

# On Constructor Rewrite Systems and the Lambda-Calculus

Ugo Dal Lago    *Simone Martini*

Dipartimento di Scienze dell'Informazione  
*Alma mater studiorum* • Università di Bologna

ICALP 2009 Track B, Rhodes – July 7th, 2009



*In memoriam*  
Peter J. Landin  
June 3rd, 2009



# Part I

## The result and the context



## The result, in general terms

- A *linear* simulation between
  - ▶ a *first order* computation model (term rewriting), and
  - ▶ a *higher order* computation model ( $\lambda$ -calculus)
- each equipped with its *most natural, intrinsic cost* parameter,
- which is *polynomially related* to the actual cost of normalization, as measured on a Turing machine



## The context: *Implicit Computational Complexity*

- A machine-free, logic-based investigation of the notion of *feasible computation*
- Feasibility through *language restrictions*, and not external measure conditions
- Incorporate computational complexity into formal methods in software development and programming language design



# Implicit Computational Complexity

- **In the large:** study and characterize complexity classes  
e.g., Bellantoni-Cook; Girard's lighth logics; etc.
- **In the small:** study and relate machine-free models of  
computation *i.e.*, models with no notion of constant-time step



# Implicit Computational Complexity

- **In the large:** study and characterize complexity classes  
e.g., Bellantoni-Cook; Girard's lighth logics; etc.
- **In the small:** study and relate machine-free models of computation *i.e.*, models with no notion of constant-time step



## ICC “in the small”

Results on specific computational “machine-free” models.

*Do we have a natural intrinsic measure for the cost of reducing a  $\lambda$ -term to normal form ?*

- The measure should be **polynomially** related to actual cost (“reasonable machine models”)
- For full  $\beta$ -reduction, the answer is **no** (or, we do not know)



## ICC “in the small”

Results on specific computational “machine-free” models.

*Do we have a natural **intrinsic** measure for the cost of reducing a  $\lambda$ -term to normal form ?*

- The measure should be **polynomially** related to actual cost (“reasonable machine models”)
- For full  $\beta$ -reduction, the answer is **no** (or, we do not know)



## Part II

### The simulation

- Weak call-by-value  $\lambda$ -calculus
- Constructor term rewriting
- and their mutual simulation



## First actor: weak call-by-value $\lambda$ -calculus

- Terms  $M ::= x \mid \lambda x.M \mid MM$
- Values  $V ::= x \mid \lambda x.M$
- Weak call-by-value reduction

$$\frac{}{(\lambda x.M)V \rightarrow_v M\{V/x\}} \qquad \frac{M \rightarrow_v N}{ML \rightarrow_v NL} \qquad \frac{M \rightarrow_v N}{LM \rightarrow_v LN}$$

- Values may be duplicated during reduction
- Is the number of reduction steps a good measure of actual cost?

(Yes: Sands, Gustavsson, and Moran, 2002)



## Second actor: Orthogonal constructor term rewriting

- Symbols, partitioned in **constructors** and **functions**
- **Patterns**: terms over constructors and variables
- Rules:  $f(\mathbf{p}_1, \dots, \mathbf{p}_n) \rightarrow_{\Xi} t$   
f is a function symbol;  $\mathbf{p}_1, \dots, \mathbf{p}_n$  are patterns;  $t$  is a (general) term.
- **Orthogonal**: no rule overlapping; left-linear
- **Call-by-value**: reduction happens with constructor terms substituted for variables.



