The 36th Annual ACM SIGPLAN – SIGACT Symposium
Principles of Programming Languages
Savannah, Georgia, USA
9AM, Thursday, January 22, 2009

Wild Control Operators

Chris Barker, *NYU Linguistics*

`http://homepages.nyu.edu/~cb125`

Slides available here: `tinyurl.com/8trc7t`

Continuations provide insight into natural language meanings. In the other direction, there are natural language constructions—operators in the wild, so to speak—that require new, more expressive control mechanisms. I analyze the English word *same* by proposing **two-hole continuations**.

Prelude

Programming languages research:

- Observe what people (programmers) want to say
- Figure out how best to let them say it

Natural language semantics research:

- Observe what people say
- Figure out what they mean when they say it

Let the characterization be **compositional**: each syntactic constituent makes the same contribution no matter where it occurs (where "contribution" can include (side) effects)

[$Same$ and $different$] appear to be totally resistant to a strictly compositional semantic analysis... —Stump (1982:2)

Plan

Part I: Natural language meaning as programs

Part II: Continuations for natural language

- Warm-up case study, focus particles:
  (1) a. John only drinks PERRIER.
  b. John only DRINKS Perrier.
- Analysis: delimited (i.e., composable) continuations, in particular, Sitaram's `fcontrol` and `run`

Part III: Main case study: *same*

(2) Anna and Bill read the same book.
= There is a book x such that
Anna read x and Bill read x.

Part IV: Analysis: two-hole continuations

Part V: NL$_{CL}$: a substructural logic for two-hole cont'ns

Part VI: Other applications

Part I: Natural language meaning as programs

Natural language has inspired formal langages

	Natural lg. phenomenon	Formal language analog
functions	John left.	Pj
variables	He left.	Px
binding	Everyone loves his mother.	$\forall x.\mathtt{loves}(\mathtt{mom}\,x)\,x$
control	If you’re hungry, eat.	`if (hungry) then (eat)`

Functions as first-class objects? [Shieber]

John loves his mother and Bill does too.

```
let does = \x -> loves (mom x) x
in and [does j, does b]
```

Behavioral analogies (to be refined)

	Natural lg.	Formal lg.
imperatives	Shut the door!	$\mathtt{print}(3)$
questions	Who left?	$\mathtt{read}(s)$
statements	He is John.	$x == 3$

Disanalogies: vagueness, ambiguity

Behavioral analogies refined (already): Side effects

(3)

	Expression	(Direct) Type	Side effect
a.	`3`	Int	[none]
b.	`print (3)`	Int	[outputs "3"]
c.	`s`	String	[none]
d.	`read (s)`	String	[reads a string from input]
e.	`3`	Int	[none]
f.	`x = 3`	Int	[binds x to 3]

Natural language side effects? [Potts]

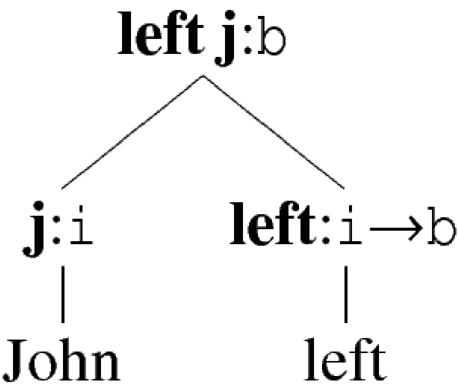
(4)

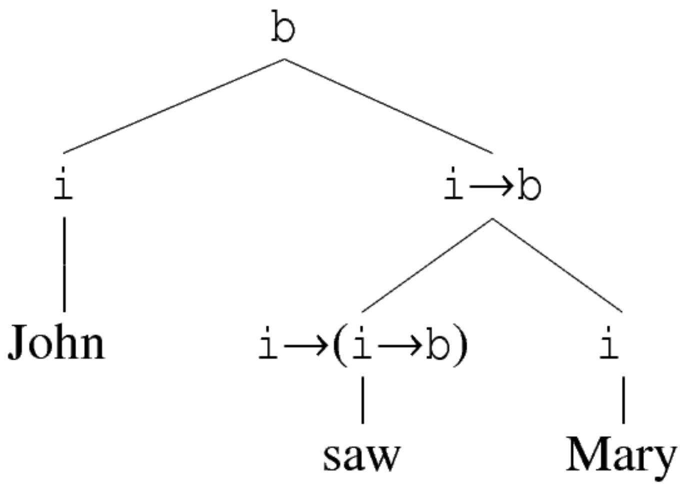
	Expression	Side effect
a.	John	[none]
b.	that idiot John	[classifies J. as an idiot]
c.	I took out the garbage.	[none]
d.	I took out the damn garbage.	[expresses attitude]
e.	We have a can opener.	[none]
f.	Suppose we have a can opener.	[creates new environment]

A simple direct basic fragment

(5)

Expression	Syntactic category (= type)	Semantic value
John	Ind	**j**
Mary	Ind	**m**
left	Ind→Bool	**left**
saw	Ind→(Ind→Bool)	**saw**





Part II: Continuations for natural language

Continunations in ...

