A Polymorphic Type System for the λ-calculus with Constructors

Barbara Petit

LIP - ENS Lyon

June 6, 2009

Introduction

The $\lambda_{\mathcal{C}}$-calculus of Arbiser, Miquel & Rìos (2006):

- Encompasses ML-style pattern matching
- *Variadic* constructors
- Separation property

But unconventional operational semantics, not obviously corresponding to any kind of cut elimination.

Challenge:

Type it.

Introduction

The $\lambda_{\mathcal{C}}$-calculus of Arbiser, Miquel & Rìos (2006):

- Encompasses ML-style pattern matching
- *Variadic* constructors
- Separation property

But unconventional operational semantics, not obviously corresponding to any kind of cut elimination.

Challenge:

Type it.

Related Works:

- ρ-calculus (Cristea, Kirchner, Liquori)
- Pure Pattern Calculus (Jay, Kesner)

$\lambda_{\mathcal{C}}$-terms.

- λ-calculus: $x \mid \lambda x.t \mid tu$
- Constructors: $c, c_1, c_2 \ldots$, $\mathtt{true}$, 0, $\mathtt{succ}$, $\mathtt{none}$...

  ... constants with no fixed arity.
- Case bindings: $\theta = \{c_1 \mapsto u_1; \ldots; c_n \mapsto u_n\}$ $(c_i \neq c_j \text{ for } i \neq j)$
- Case constructs: $\{\!|\theta|\!\} \cdot t$

$\lambda_{\mathcal{C}}$-terms.

- λ-calculus: $x \mid \lambda x.t \mid tu$
- Constructors: $c, c_1, c_2 \ldots$, $\mathtt{true}$, 0, $\mathtt{succ}$, $\mathtt{none}$...

... constants with no fixed arity.

- Case bindings: $\theta = \{c_1 \mapsto u_1; \ldots; c_n \mapsto u_n\}$ $(c_i \neq c_j \text{ for } i \neq j)$
- Case constructs: $\{\!|\theta|\!\} \cdot t$

$$If := \lambda bxy.\{\!|\mathtt{true} \mapsto x; \mathtt{false} \mapsto y|\!\} \cdot b$$

```
let if =
  fun b -> fun x -> fun y ->
    match b with
      | true -> x
      | false -> y
```

Commutation Case/application

$$\{\!|\theta|\!\}.(tu) \to (\{\!|\theta|\!\}.t)u$$

$$pred := \lambda x.\{\!|0 \mapsto 0; \mathtt{succ} \mapsto \lambda y.y|\!\} \cdot x$$

Commutation Case/application

$$\{\!|\theta|\!\}.(tu) \to (\{\!|\theta|\!\}.t)u$$

$$pred := \lambda x.\{\!|0 \mapsto 0; \mathtt{succ} \mapsto \lambda y.y|\!\} \cdot x$$

$$pred(\mathtt{succ}\ \ p) \to \{\!|0 \mapsto 0; \mathtt{succ} \mapsto \lambda y.y|\!\} \cdot (\mathtt{succ}\ \ p) \qquad \textsc{AppLam}$$

Commutation Case/application

$$\{\!|\theta|\!\}.(tu) \to (\{\!|\theta|\!\}.t)u$$

$$pred := \lambda x.\{\!|0 \mapsto 0; \mathtt{succ} \mapsto \lambda y.y|\!\} \cdot x$$

$$\begin{array}{rll} pred(\mathtt{succ}\ \ p) & \to \{\!|0 \mapsto 0; \mathtt{succ} \mapsto \lambda y.y|\!\} \cdot (\mathtt{succ}\ \ p) & \textsc{AppLam} \\ & \to (\{\!|0 \mapsto 0; \mathtt{succ} \mapsto \lambda y.y|\!\} \cdot \mathtt{succ})\ p & \textsc{CaseApp} \end{array}$$

Commutation Case/application

$$\{\!|\theta|\!\}.(tu) \to (\{\!|\theta|\!\}.t)u$$

$$pred := \lambda x.\{\!|0 \mapsto 0; \mathtt{succ} \mapsto \lambda y.y|\!\} \cdot x$$

$$\begin{array}{rll} pred(\mathtt{succ}\ \ p) \to & \{\!|0 \mapsto 0; \mathtt{succ} \mapsto \lambda y.y|\!\} \cdot (\mathtt{succ}\ \ p) & \textsc{AppLam} \\ \to & (\{\!|0 \mapsto 0; \mathtt{succ} \mapsto \lambda y.y|\!\} \cdot \mathtt{succ})\ p & \textsc{CaseApp} \\ \to & (\lambda y.y)\ p & \textsc{CaseCons} \end{array}$$

