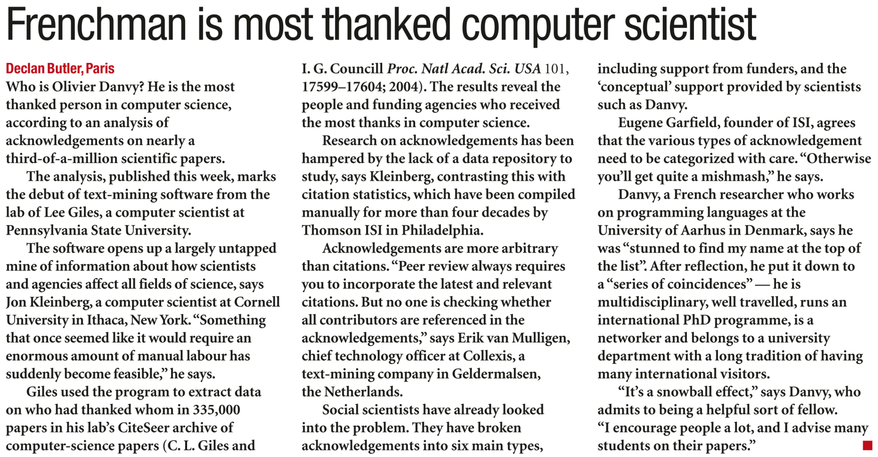
Encoding products

Sam Lindley

Olivierfest, Singapore, October 2025

Prologue

Most thanked computer scientist: Olivier Danvy



Frenchman is most thanked computer scientist

Declan Butler, Paris

Who is Olivier Danvy? He is the most thanked person in computer science, according to an analysis of acknowledgements on nearly a third-of-a-million scientific papers.

The analysis, published this week, marks the debut of text-mining software from the lab of Lee Giles, a computer scientist at Pennsylvania State University.

The software opens up a largely untapped mine of information about how scientists and agencies affect all fields of science, says Jon Kleinberg, a computer scientist at Cornell University in Ithaca, New York. "Something that once seemed like it would require an enormous amount of manual labour has suddenly become feasible," he says.

Giles used the program to extract data on who had thanked whom in 335,000 papers in his lab's CiteSeer archive of computer-science papers (C. L. Giles and I. G. Councill *Proc. Natl Acad. Sci. USA* 101, 17599–17604; 2004). The results reveal the people and funding agencies who received the most thanks in computer science.

Research on acknowledgements has been hampered by the lack of a data repository to study, says Kleinberg, contrasting this with citation statistics, which have been compiled manually for more than four decades by Thomson ISI in Philadelphia.

Acknowledgements are more arbitrary than citations. "Peer review always requires you to incorporate the latest and relevant citations. But no one is checking whether all contributors are referenced in the acknowledgements," says Erik van Mulligen, chief technology officer at Collexis, a text-mining company in Geldermalsen, the Netherlands.

Social scientists have already looked into the problem. They have broken acknowledgements into six main types, including support from funders, and the 'conceptual' support provided by scientists such as Danvy.

Eugene Garfield, founder of ISI, agrees that the various types of acknowledgement need to be categorized with care. "Otherwise you'll get quite a mishmash," he says.

Danvy, a French researcher who works on programming languages at the University of Aarhus in Denmark, says he was "stunned to find my name at the top of the list". After reflection, he put it down to a "series of coincidences" — he is multidisciplinary, well travelled, runs an international PhD programme, is a networker and belongs to a university department with a long tradition of having many international visitors.

"It's a snowball effect," says Danvy, who admits to being a helpful sort of fellow. "I encourage people a lot, and I advise many students on their papers." ■

What do we mean by expressive power?

What do we mean by expressive power?

Some possible answers:

- Computability

What do we mean by expressive power?

Some possible answers:

- Computability
- Algorithmic complexity

What do we mean by expressive power?

Some possible answers:

- Computability
- Algorithmic complexity
- **Macro expressiveness**
  - **local** versus **global** encodings
  - **compositionality** is a given

  [Felleisen, 1990, "On the expressive power of programming languages"]

Other aspects of expressive power

Types

parametric versus type-indexed encodings, type inference, subtyping, ...

Other aspects of expressive power

Types
: parametric versus type-indexed encodings, type inference, subtyping, …

Notions of observation
: convertibility, reduction, normalisation, denotation, up to administrative reduction, …

Other aspects of expressive power

Types

parametric versus type-indexed encodings, type inference, subtyping, ...