Computer science:

- Reynolds, Plotkin, Strachey and Wadsworth, ...
- Felleisen, Danvy and Filinsky, ...: delimited (composable) continuations

Logic:

- Griffin, Parigot: computational content for classical proofs

Natural Language:

- Montague, Rooth, Hendriks, Groenendijk and Stokhof, ...
- Quantification (Barker 2001; 2002)
- Quantification, logically (de Groote 2001)
- WH-questions (WHo, WHat, WHen, WHow), (Shan 2002)
- Type-Logical Grammar (Barker & Shan, Moortgat & Bernardi)

Continuations are (pieces of) context

Expression Γ:

```
(f 2 (g 3 4)):Bool
```

Continuation $\Gamma[\,]$ of the subexpression `3` relative to Γ:

```
(f 2 (g [ ] 4)):Int --> Bool
```

$\Gamma[\,]$ is what I will call a **one-hole continuation**.

Programming challenge: provide uniform access to continuations

Three strategies:

- reduction rules (operationally);
- CPS xform (denotationally); and
- logically (see below...)

Simple continuations, operationally: a shift operator, †

The semantic argument of a quantifier is its continuation:

$$[\![John\ saw\ everyone]\!] = \mathbf{everyone}(\lambda x.\mathbf{saw}\, x\, \mathbf{j})$$

$$\Gamma[\dagger M] \rhd_{\dagger} M(\lambda m \Gamma[m])$$

$[\![everyone]\!]$ = $\dagger(\lambda P.\forall x.Px)$

John saw everyone. $([\![saw]\!][\![everyone]\!])[\![John]\!]$

$$= (\mathbf{saw}(\dagger(\lambda P.\forall x.Px)))\mathbf{j}$$

$$= (\lambda P.\forall x.Px)(\lambda x.\mathbf{saw}\, x\, \mathbf{j})$$

$$= \forall x.\mathbf{saw}\, x\, \mathbf{j}$$

Continuations, a simple CPS transform

$\overline{\alpha} = \lambda\kappa.\kappa\alpha$ (for any constant α)
$\overline{MN} = \lambda\kappa.\overline{M}(\lambda m.\overline{N}(\lambda n.\kappa(mn)))$

	Direct type	**CPS'd type**
In general here,	X	(X$\rightarrow\sigma$)$\rightarrow\sigma$
Specifically,	A$\rightarrow$B	((A$\rightarrow$B)$\rightarrow\sigma$)$\rightarrow\sigma$

$$\begin{aligned}
[\![\mathit{John\ left}]\!] &= \lambda\kappa.[\lambda\kappa.\kappa\mathbf{left}](\lambda f.[\lambda\kappa.\kappa\mathbf{j}](\lambda x.\kappa(fx))) \\
&\rightsquigarrow \kappa.\kappa(\mathbf{left}\,\mathbf{j}) \\
[\![\mathit{Everyone\ left}]\!] &= \lambda\kappa.[\lambda\kappa.\kappa\mathbf{left}](\lambda f.[\lambda\kappa.\forall x.\kappa x](\lambda x.\kappa(fx))) \\
&\rightsquigarrow \lambda\kappa.\forall x.\kappa(\mathbf{left}\,x),
\end{aligned}$$

$$[\![\mathit{John\ saw\ everyone}]\!] \rightsquigarrow \lambda\kappa.\forall x.\kappa(\mathbf{saw}\,x\,\mathbf{j})$$

Composable continuations; evaluation order

(6) Different CPS transforms result in different evaluation orders:

a. $\overline{MN} = \lambda\kappa.\overline{\mathrm{M}}(\lambda m.\overline{\mathrm{N}}(\lambda n.\kappa(mn)))$ Left-to-right (as before)

b. $\overline{MN} = \lambda\kappa.\overline{\mathrm{N}}(\lambda n.\overline{\mathrm{M}}(\lambda m.\kappa(mn)))$ Right-to-left

(7) a. Someone saw everyone.

b. $\exists x \forall y.\mathbf{saw}\,x\,y$ left-to-right

c. $\forall y \exists x.\mathbf{saw}\,x\,y$ right-to-left

Focus/*only* as `fcontrol`/`run` (respectively) [Sitaram]

`fcontrol`: takes one argument x and throws $\langle x, \kappa \rangle$, where κ is the continuation of `(fcontrol x)` delimited by `run`

`run`: takes two arguments, an expression containing at least zero occurrences of `fcontrol`, and a handler routine

```
(+ 1 (run (* 2 (+ (fcontrol 3) 4))
          (lambda (x k) (k (k x)))))
```

```
= (+ 1 ((lambda (x k) (k (k x)))
        (3 (lambda y (* 2 (+ y 4))))))
= (+ 1 ((lambda y (* 2 (+ y 4)))
        ((lambda y (* 2 (+ y 4)))
         3)))
= (+ 1 ((lambda y (* 2 (+ y 4)))
        (* 2 (+ 3 4))))
= 37
```

(8)a. John only drinks PERRIER.

b. John $(\texttt{run}\ (\mathbf{drinks}(\texttt{fcontrol}\,\mathbf{Perrier}))$
$(\lambda x\kappa y.(\mathbf{and}(\kappa xy)(\forall z(\mathbf{or}(\mathbf{equal}\,x\,z)(\mathbf{not}(\kappa zy)))))))$

c. $\kappa = \mathbf{drinks}$

d. $\mathbf{(and\ (drinks\ \ Perrier\ \ j)}$
$\mathbf{(}\forall\ z\ \mathbf{(or\ (equal\ \ Perrier\ \ }z\mathbf{)}$
$\mathbf{(not\ (drinks\ \ }z\ \mathbf{\ j)))))}$

e. John drinks Perrier.

f. There is nothing else that John drinks other than Perrier.

