

rest of computation

# Continuations from Three Angles

Youyou Cong (Tokyo Institute of Technology)

FLOPS 2024 Keynote

# Continuations in real life

Thu 16 May			
Displayed time zone: <b>Osaka, Sapporo, Tokyo</b> <a href="#">change</a>			
	09:30 - 10:30	<b>Invited Talk</b> Chair(s): <b>Oleg Kiselyov</b> Tohoku University	<b>FLOPS 2024</b>
Now	09:30 60m	☆ <b>Verse: A New Functional Logic Language</b> <i>Keynote</i> Lennart Augustsson Epic Games	
	10:30 - 11:00	<b>Coffee break</b>	<b>FLOPS 2024</b>
in 59 min			
		⋮	
	15:30 60m	☆ <b>Continuations from Three Angles</b> <i>Keynote</i> Youyou Cong Tokyo Institute of Technology	
	16:30 - 16:40	<b>Closing</b>	<b>FLOPS 2024</b>

# Delimited continuations in real life

Thu 16 May			
Displayed time zone: <b>Osaka, Sapporo, Tokyo</b> <a href="#">change</a>			
	09:30 - 10:30	<b>Invited Talk</b> Chair(s): <b>Oleg Kiselyov</b> Tohoku University	FLOPS 2024
Now	09:30 60m	☆ <b>Verse: A New Functional Logic Language</b> <i>Keynote</i> Lennart Augustsson Epic Games	
	10:30 - 11:00	<b>Coffee break</b>	FLOPS 2024
in 59 min			
		⋮	
	12:00 - 14:00	<b>Lunch</b>	FLOPS 2024
	14:00 - 19:45	<b>Excursion and banquet</b>	FLOPS 2024

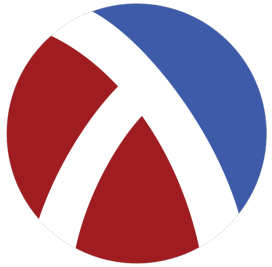
---

Fri 17 May

# Expressiveness of continuations

- Exceptions
- Nondeterminism
- State
- Generators/iterators
- Futures/promises
- Async/await

