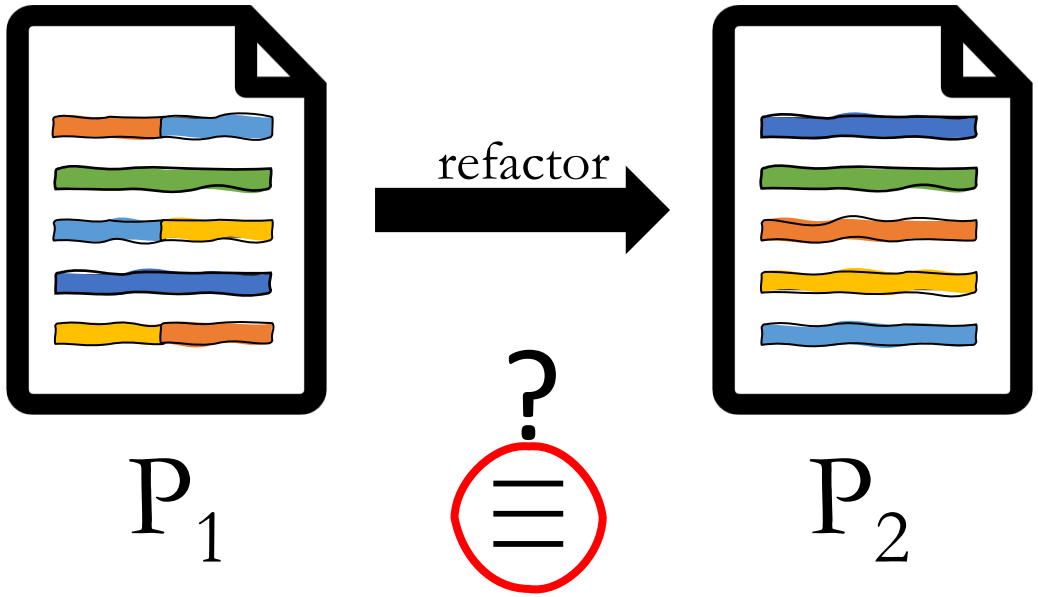
Nagy megbízhatóságú refaktorálás Erlangban

Horpácsi Dániel • Simon Thompson
Bajka Ákos • Bereczky Péter • Katkó Dominik • Sághi György • Szalay Bence • Vadász András

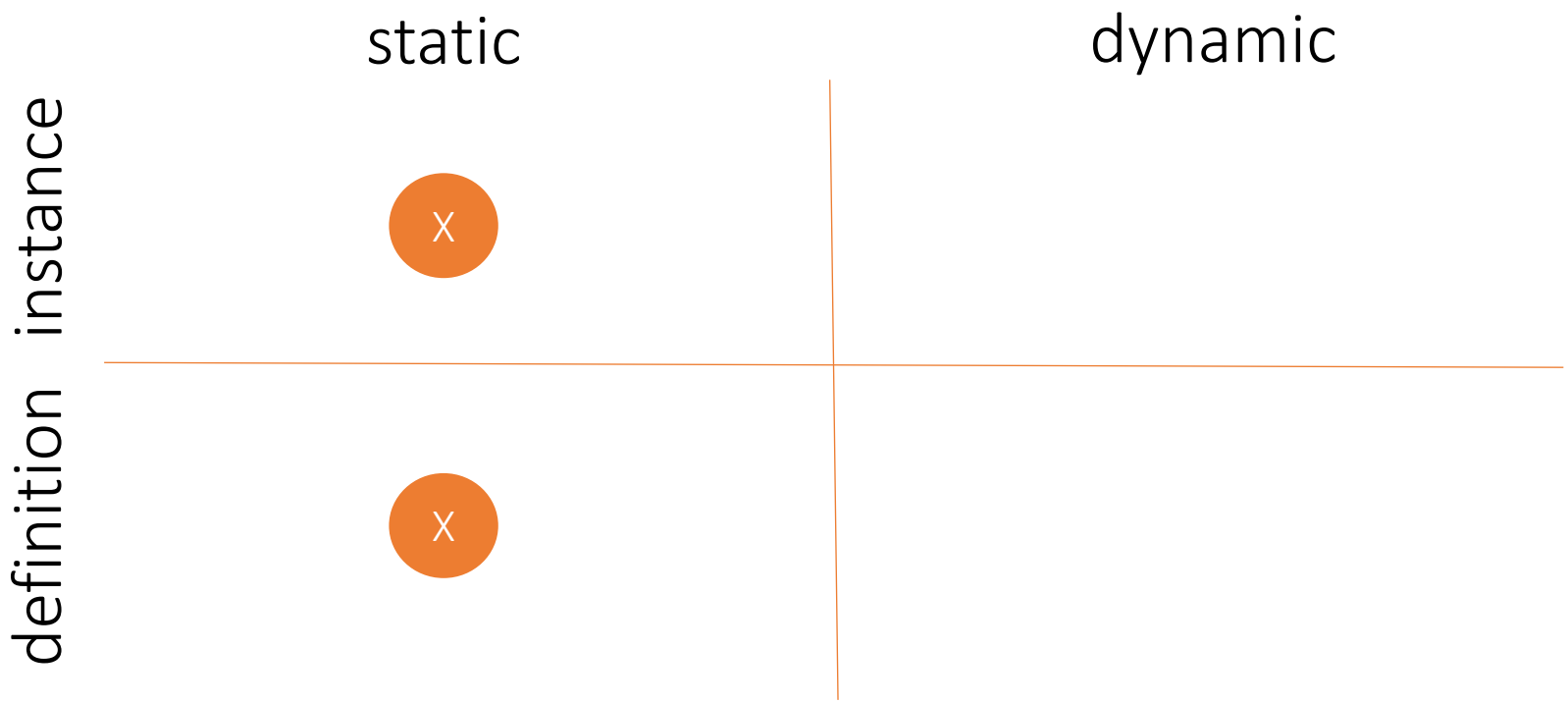
TKP Workshop, 2022. január 13. (online)

