IFP style proofs in the Coq proof assistant

Holger Thies
Kyoto University
Joint work with Sewon Park and Hideki Tsuiki

Continuity, Computability, Constructivity 2021
September 19-24, Online

IFP and Coq

IFP	Coq
• Intutionistic first-order logic	• Constructive Type Theory
• Can add (nc) axioms	• Can add axioms
• Program Extraction	• Program Extraction, Computation
• Well-suited for proofs over abstract mathematical spaces (real numbers,...)	• General purpose, many libraries
• Partiality	• Only total functions

IFP has been developed by U. Berger and collaborators (M. Seisenberger, O. Petrovska, H. Tsuiki) since 2009.

IFP Overview

IFP is a schema for a proof system over different abstract mathematical spaces.

IFP Overview

IFP is a schema for a proof system over different abstract mathematical spaces.

An instance of IFP consists of a language $\mathcal{L}$ and a set of axioms $\mathcal{A}$.

IFP Overview

IFP is a schema for a proof system over different abstract mathematical spaces.
An instance of IFP consists of a language $\mathcal{L}$ and a set of axioms $\mathcal{A}$.
The Language $\mathcal{L}$ consists of

1. Sorts $\iota, \iota_1, \ldots$ as names for abstract mathematical spaces

IFP Overview

IFP is a schema for a proof system over different abstract mathematical spaces.
An instance of IFP consists of a language $\mathcal{L}$ and a set of axioms $\mathcal{A}$.
The Language $\mathcal{L}$ consists of

1. Sorts $\iota, \iota_1, \ldots$ as names for abstract mathematical spaces
2. Terms s, t consisting of variables, constants and function symbols

IFP Overview

IFP is a schema for a proof system over different abstract mathematical spaces.
An instance of IFP consists of a language $\mathcal{L}$ and a set of axioms $\mathcal{A}$.
The Language $\mathcal{L}$ consists of

1. Sorts $\iota, \iota_1, \ldots$ as names for abstract mathematical spaces
2. Terms s, t consisting of variables, constants and function symbols
3. Predicate constants of fixed arity

IFP Overview

IFP is a schema for a proof system over different abstract mathematical spaces.

An instance of IFP consists of a language $\mathcal{L}$ and a set of axioms $\mathcal{A}$.

The Language $\mathcal{L}$ consists of

1. Sorts $\iota, \iota_1, \ldots$ as names for abstract mathematical spaces
2. Terms s, t consisting of variables, constants and function symbols
3. Predicate constants of fixed arity

IFP Overview

IFP is a schema for a proof system over different abstract mathematical spaces.
An instance of IFP consists of a language $\mathcal{L}$ and a set of axioms $\mathcal{A}$.
The Language $\mathcal{L}$ consists of

1. Sorts $\iota, \iota_1, \ldots$ as names for abstract mathematical spaces
2. Terms s, t consisting of variables, constants and function symbols
3. Predicate constants of fixed arity

Relative to $\mathcal{L}$ we define

- Formulas: $s = t$, $P(t)$, $A \land B$, $A \lor B$, $A \to B$, $\forall x\, A$, $\exists x\, A$.

IFP Overview

IFP is a schema for a proof system over different abstract mathematical spaces.
An instance of IFP consists of a language $\mathcal{L}$ and a set of axioms $\mathcal{A}$.
The Language $\mathcal{L}$ consists of

1. Sorts $\iota, \iota_1, \ldots$ as names for abstract mathematical spaces
2. Terms s, t consisting of variables, constants and function symbols
3. Predicate constants of fixed arity

Relative to $\mathcal{L}$ we define

- Formulas: $s = t$, $P(t)$, $A \wedge B$, $A \vee B$, $A \to B$, $\forall x\, A$, $\exists x\, A$.
- Predicates: Variables, constants, $\lambda x\, A$, $\mu(\Phi)$, $\nu(\Phi)$.

IFP Overview

IFP is a schema for a proof system over different abstract mathematical spaces.
An instance of IFP consists of a language $\mathcal{L}$ and a set of axioms $\mathcal{A}$.
The Language $\mathcal{L}$ consists of

1. Sorts $\iota, \iota_1, \ldots$ as names for abstract mathematical spaces
2. Terms s, t consisting of variables, constants and function symbols
3. Predicate constants of fixed arity

Relative to $\mathcal{L}$ we define

- Formulas: $s = t$, $P(t)$, $A \wedge B$, $A \vee B$, $A \to B$, $\forall x\, A$, $\exists x\, A$.
- Predicates: Variables, constants, $\lambda x\, A$, $\mu(\Phi)$, $\nu(\Phi)$.
- Operators: $\lambda X\, P$ (P strictly positive in X)

Axioms

A formula is called non-computational if it is disjunction-free and contains no free predicate variables.