---

(9) a. John only DRINKS Perrier.

b. John $(\texttt{run}\ ((\texttt{fcontrol}\,\mathbf{drinks})\mathbf{Perrier})$
$(\lambda x\kappa y.(\mathbf{and}(\kappa xy)(\forall z(\mathbf{or}(\mathbf{equal}\,x\,z)(\mathbf{not}(\kappa zy)))))))$

c. $\kappa = \lambda R.R\,\mathbf{Perrier}$

d. $\mathbf{(and\ (drinks\ \ Perrier\ \ j)}$
$\mathbf{(}\forall\ z\ \mathbf{(or\ (equal\ \ drinks\ \ }z\mathbf{)}$
$\mathbf{(not\ (}z\ \mathbf{\ Perrier\ \ j)))))}$

e. John drinks Perrier.

f. There is nothing else that John does with Perrier other than drink it.

One slightly more complex example

(10) Mary only tried to dance with F(JOHN).

$$\textbf{(and (tried-to-dance-with}\,\mathbf{j}\,\mathbf{m})\;(\forall\, z\;(\textbf{or (equal}\,\mathbf{j}\,z)\;(\textbf{not (tried-to-dance-with}\;z\;\mathbf{m})))))$$

(11) Mary only tried to F(DANCE) with John.

$$\textbf{(and (tried-to-dance-with}\,\mathbf{j}\,\mathbf{m})\;(\forall\, z\;(\textbf{or (equal}\,\textbf{dance}\;z)\;(\textbf{not (tried-to-}z\textbf{-with j m})))))$$

Assessment:

- `fcontrol`/`run` proposed for purely computational reasons
- Nevertheless, fits a natural language construction beautifully
- In some sense, then, `fcontrol`/`run` is a “natural” operator

Part III: Main case study: *same*

External *same* vs. sentence-internal *same*

(12) [Ivan holds a copy of $Emma$]
Anna and Bill read the same book.

(13) a. Anna and Bill read the held-by-Ivan book. **External**
b. context tells us that same = held-by-Ivan

(14) a. Anna and Bill read the read-by-them book. **Internal**
b. There is some book x such that Gina read x and Bill read

External: pragmatic, deictic, anaphoric
Internal: semantic, quantificational, mediated by the grammar

Carlson: For the internal reading,

“the sentence, in some way or other, provides its own context.”

context = continuation

Red herring: types versus tokens [Nunberg, Lasersohn]

(15) a. I drive a Ford Falcon and Enzo drives the same car.
b. #I was driving south on 280 in my Ford Falcon when I smashed into the same car.

Many other related but distinct constructions involving *same*.

The only one I am dealing with today is an internal reading for *the same N*, where N is some noun (e.g., *books*), without relative clause or *as* phrase.

(16) a. Anna and Bill read the same book.
b. That's the same book that Ivan read. Explicit property
c. This is the same (size) as that. Explicit standard
d. Those two are the same. Predicative
e. Etc...

CHILDES database (CMU): Providence data set, Alex, file ale51.cha
Conversation in Massachusetts 28 Apr 2004 (Alex 3 years 4 mos. old)
about beans: kidney beans, black-eyed peas, great northern pinto beans

CHI: yy look at this one.
ˈhɛoˈlʊkæˈdɪsˈwʌn
MOT: which one?
CHI: yy this one.
nəˈdɪsˈwʌn
MOT: oh that's differen(t).
CHI: that's different.
ˈdæsˈdɪfwɪn
CHI: xx look at this one.
ˈlʊkæˈdɪsˈwʌn
MOT: they're all differen(t) Alex.
where did that xx +//.
oh there it is see that
really big gigantic one ?
tha(t) one's huge.

CHI: oh [x 4] I got something.
ˈo ˈɑˈgɑˈsʌmti
MOT: oh I foun(d) another one.
another one.
CHI: it's yy the same size.
ˈɪsˈɑːt əˈseɪmˈsaɪz
MOT: it's which one.
CHI: the same size.
ðəˈseɪmˈsaɪz
MOT: they match?
CHI: yes they same same size like that.
ˈjɛs ˈðeɪˈseɪmˈseɪmˈsaɪz ˈaɪkˈdæt
CHI: yy yy same size.
ˈdʒʌsaɪkˈʃeɪmˈsaɪːz
MOT: mmmm.
CHI: they yy the same size.
ˈðeɪˈdɑ ðəˈseɪmˈsaɪːz

Quotations from some older programmers:

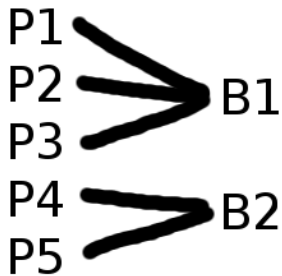
Taken from random POPL papers:

- Two variables are considered equal if we encounter them in **the same syntactic position** in the two function bodies.
- Take maximal set of polynomials with **same zeroes**
- Any GC may be lifted to a GI by identifying in an equivalence class those values of the abstract domain with **the same concretization**.
- When none of the guards is enabled, the next state Y will have **the same value** as the current state X.
- We analyze the bodies of functions of **the same name** and match their abstract syntax trees structurally.
- The remaining two judgements in the premise insist, **respectively**, that the override preserve the type of the object being updated, and that the new body provided for l j have the same type B j as the original body.