Gentle reminder: refactoring correctness



$$\forall P: (P \stackrel{?}{\equiv} \text{refactor}(P))$$

Gentle reminder: formal verification of refactoring



We formalize our definitions and prove the theorems in **Coq, a formal proof management system**.
Therefore, the proofs are all machine-checked, which **provides a very high assurance of correctness**.

Our approach in a nutshell

Define language semantics to formally capture program behavior

Define (contextual) equivalence relations that imply observational behavioral equivalence

Prove that some generic (parametrized) transformations map every program to an equivalent program

Derive several correct-by-construction refactoring steps by instantiating the generic transformations

Compose the correct refactoring steps into compound, widely applicable code paraphrasers by imperative means

Our approach in a nutshell

Define language semantics to formally capture program behavior

Define (contextual) equivalence relations that imply observational behavioral equivalence

Prove that some generic (parametrized) transformations map every program to an equivalent program

Derive several correct-by-construction refactoring steps by instantiating the generic transformations

Compose the correct refactoring steps into compound, widely applicable code paraphrasers by imperative means

In focus

Formalization of Core Erlang: towards a complete language definition

Formalization of Matching Logic: towards a basis for semantics reasoning

Refactoring and LSP: a modern interface to applying Erlang refactoring

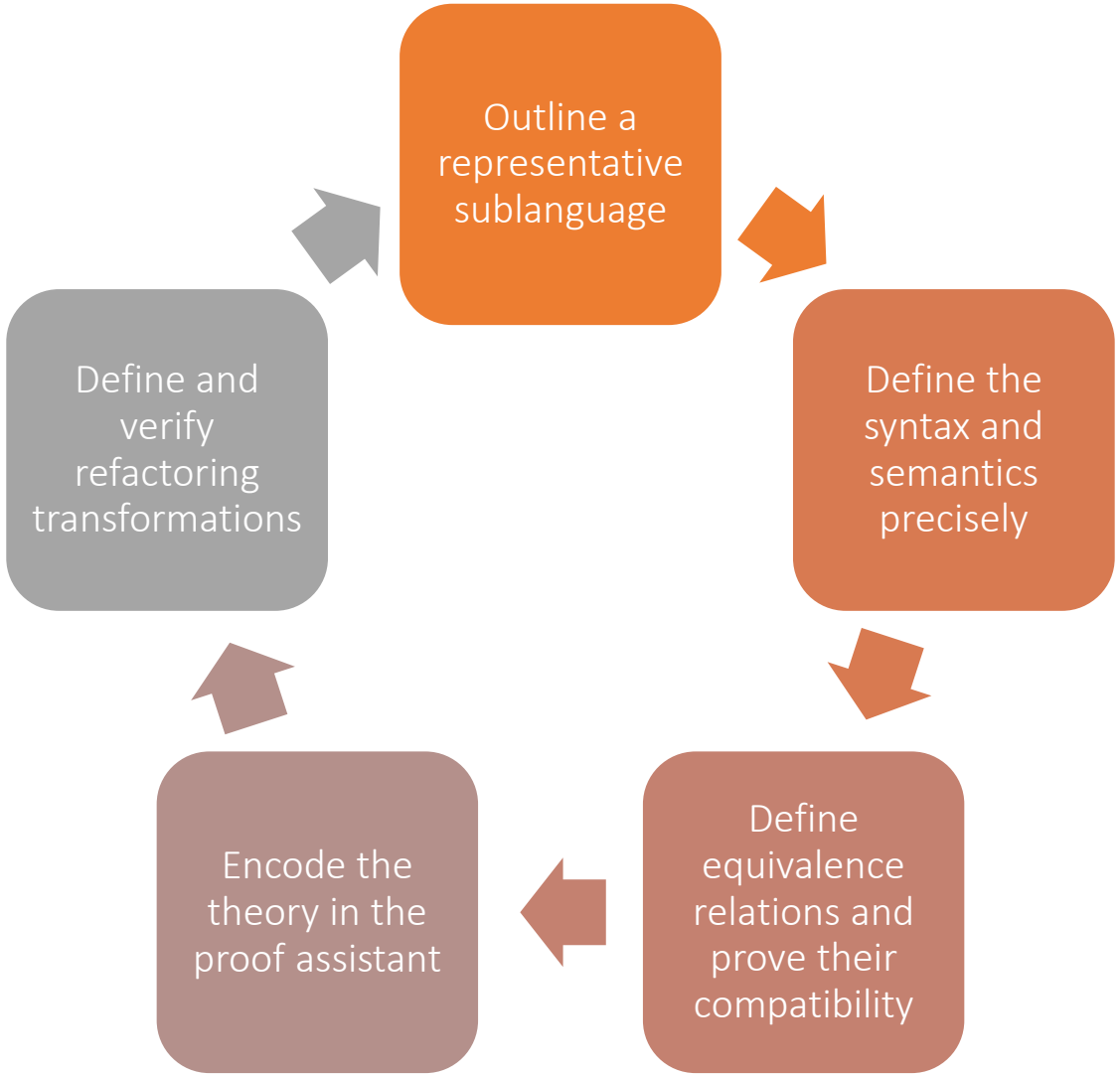
In focus

Formalization of Core Erlang: towards a complete language definition

Formalization of Matching Logic: towards a basis for semantics reasoning

Refactoring and LSP: a modern interface to applying Erlang refactoring

Our process towards a complete Erlang theory

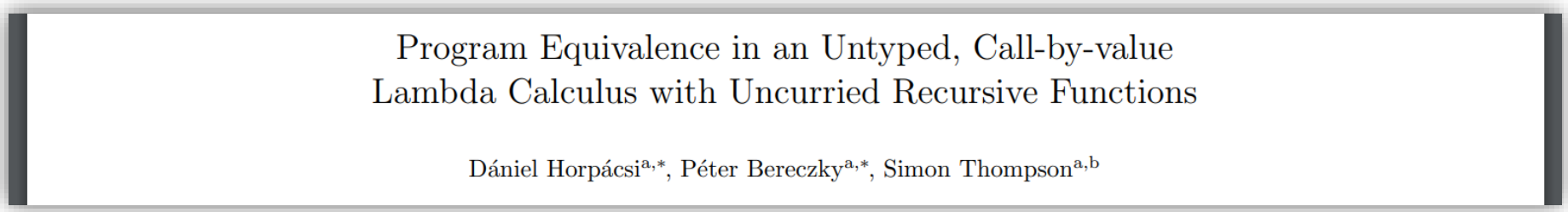


Formalization of Core Erlang

In 2021-H1, we have developed

- **Frame stack semantics** for a minimalistic dialect of Core Erlang – a novel approach
- **Behavioral and contextual equivalence** definitions for the minimal language, adapting the best of the related work in the field
- Solutions to several formalization issues (e.g., function closure representation and equivalence, expression scoping, parallel substitutions, equivalence correspondences)

The results have been submitted to Journal of Logical and Algebraic Methods in Programming (JLAMP) in September 2021, currently awaiting reviews.