## The result

From  $\lambda$  to constructor rewriting:

$$\begin{array}{ccc} M & \xrightarrow{\quad \overset{n}{v} \quad} & N \\ \downarrow [\ ]_{\Phi} & & \uparrow \langle \rangle \\ [M]_{\Phi} & \xrightarrow{\quad \overset{n}{t} \quad} & \end{array}$$

From constructor rewriting to  $\lambda$ :

$$\begin{array}{ccc} \mathbf{f}(t_1, \dots, t_h) & \xrightarrow{\quad \overset{n}{v} \quad} & u \\ \downarrow [\ ]_{\wedge} \langle \rangle & & \downarrow \\ [\mathbf{f}]_{\wedge} \langle \langle t_1 \rangle \rangle \dots \langle \langle t_h \rangle \rangle & \xrightarrow{\quad \overset{kn}{v} \quad} & \langle \langle u \rangle \rangle \end{array}$$



## Relevance: general

- In weak call-by-value, the number of  $\beta$  steps is a good cost model:
  - ▶ We may *actually compute* on a Turing machine the normal form of  $M$  in that bound (modulo a poly)
  - ▶ Any TM machine may be simulated by a lambda-term normalizing in the same time of the TM (modulo a poly)

Sands, Gustavsson, and Moran, 2002

Also: [second part of the paper](#)

Dal Lago and M., 2006

*Constructor rewriting and weak c-b-v  $\lambda$ -calculus are both reasonable machines, under their natural cost models: number of rewriting steps, and number of beta-reductions.*

The *logical*, abstract notion of computation may be used as an actual cost measure.



## Relevance: general

- In weak call-by-value, the number of  $\beta$  steps is a good cost model:
  - ▶ We may *actually compute* on a Turing machine the normal form of  $M$  in that bound (modulo a poly)
  - ▶ Any TM machine may be simulated by a lambda-term normalizing in the same time of the TM (modulo a poly)

Sands, Gustavsson, and Moran, 2002

Also: [second part of the paper](#)

Dal Lago and M., 2006

*Constructor rewriting and weak c-b-v  $\lambda$ -calculus are both reasonable machines, under their natural cost models: number of rewriting steps, and number of beta-reductions.*

The *logical*, abstract notion of computation may be used as an actual cost measure.



## Relevance: specific

- The simulation cannot be obtained with Church-like encoding of data

- ▶ There is no constant-time predecessor on Church numerals

Parigot, 1990

- ▶ Instead

$$\begin{array}{ccc} \text{Pred}(\text{succ}(n)) & \longrightarrow & \overset{1}{n} \\ \downarrow & & \downarrow \\ [\text{pred}]_{\wedge} \langle\langle \text{succ}(n) \rangle\rangle & \longrightarrow & \overset{k}{\underset{v}{\langle\langle n \rangle\rangle}} \end{array}$$



## Relevance: specific

- The simulation cannot be obtained with Church-like encoding of data

- ▶ There is no constant-time predecessor on Church numerals

Parigot, 1990

- ▶ Instead

$$\begin{array}{ccc} \text{Pred}(\text{succ}(n)) & \longrightarrow & \overset{1}{n} \\ \downarrow & & \downarrow \\ [\text{pred}]_{\wedge} \langle\langle \text{succ}(n) \rangle\rangle & \longrightarrow & \overset{k}{\underset{v}{\langle\langle n \rangle\rangle}} \end{array}$$



## First simulation: From $\lambda$ to constructor rewriting

Idea: full defunctionalization

Any  $\lambda$ -abstraction becomes a constructor

$$[x]_{\Phi} = x;$$

$$[\lambda x.M]_{\Phi} = \mathbf{c}_{x,M}(x_1, \dots, x_n), \text{ where } \text{FV}(\lambda x.M) = x_1, \dots, x_n;$$

$$[MN]_{\Phi} = \mathbf{app}([M]_{\Phi}, [N]_{\Phi}).$$

- Constructors:  $\mathbf{c}_{x,M}$  for any  $M$  and any  $x$ .
- Functions:  $\mathbf{app}$ .
- Reduction rules:

$$\mathbf{app}(\mathbf{c}_{x,M}(x_1, \dots, x_n), x) \rightarrow [M]_{\Phi}$$



## First simulation, 2

- In the other direction:

$$\langle x \rangle_{\wedge} = x$$

$$\langle \mathbf{app}(u, v) \rangle_{\wedge} = \langle u \rangle_{\wedge} \langle v \rangle_{\wedge}$$

$$\langle \mathbf{c}_{x, M}(t_1, \dots, t_n) \rangle_{\wedge} = (\lambda x. M) \{ \langle t_1 \rangle_{\wedge} / x_1, \dots, \langle t_n \rangle_{\wedge} / x_n \}$$

