

A λ -calculus with constructors

Ariel Arbiser (U. Buenos Aires)
Alexandre Miquel (U. Paris 7, PPS)
Alejandro Ríos (U. Buenos Aires)

August 13th, 2006
RTA'06 – Seattle

Motivation

- Understanding (ML-style) pattern-matching
 ⇒ Want for better granularity
- Filling some holes in the operational semantics

	Introduction	Elimination
Functions	Abstraction	Application
Constructed values	Constructor	Case analysis

What does it mean...

- ... to apply an argument to a constructor?
- ... to perform case analysis on an abstraction?

Typing arguments reject these constructions...

... but is it sufficient to deduce they are meaningless?

Case analysis on constant constructors (1/2)

To the λ -calculus we add:

- **Constructors** (i.e. constants)

true, false, 0, S, ..., c, d, ...

- **Case construct**

$$\{c_1 \mapsto M_1; \dots; c_n \mapsto M_n\}. N \quad (c_i \neq c_j \text{ if } i \neq j)$$

$$\equiv \text{match } N \text{ with } c_1 \rightarrow M_1 \mid \dots \mid c_n \rightarrow M_n \quad (\text{Caml})$$

- **Reduction rule**

$$(\text{CaseCons}) \quad \{c_1 \mapsto M_1; \dots; c_n \mapsto M_n\}. c_i \rightarrow M_i$$

Case analysis on constant constructors (2/2)

- **Example:** Booleans

$$\text{if} \equiv \lambda bxy. \{\text{true} \mapsto x; \text{false} \mapsto y\}. b$$
$$\begin{aligned} \text{if true } M_1 M_2 &\xrightarrow{*} \{\text{true} \mapsto M_1; \text{false} \mapsto M_2\}. \text{true} \\ &\xrightarrow{*} M_1 \end{aligned}$$
$$\text{if false } M_1 M_2 \xrightarrow{*} M_2$$

- How to add arguments to constructors?

- 1 Introduce variable binding in patterns

$$c(x_1, \dots, x_n) \mapsto M_1\{x_1; \dots; x_n\}$$

- 2 Go to next slide

Entering THE TWILIGHT ZONE ... (1/2)

- A bizarre reduction rule

$$(\text{CaseApp}) \quad \{\theta\}.(MN) \rightarrow (\{\theta\}.M)N$$

... that allows to write the predecessor function:

$$\text{pred} \equiv \lambda x. \{0 \mapsto 0; S \mapsto \lambda z. z\}. x$$

- Computational behaviour:

$$\text{pred } 0 \rightarrow_{\beta} \{0 \mapsto 0; S \mapsto \lambda z. z\}. 0 \xrightarrow{\text{CaseCons}} 0$$

$$\begin{aligned} \text{pred } (S M) &\rightarrow_{\beta} \{0 \mapsto 0; S \mapsto \lambda z. z\}. (S M) \\ &\rightarrow_{\text{CaseApp}} \{0 \mapsto 0; S \mapsto \lambda z. z\}. S M \\ &\rightarrow_{\text{CaseCons}} (\lambda z. z) M \\ &\rightarrow_{\beta} M \end{aligned}$$

Entering THE TWILIGHT ZONE ... (2/2)

- Reduction rule **CaseApp** goes against typing:

$$\begin{array}{ccc}
 \{\theta\}. (MN) & \rightarrow & (\{\theta\}. M)N \\
 \begin{array}{c} \nearrow \\ M \text{ is applied...} \\ \dots \text{ thus } M \text{ has an } \mathbf{arrow} \text{ type} \end{array} & & \begin{array}{c} \uparrow \\ M \text{ is pattern-matched} \\ \dots \text{ thus } M \text{ has a } \mathbf{sum} \text{ type} \end{array}
 \end{array}$$

- Intuitions:

- Case construct is a form of **head-linear substitution**
- Application and pattern-matching **commute**

$$\{\theta_n\}. \dots \{\theta_1\}. M \ N_1 \ \dots \ N_k$$

⇒ Evaluation contexts with **two independent stacks**

(Actually, case constructs can be merged using a rule CaseCase)

Syntax of $\lambda\mathcal{B}_c$ Terms (M, N) and case bindings (θ, ϕ)

M, N	$::=$	x	$ $	$\lambda x. M$	$ $	MN	(λ -calculus)
		\boxtimes					(daimon)
		c					(constructor)
		$\{\theta\}. M$					(case analysis)
θ, ϕ	$::=$	$c_1 \mapsto M_1; \dots;$		$c_n \mapsto M_n$			($c_i \neq c_j$ when $i \neq j$)