Notions of observation

convertibility, reduction, normalisation, denotation, up to administrative reduction, ...

Properties

soundness, completeness, contextual equivalence, adequacy, full abstraction, ...

Other aspects of expressive power

Types

parametric versus type-indexed encodings, type inference, subtyping, ...

Notions of observation

convertibility, reduction, normalisation, denotation, up to administrative reduction, ...

Properties

soundness, completeness, contextual equivalence, adequacy, full abstraction, ...

Structure

Galois connections, adjunctions, logical relations, bisimulations, ...

Quiz

Quiz

Can function types encode product types?

Quiz

Can function types encode product types?

Can positive iso-recursive types encode positive equi-recursive types?

Quiz

Can function types encode product types?

Can positive iso-recursive types encode positive equi-recursive types?

Can existential types encode universal types?

Quiz

Can function types encode product types?

Can positive iso-recursive types encode positive equi-recursive types?

Can existential types encode universal types?

Can simple-typed lambda calculus encode System F?

Quiz

Can function types encode product types?

Can positive iso-recursive types encode positive equi-recursive types?

Can existential types encode universal types?

Can simple-typed lambda calculus encode System F?

Can row polymorphism encode row subtyping?

Encoding products

Motivation

The standard Church encoding for a pair can only be ascribed a type in simply-typed lambda calculus if both components of the pair have the same type.

Motivation

The standard Church encoding for a pair can only be ascribed a type in simply-typed lambda calculus if both components of the pair have the same type.

$$\begin{aligned}
\textbf{pair}\ l\ r &\equiv \lambda s.s\ l\ r \\
\textbf{fst}\ p &\equiv p\ (\lambda x.\lambda y.x) \\
\textbf{snd}\ p &\equiv p\ (\lambda x.\lambda y.y)
\end{aligned}$$

Motivation

The standard Church encoding for a pair can only be ascribed a type in simply-typed lambda calculus if both components of the pair have the same type.

$$\begin{aligned} \textbf{pair}\ l\ r &\equiv \lambda s.s\ l\ r \\ \textbf{fst}\ p &\equiv p\ (\lambda x.\lambda y.x) \\ \textbf{snd}\ p &\equiv p\ (\lambda x.\lambda y.y) \end{aligned}$$

l and r, and hence x and y, must have the same type as the return type of s is fixed:

$$\begin{aligned} \textbf{fst}\ (\textbf{pair}\ l\ r) &\equiv (\lambda s.s\ l\ r)\ (\lambda x.\lambda y.x) \\ \textbf{snd}\ (\textbf{pair}\ l\ r) &\equiv (\lambda s.s\ l\ r)\ (\lambda x.\lambda y.y) \end{aligned}$$

Motivation

The standard Church encoding for a pair can only be ascribed a type in simply-typed lambda calculus if both components of the pair have the same type.

$$\begin{aligned} \textbf{pair}\ l\ r &\equiv \lambda s.s\ l\ r \\ \textbf{fst}\ p &\equiv p\ (\lambda x.\lambda y.x) \\ \textbf{snd}\ p &\equiv p\ (\lambda x.\lambda y.y) \end{aligned}$$

l and r, and hence x and y, must have the same type as the return type of s is fixed:

$$\begin{aligned} \textbf{fst}\ (\textbf{pair}\ l\ r) &\equiv (\lambda s.s\ l\ r)\ (\lambda x.\lambda y.x) \\ \textbf{snd}\ (\textbf{pair}\ l\ r) &\equiv (\lambda s.s\ l\ r)\ (\lambda x.\lambda y.y) \end{aligned}$$

Do alternative simply-typed encodings exist for heterogeneous pairs?

Call-by-name CPS

$$\begin{aligned}
\mathcal{N}[\![A]\!] &= (A^\star \to R) \to R \\
X^\star &= X \\
(A \times B)^\star &= \mathcal{N}[\![A]\!] \times \mathcal{N}[\![B]\!] \\
(A \to B)^\star &= \mathcal{N}[\![A]\!] \to \mathcal{N}[\![B]\!]
\end{aligned}$$

Call-by-name CPS

$$\begin{aligned}
\mathcal{N}[\![X]\!] &= (X \to R) \to R \\
\mathcal{N}[\![A \to B]\!] &= ((\mathcal{N}[\![A]\!] \to \mathcal{N}[\![B]\!]) \to R) \to R \\
\mathcal{N}[\![A \times B]\!] &= ((\mathcal{N}[\![A]\!] \times \mathcal{N}[\![B]\!]) \to R) \to R
\end{aligned}$$

