

---

# Cycles and Control Operators



Zena M. Ariola  
University of Oregon

20 March 2006

---

---

Coinductive Proofs  
≡  
Infinitary Rewriting  
≡  
 $\lambda$ -calculus with Control Operators

---

# Why Control Operators?

- Control operators can express **recursion**. Filinski, Recursion from Iteration. Lisp and Symbolic Computation, 1994
- Control operators can express **streams**
- Control operators can express **irregular trees**
- Would it be nice if stream equality can be done using usual equality reasoning?

---

# Why Control Operators? (Cont.)

- Control Operators have a logic foundation
- We have expressiveness results for control operators :

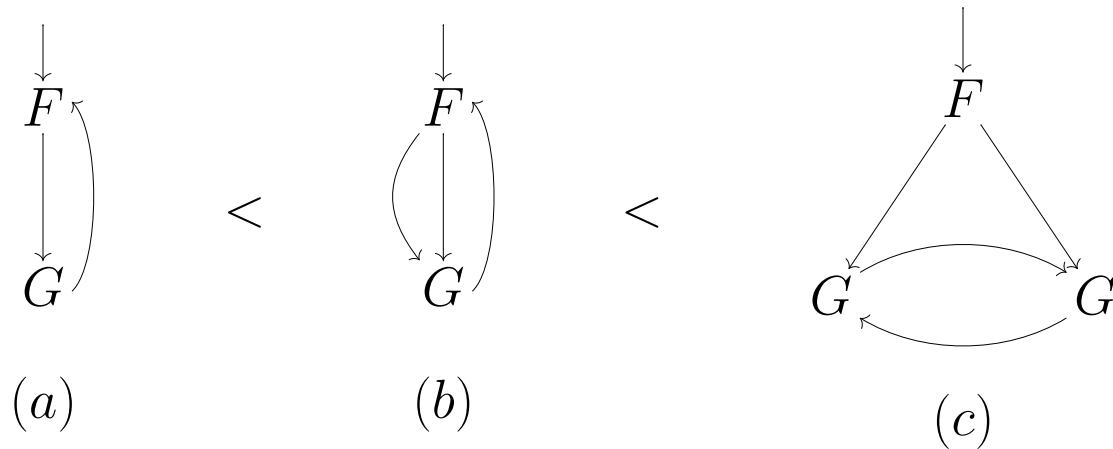
*$\mathcal{C}$  is more expressive than callcc  
shift/reset are more expressive than  $\mathcal{C}$*

- By relating cycles to control operators we hope to derive a classification of cyclic definitions and of patterns in streams

---

# Expressiveness (S. Blom, PhD 2001)

Can we say that some cycles are more expressive than others?



a) Vertical sharing :  $\mu$

b) Vertical and horizontal sharing:  $\mu$  and  $let$

c) Twisted sharing:  $letrec$

---

# Continuations

---

# Continuations vs Functions

A **continuation** represents a label or control point. Unlike a function, a continuation never returns to the invocation point.

Invoking a continuation corresponds to performing a **jump** or **go to**. If  $k$  stands for  $\lambda x.x$  then :

$$\begin{array}{l} 1 + (\lambda x.x) 5 \rightarrow 1 + 5 \\ 1 + k 5 \quad \rightarrow 5 \end{array}$$

What is  $\lambda x.x$ ?

$\lambda x.Abort\ x$

The presence of the **Abort** is what distinguishes a continuation from a function

$$1 + (\lambda x.Abort\ x) 5 \rightarrow 1 + Abort\ 5 \rightarrow 5$$

*Abort M aborts everything, returns M to the top-level*

---

# Control Operators

$$(2 + 3) + (8 * 9) + (2 + 4)$$

Evaluate  $2+3$ :

$$5 + (8 * 9) + (2 + 4)$$

Evaluate  $8+9$  and remember what needs to be done afterwards:

$$\lambda x. Abort (5 + x + (2 + 4))$$

How do we define that continuation?

$$(2 + 3) + \mathcal{C}(\lambda k.??) + (2 + 4)$$

In a left-to-right evaluation,  $k$  is bound to  $\lambda x. Abort (5 + x + (2 + 4))$ . In a right-to-left evaluation,  $k$  is bound to  $\lambda x. Abort ((2 + 3) + x + 6)$ .

