A functional interpretation with state

Thomas Powell

Technische Universität Darmstadt

Logic in Computer Science (LICS 18)

University of Oxford

12 July 2018

Background: Proof Interpretations

Proof interpretations allow us to give a computational interpretation to mathematical statements.

Background: Proof Interpretations

Proof interpretations allow us to give a computational interpretation to mathematical statements.

If $\mathcal{T} \vdash A$ then $\mathcal{S} \vdash \forall x A_I(x, tx)$, where:

Background: Proof Interpretations

Proof interpretations allow us to give a computational interpretation to mathematical statements.

$$\text{If } \mathcal{T} \vdash A \text{ then } \mathcal{S} \vdash \forall x A_I(x, tx), \text{ where:}$$

- $A_I(x, y)$ is computationally neutral;
- $A \leftrightarrow \forall x \exists y A_I(x, y)$;
- t is a term of $\mathcal{S}$ formally extracted from the proof of A.

Background: Proof Interpretations

Proof interpretations allow us to give a computational interpretation to mathematical statements.

$$\text{If } \mathcal{T} \vdash A \text{ then } \mathcal{S} \vdash \forall x A_I(x, tx), \text{ where:}$$

- $A_I(x, y)$ is computationally neutral;
- $A \leftrightarrow \forall x \exists y A_I(x, y)$;
- t is a term of $\mathcal{S}$ formally extracted from the proof of A.

You may have seen proof interpretations in the following contexts:

- Foundational problems: $\mathsf{Con}(\mathcal{S}) \Rightarrow \mathsf{Con}(\mathcal{T})$.
- Proof mining: New quantitative results in numerical analysis, ergodic theory, convex optimization...
- Category theory: Dialectica categories as models of linear logic etc.
- Formal program extraction: Implementation of proof interpretations in Minlog, Agda Coq...

Motivation

One of my current interests is to understand Gödel's functional interpretation of strong classical theories, using concepts from imperative programming such as

- global state;
- monadic transformations;
- abstract machines;
- Hoare logic.

Motivation

One of my current interests is to understand Gödel's functional interpretation of strong classical theories, using concepts from imperative programming such as

- global state;
- monadic transformations;
- abstract machines;
- Hoare logic.

Why?

1. Applications of proof interpretations in computer science should make use of programming paradigms which are used in practice.

2. The above concepts provide us with a natural means of understanding the functional interpretation of non-trivial classical principles.

3. Combining techniques from two different areas always leads to new ideas!

The drinkers paradox: $\exists x(P(x) \to \forall y P(y))$

In any pub there is a person such that if they are drinking, then everyone is drinking

The drinkers paradox: $\exists x(P(x) \rightarrow \forall y P(y))$

> *In any pub there is a person such that if they are drinking, then everyone is drinking*

There is no effective way to find a witness for x, as its existence depends on the law of excluded-middle. The classical functional interpretation carries out the following steps:

$$\boxed{\exists x(P(x) \rightarrow \forall y P(y))}$$

The drinkers paradox: $\exists x(P(x) \to \forall y P(y))$

> *In any pub there is a person such that if they are drinking, then everyone is drinking*

There is no effective way to find a witness for x, as its existence depends on the law of excluded-middle. The classical functional interpretation carries out the following steps:

$$\boxed{\exists x(P(x) \to \forall y P(y))} \Leftrightarrow \exists x \forall y (P(x) \to P(y))$$

The drinkers paradox: $\exists x(P(x) \to \forall y P(y))$

> *In any pub there is a person such that if they are drinking, then everyone is drinking*

There is no effective way to find a witness for x, as its existence depends on the law of excluded-middle. The classical functional interpretation carries out the following steps:

$$\begin{aligned}\boxed{\exists x(P(x) \to \forall y P(y))} &\Leftrightarrow \exists x \forall y (P(x) \to P(y)) \\ &\Leftrightarrow \neg \forall x \exists y \neg (P(x) \to P(y))\end{aligned}$$

The drinkers paradox: $\exists x(P(x) \to \forall y P(y))$

> *In any pub there is a person such that if they are drinking, then everyone is drinking*

