Intensionality,
Intensional Recursion,
and the Gödel-Löb axiom

Alex Kavvos
Department of Computer Science, University of Oxford

IMLA 2017, 18 July 2017
What this talk is about

• It is about using modal logic, to present a typing discipline for programs-as-data.

• It is about investigating the central rule/axiom of provability logic in this setting.
What this talk is about

• It is about using modal logic, to present a typing discipline for programs-as-data.

• It is about investigating the central rule/axiom of provability logic in this setting.

Curry-Howard isomorphism for intensional programming
What this talk is about

- It is about using modal logic, to present a typing discipline for programs-as-data.
- It is about investigating the central rule/axiom of provability logic in this setting.

Curry-Howard isomorphism for intensional programming

intensional recursion
Programs-as-data

• More than just ‘functions as first-class-citizens.’

  • The extensional paradigm: a program can call a functional argument at a finite set of points.

• Instead, very close to the idea of Gödel numbering.

  • The intensional paradigm: a program can inspect the source code of its functional argument, and can do rather arbitrary things with it (inspect, simulate, deconstruct, count its symbols…).

• Non-functional operations.

• Homoiconicity: when one does not need coding at all; e.g. LISP.
Programs-as-data

- More than just 'functions as first-class-citizens.'
  - The **extensional paradigm**: a program can call a functional argument at a finite set of points.
  - Instead, very close to the idea of **Gödel numbering**.
  - The **intensional paradigm**: a program can inspect the source code of its functional argument, and can do rather arbitrary things with it (inspect, simulate, deconstruct, count its symbols...).

How can we do this in a typed, well-structured, safe, coding-free manner?
Intensional Recursion

• A very strong kind of recursion, discovered by Kleene in 1938. For CS, lost in the mists of time (Abramsky).

• In the untyped $\lambda$-calculus:

First Recursion Theorem \[ \forall f \in \Lambda. \exists u \in \Lambda. u = f u \]
Second Recursion Theorem \[ \forall f \in \Lambda. \exists u \in \Lambda. u = f \, \ulcorner u \urcorner \]
Enumeration Theorem \[ \exists E \in \Lambda. \forall u \in \Lambda^0. E \, \ulcorner u \urcorner = u \]

Given EN, the SRT implies the FRT, hence it is stronger.

But what does it really do?
Types for Intensionality

• Strangely, intensionality follows a typing discipline.

• Suppose \( u : A \); let’s say \( u \vdash : \Box A \)

• Then well-known combinators of \( \lambda \)-calculus that perform operations on Gödel numbers acquire types; e.g.

\[
\begin{align*}
gnum (M) &= (M) \\
E (M) &= M \\
\text{app} (M, N) &= (M \cdot N)
\end{align*}
\]
Types for Intensionality

• Strangely, intensionality follows a typing discipline.

• Suppose \( u : A \); let’s say \( \ulcorner u \urcorner : \Box A \)

• Then well-known combinators of \( \lambda \)-calculus that perform operations on Gödel numbers acquire types; e.g.

  \[
  \text{gnum} : \Box A \rightarrow \Box \Box A \\
  \text{gnum}\ulcorner M\urcorner = \ulcorner \ulcorner M\urcorner \urcorner \\
  \text{E}\ulcorner M\urcorner = M
  \]

  \[
  \text{app}\ulcorner M\urcorner \ulcorner N\urcorner = \ulcorner M \, N\urcorner
  \]
Types for Intensionality

• Strangely, intensionality follows a typing discipline.

• Suppose \( u : A \); let’s say \( u \Downarrow : \square A \)

• Then well-known combinators of \( \lambda \)-calculus that perform operations on Gödel numbers acquire types; e.g.

\[
\begin{align*}
gnum : & \square A \rightarrow \square \square A \\
gnum \left( M \Downarrow \right) & = \square \left( M \Downarrow \right) \\
E & \left( M \Downarrow \right) = M \\
app : & \square (A \rightarrow B) \rightarrow \square A \rightarrow \square B \\
app \left( M \Downarrow, N \Downarrow \right) & = \square (M \downarrow N \downarrow)
\end{align*}
\]
Types for Intensionality

• Strangely, intensionality follows a typing discipline.

