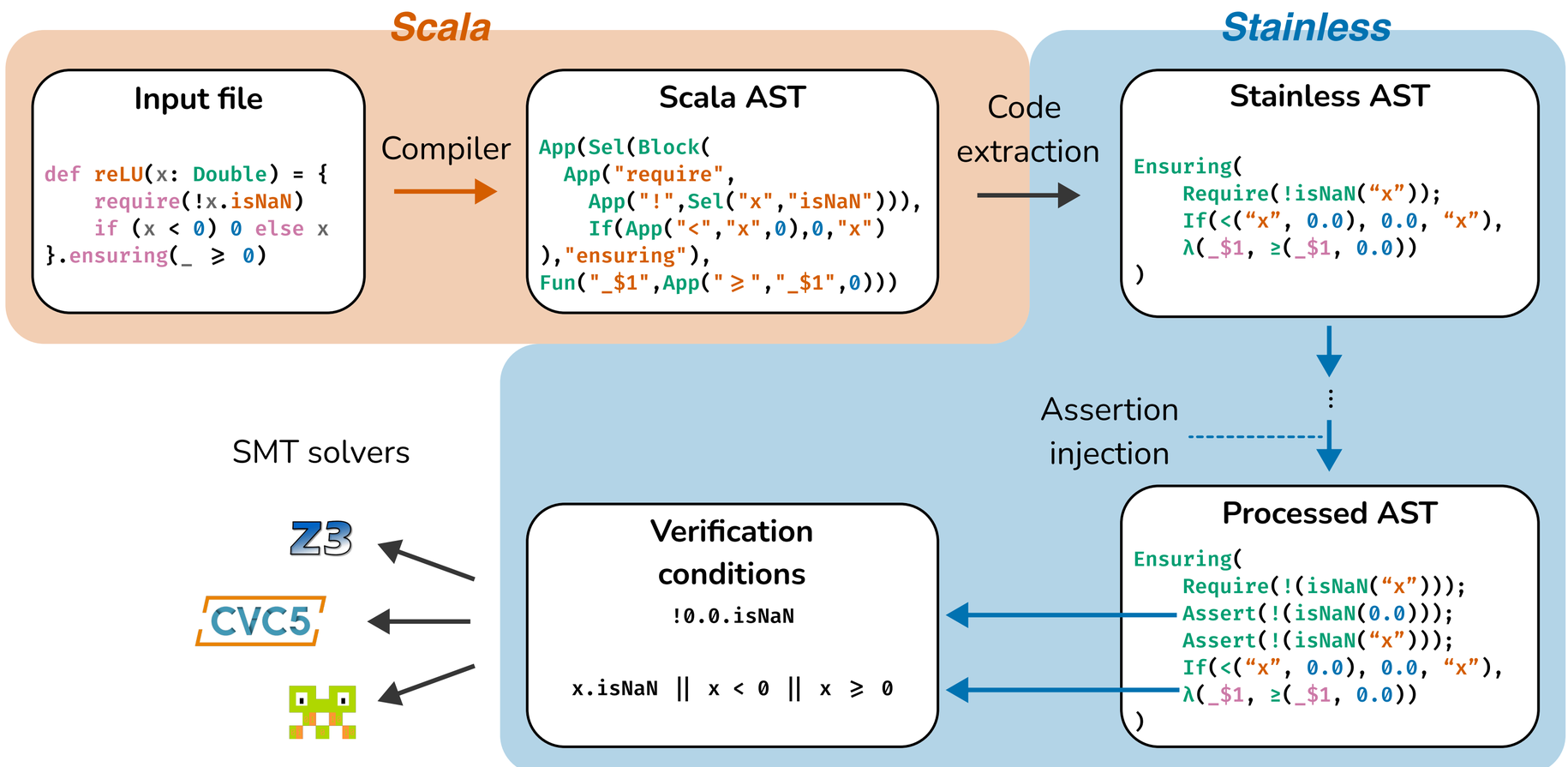
Verifying Floating-Point Programs in Stainless

Andrea Gilot, Axel Bergström, Eva Darulova

UPPSALA UNIVERSITET

We add support for bit-precise floating-point reasoning to the Stainless verifier for Scala, enabling sound reasoning about numerical programs.

Motivation. Floating-point numbers are unintuitive: `NaN` ≠ `NaN` and rounding errors accumulate, introducing bugs in the code.



Polymorphic equality

Problem: Reflexivity of polymorphic equality is unsound when mixed with floating points.

`T = T ⇒ NaN = NaN` ✗

Solution: We extend Stainless with a **`@noeq`** annotation to treat polymorphic equality as an uninterpreted symbol.

```
@noeq
def id[T](t: T) = {t}
.ensuring(res ⇒ res == t)
// postcondition fails ...

id[Float];
// ... but instantiation
succeeds
```

Case Study: Verified Math Standard library

Problem: SMT-solvers do not support transcendental functions.

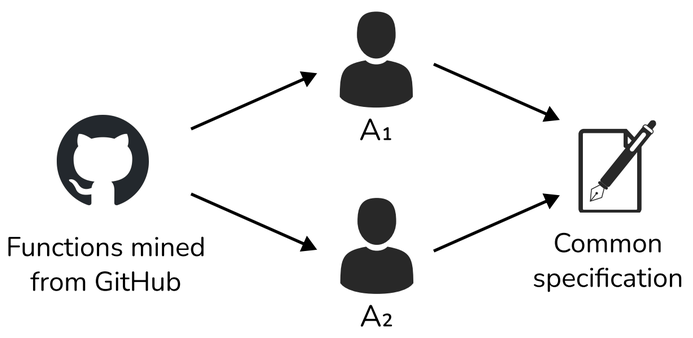
Solution: We verify properties of FdLibm implementations of transcendental functions.

```
def cos(x: Double): Double = {
  // FdLibm implementation
}.ensuring(res =>
  (x.isNaN || x.isInfinity) == res.isNaN
  && (res.isNaN || -1.0d <= res && res <= 1.0d)
)
```

Previous work **axiomatised** these properties, we **verify** them.

User benchmarks

We evaluate Stainless on **user code** by mining floating-point functions on GitHub.



We **independently** add contracts to mined functions, then converge on a common specification.

Bitwuzla

We add Bitwuzla support to Stainless

On our benchmark set Bitwuzla is **significantly faster** than other solvers but **lacks support** for many features of the language.

Proportion of VCs solved on FdLibm benchmarks

Solver	%VCs solved
Z3	78.1%
CVC5	88.7%
Bitwuzla	84.8%

10.6% of VCs are solved **by Bitwuzla only**.

Assertion injection

Non-handling of NaN is a frequent source of bugs. Stainless **automatically** checks that NaN values are not involved in comparisons, equalities and type casts.

Evaluation

`cos(x)` `exp(x)` `pow(x,y)`

FdLibm implementations

KeY benchmarks (adapted from Java)

Functions mined from GitHub

Using Z3 + CVC5 + Bitwuzla, Stainless solves

18/18 benchmarks	8/8 benchmarks	73/79 benchmarks

Funded by the European Union

European Research Council
Established by the European Commission

Read the paper!