# Implementations of continuations



Racket



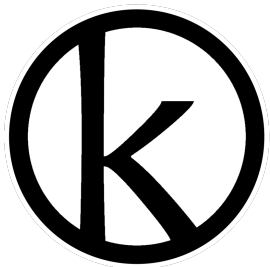
OCaml



Haskell



Prolog



Koka

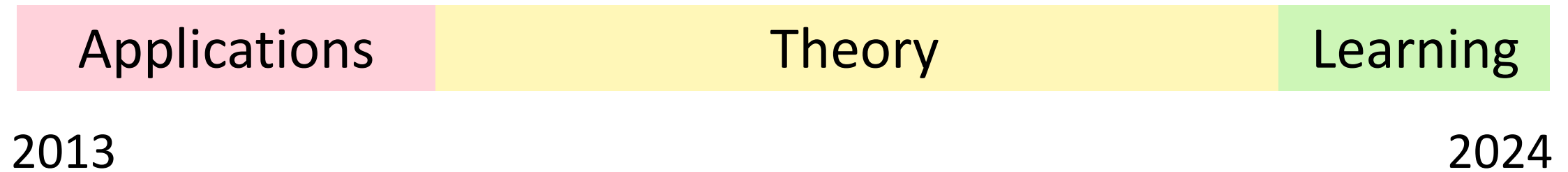


Links



Effekt

# My continuations research



# 2013: Joining NL lab at Ochanomizu



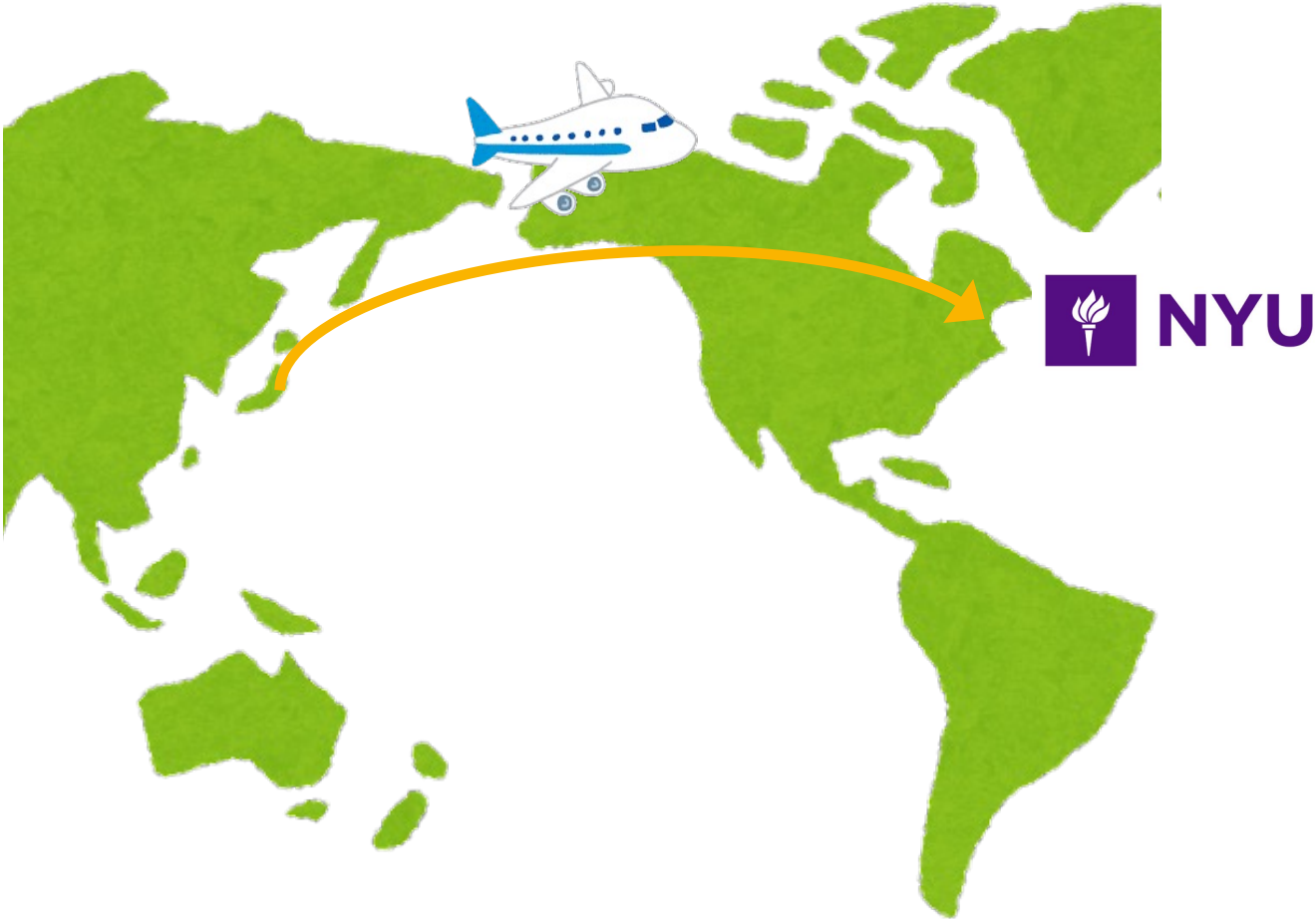
## Bekkilab Official Website

Give everyone on earth one source of truth for logic, language and information science.

戸次研究室では、数理言語学の研究をしています。数理言語学とは、数理モデルを用いて自然言語の構造を解明する学問です。主に、組合せ範疇文法(Combinatory Categorical Grammar) / 依存型意味論(Dependent Type Theory) / 圏論(Category theory)などを用いて言語現象（日本語の統語構造 / 照応・前提 / 談話関係 / 一般化量化子 / モダリティ / 慣習的含み / 敬語 / フォーカス / 含意関係認識 / 日本語の機能表現 / 日本語のテンス・アスペクトなど）全体を統一的に説明する理論と、その背後にある構造を追究しています。



# A research visit to NYU



Chris Barker



# Calculating the meaning of sentences

John loves Mary  $\longrightarrow$  *Love(j, m)*

# Calculating the meaning of sentences

John  $\triangleleft$  (loves  $\triangleright$  Mary)  $\longrightarrow$   $Love(j, m)$

John =  $j$

Mary =  $m$

loves =  $\lambda o. \lambda s. Love(s, o)$

# Quantification [Shan '04]

John loves everyone  $\longrightarrow \forall x. \textit{Love}(j, x)$

# Quantification [Shan '04]

John  $\triangleleft$  (loves  $\triangleright$  everyone)  $\longrightarrow$   $\forall x. \textit{Love}(j, x)$

# Quantification [Shan '04]

John  $\triangleleft$  (loves  $\triangleright$  everyone)  $\longrightarrow \forall x. Love(j, x)$

John =  $j$

loves =  $\lambda o. \lambda s. Love(s, o)$

everyone = ??

