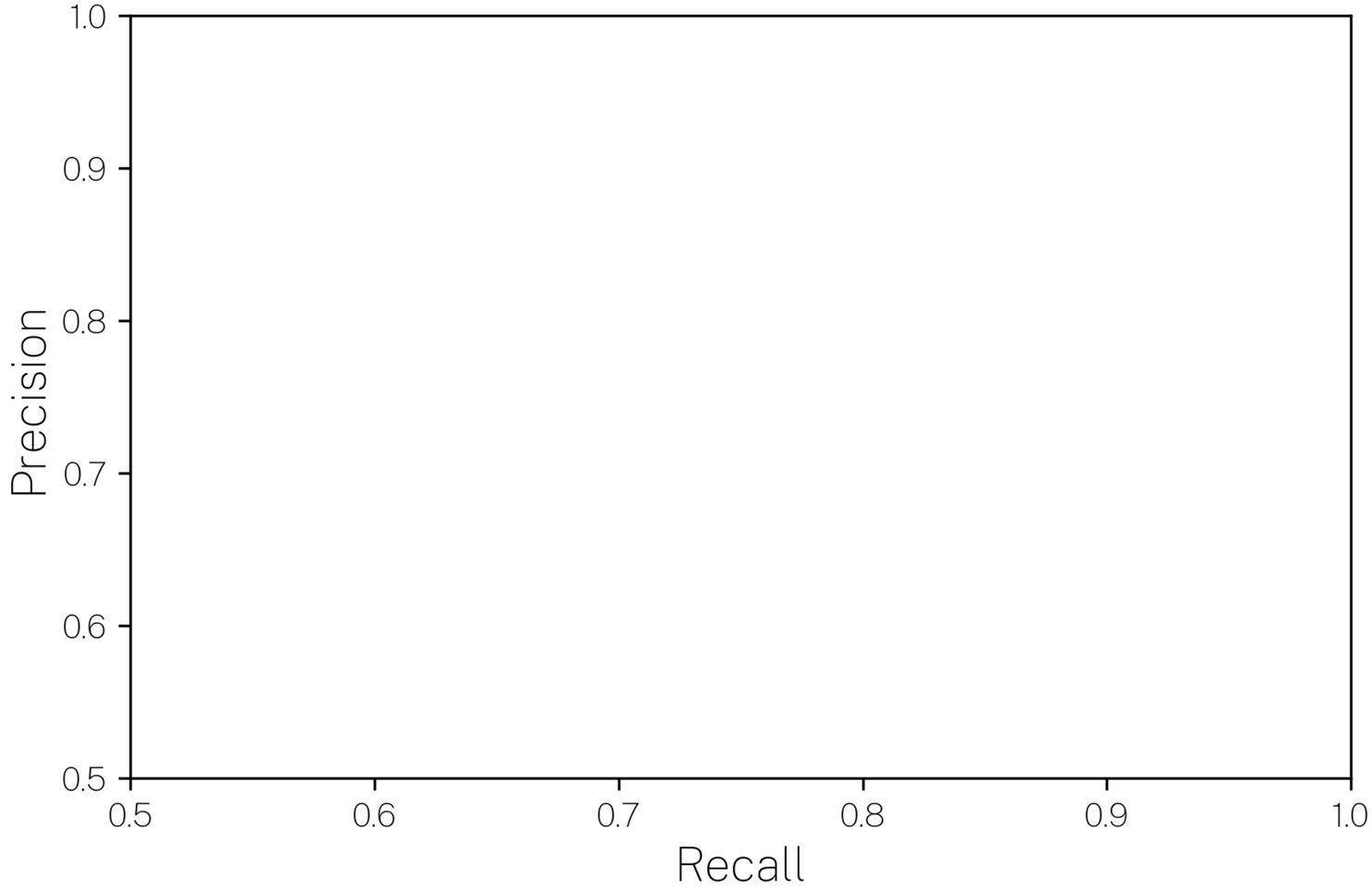
RQ1: Does ExplaniFloat accurately quantify error?

RQ2: Does ExplaniFloat suffer from using machine floats?

