

# A Polymorphic Type System for the $\lambda$ -calculus with Constructors

Barbara Petit

LIP - ENS Lyon

# Introduction

The  $\lambda_C$ -calculus of Arbiser, Miquel & Ríos (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.

## 1 The $\lambda$ -calculus with constructors

### 2 Type system

- Usual system  $F$  rules
- Type of case bindings
- Type of case constructs
- Type of data-structures

### 3 Realisability semantic

# The $\lambda_c$ -calculus.

- $\lambda$ -calculus:  $x \mid \lambda x. t \mid tu$
- a Daimon:  $\boxtimes$  (an “exit” constant)
- Constructors:  $c, c_1, c_2 \dots, \text{true}, 0, \text{succ}, \text{none} \dots$   
*... constants with no fixed arity.*
- Case bindings:  $\theta = \{c_1 \mapsto u_1; \dots; c_n \mapsto u_n\}$  ( $c_i \neq c_j$  for  $i \neq j$ )
- Case constructs:  $\{\theta\} \cdot t$

# Pattern matching (I)

Case binding + Constructor = Pattern matching

```
If :=  $\lambda bxy. \{ \text{true} \mapsto x ; \text{false} \mapsto y \} \cdot b$ 
```

In ML:

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

# Pattern matching (II)

What about

```
let pred =  
  fun n ->  
    match n with  
    | 0 -> 0  
    | S(p) -> p
```

Remember:

$$\theta = \{c_1 \mapsto u_1 ; \dots ; c_n \mapsto u_n\}$$

*constructors not applied in case bindings*

# Commutation Case/application

$$\{\theta\}.(tu) \rightarrow (\{\theta\}.t)u$$

*pred* :=  $\lambda x. \{ 0 \mapsto 0 ; \text{succ} \mapsto \lambda y.y \} \cdot x$

# Commutation Case/application

$$\{\theta\}.(tu) \rightarrow (\{\theta\}.t)u$$

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

$$pred(succ p) \rightarrow \{0 \mapsto 0; succ \mapsto \lambda y.y\} \cdot (succ p) \quad \text{LAMAPP}$$

# Commutation Case/application

$$\{\theta\}.(tu) \rightarrow (\{\theta\}.t)u$$

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

$$\begin{aligned} pred(succ\ p) &\rightarrow \{0 \mapsto 0; succ \mapsto \lambda y.y\} \cdot (succ\ p) && \text{LAMAPP} \\ &\rightarrow (\{0 \mapsto 0; succ \mapsto \lambda y.y\} \cdot succ)\ p && \text{CASEAPP} \end{aligned}$$

# Commutation Case/application

$$\{\theta\}.(tu) \rightarrow (\{\theta\}.t)u$$

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

$$\begin{aligned} pred(succ\ p) &\rightarrow \{0 \mapsto 0; succ \mapsto \lambda y.y\} \cdot (succ\ p) && \text{LAMAPP} \\ &\rightarrow (\{0 \mapsto 0; succ \mapsto \lambda y.y\} \cdot succ)\ p && \text{CASEAPP} \\ &\rightarrow (\lambda y.y)\ p && \text{CASECONS} \end{aligned}$$

# Commutation Case/application

$$\{\theta\}.(tu) \rightarrow (\{\theta\}.t)u$$

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

$$\begin{aligned} pred(succ\ p) &\rightarrow \{0 \mapsto 0; succ \mapsto \lambda y.y\} \cdot (succ\ p) && \text{LAMAPP} \\ &\rightarrow (\{0 \mapsto 0; succ \mapsto \lambda y.y\} \cdot succ)\ p && \text{CASEAPP} \\ &\rightarrow (\lambda y.y)\ p && \text{CASECONS} \\ &\rightarrow p && \text{LAMAPP} \end{aligned}$$