Program Equivalence in an Untyped, Call-by-value
Lambda Calculus with Uncurried Recursive Functions

Dániel Horpácsi[a,*], Péter Bereczky[a,*], Simon Thompson[a,b]

Formalization of Core Erlang

In 2021-H1, we have developed

- **Frame stack semantics** for a minimalistic dialect of Core Erlang – novel approach
- **Behavioral and contextual equivalence** definitions for the minimal language, adapting the best of the related work in the field
- Solutions to several formalization issues (e.g., function closure representation and equivalence, expression scoping, parallel substitutions, equivalence correspondences)

/harp-project/Core-Erlang-mini

Formalization of Core Erlang

In 2021-H2, we focused on **extending language coverage**

- Adding the rest of the expr… attern matching, list and tuple types, except…
- Formalizing the Erlang mod… unctions, export and import, MFA calls, related exce…
- Developing semantics for concurrent and distributed language features (processes and PIDs, process-local computation, send and receive, node interaction)
- And apparently, the static semantics (scoping, substitution) and the equivalence definitions need to be extended accordingly.

- Separation of process-local and node semantics
- Representation of processes and nodes
- Pattern matching in receive (mailbox query)

Publications to be written in 2022-H1 in cooperation with the involved students.

Formalization of Core Erlang

In 2021-H2, we focused on **extending language coverage**

- Adding the rest of the expression kinds to the frame stack semantics (pattern matching, list and tuple types, exceptions and IO statements)
- Formalizing the Erlang module system (modules with local and global functions, export and import, MFA calls, related exceptions)
- Developing semantics for concurrent and distributed language features (processes and PIDs, process-local computation, send and receive, node interaction)
- And apparently, the static semantics (scoping, substitution) and the equivalence definitions need to be extended accordingly.

/harp-project/Core-Erlang-Formalization

In focus

Formalization of Core Erlang: towards a complete language definition

Formalization of Matching Logic: towards a basis for semantics reasoning

Refactoring and LSP: a modern interface to applying Erlang refactoring

Rationale

Matching Logic (ML) is a formal system for reasoning about programming languages.

Numerous programming languages are already formalized in ML (in the K framework)

- So, our solutions will be available to those languages out of the box

ML provides a nice intermediate calculus between the PL and CIC

- Reasoning about equivalence will be carried out on a higher level

Reasoning about equivalence in ML has been studied in related work

- We can reuse those algorithms given for equivalence reasoning in ML

ML and its ecosystem is actively developed and is of interest in the community

- Seemingly, it is a nice idea to contribute to

A formalization of ML

- In 2021, in collaboration with UIUC, we have developed an initial locally nameless formalization of ML in the Coq proof assistant
- Simple theories (such as FOL and LTL) and soundness were also formalized