An idea worth briefly considering:

(17) Maybe *the same book* is secretly an indefinite.
That is, it means roughly *There is some book x such that..*

a. "That's **the wrong answer**!"
b. How many people read a book?
c. How many people read the same book?



(18) a. Anna and Bill have never lived in the same city.
b. $\neq$ There is a city x such that Anna and Bill have never lived in x. (Choose x = Perth)

Theorem [Keenan]: *same* is provably not equivalent to a simple indefinite.

Operational analysis

Intuition: *same* needs access to a distant expression's cont'n.

Normal (delimited) continuation (from before):

$$\Gamma[\dagger M] \rhd_\dagger M(\lambda m \Gamma[m])$$

New: two-hole continuation:

$$M(\lambda m \Gamma[m][\ddagger N]) \rhd_\ddagger M(N(\lambda nm.\Gamma[m][n]))$$

(19) John (served †everyone)
$\rhd_\dagger$ everyone($\lambda m.$ John (served m))

(20) ‡(the same waiter) (served †everyone)
$\rhd_\dagger$everyone($\lambda m.$ ‡(the same waiter) (served m))
$\rhd_\ddagger$everyone((the same waiter) $\lambda nm.\ n$ (served m))

Semantics for *same* (slide 1 of 2)

Step 1 of 4: Nouns

```
type Noun = [Int] -> Bool
primes :: Noun
primes = all (\i -> and [1<i, null [x|x <- [2..i-1]
                                        , mod i x == 0]])
primes [2,3,5,7] = True
primes [2,3,5,7,9] = False
```

Step 2 of 4: Verbs

```
follow :: [Int] -> [Int] -> Bool
follow is js = and [i>j|i<-is,j<-js]   -- cross product
[7,9] ‘follow‘  [3,4] = True
[7,9] ‘follow‘ [3,8] = False
```

Note: Haskell is a VSO language, like Chamorro or Irish

Semantics for *same* (slide 2 of 2)

Step 3 of 4: impure conjunction

```
andNL :: [Int] -> [Int] -> ([Int] -> Bool) -> Bool
andNL is js f = f (is ++ js)
([7] `andNL` [9]) (\x -> x `follow` [3])
```

Step 3 of 4: same

```
theSame :: Noun -> ([Int] -> [Int] -> Bool) -> [Int] -> Bool
  --                -------------------------    -------------
  --                two-hole continuation        one-hole cont.

theSame n kk js =
  1 == length (group [[i|i<-[1..28], n [i], kk [j] [i]]|j<-js])
```

`group` splits its list argument into a list of lists of equal, adjacent elements: `group` "Mississippi" == ["M","i","ss","i","ss","i","pp","i"]

Examples

(21) a. 8 and 9 follow the same primes.

b. † (8 and 9) (follow ‡ (the same primes))

c. $\triangleright_{\dagger}$ (8 and 9)$(\lambda n.\mathbf{follow}(\ddagger(\mathbf{theSame\,primes}))\,n)$

d. $\triangleright_{\ddagger}$ (8 and 9) (($\mathbf{theSame\ \ primes}$) ($\lambda m n.\mathbf{follow}\,m\,n$))

e. `([8] ‘andNL‘ [9]) (theSame primes follow)`

f. `1 == length (group [[2,3,5,7], [2,3,5,7]])`

g. `True`

(22) a. 8 and 12 follow the same primes.

b. `1 == length (group [[2,3,5,7], [2,3,5,7,11]])`

c. `False`

(23) a. The same primes follow 8 and 9.

b. ‡ (the same primes) (follow † (8 and 9))

c. $\triangleright_{\dagger}$ (8 and 9)($\lambda n.(\mathbf{follow}\,n)(\ddagger(\mathbf{theSame\,primes})$))