---

# Evaluation of $\mathcal{C}(\lambda k.M)$

$$2 + \mathcal{C}(\lambda k.5) + (2 + 4)$$

$k$  is bound to  $\lambda x.Abort (2 + x + (2 + 4))$

The surrounding context is abandoned and replaced with the body of the  $\mathcal{C}$ -expression with  $k$  substituted with a functional representation of the surrounding context.

$$2 + \mathcal{C}(\lambda k.5) + (2 + 4) \rightarrow 5[k := \lambda x.Abort (2 + x + (2 + 4))] = 5$$

---

# Evaluation of $\mathcal{C}(\lambda k.M)$

$$\begin{aligned} 2 + \mathcal{C}(\lambda k.k\ 5) + (2 + 4) &\rightarrow \\ k\ 5[k := \lambda x.Abort\ (2 + x + (2 + 4))] &= \\ (\lambda x.Abort\ (2 + x + (2 + 4)))\ 5 &\rightarrow \\ Abort\ (2 + 5 + (2 + 4)) &\rightarrow \\ 2 + 5 + (2 + 4) &\rightarrow \\ 13 & \end{aligned}$$

$$\begin{aligned} 2 + \mathcal{C}(\lambda k.1 + k\ 5) + (2 + 4) &\rightarrow \\ 1 + k\ 5[k := \lambda x.Abort\ (2 + x + (2 + 4))] &= \\ 1 + (\lambda x.Abort\ (2 + x + (2 + 4)))\ 5 &\rightarrow \\ 1 + Abort\ (2 + 5 + (2 + 4)) &\rightarrow \\ Abort\ (2 + 5 + (2 + 4)) &\rightarrow \\ 2 + 5 + (2 + 4) & \end{aligned}$$

---

# $c$ vs $\text{Callcc}$

$$\text{callcc}(\lambda k.M) = \mathcal{C}(\lambda k.k M)$$

$$2 + \mathcal{C}(\lambda k.5) + (2 + 4) \rightarrow 5$$

$$2 + \text{callcc}(\lambda k.5) + (2 + 4) \rightarrow 2 + 5 + (2 + 4)$$

---

# The fixpoint combinator

$$Y M = ((\lambda z.z (\lambda x.\lambda n.(M (z x) n)))(callcc(callcc(\lambda x.x))))$$

---

# Continuations and Logic

---

# Minimal Logic and $\lambda$ -calculus

$$A ::= X \mid A \rightarrow A$$
$$M, N ::= x \mid \lambda x.M \mid MN$$
$$\frac{}{\Gamma, x : A \vdash x : A}$$
$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \rightarrow B}$$
$$\frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

---

# Control operators and logic

- *Abort* corresponds to **Ex Falso Quodlibet**

$$\frac{\Gamma \vdash M : \perp}{\Gamma \vdash \text{Abort}(M) : A}$$

- *C* corresponds to Proof by contradiction (**Double Negation law**):

$$\frac{\Gamma \vdash M : (A \rightarrow \perp) \rightarrow \perp}{\Gamma \vdash C(M) : A}$$

- *callcc* corresponds to **Pierce Law** :

$$\frac{\Gamma \vdash M : (A \rightarrow B) \rightarrow B}{\Gamma \vdash \text{callcc}(M) : A}$$

---

# Control Operators and their expressiveness

Minimal Classical Logic = Minimal Logic + Pierce Law

Classical Logic = Minimal Classical Logic + Ex Falso Quodlibet

$\lambda$ -calculus + callcc = Minimal Classical Logic

$\lambda$ -calculus +  $\mathcal{C}$  = Classical Logic

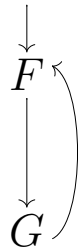
$\mathcal{C}$  is more expressive than callcc