- $\langle [M]_{\Phi} \rangle_{\wedge} = M$
- For *canonical*  $t$ , if  $t \rightarrow u$ , then  $\langle t \rangle_{\wedge} \rightarrow_v \langle u \rangle_{\wedge}$

### Theorem (Simulation)

Let  $M$  be a closed  $\lambda$ -term. The following are equivalent:

- 1  $M \rightarrow_v^n N$  where  $N$  is in normal form;
- 2  $[M]_{\Phi} \rightarrow^n t$  where  $\langle t \rangle_{\wedge} = N$  and  $t$  is in normal form.



## Second simulation: From constructor rewriting to $\lambda$

First: Encode *data*, i.e. constructor terms

- Use **Scott numerals**-like encoding:

$$\begin{aligned}\underline{0} &\equiv \lambda x_1. \lambda x_2. x_1 \\ \underline{n+1} &\equiv \lambda x_1. \lambda x_2. \underline{n}\end{aligned}$$

- Here:  $\langle\langle \rangle\rangle_{\Lambda}$  : constructor terms  $\rightarrow$   $\lambda$ -terms  
For constructors  $\mathbf{c}_1, \dots, \mathbf{c}_g$ :

$$\langle\langle \mathbf{c}_i(t_1 \dots, t_n) \rangle\rangle_{\Lambda} \equiv \lambda x_1. \dots. \lambda x_g. \lambda y. x_i \langle\langle t_1 \rangle\rangle_{\Lambda} \dots \langle\langle t_n \rangle\rangle_{\Lambda}.$$

- $\perp \equiv \lambda x_1. \dots. \lambda x_g. \lambda y. y$  denotes an error value



## Second simulation, 2

Second: Encode *pattern matching*

- On an example:

$$f(\mathbf{p}_1^1(x_1, x_2), \mathbf{p}_2^1(x_3), \mathbf{p}_3^1(x_4)) \rightarrow t_1$$
$$f(\mathbf{p}_1^2(x_5), \mathbf{p}_2^2(x_6, x_7), \mathbf{p}_3^2(x_8)) \rightarrow t_2$$

- Given such a sequence  $\alpha_1, \alpha_2$  of patterns, construct a selector  $M_{\alpha_1, \alpha_2}^3$  s.t., for  $k$  depending only on  $\alpha_1, \alpha_2$



## Second simulation, 2

Second: Encode *pattern matching*

- On an example:

$$\begin{aligned} f(\mathbf{p}_1^1(x_1, x_2), \mathbf{p}_2^1(x_3), \mathbf{p}_3^1(x_4)) &\rightarrow t_1 \\ f(\mathbf{p}_1^2(x_5), \mathbf{p}_2^2(x_6, x_7), \mathbf{p}_3^2(x_8)) &\rightarrow t_2 \end{aligned}$$

- Given such a sequence  $\alpha_1, \alpha_2$  of patterns, construct a selector  $M_{\alpha_1, \alpha_2}^3$  s.t., for  $k$  depending only on  $\alpha_1, \alpha_2$

$$\begin{aligned} M_{\alpha_1, \alpha_2}^3 \langle\langle \mathbf{p}_1^1(t_1, t_2) \rangle\rangle \wedge \langle\langle \mathbf{p}_2^1(t_3) \rangle\rangle \wedge \langle\langle \mathbf{p}_3^1(t_4) \rangle\rangle \wedge V_1 V_2 \\ \rightarrow_v^k V_1 \langle\langle t_1 \rangle\rangle \wedge \dots \langle\langle t_4 \rangle\rangle \wedge \end{aligned}$$