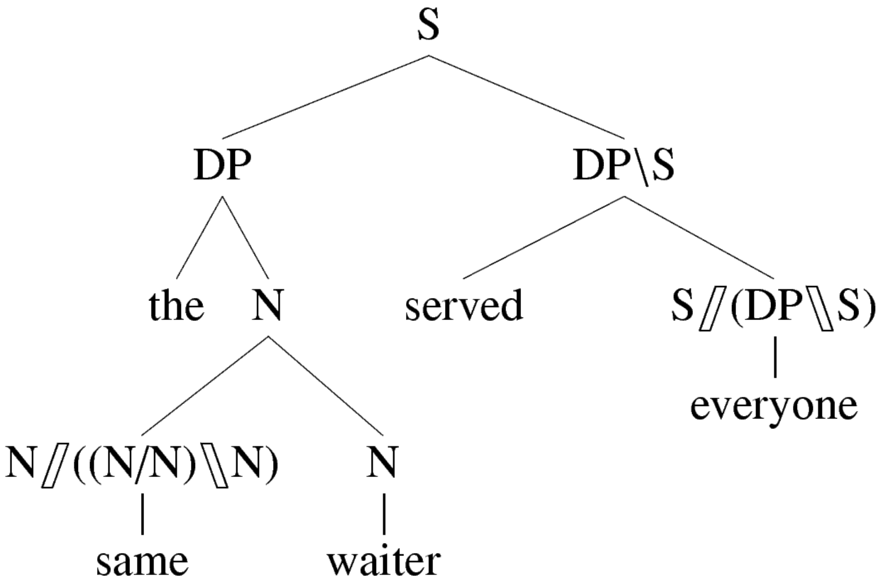
d. $\triangleright_{\ddagger}$ (8 and 9) (($\mathbf{theSame\ \ primes}$) ($\lambda m n.\mathbf{follow}\,n\,m$))

e. `([8] ‘andNL‘ [9]) (theSame primes (flip follow))`

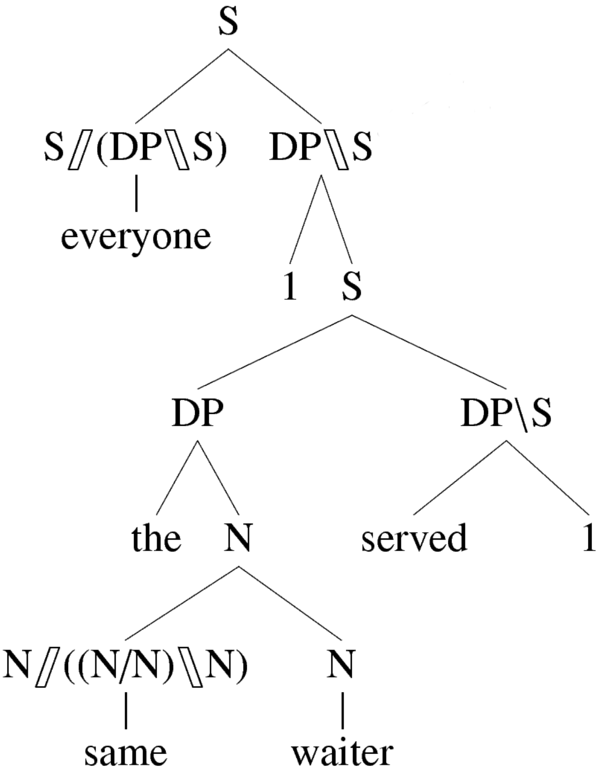
f. `1 == length (group [[11,13,17,19,23],[11,13,17,19,23]])`

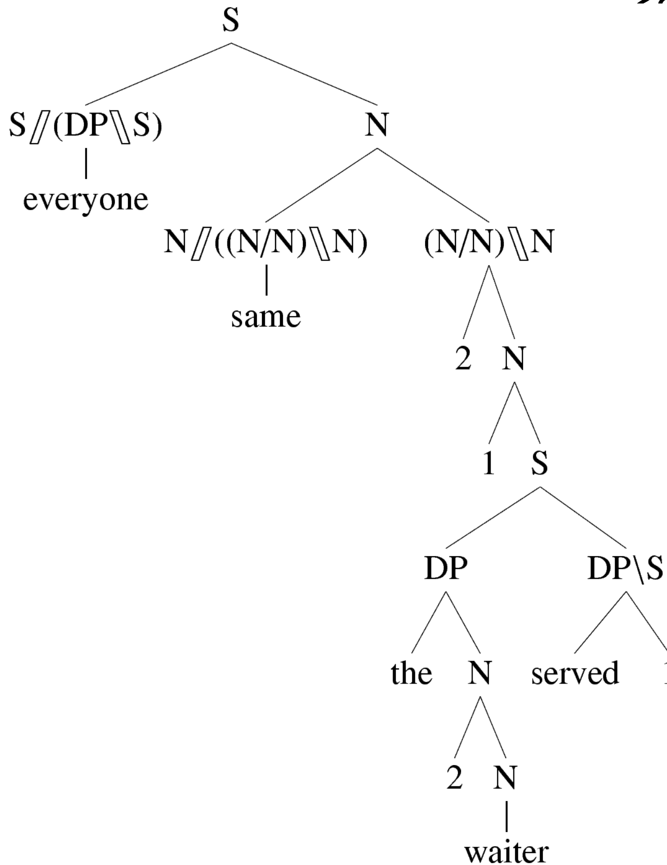
g. `True`

(24) The same waiter served everyone. [Stump; Heim]



Parasitic scope





```
Date: Thu, 22 Jan 2009 02:27:14 +0000
From: oleg@okmij.org
Subject: The same waiter served everybody

Hello!
        Here is the computed denotation for the example where same i
evaluated first: ‘The same waiter served everyone’

  .<(Everyone
    (fun x_3 ->
      ((((* cross-stage persistent value (as id: same’) *))
         (fun f_2 -> fun x_1 -> (Served ((The (f_2 (Waiter))), x_1))

The denotation was computed by the following term (once again, the
explicit ‘let’ ensured the left-to-right evaluation order):

let t5 = top (fun () ->
 let sw = etasame (fun x -> served (the (same waiter)) x)
 in sw (everyone ()));;
```