Call-by-name CPS

$$\begin{array}{rcl}
\mathcal{N}[\![X]\!] & = & (X \to R) \to R \\
\mathcal{N}[\![A \to B]\!] & = & ((\mathcal{N}[\![A]\!] \to \mathcal{N}[\![B]\!]) \to R) \to R \\
\mathcal{N}[\![A \times B]\!] & = & ((\mathcal{N}[\![A]\!] \times \mathcal{N}[\![B]\!]) \to R) \to R
\end{array}$$

$$\begin{array}{rcl}
\mathcal{N}[\![x]\!] & = & \lambda k.x\ k \\
\mathcal{N}[\![\lambda x.M]\!] & = & \lambda k.k\ (\lambda x.\mathcal{N}[\![M]\!]) \\
\mathcal{N}[\![M\ N]\!] & = & \lambda k.\mathcal{N}[\![M]\!]\ (\lambda f.f\ \mathcal{N}[\![N]\!]\ k) \\
\mathcal{N}[\![\mathbf{pair}\ M\ N]\!] & = & \lambda k.k\ (\mathbf{pair}\ \mathcal{N}[\![M]\!]\ \mathcal{N}[\![N]\!]) \\
\mathcal{N}[\![\mathbf{fst}\ M]\!] & = & \lambda k.\mathcal{N}[\![M]\!]\ (\lambda p.(\mathbf{fst}\ p)\ k) \\
\mathcal{N}[\![\mathbf{snd}\ M]\!] & = & \lambda k.\mathcal{N}[\![M]\!]\ (\lambda p.(\mathbf{snd}\ p)\ k)
\end{array}$$

Call-by-name CPS

Products are encodable via a curried **global** CPS translation

$$\begin{array}{rcl}
\mathcal{C}[\![X]\!] & = & (X \to R) \to R \\
\mathcal{C}[\![A \to B]\!] & = & ((\mathcal{C}[\![A]\!] \to \mathcal{C}[\![B]\!]) \to R) \to R \\
\mathcal{C}[\![A \times B]\!] & = & (\mathcal{C}[\![A]\!] \to \mathcal{C}[\![B]\!] \to R) \to R
\end{array}$$

$$\begin{array}{rcl}
\mathcal{C}[\![x]\!] & = & \lambda k.x\ k \\
\mathcal{C}[\![\lambda x.M]\!] & = & \lambda k.k\ (\lambda x.\mathcal{C}[\![M]\!]) \\
\mathcal{C}[\![M\ N]\!] & = & \lambda k.\mathcal{C}[\![M]\!]\ (\lambda f.f\ \mathcal{C}[\![N]\!]\ k) \\
\mathcal{C}[\![\mathbf{pair}\ M\ N]\!] & = & \lambda k.k\ \mathcal{C}[\![M]\!]\ \mathcal{C}[\![N]\!] \\
\mathcal{C}[\![\mathbf{fst}\ M]\!] & = & \lambda k.\mathcal{C}[\![M]\!]\ (\lambda x.\lambda y.x\ k) \\
\mathcal{C}[\![\mathbf{snd}\ M]\!] & = & \lambda k.\mathcal{C}[\![M]\!]\ (\lambda x.\lambda y.y\ k)
\end{array}$$

Call-by-name CPS

Products are encodable via a curried **global** CPS translation

$$\begin{array}{rcl}
\mathcal{C}[\![X]\!] & = & (X \to R) \to R \\
\mathcal{C}[\![A \to B]\!] & = & ((\mathcal{C}[\![A]\!] \to \mathcal{C}[\![B]\!]) \to R) \to R \\
\mathcal{C}[\![A \times B]\!] & = & (\mathcal{C}[\![A]\!] \to \mathcal{C}[\![B]\!] \to R) \to R
\end{array}$$

$$\begin{array}{rcl}
\mathcal{C}[\![x]\!] & = & \lambda k.x\ k \\
\mathcal{C}[\![\lambda x.M]\!] & = & \lambda k.k\ (\lambda x.\mathcal{C}[\![M]\!]) \\
\mathcal{C}[\![M\ N]\!] & = & \lambda k.\mathcal{C}[\![M]\!]\ (\lambda f.f\ \mathcal{C}[\![N]\!]\ k) \\
\mathcal{C}[\![\mathbf{pair}\ M\ N]\!] & = & \lambda k.k\ \mathcal{C}[\![M]\!]\ \mathcal{C}[\![N]\!] \\
\mathcal{C}[\![\mathbf{fst}\ M]\!] & = & \lambda k.\mathcal{C}[\![M]\!]\ (\lambda x.\lambda y.x\ k) \\
\mathcal{C}[\![\mathbf{snd}\ M]\!] & = & \lambda k.\mathcal{C}[\![M]\!]\ (\lambda x.\lambda y.y\ k)
\end{array}$$

What about a **local** encoding?