# Quantification [Shan '04]

John  $\triangleleft$  (loves  $\triangleright$  everyone)  $\longrightarrow \forall x. Love(j, x)$

John =  $j$

loves =  $\lambda o. \lambda s. Love(s, o)$

everyone =  $Sk. \forall x. k x$

shift

# Quantification [Shan '04]

John  $\triangleleft$  (loves  $\triangleright$  everyone)  $\longrightarrow \forall x. Love(j, x)$

reset  $\langle E[Sk.e] \rangle \rightarrow \langle e[\lambda y. \langle E[y] \rangle / k] \rangle$   
shift

# Quantification [Shan '04]

$\langle \text{John} \triangleleft (\text{loves} \triangleright \text{everyone}) \rangle \longrightarrow \forall x. \text{Love}(j, x)$

John =  $j$

loves =  $\lambda o. \lambda s. \text{Love}(s, o)$

everyone =  $Sk. \forall x. k x$



# Deep vs. shallow semantics in PL

deep

(Danvy & Filinski's shift)

$\langle E[Sk.e] \rangle$



$k = \lambda y. \langle E[y] \rangle$

shallow

(Felleisen's control)

$\langle E[Ck.e] \rangle$



$k = \lambda y. E[y]$

# Deep vs. shallow semantics in NL [Cong+ '15]

John introduced someone to everyone

$\exists x. \forall y. \text{Introduce}(j, x, y)$

direct scope reading  
(natural)

$\forall y. \exists x. \text{Introduce}(j, x, y)$

inverse scope reading  
(hard)

# Deep vs. shallow semantics in NL [Cong+ '15]

John introduced someone to everyone

$\exists x. \forall y. \text{Introduce}(j, x, y)$

$\forall y. \exists x. \text{Introduce}(j, x, y)$

someone =  $\lambda k. \exists x. k\ x$

everyone =  $\lambda k. \forall y. k\ y$

deep, i.e., continuation  
includes reset

# Deep vs. shallow semantics in NL [Cong+ '15]

John introduced someone to everyone

$\exists x. \forall y. \textit{Introduce}(j, x, y)$

$\forall y. \exists x. \textit{Introduce}(j, x, y)$

shallow, i.e., continuation  
does not include reset

someone =  $\mathbf{C}k. \exists x. k\ x$

everyone =  $\mathbf{C}k. \forall y. k\ y$

# Deep vs. shallow semantics in NL [Cong+ '15]

John introduced someone to everyone

$\exists x. \forall y. \text{Introduce}(j, x, y)$

$\forall y. \exists x. \text{Introduce}(j, x, y)$



someone =  $\mathbf{S}k. \exists x. k x$

everyone =  $\mathbf{S}k. \forall y. k y$

someone =  $\mathbf{C}k. \exists x. k x$

everyone =  $\mathbf{C}k. \forall y. k y$

# 2016: Joining PL lab at Ochanomizu

## Asai Laboratory

トップ

研究内容など

メンバー

今年度の授業

過去のニュース

書籍

お茶の水女子大学理学部情報科学科 浅井研究室のホームページです。

■ [浅井先生のHPはこちら](#)