There is no effective way to find a witness for x, as its existence depends on the law of excluded-middle. The classical functional interpretation carries out the following steps:

$$\begin{aligned}
\boxed{\exists x(P(x) \to \forall y P(y))} &\Leftrightarrow \exists x \forall y(P(x) \to P(y)) \\
&\Leftrightarrow \neg \forall x \exists y \neg (P(x) \to P(y)) \\
&\Leftrightarrow \neg \exists f \forall x \neg (P(x) \to P(fx))
\end{aligned}$$

The drinkers paradox: $\exists x(P(x) \to \forall y P(y))$

> *In any pub there is a person such that if they are drinking, then everyone is drinking*

There is no effective way to find a witness for x, as its existence depends on the law of excluded-middle. The classical functional interpretation carries out the following steps:

$$
\begin{aligned}
\boxed{\exists x(P(x) \to \forall y P(y))} &\Leftrightarrow \exists x \forall y(P(x) \to P(y)) \\
&\Leftrightarrow \neg \forall x \exists y \neg(P(x) \to P(y)) \\
&\Leftrightarrow \neg \exists f \forall x \neg(P(x) \to P(fx)) \\
&\Leftrightarrow \boxed{\forall f \exists x(P(x) \to P(fx))}.
\end{aligned}
$$

The drinkers paradox: $\exists x(P(x) \rightarrow \forall y P(y))$

> *In any pub there is a person such that if they are drinking, then everyone is drinking*

There is no effective way to find a witness for x, as its existence depends on the law of excluded-middle. The classical functional interpretation carries out the following steps:

$$\begin{aligned} \boxed{\exists x(P(x) \rightarrow \forall y P(y))} &\Leftrightarrow \exists x \forall y(P(x) \rightarrow P(y)) \\ &\Leftrightarrow \neg\forall x \exists y \neg(P(x) \rightarrow P(y)) \\ &\Leftrightarrow \neg\exists f \forall x \neg(P(x) \rightarrow P(fx)) \\ &\Leftrightarrow \boxed{\forall f \exists x(P(x) \rightarrow P(fx))}. \end{aligned}$$

Ineffective statement: *There exists some ideal drinker x such that if x drinks, then all people y drink.*

The drinkers paradox: $\exists x(P(x) \to \forall y P(y))$

> *In any pub there is a person such that if they are drinking, then everyone is drinking*

There is no effective way to find a witness for x, as its existence depends on the law of excluded-middle. The classical functional interpretation carries out the following steps:

$$
\begin{aligned}
\boxed{\exists x(P(x) \to \forall y P(y))} &\Leftrightarrow \exists x \forall y (P(x) \to P(y)) \\
&\Leftrightarrow \neg \forall x \exists y \neg (P(x) \to P(y)) \\
&\Leftrightarrow \neg \exists f \forall x \neg (P(x) \to P(fx)) \\
&\Leftrightarrow \boxed{\forall f \exists x (P(x) \to P(fx))}.
\end{aligned}
$$

Ineffective statement: *There exists some ideal drinker x such that if x drinks, then all people y drink.*

Effective reformulation: *For any function f there exists an approximate drinker x such that if x drinks, then person fx drinks.*

The interpreted drinkers paradox: $\forall f \exists x(P(x) \to P(fx))$

How do we compute a witness for x? There are two widely seen methods:

The interpreted drinkers paradox: $\forall f \exists x (P(x) \to P(fx))$

How do we compute a witness for x? There are two widely seen methods:

- The original functional interpretations extracts an exact witness, but requires decidability of quantifier-free formulas:

$$\Phi(f) := \begin{cases} 0 & \text{if } P(f0) \\ f0 & \text{if } \neg P(f0) \end{cases}$$

  Used for applications in discrete mathematics.

The interpreted drinkers paradox: $\forall f \exists x (P(x) \to P(fx))$

How do we compute a witness for x? There are two widely seen methods:

- The original functional interpretations extracts an exact witness, but requires decidability of quantifier-free formulas:

$$\Phi(f) := \begin{cases} 0 & \text{if } P(f0) \\ f0 & \text{if } \neg P(f0) \end{cases}$$

  Used for applications in discrete mathematics.

