The Duality of Construction

Paul Downen Zena M. Ariola

University of Oregon

April 9, 2014

The sequent calculus

Sequent calculus vs. Natural deduction

- Natural deduction tells us about pure functional programming
- Sequent calculus tells us about programming with duality
  - Flow of Information: producers are dual to consumers
  - Evaluation: Call-by-value is dual to call-by-name
  - Construction: data structures are dual to co-data (abstract objects with procedural interface)

Previous Work

- Curien and Herbelin (2000)
- Wadler (2003, 2005)
- Zeilberger (2008, 2009)
- Munch-Maccagnoni (2009) and Curien (2010)

Sequent calculus: a symmetric language

Commands c

$$\langle v \| e \rangle$$

Producers v	Consumers e (contexts)
Function abstraction $\lambda x.v$	Function call (call stack) $v \cdot e$
Input variable x	Output variable (co-variable) α
Output abstraction $\mu\alpha.c$	Input abstraction (let binding) $\tilde{\mu}x.c$
...	...

Flow of information

$$\xRightarrow{\text{flow of information}}$$

$$\langle v \| \tilde{\mu}x.c \rangle \quad \longrightarrow \quad c\,\{v/x\}$$

$$\xRightarrow[\text{productive info}]{}$$

Flow of information

$$\xrightarrow{\text{flow of information}}$$

$$c\,\{e/\alpha\} \quad \longleftarrow \quad \langle \mu\alpha.c \| e \rangle$$

$$\xleftarrow[\text{consumptive info}]{}$$

Not so fast. . .

Fundamental dilemma of classical computation

$$\begin{tikzcd}[column sep=small]
& \langle \mu\alpha.c_1 \| \tilde{\mu}x.c_2 \rangle \arrow[dl] \arrow[dr] & \\
c_1\left\{(\tilde{\mu}x.c_2)/\alpha\right\} & & c_2\left\{(\mu\alpha.c_1)/x\right\}
\end{tikzcd}$$

Fundamental dilemma of classical computation

$$\begin{tikzcd}
 & \langle \mu_.c_1 \| \tilde{\mu}_.c_2 \rangle \arrow[dl] \arrow[dr] & \\
c_1 & & c_2
\end{tikzcd}$$

Impact of strategy on substitution

- Call-by-value: Refined notion of "value"
  - Subset of producers (terms)
  - Variables stand in for values
- Call-by-name: Refined notion of "strict context" ("co-value")
  - Subset of consumers (co-terms)
  - Co-variables stand in for co-values

Parameterizing by the strategy

- Single calculus uses unspecified "values" and "co-values"
- Only substitute (co-)values for (co-)variables
- Strategy = definition of (co-)values
- Impact of strategy isolated to the two parameterized rules for substitution

Parameterizing by the strategy

$$
\begin{aligned}
&(\mu_E) && \langle \mu\alpha.c \| E \rangle = c\,\{E/\alpha\} \\
&(\tilde{\mu}_V) && \langle V \| \tilde{\mu}x.c \rangle = c\,\{V/x\} \\
\\
&(\eta_\mu) && \mu\alpha.\langle v \| \alpha \rangle = v \\
&(\eta_{\tilde{\mu}}) && \tilde{\mu}x.\langle x \| e \rangle = e
\end{aligned}
$$

Some strategies (and their dual)

Call-by-value:

- Variables are values
- Every consumer is a co-value

is dual to. . .

Call-by-name:

- Every producer is a value
- Co-variables are co-values

Some strategies (and their dual)

Call-by-value:

$$V \in \mathit{Value} ::= x \qquad\qquad E \in \mathit{CoValue} ::= e$$

is dual to...

Call-by-name:

$$V \in \mathit{Value} ::= v \qquad\qquad E \in \mathit{CoValue} ::= \alpha$$

Some strategies (and their dual)

Lazy call-by-value (aka call-by-need):

- Values same as call-by-value
- Co-values may contain delayed let-bindings

is dual to...

