# Fengyun Liu

PhD, EPFL

Lausanne, 1007, Switzerland gurulfy@gmail.com fengy.me +41 (0)78 793 1094


## OBJECTIVE

I'm looking for research jobs related to programming languages.

I'm passionate about using programming language and formal methods to address programmability, performance, security, reliability and productivity challenges.

## EDUCATION

PhD in Computer Science *2016.6 – 2020.6*
*EPFL, Switzerland*
Advisor: Prof. Martin Odersky
Thesis: Safe initialization of objects

Master in Computer Science *2014.9 – 2016.2*
*EPFL, Switzerland*
*thesis:* A Study of Capability-Based Effect Systems

Master in Western Philosophy *2007.9 – 2010.6*
*Nanjing University, China*

Bachelor in Software Engineering *2003.9 – 2007.6*
*Nanjing University, China*

## RESEARCH

I work extensively on the Scala 3 compiler, Dotty, with over 900 commits.

**Safe Initialization of Objects**
Safe object initialization is an open problem since the introduction of object-oriented programming. Checking safe initialization in the presence of aliasing and virtual method calls is a challenge.

Identifying *local reasoning* as the fundamental reasoning principle for designing initialization systems, we propose a type-and-effect inference system that can effectively ensure object initialization with zero syntactic overhead. A technical novelty is the introduction of the concept of *potentials* to control aliasing in type-and-effect systems.

It is by far the most expressive and usable initialization system, and its reliability is grounded on formal semantics and soundness proof.

**Exhaustivity Check**
I propose a new and generic algorithm for checking exhaustivity of pattern matching based on a *space algebra*, which is much simpler than the DPLL-based algorithm in Scala 2. The new algorithm fixes more than 20 open issues in the Scala issue tracker. The algorithm enables the handling of path-dependent

types and GADTs which is not well supported in the Scala 2 compiler. It is later adopted in the Swift compiler and fixes 7 open issues on its first implementation.

**Implicit State Machines**
What is the essence of registers and sequential circuits? In this work we propose a fundamental abstraction for sequential circuits that plays the same role as Boolean algebra for combinational circuits. The concept affords a novel view of sequential circuits. We conjecture the abstraction has big potential for logic synthesis, logic verification and novel hardware architectures.

**Meta-programming**
The project gestalt is an experimental macro system for Scala 3. The problem it tackles is how to support portable macros among different compiler implementations, where each has its own AST definitions. Reflecting on the nature of macros being transforming the *shape* of programs, not the *material*, we propose to implement macros based on abstract extractors and constructors, in contrast to conversion-based standard ASTs. It concretely demonstrates that there exists a better approach to macros than Scalameta. I'm also deeply involved in developing the theory for meta-programming in Scala 3.

**Foundations for Programming with Capabilities**
In the project stoic, we study the fundamental abstractions for programming with capabilities. We propose *stoic functions* as a key abstraction for disciplined programming with capabilities. Stoic functions may not capture capabilities or non-stoic functions from the environment. In a calculus with mutations, we prove the property of *non-interference* and show that *effect polymorphism* can be achieved without introducing effect variables.

## PUBLICATION REPORTS

Digital Design with Implicit State Machines (*under submission*)
*F. Liu, A. Prokopec, M. Odersky*

A Theory of Quoted Code Patterns (*under submission*)
*N. Stucki, F. Liu, A. Biboudis, M. Odersky*

A Type-and-Effect System for Safe Initialization (*to appear*)
*F. Liu, O. Lhotak, A. Biboudis, P. Giarrusso, M. Odersky, OOPSLA, 2020*

Stoic: Towards Disciplined Capabilities (*Technical Report*)
*F. Liu, S. Stucki, N. Amin, P. Giarrusso, M. Odersky, 2019*

Theory and Practice of Coroutines with Snapshots
*A. Prokopec, F. Liu, ECOOP, 2018*