- Diller-Nahm or Herbrand variants of the functional interpretation extract a finite sequence of witnesses. No longer require decidability.

$$\Phi(f) := [0, f0]$$

  Used for theories with non-decidable atomic formulas (e.g. nonstandard analysis) and for applications in category theory.

The interpreted drinkers paradox: $\forall f \exists x (P(x) \to P(fx))$

In the paper, a new variant of the functional interpretation is developed, which combines these two approaches: We store assumptions about our realizer in a global state.

The interpreted drinkers paradox: $\forall f \exists x (P(x) \to P(fx))$

In the paper, a new variant of the functional interpretation is developed, which combines these two approaches: We store assumptions about our realizer in a global state.

There are two possible realizers for the drinkers paradox

- $\Phi_L(f, \pi) := \langle 0, \pi :: P(f0) \rangle$
- $\Phi_R(f, \pi) := \langle f0, \pi :: \neg P(f0) \rangle$

Here, π is a global state of assumptions, which is updated to include new assumptions made during the computation.

Both realizers are correct relative to the state.

The interpreted drinkers paradox: $\forall f \exists x(P(x) \to P(fx))$

In the paper, a new variant of the functional interpretation is developed, which combines these two approaches: We store assumptions about our realizer in a global state.

There are two possible realizers for the drinkers paradox

- $\Phi_L(f, \pi) := \langle 0, \pi :: P(f0) \rangle$
- $\Phi_R(f, \pi) := \langle f0, \pi :: \neg P(f0) \rangle$

Here, π is a global state of assumptions, which is updated to include new assumptions made during the computation.

Both realizers are correct relative to the state.

This is a very simple case instance of a much more general framework developed in the paper.

A real world example: Ramsey's theorem for pairs

A real world example: Ramsey's theorem for pairs

Theorem

For any colouring $c : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$, *there exists an infinite set* $X \subseteq \mathbb{N}$ *that is pairwise monochromatic.*

A real world example: Ramsey's theorem for pairs

Theorem

For any colouring $c : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$, *there exists an infinite set* $X \subseteq \mathbb{N}$ *that is pairwise monochromatic.*

Theorem (Finitized version)

For any colouring $c : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ *and functional* $\varepsilon : \mathcal{P}(\mathbb{N}) \to \mathbb{N}$, *there exists a finite approximation* $X_\varepsilon \subseteq \mathbb{N}$ *to a monochromatic set, which is valid up to the point* $\varepsilon(X_\varepsilon)$.

A real world example: Ramsey's theorem for pairs

Theorem

For any colouring $c : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$, there exists an infinite set $X \subseteq \mathbb{N}$ that is pairwise monochromatic.

Theorem (Finitized version)

For any colouring $c : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ and functional $\varepsilon : \mathcal{P}(\mathbb{N}) \to \mathbb{N}$, there exists a finite approximation $X_\varepsilon \subseteq \mathbb{N}$ to a monochromatic set, which is valid up to the point $\varepsilon(X_\varepsilon)$.

From the classical proof of Ramsey's theorem, we would extract a program

$$\Phi : S \to (\mathcal{P}(\mathbb{N}) \to \mathbb{N}) \to \mathcal{P}(\mathbb{N}) \times S,$$

which from an ε and initial state π_0 would return a pair $\Phi(\varepsilon, \pi_I) = \langle X, \pi_O \rangle$

A real world example: Ramsey's theorem for pairs

Theorem

For any colouring $c : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$, there exists an infinite set $X \subseteq \mathbb{N}$ that is pairwise monochromatic.

Theorem (Finitized version)

For any colouring $c : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ and functional $\varepsilon : \mathcal{P}(\mathbb{N}) \to \mathbb{N}$, there exists a finite approximation $X_\varepsilon \subseteq \mathbb{N}$ to a monochromatic set, which is valid up to the point $\varepsilon(X_\varepsilon)$.

From the classical proof of Ramsey's theorem, we would extract a program

$$\Phi : S \to (\mathcal{P}(\mathbb{N}) \to \mathbb{N}) \to \mathcal{P}(\mathbb{N}) \times S,$$

which from an ε and initial state π_0 would return a pair $\Phi(\varepsilon, \pi_I) = \langle X, \pi_O \rangle$

- X is a finite approximation to a monochromatic set;
- the π_O is the output state, which contains atomic formulas of the form $c(m, n) = b$ listing 'interactions with the environment' which occured during the computation of X.