/harp-project/AML-Formalization

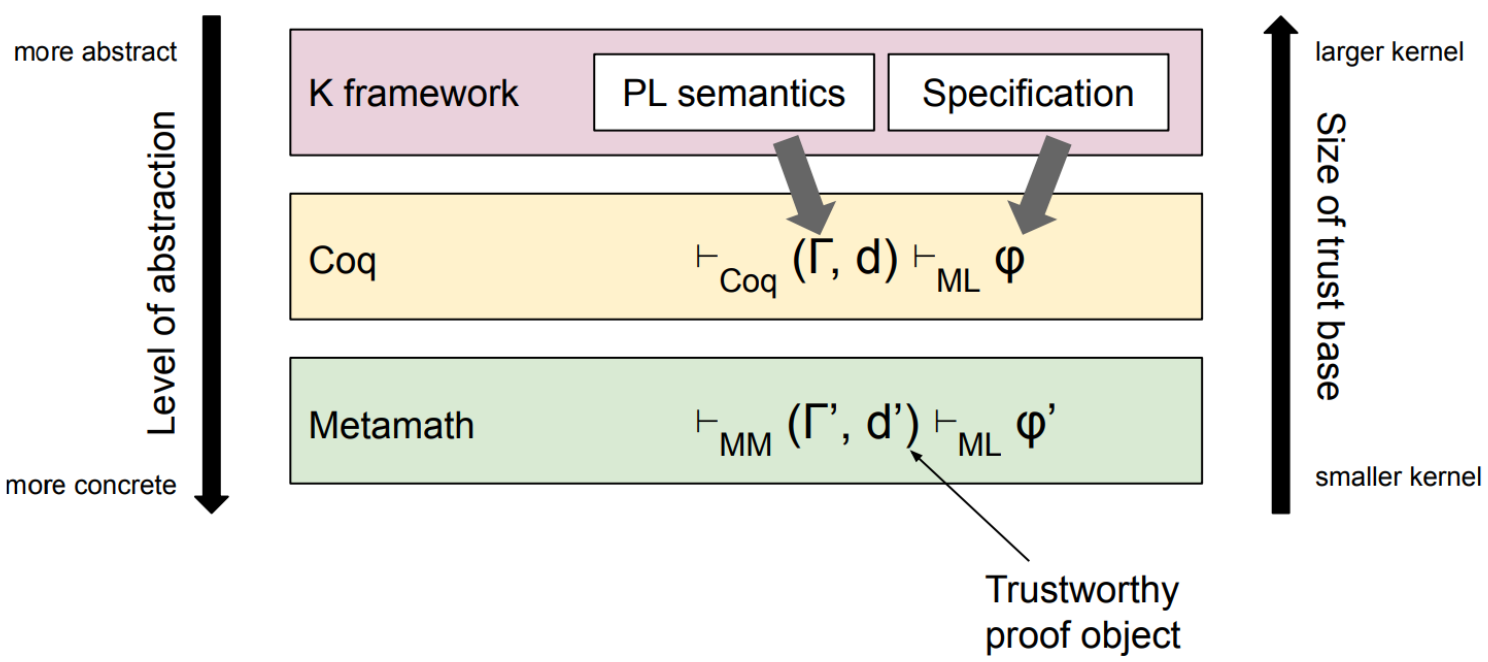
A formalization of ML

- In 2021, in collaboration with UIUC, we have developed an initial locally nameless formalization of ML in the Coq proof assistant
- Simple theories (such as FOL and LTL) and soundness were also formalized

Our partner (Runtime Verification Inc.) has offered a €100k grant for developing a complete framework around our ML formalization. The agreement has been signed in December.

- In 2021-H2, the focus was shifted towards usability and applicability
  - We keep working on improving the current formalization
  - And we are working on building a framework around the formalization

Outline of the new project



WP1: Matching Logic proof mode in Coq

WP2: Import K definitions into Coq-ML

WP3: Export Coq proofs into MetaMath

In focus

Formalization of Core Erlang: towards a complete language definition

Formalization of Matching Logic: towards a basis for semantics reasoning

Refactoring and LSP: a modern interface to applying Erlang refactoring

Erlang refactoring in VSCode

- Our long-term vision is to make verified refactoring steps for Erlang available in IDEs
- In 2021H2, we have been working on integrating an existing Erlang refactoring tool into VSCode via LSP (Language Server Protocol), in cooperation with the Erlang LS developers

The biggest challenge is handling interaction in an IDE-independent way

  - For instance, in case of folding or unfolding definitions
  - We are working on a generic solution based on transient files

- **Expected academic outcome is a TDK and/or a short paper**

In focus

Formalization of Core Erlang: towards a complete language definition

Formalization of Matching Logic: towards a basis for semantics reasoning

Refactoring and LSP: a modern interface to applying Erlang refactoring

Dissemination 2021H2

- Program Equivalence in an Untyped, Call-by-value Lambda Calculus with Uncurried Recursive Functions
  - Submitted to JLAMP

- Mechanizing Matching Logic In Coq
  - Rejected at CPP'21, to be submitted to ITP'22

- Public repositories on GitHub
  - https://github.com/harp-project

High Assurance Refactoring Project

Horpácsi Dániel
daniel-h@elte.hu

Az Alkalmazásiterület-specifikus nagy megbízhatóságú informatikai megoldások című projekt a Nemzeti Kutatási Fejlesztési és Innovációs Alapból biztosított támogatással, a Tématerületi kiválósági program (TKP2020-NKA-06, Nemzeti Kihívások Alprogram) finanszírozásában valósult meg.

NATIONAL RESEARCH, DEVELOPMENT AND INNOVATION OFFICE HUNGARY

PROGRAM FINANCED FROM THE NRDI FUND