Localising CPS

Untyped

$$
\begin{aligned}
\mathcal{U}[\![\textbf{pair}\ M\ N]\!] &= \lambda s.s\,\mathcal{U}[\![M]\!]\,\mathcal{U}[\![N]\!] \\
\mathcal{U}[\![\textbf{fst}\ M]\!] &= \mathcal{U}[\![M]\!]\ (\lambda x.\lambda y.x) \\
\mathcal{U}[\![\textbf{snd}\ M]\!] &= \mathcal{U}[\![M]\!]\ (\lambda x.\lambda y.y)
\end{aligned}
$$

Localising CPS

Untyped

$$\begin{array}{rcl}
\mathcal{U}[\![\mathbf{pair}\ M\ N]\!] & = & \lambda s.s\,\mathcal{U}[\![M]\!]\,\mathcal{U}[\![N]\!] \\
\mathcal{U}[\![\mathbf{fst}\ M]\!] & = & \mathcal{U}[\![M]\!]\ (\lambda x.\lambda y.x) \\
\mathcal{U}[\![\mathbf{snd}\ M]\!] & = & \mathcal{U}[\![M]\!]\ (\lambda x.\lambda y.y)
\end{array}$$

Simply typed — homogeneous products

$$\begin{array}{rcl}
\mathcal{H}[\![A \times A]\!] & = & (\mathcal{H}[\![A]\!] \to \mathcal{H}[\![A]\!] \to \mathcal{H}[\![A]\!]) \to \mathcal{H}[\![A]\!] \\
\mathcal{H}[\![\mathbf{pair}\ M^A\ N^A]\!] & = & \lambda s^{\mathcal{H}[\![A]\!] \to \mathcal{H}[\![A]\!] \to \mathcal{H}[\![A]\!]}.s\ \mathcal{H}[\![M]\!]\ \mathcal{H}[\![N]\!] \\
\mathcal{H}[\![\mathbf{fst}\ M^{A \times A}]\!] & = & \mathcal{H}[\![M]\!]\ (\lambda x^{\mathcal{H}[\![A]\!]}.\lambda y^{\mathcal{H}[\![A]\!]}.x) \\
\mathcal{H}[\![\mathbf{snd}\ M^{A \times A}]\!] & = & \mathcal{H}[\![M]\!]\ (\lambda x^{\mathcal{H}[\![A]\!]}.\lambda y^{\mathcal{H}[\![A]\!]}.y)
\end{array}$$

Localising CPS

Untyped

$$\begin{array}{rcl} \mathcal{U}[\![\textbf{pair}\ M\ N]\!] & = & \lambda s.s\,\mathcal{U}[\![M]\!]\,\mathcal{U}[\![N]\!] \\ \mathcal{U}[\![\textbf{fst}\ M]\!] & = & \mathcal{U}[\![M]\!]\ (\lambda x.\lambda y.x) \\ \mathcal{U}[\![\textbf{snd}\ M]\!] & = & \mathcal{U}[\![M]\!]\ (\lambda x.\lambda y.y) \end{array}$$

Simply typed — homogeneous products

$$\begin{array}{rcl} \mathcal{H}[\![A \times A]\!] & = & (\mathcal{H}[\![A]\!] \to \mathcal{H}[\![A]\!] \to \mathcal{H}[\![A]\!]) \to \mathcal{H}[\![A]\!] \\ \mathcal{H}[\![\textbf{pair}\ M^A\ N^A]\!] & = & \lambda s^{\mathcal{H}[\![A]\!] \to \mathcal{H}[\![A]\!] \to \mathcal{H}[\![A]\!]}.s\ \mathcal{H}[\![M]\!]\ \mathcal{H}[\![N]\!] \\ \mathcal{H}[\![\textbf{fst}\ M^{A\times A}]\!] & = & \mathcal{H}[\![M]\!]\ (\lambda x^{\mathcal{H}[\![A]\!]}.\lambda y^{\mathcal{H}[\![A]\!]}.x) \\ \mathcal{H}[\![\textbf{snd}\ M^{A\times A}]\!] & = & \mathcal{H}[\![M]\!]\ (\lambda x^{\mathcal{H}[\![A]\!]}.\lambda y^{\mathcal{H}[\![A]\!]}.y) \end{array}$$

