Simulating call by value in Combinatory Logic

Paolo Parisen Toldin

University of Bologna

10 june 2010

Introduction

Motivations:

- What is the dynamic significance of the encoding of λ-calculus into Combinatory Logic?
  - Which notion of reduction in Λ can be simulated in $\mathbf{CL}$?
- How much does it cost to do a step of reduction in λ-calculus with cbv?
  - We can assume the cost of a reduction step to be unitary. [Moran04, DallagoMartini09]
  - Let's translate in $\mathbf{CL}$ and simulate it. We will reduce to λ calculus weak.
- What are the relations between the complexity of λ steps and a $\mathbf{CL}$ step?
- We shall consider the λ-calculus weak.

Bibliography

Related works in bibliography:

- *Call-by-Value Combinatory Logic and the Lambda-Value Calculus* - J.Gateley & B.F.Duba
- *A New Implementation Tecnique for Applicative Languages* - D.A.Turner
- *On Constructor Rewrite Systems and Lambda Calculus* - U.Dal lago & S.Martini

Translation $\Lambda \rightarrow$CL

The Curry translation:

$$\begin{array}{lcl} [x]_\eta & = & x \\ [MN]_\eta & = & [M]_\eta [N]_\eta \\ [\lambda x.M]_\eta & = & [x]_\mu.[M]_\eta \end{array}$$

where

$$\begin{array}{lcll} [x]_\mu.M & = & \mathbf{K}M & x \notin FV(M) \\ [x]_\mu.x & = & \mathbf{I} & \\ [x]_\mu.\mathbf{C} & = & \mathbf{KC} & \text{with } \mathbf{C} \text{ combinator} \\ [x]_\mu.MN & = & \mathbf{S}([x]_\mu.M)([x]_\mu.N) & \text{otherwise} \end{array}$$

Bad Properties

- It does not map normal forms to normal forms.

Example

$\lambda x.\Delta\Delta$ is NF in the weak λ-calculus.
$[\lambda x.\Delta\Delta]_\eta = \mathbf{K}((\mathbf{SII})(\mathbf{SII}))$ is not NF in $\mathbf{CL}$

- In general, strong reduction cannot be simulated.

Example

$\lambda x.(\lambda y.y)x \to \lambda x.x$
$[\lambda x.(\lambda y.y)x] = \mathbf{S}(\mathbf{KI})\mathbf{I}$

- Weak reduction cannot be simulated exactly: λ weak is not confluent while $\mathbf{CL}$ is confluent.

Another abstraction algorithm

Do we need to choose another abstraction algoritm? NO
consider the follow abstraction algorithm:

$$
\begin{array}{llll}
[x]_\nu.y & = & \mathbf{K}y & x \neq y \\
[x]_\nu.x & = & \mathbf{I} & \\
[x]_\nu.\mathbf{C} & = & \mathbf{KC} & \text{with } \mathbf{C} \text{ combinator} \\
[x]_\nu.MN & = & \mathbf{S}([x]_\nu.M)([x]_\nu.N) & \text{otherwise}
\end{array}
$$

This algoritm differs from the previus for the first rule.
It maps NF to NF, but it cannot be used for simulating call by value: it does not preserve the substitution.

Translation CL→Λ

From Combinatory Logic to λ calculus, instead of using the standard translation

$$
\begin{aligned}
[x]_\lambda &= x & [\mathbf{I}]_\lambda &= \lambda x.x \\
[\mathbf{K}]_\lambda &= \lambda xy.x & [\mathbf{S}]_\lambda &= \lambda xyz.xz(yz) \\
[XY]_\lambda &= [X]_\lambda[Y]_\lambda
\end{aligned}
$$

we'll use the followed one:

$$
\begin{aligned}
[x]_\lambda &= x & [\mathbf{I}]_\lambda &= \lambda x.x \\
[\mathbf{K}M]_\lambda &= \lambda y.[M]_\lambda & [\mathbf{S}MN]_\lambda &= \lambda z.[M]_\lambda z([N]_\lambda z) \\
[XY]_\lambda &= [X]_\lambda[Y]_\lambda
\end{aligned}
$$

we'll se why...