Composition of case bindings $(\theta \circ \phi)$:

$$\theta \circ \{c_i \mapsto M_i\}_{i=1..n} \equiv \{c_i \mapsto \{\theta\}. M_i\}_{i=1..n}$$

Note: $\theta \circ (\phi \circ \rho) \not\equiv (\theta \circ \phi) \circ \rho$ but $\theta \circ (\phi \circ \rho) \rightarrow_{\text{CaseCase}}^* (\theta \circ \phi) \circ \rho$

Reduction rules of $\lambda\mathcal{B}_C$

AppLam (β)	$(\lambda x . M)N \rightarrow M\{x := N\}$	
AppDai	$\boxtimes N \rightarrow \boxtimes$	
LamApp (η)	$\lambda x . Mx \rightarrow M$	$(x \notin FV(M))$
LamDai	$\lambda x . \boxtimes \rightarrow \boxtimes$	
CaseCons	$\{\theta\} . c \rightarrow M$	$((c \mapsto M) \in \theta)$
CaseDai	$\{\theta\} . \boxtimes \rightarrow \boxtimes$	
CaseApp	$\{\theta\} . (MN) \rightarrow (\{\theta\} . M)N$	
CaseLam	$\{\theta\} . \lambda x . M \rightarrow \lambda x . \{\theta\} . M$	$(x \notin FV(\theta))$
CaseCase	$\{\theta\} . \{\phi\} . M \rightarrow \{\theta \circ \phi\} . M$	

\mathcal{B}_C = union of all reduction rules but AppLam (β)

Lemma: The \mathcal{B}_C -calculus is SN

Encoding ML-style pattern-matching

- ML-style pattern-matching:

$$\text{match } N \text{ with } c(\vec{x}) \rightarrow M \mid \dots$$

becomes

$$\{c \mapsto \lambda \vec{x}. M; \dots\}. N$$

- Predecessor function

$$\text{pred} = \lambda x. \{0 \mapsto 0; S \mapsto \lambda z. z\}. x$$

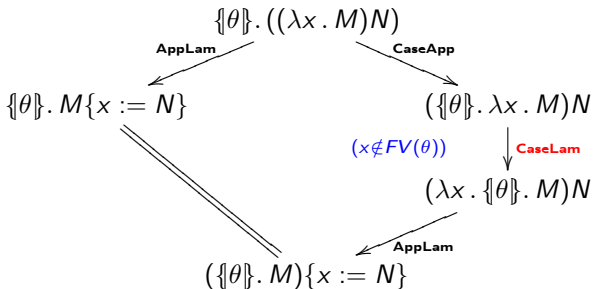
- Addition on unary integers:

$$\text{plus} = \mathbf{Y} (\lambda fxy. \{0 \mapsto y; S \mapsto \lambda z. S(f z y)\}. x)$$

- Variadic constructors: $c \bar{n} M_1 \dots M_n$

Critical pairs: an example

The critical pair **CaseApp/AppLam**



... is solved using the rule **CaseLam**:

$$(\text{CaseLam}) \quad \{\theta\}. \lambda x. M \quad \rightarrow \quad \lambda x. \{\theta\}. M \quad (x \notin FV(\theta))$$

The 13 critical pairs

The critical pair...	... is solved using
AppLam/LamApp	
AppLam/LamDai	AppDai
LamApp/AppLam	
LamApp/AppDai	LamDai
CaseApp/AppLam	AppLam, CaseLam
CaseApp/AppDai	AppDai, CaseDai
CaseLam/LamApp	LamApp, CaseApp
CaseLam/LamDai	LamDai, CaseDai
CaseCase/CaseCons	CaseCons
CaseCase/CaseDai	CaseDai
CaseCase/CaseApp	CaseCase, CaseApp
CaseCase/CaseLam	CaseCase, CaseLam
CaseCase/CaseCase	CaseCase

- 6 critical pairs introduce an **extra rule** \Rightarrow Closure conditions

Chasles criterion

$$6 \text{ closure conditions}^1 \left\{ \begin{array}{l} \text{AppLam, LamDai} \vdash \text{AppDai} \\ \text{LamApp, AppDai} \vdash \text{LamDai} \\ \text{CaseApp, AppLam} \vdash \text{CaseLam} \\ \text{CaseApp, AppDai} \vdash \text{CaseDai} \\ \text{CaseLam, LamApp} \vdash \text{CaseApp} \\ \text{CaseLam, LamDai} \vdash \text{CaseDai} \end{array} \right.$$

Theorem (Church-Rosser)

For all $2^9 = 512$ subsystems $s \subset \{\text{AppLam}; \dots; \text{CaseCase}\}$
the following are equivalent:

- 1 s fulfills the 6 closure conditions (248/512)
- 2 s is locally confluent
- 3 s is confluent (Church-Rosser)

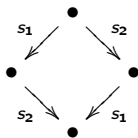
Note: The full system obviously fulfills the closure conditions!