Axioms

A formula is called <u>non-computational</u> if it is disjunction-free and contains no free predicate variables.

Axioms in IFP are closed non-computational (disjunction-free) formulas.

Example:

$$\forall x \, \forall y \, \neg(x < y) \to y \le x$$

is OK but

$$\forall x \, \forall y \, x < y \lor y \le x$$

is not.

Induction rules

IFP contains the following rules for strictly positive induction and coinduction:

$$\frac{}{\Phi(\mu(\Phi)) \subseteq \mu(\Phi)}\ \mathrm{CL}(\Phi) \qquad \frac{\Phi(P) \subseteq P}{\mu(\Phi) \subseteq P}\ \mathrm{IND}(\Phi, P)$$

$$\frac{}{\nu(\Phi) \subseteq \Phi(\nu(\Phi))}\ \mathrm{COCL}(\Phi) \qquad \frac{P \subseteq \Phi(P)}{P \subseteq \nu(\Phi)}\ \mathrm{COIND}(\Phi, P)$$

$A \subseteq B$ is short for $\forall x\, A\, x \to B\, x$.

Program Extraction

- IFP's primary goal is program extraction

Program Extraction

- IFP's primary goal is program extraction
- Careful distinction between computational and non-computational (Harrop) formulas

Program Extraction

- IFP's primary goal is program extraction
- Careful distinction between computational and non-computational (Harrop) formulas
- Harrop expressions are realized by **Nil**

Program Extraction

- IFP's primary goal is program extraction
- Careful distinction between computational and non-computational (Harrop) formulas
- Harrop expressions are realized by **Nil**
- Uniform interpretation of quantifiers:

$$a\ R\ \Diamond x\, A = \Diamond x\ (a\ R\ A) \text{ for } \Diamond \in \{\exists, \forall\}$$

Program Extraction

- IFP's primary goal is program extraction
- Careful distinction between computational and non-computational (Harrop) formulas
- Harrop expressions are realized by **Nil**
- Uniform interpretation of quantifiers:

$$a\ R\ \Diamond x\, A = \Diamond x\ (a\ R\ A) \text{ for } \Diamond \in \{\exists, \forall\}$$

- Implemented in the Prawf proof assistant (U. Berger, O. Petrovska, H. Tsuiki)

Example: The language of real numbers

Language:

- Sorts: one sort R
- Constants: $-1, 0, 1$
- Functions: $+, -, *, /, \ldots$
- Predicate constants: $<, \leq$

Axioms:

- Disjunction-free formulation of axioms of real closed field etc.

Example: Natural numbers

We can define natural numbers inductively by

$$N(x) = \mu(\lambda X \lambda x \, (x = 0 \vee X(x-1)))$$

Induction and Closure rules:

$$\frac{}{\forall x \, ((x = 0 \vee N(x-1)) \to N(x))} \ \mathrm{CL}(N)$$

$$\frac{\forall x \, (x = 0 \vee P(x-1)) \to P(x)}{\forall x \, N(x) \to P(x)} \ \mathrm{IND}(N, P)$$

IFP-Coq

For each language $\mathcal{L}$ and set of axioms $\mathcal{A}$, we define a set of Coq axioms by

1. For each sort ι in $\mathcal{L}$, define ι as a term constant (axiom) of Prop.

IFP-Coq

For each language $\mathcal{L}$ and set of axioms $\mathcal{A}$, we define a set of Coq axioms by

1. For each sort ι in $\mathcal{L}$, define ι as a term constant (axiom) of Prop.
2. For each constant c of sort ι, define c as a term constant (axiom) of type ι.

IFP-Coq

For each language $\mathcal{L}$ and set of axioms $\mathcal{A}$, we define a set of Coq axioms by

1. For each sort ι in $\mathcal{L}$, define ι as a term constant (axiom) of Prop.
2. For each constant c of sort ι, define c as a term constant (axiom) of type ι.
3. For each function symbol f of arity $\iota_1 \times \cdots \times \iota_n \to \iota$, define f as a term constant (axiom) of type $\iota_1 \to \cdots \to \iota_n \to \iota$.