Part V: NL$_{CL}$: a substructural logic for two-hole continuations

$\mathbf{NL}_{CL}$: a two-tensor, non-associative Lambek grammar

$$\frac{\Gamma \vdash A \qquad \Sigma[B] \vdash C}{\Sigma[(\Gamma \bullet A\backslash B)] \vdash C}\ \backslash L \qquad \frac{A \bullet \Gamma \vdash C}{\Gamma \vdash A\backslash C}\ \backslash R$$

$$\frac{\Gamma \vdash A \qquad \Sigma[B] \vdash C}{\Sigma[(B/A \bullet \Gamma)] \vdash C}\ /L \qquad \frac{\Gamma \bullet B \vdash C}{\Gamma \vdash C/B}\ /R$$

$$\frac{\Gamma \vdash A \qquad \Sigma[B] \vdash C}{\Sigma[(\Gamma \circ A\backslash\!\!\backslash B)] \vdash C}\ \backslash\!\!\backslash L \qquad \frac{A \circ \Gamma \vdash C}{\Gamma \vdash A\backslash\!\!\backslash C}\ \backslash\!\!\backslash R$$

$$\frac{\Gamma \vdash A \qquad \Sigma[B] \vdash C}{\Sigma[(B/\!\!/A \circ \Gamma)] \vdash C}\ /\!\!/L \qquad \frac{\Gamma \circ B \vdash C}{\Gamma \vdash C/\!\!/B}\ /\!\!/R$$

$$\frac{\Sigma[\Gamma[p]] \vdash A}{\Sigma[p \circ \lambda x.\Gamma[x]] \vdash A}\ \lambda$$

No weakening, no contraction, no interchange; ‘p’ for ‘plug’

Details about $\Gamma\lceil p \rceil$ structures

As usual with λ, we pay for conceptual simplicity with some definitional complexity.

$$\Gamma\lceil p \rceil ::= \quad p \quad | \quad \lambda y.\, \Gamma\lceil p \rceil \quad | \quad q \bullet \Gamma\lceil p \rceil \quad | \quad \Gamma\lceil p \rceil \bullet q$$

This λ "abstracts" only over structures built from $\bullet$ and λ.

Allowed: $\dfrac{A}{A \circ \lambda x.x}$ $\quad \dfrac{A \bullet B}{A \circ \lambda x.(x \bullet B)}$ $\quad \dfrac{\lambda x.(x \bullet B)}{B \circ \lambda y \lambda x.(x \bullet y)}$

Disallowed: $\dfrac{A \circ B}{A \circ \lambda x.(x \circ B)}$ $\quad \dfrac{\lambda x.(x \bullet B)}{B \circ \lambda x \lambda y.(x \bullet y)}$

Crucially linear: x fresh (distinct from every other symbol in Γ).

Where are the continuations? [Shan]

$A \mathbin{\backslash\!\!\backslash} B$: one-hole continuation accepting an argument of type A and returning a result of type B.

Example derivation of *John saw everyone*
everyone: $\mathsf{S} /\!\!/ (\mathsf{DP} \mathbin{\backslash\!\!\backslash} \mathsf{S})$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\mathsf{DP} \bullet ((\mathsf{DP}\backslash \mathsf{S})/\mathsf{DP} \bullet \mathsf{DP}) \vdash \mathsf{S}}}{\textit{John} \bullet (\textit{saw} \bullet \mathsf{DP}) \vdash \mathsf{S}}\,\mathsf{LEX}}{\mathsf{DP} \circ \lambda x(\textit{John} \bullet (\textit{saw} \bullet x)) \vdash \mathsf{S}}\,\lambda}{\lambda x(\textit{John} \bullet (\textit{saw} \bullet x)) \vdash \mathsf{DP} \mathbin{\backslash\!\!\backslash} \mathsf{S}}\,\backslash\!\!\backslash R \quad \mathsf{S} \vdash \mathsf{S}}{\dfrac{\mathsf{S} /\!\!/ (\mathsf{DP} \mathbin{\backslash\!\!\backslash} \mathsf{S}) \circ \lambda x(\textit{John} \bullet (\textit{saw} \bullet x)) \vdash \mathsf{S}}{\textit{John} \bullet (\textit{saw} \bullet \mathsf{S} /\!\!/ (\mathsf{DP} \mathbin{\backslash\!\!\backslash} \mathsf{S})) \vdash \mathsf{S}}\,\lambda}\,/\!\!/ L$$

$$\mathbf{everyone}(\lambda x.\mathbf{saw}\, x\, \mathbf{j})$$

Ok, where are the two-hole continuations?

$\color{red}{A\backslash\!\!\backslash(}\color{blue}{B\backslash\!\!\backslash C}\color{red}{)}$: two-hole continuation accepting an argument of type A and returning a one-hole continuation of type $\color{blue}{B\backslash\!\!\backslash C}$.