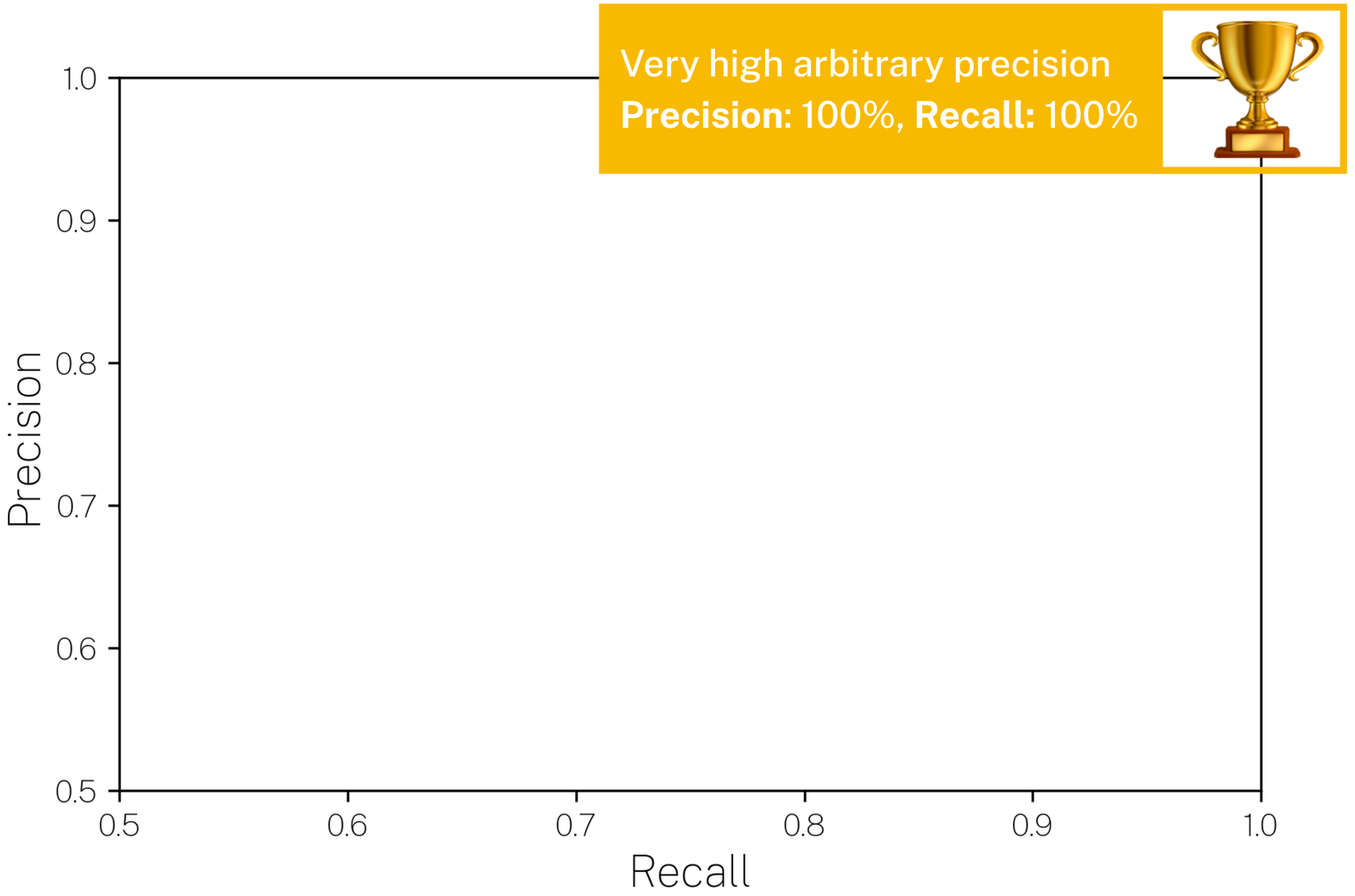
Evaluated over the Herbie 2.0 Benchmark suite.