IFP-Coq

For each language $\mathcal{L}$ and set of axioms $\mathcal{A}$, we define a set of Coq axioms by

1. For each sort ι in $\mathcal{L}$, define ι as a term constant (axiom) of Prop.
2. For each constant c of sort ι, define c as a term constant (axiom) of type ι.
3. For each function symbol f of arity $\iota_1 \times \cdots \times \iota_n \to \iota$, define f as a term constant (axiom) of type $\iota_1 \to \cdots \to \iota_n \to \iota$.
4. For each predicate symbol P of arity $(\iota_1, \cdots \iota_n)$, define P as a term constant (axiom) of type $\iota_1 \to \cdots \to \iota_n \to \texttt{Prop}$.

IFP-Coq

For each language $\mathcal{L}$ and set of axioms $\mathcal{A}$, we define a set of Coq axioms by

1. For each sort ι in $\mathcal{L}$, define ι as a term constant (axiom) of Prop.
2. For each constant c of sort ι, define c as a term constant (axiom) of type ι.
3. For each function symbol f of arity $\iota_1 \times \cdots \times \iota_n \to \iota$, define f as a term constant (axiom) of type $\iota_1 \to \cdots \to \iota_n \to \iota$.
4. For each predicate symbol P of arity $(\iota_1, \cdots \iota_n)$, define P as a term constant (axiom) of type $\iota_1 \to \cdots \to \iota_n \to \texttt{Prop}$.
5. For each operator symbol Q of arity $(\iota_1, \cdots \iota_n)$, define Q as a term constant (axiom) of type $(\iota_1 \to \cdots \to \iota_n \to \texttt{Prop}) \to (\iota_1 \to \cdots \to \iota_n \to \texttt{Prop})$.

Translating Formulas i

1. $H(c : \iota) = \ \vdash c : \iota$,
2. $H(f : \iota_1 \times \cdots \times \iota_n \to \iota) = \ \vdash f : \iota_1 \to \cdots \to \iota_n \to \iota$,
3. $H(x : \iota) = x : \iota \vdash x : \iota$,
4. $H(C : \text{ predicate}(\iota_1, \ldots, \iota_d)) = \ \vdash C : \iota_1 \to \cdots \to \iota_n \to \texttt{Prop}$,
5. $H(X : \text{ predicate}(\iota_1, \ldots, \iota_d)) = X : \iota_1 \to \cdots \to \iota_d \to \texttt{Prop} \vdash X : \iota_1 \to \cdots \to \iota_d \to \texttt{Prop}$,
6. $H(f(t_1, \cdots, t_n) : \iota) = \Gamma \vdash f\ t'_1\ \cdots\ t'_n : \iota$ when $H(t_i : \iota_i) = \Gamma_i \vdash t'_i : \iota_i$ and $\Gamma = \bigcup_i \Gamma_i$,
7. $H(t_1 = t_2) = \Gamma \vdash t'_1 = t'_2 : \texttt{Prop}$ when $H(t_1) = \Gamma_1 \vdash t'_1 : \iota$, $H(t_2) = \Gamma_2 \vdash t'_2 : \iota$, and $\Gamma = \Gamma_1 \cup \Gamma_2$,

Translating Formulas ii

8. $H(P \vee Q) = \Gamma \vdash P' \vee Q'$ when $H(P) = \Gamma_1 \vdash P' : \texttt{Prop}$, $H(Q) = \Gamma_2 \vdash Q' : \texttt{Prop}$, and $\Gamma = \Gamma_1 \cup \Gamma_2$,
9. $H(P \wedge Q) = \Gamma \vdash P' \wedge Q'$ when $H(P) = \Gamma_1 \vdash P' : \texttt{Prop}$, $H(Q) = \Gamma_2 \vdash Q' : \texttt{Prop}$, and $\Gamma = \Gamma_1 \cup \Gamma_2$,
10. $H(P \to Q) = \Gamma \vdash P' \to Q'$ when $H(P) = \Gamma_1 \vdash P' : \texttt{Prop}$, $H(Q) = \Gamma_2 \vdash Q' : \texttt{Prop}$, and $\Gamma = \Gamma_1 \cup \Gamma_2$,
11. $H(\exists x\ P) = \Gamma \setminus (x : \iota) \vdash \exists (x : \iota).\ P'$ when $H(x) = x : \iota$ and $H(P) = \Gamma \vdash P' : \texttt{Prop}$,
12. $H(\forall x\ P) = \Gamma \setminus (x : \iota) \vdash \forall (x : \iota).\ P'$ when $H(x) = x : \iota$ and $H(P) = \Gamma \vdash P' : \texttt{Prop}$,
13. $H(\lambda x\ P) = \Gamma \setminus (x : \iota) \vdash \lambda (x : \iota).\ P'$ when $H(x) = x : \iota$ and $H(P) = \Gamma \vdash P' : \texttt{Prop}$,