---

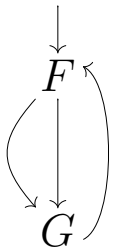
# Cycles and Logic

---

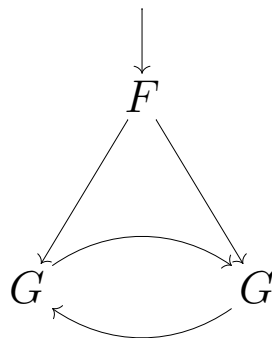
# Cycles



Vertical sharing is minimal classical



Is Horizontal sharing classical?



twisted sharing needs delimited control operators

---

# Delimited Control Operators : shift

The continuation does not include the abort

$$\mathcal{C}(\lambda k.1 + k\ 2) \dashv\dashv 2$$
$$\mathit{shift}(\lambda k.1 + k\ 2) \dashv\dashv 3$$

---

# Delimited Control Operators (Cont.)

*if*  $\mathcal{C}(\lambda k.k(\text{is-zero}(k \text{ true} - 1)))$  *then* 2 *else* 9  $\rightarrow$  2

*if* *shift*( $\lambda k.k(\text{is-zero}(k \text{ true} - 1)))$  *then* 2 *else* 9  $\rightarrow$  9

$\mathcal{C}(\lambda k.k(k - 1)) + 2 \rightarrow$  3

*shift*( $\lambda k.k(k - 1)) + 2 \rightarrow$  5

---

# Delimited Control Operators: reset

A reset delimits the continuation captured by shift

$$1 + \mathit{reset} (2 + \mathit{shift}(\lambda k.k \ 2))$$

$k$  is bound to  $\lambda x.2 + x$  and not to  $\lambda x.1 + (2 + x)$

In  $\mathit{shift}(\lambda k.M)$ , continuation  $k$  extends to the **dynamically closest reset**:

$$\mathit{reset}((\mathit{reset}(\lambda x.\mathit{shift}(\lambda k.(\lambda_.3))))(\lambda_.\mathit{shift}(\lambda k.4))) \rightarrow \mathit{reset}(((\lambda x.\mathit{shift}(\lambda k.(\lambda_.3))))(\lambda_.\mathit{shift}(\lambda k.4)))$$

---

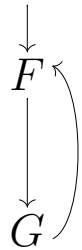
# Dynamic scope + control $\Rightarrow$ state

$$\frac{}{\Gamma, x : A; T \vdash x : A; T} Ax \qquad \frac{\Gamma, x : A; U \vdash M : B; T}{\Gamma; T' \vdash \lambda x. M : A \multimap_T B; T'} \rightarrow$$
$$\frac{\Gamma; U_1 \vdash M : A \multimap_{T_1} B; T_2 \qquad \Gamma; T_1 \vdash N : A; U_1}{\Gamma; U_2 \vdash MN : B; T_2} \rightarrow$$

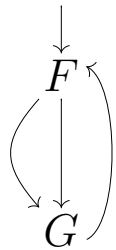
How do we interpret these effects?  
Subtractive Logic:  $A - B = A \wedge \neg B$

---

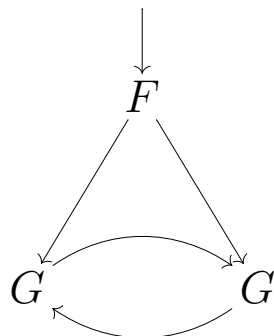
# Cycles



Vertical sharing is minimal classical



Is Horizontal sharing classical?



Is Twisted sharing subtractive logic?

- 
- Can shift and reset represent irregular trees
  - Can stream equality be realized using simple equality reasoning
  - Cycles and streams represent proofs. What can we learn from using known theorem provers or proof assistants

---

# Conclusion

Cycles

Streams

Continuations