The main result

Theorem

Suppose that $\mathsf{HA}^+ \vdash A(\vec{b})$*. Then for any collection of approximations* φ_P *satisfying* $\chi_P \approx_{0\to 0} \varphi_P$*, there is a corresponding sequence of state-sensitive terms* $\vec{t}$ *satisfying*

$$\mathsf{E\text{-}HA}^\omega_S \vdash \forall \vec{v} \in \Delta_{\vec{\tau}}, \pi \left(\left\| A(\vec{b}) \right\|_{\vec{v}}^{\vec{t}\vec{b}} \pi \to \left\{ A(\vec{b}) \right\}_{\vec{v}}^{\vec{t}\vec{b}} \pi \right)$$

which can be formally extracted from the proof of $A(\vec{b})$*.*

The main result

Theorem

Suppose that $\mathsf{HA}^+ \vdash A(\vec{b})$. *Then for any collection of approximations* φ_P *satisfying* $\chi_P \approx_{0\to 0} \varphi_P$, *there is a corresponding sequence of state-sensitive terms* $\vec{t}$ *satisfying*

$$\mathsf{E\text{-}HA}^{\omega}_{S} \vdash \forall \vec{v} \in \Delta_{\vec{\tau}}, \pi \left(\left\| A(\vec{b}) \right\|^{\vec{t}\vec{b}}_{\vec{v}} \pi \to \left\{ A(\vec{b}) \right\}^{\vec{t}\vec{b}}_{\vec{v}} \pi \right)$$

which can be formally extracted from the proof of $A(\vec{b})$.

Comments

The main result

Theorem

Suppose that $\mathsf{HA}^+ \vdash A(\vec{b})$*. Then for any collection of approximations* φ_P *satisfying* $\chi_P \approx_{0\to 0} \varphi_P$*, there is a corresponding sequence of state-sensitive terms* $\vec{t}$ *satisfying*

$$\mathsf{E\text{-}HA}^\omega_S \vdash \forall \vec{v} \in \Delta_{\vec{\tau}}, \pi \left(\left\| A(\vec{b}) \right\|_{\vec{v}}^{\vec{t}\vec{b}} \pi \to \left\{ A(\vec{b}) \right\}_{\vec{v}}^{\vec{t}\vec{b}} \pi \right)$$

which can be formally extracted from the proof of $A(\vec{b})$*.*

Comments

- $\left\{ A(\vec{b}) \right\}_{\vec{v}}^{\vec{t}\vec{b}} \pi$ is an analogue of the usual functional interpretations.

The main result

Theorem

Suppose that $\mathsf{HA}^{+} \vdash A(\vec{b})$*. Then for any collection of approximations* φ_P *satisfying* $\chi_P \approx_{0\to 0} \varphi_P$*, there is a corresponding sequence of state-sensitive terms* $\vec{t}$ *satisfying*

$$\mathsf{E\text{-}HA}^{\omega}_{S} \vdash \forall \vec{v} \in \Delta_{\vec{\tau}}, \pi \left(\left\| A(\vec{b}) \right\|^{\vec{t}\vec{b}}_{\vec{v}} \pi \rightarrow \left\{ A(\vec{b}) \right\}^{\vec{t}\vec{b}}_{\vec{v}} \pi \right)$$

which can be formally extracted from the proof of $A(\vec{b})$.

Comments

- $\left\{ A(\vec{b}) \right\}^{\vec{t}\vec{b}}_{\vec{v}} \pi$ is an analogue of the usual functional interpretations.
- $\left\| A(\vec{b}) \right\|^{\vec{t}\vec{b}}_{\vec{v}} \pi$ is a new special state component.