Polymorphic

$$\begin{array}{rcl} \mathcal{F}[\![A \times B]\!] & = & \forall Z.(\mathcal{F}[\![A]\!] \to \mathcal{F}[\![B]\!] \to Z) \to Z \\ \mathcal{F}[\![\textbf{pair}_{A,B}\ M\ N]\!] & = & \Lambda Z.\lambda s^{\mathcal{F}[\![A]\!] \to \mathcal{F}[\![B]\!] \to Z}.s\ \mathcal{F}[\![M]\!]\ \mathcal{F}[\![N]\!] \\ \mathcal{F}[\![\textbf{fst}_{A,B}\ M]\!] & = & \mathcal{F}[\![M]\!]\ \mathcal{F}[\![A]\!]\ (\lambda x^{\mathcal{F}[\![A]\!]}.\lambda y^{\mathcal{F}[\![B]\!]}.x) \\ \mathcal{F}[\![\textbf{snd}_{A,B}\ M]\!] & = & \mathcal{F}[\![M]\!]\ \mathcal{F}[\![B]\!]\ (\lambda x^{\mathcal{F}[\![A]\!]}.\lambda y^{\mathcal{F}[\![B]\!]}.y) \end{array}$$

No local encoding of $X \times Y$

We seek β-normal forms $\mathsf{fst}_{X,Y}$ and $\mathsf{snd}_{X,Y}$ such that:

$$\begin{array}{rcl} [\![p : X \times Y \vdash \mathbf{fst}\ p : X]\!] & = & p : [\![X \times Y]\!] \vdash \mathsf{fst}_{X,Y} : X \\ [\![p : X \times Y \vdash \mathbf{snd}\ p : Y]\!] & = & p : [\![X \times Y]\!] \vdash \mathsf{snd}_{X,Y} : Y \end{array}$$

No local encoding of $X \times Y$

We seek β-normal forms $\mathsf{fst}_{X,Y}$ and $\mathsf{snd}_{X,Y}$ such that:

$$\begin{aligned}
[\![p : X \times Y \vdash \textbf{fst}\ p : X]\!] &= p : [\![X \times Y]\!] \vdash \mathsf{fst}_{X,Y} : X \\
[\![p : X \times Y \vdash \textbf{snd}\ p : Y]\!] &= p : [\![X \times Y]\!] \vdash \mathsf{snd}_{X,Y} : Y
\end{aligned}$$

So we must have $m, n, M_1, ..., M_m, N_1, ..., N_n$ such that:

$$\begin{aligned}
\mathsf{fst}_{X,Y} &= p\ M_1\ \ldots\ M_m \\
\mathsf{snd}_{X,Y} &= p\ N_1\ \ldots\ N_n
\end{aligned}$$

No local encoding of $X \times Y$

We seek β-normal forms $\mathsf{fst}_{X,Y}$ and $\mathsf{snd}_{X,Y}$ such that:

$$\begin{array}{rcl} [\![p : X \times Y \vdash \mathbf{fst}\ p : X]\!] & = & p : [\![X \times Y]\!] \vdash \mathsf{fst}_{X,Y} : X \\ [\![p : X \times Y \vdash \mathbf{snd}\ p : Y]\!] & = & p : [\![X \times Y]\!] \vdash \mathsf{snd}_{X,Y} : Y \end{array}$$

So we must have $m, n, M_1, ..., M_m, N_1, ..., N_n$ such that:

$$\begin{array}{rcl} \mathsf{fst}_{X,Y} & = & p\ M_1\ \ldots\ M_m \\ \mathsf{snd}_{X,Y} & = & p\ N_1\ \ldots\ N_n \end{array}$$

The typing rule for application means that we also have

$$A_1 \to \cdots \to A_m \to X \quad = \quad [\![X \times Y]\!] \quad = \quad B_1 \to \cdots \to B_n \to Y$$

where:

$$\begin{array}{c} (p : [\![X \times Y]\!] \vdash M_i : A_i)_{1 \leq i \leq m} \\ (p : [\![X \times Y]\!] \vdash N_j : B_i)_{1 \leq j \leq n} j \end{array}$$