• Suppose \( u : A \); let’s say \( \ulcorner u \ulcorner : \square A \)

• Then well-known combinators of \( \lambda \)-calculus that perform operations on Gödel numbers acquire types; e.g.

\[
\begin{align*}
gnum &: \square A \rightarrow \square \square A & \quad E &: \square A \rightarrow A \\
\ulcorner M \ulcorner &= \ulcorner \ulcorner M \ulcorner \ulcorner & \quad E \ulcorner M \ulcorner &= M \\
app &: \square (A \rightarrow B) \rightarrow \square A \rightarrow \square B \\
\ulcorner M \ulcorner \ulcorner N \ulcorner &= \ulcorner M \ N \ulcorner
\end{align*}
\]
Types for Intensionality

• Strangely, intensionality follows a typing discipline.

• Suppose \( u : A \); let's say \( \llbracket u \rrbracket : \square A \)

• Then well-known combinators of \( \lambda \)-calculus that perform operations on Gödel numbers acquire types; e.g.

\[
\begin{align*}
gnum & : \square A \rightarrow \square \square A & E & : \square A \rightarrow A \\
gnum \llbracket M \rrbracket & = \llbracket \llbracket M \rrbracket \rrbracket & E \llbracket M \rrbracket & = M \\
app & : \square(A \rightarrow B) \rightarrow \square A \rightarrow \square B & \text{It's S4!}
\end{align*}
\]

\[
\begin{align*}
app \llbracket M \rrbracket \llbracket N \rrbracket & = \llbracket M N \rrbracket
\end{align*}
\]
Types for Intensional Recursion
Types for Intensional Recursion

• Take \( u : A \) so that \( u = f \left[ u \right] \)
Types for Intensional Recursion

• Take $u : A$ so that $u = f \ulcorner u \urcorner$

• Then it is forced that $f : \Box A \to A$
Types for Intensional Recursion

• Take $u : A$ so that $u = f \downarrow u$

• Then it is forced that $f : \Box A \rightarrow A$

• Yields the following
Types for Intensional Recursion

- Take $u : A$ so that $u = f \left[ u \right]$

- Then it is forced that $f : \Box A \rightarrow A$

- Yields the following

Logical interpretation of the Second Recursion Theorem

\[
\begin{align*}
  f : \Box A & \rightarrow A \\
  \hline \\
  u : A
\end{align*}
\]
Types for Intensional Recursion

• Take $u : A$ so that $u = f \upharpoonright u$.

• Then it is forced that $f : \Box A \to A$.

• Yields the following:

  Logical interpretation of the Second Recursion Theorem

  $f : \Box A \to A$

  $u : A$

  ... such that $u = f \upharpoonright u$.
Types for Intensional Recursion

• Take \( u : A \) so that \( u = f \upharpoonright u \downharpoonright \)

• Then it is forced that \( f : \Box A \to A \)

• Yields the following

Logical interpretation of the Second Recursion Theorem

\[
f : \Box A \to A \\
\therefore u \downharpoonright : \Box A
\]

... such that \( u = f \upharpoonright u \downharpoonright \)
Prospectus

• We will first revisit Davies & Pfenning’s S4.

• We will add intensional operations to it.

• Then we will add intensional recursion.

• The resulting system is called Intensional PCF.
Prospectus

- We will first revisit Davies & Pfenning’s S4.
- We will add intensional operations to it.
- Then we will add intensional recursion.
- The resulting system is called Intensional PCF.