# Reduction rules

$$\text{APPLAM} \quad (\lambda x.t)u \rightarrow t\{x \leftarrow u\}$$

$$\text{LAMAPP} \quad \lambda x.tx \rightarrow t \quad (x \notin \mathcal{FV}(t))$$

$$\text{CASECONS} \quad \{\theta\} \cdot c \rightarrow t \quad ((c \mapsto t) \in \theta)$$

# Reduction rules

$$\text{APPLAM} \quad (\lambda x.t)u \rightarrow t\{x \leftarrow u\}$$

$$\text{LAMAPP} \quad \lambda x.tx \rightarrow t \quad (x \notin \mathcal{FV}(t))$$

$$\text{CASECONS} \quad \{\theta\} \cdot c \rightarrow t \quad ((c \mapsto t) \in \theta)$$

$$\text{CASEAPP} \quad \{\theta\} \cdot (tu) \rightarrow (\{\theta\} \cdot t)u$$

# Reduction rules

$$\text{APPLAM} \quad (\lambda x.t)u \rightarrow t\{x \leftarrow u\}$$

$$\text{LAMAPP} \quad \lambda x.tx \rightarrow t \quad (x \notin \mathcal{FV}(t))$$

$$\text{CASECONS} \quad \{\theta\} \cdot c \rightarrow t \quad ((c \mapsto t) \in \theta)$$

$$\text{CASEAPP} \quad \{\theta\} \cdot (tu) \rightarrow (\{\theta\} \cdot t)u$$

$$\text{CASELAM} \quad \{\theta\} \cdot \lambda x.t \rightarrow \lambda x.\{\theta\} \cdot t \quad (x \notin \mathcal{FV}(\theta))$$

*for confluence*

$$\text{CASECASE} \quad \{\theta\} \cdot (\{\phi\} \cdot t) \rightarrow \{\theta \circ \phi\} \cdot t$$

$$\text{with } \theta \circ \{c_1 \mapsto t_1; \dots; c_n \mapsto t_n\} := \{c_1 \mapsto \{\theta\} \cdot t_1; \dots; c_n \mapsto \{\theta\} \cdot t_n\}$$

*for separation*

# Reduction rules

$$\text{APPLAM} \quad (\lambda x.t)u \rightarrow t\{x \leftarrow u\}$$

$$\text{APPDAI} \quad \boxtimes u \rightarrow \boxtimes$$

$$\text{LAMAPP} \quad \lambda x.tx \rightarrow t \quad (x \notin \mathcal{FV}(t))$$

$$\text{LAMDAI} \quad \lambda x.\boxtimes \rightarrow \boxtimes$$

$$\text{CASECONS} \quad \{\theta\} \cdot c \rightarrow t \quad ((c \mapsto t) \in \theta)$$

$$\text{CASEDAI} \quad \{\theta\} \cdot \boxtimes \rightarrow \boxtimes$$

$$\text{CASEAPP} \quad \{\theta\} \cdot (tu) \rightarrow (\{\theta\} \cdot t)u$$

$$\text{CASELAM} \quad \{\theta\} \cdot \lambda x.t \rightarrow \lambda x.\{\theta\} \cdot t \quad (x \notin \mathcal{FV}(\theta))$$

*for confluence*

$$\text{CASECASE} \quad \{\theta\} \cdot (\{\phi\} \cdot t) \rightarrow \{\theta \circ \phi\} \cdot t$$

$$\text{with } \theta \circ \{c_1 \mapsto t_1; \dots; c_n \mapsto t_n\} := \{c_1 \mapsto \{\theta\} \cdot t_1; \dots; c_n \mapsto \{\theta\} \cdot t_n\}$$

*for separation*

# Pattern matching as head linear substitution

Case analysis on the head-position subterm...

$$t \quad := \quad \{\theta\} \cdot (\lambda x_1 \dots x_n. y \ u_1 \dots u_k)$$

# Pattern matching as head linear substitution

Case analysis on the head-position subterm...

$$\begin{aligned} t & := \{\theta\} \cdot (\lambda x_1 \dots x_n. y u_1 \dots u_k) \\ & \rightarrow^* \lambda x_1 \dots x_n. \{\theta\} \cdot (y u_1 \dots u_k) \end{aligned}$$