研究室について



あさいです。

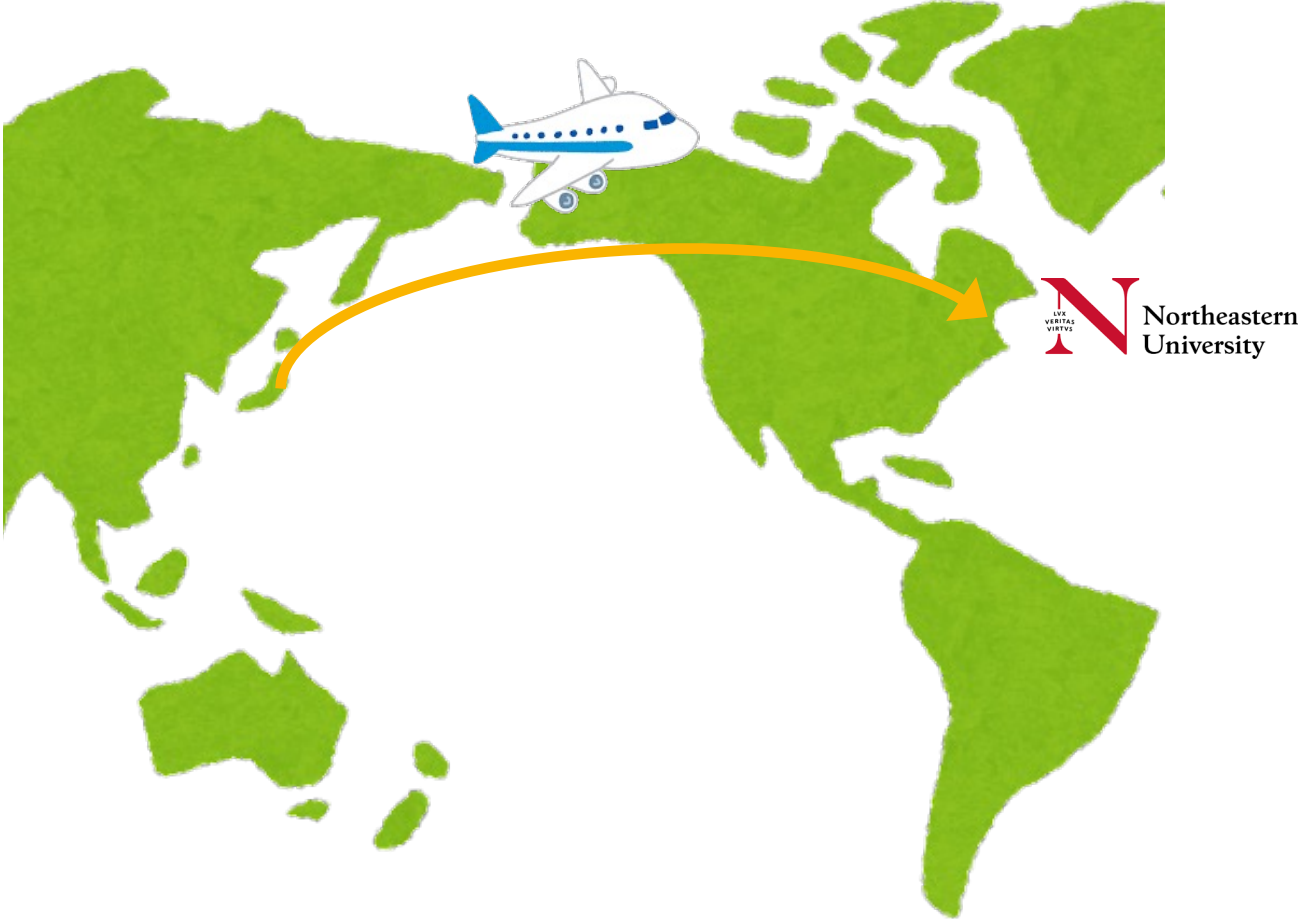
プログラミング言語の基礎理論を研究しています。

どうすれば無駄なくプログラムを実行できるか、楽にプログラムを作れるようになるか、プログラムの誤りを減らせるか。

対象をよく理解しその本質をとらえると、自然と物事は簡単なものの組み合わせになってきます。

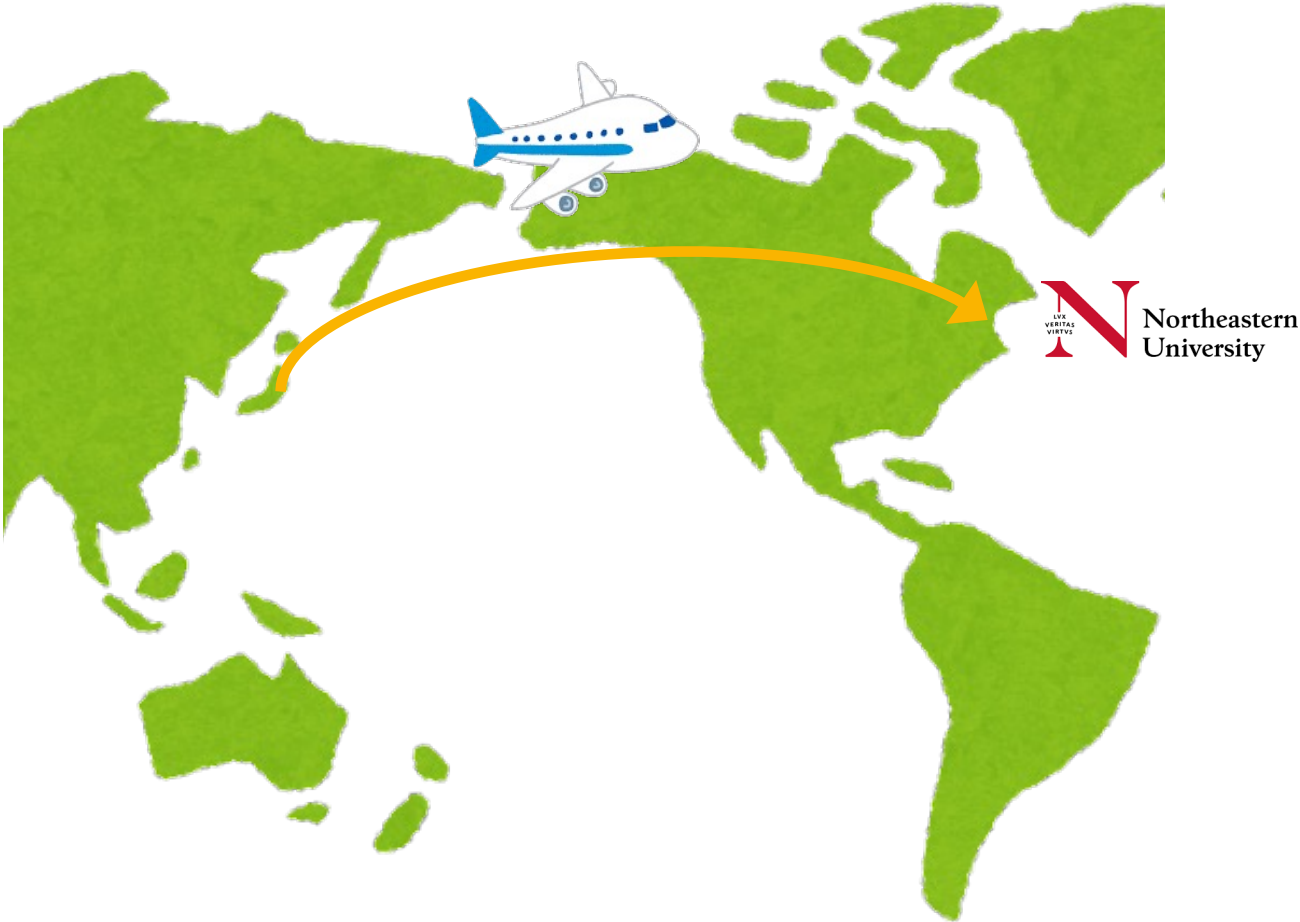


# A research visit to NEU



Matthias Felleisen

# A research visit to NEU



William Bowman



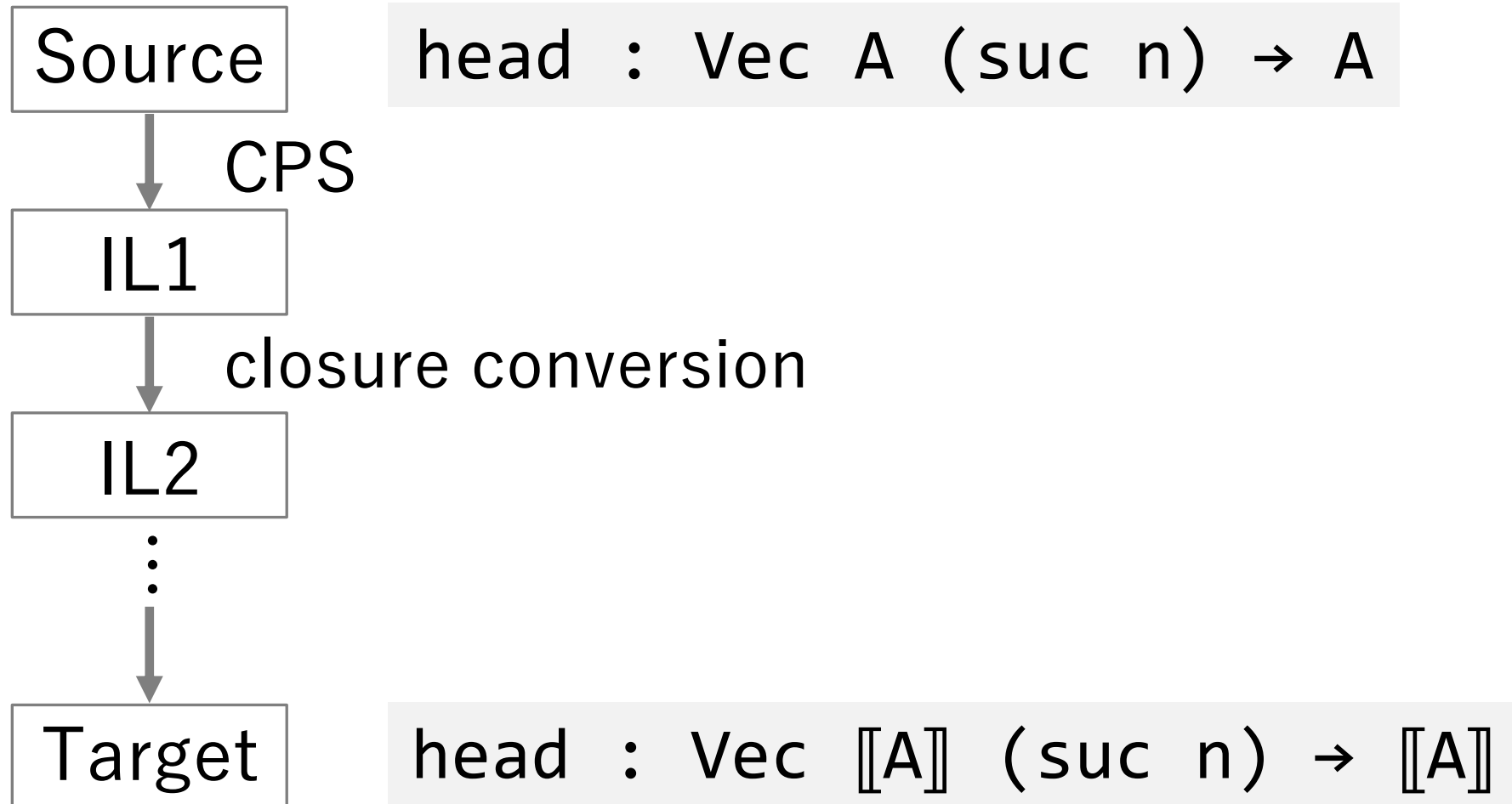
Nick Rioux



Amal Ahmed



# Dependent type preserving compilation



# Known challenge

Barthe,  
Hatcliff,  
Sørensen  
HOSC '99

they [CPS translations] cannot be extended readily to  $\Sigma$  types as the usual translation for pairs does not preserve typing.

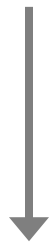
$$\frac{\Gamma \vdash e_1 : A \quad \Gamma \vdash e_2 : B[e_1/x]}{\Gamma \vdash (e_1, e_2) : \Sigma x : A. B}$$

$$\frac{\Gamma \vdash e : \Sigma x : A. B}{\Gamma \vdash \text{snd } e : B[\text{fst } e/x]}$$

# CPS'ing second projection (simply typed)

$\text{snd } e : B$

where  $e : A \times B$



