Intuitionistic Fixed Point Logic and Program Extraction (with Prawf)

Prifysgol Abertawe
Swansea University

Olga Petrovska
(joint work with Ulrich Berger (SU) and Hideki Tsuiki (Kyoto University))

6-8 April, BCTCS 2020 (Coronavirus Edition )

This work was supported by the Marie Curie International Research Staff Exchange Schemes *Computable Analysis* (PIRSES-GA-2011-294962) and *Correctness by Construction* (FP7-PEOPLE-2013-IRSES-612638) as well as the Marie Curie RISE project *Computing with Infinite Data* (H2020-MSCA-RISE-2016-731143) and the EPSRC Doctoral Training Grant No. 1818640.

Motivation

Creating a formal system exploiting Curry-Howard isomorphism to extract useful and ‘correct-by-construction’ programs from proofs about abstract mathematics.

Motivation

Creating a formal system exploiting Curry-Howard isomorphism to extract useful and ‘correct-by-construction’ programs from proofs about abstract mathematics.

Existing systems:

Motivation

Creating a formal system exploiting Curry-Howard isomorphism to extract useful and ‘correct-by-construction’ programs from proofs about abstract mathematics.

Existing systems:

- Minlog (H. Schwichtenberg): `gsso9..vvv-l`sgdl`shj-tmh,ltdmbgdm-cd.}knfhj.lhmknf.hmcdw-ogo`

Motivation

Creating a formal system exploiting Curry-Howard isomorphism to extract useful and ‘correct-by-construction’ programs from proofs about abstract mathematics.

Existing systems:

- Minlog (H. Schwichtenberg): `gsso9..vvv-l`sgdl`shj-tmh,ltdmbgdm-cd.}knfhj.lhmknf.hmcdw-ogo`

- Nuprl, Isabelle, Coq etc.

Motivation

Creating a formal system exploiting Curry-Howard isomorphism to extract useful and ‘correct-by-construction’ programs from proofs about abstract mathematics.

Existing systems:

- Minlog (H. Schwichtenberg): `gsso9..vvv-l`sgdl`shj-tmh,ltdmbgdm-cd.}knfhj.lhmknf.hmcdw-ogo`
- Nuprl, Isabelle, Coq etc.
- Prawf *NEW*

Agenda

- Intuitionistic Fixed Point Logic
- Realizability
- Soundness
- Demo

Intuitionistic Fixed Point Logic (IFP) as a schema

First-order logic with lambda abstractions and fixed point operators

IFP is a schema

(1) *Sorts* $\iota, \iota_1, \ldots$ as names for spaces of abstract mathematical objects.

(2) *Terms* $(\vec{t})$ that include *variables*, *constants* of fixed sorts and *function symbols* types $\vec{\iota} \to \iota'$.

(3) *Predicate constants* of fixed arities $(\vec{\iota})$.

$$
\begin{array}{lll}
\text{Formulas } A, B & ::= & P(\vec{t}) \\
 & \mid & A \wedge B \mid A \vee B \mid A \to B \mid \forall x\, A \mid \exists x\, A \\
\text{Predicates } P, Q & ::= & X \mid P \mid \lambda x\, A \mid \\
\text{Operators} & ::= & X\, P \quad (P \text{ is strictly positive in } X)
\end{array}
$$

Intuitionistic Fixed Point Logic (IFP)

- Intuitionistic Predicate Logic

  Natural deduction with equality

- Inductions and Coinduction

$$\frac{}{(\quad)}\ \mathrm{cl} \qquad \frac{(P) \quad P}{P}\ \mathrm{ind}$$

$$\frac{}{(\quad)}\ \mathrm{cocl} \qquad \frac{P \qquad (P)}{P}\ \mathrm{coind}$$

- Axioms consisting of closed disjunction-free formulas

  e.g., $\forall x\, y(x + y = y + x)$

Realizability and RIFP

A *realizer* is an object that "realizes" a formula from a formal theory, i.e. serves as a confirmation of its truth.

