The undecidability of finitary PCF in Coq

First Bachelor seminar talk

Fabian Brenner
Advisors: Yannick Forster, Dominik Kirst
Supervisor: Prof. Gert Smolka

April 30, 2024

Programming Systems Lab
Saarland University

Programming Computable Functions (PCF)

Definition

Extension of simply typed λ-calculus

$$T_1, T_2 := \mathtt{Nat} \mid T_1 \to T_2$$

$$s, t, u := \lambda x.\ s \mid s\ t \mid x \mid 0 \mid \mathtt{succ} \mid \mathtt{match}\ s\ \mathtt{with}\ (0 \Rightarrow t)\ (\mathtt{succ}\ n \Rightarrow u) \mid \mathtt{fix}\ f.\ s$$

Operational semantics

$$\mathtt{match}\ 0\ \mathtt{with}\ (0 \Rightarrow t)\ (\mathtt{succ}\ n \Rightarrow u) \to t \mid \ldots$$

Observation: If $\perp$ diverges, $\mathtt{match}\ \perp\ \mathtt{with}\ (0 \Rightarrow t)\ (\mathtt{succ}\ n \Rightarrow u)$ diverges.

- Introduced by Plotkin [1977] based on unpublished material of Scott [1993]
- Essential in the development of denotational semantics and domain theory
- Multiple models for PCF developed, none completely satisfactory

A long-standing open problem

Problem (Full abstraction problem of PCF)

Is there a fully abstract model of PCF that is concrete and independent of syntax?

Minimal requirement for such a model: For PCF_2, presentation should only require computable operations on finitely represented objects belonging to a decidable set

Definition (PCF_2)

Extension of simply typed λ-calculus

$$T_1, T_2 := \cancel{\mathtt{Nat}}\ \mathtt{Bool} \mid T_1 \to T_2$$

$$s, t, u := \lambda x.\ s \mid s\ t \mid x \mid \mathtt{true} \mid \mathtt{false} \mid \cancel{\mathtt{fix}\ f.\ s}\ \bot$$

$$\mid \mathtt{match}\ s\ \mathtt{with}\ (\mathtt{true} \Rightarrow t)\ (\mathtt{false} \Rightarrow u)$$

Operational semantics

$$\mathtt{match}\ \mathtt{true}\ \mathtt{with}\ (\mathtt{true} \Rightarrow t)\ (\mathtt{false} \Rightarrow u) \to t,$$

$$\mathtt{match}\ \bot\ \mathtt{with}\ (\mathtt{true} \Rightarrow t)\ (\mathtt{false} \Rightarrow u) \to \bot \mid \ldots$$

Solution of the problem

Theorem (Loader 2000)

No fully abstract model of PCF fulfilling the minimal criterion exists.

Recap (Minimal criterion)

For PCF_2, presentation should only require computable operations on finitely represented objects belonging to a decidable set.

Definition

A model of a PCF is called **fully abstract** iff: Terms s, t have same representation in model iff they are contextually equivalent

Contextual equivalence

Theorem (Loader 2000)

Contextual equivalence of PCF_2 is undecidable.

Recap (PCF_2)

<u>Extension of simply typed λ-calculus</u>

$$s, t, u := \lambda x.\ s \mid s\ t \mid x \mid \texttt{true} \mid \texttt{false} \mid \bot \mid \texttt{match}\ s\ \texttt{with}\ (\texttt{true} \Rightarrow t)\ (\texttt{false} \Rightarrow u)$$

$$T_1, T_2 := \texttt{Bool} \mid T_1 \to T_2$$

Definition

Two terms $\Gamma \vdash s, t : A$ are **contextually equivalent**:

$$\forall C v.\ C : (\Gamma, A) \rightsquigarrow (\emptyset, \texttt{Bool}) \longrightarrow C[s] \Downarrow v \longleftrightarrow C[t] \Downarrow v$$

Proof of Loader's theorem

Theorem (Loader 2000)

Contextual equivalence of PCF_2 is undecidable.

$\leq_m$: many-one reducible

SR: Word problem for string rewriting systems

CE: Contextual equivalence on PCF_2

$$\mathrm{SR} \leq_m \text{CIE-SYS} \leq_m \text{CE-SYS} \leq_m \mathrm{CE}$$

- "the proof is long and technical, and consists of intricate syntactic arguments" [Higher-Order Computability: Longley, Normann]

Surprising result

- PCF_2 is strongly normalizing
- Related problems are decidable:
  - Contextual equivalence in PCF_1 [Loader 1998, Schmidt-Schauß 1999]
  - Contextual equivalence of the simply typed λ-calculus with finite types [Ghani 1995, Scherer 2016]

Decidability for PCF_1

Definition (PCF_1)

Extension of simply typed λ-calculus

$$T_1, T_2 := \texttt{Unit} \mid T_1 \to T_2$$

$$s, t, u := \lambda x.\ s \mid s\ t \mid x \mid () \mid \bot \mid \texttt{match}\ s\ \texttt{with}\ () \Rightarrow t$$

Operational semantics

$$(\texttt{match}\ ()\ \texttt{with}\ () \Rightarrow t) \to t, \quad (\texttt{match}\ \bot\ \texttt{with}\ () \Rightarrow t) \to \bot$$

Theorem (Loader 1998, Schmidt-Schauß 1999)

Contextual equivalence of PCF_1 is decidable.