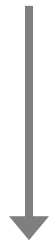
$[[B]] \rightarrow \perp$

$\lambda k. [[e]] (\lambda v. \text{snd } v k) : ([[B]] \rightarrow \perp) \rightarrow \perp$

$([[B]] \rightarrow \perp) \rightarrow \perp$

# CPS'ing second projection (dependently typed)

$\text{snd } e : B[\text{fst } e / x] \quad \text{where } e : \Sigma x : A. B$



$\llbracket B \rrbracket [\llbracket \text{fst } e \rrbracket / x] \rightarrow \perp$

$\lambda k. \llbracket e \rrbracket (\lambda v. \text{snd } v k) : (\llbracket B \rrbracket [\llbracket \text{fst } v \rrbracket / x] \rightarrow \perp) \rightarrow \perp$   
 $(\llbracket B \rrbracket [\text{fst } v / x] \rightarrow \perp) \rightarrow \perp$

# Intuition

$\text{snd } e : B[\text{fst } e / x]$     where  $e : \Sigma x : A. B$



unique if  $e$  is pure

$\lambda k. \llbracket e \rrbracket (\lambda v. \text{snd } v k) : (\llbracket B \rrbracket [\llbracket \text{fst } v \rrbracket / x] \rightarrow \perp) \rightarrow \perp$

