Jan-Paul Vincent Ramos-Dávila

jpramos.me | jvr34@cornell.edu | github/jpvinnie | linkedin/jpv-ramos

Education

Cornell University 2021 - 2025
Bachelor of Arts in Computer Science & Philosophy

Experience

Carnegie Mellon University 06/2022 - Present
Research Intern
Advised by Dr. Jonathan Aldrich & Dr. Joshua Sunshine on Gradual Verification: a state of the art verification technique that leverages partial specifications for scalability.
Summer/Fall '22 Fixed optimization bugs and implemented a Property Based Testing tool for evaluating the soundness of Gradual C_0.[3][8]
Spring '23 Developed formal proofs for establishing semantic correspondence to ensure soundness between the static and dynamic verifiers.

Cornell University 10/2021 - 12/2022
Undergraduate Research Assistant
Advised by Dr. Adrian Sampson on the developing the Calyx Compiler Infrastructure for Accelerator Generators. Calyx's control language simplifies encoding of high-level semantics.
Winter '21/Fall '22 Fixed compiler front-end bugs and implemented *Graphicionado Graph Analytics* algorithm in Calyx.
Spring '22 Worked on a symbolic execution tool for verifying parallelism.[2][4][5]

Google 07/2021 - 08/2021
Computer Science Summer Institute
Learned programming fundamentals in JavaScript directly from Google engineers and got an inside look at Google employee tools used for web development. Developed a peer-to-peer instant messaging system with the Express.js framework and Heroku hosting.[7]

Publications

POPL 2023 (SRC) **Evaluating Soundness of a Gradual Verifier with Property Based Testing**
Jan-Paul Ramos-Dávila (Extended Abstract)

CURJ Vol. 2 No. 1 **Evaluating Soundness of a Gradual Verifier with Property Based Testing**
Jan-Paul Ramos-Dávila

Coursework Projects

RNAfoldml *CS 3110 Functional Programming*
OCaml package that enables users to input both RNA sequences in FASTA format and a set of constraints to predict RNA secondary structure.[1]

***JAECC* Eta Compiler** *CS 4120 Introduction to Compilers*
Compiler for the *Eta* programming language in OCaml. Funtional design with the use of *GADT*s.[1][9]

How Do Code and Mined Specs Co-evolve? *CS 6156 Runtime Verification*
Empirical study on temporal behaviors of specifications for code verification in open-source codebases.[6]

Honors

Winner, Third Place, ACM SIGPLAN Symposium POPL Student Research Competition 2023
Travel Scholarship, ACM SIGPLAN Conference PLDI 2022
Finalist, Mathematics, Regeneron International Science and Engineering Fair 2020 & 2021

Technical Skills

Languages: OCaml[1], Python[2], Scala[3], Rust[4], Racket[5], Java[6], JavaScript[7], C[8], Haskell[9], English, Español, Italiano
Tools: Unix, Git, VSCode, IntelliJ IDEA, Neovim, Docker, Heroku, LaTeX