# Pattern matching as head linear substitution

Case analysis on the head-position subterm...

$$\begin{aligned} t & := \{\theta\} \cdot (\lambda x_1 \dots x_n. y \ u_1 \dots u_k) \\ & \rightarrow^* \lambda x_1 \dots x_n. \{\theta\} \cdot (y \ u_1 \dots u_k) \\ & \rightarrow^* \lambda x_1 \dots x_n. (\{\theta\} \cdot y) \ u_1 \dots u_k \end{aligned}$$

# Pattern matching as head linear substitution

Case analysis on the head-position subterm...

$$\begin{aligned} t & := \{\theta\} \cdot (\lambda x_1 \dots x_n. y \ u_1 \dots u_k) \\ & \rightarrow^* \lambda x_1 \dots x_n. \{\theta\} \cdot (y \ u_1 \dots u_k) \\ & \rightarrow^* \lambda x_1 \dots x_n. (\{\theta\} \cdot y) \ u_1 \dots u_k \end{aligned}$$

... which might be not so intuitive:

$$\{\theta\} \cdot (f \ 42) \rightarrow (\{\theta\} \cdot f) \ 42$$

# Difficulties

$$\{\theta\} \cdot (tu) \equiv (\{\theta\}.t) u$$

# Difficulties

$$\{\theta\} \cdot (tu) \equiv (\{\theta\}.t) u$$

$t$  : Arrow type

# Difficulties

$$\{\theta\} \cdot (tu) \equiv (\{\theta\}.t) u$$

$t$  : Arrow type       $t$  : Sum type

# Difficulties

$$\{\theta\} \cdot (tu) \equiv (\{\theta\}.t) u$$

$t$  : Arrow type       $t$  : Sum type

Solution:

$$a + b \cong X \rightarrow ( a(X) + b(X) )$$

*In fact  $a(X) \cup b(X)$*

## 1 The $\lambda$ -calculus with constructors

## 2 Type system

- Usual system  $F$  rules
- Type of case bindings
- Type of case constructs
- Type of data-structures

## 3 Realisability semantic

# A polymorphic type system...

The type system includes system  $F$ :

$$T, U := X \mid T \rightarrow U \mid \forall X.T$$

$$\frac{}{\Gamma \vdash x : T}^{(x:T) \in \Gamma} \quad \frac{}{\Gamma \vdash \lambda : T}$$

$$\frac{\Gamma \vdash t : U \rightarrow T \quad \Gamma \vdash u : U}{\Gamma \vdash tu : T}$$

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

# A polymorphic type system with subtyping

... and even  $F_{\preccurlyeq}$ :

$$T, U := X \mid T \rightarrow U \mid \forall X. T$$

$$\frac{\Gamma \vdash t : U \quad U \preccurlyeq T}{\Gamma \vdash t : T}$$

$$\frac{-}{T \preccurlyeq T} \quad \frac{T \preccurlyeq U \quad U \preccurlyeq U'}{T \preccurlyeq U'} \quad \frac{U' \preccurlyeq U \quad T \preccurlyeq T'}{U \rightarrow T \preccurlyeq U' \rightarrow T'}$$

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

# Typing data-structures

*Example:* Option type.

$$?T \quad := \quad \mathbf{none} \cup \mathbf{some} \ T$$

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

# Typing data-structures

*Example:* Option type.

$$?T \quad := \quad \mathbf{none} \cup \mathbf{some} \ T$$

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

*Data-types:*

- A type  $\mathbf{c}$  for every constructor  $c$
- A *type application* for data-types:

$$(t_i : T_i)_{i=1}^n \quad \Rightarrow \quad \mathbf{c}t_1 \dots t_n : \mathbf{c}T_1 \dots T_n$$

- Union type (for algebraic types)

# The problematic application type

- A type application to type applications?

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

# The problematic application type

- A type application to type applications?

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

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

# The problematic application type

- A type application to type applications?