**THEOREM.**
‘Full’ reduction of Intensional PCF is confluent. Hence, Intensional PCF is consistent.
I. Curry-Howard and $S4$
Curry-Howard

Annotate sequents with **proof terms** (= 'summary' of derivation).

\[
\begin{align*}
\Gamma & \vdash A \\
\Gamma & \vdash B \\
\Gamma & \vdash A \land B \\
\Gamma & \vdash A \times B \\
\Gamma & \vdash A
\end{align*}
\]

\[
\begin{align*}
\Gamma & \vdash M : A \\
\Gamma & \vdash N : B \\
\Gamma & \vdash \langle M, N \rangle : A \times B \\
\Gamma & \vdash \pi_1(M) : A
\end{align*}
\]

\[\pi_1(\langle M, N \rangle) \rightarrow M\]
Curry-Howard

Annotate sequents with **proof terms** (= ‘summary’ of derivation).

\[
\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \quad \rightarrow \quad \frac{\Gamma \vdash \lambda x.M : A \rightarrow B}{\Gamma, x : A \vdash M : B}
\]

\[
\frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \quad \rightarrow \quad \frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}
\]

\((\lambda x.M)N \rightarrow M[N/x]\)
Dual-context systems

- A kind of natural deduction with **two contexts**, introduced by Girard, developed by many over the 1990s (Davies and Pfenning, Andreoli, Wadler, Barber and Plotkin, ...)

- Judgments:

  \[ \Delta ; \Gamma \vdash M : A \]

  - modal assumptions
  - intuitionistic assumptions

  In our setting, \( \Delta = \) code/intensional variables,
  \( \Gamma = \) value/extension variables.
The Modal Rules

\[ \Delta ; \cdot \vdash M : A \]  \[ \Delta ; \Gamma \vdash \text{box } M : \square A \]  (\square I)  

\[ \Delta, u : A, \Delta' ; \Gamma \vdash u : A \]  \[ \Delta, u : A ; \Gamma \vdash N : C \]  \[ \Delta ; \Gamma \vdash \text{let box } u \leftarrow M \text{ in } N : C \]  (\square \varepsilon)  

• Hiding \( \Delta \), it looks just like simply-typed \( \lambda \)-calculus.

• This is augmented with the reduction

\[
\text{let box } u \leftarrow \text{box } M \text{ in } N \rightarrow N[M/u]
\]
The Modal Rules

\[
\frac{\Delta ; \cdot \vdash M : A}{\Delta ; \Gamma \vdash \text{box } M : \Box A} \quad (\Box I)
\]

\[
\frac{\Delta, u : A, \Delta' ; \Gamma \vdash u : A}{\Delta, u : A, \Delta' ; \Gamma \vdash \text{let box } u \leftarrow M \text{ in } N : C} \quad (\Box \text{var})
\]

\[
\frac{\Delta ; \Gamma \vdash M : \Box A \quad \Delta, u : A ; \Gamma \vdash N : C}{\Delta ; \Gamma \vdash \text{let box } u \leftarrow M \text{ in } N : C} \quad (\Box \text{E})
\]

**THEOREM (Davies-Pfenning).** This system captures S4; satisfies all the expected structural rules; and is confluent and strongly normalising.

\[
\text{let box } u \leftarrow \text{box } M \text{ in } N \longrightarrow N[M/u]
\]
Example

\[\text{app} \equiv \lambda f. \lambda x. \text{let box } u \leftarrow f \text{ in let box } v \leftarrow x \text{ in box } (u \; v)\]

\[\vdash \text{app} : \Box(A \rightarrow B) \rightarrow \Box A \rightarrow \Box B\]

\[\text{app } (\text{box } F')(\text{box } M) \rightarrow^* \text{box } (FM)\]
• Davies and Pfenning defined the above system for **homogeneous, staged metaprogramming** (POPL 1996), which was also implemented and tested.

• The purpose of that language was to separate the **static** and **dynamic** phases: some (modal) things would happen at compile-time, some (intuitionistic) things at run-time.