- A collection of diverse loop-free numerical programs

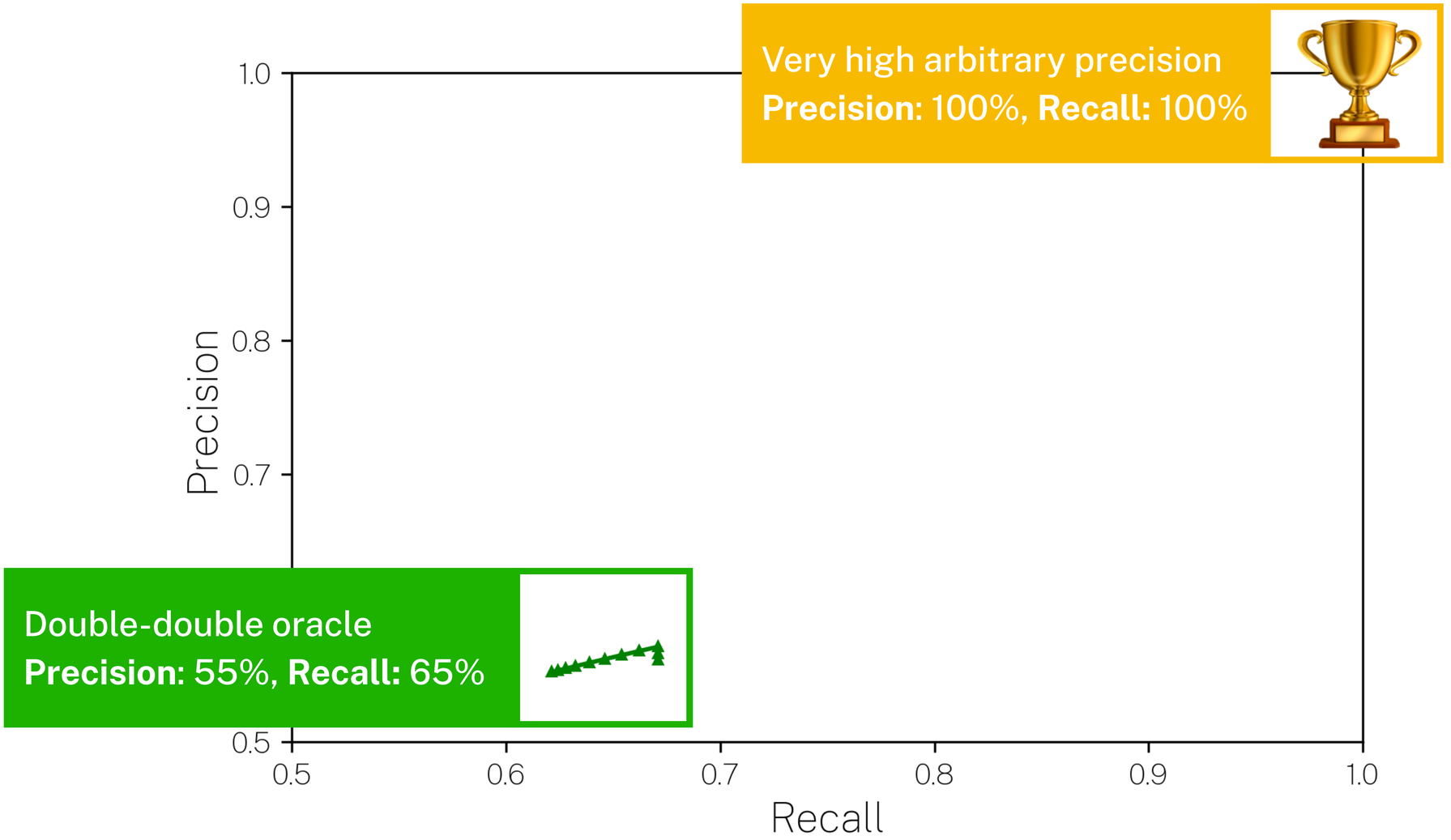
Evaluating ExplaniFloat