Lazy call-by-name:

- Values may contain delayed co-let-bindings (callcc)
- Co-values same as call-by-name

Some strategies (and their dual)

Lazy call-by-value (aka call-by-need):

$$
\begin{aligned}
V \in \mathit{Value} &::= x \\
E \in \mathit{CoValue} &::= \alpha \mid \tilde{\mu}x.\langle v \| \tilde{\mu}y.\langle x \| E \rangle \rangle
\end{aligned}
$$

is dual to...

Lazy call-by-name:

$$
\begin{aligned}
V \in \mathit{Value} &::= x \mid \mu\alpha.\langle \mu\beta.\langle V \| \alpha \rangle \| e \rangle \\
E \in \mathit{CoValue} &::= \alpha
\end{aligned}
$$

Two dual approaches to organize information

Data types

- Defined by rules of creation (constructors)
- Producer: fixed shapes given by constructors
- Consumer: case analysis on constructions
- Like ADTs in ML and Haskell

Declaring sums as data

(G)ADT:

$$\begin{aligned}&\textbf{data}\ \mathrm{Either}\ a\ b\ \textbf{where}\\ &\qquad \mathrm{Left} :: a \to \mathrm{Either}\ a\ b\\ &\qquad \mathrm{Right} :: b \to \mathrm{Either}\ a\ b\end{aligned}$$

Sequent:

$$\begin{aligned}&\textbf{data}\ a \oplus b\ \textbf{where}\\ &\qquad \mathrm{Left} : a \vdash a \oplus b|\\ &\qquad \mathrm{Right} : b \vdash a \oplus b|\end{aligned}$$

Declaring sums as data

- Producer: two constructors (left or right)

$$\mathrm{Left}(v_1) \qquad\qquad \mathrm{Right}(v_2)$$

- Consumer: consider shape of input
  - "If I'm given Left, do this"
  - "If I'm given Right, do that"

$$\tilde{\mu}[\mathrm{Left}(x).c_1 \,|\, \mathrm{Right}(y).c_2]$$

Co-data types

- Defined by rules of observation (messages)
- Consumer: fixed shapes given by observations
- Producer: case analysis on messages
- Like interfaces for abstract objects

Declaring products as co-data

$$\begin{aligned}
&\textbf{codata}\ a \mathbin{\&} b\ \textbf{where} \\
&\qquad \mathrm{First} : \mid a \mathbin{\&} b \vdash a \\
&\qquad \mathrm{Second} : \mid a \mathbin{\&} b \vdash b
\end{aligned}$$

Declaring products as co-data

- Consumer: two observations (first or second)

$$\mathsf{First}[e_1] \qquad\qquad \mathsf{Second}[e_2]$$

- Producer: consider shape of output
  - "If I'm asked for first, do this"
  - "If I'm asked for second, do that"

$$\mu(\mathsf{First}[\alpha].c_1 \,|\, \mathsf{Second}[\beta].c_2)$$

Declaring functions as co-data

$$\begin{aligned}&\textbf{codata}\ a \to b\ \textbf{where}\\&\qquad \mathsf{Call} : a \mid a \to b \vdash b\end{aligned}$$

Declaring functions as co-data

- Consumer: one observation (function call)
  - Argument
  - What to do with result

$$\mathsf{Call}[v, e]$$

- Producer: consider shape of output
  - Function pops argument off call-stack

$$\mu(\mathsf{Call}[x, \alpha].c) = \lambda x.\mu\alpha.c$$

Evaluating data and co-data

- Two fundamental principles of data and co-data:
  - β: Case analysis breaks apart structure
  - η: Forwarding is unobservable
- Does not perform substitution
  - And therefore does not reference strategy
  - Hold in the presence of effects (control, non-termination)