Commutation Case/application

$$\{\!|\theta|\!\}.(tu) \to (\{\!|\theta|\!\}.t)u$$

$$pred := \lambda x.\{\!|0 \mapsto 0; \mathtt{succ} \mapsto \lambda y.y|\!\} \cdot x$$

$$\begin{array}{rll}
pred(\mathtt{succ}\ \ p) & \to \{\!|0 \mapsto 0; \mathtt{succ} \mapsto \lambda y.y|\!\} \cdot (\mathtt{succ}\ \ p) & \textsc{AppLam} \\
& \to (\{\!|0 \mapsto 0; \mathtt{succ} \mapsto \lambda y.y|\!\} \cdot \mathtt{succ})\ p & \textsc{CaseApp} \\
& \to (\lambda y.y)\ p & \textsc{CaseCons} \\
& \to p & \textsc{AppLam}
\end{array}$$

Reduction rules

AppLam	$(\lambda x.t)u$	$\rightarrow$	$t\{x \leftarrow u\}$	
LamApp	$\lambda x.tx$	$\rightarrow$	t	$(x \notin \mathcal{FV}(t))$
CaseCons	$\{\!\|\theta\|\!\} \cdot c$	$\rightarrow$	t	$((c \mapsto t) \in \theta)$

Reduction rules

AppLam	$(\lambda x.t)u$	$\rightarrow$	$t\{x \leftarrow u\}$	
LamApp	$\lambda x.tx$	$\rightarrow$	t	$(x \notin \mathcal{FV}(t))$
CaseCons	$\{\!\|\theta\|\!\} \cdot c$	$\rightarrow$	t	$((c \mapsto t) \in \theta)$
CaseApp	$\{\!\|\theta\|\!\} \cdot (tu)$	$\rightarrow$	$(\{\!\|\theta\|\!\} \cdot t)u$	

Reduction rules

$$\textsc{AppLam} \quad (\lambda x.t)u \to t\{x \leftarrow u\}$$

$$\textsc{LamApp} \quad \lambda x.tx \to t \qquad (x \notin \mathcal{FV}(t))$$

$$\textsc{CaseCons} \quad \{\!|\theta|\!\} \cdot c \to t \qquad ((c \mapsto t) \in \theta)$$

$$\textsc{CaseApp} \quad \{\!|\theta|\!\} \cdot (tu) \to (\{\!|\theta|\!\} \cdot t)u$$

$$\textsc{CaseLam} \quad \{\!|\theta|\!\} \cdot \lambda x.t \to \lambda x.\{\!|\theta|\!\} \cdot t \qquad (x \notin \mathcal{FV}(\theta))$$

for confluence

$$\textsc{CaseCase} \quad \{\!|\theta|\!\} \cdot (\{\!|\phi|\!\} \cdot t) \to \{\!|\theta \circ \phi|\!\} \cdot t$$

with $\theta \circ \{c_1 \mapsto t_1; \ldots; c_n \mapsto t_n\} := \{c_1 \mapsto \{\!|\theta|\!\} \cdot t_1; \ldots; c_n \mapsto \{\!|\theta|\!\} \cdot t_n\}$

for separation

A polymorphic type system

The type system includes system F with subtyping:

$$T, U \ := \quad X \mid T \to U \mid \forall X.T$$

$$\frac{-}{\Gamma \vdash x : T}(x:T)\in\Gamma \qquad \frac{\Gamma \vdash t : U \to T \quad \Gamma \vdash u : U}{\Gamma \vdash tu : T}$$

$$\frac{\Gamma, x : U \vdash t : T}{\Gamma \vdash \lambda x.t : U \to T} \qquad \frac{\Gamma \vdash t : T}{\Gamma \vdash t : \forall X.T} X \notin \mathcal{TV}(\Gamma)$$

A polymorphic type system

The type system includes system F with subtyping:

$$T, U \ := \quad X \mid T \to U \mid \forall X.T$$

$$\frac{-}{\Gamma \vdash x : T}(x:T)\in\Gamma \qquad \frac{\Gamma \vdash t : U \to T \quad \Gamma \vdash u : U}{\Gamma \vdash tu : T}$$

$$\frac{\Gamma, x : U \vdash t : T}{\Gamma \vdash \lambda x.t : U \to T} \qquad \frac{\Gamma \vdash t : T}{\Gamma \vdash t : \forall X.T} X \notin \mathcal{TV}(\Gamma) \qquad \frac{\Gamma \vdash t : U \quad U \preccurlyeq T}{\Gamma \vdash t : T}$$

$$T \preccurlyeq T \qquad \frac{T \preccurlyeq U \quad U \preccurlyeq U'}{T \preccurlyeq U'} \qquad \frac{U' \preccurlyeq U \quad T \preccurlyeq T'}{U \to T \preccurlyeq U' \to T'}$$

$$\frac{T \preccurlyeq U}{T \preccurlyeq \forall X.U} X \notin \mathcal{TV}(T) \qquad \frac{-}{\forall X.T \preccurlyeq T\{X \leftarrow U\}} \quad \dots$$