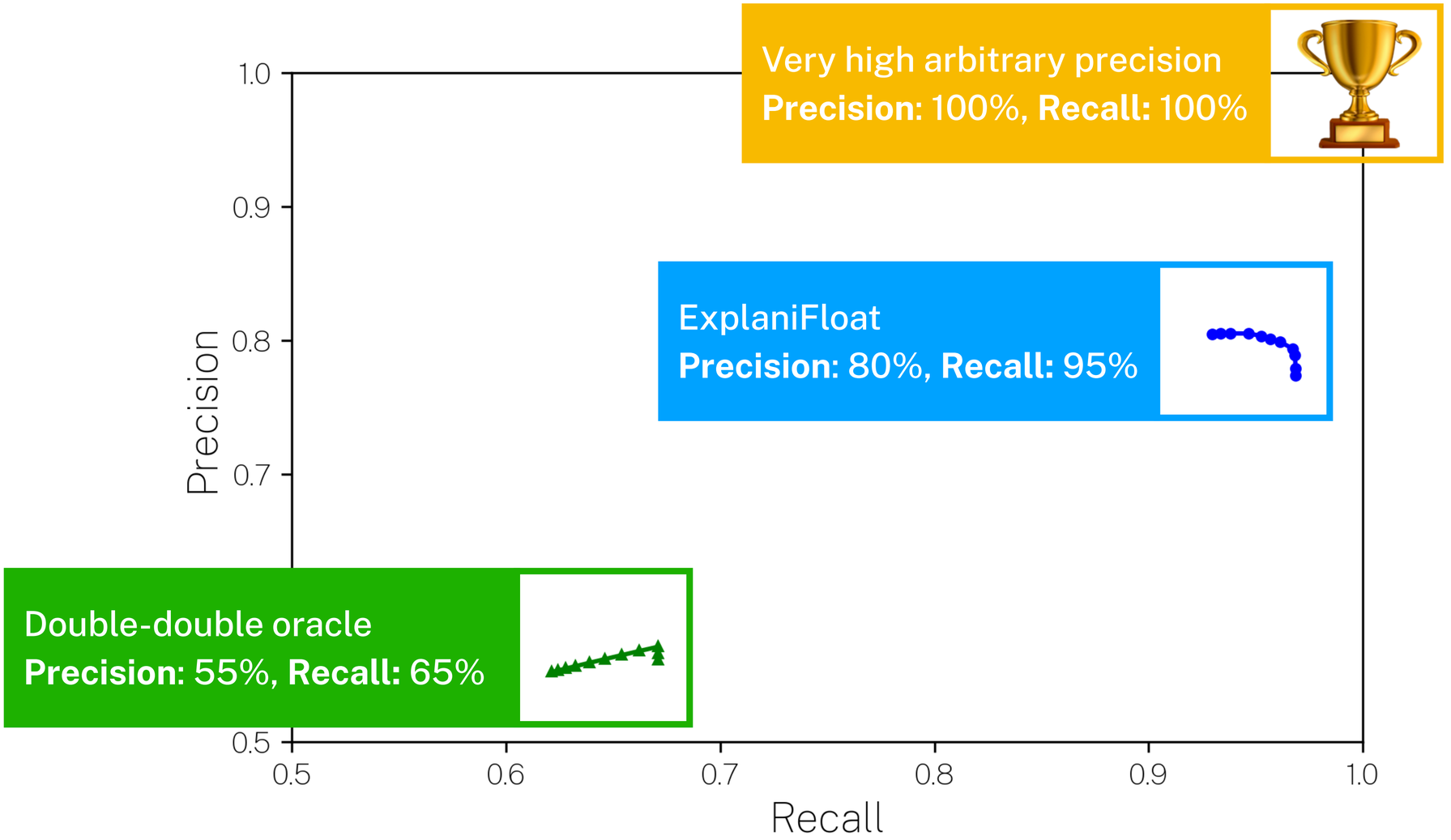
Evaluating ExplaniFloat



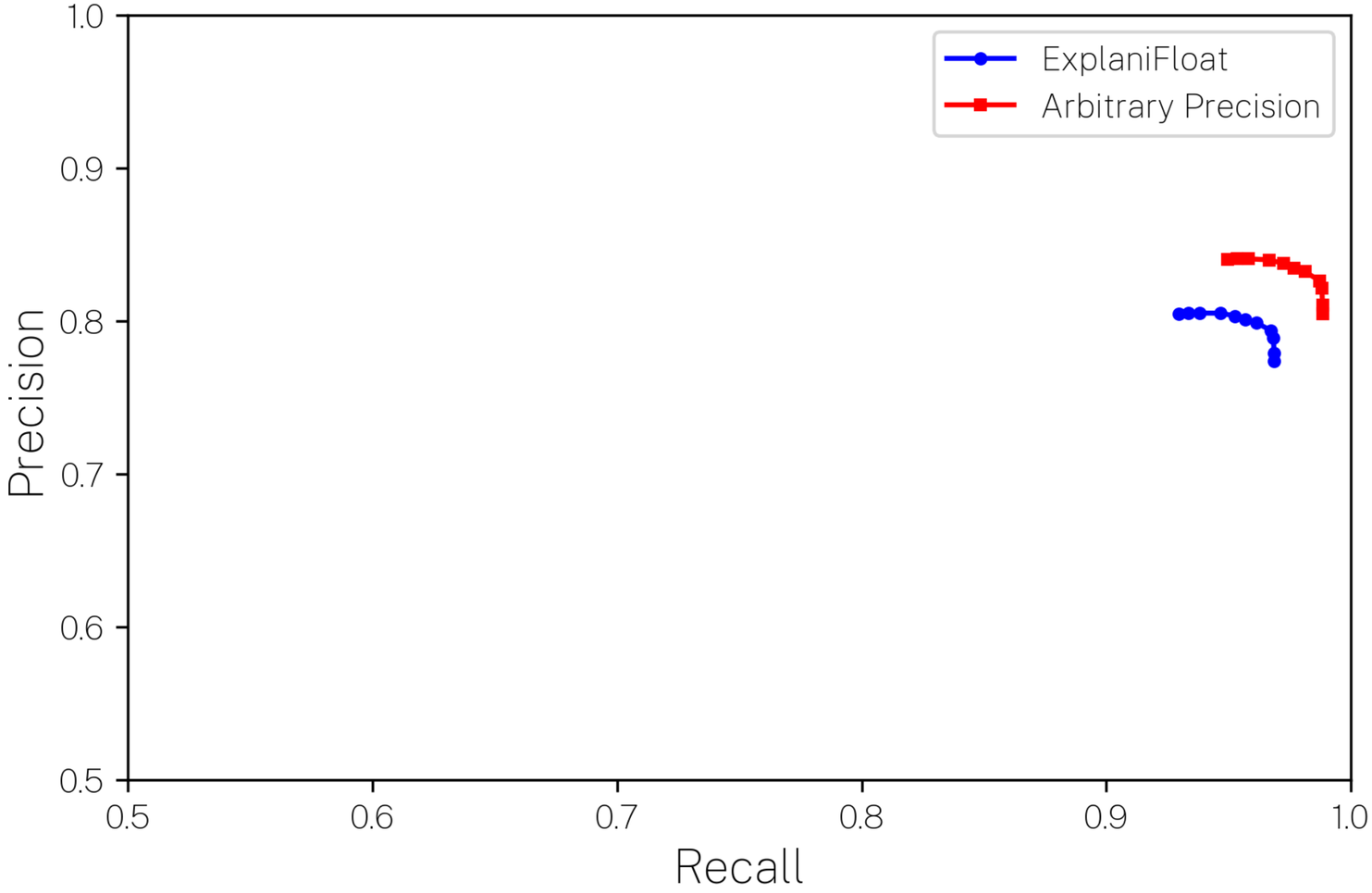
Does ExplaniFloat accurately quantify error?