Evaluating functions as co-data

$$(\beta) \qquad \langle \lambda x.v' \| v \cdot e \rangle = \langle v \| \tilde{\mu} x.\langle v' \| e \rangle \rangle$$

$$(\eta) \qquad \lambda x.\mu\alpha.\langle z \| x \cdot \alpha \rangle = z$$

$$(\beta) \quad \langle \mu(\mathsf{Call}[x, \alpha].c) \| \mathsf{Call}[v, e] \rangle = \langle v \| \tilde{\mu} x.\langle \mu\alpha.c \| e \rangle \rangle$$

$$(\eta) \quad \mu(\mathsf{Call}[x, \alpha].\langle z \| \mathsf{Call}[x, \alpha] \rangle) = z$$

Evaluating sums as data

$$(\beta) \qquad \left\langle \mathsf{Left}(v) \middle\| \begin{array}{ll} \tilde{\mu}[\,\mathsf{Left}(x). & c_1 \\ \quad \mid \mathsf{Right}(y). & c_2] \end{array} \right\rangle = \langle v \| \tilde{\mu} x . c_1 \rangle$$

$$(\beta) \qquad \left\langle \mathsf{Right}(v) \middle\| \begin{array}{ll} \tilde{\mu}[\,\mathsf{Left}(x). & c_1 \\ \quad \mid \mathsf{Right}(y). & c_2] \end{array} \right\rangle = \langle v \| \tilde{\mu} y . c_2 \rangle$$

$$(\eta) \qquad \begin{array}{ll} \tilde{\mu}[\,\mathsf{Left}(x). & \langle \mathsf{Left}(x) \| \gamma \rangle \\ \quad \mid \mathsf{Right}(y). & \langle \mathsf{Right}(y) \| \gamma \rangle] \end{array} = \gamma$$

Evaluating products as co-data

$$(\beta) \qquad \left\langle \begin{array}{l} \mu(\,\mathsf{First}[\alpha].\quad c_1 \\ \quad\mid \mathsf{Second}[\beta].\quad c_2) \end{array} \middle\| \mathsf{First}[e] \right\rangle = \langle \mu\alpha.c_1 \| e \rangle$$

$$(\beta) \qquad \left\langle \begin{array}{l} \mu(\,\mathsf{First}[\alpha].\quad c_1 \\ \quad\mid \mathsf{Second}[\beta].\quad c_2) \end{array} \middle\| \mathsf{Second}[e] \right\rangle = \langle \mu\beta.c_2 \| e \rangle$$

$$(\eta) \qquad \begin{array}{l} \mu(\,\mathsf{First}[\alpha].\quad \langle z \| \mathsf{First}[\alpha] \rangle \\ \quad\mid \mathsf{Second}[\beta].\quad \langle z \| \mathsf{Second}[\beta] \rangle) \end{array} = z$$

General characterization of data and co-data

- Constructors dual to messages, case abstractions dual to abstract objects
- All basic connectives of linear/polarized logic fit into same general pattern
  - The ordinary: $\to$, $\otimes$, $\oplus$, $\&$, ...
  - The exotic: $\parr$, $\neg$, ...
- All other behavior derived from β, η, and substitution:
  - Usual call-by-name and call-by-value λ-calculus β and η rules
  - Wadler's (2003) ς rules for lifting components out of structures

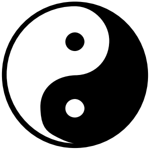
Summary

- Single theory of the sequent calculus parameterized by various strategies
- User-defined data and co-data defined by β and η independent of strategy
- Illustrate call-by-name, call-by-value, and lazy versions of both

Summary

- Generalize known dualities of computation
  - General duality between various strategies
  - General duality between data and co-data types
- Two or more strategies in the same program
  - Use kinds to denote strategies
  - Well-kindedness preserves consistency
  - Extends the polarized view of evaluation strategy

Questions?



Answers!