Simplicitly: Foundations and Applications of Implicit Function Types
*M. Odersky, O. Blanvillain, F. Liu, A. Biboudis, H. Miller et al, POPL, 2017*

A Generic Algorithm for Checking Exhaustivity of Pattern Matching

*F. Liu, Scala Symposium, 2016*

A Study of Capability-based Effect Systems
*F. Liu, master thesis, EPFL, 2016*

## Open-Source Contribution

Dotty: The Scala 3 compiler
With over 800 commits, my main contributions include the following :

- I propose and implement a simpler algorithm for exhaustivity check which fixes more than 20 open issues. The algorithm is later adopted by the Swift compiler, and 7 open issues are fixed immediately.
- I propose and implement a type-and-effect inference system for safe initialization of objects. It is by far the most expressive and usable initialization system in programming languages. The reliability of the system is based on formalization and proofs.

Regressional Benchmarking Framework for Dotty
I conceptualize and implement the regressional benchmarking framework for Dotty, with over 1000 commits. The system is commonly used in the development of Dotty.

- It supports on-demand performance test for open PRs
- It supports extensible custom test profiles
- The system does not depend on local database systems

Hashdiff: A Ruby Library for Computing Diffs
I design and implement the algorithm based on dynamic programming, and have been maintaining it for 7+ years. The library has 18.5k dependencies on GitHub, and has more than 63 million downloads.

## Work Experience

*Compiler Developer for Scala* *2016.3 – present*
EPFL, Lausanne, Switzerland

*Full-stack Ruby on Rails Web Developer* *2011.7 – 2014.8*
Freelancer, Nanjing, China

*Software Engineer* *2010.7 – 2011.6*
Fujitsu-Nanda Software Tech Inc., Nanjing, China

## Mentorship

Radoslaw Wasko, master student, EPFL *2020.2 – present*
*Guide the student on Coq mechanization of a calculus of quoted pattern matching for meta-programming.*

Clément Blaudeau, master student, EPFL *2020.2 – present*
*Guide the student on Coq mechanization of a calculus for safe initialization.*

Simon Le Bail-Collet, master student, EPFL *2020.2 – present*

*Guide the student on the design and implementation of stackful coroutines based on CPS transformation and the Scala 3 macro system.*

Vlad Mihaescu, master student, EPFL *2019.9 – 2020.1*
*Guide the student on implementation of a staged SQL to C compiler.*

Loïc Wisniewski, master student, EPFL *2018.9 – 2019.1*
*Guide the student to experiment constant expression types in Scala 3 compiler.*

Thomas Garcia, master student, EPFL *2018.9 – 2019.1*
*Guide the student to experiment SAT solving with the help of deep learning.*

Valdis Adamsons, master student, EPFL *2017.2 – 2017.6*
*Guide student to implement macros in the experimental macro system Gestalt.*

George Zakhour, master student, EPFL *2017.2 – 2017.6*
*Guide the student on the study of an effect calculus.*

## TEACHING ASSISTANT

I have been teaching assistant to the following courses:

*Foundations of Software*, EPFL *Fall 2017, 2018, 2019*
*Advanced Compiler Construction*, EPFL *Spring 2019*
*Parallelism and Concurrency*, EPFL *Spring 2018*

For the master course *foundation of software* on the theory of programming languages, I develop the content for System $F_\omega$ and dependent types and give the lecture. I also develop a Coq project based on Jupyter notebook to teach students how to do proofs in Coq and see concretely the connection between type theory and proof theory.

## REFERENCES

Prof. **Martin Odersky**
martin.odersky@epfl.ch
Programming Methods Laboratory, EPFL

Prof. **Ondřej Lhoták**
olhotak@uwaterloo.ca
Programming Languages Group, University of Waterloo

## EXTRA

Run full *Lausanne Marathon*, 2018
Receive *teaching assistant award* from IC school of EPFL, 2019