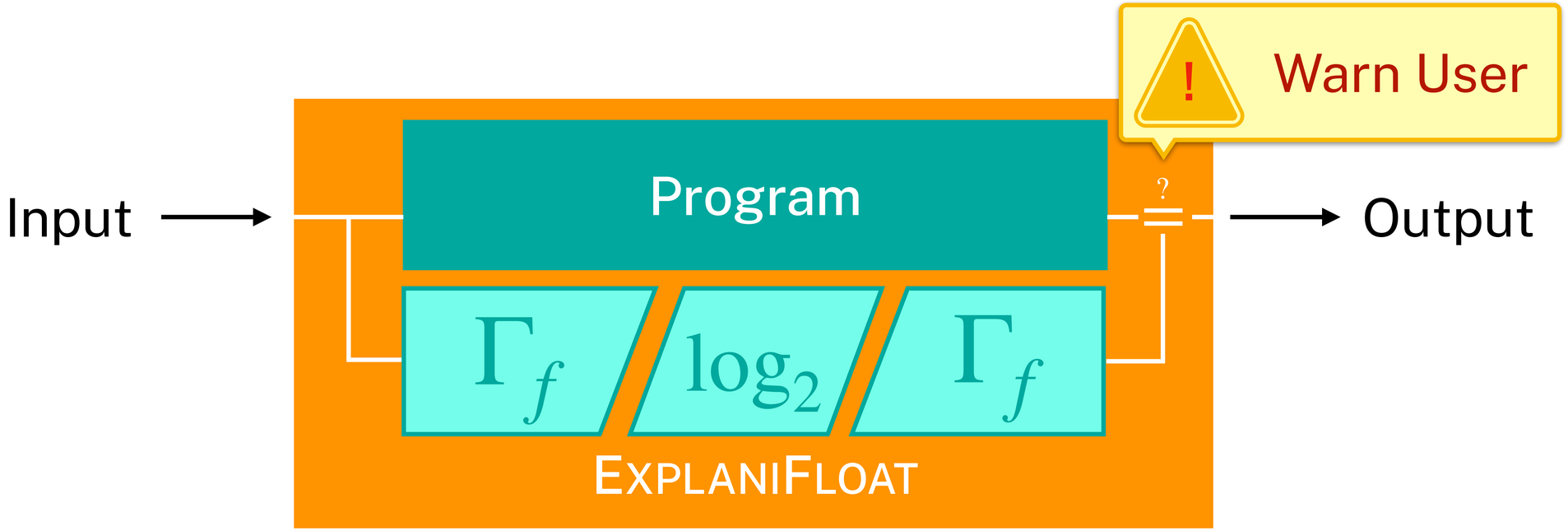
Does ExplaniFloat accurately quantify error?



Does ExplaniFloat suffer from using machine floats?



ExplaniFloat: A **fast** and **accurate** floating-point debugger



bhargavk@cs.utah.edu

Programming Languages at Utah