$\text{fst } v = \text{fst } \text{val-of}(\llbracket e \rrbracket) = \text{fst } (\llbracket e \rrbracket \text{id}) = \llbracket \text{fst } e \rrbracket$

# Solution [Bowman, Cong, Rioux, Ahmed '17]

1. Polymorphic answer type  $e : A$



$\lambda\alpha. \lambda k. e' : \Pi\alpha. ([A] \rightarrow \alpha) \rightarrow \alpha$

$\lambda\alpha. \lambda k. \llbracket e \rrbracket \alpha (\lambda v. \text{snd } v \alpha k) : \Pi\alpha. ([B] [\llbracket \text{fst } e \rrbracket / x] \rightarrow \alpha) \rightarrow \alpha$

$\text{fst } v = \text{fst } \text{val-of}(\llbracket e \rrbracket) = \text{fst } (\llbracket e \rrbracket \text{ C id}) = \llbracket \text{fst } e \rrbracket$

where  $C = \Sigma x : [A]' . [B]'$

# Solution [Bowman, Cong, Rioux, Ahmed '17]

## 2. New typing rule

$$\frac{\Gamma \vdash e_1 : \forall \alpha. (A \rightarrow \alpha) \rightarrow \alpha \quad \Gamma, \mathbf{v = e_1 \ A \ id} \vdash e_2 : B}{\Gamma \vdash e_1 \ B \ (\lambda v. e_2) : B}$$