IFP for Realisers (RIFP)

The Scott domain of realizers is defined by the recursive domain equation

$$D = \mathbf{Nil} + \mathbf{Lt}(D) + \mathbf{Rt}(D) + \mathbf{Pair}(D \quad D) + \mathbf{F}(D \;\; ! \;\; D)$$

where $+$ denotes the separated sum, $\quad$ the Cartesian product and $D \;\; ! \;\; D$ is the continuous function space.

Non-Computational and Harrop Expressions

A *Harrop* expression contains no disjunction or free predicate variable at a strictly positive position [1].

A *non-computational* expression contains neither disjunctions nor free predicate variable.

[1] predicate variable is not free in the premise of an implication

Realizability and Simplified Realizability

We assing to every

- non-Harrop formula A a predicate $\mathbf{R}(A)$ with one argument for realizers
- non-Harrop predicate P a predicate $\mathbf{R}(P)$ with an extra argument for realizers
- non-Harrop operator an operator $\mathbf{R}(\ )$ with an extra argument for realizers

- Harrop formula A a formula $\mathbf{H}(A)$
- Harrop predicate P a predicate $\mathbf{H}(P)$ of the same arity
- Harrop operator an operator $\mathbf{H}(\ )$ of the same arity

Realizability interpretation

$$
\begin{aligned}
a \,\mathbf{r}\, A &= \mathbf{H}(A) \wedge a = \mathbf{Nil} \ (A \text{ Harrop}) \\
\mathbf{R}(P) &= \lambda(\vec{x}, a)\,(\mathbf{H}(P) \wedge a = \mathbf{Nil}) \ (P \text{ Harrop}) \\
&\text{Otherwise} \\
a \,\mathbf{r}\, P(\vec{t}) &= \mathbf{R}(P)(\vec{t}, a) \qquad\qquad \mathbf{H}(P(\vec{t})) = \mathbf{H}(P)(\vec{t}) \\
c \,\mathbf{r}\, (A \wedge B) &= ((\pi_l c) \,\mathbf{r}\, A \wedge (\pi_r c) \,\mathbf{r}\, B) \qquad \mathbf{H}(A \wedge B) = \mathbf{H}(A) \wedge \mathbf{H}(B) \\
&\quad (\text{neither } A \text{ nor } B \text{ Harrop}) \\
a \,\mathbf{r}\, (A \wedge B) &= a \,\mathbf{r}\, A \wedge \mathbf{H}(B) \quad (B \text{ Harrop}) \\
b \,\mathbf{r}\, (A \wedge B) &= \mathbf{H}(A) \wedge b \,\mathbf{r}\, B \quad (A \text{ Harrop}) \\
c \,\mathbf{r}\, (A \vee B) &= \exists a\,(c = \mathbf{Lt}(a) \wedge a \,\mathbf{r}\, A \vee c = \mathbf{Rt}(a) \wedge a \,\mathbf{r}\, B) \\
f \,\mathbf{r}\, (A \to B) &= \forall a\,(a \,\mathbf{r}\, A \to (f\, a) \,\mathbf{r}\, B)) \qquad \mathbf{H}(A \to B) = \mathbf{r}\, A \to \mathbf{H}(B) \\
&\quad (\text{neither } A \text{ nor } B \text{ Harrop}) \\
b \,\mathbf{r}\, (A \to B) &= \mathbf{H}(A) \to b \,\mathbf{r}\, B \quad (A \text{ Harrop}) \\
a \,\mathbf{r}\, \Diamond x\, A &= \Diamond x\, (a \,\mathbf{r}\, A) \quad (\Diamond \in \{\forall, \exists\}) \qquad \mathbf{H}(\Diamond x\, A) = \Diamond x\, \mathbf{H}(A) \\
\mathbf{R}(X) &= \tilde{X} \qquad\qquad \mathbf{H}(P) = P \\
\mathbf{R}(\Diamond\, \Phi) &= \Diamond\, \mathbf{R}(\Phi) \quad (\Diamond \in \{\nu, \mu\}) \qquad \mathbf{H}(\Diamond\, \Phi) = \Diamond\, \mathbf{H}(\Phi) \\
\mathbf{R}(\lambda \vec{x}\, A) &= \lambda \vec{x}\, \mathbf{R}(A) \quad (= \lambda(\vec{x}, a)\, a \,\mathbf{r}\, A) \qquad \mathbf{H}(\lambda \vec{x}\, A) = \lambda \vec{x}\, \mathbf{H}(A) \\
\mathbf{R}(\lambda X\, P) &= \lambda \tilde{X}\, \mathbf{R}(P) \qquad\qquad \mathbf{H}(\lambda X\, P) = \lambda X\, \mathbf{H}_X(P)
\end{aligned}
$$

