보다 친숙한 Coq 증명을 위한 방안 연구 II

이계식
서울대, ROPAS

ROSAEC 워크샵, 통영

Outline

Motivation

How close are we to a world where every paper on programming languages is accompanied by an electronic appendix with machine-checked proofs?

(The POPLmark challenge)

Theorem provers

- Coq
- Isabelle/HOL
- Agda
- Twelf, ACL2, Nuprl, PVS, Mizar, ...

Theorem provers

- Coq
- Isabelle/HOL
- Agda
- Twelf, ACL2, Nuprl, PVS, Mizar, ...

Why Coq?

Because Sungwoo Park and I are Here!

A joke? No!

- Environment is a very important factor.
- Why Isabelle/HOL for L4.Verified project?
  - Experts were around!
- Coq Users @ Korea
  - Mailing lists in Korean
  - Wellcome any questions and suggestions

Why Coq?

Because Sungwoo Park and I are Here!

A joke? No!

- Environment is a very important factor.
- Why Isabelle/HOL for L4.Verified project?
  - Experts were around!
- Coq Users @ Korea
  - Mailing lists in Korean
  - Wellcome any questions and suggestions

Why Coq?

Because Sungwoo Park and I are Here!

A joke? No!

- Environment is a very important factor.
- Why Isabelle/HOL for L4.Verified project?
  - Experts were around!
- Coq Users @ Korea
  - Mailing lists in Korean
  - Wellcome any questions and suggestions

About Coq

- A functional programming language based on a very strong type theory
  - CIC (Calculus of Inductive Constructions)
- A formal proof management system
- A formal language to provide
  - mathematical definitions,
  - executable algorithms and theorems,
  - environments for semi-interactive development of machine-checked proofs.

About Coq

- A functional programming language based on a very strong type theory
  - CIC (Calculus of Inductive Constructions)
- A formal proof management system
- A formal language to provide
  - mathematical definitions,
  - executable algorithms and theorems,
  - environments for semi-interactive development of machine-checked proofs.

Criteria for an easy access to mechnization

- Cost of entry
  - how much does a user need to know in order to successfully develop a mechanization?
- Difficulty
  - in defining syntax and proving properties
  - POPLmark
- Efficiency
  - in handling of definitions and proofs
  - Appel and Leroy's CIVmark
- Transparency
  - how intuitive a formalization technique is.

Criteria for an easy access to mechnization

- Cost of entry
  - how much does a user need to know in order to successfully develop a mechanization?
- Difficulty
  - in defining syntax and proving properties
  - POPLmark
- Efficiency
  - in handling of definitions and proofs
  - Appel and Leroy's CIVmark
- Transparency
  - how intuitive a formalization technique is.

Criteria for an easy access to mechnization

- Cost of entry
  - how much does a user need to know in order to successfully develop a mechanization?
- Difficulty
  - in defining syntax and proving properties
  - POPLmark
- Efficiency
  - in handling of definitions and proofs
  - Appel and Leroy's CIVmark
- Transparency
  - how intuitive a formalization technique is.

Context-free Type Theory

joint work with Sungwoo Park, Jonghyun Park

Classical style of type theory

- Judgement form of type theory:

$$\Gamma \vdash M : A$$

- Γ contains the types of the variables occurring in terms:

$$\Gamma := x_1 : A_1, \ldots, x_n : A_n$$

- Γ binds the free variables.
- In this sense, a free variable is also globally bound.

STLC in the classical style

Types

$$\tau ::= \alpha \mid \tau \to \tau$$

Terms

$$e ::= x \mid e\ e \mid \lambda x : \tau . e$$

Typing rules

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma} \qquad \frac{\Gamma \vdash e_1 : \sigma \to \tau \quad \Gamma \vdash e_2 : \sigma}{\Gamma \vdash e_1\ e_2 : \tau} \qquad \frac{\Gamma, x : \sigma \vdash e : \tau}{\Gamma \vdash \lambda x : \sigma . e : \sigma \to \tau}$$

Context-free type theory

- Judgements has the shape of

$$M : A$$

- Each free variable is annotated with type information: x^{σ}
- The globally bound variables become really free.
- Geuvers et al. (LFMTP 2010): equivalence proof of both styles in case of PTS.
  - but focused on the equivalence, not on mechanization itself.

STLC without explicit contexts

Types

$$\tau ::= \alpha \mid \tau \to \tau$$

Terms

$$e ::= x \mid x^{\sigma} \mid e\, e \mid \lambda x : \tau . e$$

Typing rules

$$\frac{}{\vdash x^{\sigma} : \sigma} \qquad \frac{\vdash e_1 : \sigma \to \tau \quad \Gamma \vdash e_2 : \sigma}{\vdash e_1\, e_2 : \tau} \qquad \frac{e : \tau \quad x^{\sigma} \text{ fresh}}{\lambda y : \sigma . e[x^{\sigma} \backslash y] : \sigma \to \tau}$$

Saving lemmas

Weakening

$$\left.\begin{array}{r} \Gamma \subseteq \Delta \\ \Gamma \vdash M : A \end{array}\right\} \Rightarrow \Delta \vdash M : A$$

- (Weakening) need not be mentioned.
- Lemmas involving contexts
- and more?

Comparison

- context + Exists-Fresh
- context-free + Exists-Fresh
- context + cofinite
- context-free + cofinite

Comparison

- context + Exists-Fresh
- context-free + Exists-Fresh
- context + cofinite
- context-free + cofinite is the best? $\Rightarrow$ It depends.