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

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

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

$$\rightsquigarrow \vdash \delta\delta : \Delta\Delta \quad \text{☹️}$$

- Restricted application type

# Syntax of Types

*DataTypes* :  $D, E := c \mid DT$

*Types* :  $T, U := D \mid T \rightarrow U$

Two classes of types

# Syntax of Types

*DataTypes* :  $D, E := c \mid DT \mid D \cup E \mid D \cap E$

*Types* :  $T, U := D \mid T \rightarrow U \mid T \cup U \mid T \cap U$

Union and intersection

# Syntax of Types

*DataTypes* :  $D, E := c \mid DT \mid D \cup E \mid D \cap E$   
 $\mid \forall X. D \mid \exists X. D$

*Types* :  $T, U := X \mid D \mid T \rightarrow U \mid T \cup U \mid T \cap U$   
 $\mid \forall X. T \mid \exists X. T$

## Quantification

# Syntax of Types

*DataTypes* :  $D, E := \alpha \mid c \mid DT \mid D \cup E \mid D \cap E$   
 $\mid \forall X.D \mid \exists X.D \mid \forall \alpha.D \mid \exists \alpha.D$

*Types* :  $T, U := X \mid D \mid T \rightarrow U \mid T \cup U \mid T \cap U$   
 $\mid \forall X.T \mid \exists X.T \mid \forall \alpha.T \mid \exists \alpha.T$

Two spaces of type variables

# Typing the case binding: an example

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

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

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

# Typing the case binding: an example

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

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

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

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

# Typing the case binding: an example

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

$$\theta_{pred} := \{ \mathbf{0} \mapsto \mathbf{0}; \text{succ} \mapsto \lambda y.y \}$$

$$\mathbf{0} : nat \qquad \lambda y.y : nat \rightarrow nat$$

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

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

# Typing the case binding: informal rules

$$\vdash 0 : \mathit{nat} \quad \vdash \lambda y.y : \mathit{nat} \rightarrow \mathit{nat}$$

---

$$\vdash \{0 \mapsto 0 ; \mathit{succ} \mapsto \lambda y.y\} : (\mathbf{0} \rightarrow \mathit{nat}) \cap ( (\mathbf{succ} \ \mathit{nat}) \rightarrow \mathit{nat} )$$

# Typing the case binding: informal rules

$$\vdash 0 : \mathit{nat} \quad \vdash \lambda y.y : \mathit{nat} \rightarrow \mathit{nat}$$
$$\vdash \{0 \mapsto 0 ; \mathit{succ} \mapsto \lambda y.y\} : (\mathbf{0} \rightarrow \mathit{nat}) \cap ((\mathbf{succ} \ \mathit{nat}) \rightarrow \mathit{nat})$$

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

# Typing the case binding: informal rules

$$\frac{\vdash 0 : \mathit{nat} \quad \vdash \lambda y.y : \mathit{nat} \rightarrow \mathit{nat}}{\vdash \{0 \mapsto 0 ; \mathit{succ} \mapsto \lambda y.y\} : (\mathbf{0} \rightarrow \mathit{nat}) \cap ((\mathbf{succ} \mathit{nat}) \rightarrow \mathit{nat})}$$

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

$$(\mathbf{0} \rightarrow \mathit{nat}) \cap ((\mathbf{succ} \mathit{nat}) \rightarrow \mathit{nat}) \preccurlyeq (\mathbf{0} \cup (\mathbf{succ} \mathit{nat})) \rightarrow \mathit{nat}$$

# Typing the case binding: formal rules

$$\frac{\Gamma \vdash u_i : (\vec{U}_i \rightarrow T_i)_{i=1}^n}{\Gamma \vdash \theta : \bigcap_i (\mathbf{c}_i \vec{U}_i \rightarrow T_i)}$$

with  $\theta = \{c_1 \mapsto u_1; \dots; c_n \mapsto u_n\}$

Vectorial notation:

$$\begin{aligned} \mathbf{c} \vec{T} &:= \mathbf{c} T_1 \dots T_n \\ \vec{T} \rightarrow U &:= T_1 \rightarrow \dots \rightarrow T_n \rightarrow U \end{aligned}$$