Example: ***Everyone read the same book*:**

$$
\begin{prooftree}
\AxiomC{$\vdots$}
\noLine
\UnaryInfC{$\mathrm{DP} \bullet (\mathit{read} \bullet (\mathit{the} \bullet (\mathrm{N/N} \bullet \mathit{book}))) \vdash \mathrm{S}$}
\RightLabel{λ}
\UnaryInfC{$\mathrm{DP} \circ \lambda x(x \bullet (\mathit{read} \bullet (\mathit{the} \bullet (\mathrm{N/N} \bullet \mathit{book})))) \vdash \mathrm{S}$}
\RightLabel{$\backslash\!\!\backslash R$}
\UnaryInfC{$\lambda x(x \bullet (\mathit{read} \bullet (\mathit{the} \bullet (\mathrm{N/N} \bullet \mathit{book})))) \vdash \mathrm{DP}\backslash\!\!\backslash\mathrm{S}$}
\RightLabel{λ}
\UnaryInfC{$\mathrm{N/N} \circ \lambda y\lambda x(x \bullet (\mathit{read} \bullet (\mathit{the} \bullet (y \bullet \mathit{book})))) \vdash \mathrm{DP}\backslash\!\!\backslash\mathrm{S}$}
\RightLabel{$\backslash\!\!\backslash R$}
\UnaryInfC{$\lambda y\lambda x(x \bullet (\mathit{read} \bullet (\mathit{the} \bullet (y \bullet \mathit{book})))) \vdash \color{red}{(\mathrm{N/N})\backslash\!\!\backslash(}\color{blue}{\mathrm{DP}\backslash\!\!\backslash\mathrm{S}}\color{red}{)}$}
\AxiomC{$\mathrm{DP}\backslash\!\!\backslash\mathrm{S} \vdash \mathrm{DP}\backslash\!\!\backslash\mathrm{S}$}
\RightLabel{$/\!\!/L$}
\BinaryInfC{$(\mathrm{DP}\backslash\!\!\backslash\mathrm{S})/\!\!/((\mathrm{N/N})\backslash\!\!\backslash(\mathrm{DP}\backslash\!\!\backslash\mathrm{S})) \circ \lambda y\lambda x(x \bullet (\mathit{read} \bullet (\mathit{the} \bullet (y \bullet \mathit{book})))) \vdash \mathrm{DP}\backslash\!\!\backslash\mathrm{S}$}
\RightLabel{λ}
\UnaryInfC{$\lambda x(x \bullet (\mathit{read} \bullet (\mathit{the} \bullet ((\mathrm{DP}\backslash\!\!\backslash\mathrm{S})/\!\!/((\mathrm{N/N})\backslash\!\!\backslash(\mathrm{DP}\backslash\!\!\backslash\mathrm{S})) \bullet \mathit{book})))) \vdash \mathrm{DP}\backslash\!\!\backslash\mathrm{S}$}
\AxiomC{$\mathrm{S} \vdash \mathrm{S}$}
\RightLabel{$/\!\!/L$}
\BinaryInfC{$S/\!\!/(\mathrm{DP}\backslash\!\!\backslash\mathrm{S}) \circ \lambda x(x \bullet (\mathit{read} \bullet (\mathit{the} \bullet ((\mathrm{DP}\backslash\!\!\backslash\mathrm{S})/\!\!/((\mathrm{N/N})\backslash\!\!\backslash(\mathrm{DP}\backslash\!\!\backslash\mathrm{S})) \bullet \mathit{book})))) \vdash \mathrm{S}$}
\RightLabel{λ}
\UnaryInfC{$S/\!\!/(\mathrm{DP}\backslash\!\!\backslash\mathrm{S}) \bullet (\mathit{read} \bullet (\mathit{the} \bullet ((\mathrm{DP}\backslash\!\!\backslash\mathrm{S})/\!\!/((\mathrm{N/N})\backslash\!\!\backslash(\mathrm{DP}\backslash\!\!\backslash\mathrm{S})) \bullet \mathit{book}))) \vdash \mathrm{S}$}
\RightLabel{LEX}
\UnaryInfC{$\mathit{everyone} \bullet (\mathit{read} \bullet (\mathit{the} \bullet (\mathit{same} \bullet \mathit{book}))) \vdash \mathrm{S}$}
\end{prooftree}
$$

$$\mathbf{everyone}(\mathbf{same}(\lambda f(\lambda y.\mathbf{read}(\mathbf{the}(f(\mathbf{book})))\,y)))$$

Part VI: Other applications

Respectively

(25) †(2 and 3) evenly divide ‡((4 and 6) respectively).
Wrong:

```
[(x,y) | x <- [2,3]
       , y <- [4,6]
       , x mod y == 0]
```

Right:

```
(map (lambda (x y) (zero? (modulo y x)))
     '(2 3)
     '(4 6))
```

Requires programmer to compute a two-hole continuation.

```
resp :: [Int] -> ([Int] -> [Int] -> Bool) -> [Int] -> Bool
resp [] _ [] = True
resp (a:as) kk (b:bs) = if kk [b] [a] then resp as kk bs else False
```

(26) a. 3 and 9 follow 2 and 8 respectively.
b. `([3] ‘andNL‘ [9]) (resp [2,8] follow) = True`