Typing data-structures

Example: Option type.

$$T\ opt \quad := \quad \textbf{none} \cup \textbf{some}\ T$$

$$u : T\ opt \quad \Rightarrow \quad (u = \texttt{none}) \text{ or } (u = \texttt{some}\ \ t) \text{ with } t : T$$

Typing data-structures

Example: Option type.

$$T\ opt \quad := \quad \textbf{none} \cup \textbf{some}\ T$$

$$u : T\ opt \quad \Rightarrow \quad (u = \texttt{none}) \text{ or } (u = \texttt{some}\ \ t) \text{ with } t : T$$

Data-types:

- A type **c** for every constructor c
- A *type application* for data-types:
  $(t_i : T_i)_{i=1}^{n} \quad \Rightarrow \quad c\,t_1 \ldots t_n : \mathbf{c}\,T_1 \ldots T_n$
- Union type

The problematic application type

- A type application to type applications?

$$\frac{\vdash t : T \qquad \vdash u : U}{\vdash tu : TU}$$

The problematic application type

- A type application to type applications?

$$\frac{\vdash t : \ T \qquad \vdash u : \ U}{\vdash tu : \ TU}$$

- $\delta := \lambda x.xx \qquad \Delta := \forall X.(X \to X) \to \forall X.(X \to X)$
  $\rightsquigarrow \quad \vdash \delta : \ \Delta$

The problematic application type

- A type application to type applications?

$$\frac{\vdash t : T \qquad \vdash u : U}{\vdash tu : TU}$$

- $\delta := \lambda x.xx \qquad \Delta := \forall X.(X \to X) \to \forall X.(X \to X)$

  $\rightsquigarrow \quad \vdash \delta : \Delta$

  $\rightsquigarrow \quad \vdash \delta\delta : \Delta\Delta$ ☹

- Restricted application type

Syntax of Types

$$\textit{DataTypes}: \quad D, E := \quad c \mid DT$$

$$\textit{Types}: \quad T, U := \quad D \mid T \to U$$

Two classes of types

Syntax of Types

$$\textit{DataTypes}: \quad D, E := \quad c \mid DT \mid D \cup E \mid D \cap E$$

$$\textit{Types}: \quad T, U := \quad D \mid T \to U \mid T \cup U \mid T \cap U$$

Union and intersection

Syntax of Types

$$\begin{array}{lll} \textit{DataTypes}: & D, E := & c \mid DT \mid D \cup E \mid D \cap E \\ & & \mid \forall X.D \\ \textit{Types}: & T, U := & X \mid D \mid T \to U \mid T \cup U \mid T \cap U \\ & & \mid \forall X.T \end{array}$$

Quantification

Syntax of Types

$$\begin{array}{lll} \textit{DataTypes}: & D, E := & \alpha \mid c \mid DT \mid D \cup E \mid D \cap E \\ & & \mid \forall X.D \mid \forall \alpha.D \\ \textit{Types}: & T, U := & X \mid D \mid T \to U \mid T \cup U \mid T \cap U \\ & & \mid \forall X.T \mid \forall \alpha.T \end{array}$$

Two spaces of type variables

Typing the case binding (I)

Example $nat \equiv \mathbf{0} \cup \mathbf{succ}\ nat$.

$$\theta_{pred} := \{\ 0 \ \mapsto\ 0\ ; \quad \texttt{succ} \ \mapsto\ \lambda y.y\ \}$$

$$\theta_{pred} :\ nat \to nat$$

Typing the case binding (I)

Example $nat \equiv \mathbf{0} \cup \mathbf{succ}\ nat$.

$$\theta_{pred} := \{\ 0 \ \mapsto\ 0\ ; \quad \texttt{succ} \ \mapsto\ \lambda y.y\ \}$$

$$\theta_{pred} :\ nat \to nat$$

$$\theta_{pred} :\ \begin{cases} \mathbf{0} \to nat \\ (\ \mathbf{succ}\ nat) \to nat \end{cases}$$