No local encoding of $X \times Y$

We seek β-normal forms $\mathsf{fst}_{X,Y}$ and $\mathsf{snd}_{X,Y}$ such that:

$$\begin{array}{rcl} [\![p : X \times Y \vdash \mathbf{fst}\ p : X]\!] & = & p : [\![X \times Y]\!] \vdash \mathsf{fst}_{X,Y} : X \\ [\![p : X \times Y \vdash \mathbf{snd}\ p : Y]\!] & = & p : [\![X \times Y]\!] \vdash \mathsf{snd}_{X,Y} : Y \end{array}$$

So we must have $m, n, M_1, ..., M_m, N_1, ..., N_n$ such that:

$$\begin{array}{rcl} \mathsf{fst}_{X,Y} & = & p\ M_1\ \ldots\ M_m \\ \mathsf{snd}_{X,Y} & = & p\ N_1\ \ldots\ N_n \end{array}$$

The typing rule for application means that we also have

$$A_1 \to \cdots \to A_m \to X \quad = \quad [\![X \times Y]\!] \quad = \quad B_1 \to \cdots \to B_n \to Y$$

where:

$$\begin{array}{c} (p : [\![X \times Y]\!] \vdash M_i : A_i)_{1 \leq i \leq m} \\ (p : [\![X \times Y]\!] \vdash N_j : B_i)_{1 \leq j \leq n} j \end{array}$$

But these equations could only hold if X and Y were the same type!

Hang on a minute!

Type-indexed local encodings of products are well-known in PCF and System T.

Examples:

- [Longley and Normann, 2015]
  Higher-order computability
- [Kiselyov, 2021]
  `http://okmij.org/ftp/Computation/simple-encodings.html#product`

How do we reconcile the existence of such encodings with the non-existence result?

No local encoding of $X \times (X \to X)$

Consider $X \times (X \to X)$. We seek β-normal forms $\mathrm{fst}_{X,X\to X}$ and $\mathrm{snd}_{X,X\to X}$ such that:

$$\begin{aligned}
[\![p : X \times (X \to X) \vdash \mathbf{fst}\ p : X]\!] &= p : [\![X \times (X \to X)]\!] \vdash \mathrm{fst}_{X,X\to X} : X \\
[\![p : X \times (X \to X) \vdash \mathbf{snd}\ p : X \to X]\!] &= p : [\![X \times (X \to X)]\!] \vdash \mathrm{snd}_{X,X\to X} : X \to X
\end{aligned}$$

No local encoding of $X \times (X \to X)$

Consider $X \times (X \to X)$. We seek β-normal forms $\mathsf{fst}_{X,X\to X}$ and $\mathsf{snd}_{X,X\to X}$ such that:

$$\begin{array}{rcl} [\![p : X \times (X \to X) \vdash \mathbf{fst}\ p : X]\!] & = & p : [\![X \times (X \to X)]\!] \vdash \mathsf{fst}_{X,X\to X} : X \\ [\![p : X \times (X \to X) \vdash \mathbf{snd}\ p : X \to X]\!] & = & p : [\![X \times (X \to X)]\!] \vdash \mathsf{snd}_{X,X\to X} : X \to X \end{array}$$

As before, $\mathsf{fst}_{X,X\to X}$ must be of the form

$$p\ M_1\ \ldots\ M_m$$

and hence:

$$[\![X \times (X \to X)]\!] = A_1 \to \cdots \to A_m \to X$$

No local encoding of $X \times (X \to X)$ (continued)

Two choices for $\mathsf{snd}_{X, X \to X}$:

No local encoding of $X \times (X \to X)$ (continued)

Two choices for $\mathsf{snd}_{X,X\to X}$:

1. $p\ N_1\ \ldots\ N_{m-1}$

No local encoding of $X \times (X \to X)$ (continued)

Two choices for $\mathsf{snd}_{X,X\to X}$:

1. $p\ N_1\ \dots\ N_{m-1}$

$$\begin{aligned} \Longrightarrow\ A_m = X \text{ and } M_m &= p\ M'_1\ \dots\ M'_m \\ M'_m &= p\ M''_1\ \dots\ M''_m \\ &\dots \end{aligned}$$

No local encoding of $X \times (X \to X)$ (continued)

Two choices for $\mathsf{snd}_{X,X\to X}$:

1. $p\ N_1\ \dots\ N_{m-1}$

$$\begin{aligned} \Longrightarrow\ A_m = X \text{ and } M_m &= p\ M'_1\ \dots\ M'_m \\ M'_m &= p\ M''_1\ \dots\ M''_m \\ &\dots \end{aligned}$$

No finite such snd can exist.