Example: binding under effects: †*Everyone said* ‡*he left*:

$$[\![\ddagger he]\!] = \lambda\kappa\lambda x.\kappa x x : (\mathrm{DP}\backslash\!\!\backslash \mathrm{S})/\!\!/(\mathrm{DP}\backslash\!\!\backslash(\mathrm{DP}\backslash\!\!\backslash \mathrm{S}))$$

$$(i \to i \to b) \to i \to b$$

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{
        \dfrac{
          \dfrac{
            \dfrac{
              \dfrac{
                \dfrac{
                  \dfrac{\mathrm{DP} \bullet (\textit{said} \bullet (\mathrm{DP} \bullet \textit{left})) \vdash \mathrm{S}}
                  {\mathrm{DP} \circ \lambda x(x \bullet (\textit{said} \bullet (\mathrm{DP} \bullet \textit{left}))) \vdash \mathrm{S}}\lambda
                }
                {\lambda x(x \bullet (\textit{said} \bullet (\mathrm{DP} \bullet \textit{left}))) \vdash \mathrm{DP}\backslash\!\!\backslash \mathrm{S}}\backslash\!\!\backslash R
              }
              {\mathrm{DP} \circ \lambda y\lambda x(x \bullet (\textit{said} \bullet (y \bullet \textit{left}))) \vdash \mathrm{DP}\backslash\!\!\backslash \mathrm{S}}\lambda
            }
            {\lambda y\lambda x(x \bullet (\textit{said} \bullet (y \bullet \textit{left}))) \vdash \mathrm{DP}\backslash\!\!\backslash(\mathrm{DP}\backslash\!\!\backslash \mathrm{S})}\backslash\!\!\backslash R
            \quad \mathrm{DP}\backslash\!\!\backslash \mathrm{S} \vdash \mathrm{DP}\backslash\!\!\backslash \mathrm{S}
          }
          {(\mathrm{DP}\backslash\!\!\backslash \mathrm{S})/\!\!/(\mathrm{DP}\backslash\!\!\backslash(\mathrm{DP}\backslash\!\!\backslash \mathrm{S})) \circ \lambda y\lambda x(x \bullet (\textit{said} \bullet (y \bullet \textit{left}))) \vdash \mathrm{DP}\backslash\!\!\backslash \mathrm{S}}/\!\!/L
        }
        {\textit{he} \circ \lambda y\lambda x(x \bullet (\textit{said} \bullet (y \bullet \textit{left}))) \vdash \mathrm{DP}\backslash\!\!\backslash \mathrm{S}}\mathrm{LEX}
      }
      {\lambda x(x \bullet (\textit{said} \bullet (\textit{he} \bullet \textit{left}))) \vdash \mathrm{DP}\backslash\!\!\backslash \mathrm{S}}\lambda
      \quad \mathrm{S} \vdash \mathrm{S}
    }
    {\mathrm{S}/\!\!/(\mathrm{DP}\backslash\!\!\backslash \mathrm{S}) \circ \lambda x(x \bullet (\textit{said} \bullet (\textit{he} \bullet \textit{left}))) \vdash \mathrm{S}}/\!\!/L
  }
  {\textit{everyone} \circ \lambda x(x \bullet (\textit{said} \bullet (\textit{he} \bullet \textit{left}))) \vdash \mathrm{S}}\mathrm{LEX}
}
{\textit{everyone} \bullet (\textit{said} \bullet (\textit{he} \bullet \textit{left})) \vdash \mathrm{S}}\lambda
$$

$$\mathbf{everyone}((\lambda\kappa\lambda x.\kappa x x)(\lambda y\lambda x.\mathbf{said}(\mathbf{left}\,x)\,y)) = \mathbf{eo}(\lambda z.\mathbf{said}(\mathbf{lft}\,z)\,z)$$

cf. Morrill, Fadda & Valentín 2007:52

Wrong: `let x = everyone in Someone said x thinks x is intelligent`

What else?

(27) Polymorphic *same*:
John read and reviewed the same book.

(28) distinct, separate, similar [Carlson]
identical, unrelated, mutually incompatible, opposite

(29) The ‡average American owns †2.1 cars. [Stanley & Kennedy]

(30) “Resumptive” uses [Keenan]:
The same people ordered the same dishes.

What to remember about this talk

- First compositional analysis ever of a basic vocabulary item.
- The solution involves continuations.
- Yea, theory of programming lgs!
- But not ordinary (one-hole) continuations.
- Two-hole continuations provide new expressivity.
- They allow an expression to control the context within which another effectful expression will be evaluated.

Thanks: Benjamin Pierce, Oleg Kiselyov, Chung-chieh Shan

These slides: tinyurl.com/8trc7t

Barker, Chris. 2004. Continuations in natural language. In Hayo Thielecke (ed). *Proceedings of the Fourth ACM SIGPLAN Continuations Workshop (CW'04)*. Technical Report CSR-04-1, School of Computer Science, University of Birmingham, B15 2TT, United Kingdom. 1–11.

Barker, Chris. 2008. Parasitic Scope. *Linguistics and Philosophy*. **30.4**: 407–444.

END