# Typing the case binding: formal rules

$$(T_1 \rightarrow U_1) \cap (T_2 \rightarrow U_2) \rightsquigarrow (T_1 \cup T_2) \rightarrow (U_1 \cup U_2)$$

# Typing the case binding: formal rules

$$(T_1 \rightarrow U_1) \cap (T_2 \rightarrow U_2) \approx (T_1 \cup T_2) \rightarrow (U_1 \cup U_2)$$

Note:

$$(T_1 \rightarrow U_1) \cap (T_2 \rightarrow U_2) \approx (T_1 \cap T_2) \rightarrow (U_1 \cap U_2)$$

also holds...

## Remark: Non-uniqueness of typing

$$\theta_{pred} = \{ 0 \mapsto 0; \text{succ} \mapsto \lambda y.y \}$$

$$\frac{\vdash 0 : nat \quad \vdash \text{succ} : nat \rightarrow nat}{\vdash \theta_{pred} : (\mathbf{0} \rightarrow nat) \cap ((\mathbf{succ} \text{ nat}) \rightarrow nat)}$$

The diagram shows two arrows originating from the top expression. The left arrow points to the expression  $\vdash \theta_{pred} : (\mathbf{0} \rightarrow nat) \cap ((\mathbf{succ} \text{ nat}) \rightarrow nat)$ . The right arrow points to the expression  $\vdash \theta_{pred} : (\mathbf{0} \rightarrow nat) \cap (\mathbf{succ} \rightarrow (nat \rightarrow nat))$ .

$$\vdash \theta_{pred} : (\mathbf{0} \rightarrow nat) \cap (\mathbf{succ} \rightarrow (nat \rightarrow nat))$$

# Typing the case construct (I)

A first attempt:

$$\frac{\Gamma \vdash t : T \quad \Gamma \vdash \theta : T \rightarrow T'}{\Gamma \vdash \{\theta\} \cdot t : T'}$$

# Typing the case construct (I)

A first attempt:

$$\frac{\Gamma \vdash t : T \quad \Gamma \vdash \theta : T \rightarrow T'}{\Gamma \vdash \{\theta\} \cdot t : T'}$$

$$\frac{\frac{\frac{\vdash 0 : \mathbf{nat} \quad \vdash \lambda y. y : \mathbf{nat} \rightarrow \mathbf{nat}}{\vdash \theta_{pred} : \mathbf{0} \rightarrow \mathbf{nat} \cap (\mathbf{succ} \ \mathbf{nat}) \rightarrow \mathbf{nat}} \quad \mathbf{0} \rightarrow \mathbf{nat} \cap (\mathbf{succ} \ \mathbf{nat}) \rightarrow \mathbf{nat} \preccurlyeq \mathbf{nat} \rightarrow \mathbf{nat}}{\mathbf{x} : \mathbf{nat} \vdash \theta_{pred} : \mathbf{nat} \rightarrow \mathbf{nat}} \quad \mathbf{x} : \mathbf{nat} \vdash \mathbf{x} : \mathbf{nat}}}{\mathbf{x} : \mathbf{nat} \vdash \{\theta_{pred}\} \cdot \mathbf{x} : \mathbf{nat}}}{\vdash \lambda \mathbf{x}. \{\theta_{pred}\} \cdot \mathbf{x} : \mathbf{nat} \rightarrow \mathbf{nat}}$$