¹ $r_1, r_2 \vdash r_3$ means: $r_1 \in s \wedge r_2 \in s \Rightarrow r_3 \in s$

Proof of Church-Rosser theorem

Consider the more general commutation problem for all pairs (s_1, s_2) of subsystems²

- Define **11 binary closure conditions (BCC)**
Analogous to 6 closure conditions, but for pairs (s_1, s_2)



Theorem (General commutation)

For all pairs of subsystems (s_1, s_2) the following are equivalent:

- 1 (s_1, s_2) fulfills the binary closure conditions
 - 2 s_1 and s_2 locally commute
 - 3 s_1 and s_2 commute
- $3 \Rightarrow 2 \Rightarrow 1$: Obvious
 - $1 \Rightarrow 2$: Induction on top term
 - $2 \Rightarrow 3$: **Divide and conquer technique** (next slide)

²There are $512 \times (512 + 1)/2 = 131\,328$ (unordered) pairs of subsystems

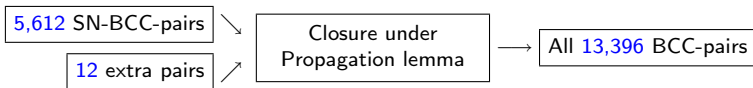
Divide and conquer technique (computer assisted)

- 131,328 pairs (s_1, s_2) (of subsystems)
- 13,396 BCC-pairs (s_1, s_2)
We know they **locally commute** \Rightarrow candidates for commutation
- 5,612 BCC-pairs (s_1, s_2) not involving AppLam ($= \beta$)
We know they **commute** (since local commut. + SN)
What about the 7,784 BCC-pairs involving AppLam?

Propagation lemma

If $s // s_1$ and $s // s_2$, then $s // (s_1 \cup s_2)$

- Reduces the whole problem to... 12 commutation lemmas



Observers

Contexts with one hole $C[\]$ are defined as usual

Evaluation contexts

$$E[\] ::= [\]N_1 \cdots N_n \mid \{\theta\}.\ [\]N_1 \cdots N_n$$

Daimon \boxtimes absorbs all evaluation contexts: $E[\boxtimes] \xrightarrow{*} \boxtimes$

Undefined terms

Terms of the form: $\lambda \vec{x}. (\{\theta\}. c)N_1 \cdots N_n$ ($c \notin \text{dom}(\theta)$)

Undefined terms absorb all evaluation contexts too!

Completely defined normal terms

Normal terms with no subterm of the form $\{\theta\}. c$ ($c \notin \text{dom}(\theta)$)

Separation theorem

Notions of separation

Two terms M_1 and M_2 are

- **Weakly separable** if there exists $C[]$ such that either
 - $C[M_1] \xrightarrow{*} \text{⊥}$ and $C[M_2] \xrightarrow{*} \text{undef}$; or
 - $C[M_1] \xrightarrow{*} \text{undef}$ and $C[M_2] \xrightarrow{*} \text{⊥}$.
- **Strongly separable** if there exists $C_1[]$ and $C_2[]$ such that
 - $C_1[M_1] \xrightarrow{*} \text{⊥}$ and $C_1[M_2] \xrightarrow{*} \text{undef}$; and
 - $C_2[M_1] \xrightarrow{*} \text{undef}$ and $C_2[M_2] \xrightarrow{*} \text{⊥}$.

Theorem (Separation)

Let M_1 and M_2 be completely defined normal terms.
If $M_1 \not\equiv M_2$, then M_1 and M_2 are **weakly separable**.

The proof of separation

Lots of technicalities, nothing really new...

(variations on the Böhm-out technique)

... but an interesting zoology:

$$\{\theta\}. \langle M_1; \dots; M_k; *_{n-k} \rangle \xrightarrow{*} \{\theta \mid M_1; \dots; M_k; *_{n-k}\}$$

Case + Tuple = Casuple

⇒ cf paper for details

Conclusion

Pattern-matching as a head-linear substitution

$$\begin{aligned}\{\theta\}.(MN) &\rightarrow (\{\theta\}.M)N \\ \{\theta\}.(\lambda x. M) &\rightarrow \lambda x. \{\theta\}.M\end{aligned}$$

is expressive and **computationally consistent**:

- Church-Rosser theorem
- Separation theorem

Future work:

- Confluence by “divide and conquer” for other systems?
- Which Böhm trees for $\lambda\mathcal{B}_c$?
- Which type system for $\lambda\mathcal{B}_c$? (**logical meaning**)
- Are other design choices possible?