• But, even though mentioned in the paper (MSCS 2001), **intensionality** is completely absent! Everything is functional.
II. Intensional operations
Intensional Operations

The quintessential example: is the term an application?

\[
is\text{-app} \ (\text{box } PQ) \rightarrow \text{true} \\
is\text{-app} \ (\text{box } M) \rightarrow \text{false} \quad \text{if} \quad M \neq PQ
\]

This function can almost be considered a criterion of intensionality.
Intensional Operations: first attempt

- Let’s suppose any function on terms $f : \mathcal{T}(A) \rightarrow \mathcal{T}(B)$ can be included as a constant $\tilde{f} : \square A \rightarrow \square B$

  $$\tilde{f}(\text{box } M) \rightarrow \text{box } f(M)$$

  This is not confluent.

let box $u \leftarrow \text{box } PQ$ in is-app (box $u$)

is-app (box $PQ$)  let box $u \leftarrow \text{box } PQ$ in false

  $\downarrow$  $\downarrow$
  true  false
Intensional Operations, second attempt

- The problem: \( M \rightarrow N \) yet \( M[P/u] \not\rightarrow N[P/u] \)

- Violated because of constants like \texttt{is-app}.

- How to fix? Consider \texttt{substitutive} intensional operations
  \( f : T(A) \rightarrow T(B) \) such that \( f(N[P/u]) \equiv f(N)[P/u] \)

- Indeed a fix. But a standard \texttt{naturality argument} yields
  \( f(P) \equiv f(u[P/u]) \equiv f(u)[P/u] \)

- So already defined by \( \tilde{f} = \lambda x. \text{let box } u \leftarrow x \text{ in box } f(u) \)
  \( \tilde{f}(\text{box } M) \rightarrow^* \text{box } f(u)[M/u] \equiv \text{box } f(u[M/u]) \equiv \text{box } f(M) \)

- ... so we have achieved precisely \texttt{nothing}. 
Intensional Operations, with success

- Solution: restrict everything to closed terms:
  \[ \mathcal{T}(A) = \{ M \mid \cdot ; \cdot \vdash M : A \} \]
- ... and for each \( f : \mathcal{T}(A) \rightarrow \mathcal{T}(B) \)
  add a constant \( \tilde{f} : \Box A \rightarrow \Box B \)
  with reduction \( \tilde{f}(\text{box } M) \rightarrow \text{box } f(M) \)
  which happens only when \( M \) is closed.

- It so happens that this is confluent! Will see in a moment.
III. Intensional Recursion
Löb’s rule

Without further ado:

$$\Delta ; \Box A \vdash A$$

$$\Delta ; \Gamma \vdash \Box A$$

Observation by Abramsky: if one erases the boxes, it’s PCF!

We use this form, prompted by proof-theoretic considerations (see K, LICS 2017).
Löb’s rule

Without further ado:

\[
\begin{align*}
\Delta ; z : \square A & \vdash M : A \\
\hline
\Delta ; \Gamma & \vdash \text{fix } z \text{ in box } M : \square A
\end{align*}
\]

Observation by Abramsky: if one erases the boxes, it’s PCF!

We use this form, prompted by proof-theoretic considerations (see K, LICS 2017).
Łöb’s rule

Without further ado:

\[
\Delta ; z : \Box A \vdash M : A \\
\Delta \vdash \text{fix } z \text{ in box } M : \Box A
\]

\[
\text{fix } z \text{ in box } M \rightarrow \text{box } M[\text{fix } z \text{ in box } M/z]
\]

Observation by Abramsky: if one erases the boxes, it’s PCF!

We use this form, prompted by proof-theoretic considerations (see K, LICS 2017).
Löb’s rule

Without further ado:

\[
\frac{\Delta ; z : \ A \vdash M : A}{\Delta ; \Gamma \vdash \text{fix } z \text{ in } M : A}
\]

\[
\text{fix } z \text{ in } M \longrightarrow M[\text{fix } z \text{ in } M/z]
\]

Observation by Abramsky: if one erases the boxes, it’s PCF!

We use this form, prompted by proof-theoretic considerations (see K, LICS 2017).
An objection

“But Löb’s rule, in conjunction with \textbf{S4}, means that every type is inhabited!”