No local encoding of $X \times (X \to X)$ (continued)

Two choices for $\mathsf{snd}_{X,X\to X}$:

1. $p\ N_1\ \dots\ N_{m-1}$

$$\Longrightarrow\ A_m = X \text{ and } \begin{array}{l} M_m = p\ M'_1\ \dots\ M'_m \\ M'_m = p\ M''_1\ \dots\ M''_m \\ \dots \end{array}$$

No finite such snd can exist.

2. $\lambda z.N'$

No local encoding of $X \times (X \to X)$ (continued)

Two choices for $\mathsf{snd}_{X, X \to X}$:

1. $p\ N_1\ \ldots\ N_{m-1}$

$$\begin{aligned} \Longrightarrow\ A_m = X \text{ and } & M_m = p\ M'_1\ \ldots\ M'_m \\ & M'_m = p\ M''_1\ \ldots\ M''_m \\ & \ldots \end{aligned}$$

   No finite such snd can exist.

2. $\lambda z.N'$

$$\Longrightarrow\ [\![\mathbf{snd}\ (\mathbf{pair}\ x\ y)]\!] = \lambda z.N'[[\![\mathbf{pair}\ x\ y]\!]/p]$$

No local encoding of $X \times (X \to X)$ (continued)

Two choices for $\mathsf{snd}_{X,X\to X}$:

1. $p\ N_1\ \ldots\ N_{m-1}$

   $$\Longrightarrow\ A_m = X \text{ and } \begin{array}{l} M_m = p\ M'_1\ \ldots\ M'_m \\ M'_m = p\ M''_1\ \ldots\ M''_m \\ \ldots \end{array}$$

   No finite such snd can exist.

2. $\lambda z.N'$

   $$\Longrightarrow\ [\![\mathbf{snd}\ (\mathbf{pair}\ x\ y)]\!] = \lambda z.N'[[\![\mathbf{pair}\ x\ y]\!]/p]$$

   No lambda abstraction can be β-converted to y.

No local encoding of $X \times (X \to X)$ (continued)

Two choices for $\mathsf{snd}_{X,X\to X}$:

1. $p\ N_1\ \ldots\ N_{m-1}$

   $$\Longrightarrow\ A_m = X \text{ and } \begin{array}{l} M_m = p\ M'_1\ \ldots\ M'_m \\ M'_m = p\ M''_1\ \ldots\ M''_m \\ \ldots \end{array}$$

   No finite such snd can exist.

2. $\lambda z.N'$

   $$\Longrightarrow\ [\![\mathbf{snd}\ (\mathbf{pair}\ x\ y)]\!] = \lambda z.N'[[\![\mathbf{pair}\ x\ y]\!]/p]$$

   No lambda abstraction can be β-converted to y.

   But what if we allow η-conversion?

Local encoding of $X \times (X \to X)$ with η

$$\mathcal{E}[\![X \times (X \to X)]\!] \quad = \quad (X \to (X \to X) \to X) \to X$$

$$\begin{aligned}
\mathcal{E}[\![\textbf{pair}\ M^{X}\ N^{X \to X}]\!] &= \lambda f.f\ \mathcal{E}[\![M]\!]\ \mathcal{E}[\![N]\!] \\
\mathcal{E}[\![\textbf{fst}\ M^{X \times (X \to X)}]\!] &= \mathcal{E}[\![M]\!]\ (\lambda x\ y.x) \\
\mathcal{E}[\![\textbf{snd}\ M^{X \times (X \to X)}]\!] &= \lambda z.\mathcal{E}[\![M]\!]\ (\lambda x\ y.y\ z)
\end{aligned}$$