$\lambda \alpha. \lambda k. \llbracket e \rrbracket \alpha \ (\lambda v. \text{snd } v \ \alpha \ k) : \Pi \alpha. (\llbracket B \rrbracket [\llbracket \text{fst } e \rrbracket / x] \rightarrow \alpha) \rightarrow \alpha$

$\mathbf{fst \ v = \ fst \ val-of(\llbracket e \rrbracket) = \ fst \ (\llbracket e \rrbracket \ C \ id) = \llbracket \text{fst } e \rrbracket}$

where  $C = \Sigma x : \llbracket A \rrbracket'. \llbracket B \rrbracket'$

# Solution [Bowman, Cong, Rioux, Ahmed '17]

## 3. New equivalence rule

$$\frac{}{e \text{ B } k \equiv k \text{ (e A id)}}$$

$$\lambda\alpha.\lambda k. \llbracket e \rrbracket \alpha (\lambda v. \text{snd } v \alpha k) : \Pi\alpha. (\llbracket B \rrbracket [\llbracket \text{fst } e \rrbracket / x] \rightarrow \alpha) \rightarrow \alpha$$

$$\text{fst } v = \text{fst } \text{val-of}(\llbracket e \rrbracket) = \text{fst } (\llbracket e \rrbracket \text{ C id}) = \llbracket \text{fst } e \rrbracket$$

$$\text{where } C = \Sigma x : \llbracket A \rrbracket'. \llbracket B \rrbracket'$$



# Type preservation

If  $\Gamma \vdash e : A$  in the source,  
then  $[[\Gamma]] \vdash [[e]] : \Pi\alpha. ([[A]] \rightarrow \alpha) \rightarrow \alpha$  in the target.

Proof: By induction on the typing derivation.

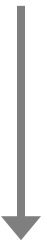
# Known challenge

Herbelin  
TLCA '05

We show that a minimal dependent type theory based on  $\Sigma$ -types and equality is degenerated in presence of computational classical logic.

# CPS of second projection (pure)

$\text{snd } e : B[\text{fst } e / x]$     where  $e : \Sigma x : A. B$



unique if e is pure

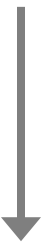
$\lambda\alpha.\lambda k. \llbracket e \rrbracket \alpha (\lambda v. \text{snd } v \alpha k) : \Pi\alpha. (\llbracket B \rrbracket [\llbracket \text{fst } e \rrbracket / x] \rightarrow \alpha) \rightarrow \alpha$

$\text{fst } v = \text{fst } \text{val-of}(\llbracket e \rrbracket) = \text{fst } (\llbracket e \rrbracket C \text{id}) = \llbracket \text{fst } e \rrbracket$

where  $C = \Sigma x : \llbracket A \rrbracket'. \llbracket B \rrbracket'$

# CPS of second projection (impure)

$\text{snd } e : B[\text{fst } e / x]$  where  $e : \Sigma x : A. B$



not unique if  $e$  is impure

(e.g.,  $e = S k. k \ 1 + k \ 2 \Rightarrow v = 1, 2$ )

$\lambda \alpha. \lambda k. \llbracket e \rrbracket \alpha (\lambda v. \text{snd } v \ \alpha \ k) : \Pi \alpha. (\llbracket B \rrbracket [\llbracket \text{fst } e \rrbracket / x] \rightarrow \alpha) \rightarrow \alpha$

$\text{fst } v \neq \text{fst } \text{val-of}(\llbracket e \rrbracket) = \text{fst } (\llbracket e \rrbracket \ C \ \text{id}) = \llbracket \text{fst } e \rrbracket$

where  $C = \Sigma x : \llbracket A \rrbracket'. \llbracket B \rrbracket'$

# Solution: purity restriction [Cong & Asai '18]

Types may depend only on pure terms

✓ Vec Nat 3

✗ Vec Nat Sk. 3

# Solution: purity restriction [Cong & Asai '18]

pure

$\Gamma \vdash_p e : A$

impure (change context  
type from  $\alpha$  to  $\beta$ )

$\Gamma \vdash_{\alpha, \beta} e : A$

# Solution: purity restriction [Cong & Asai '18]

e pure  $\Rightarrow$   
may appear in types

$$\frac{\Gamma \vdash_p e : \Sigma x : A. B}{\Gamma \vdash_p \text{snd } e : B[\text{fst } e / x]}$$

e impure  $\Rightarrow$   
may not appear in types

$$\frac{\Gamma \vdash_{\alpha, \beta} e : A \times B}{\Gamma \vdash_{\alpha, \beta} \text{snd } e : B}$$

# Type preservation

- If  $\Gamma \vdash_p e : A$  in the source,  
then  $[[\Gamma]] \vdash [[e]] : \Pi\alpha. ([[A]] \rightarrow \alpha) \rightarrow \alpha$  in the target.
- If  $\Gamma \vdash_{\alpha,\beta} e : A$  in the source,  
then  $[[\Gamma]] \vdash [[e]] : ([[A]] \rightarrow [[\alpha]]) \rightarrow [[\beta]]$  in the target.

