

教育背景

南京大学 2023.09 - 2026.06 (预计)  
硕士, 计算机科学与技术 | PASCAL Lab, 导师: 李 | 研究方向为程序语言与程序分析。  
助教工作: 编译原理 (2024 春)

北京航空航天大学 2019.09 - 2023.06  
本科, 计算机科学与技术 | GPA: 3.84/4.00, 排名前 10%, 获得推荐免试研究生资格。  
助教工作: 程序设计基础 (2020 秋), 面向对象设计与构建 (2021 秋, 2022 春 | S.T.A.R. 团队, 负责课程设计和系统开发运维)

工作经历

NVIDIA 2025.02 - 至今  
OCG(Optimizing Code Generator) team GPU 编译器 LLVM 后端实习生

- 参与统一 NVIDIA GPU 图形编译器与 NVVM 的访存指令向量化器, 使得图形编译器的向量化器与 LLVM 上游保持一致:
  - 基于 LLVM GPU 访存向量化器的核心算法设计了多地址图形访存指令的编码方案, 为多条图形访存指令实现了引用分析和向量化, 同时确保尽可能减小代码库与上游代码的差异, 减小后续维护成本;
  - 添加了多个 GPU 访存指令向量化相关的优化, 包括非规则访存序列向量化、整数地址向量化的支持等; 参与实现了新的访存指令对齐宽度推测 pass, 提升向量化器的优化效果, 同时减小了代码库的耦合度;

Rust Foundation Fellowship Program 2024.09 - 2025.09  
Rust 基金会开源资助 (全球约 20 人) Project Fellow

- 作为 Rust-lang Organization 成员 (rust-analyzer-contributors-team) 和 rust-analyzer 维护者 (Rust 语言官方 IDE) 之一, 社区中贡献排名在前 1%, 参与 issues 处理、PR 审核等维护工作; 同时参与维护 rust 语言社区其他项目, 如 rust-clippy 等;
- 实现了控制流高亮、快照测试更新等多项功能, 并参与了大量 bug 修复, 增强了 IDE 在代码理解、自动生成等多方面的能力;
- 为项目的 unicode 断字断行模块编写了 NEON 下的 SIMD 实现, 使该模块在 ARM 平台上提速 6.5 倍;
- v0.3.1992 事故救火: 社区在发布小版本 4 小时后, 发现该版本存在导致资源耗尽且无法结束进程的恶性 BUG。本人在 3 小时内定位到错误算法, 并设计新算法解决了问题。该紧急修复控制了事故影响范围, 避免影响全球 Rust 开发者的工作。

奖项荣誉

- 2022 年本科生国家奖学金 (该学年成绩排名 1/195); 北京航空航天大学优秀毕业生;
- 2021 年全国大学生计算机系统能力大赛·编译系统设计赛 (华为毕昇杯) 一等奖, 总排名第二;
- 蓝桥杯 C++ 程序设计竞赛 A 组北京赛区一等奖、国赛三等奖;
- 另获其他各类省级奖项、校级奖项、奖学金共十余次。

项目开发

Vizsla roife/vizsla (WIP)  
面向芯片前端设计的现代化 IDE · 硕士毕设项目 Rust / SystemVerilog

- 实现了一套面向可综合 SystemVerilog 的语义分析框架以及 IDE 基础设施, 旨在为芯片设计配备现代 IDE 功能;
- 基于增量计算架构, 设计并实现了一套增量分析 IR 和增量分析 pass, 使得代码分析器无需全量更新即可得到准确的分析结果;
- 项目在功能、性能与可用性等指标上均达到业界先进水平: 已完成面向 SystemVerilog 的代码导航定位、语义重构、代码补全、语义高亮、代码诊断等数十项现代 IDE 特性, 并能够利用增量语义分析在各项功能上做到毫秒级延迟;
- 基于语言服务器协议, 适配 VS Code、Emacs、NeoVim 等主流编辑器。

Ailurus roife/ailurus  
编程语言及工具链设计探索 · 个人兴趣项目 Rust

- 基于 Martin-Löf 类型论; 支持 dependent type、dependent pattern matching、inductive datatype 等特性。实现了 propositional equality, 使用 Normalization by Evaluation 进行等价检查, 可实现简单的定理证明;
- 采用基于 typeclass 的 ad-hoc polymorphism, 并基于此实现了运算符重载, 实现了灵活的代码复用机制;

- 实现了 **module system**，支持代码的命名空间管理和封装隔离，解决大型项目中的代码组织和依赖管理问题；
- 旨在作为实验平台，探索现代编程语言工具链（如编译器、IDE 等）的协同设计架构，提高编程语言开发的效率和可维护性。

## Ayame

 [No-SF-Work/ayame](#)

SysY (C 子集) 到 ARMv7 的编译器 · 毕昇杯比赛项目

Java / LLVM-IR / ARM

- 合作项目，个人主要负责编写面向 Machine IR 和体系结构的后端优化和代码生成，完成了基于图着色的**迭代寄存器合并算法、指令调度、死代码删除、窥孔优化**等，同时参与了部分语法树模块的编写；
- 同时负责项目的测试和 DevOps，利用 docker 和 GitLab CI 搭建了测试评估流程，并编写了 Python 脚本自动分析测试结果；
- 项目从零开始，完成了从语法解析到代码生成的完整编译器 pipeline，编写了大量 SSA IR 与 Machine IR 上的优化，最终在比赛中获一等奖。本项目在比赛中总排名第二，在**近一半样例上排名第一**，并在 1/3 的样例上优化效果超越 gcc -O3 与 clang -O3。

