LISA

Proof Assistant in Scala

Simon Guilloud, Sankalp Gambhir, Viktor Kunčak

- A tool to **formalize mathematics**
- Based on **first order logic** and **set theory**
- Used like a **Scala** library

Writing Proofs

```
val fixedPointDoubleApplication = Theorem(
        ∀(y, P(y) ⟹ P(f(y))) ⊢ P(x) ⟹ P(f(f(x)))
    ) {
    assume(∀(x, P(x) ⟹ P(f(x))))
    val step1 = have(P(x) ⟹ P(f(x))) by InstantiateForall
    val step2 = have(P(f(x)) ⟹ P(f(f(x)))) by InstantiateForall

    have(thesis) by Tautology.from(step1, step2)
}
```

- States and proves a **theorem**
- Actual Scala code with Domain Specific Language
- Programming features and proofs can be mixed to write **automated proof tactics**.

Kernel

- Deduction system is **Sequent Calculus**.
- Quadratic algorithm to normalizes formulas.
- Based on **Ortholattices**
  - $\approx$ Boolean Algebra without distributivity.
  - Many syntactic transformations automated
  - Shorter, simpler proofs
  - No heuristic

Set Theory

- Library based on **ZFC**
- Can formalize all mathematics
- Contains (yet)
  - Functions and Relations
  - Transfinite Recursion (ordinals)
  - Cantor's theorem

Going Further

github.com/epfl-lara/lisa

Interested in joint work or a project? Contact us!

Contact
Simon Guilloud
simon.guilloud@epfl.ch

EPFL

Laboratory For Automated Reasoning And Analysis