Typing the case binding (I)

Example $nat \equiv \mathbf{0} \cup \mathbf{succ}\ nat$.

$$\theta_{pred} := \{\ 0 \mapsto 0\ ;\quad \texttt{succ} \mapsto \lambda y.y\ \}$$

$$0 : nat \qquad\qquad \lambda y.y : nat \to nat$$

$$\theta_{pred} :\ nat \to nat$$

$$\theta_{pred} :\ \begin{cases} \mathbf{0} \to nat \\ (\ \mathbf{succ}\ nat) \to nat \end{cases}$$

Typing the case binding (I)

Example $nat \equiv \mathbf{0} \cup \mathbf{succ}\ nat$.

$$\theta_{pred} := \{\ 0 \mapsto 0\ ;\quad \texttt{succ} \mapsto \lambda y.y\ \}$$

$$0 : nat \qquad\qquad \lambda y.y : nat \to nat$$

$$\theta_{pred} : \ nat \to nat$$

$$\theta_{pred} : \begin{cases} \mathbf{0} \to nat \\ (\ \mathbf{succ}\ nat) \to nat \end{cases} \quad \hookrightarrow^{?} \quad \theta_{pred} : \big(\mathbf{0} \cup (\mathbf{succ}\ nat)\big) \to nat$$

Typing the case binding (II)

$$\frac{\Gamma \vdash u_i : \left(\vec{U_i} \to T_i\right)_{i=1}^{n}}{\Gamma \vdash \theta : \bigcap_i c_i \vec{U_i} \to T_i}$$

with $\theta = \{c_1 \mapsto u_1, \ldots, c_n \mapsto u_n\}$

Vectorial notation:

$$\vec{T}, \vec{U} := \emptyset \mid \vec{T}; U$$

$$c\emptyset = c \qquad c(\vec{T}; T) = (c\vec{T})T$$

$$\emptyset \to U = U \qquad (\vec{T}; T) \to U = \vec{T} \to (T \to U)$$

Typing the case binding (II)

$$\frac{\Gamma \vdash u_i : \left(\vec{U_i} \to T_i\right)_{i=1}^{n}}{\Gamma \vdash \theta : \bigcap_i c_i \vec{U_i} \to T_i}$$

with $\theta = \{c_1 \mapsto u_1, \ldots, c_n \mapsto u_n\}$

Vectorial notation:

$$\vec{T}, \vec{U} := \emptyset \mid \vec{T}; U$$

$$c\emptyset = c \qquad c(\vec{T}; T) = (c\vec{T})T$$

$$\emptyset \to U = U \qquad (\vec{T}; T) \to U = \vec{T} \to (T \to U)$$

$$(T_1 \to U_1) \cap (T_2 \to U_2) \preccurlyeq T_1 \cup T_2 \to U_1 \cup U_2$$

Typing the case construct

$$\frac{\Gamma \vdash t : \vec{U} \to T \qquad \Gamma \vdash \theta : T \to T'}{\Gamma \vdash \{\!|\theta|\!\} \cdot t : \vec{U} \to T'}$$

Typing the case construct

$$\frac{\Gamma \vdash t : \vec{U} \to T \qquad \Gamma \vdash \theta : T \to T'}{\Gamma \vdash \{\!|\theta|\!\} \cdot t : \vec{U} \to T'}$$

$$\dfrac{\dfrac{\dfrac{\vdash 0 : nat \qquad \vdash \lambda y.y : nat \to nat}{\vdash \theta_{pred} : \mathbf{0} \to nat \cap (\mathbf{succ}\ nat) \to nat} \qquad \mathbf{0} \to nat \cap (\mathbf{succ}\ nat) \to nat \preccurlyeq nat \to nat}{x : nat \vdash \theta_{pred} : nat \to nat} \qquad x : nat \vdash x : nat}{x : nat \vdash \{\!|\theta_{pred}|\!\} \cdot x : nat}$$

$$\frac{x : nat \vdash \{\!|\theta_{pred}|\!\} \cdot x : nat}{\vdash \lambda x.\{\!|\theta_{pred}|\!\} \cdot x : nat \to nat}$$