Indeed, if we let $\text{eval}_A \equiv \lambda x. \text{let box } u \leftarrow x \text{ in } u$

$\Omega_A \equiv \text{fix } z \text{ in box } \text{eval}_A \; z$

then $\vdash \Omega_A : \Box A$ and hence $\vdash \text{eval}_A \; \Omega_A : A$

with $\Omega_A \rightarrow \text{box } (\text{eval}_A \; \Omega_A)$ and $\text{eval}_A \; \Omega_A \rightarrow^* \text{eval}_A \; \Omega_A$

Answer: \textbf{It’s OK}. If we want general recursion, which the SRT gives, there will be non-normalising terms. Like PCF, not a logic but a programming language: \textbf{the terms still matter}. 
Confluence

- As long as we do not admit \( \frac{M \rightarrow N}{\text{box } M \rightarrow \text{box } N} \),

**THEOREM.** The resulting system is confluent, and hence consistent.

- The proof uses the standard parallel reduction method of Tait and Martin-Löf.

- The fact that intensional operations only reduce when the term is closed is crucial to the argument.
Conclusions
Conclusions

• We have constructed a modal $\lambda$-calculus, inspired by S4 and GL, with intensional (non-functional) operations, and intensional recursion.
Conclusions

• We have constructed a modal $\lambda$-calculus, inspired by S4 and GL, with intensional (non-functional) operations, and intensional recursion.

• We proved that it is confluent, and hence consistent. Price to pay: intensional operations on closed terms only (a thorn known to metaprogrammers...).
Conclusions

• We have constructed a modal $\lambda$-calculus, inspired by S4 and GL, with intensional (non-functional) operations, and intensional recursion.

• We proved that it is confluent, and hence consistent. Price to pay: intensional operations on closed terms only (a thorn known to metaprogrammers…).

• The type of the Gödel-Löb axiom is inhabited by an an intensional fixed point combinator. The standard fixed point combinator ($Y$) is definable in the system.
Conclusions

• We have constructed a modal $\lambda$-calculus, inspired by S4 and GL, with intensional (non-functional) operations, and intensional recursion.

• We proved that it is confluent, and hence consistent. Price to pay: intensional operations on closed terms only (a thorn known to metaprogrammers…).

• The type of the Gödel-Löb axiom is inhabited by an an intensional fixed point combinator. The standard fixed point combinator ($Y$) is definable in the system.

• Open questions:
Conclusions

• We have constructed a modal $\lambda$-calculus, inspired by S4 and GL, with intensional (non-functional) operations, and intensional recursion.

• We proved that it is confluent, and hence consistent. Price to pay: intensional operations on closed terms only (a thorn known to metaprogrammers...).

• The type of the Gödel-Löb axiom is inhabited by an an intensional fixed point combinator. The standard fixed point combinator ($Y$) is definable in the system.

• Open questions:

  • which are the correct primitives?
Conclusions

• We have constructed a modal $\lambda$-calculus, inspired by $S4$ and $GL$, with intensional (non-functional) operations, and intensional recursion.

• We proved that it is confluent, and hence consistent. Price to pay: intensional operations on closed terms only (a thorn known to metaprogrammers...).

• The type of the Gödel-Löb axiom is inhabited by an an intensional fixed point combinator. The standard fixed point combinator ($Y$) is definable in the system.

• Open questions:
  
  • which are the correct primitives?
  
  • what is the expressivity of this system? what does it do?
Conclusions

• We have constructed a modal \( \lambda \)-calculus, inspired by S4 and GL, with intensional (non-functional) operations, and intensional recursion.

• We proved that it is confluent, and hence consistent. Price to pay: intensional operations on closed terms only (a thorn known to metaprogrammers…).

• The type of the Gödel-Löb axiom is inhabited by an an intensional fixed point combinator. The standard fixed point combinator (\( Y \)) is definable in the system.

• Open questions:
  • which are the correct primitives?
  • what is the expressivity of this system? what does it do?

Thank you for your attention.