Yusuke Izawa

2 Chome-12-1 Ookayama
Meguro City, Tokyo 152-8550, Japan

Phone: +8190-4027-6104
Email: izawa@prg.is.titech.ac.jp
URL: https://3ttyon.github.io

Current position

PhD Candidate, Department of Mathematical and Computing Science, Tokyo Institute of Technology

Areas of specialisation

Computer Science; Programming Language

Work experience

2018-2019 Recruit Marketing Partners, Inc., Software Engineer, Self-employment.
2018 Cookpad, Inc., Software Engineer, Internship. (Won **2nd Place**)
2016-2017 FOLIO, Inc., Software Engineer, Internship.
2016-2017 KADOKAWA, Inc., Software Engineer, Internship.
2015-2016 Summaly, Inc., Software Engineer, Internship.

Education

2020 MSc in Mathematical and Computing Science, Tokyo Institute of Technology
2018 BSc in Mathematical and Computing Science, Tokyo Institute of Technology

Grants, honours & awards

2020 Tokyo Tech Tsubame Scholarship for Doctoral Students, Tokyo Institute of Technology
2019 **2nd Place**, ACM Student Research Competition, Graduate Category [⋆]
2019 Information Science Incentive Fund, Department of Mathematical and Computing Science, Tokyo Institute of Technology
2015 First semester tuition waiver, Tokyo Institute of Technology
2014 The Showa Scholarship Foundation

Publications & talks

2020 Yusuke Izawa, and Hidehiko Masuhara. Toward Hybrid Compilation in a Practical Meta-tracing JIT Compiler. Programming Language Mentoring Workshop, 41st ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '20). June 2020. Oral Presentation.

2020 Yusuke Izawa, and Hidehiko Masuhara. Integrating Different JIT Compilation Strategies in a Meta-tracing JIT Compiler. preprint. 12pages.

2020 Yusuke Izawa, and Hidehiko Masuhara. Making different JIT compilations dancing to the same tune, acting in the meta-level. The 22nd JSSST Workshop on Programming and Programming Languages (PPL '20), March 2020. Poster Presentation.

2020 Yusuke Izawa, Hidehiko Masuhara, Tomoyuki Aotani, and Youyou Cong. A stack hybridization for meta-hybrid just-in-time compilation. In Kei Ito, editor, Proceedings of the 36th JSSST Annual Conference, pages No. 2–L. Japan Society for Software Science and Technology (JSSST PPL '20), August 2019.

2019 Yusuke Izawa, Hidehiko Masuhara, and Tomoyuki Aotani. 2019. Extending a meta-tracing compiler to mix method and tracing compilation. In Proceedings of the Conference Companion of the 3rd International Conference on Art, Science, and Engineering of Programming (Programming '19). ACM, New York, NY, USA, Article 5, 1–3. DOI: https://doi.org/10.1145/3328433.3328439

2019 Yusuke Izawa. BacCaml: the meta-hybrid just-in-time compiler. In Proceedings of the Conference Companion of the 3rd International Conference on Art, Science, and Engineering of Programming (Programming '19). ACM, New York, NY, USA, Article 32, 1–3. DOI:https://doi.org/10.1145/3328433.3328466 (**Awarded** [⋆])

2018 Yusuke Izawa, Hidehiko Masuhara, and Tomoyuki Aotani, Proposal of Meta-hybrid JIT Compiler, The 20nd JSSST Workshop on Programming and Programming Languages (PPL '18), March 2018. Poster Presentation.

2018 Yusuke Izawa, Hidehiko Masuhara, and Tomoyuki Aotani, Meta-hybrid JIT Compilation Approach to the Path Divergence Problem, Kumiki 6.0, November 2018. Oral Presentation.

Teaching

2020 (3Q) Programming II, Tokyo Institute of Technology, Teaching Assistant
2019 (1Q) Programming II, Tokyo Institute of Technology, Teaching Assistant
2019 (1Q) Introduction to Computer Science, Tokyo Institute of Technology, Teaching Assistant
2018 (3Q) Programming I, Tokyo Institute of Technology, Teaching Assistant
2018 (1Q) Information Literacy 1, Tokyo Institute of Technology, Teaching Assistant

Academic services

2020 Candidate of Programming Language Mentoring Workshop, PLDI 2020, June 2020.
2019 Member of Student Volunterr, Programming 2019, April 2019.

Projects

BacCaml (source code at GitHub)

Trace-based compilation and method-based compilation are two major compilation strategies in JIT compilers. In general, the former excels in compiling programs with deeper method calls and more dynamic branches, while the latter is suitable wide range of programs.

This project aims at developing fundamental mechanism for compiling with both trace-based and method-based strategies. Instead of developing a compiler for one particular language, we provide such a mechanism in a meta-compilation framework, which generates a virtual machine with a JIT compiler from an interpreter definition of a programming language.

We are developing the BacCamel meta-compiler framework as a proof-of-concept, which is based on the MinCaml compiler.