Typing data-structures

$\vdash \mathtt{c} : \mathbf{c}$

$\vdash \mathtt{0} : \mathbf{0}$

$\vdash \mathtt{succ} : \mathbf{succ}$

$\vdash \mathtt{succ\ 0} : \mathbf{succ}\ \mathit{nat}?$

Typing data-structures

$$\vdash \mathtt{c} : \mathbf{c} \qquad\qquad D \preccurlyeq T \to DT$$

$$\vdash \mathtt{0} : \mathbf{0} \qquad\qquad \vdash \mathtt{succ} : \mathbf{succ}$$
$$\vdash \mathtt{succ\ 0} : \mathbf{succ}\ nat$$

$$\dfrac{\dfrac{\vdash \mathtt{succ}\colon \mathbf{succ} \qquad \mathbf{succ} \preccurlyeq nat \to \mathbf{succ}\ nat}{\vdash \mathtt{succ}\colon nat \to \mathbf{succ}\ nat} \qquad \mathbf{succ}\ nat \preccurlyeq nat}{\vdash \mathtt{succ}\colon nat \to nat}$$

Typing data-structures

$$\vdash \mathtt{c} : \mathbf{c} \qquad\qquad D \preccurlyeq T \to DT$$

$$\vdash \mathtt{0} : \mathbf{0} \qquad \vdash \mathtt{succ} : \mathbf{succ}$$
$$\vdash \mathtt{succ\ 0} : \mathbf{succ}\ \mathit{nat}$$

$$\dfrac{\dfrac{\vdash \mathtt{succ}\colon \mathbf{succ} \qquad \mathbf{succ} \preccurlyeq \mathit{nat} \to \mathbf{succ}\ \mathit{nat}}{\vdash \mathtt{succ}\colon \mathit{nat} \to \mathbf{succ}\ \mathit{nat}} \qquad \mathbf{succ}\ \mathit{nat} \preccurlyeq \mathit{nat}}{\vdash \mathtt{succ}\colon \mathit{nat} \to \mathit{nat}}$$

+ usual intersection and union subtyping rules

1 The λ-calculus with constructors

2 Type system

3 Realisability semantic

Reducibility candidates

$$\vdash t : T \quad \rightsquigarrow \quad t \in [T] \qquad ([T] \in \mathcal{CR})$$

Reducibility candidates

$S \subseteq \Lambda_0$ is a reducibility candidate if:

- (CR1) Perfect normalisation: $S \subseteq PN_0$
- (CR2) Stability by reduction: $t \in S \Rightarrow Red_1(t) \subseteq S$
- (CR3) Stability by neutral expansion: If t is neutral, then $Red_1(t) \subseteq S \;\Rightarrow\; t \in S$

$$\vdash t : T \quad \rightsquigarrow \quad t \in [T] \qquad ([T] \in \mathcal{CR})$$

t defined: no subterm $\{\!|\theta|\!\} \cdot c$ with $c \notin dom(\theta)$

t perfectly normalising: t is SN and every reduct is defined

t value: $t = \lambda x.t_0$ or $t = ct_1 \dots t_k$

Interpretation of types

Arrow types:

$$[T \to U] = \{t \in \Lambda_0 / u \in [T] \Rightarrow tu \in [U]\}$$

Interpretation of types

Arrow types:

$$[T \to U] = \{t \in \Lambda_0 / u \in [T] \Rightarrow tu \in [U]\}$$

Data types:

$$\mathcal{D} \in \mathcal{CR} \text{ data-candidate:} \quad t \text{ value } \in \mathcal{D} \Rightarrow t = ct_1 \dots t_k$$

Interpretation of types

Arrow types:

$$[T \to U] = \{t \in \Lambda_0 / u \in [T] \Rightarrow tu \in [U]\}$$

Data types:

$$\mathcal{D} \in \mathcal{CR} \text{ data-candidate:} \quad t \text{ value } \in \mathcal{D} \Rightarrow t = ct_1 \dots t_k$$

Union types:

$$\mathcal{CR} \text{ stable by union ([Riba])}$$

Perfect normalisation

Under the rug: An intermediate calculus $\lambda_{\mathcal{C}}$'

$$\vdash t : T \quad \Rightarrow \quad \tilde{t} \in [T]$$

$$\tilde{t} \in PN_0 \quad \Rightarrow \quad t \in PN_0$$

Perfect normalisation

Under the rug: An intermediate calculus $\lambda_{\mathcal{C}}$'

$$\vdash t : T \quad \Rightarrow \quad \tilde{t} \in [T]$$

$$\tilde{t} \in PN_0 \quad \Rightarrow \quad t \in PN_0$$

Typed $\lambda_{\mathcal{C}}$ is perfectly normalising

Conclusion

Further work:

- Fixpoint operator
- KAM with a case stack?
- Decidable fragment?
- Logical interpretation of data types (CPS?)

Conclusion

Further work:

- Fixpoint operator
- KAM with a case stack?
- Decidable fragment?
- Logical interpretation of data types (CPS?)

Thank you!