Proof: By induction on the typing derivation.



# 2019: Joining PRG at Tokyo Tech

## Programming Research Group

Department of Mathematical and Computing Science,  
Tokyo Institute of Technology



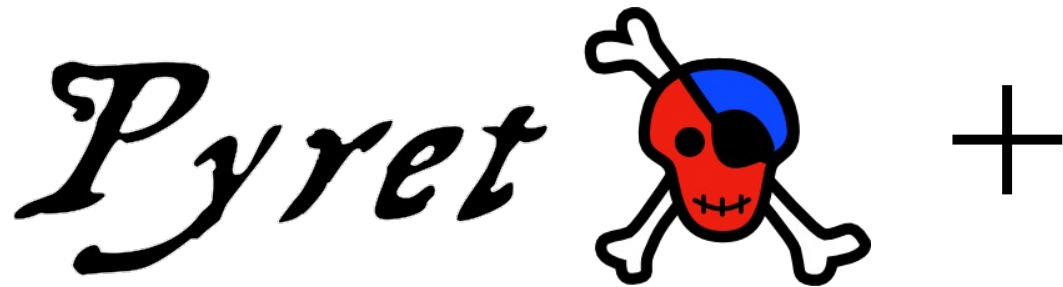
# A visitor from Linköping

**li.u** LINKÖPING  
UNIVERSITY



Filip Strömbäck

# A language for learning continuations



- control constructs
- stepping evaluator
- continuation visualizer

# Example of Pyret program

```
fun sum(l):  
  doc: "compute the sum of numbers in list l"  
  cases (List) l:  
    | empty => 0  
    | link(x, xs) => x + sum(xs)  
  end  
end
```

# Pyret online editor

The screenshot shows the Pyret online editor interface. At the top, there is a navigation bar with a skull and crossbones icon, a 'View' dropdown, and a 'Connect to Google Drive' button. To the right of the navigation bar are two buttons: a blue 'Run' button and a grey 'Stop' button.

The code editor on the left contains the following code:

```
1 use context essentials2021
2 fun sum(l):
3   doc: "compute the sum of numbers in list l"
4   cases (List) l:
5     | empty => 0
6     | link(x, xs) => x + sum(xs)
7   end
8 end
9
10 sum(10)
11
```

The console on the right displays an error message:

The **List** annotation  
was not satisfied by the value  
10  
(Show program evaluation trace...)

The error message is highlighted with a blue border. A dashed box highlights the code snippet that caused the error: `4 cases (List) l:`. A link `definitions:///3:9-3:13` is visible next to the error message.

# Extension 1: grab/delimit


```
delimit: reset  
  2 * grab(k): k(k(5)) end  
end # 20 shift
```

Cf: lambda abstraction

```
lam(x): 2 * x end
```

# Extension 2: send/run

**run:**

2 \* **send**(5) 

with handler(x, k): k(k(x))

end # 20

# Extension 3: effect handlers

```
effect Exn:  
  | raise(v)  
end
```

```
handler (Exn) abort:  
  | raise(v), k => v  
end
```

```
handle: 2 * raise(5) with abort # 5
```



# Stepping evaluator

Stepping	Previous	Next
<pre>delimit:   2 * grab(k):     k(k(5))   end end</pre>	<p>1/7</p> <p>⇒</p>	<pre>{(\$):   delimit:     2 * \$   end }({(\$):   delimit:     2 * \$   end }(5))</pre>

# A course on continuations (June – July)

1. Introduction
2. FP basics
3. CPS
4. grab/delimit
5. send/run
6. Effect handlers
7. Mixing effects
8. Linguistic applications
9. Untyped  $\lambda$  calculus
10. Typed  $\lambda$  calculus
11.  $\lambda$  + grab/delimit
12.  $\lambda$  + send/run
13.  $\lambda$  + effect handlers
14. Wrap-up

# Continuation of my research

- Applications:  
Reactive programming, probabilistic programming
- Theory:  
Compilation of non-algebraic effects
- Learning:  
Students' difficulties, learning support

# Wrapping up

Continuations are interesting to study!

Thanks to...