## Second simulation, 2

Second: Encode *pattern matching*

- On an example:

$$\begin{aligned} f(\mathbf{p}_1^1(x_1, x_2), \mathbf{p}_2^1(x_3), \mathbf{p}_3^1(x_4)) &\rightarrow t_1 \\ f(\mathbf{p}_1^2(x_5), \mathbf{p}_2^2(x_6, x_7), \mathbf{p}_3^2(x_8)) &\rightarrow t_2 \end{aligned}$$

- Given such a sequence  $\alpha_1, \alpha_2$  of patterns, construct a selector  $M_{\alpha_1, \alpha_2}^3$  s.t., for  $k$  depending only on  $\alpha_1, \alpha_2$

$$\begin{aligned} M_{\alpha_1, \alpha_2}^3 \langle\langle \mathbf{p}_1^2(t_5) \rangle\rangle \wedge \langle\langle \mathbf{p}_2^2(t_6, t_7) \rangle\rangle \wedge \langle\langle \mathbf{p}_3^2(t_8) \rangle\rangle \wedge V_1 V_2 \\ \rightarrow_v^k V_2 \langle\langle t_1 \rangle\rangle \wedge \dots \langle\langle t_4 \rangle\rangle \wedge \end{aligned}$$



## Second simulation, 2

Second: Encode *pattern matching*

- On an example:

$$\begin{aligned} f(\mathbf{p}_1^1(x_1, x_2), \mathbf{p}_2^1(x_3), \mathbf{p}_3^1(x_4)) &\rightarrow t_1 \\ f(\mathbf{p}_1^2(x_5), \mathbf{p}_2^2(x_6, x_7), \mathbf{p}_3^2(x_8)) &\rightarrow t_2 \end{aligned}$$

- Given such a sequence  $\alpha_1, \alpha_2$  of patterns, construct a selector  $M_{\alpha_1, \alpha_2}^3$  s.t., for  $k$  depending only on  $\alpha_1, \alpha_2$

$$\begin{array}{cccccc} M_{\alpha_1, \alpha_2}^3 & X_1 & X_2 & X_3 & V_1 & V_2 \\ & & & & \rightarrow_v^k & \perp \end{array}$$

if any of the  $X_i$  does not match one of  $\alpha_1, \alpha_2$ , or is  $\perp$ .



## Second simulation, 3

Third: Solve *mutual recursion*

$$\begin{aligned} \mathbf{f}_i(\alpha_i^1) &\rightarrow t_i^1 \\ &\vdots \\ \mathbf{f}_i(\alpha_i^n) &\rightarrow t_i^n. \end{aligned}$$

*C-b-v fixpoint operators*

For any  $h$ , there are  $H_1, \dots, H_h$  and a *bound*  $m$ , such that:

$$H_i V_1 \dots V_h \rightarrow_v^m V_i(\lambda x. H_1 V_1 \dots V_h x) \dots (\lambda x. H_h V_1 \dots V_h x).$$

$$[\mathbf{f}_i]_{\wedge} \equiv H_i V_1 \dots (\overline{\lambda x. \overline{\lambda y. M_{\alpha_i^1, \dots, \alpha_i^n} y}(\overline{\lambda z} \langle t_i^1 \rangle_{\wedge}) \dots (\overline{\lambda z} \langle t_i^n \rangle_{\wedge})}) \dots V_h$$



## Second simulation, 4

Theorem: There is  $k$  such that for any  $\mathbf{f}$

1

$$\begin{array}{ccc} \mathbf{f}(t_1, \dots, t_h) & \longrightarrow & {}^n u \in \mathcal{C}(\Xi) \\ \downarrow \llbracket \cdot \rrbracket_{\wedge} & & \downarrow \\ \llbracket \mathbf{f} \rrbracket_{\wedge} \llbracket t_1 \rrbracket \dots \llbracket t_h \rrbracket & \longrightarrow & {}_{\vee}^{kn} \llbracket u \rrbracket \end{array}$$