Local encoding of $X \times (X \to X)$ with η

$$
\begin{array}{rcl}
\mathcal{E}[\![X \times (X \to X)]\!] & = & (X \to (X \to X) \to X) \to X \\
\mathcal{E}[\![\mathbf{pair}\ M^X\ N^{X \to X}]\!] & = & \lambda f.f\ \mathcal{E}[\![M]\!]\ \mathcal{E}[\![N]\!] \\
\mathcal{E}[\![\mathbf{fst}\ M^{X \times (X \to X)}]\!] & = & \mathcal{E}[\![M]\!]\ (\lambda x\ y.x) \\
\mathcal{E}[\![\mathbf{snd}\ M^{X \times (X \to X)}]\!] & = & \lambda z.\mathcal{E}[\![M]\!]\ (\lambda x\ y.y\ z)
\end{array}
$$

Now we have

$$
\begin{array}{l}
\mathcal{E}[\![\mathbf{fst}\ (\mathbf{pair}\ x\ y)]\!] \sim_\beta x \\
\mathcal{E}[\![\mathbf{snd}\ (\mathbf{pair}\ x\ y)]\!] \sim_\beta \lambda z.y\ z \sim_\eta y
\end{array}
$$

Local encoding of $A \times B$ with η and a single base type X

$$\begin{array}{rcl}
\mathcal{E}[\![A \times B]\!] & = & (\mathcal{E}[\![A]\!] \to \mathcal{E}[\![B]\!] \to X) \to X \\
\mathcal{E}[\![\textbf{pair}\ M\ N]\!] & = & \lambda f.f\ \mathcal{E}[\![M]\!]\ \mathcal{E}[\![N]\!] \\
\mathcal{E}[\![\textbf{fst}\ M^{A_1 \to \ldots A_m \to X, B}]\!] & = & \lambda z_1\ \ldots\ z_m.\mathcal{E}[\![M]\!]\ (\lambda x\ y.x\ z_1\ \ldots\ z_m) \\
\mathcal{E}[\![\textbf{snd}\ M^{A, B_1 \to \ldots B_n \to X}]\!] & = & \lambda z_1\ \ldots\ z_n.\mathcal{E}[\![M]\!]\ (\lambda x\ y.y\ z_1\ \ldots\ z_n)
\end{array}$$

Local encoding of $A \times B$ with η and a single base type X

$$\begin{array}{rcl}
\mathcal{E}[\![A \times B]\!] & = & (\mathcal{E}[\![A]\!] \to \mathcal{E}[\![B]\!] \to X) \to X \\
\mathcal{E}[\![\textbf{pair}\ M\ N]\!] & = & \lambda f.f\ \mathcal{E}[\![M]\!]\ \mathcal{E}[\![N]\!] \\
\mathcal{E}[\![\textbf{fst}\ M^{A_1 \to \ldots A_m \to X, B}]\!] & = & \lambda z_1\ \ldots\ z_m.\mathcal{E}[\![M]\!]\ (\lambda x\ y.x\ z_1\ \ldots\ z_m) \\
\mathcal{E}[\![\textbf{snd}\ M^{A, B_1 \to \ldots B_n \to X}]\!] & = & \lambda z_1\ \ldots\ z_n.\mathcal{E}[\![M]\!]\ (\lambda x\ y.y\ z_1\ \ldots\ z_n)
\end{array}$$

This is a **type-indexed** local encoding.

Can function types encode product types?

Can function types encode product types?

It depends...

Can function types encode product types?

It depends...

- Parametric global CPS encoding
- Parametric local Church encodings
  - untyped
  - simple types, but only homogeneous products
  - polymorphic
- Multiple base types — no local encoding
- Single base type without η — no local encoding
- Single base type with η — type-indexed local encoding

Can function types encode product types?

It depends...

- Parametric global CPS encoding
- Parametric local Church encodings
  - untyped
  - simple types, but only homogeneous products
  - polymorphic
- Multiple base types — no local encoding
- Single base type without η — no local encoding
- Single base type with η — type-indexed local encoding

*none of these encodings preserves the η-rule for products

Epilogue

Closing thoughts

Expressiveness results are remarkably fragile

Closing thoughts

Expressiveness results are remarkably fragile

Several ideas and results in this talk are implicit in a 1974 paper by Barendregt (thanks to an anonymous reviewer for making the connection!)
[Barendregt, 1974, "Pairing without conventional restraints"]

Closing thoughts

Expressiveness results are remarkably fragile

Several ideas and results in this talk are implicit in a 1974 paper by Barendregt (thanks to an anonymous reviewer for making the connection!)
[Barendregt, 1974, “Pairing without conventional restraints”]

My thoughts on expressiveness are rooted in Olivier’s work on lambda calculus, continuations, and program transformations

Thank you Olivier