Translating Formulas iii

14. $H(\lambda X\ P) = \Gamma \setminus (X : (\iota_1 \to \cdots \to \iota_n \to \texttt{Prop})) \vdash \lambda(X : (\iota_1 \cdots \to \iota_n \to \texttt{Prop})).\ P'$ when
    $H(X) = X : \iota_1 \to \cdots \to \iota_n \to \texttt{Prop}$ and
    $H(P) = \Gamma \vdash P' : \texttt{Prop}$,

Inductive Types

For each expression $\mu(\Phi)$ in IFP we define an inductive type in Coq:

```
Inductive MPhi : (ι -> Prop) :=
    MPhic: forall x,  (Phi Mphi x) -> Mphi x.
```

where Phi is the translation of Φ.

Inductive Types

For each expression $\mu(\Phi)$ in IFP we define an inductive type in Coq:

```
Inductive MPhi : (ι -> Prop) :=
    MPhic: forall x,  (Phi Mphi x) -> Mphi x.
```

where Phi is the translation of Φ.

The right induction principle will in general not be generated by Coq but

```
Lemma ind_Phi: forall (P : ι -> Prop),
    forall x, ((Phi P x) -> P x) -> Mphi x -> P x.
```

can be proven.

Example

We formalized the real number examples from

Ulrich Berger, Hideki Tsuiki: **Intutionistic Fixed Point Logic.**
Annals of Pure and Applied Logic 172.3 (2021): 102903.

Example

We formalized the real number examples from

Ulrich Berger, Hideki Tsuiki: **Intutionistic Fixed Point Logic.**
Annals of Pure and Applied Logic 172.3 (2021): 102903.

Recall the definition of natural numbers in IFP:

$$N(x) = \mu(\lambda X \lambda x \, (x = 0 \vee X(x-1)))$$

$\Rightarrow$ Demo in Coq.

Gray Code and Signed Digit Representation

The signed-digit representation for a real number $x \in [-1, 1]$ can be defined by $\nu(\Phi_{\mathrm{SD}})$ with

$$\Phi_{\mathrm{SD}} := \lambda X \lambda x \, \exists d \in \mathrm{SD} \; |2x - d| \leq 1 \wedge X(2x - d)$$

The infinite gray-code representation can be defined by $\nu(\Phi_{\mathrm{G}})$ with

$$\Phi_{\mathrm{G}} := \lambda X \lambda x \; |x| \leq 1 \wedge (x \neq 0 \to x \leq 0 \vee x \geq 0) \wedge X(1 - 2\,|x|)$$

Mixed Induction/Coinduction

Nested inductive/coinductive definitions are used in IFP e.g. to define uniformly continuous functions.

$$\text{For } d \in \mathbb{R}, \text{ let } av_d(x) := \frac{d+x}{2} \text{ and}$$

$$C_1 := \nu F\, \mu G\, \lambda h\, \exists i \in \mathrm{SD}\, \exists f \in F(h = av_e \circ f) \vee \forall d \in \mathrm{SD}\, h \circ av_d \in G.$$

Mixed Induction/Coinduction

Nested inductive/coinductive definitions are used in IFP e.g. to define uniformly continuous functions.

$$\text{For } d \in \mathbb{R}, \text{ let } av_d(x) := \frac{d+x}{2} \text{ and}$$

$$C_1 := \nu F\, \mu G\, \lambda h\, \exists i \in \mathrm{SD}\, \exists f \in F (h = av_e \circ f) \vee \forall d \in \mathrm{SD}\; h \circ av_d \in G.$$

Coq does not accept the coinductive proof because it is not formally guarded.

Future work

- Formalization of RIFP in Coq
- Program extraction
- Translation from Coq proofs to IFP proofs
- Extension to CFP

Happy Birthday, Ulrich!