Comparison

- context + Exists-Fresh
- context-free + Exists-Fresh
- context + cofinite
- context-free + cofinite is the best? $\Rightarrow$ It depends.

A Generic Formal Metatheory Framework

joint work with B. Oliveira, S. Cho, K. Yi, and S. Park

Main challenges

- Binding: Dealing with binding requires a lot of basic definitions and proofs.
- Problem: How to reuse prior definitions and proofs?

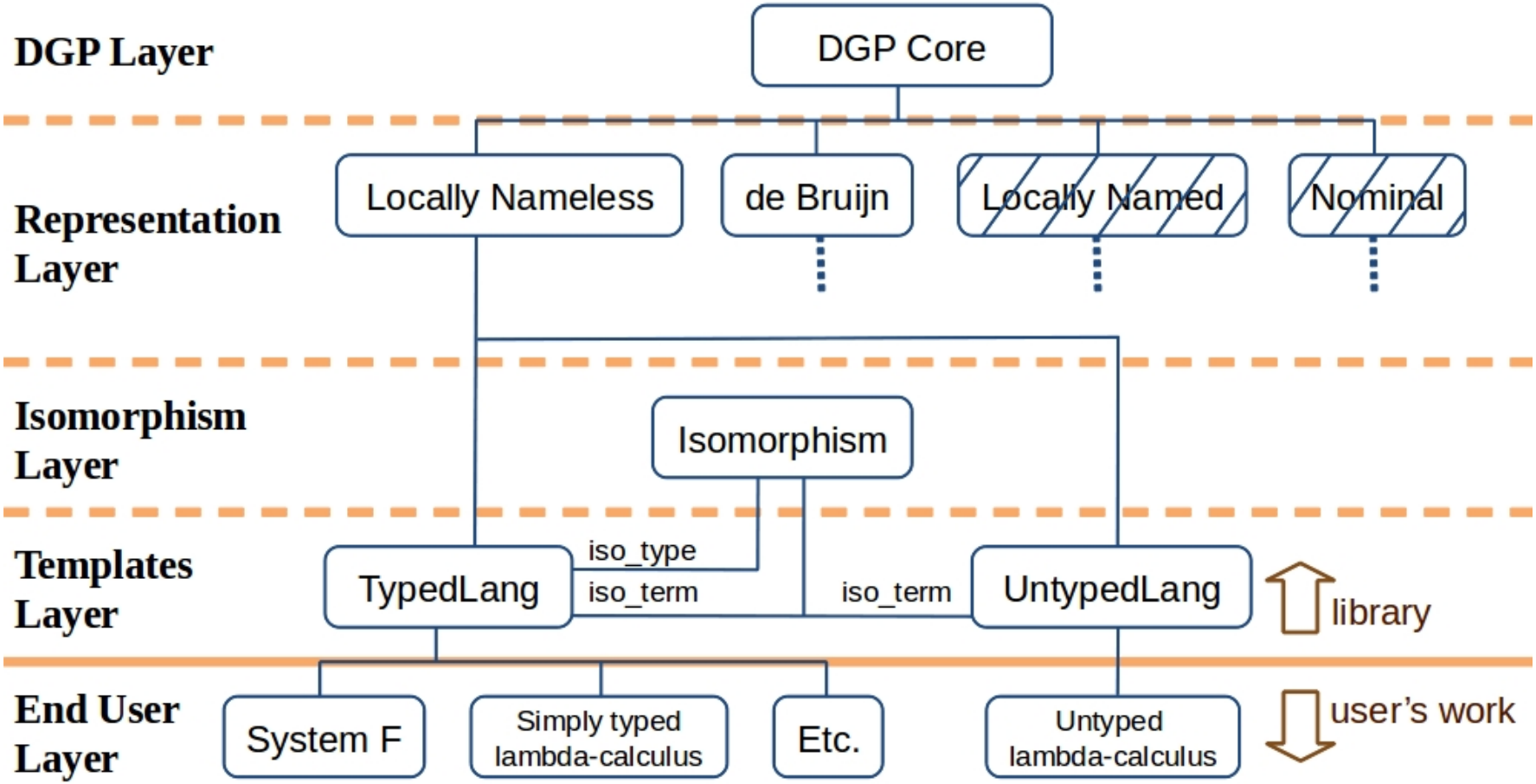
GMeta

- A generic metatheory library for first-order representations
- Infrastructure defined once and for all, and reused for each supported language.
- Parametrizable over
  - the object calculus/language
  - the type of the first-order representation

Infrastructure Overhead

- Common operations
  - free and bound variables; substitutions, α-conversion, etc.
- Lemmas about operations
  - permutations lemma, etc.

GMeta overview



Saving overhead

Cased studies compared against reference solutions by Aydemir et al. (2008):

		boilerplate	total
STLC	GMeta vs. Aydemir et al.	87.5 %	45%
$F_{<:}$	GMeta vs. Aydemir et al.	82 %	56%

Short- and mid-term goal was ...

- A slight extension of DGP core
  - to make the expressions more conventional
  - to include systems from logic and mathematics
- Extension of the meta-level library
  - support for locally-named approach
  - support for a variety of quantifications styles
  - support for multi-binders and mutually inductive definitions

Future work

- Support for the nominal approach, the most difficult way
- Combination of context-free style and GMeta

Conclusion

- Using GMeta or context-free style is not just about saving boilerplate.
- It also shows you what to do.
- Having a look at it would help you get an easier access to Coq.