Interleaving multiple strategies

Conflicts between strategies

$$\langle \mu\alpha.c_1 \| \tilde{\mu}x.c_2 \rangle$$

	$\mu\alpha.c_1$	$\tilde{\mu}x.c_2$
CBV	non-value	co-value
CBN	value	non-co-value

Conflicts between strategies

$$\langle \mu\alpha.c_1 \| \tilde{\mu}x.c_2 \rangle$$

	$\mu\alpha.c_1$	$\tilde{\mu}x.c_2$
CBV	non-value	co-value
CBN	value	non-co-value

OK

Conflicts between strategies

$$\langle \mu\alpha.c_1 \| \tilde{\mu}x.c_2 \rangle$$

	$\mu\alpha.c_1$	$\tilde{\mu}x.c_2$
CBV	non-value	co-value
CBN	value	non-co-value

OK

Conflicts between strategies

$$\langle \mu\alpha.c_1 \| \tilde{\mu}x.c_2 \rangle$$

	$\mu\alpha.c_1$	$\tilde{\mu}x.c_2$
CBV	non-value	co-value
CBN	value	non-co-value

non-deterministic

Conflicts between strategies

$$\langle \mu\alpha.c_1 \| \tilde{\mu}x.c_2 \rangle$$

	$\mu\alpha.c_1$	$\tilde{\mu}x.c_2$
CBV	non-value	co-value
CBN	value	non-co-value

stuck

Well-kindedness preserves consistency

$$\frac{\Gamma \vdash v :: \mathcal{S} | \Delta \quad \Gamma | e :: \mathcal{S} \vdash \Delta}{\langle v \| e \rangle : \Gamma \vdash \Delta} \; Cut$$

- $\langle CBV \| CBV \rangle$: well-kinded, call-by-value command
- $\langle CBN \| CBN \rangle$: well-kinded, call-by-name command
- $\langle CBV \| CBN \rangle$: ill-kinded, non-deterministic command
- $\langle CBN \| CBV \rangle$: ill-kinded, stuck command

The polarized regime

. . . as an instance of the general theory:

- Only two kinds (therefore only two strategies)
  - Positive: call-by-value
  - Negative: call-by-name
- Pick strategy of (co-)data types to maximize η
  - Positive: data
  - Negative: co-data

Annotating variables

$$\frac{}{\Gamma, x :: \mathcal{S} \vdash x^{\mathcal{S}} :: \mathcal{S} | \Delta}\ \textit{Var} \qquad \frac{}{\Gamma | \alpha^{\mathcal{S}} :: \mathcal{S} \vdash \alpha :: \mathcal{S}, \Delta}\ \textit{CoVar}$$

$$\frac{c : (\Gamma \vdash \alpha :: \mathcal{S}, \Delta)}{\Gamma \vdash \mu\alpha^{\mathcal{S}}.c :: \mathcal{S} | \Delta}\ \textit{Act} \qquad \frac{c : (\Gamma, x :: \mathcal{S} \vdash \Delta)}{\Gamma | \tilde{\mu} x^{\mathcal{S}}.c :: \mathcal{S} \vdash \Delta}\ \textit{CoAct}$$

The problem with annotating commands

- Annotating commands (cuts) with a strategy:
  - $\langle v \| e \rangle^{\mathcal{V}}$: call-by-value
  - $\langle v \| e \rangle^{\mathcal{N}}$: call-by-name
- Loss of determinism

$$
\begin{tikzcd}
 & \langle \mu\_.c_1 \| \tilde{\mu}x.\langle x \| \tilde{\mu}\_.c_2 \rangle^{\mathcal{V}} \rangle^{\mathcal{N}} \arrow[dl, "\tilde{\mu}"'] \arrow[dr, "\tilde{\mu} \text{ or } \eta_{\tilde{\mu}}"] & \\
\langle \mu\_.c_1 \| \tilde{\mu}\_.c_2 \rangle^{\mathcal{V}} \arrow[d, "\mu"'] & & \langle \mu\_.c_1 \| \tilde{\mu}\_.c_2 \rangle^{\mathcal{N}} \arrow[d, "\tilde{\mu}"] \\
c_1 & & c_2
\end{tikzcd}
$$