Call-by-Value in $\mathbf{CL}$

Values:

$$V_2 = \{\mathbf{S}\}$$
$$V_1 = \{\mathbf{K}\} \cup \{MN | M \in V_2 \wedge N \in \mathbf{CL}\}$$
$$V_0 = \{\mathbf{I}\} \cup \{MN | M \in V_1 \wedge N \in \mathbf{CL}\}$$
$$\mathbf{V} = \{M | M \in V_0 \vee M \in V_1 \vee M \in V_2 \vee M = x\}$$

$$\frac{M \in \mathbf{V}}{\mathbf{I}M \triangleright_{wCBV} M} \qquad \frac{N \in \mathbf{V}}{\mathbf{K}MN \triangleright_{wCBV} M}$$

$$\frac{P \in \mathbf{V}}{\mathbf{S}MNP \triangleright_{wCBV} (MP)(NP)}$$

$$\frac{M \triangleright_{wCBV} N}{ML \triangleright_{wCBV} NL} \qquad \frac{M \triangleright_{wCBV} N \qquad L \notin V_1 \cup V_2}{LM \triangleright_{wCBV} LN}$$

Call-by-Value in $\mathbf{CL}$ V.2

Let's reformulate the rules in the following way. Let's consider $\mathbf{S}, \mathbf{K}, \mathbf{I}$ as functions, respectively, 2-ary, 1-ary, 0-ary.
Here, simply, the set of values is defined as:
$\mathbf{V} = \{M | M = I \vee M = K(P) \vee M = S(P,Q) \vee M = x, \text{for some P,Q}\}$

$$\frac{M \in \mathbf{V}}{\mathbf{I}M \triangleright_{wCBV} M} \qquad \frac{N \in \mathbf{V}}{\mathbf{K}(M)N \triangleright_{wCBV} M}$$

$$\frac{P \in \mathbf{V}}{\mathbf{S}(M,N)P \triangleright_{wCBV} (MP)(NP)} \qquad \frac{M \triangleright_{wCBV} N}{ML \triangleright_{wCBV} NL}$$

The $[.]_\mu$ abstraction produces always a term in $\mathbf{V}$!

Results:

Theorem (lambda to combinatory)

If $M \to_{\lambda wCBV} M'$ *then* $[M]_\eta \rhd_{wCBV}^{2\times|M|-1} [M']_\eta$

Theorem (combinatory to lambda)

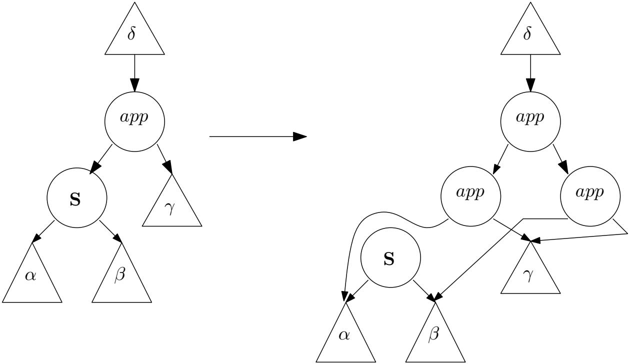
If $M \rhd_{wCBV} M'$ *then* $[M]_\lambda \to^*_{\lambda w} [M']_\lambda$

Theorem (combinatory to lambda)

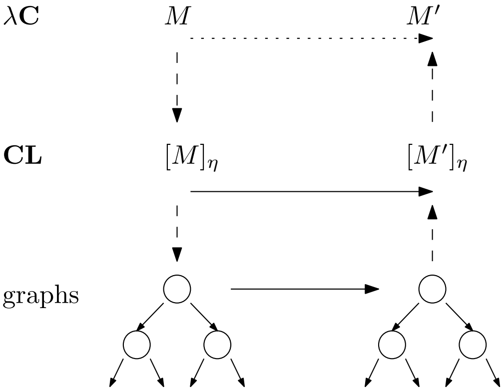
$[\cdot]_\lambda$ *maps NFs to NFs.*

Graph Implementation

- The reduction steps in **CL** may duplicate sub-terms.
- We can implement it by using term graph rewriting as explained in [Turner79] by sharing these subgraphs.



General Overview



Work in Progress

What about the call by name?

We have several ideas, still a working in progress.
The main problem lies in the fact that doing only leftmost step is not enough.

$(\lambda z.\lambda x.xz)a \rightarrow \lambda x.xa$

$(\mathbf{S}\underbrace{(\mathbf{K}(\mathbf{SI}))}\underbrace{((\mathbf{S}(\mathbf{KK}))\mathbf{I})))}a \triangleright [\mathbf{K}(\mathbf{SI})a][((\mathbf{S}(\mathbf{KK}))\mathbf{I})a]\ldots$

Profiling tool

With Marco Gaboardi, i'm developing a general framework to deal with languages based on Combinatory Logic.

CoLoBo - Combinatory Logic in Bologna (http://colobo.sourceforge.net/)

The idea is to develop a tool to perform evaluation of quantitative properties (e.g. number of steps). It also works as a generic interpreter.

End

Questions?