## LLVM-Lite





 [roife/llvm-lite](#)

面向深度学习神经网络算子的轻量端侧编译器 · 本科毕设课题














C++ / LLVM-IR

- 课题旨在利用端侧推理设备已知的**形状信息**，对深度学习算子进行**二次编译优化**，以减少算子运行时的时空开销；
- 项目包含运行在推理设备的 LLVM IR **轻量编译器**和对 LLVM Codegen 模块的**裁剪工作**。针对目标 workloads，优化器选择性实现了 **SCCP**、**DCE** 等优化，裁剪工作移除了无关支持，且只保留必要优化，从而以最小的开销取得最好的优化结果；
- 毕业设计获得**优秀**评价。成功将 conv2d 算子和 softmax 算子的推理时间降低 **6%**，并将生成的二进制目标文件减小 **38%**；

## 其他个人项目

-  [Caniformia/HangGai](#) (Vue/RoR / SwiftUI, 合作) 面向北航航概课程的学习应用，支持 Web 端/移动端，已上架 [AppStore](#)；
-  [roife/firefly](#) (Rust) 使用类型系统约束的**神经网络训练/推理框架**，实现了卷积、全连接等算子，并完成了 MNIST 分类；
-  [roife/mole](#) (Verilog / MIPS) 五级流水线 CPU，完成了 **50+** 条指令及**转发/停顿机制**；实现了协处理器 CP0 以响应**中断/异常**；
-  [roife/mos](#) (C / MIPS) 采用 **exokernel** 的 OS 内核，从 bootloader 开始实现了内存映射、进程调度、文件系统、系统调用、Shell 等；

## 🔗 开源社区贡献

-  负责维护官方 IDE (语言服务器)  [rust-lang/rust-analyzer](#)；在 rust 社区也贡献过  [rust-lang/rust](#),  [rust-lang/rust-clippy](#),  [rust-lang/rustup](#),  [rust-lang/rust-mode](#) 等项目；
-  [llvm/llvm-project](#),  [clangd/vscode-clangd](#),  [zed-industries/zed](#),  [MikePopoloski/slang](#),  [google/autocxx](#),  [yuin/goldmark](#),  [moonbitlang/tree-sitter-moonbit](#), 更多项目见 [GitHub](#)。

## 📁 专业技能

- |               |  |
|---------------|--|
| <b>编程语言</b>   | 能力不局限于特定编程语言。熟悉 C, C++, Rust, Java, Python, JavaScript/TypeScript, Verilog/SystemVerilog, EmacsLisp；学习并使用过 Ruby, Swift, OCaml, Haskell, Coq, Agda 等；   |
| <b>程序语言理论</b> | <ul style="list-style-type: none"> <li>▸ 类型论、形式语义、形式语言与自动机、可计算性等理论；学习过 Coq、Agda 等定理证明器的使用；</li> <li>▸ Hindley-Milner, System F, Dependent Type 等<b>类型系统</b>的理论和实现；</li> </ul>  |
| <b>编译器设计</b>  | <b>3 年经验</b> 。编译器从语法解析到代码生成的全 <b>pipeline 开发</b> ，尤其熟悉 <b>编译优化</b> 和 <b>LLVM</b> ： <ul style="list-style-type: none"> <li>▸ 面向对象、函数式等多种范式编程语言的编译过程，以及双向类型检查等编程语言特性的实现；</li> <li>▸ 多种 <b>IR</b> (SSA, MLIR, CPS 等)、<b>优化</b> (Mem2Reg, GVN、寄存器分配等)；熟悉 <b>LLVM</b> 上的分析/优化开发；</li> </ul> |
| <b>程序分析</b>   | 常见 <b>静态分析</b> 算法（数据流分析、控制流分析、IFDS、采用不同敏感度的 <b>指针分析</b> 等）   |
| <b>语言工具链</b>  | <ul style="list-style-type: none"> <li>▸ <b>IDE 开发 2 年经验</b>。熟悉基于<b>增量计算</b>的 IDE 架构，尤其熟悉 rust-analyzer 和 clangd 的架构和实现；</li> <li>▸ 熟悉语言服务器协议 (LSP, Language Server Protocol) 和 VS Code 等编辑器的编程语言插件开发；</li> </ul>  |
| <b>体系结构</b>   | ARM, X86 等常见指令集的架构，现代处理器的架构和乱序执行等概念；NVIDIA GPU 架构；   |
| <b>应用开发</b>   | <ul style="list-style-type: none"> <li>▸ Ruby on Rails, Django 等 Web 后端框架，和使用 SwiftUI 的 iOS 应用开发；</li> <li>▸ PostgreSQL、Redis 等数据库的使用和数据库设计；Docker 和 CI/CD 配置等 DevOps 工作；</li> </ul>   |
| <b>开发环境</b>   | 熟悉 Emacs 与 VS Code，习惯在 macOS / Linux 下工作；能熟练使用生成式 AI 工具提高工作效率。   |

## 📖 其他

- **社团工作**：曾担任北航开放原子开源社团的社长，组织过多次技术分享和交流活动；
- **技术博客**：[roife.github.io](#) 创作时间超 5 年，主要内容为理论计算机和课程笔记，曾帮助大量同学完成 lab，月访问量逾 1.5k；
- **外语**：英语。