# Typing the case construct (I)

A first attempt:

$$\frac{\Gamma \vdash t : T \quad \Gamma \vdash \theta : T \rightarrow T'}{\Gamma \vdash \{\theta\} \cdot t : T'}$$

$$\frac{\frac{\frac{\vdash 0 : \mathbf{nat} \quad \vdash \lambda y. y : \mathbf{nat} \rightarrow \mathbf{nat}}{\vdash \theta_{pred} : \mathbf{0} \rightarrow \mathbf{nat} \cap (\mathbf{succ} \ \mathbf{nat}) \rightarrow \mathbf{nat}} \quad \mathbf{0} \rightarrow \mathbf{nat} \cap (\mathbf{succ} \ \mathbf{nat}) \rightarrow \mathbf{nat} \preceq \mathbf{nat} \rightarrow \mathbf{nat}}{x : \mathbf{nat} \vdash \theta_{pred} : \mathbf{nat} \rightarrow \mathbf{nat}} \quad x : \mathbf{nat} \vdash x : \mathbf{nat}}{\frac{x : \mathbf{nat} \vdash \{\theta_{pred}\} \cdot x : \mathbf{nat}}{\vdash \lambda x. \{\theta_{pred}\} \cdot x : \mathbf{nat} \rightarrow \mathbf{nat}}}$$

What about typing  $\{c \mapsto c'\} \cdot (\lambda x. c)$  ?

## Typing the case construct (II)

The complete rule:

$$\frac{\Gamma \vdash t : \vec{U} \rightarrow T \quad \Gamma \vdash \theta : T \rightarrow T'}{\Gamma \vdash \{\theta\} \cdot t : \vec{U} \rightarrow T'}$$

# Typing the case construct (II)

The complete rule:

$$\frac{\Gamma \vdash t : \vec{U} \rightarrow T \quad \Gamma \vdash \theta : T \rightarrow T'}{\Gamma \vdash \{\theta\} \cdot t : \vec{U} \rightarrow T'}$$

$$\underbrace{\{\text{none} \mapsto \boxtimes; \text{some} \mapsto \lambda y.y\}}_{\theta_{opt}: ?T \rightarrow T} \cdot (\lambda x.\text{some } x)$$

$$\frac{\frac{\vdash \boxtimes : T \quad \vdash \lambda y.y : T \rightarrow T}{\vdash \theta_{opt} : \text{none} \rightarrow T \cap (\text{some } T) \rightarrow T} \quad \text{none} \rightarrow T \cap (\text{some } T) \rightarrow T}{\leqslant ?T \rightarrow T}$$

$$\frac{\vdash \lambda x.\text{some } x : T \rightarrow ?T \quad \vdash \theta_{pred} : ?T \rightarrow T}{\vdash \{\theta_{opt}\} \cdot \lambda x.\text{some } x : T \rightarrow T}$$

# Typing data-structures

$\vdash c : \mathbf{c}$

$\vdash 0 : \mathbf{0}$

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

$\vdash \text{succ } 0 : \mathbf{succ } \text{nat?}$

# Typing data-structures

$$\vdash c : \mathbf{c} \qquad D \preccurlyeq T \rightarrow DT$$
$$\begin{array}{l} \vdash 0 : \mathbf{0} \qquad \vdash \text{succ} : \mathbf{succ} \\ \vdash \text{succ } 0 : \mathbf{succ } \mathit{nat} \end{array}$$
$$\frac{\frac{\vdash \text{succ} : \mathbf{succ} \qquad \mathbf{succ} \preccurlyeq \mathit{nat} \rightarrow \mathbf{succ } \mathit{nat}}{\vdash \text{succ} : \mathit{nat} \rightarrow \mathbf{succ } \mathit{nat}} \qquad \mathbf{succ } \mathit{nat} \preccurlyeq \mathit{nat}}{\vdash \text{succ} : \mathit{nat} \rightarrow \mathit{nat}}$$

## 1 The $\lambda$ -calculus with constructors

## 2 Type system

- Usual system  $F$  rules
- Type of case bindings
- Type of case constructs
- Type of data-structures

## 3 Realisability semantic

# Goals

Properties we want from the model:

- Strong normalisation

# Goals

Properties we want from the model:

- Strong normalisation
- No match failure

~~$\{\theta\} \cdot c$  with  $c \notin \text{dom}(\theta)$~~

# Goals

Properties we want from the model:

- Strong normalisation
- No match failure
- An alternative to subject reduction

# Goals

Properties we want from the model:

- Strong normalisation
- No match failure
- An alternative to subject reduction

} Perfect normalisation

# Values of $\lambda_c$

$t$  value  $:=$   $t = \lambda x.u \dots$

# Values of $\lambda_c$

$t$  value  $:=$   $t = \lambda x.u$  or  $t = c u_1 \dots u_k$

# Values of $\lambda_c$

$t$  value  $:=$   $t = \lambda x.u$  or  $t = c u_1 \dots u_k$

Proposition:

If  $t$  is normal with no match failure, then  $t$  is a value  
(or eventually the *daimon*)

# Reducibility candidates

$$\vdash t : T \quad \rightsquigarrow \quad t \in [T] \quad ( [T] \in CR )$$

# 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  $Red_1(t) \subseteq S \Rightarrow t \in S$

$$\vdash t : T \quad \rightsquigarrow \quad t \in [T] \quad ( [T] \in CR )$$

$t$  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 (I)

Arrow types:

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

validates  $(T_1 \rightarrow U_1) \cap (T_2 \rightarrow U_2) \preceq T_1 \cup T_2 \rightarrow U_1 \cup U_2$

# Interpretation of types (I)

Arrow types:

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

validates  $(T_1 \rightarrow U_1) \cap (T_2 \rightarrow U_2) \preceq T_1 \cup T_2 \rightarrow U_1 \cup U_2$

Data types:

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

$$[\mathbf{c} T_1 \dots T_k] = \mathbf{c} [T_1] \dots [T_k] \text{ closed by (CR3)}$$

# Interpretation of types (II)

Intersection types:

$$[T \cap U] = [T] \cap [U]$$

Stability of  $CR$  by intersection standard

# Interpretation of types (II)

Intersection types:

$$[T \cap U] = [T] \cap [U]$$

Stability of  $CR$  by intersection standard

Union types:

$$[T \cup U] = [T] \cup [U]$$

Stability of  $CR$  by union **in this calculus** ([Riba])

# Perfect normalisation

Under the rug: An intermediate calculus  $\lambda_c'$

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

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

# Perfect normalisation

Under the rug: An intermediate calculus  $\lambda_c'$

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

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

Typed  $\lambda_c$  is perfectly normalising

# Correctness of typing w.r.t reduction

Not (?) subject reduction but...

$$Bool \quad := \quad \mathbf{true} \cup \mathbf{false}$$
$$\vdash t : Bool \quad \Rightarrow \quad \tilde{t} \in [Bool] \quad \Rightarrow \quad t \rightarrow^* \text{true or false}$$

and  $\vdash \mathbf{true} : Bool, \quad \vdash \mathbf{false} : Bool$

# Correctness of typing w.r.t reduction

Not (?) subject reduction but...

$$Bool := \mathbf{true} \cup \mathbf{false}$$
$$\vdash t : Bool \Rightarrow \tilde{t} \in [Bool] \Rightarrow t \rightarrow^* \mathbf{true} \text{ or } \mathbf{false}$$

and  $\vdash \mathbf{true} : Bool, \vdash \mathbf{false} : Bool$

$$Nat := \mathbf{0} \cup (\mathbf{succ} \ Nat)$$
$$\vdash t : Nat \Rightarrow \tilde{t} \in [Nat] \Rightarrow t \rightarrow^* \mathbf{succ}^n \mathbf{0}$$

and  $\vdash \mathbf{succ}^n \mathbf{0} : Nat$

# Conclusion

Further work:

- Are all  $PN_0$  terms typable?
- Fixpoint operator
- KAM with a case stack?
- Decidable fragment?
- Logical interpretation of data types (CPS?)

# Conclusion

Further work:

- Are all  $PN_0$  terms typable?
- Fixpoint operator
- KAM with a case stack?
- Decidable fragment?
- Logical interpretation of data types (CPS?)

Thank you!