- For each type T, a finite set of closed expressions representing all equivalence classes of closed expressions is computable does not work for PCF_2

Decidability of STLC with finite types

Theorem (Ghani 1995, Scherer 2016)

Contextual equivalence of the simply typed λ-calculus with finite types is decidable.

Definition

$\beta\eta$-equivalence is the equivalence closure of β- and η-conversions

Proof sketch

1. $\beta\eta$-equivalence is decidable (since system is strongly normalizing!)
2. $\beta\eta$-equivalence coincides with contextual equivalence

Course of action

- Formalisation Loader's result in Coq
- Synthetic computability: In Coq definable the same as computable
- No need to define notion of computability, use definability in Coq
- Use Autosubst 2 and de Bruijn terms to automate substitutions
- Contribute to Coq Library of Undecidability Proofs

Current state and outlook

Reduction chain in Loader's proof:

$$\text{SR} \leq_m \text{CIE-SYS} \leq_m \text{CE-SYS} \leq_m \text{CE}$$

SR: Word problem for string rewriting systems

CE: Deciding contextual equivalence on PCF_2

So far:

- Formalised PCF_2 in Coq
- Understood blue and green reductions and formalised green reduction in Coq
- Formalisation of contextual equivalence in Coq

To do:

- Formalise blue reduction in Coq
- Deepen understanding of red reduction
- Formalise red reduction in Coq

Goals and key take-aways

Recap of Loader's result

- Solved important problem
- Technical and intricate proof
- Surprising result

Goals of this project

- Formalisation of Loader's result in Coq
- Clarification of reduction from string rewriting
- Not a formalisation of consequences or related results

References 1

- [Plotkin 1977] G.D. Plotkin, LCF considered as a programming language, Theoretical Computer Science, Volume 5, Issue 3, 1977, Pages 223-255, ISSN 0304-3975, https://doi.org/10.1016/0304-3975(77)90044-5.
- [Scott 1993] Dana S. Scott, A type-theoretical alternative to ISWIM, CUCH, OWHY, Theoretical Computer Science, Volume 121, Issues 1–2, 1993, Pages 411-440, ISSN 0304-3975, https://doi.org/10.1016/0304-3975(93)90095-B.
- [Higher-Order Computability: Longley, Normann] John Longlay, Dag Normann, Higher-Order-Computability, Chapter 7, Theorem 7.5.22, 2015, Page 342, ISSN 2190-619X

References 2

- [Loader 2000] Ralph Loader, Finitary PCF is not decidable, Theoretical Computer Science, Volume 266, Issues 1–2, 2001, Pages 341-364, ISSN 0304-3975, https://doi.org/10.1016/S0304-3975(00)00194-8.
- [Loader 1998] Ralph Loader, Unary PCF is decidable, Theoretical Computer Science, Volume 206, Issues 1–2, 1998, Pages 317-329, ISSN 0304-3975, https://doi.org/10.1016/S0304-3975(98)00048-6.
- [CLUB] Coq Library of Undecidability Proofs, https://github.com/uds-psl/coq-library-undecidability

References 3

- [Schmidt-Schauß 1999] Manfred Schmidt-Schauß, Decidability of behavioural equivalence in unary PCF, Theoretical Computer Science, Volume 216, Issues 1–2, 1999, Pages 363-373, ISSN 0304-3975, https://doi.org/10.1016/S0304-3975(98)00024-3.
- [Ghani 1995] Neil Ghani, Beta-Eta Equality for Coproducts, TLCA, 1995.
- [Scherer 2017] Gabriel Scherer, Deciding equivalence with sums and the empty type, POPL '17, 2017, Pages 374-386, ISSN 0362-1340, https://doi.org/10.1145/3009837.3009901.

Counterexample

$t_b := \lambda f.\, \texttt{match}\ (f \bot)\ \texttt{with}\ (\texttt{true} \Rightarrow f\ b)\ (\texttt{false} \Rightarrow f\ b)$ for $b \in \{\texttt{true}, \texttt{false}\}$

- In PCF_2, t_b either returns $\bot$ or f is constant

$\Rightarrow$ Contextually equivalent in PCF_2

- Not $\eta\beta$-equivalent
- Canonical translation to STLC with finite typed can get an f as input that is not constant but returns true or false on input $\bot$

$\Rightarrow$ Translation not contextually equivalent in STLC with finite types

Reduction to STLC with finite types

Recap (Ghani 1995, Scherer 2016)

Contextual equivalence of the simply typed λ-calculus with finite types is decidable.

Idea: Many-one reduction to contextual equivalence in this system

- Embed PCF_2 into STLC with finite types: model `bool` as sum type with three elements, etc.
- `match` in PCF_2 only permits returning $\bot$ if input is $\bot$, match in STLC has no such restriction
- Counterexample for correctness can be constructed

Decidability problems in Loader's proof

Definition

$\lesssim$: Loader's contextual pre-order, $\simeq$: contextual equivalence.

- CE: Given PCF terms s, t, decide if they are contextually equivalent.
- CE-SYS: Given finitely many pairs of PCF terms $s_i : \mathcal{B}$ and $b_i \in \mathit{true}, \mathit{false}$, decide if each $s_i \simeq b_i$.
- CIE-SYS: Given finitely many pairs of PCF terms $s_i, t_i : \mathcal{B}$, decide if each $s_i \lesssim t_i$.
- SR: Given string rewriting system (Σ, R) and two words W_0, W over Σ, decide if W is derivable from W_0 using the rules R.