2

$$\begin{array}{ccc} \mathbf{f}(t_1, \dots, t_h) & \longrightarrow & {}^n u \notin \mathcal{C}(\Xi) \\ \downarrow \llbracket \cdot \rrbracket_{\wedge} & & \downarrow \\ \llbracket \mathbf{f} \rrbracket_{\wedge} \llbracket t_1 \rrbracket \dots \llbracket t_h \rrbracket & \longrightarrow & {}_{\vee}^{kn} \perp \end{array}$$

3  $\mathbf{f}(t_1, \dots, t_h)$  diverges, then  $\llbracket \mathbf{f} \rrbracket_{\wedge} \llbracket t_1 \rrbracket \dots \llbracket t_h \rrbracket$  diverges.



## Part III

They are reasonable machine models

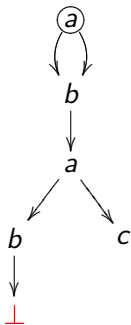
*The number of reduction steps to normal form is **polynomially related** to the actual cost of normalization, as measured on a Turing machine.*



# Term graph rewriting, sketch 1

$$a(b(a(b(x), c)), b(a(b(x), c)))$$

- Represent a terms  $t$  with a graph  $[t]_G$ , allowing sharing



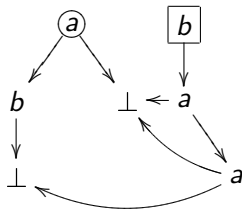
- Define a suitable “unsharing” of a graph,  $\langle G \rangle_{\mathcal{R}}$



## Term graph rewriting, sketch 2

- Represent rules with graph rewrite rules

$$a(b(x), y) \rightarrow b(a(y, a(y, x)))$$



# Graph reducibility

Any constructor rewriting can be implemented with graph rewriting.

## Theorem (Graph Reducibility)

*The following are equivalent:*

- 1  $t \rightarrow^n u$ , where  $u$  is in normal form;
- 2  $[t]_g \rightarrow^n G$ , where  $G$  is in normal form and  $\langle G \rangle_{\mathcal{R}} = u$ .



## With c-b-v $\lambda$ -calculus

### Corollary

Let  $M$  be a closed term. The following are equivalent:

- 1  $M \rightarrow_{\vee}^n N$  where  $N$  is in normal form;
- 2  $[[M]_{\Phi}]_{\Theta} \rightarrow^n G$  where  $\langle\langle G \rangle_{\Phi}\rangle_{\wedge} = N$  and  $G$  is in normal form.

- We want to relate  $n$  to the **actual cost** of reducing
- Key: When  $[M]_{\Phi} \rightarrow^* t$ , for any occurrence of a constructor  $c_{x,N}$  in  $t$ ,  $N$  is a subterm of  $M$
- We bound the **size** of the graphs along the reduction of  $[[M]_{\Phi}]_{\Theta}$



## A reasonable machine model

### Theorem

*There is a polynomial  $p : \mathbb{N}^2 \rightarrow \mathbb{N}$  such that for every  $\lambda$ -term  $M$ , the normal form of  $[[M]_{\Phi}]_{\Theta}$  can be computed in time at most  $p(|M|, n)$ .*



## Conclusions

*Constructor rewriting and weak c-b-v  $\lambda$ -calculus are both reasonable machines, under their natural cost models: number of rewriting steps, and number of beta-reductions.*

The *logical*, abstract notion of computation may be used as an actual cost measure.

- ① Extension to weak call-by-name in the full paper
- ② Full  $\beta$ -reduction?
- ③ A typed context?



## Conclusions

*Constructor rewriting and weak c-b-v  $\lambda$ -calculus are both reasonable machines, under their natural cost models: number of rewriting steps, and number of beta-reductions.*

The *logical*, abstract notion of computation may be used as an actual cost measure.

- 1 Extension to weak call-by-name in the full paper
- 2 Full  $\beta$ -reduction?
- 3 A typed context?