The main result

Theorem

Suppose that $\mathsf{HA}^+ \vdash A(\vec{b})$. *Then for any collection of approximations* φ_P *satisfying* $\chi_P \approx_{0\to 0} \varphi_P$, *there is a corresponding sequence of state-sensitive terms* $\vec{t}$ *satisfying*

$$\mathsf{E\text{-}HA}^\omega_S \vdash \forall \vec{v} \in \Delta_{\vec{\tau}}, \pi \left(\left\| A(\vec{b}) \right\|^{\vec{t}\vec{b}}_{\vec{v}} \pi \to \left\{ A(\vec{b}) \right\}^{\vec{t}\vec{b}}_{\vec{v}} \pi \right)$$

which can be formally extracted from the proof of $A(\vec{b})$.

Comments

- $\left\{ A(\vec{b}) \right\}^{\vec{t}\vec{b}}_{\vec{v}} \pi$ is an analogue of the usual functional interpretations.
- $\left\| A(\vec{b}) \right\|^{\vec{t}\vec{b}}_{\vec{v}} \pi$ is a new special state component.
- χ_P is the characteristic function of P and φ_P is its approximation relative to the state.

The main result

Theorem

Suppose that $\mathsf{HA}^{+} \vdash A(\vec{b})$. *Then for any collection of approximations* φ_P *satisfying* $\chi_P \approx_{0\to 0} \varphi_P$, *there is a corresponding sequence of state-sensitive terms* $\vec{t}$ *satisfying*

$$\mathsf{E\text{-}HA}^{\omega}_{S} \vdash \forall \vec{v} \in \Delta_{\vec{\tau}}, \pi \left(\left\| A(\vec{b}) \right\|^{\vec{t}\vec{b}}_{\vec{v}} \pi \to \left\{ A(\vec{b}) \right\}^{\vec{t}\vec{b}}_{\vec{v}} \pi \right)$$

which can be formally extracted from the proof of $A(\vec{b})$.

Comments

- $\left\{ A(\vec{b}) \right\}^{\vec{t}\vec{b}}_{\vec{v}} \pi$ is an analogue of the usual functional interpretations.
- $\left\| A(\vec{b}) \right\|^{\vec{t}\vec{b}}_{\vec{v}} \pi$ is a new special state component.
- χ_P is the characteristic function of P and φ_P is its approximation relative to the state.
- The proof involves the state monad and a logical relation on all types.

Application: Herbrand's theorem

Our state based functional interpretation gives us a new proof of Herbrand's theorem. Suppose that

$$\mathsf{PL} \vdash \exists x A(x)$$

Application: Herbrand's theorem

Our state based functional interpretation gives us a new proof of Herbrand's theorem. Suppose that

$$\mathsf{PL} \vdash \exists x A(x)$$

Then for any given state function, starting with the empty state we obtain a term t such that

$$\underbrace{P_1 \wedge \ldots \wedge P_k}_{\text{final state}} \rightarrow A(t)$$

Application: Herbrand's theorem

Our state based functional interpretation gives us a new proof of Herbrand's theorem. Suppose that

$$\mathsf{PL} \vdash \exists x A(x)$$

Then for any given state function, starting with the empty state we obtain a term t such that

$$\underbrace{P_1 \wedge \ldots \wedge P_k}_{\text{final state}} \to A(t)$$

The final state represents a branch in the underlying Herbrand tree. By quantifying over all relevant states we obtain terms $t_1, \ldots, t_n$ s.t.

$$A(t_1) \vee \ldots \vee A(t_n).$$

Application: Learning semantics

Suppose that we have a collection $P(n, i)$ of primitive recursive formulas. We use the state to record witnesses for $\exists i P(n, i)$ i.e. is an approximation to a Skolem functions for Σ_1 formulas. Suppose that

$$\mathsf{PA} \vdash \forall x \exists y A(x, y).$$

Application: Learning semantics

Suppose that we have a collection $P(n, i)$ of primitive recursive formulas. We use the state to record witnesses for $\exists i P(n, i)$ i.e. is an approximation to a Skolem functions for Σ_1 formulas. Suppose that

$$\mathsf{PA} \vdash \forall x \exists y A(x, y).$$

Then we define our state function to add only true assumptions to the state, and we extract a program $t : \mathsf{o} \to S \to \mathsf{o} \times S$ such that