Soundness

$$;\quad \vdash_{IFP} A \quad) \ \mathsf{H}(\ );\varpi\, \mathsf{r} \quad \vdash_{RIFP} p\, \mathsf{r}\, A, \text{ where } \mathsf{S}\text{-}(p) \quad \varpi:$$

*The admissibility condition is that either and P are both Harrop or both non-Harrop or is Harrop and *simple* and P is non-Harrop. Simple means that no sub-expression (of an expression in question) of a form or contains a predicate variable X free.

IFP' and the Soundness Theorem

Hideki Tsuiki suggested creating IFP' to get rid of the admissibility restriction. This also proved to be useful for simplifying program extraction implementation.

Monotonicity of the operator :

$$\mathrm{Mon}(\ ) \stackrel{\mathrm{Def}}{=} X \quad Y \ ! \quad (X) \qquad (Y)$$

where X and Y are fresh variables.

$$\frac{(P) \quad P \quad \mathrm{Mon}(\ )}{(\ ) \quad P}\ \mathrm{IND'}(\ ; P) \quad (\ )$$

$$\frac{P \qquad (P) \quad \mathrm{Mon}(\ )}{P \qquad (\ )}\ \mathrm{COIND'}(\ ; P) \quad (\ )$$

( ) free assumptions in the proof of Mon( ) must not contain X or Y free.

Soundness proof i

; `$_{IFP^0}$ $A$ ) H( );¤ r `$_{RIFP}$ p r A, where S - (p) ¤:

Proof by induction on the length of IFP' derivations.

Soundness proof ii

$\underline{Ind^0}$. Assume $\vdash_{IFP^0} (\quad (P) \quad P)$, where $\quad (P) = Q[P{=}X]$ and $\vdash_{IFP^0} Mon(\quad)$, i.e. $X \quad Y \; ! \; Q \quad Q[Y{=}X]$.

- I.h.$_{1nH}$ $\vdash_{RIFP} s\,\mathbf{r}(\quad (P) \quad P)$;
- I.h.$_{mon_{nH}}$ $\vdash_{RIFP} m\,\mathbf{r}(Mon(\quad))$;

If $\underline{\quad}$ and P are non-Harrop show:

$$
\begin{array}{rrlll}
 & f\,\mathbf{r}(\quad(\quad)\quad P) & & & \\
 & \mathbf{R}(\quad) \quad f^{\,1} & \mathbf{R}(P) & f\,\mathbf{r}(Q \quad P) \quad \mathbf{R}(Q) \quad f^{\,1} & \mathbf{R}(P) \\
= & \mathbf{R}(\quad(\quad X\,Q)) \quad f^{\,1} & \mathbf{R}(P) & \text{since} \quad = \quad X\,Q & \\
= & (\quad(\quad \not{X}\mathbf{R}(Q))) \quad f^{\,1} & \mathbf{R}(P) & \text{since } \mathbf{R}(\quad) = \quad(\mathbf{R}(\quad)) & \\
 & & & \text{and } \mathbf{R}(\quad X\,Q) = \quad \not{X}(\mathbf{R}(Q)) &
\end{array}
$$

* Proven by a separate lemma, which includes a number of equivalences like above

Soundness proof iii

By s.p. induction, it is enough to show

$$\mathbf{R}(Q)[f^{-1}\ \ \mathbf{R}(P)/\not{X}]\quad f^{-1}\ \ \mathbf{R}(P) \tag{1}$$

By i.h.$_{1nH}$ we have: $s\ \mathbf{r}\ (\quad (P)\quad (P))$, which is equivalent to

$$\mathbf{R}(Q[P/X])\quad s^{-1}\ \ \mathbf{R}(P) \tag{2}$$

By i.h.$_{mon_{nH}}$ we have $m\ \mathbf{r}\ Mon(\quad)$ and by Lemma (a) this implies

$$m\ \mathbf{r}\ (Mon(\quad)[P/Y]) \tag{3}$$

Writing out $Mon(\quad)[P/Y]$ we obtain $X\quad P\ !\ Q\quad Q[P/X]$. Hence, 3 can be rewritten as

$$\begin{aligned}
&\forall g(g\ \mathbf{r}\ (X\quad P)\ !\ (m\ g)\ \mathbf{r}\ (Q\quad Q[P/X])) \\
&\forall g(\mathbf{R}(X)\quad g^{-1}\ \ \mathbf{R}(P)\ !\ \mathbf{R}(Q)\quad (m\ g)^{-1}\ \ \mathbf{R}(Q[P/X])) && \text{by the equivalences lemma} \\
=\ &\forall g(\not{X}\quad g^{-1}\ \ \mathbf{R}(P)\ !\ \mathbf{R}(Q)\quad (m\ g)^{-1}\ \ \mathbf{R}(Q[P/X])) && \text{by def. of } \mathbf{R}(X)
\end{aligned}$$

(a) If RIFP proves $a\ \mathbf{r}\ A$ from assumptions that do not contain the predicate variable X and if P is a non-Harrop predicate of the same arity as X, then RIFP proves $a\ \mathbf{r}\ (A[P/X])$ from the same assumptions.

Soundness proof iv

$\forall g(\mathcal{X} \quad g \ ^1 \ \mathsf{R}(P) \ ! \ \mathsf{R}(Q) \quad (m\ g) \ ^1 \ \mathsf{R}(Q[P{=}X]))$

If we define g as f and $\mathcal{X} = f \ ^1 \ \mathsf{R}(P)$ and use Lemma (b), we get

$$\begin{aligned} \mathsf{R}(Q)[f \ ^1 \ \mathsf{R}(P){=}\mathcal{X}] \quad & (m\ f) \ ^1 \ \mathsf{R}(Q[P{=}X])) \\ & (m\ f) \ ^1 \ (s \ ^1 \ \mathsf{R}(P)) \qquad \text{by 2} \\ = \ & (s \quad m\ f) \ ^1 \ \mathsf{R}(P) \qquad \text{by the equivalences lemma} \end{aligned}$$

Hence, the realiser is recursively defined as $f = s \quad m\ f$

(b) If IFP, IFP', or RIFP proves $\vdash A$, then the same system proves $[P{=}X] \vdash A[P{=}X]$, $[P{=}X] \vdash A[P{=}X]$, where A, P, X are arbitrary formulas, predicates, predicate variables, respectively, and $\mathcal{X}$ is an arbitrary predicate constant that does not appear in any axiom.

Key points before the demo

- IFP is a scheme

  more flexibility, abstraction (e.g., list reversal, translation between representations)
- Use of classical logic as long as it is disjunction-free
- Prawf is build specifically for the purpose of program extraction

Demo

Future work

- Extensions for sequent calculus proofs (Yvett Szilagyi)
- Extension for CFP (Concurrent Fixed Point Logic)
- Developing theorems database in Prawf

References

U. Berger, P. O., and H. Tsuiki.
Prawf: An interactive proof system for program extraction.
To be published in proceedings of 16th Conference on Computability in Europe, CiE, 2020.

U. Berger and O. Petrovska.
Optimised program extraction for induction and coinduction.
In *Sailing Routes in the World of Computation: 14th Conference on Computability in Europe, CiE 2018, Kiel, Germany, July 30 – August 3*, pages 70–80, 2018.

U. Berger and H. Tsuiki.
Intuitionistic fixed point logic.
Unpublished manuscript available on ArXiv, 2019.

Prawf: `gssor9..oq`vesqdd-vnqcoqdrr-bnl.`

Thank you