Application: Learning semantics

Suppose that we have a collection $P(n, i)$ of primitive recursive formulas. We use the state to record witnesses for $\exists i P(n, i)$ i.e. is an approximation to a Skolem functions for Σ_1 formulas. Suppose that

$$\mathsf{PA} \vdash \forall x \exists y A(x, y).$$

Then we define our state function to add only true assumptions to the state, and we extract a program $t : \mathsf{0} \to S \to \mathsf{0} \times S$ such that

- t takes an argument x and a state π representing an approximation to Skolem function;

Application: Learning semantics

Suppose that we have a collection $P(n, i)$ of primitive recursive formulas. We use the state to record witnesses for $\exists i P(n, i)$ i.e. is an approximation to a Skolem functions for Σ_1 formulas. Suppose that

$$\mathsf{PA} \vdash \forall x \exists y A(x, y).$$

Then we define our state function to add only true assumptions to the state, and we extract a program $t : \mathsf{o} \to S \to \mathsf{o} \times S$ such that

- t takes an argument x and a state π representing an approximation to Skolem function;
- t returns a realizer $tx\pi_0$ together with a final state $tx\pi_1 \sqsupseteq \pi$ representing a better approximation to Skolem function, containing what we have learned from computing our realizer.

Application: Efficient program synthesis

Our state is not just for storing information: We can interact with it as well.

One application might be to improve the efficiency of extracted programs by e.g. avoiding repeated computations. For example:

Application: Efficient program synthesis

Our state is not just for storing information: We can interact with it as well.

One application might be to improve the efficiency of extracted programs by e.g. avoiding repeated computations. For example:

$$\Phi(f, \pi) := \begin{cases} \langle 0, \pi \rangle & \text{if } P(f0) \in \pi \\ \langle f0, \pi \rangle & \text{if } \neg P(f0) \in \pi \\ \langle 0, \pi :: P(f0) \rangle & \text{if } P(f0) \\ \langle f0, \pi :: \neg P(f0) \rangle & \text{if } \neg P(f0) \end{cases}$$

Application: Efficient program synthesis

Our state is not just for storing information: We can interact with it as well.

One application might be to improve the efficiency of extracted programs by e.g. avoiding repeated computations. For example:

$$\Phi(f,\pi) := \begin{cases} \langle 0, \pi \rangle & \text{if } P(f0) \in \pi \\ \langle f0, \pi \rangle & \text{if } \neg P(f0) \in \pi \\ \langle 0, \pi :: P(f0) \rangle & \text{if } P(f0) \\ \langle f0, \pi :: \neg P(f0) \rangle & \text{if } \neg P(f0) \end{cases}$$

Naturally, we have more sophisticated things in mind!

Future research

We now want explore the applications.

Future research

We now want explore the applications.

- We presented everything in a standard equational calculus and use the state monad. It would be nice to instead use a real programming language with state.

Future research

We now want explore the applications.

- We presented everything in a standard equational calculus and use the state monad. It would be nice to instead use a real programming language with state.
- Formalise the interpretation! (Minlog or Adga might make good candidates...)

Future research

We now want explore the applications.

- We presented everything in a standard equational calculus and use the state monad. It would be nice to instead use a real programming language with state.
- Formalise the interpretation! (Minlog or Adga might make good candidates...)
- Can the state provide us with insight into the meaning of programs extracted from complicated proofs? Can we extend our framework to include the axiom of choice/bar recursion?

Future research

We now want explore the applications.

- We presented everything in a standard equational calculus and use the state monad. It would be nice to instead use a real programming language with state.
- Formalise the interpretation! (Minlog or Adga might make good candidates...)
- Can the state provide us with insight into the meaning of programs extracted from complicated proofs? Can we extend our framework to include the axiom of choice/bar recursion?
- This is a first step in the development of a fledged imperative functional interpretation, specifically designed for extracting state sensitive programs from proofs.

Future research

We now want explore the applications.

- We presented everything in a standard equational calculus and use the state monad. It would be nice to instead use a real programming language with state.
- Formalise the interpretation! (Minlog or Adga might make good candidates...)
- Can the state provide us with insight into the meaning of programs extracted from complicated proofs? Can we extend our framework to include the axiom of choice/bar recursion?
- This is a first step in the development of a fledged imperative functional interpretation, specifically designed for extracting state sensitive programs from proofs.

THANK YOU!