

FLOPS 2018

# Formal Verification of the Correspondence between Call-by-Need and Call-by-Name

Masayuki Mizuno and Eijiro Sumii

Tohoku University

May 9, 2018

## Motivation: a gap between abstraction and implementations of non-strict languages

- Call-by-name [Abramsky 1990 etc.]:  
(high-level) abstraction of non-strict languages
- Call-by-need [Wadsworth 1971 etc.]:  
implementations of non-strict languages

Our goal is mechanized verification of  
their correspondence

## Background 1: full- $\beta$ reduction

- Reduction is non-deterministic

- $(\lambda xy. y) \Omega$   $\xrightarrow{\beta} \lambda y. y$
- $(\lambda xy. y) \underline{\underline{\Omega}}$   $\xrightarrow{\beta} (\lambda xy. y) \Omega$

$$\Omega = (\lambda x. xx) (\lambda x. xx)$$

## Background 2: call-by-name

### Definition (call-by-name)

$$E_n ::= [] \mid E_n \ M$$

$$E_n[(\lambda x.M)N] \xrightarrow{\text{name}} E_n[M[x \mapsto N]]$$

■ If  $M \xrightarrow{\beta} \lambda x.N$ , then  $M \xrightarrow{\text{name}} \lambda x.N'$

$$(\lambda xy. y) \Omega \xrightarrow{\text{name}} \lambda y. y$$

## Problem: Redundant reductions

$$\begin{aligned} & (\lambda x. xx) \ \underline{(I \ I)} \\ & \xrightarrow{\beta} \underline{(\lambda x. xx) \ I} \\ & \xrightarrow{\beta} \underline{I \ I} \\ & \xrightarrow{\beta} I \end{aligned}$$

$$\begin{aligned} & \underline{(\lambda x. xx) \ (I \ I)} \\ & \xrightarrow{\text{name}} \underline{I \ I} \ (I \ I) \\ & \xrightarrow{\text{name}} \underline{I \ (I \ I)} \\ & \xrightarrow{\text{name}} \underline{I \ I} \\ & \xrightarrow{\text{name}} I \end{aligned}$$

$$I = \lambda x. x$$

## Background 3: call-by-need

- Reuse evaluation

$$\begin{aligned} & (\lambda x. x x) (I I) \\ & \xrightarrow{\text{need}} \text{let } x = \underline{I I} \text{ in } x x \\ & \xrightarrow{\text{need}} \underline{\text{let } x = I \text{ in } x x} \\ & \xrightarrow{\text{need}} \text{let } x = I \text{ in } \underline{I x} \end{aligned}$$

- Should correspond with call-by-name

## Our contributions

- Formalization of call-by-need  $\lambda$ -calculus [Ariola+ 1995] in the Coq proof assistant
- **Simplified proof** of correspondence with call-by-name, and verification in Coq
  - using standardization theorem [Curry&Feys 1958]

# Outline

- ➊ Call-by-name and call-by-need  $\lambda$ -calculi
- ➋ Simplified proof of the correspondence
- ➌ Coq formalization
- ➍ Conclusion



# Outline

- 1 Call-by-name and call-by-need  $\lambda$ -calculi
- 2 Simplified proof of the correspondence
- 3 Coq formalization
- 4 Conclusion

## Call-by-name $\lambda$ -calculus

Terms	$L, M, N$	$::=$	$x \mid V \mid M N$
Values (WHNF)	$V$	$::=$	$\lambda x.M$
Evaluation contexts	$E_n$	$::=$	$[] \mid E_n M$

$$\frac{M \rightarrow N}{E_n[M] \xrightarrow{\text{name}} E_n[N]}$$

$$(\beta) \quad (\lambda x.M)N \rightarrow M[x \mapsto N]$$

- Reduction is deterministic
- All stuck states are of the form  $E_n[x]$

## Lemma (determinacy of call-by-name reduction)

- $\xrightarrow{\text{name}}$  is partial function
- If  $E_n[x] = E'_n[y]$  then  $x = y$
- For any term  $M$ , **exactly one** of the following holds:
  1.  $M$  is a value
  2.  $M = E_n[x]$  for some  $E_n$  and  $x$
  3.  $M \xrightarrow{\text{name}} N$  for some  $N$

## Standardization theorem [Curry&Feys 1958]

### Definition (standard reduction sequence)

A reduction sequence

$M_1 \xrightarrow[\Delta_1]{\beta} M_2 \xrightarrow[\Delta_2]{\beta} \dots \xrightarrow[\Delta_{n-1}]{\beta} M_n$  is *standard* if every  $\Delta_i$  is outer and later than  $\Delta_{i+1}$

### Theorem (standardization)

If  $M \xrightarrow{\beta} N$ , then there is a standard reduction sequence from  $M$  to  $N$

## Corollaries

Corollary (termination of  $\xrightarrow{\text{name}}$ )

If  $M \xrightarrow{\beta} V$  then,  $M \xrightarrow{\text{name}} V'$  for some  $N$

Corollary (termination of  $\xrightarrow{\text{name}} \circ \xrightarrow{\beta}$ )

If  $M \xrightarrow{\beta} V$ , then  $M$  is terminating by  
 $\xrightarrow{\text{name}} \circ \xrightarrow{\beta}$

- Used for our proof of the correspondence with call-by-need

# Call-by-need $\lambda$ -calculus [Ariola+ 1995]

Terms  $M, N ::= x \mid V \mid M N \mid \text{let } x = M \text{ in } N$

Values  $V ::= \lambda x. M$

Answers  $A ::= V \mid \text{let } x = M \text{ in } A$

Evalctx  $E, E' ::= [] \mid E M \mid \text{let } x = M \text{ in } E$   
 $\mid \text{let } x = E \text{ in } E'[x]$

(I)  $(\lambda x. M)N \rightarrow \text{let } x = N \text{ in } M$

(V)  $\text{let } x = V \text{ in } E[x] \rightarrow \text{let } x = V \text{ in } E[V]$

(C)  $(\text{let } x = M \text{ in } A) N \rightarrow \text{let } x = M \text{ in } A N$

(A)  $\text{let } y = (\text{let } x = M \text{ in } A) \text{ in } E[y]$   
 $\rightarrow \text{let } x = M \text{ in let } y = A \text{ in } E[y]$

$\xrightarrow{\text{I}}$  reduction only using (I)

$\xrightarrow{\text{VCA}}$  reduction only using (V), (C) and (A) (administrative)

## Lemma (determinacy of call-by-need reduction)

- $\xrightarrow{I}$  is a partial function
- $\xrightarrow{VCA}$  is a partial function
- If  $E[x] = E'[y]$  then  $x = y$
- For any term  $M$ , **exactly one** of the following holds:
  1.  $M$  is an answer
  2.  $M = E[x]$  for some  $E$  and  $x$
  3.  $M \xrightarrow{I} N$  for some  $N$
  4.  $M \xrightarrow{VCA} N$  for some  $N$

# Outline

- 1 Call-by-name and call-by-need  $\lambda$ -calculi
- 2 Simplified proof of the correspondence**
- 3 Coq formalization
- 4 Conclusion



## Definition (correspondence of terms)

$\Downarrow$ : call-by-need terms  $\rightarrow$  call-by-name terms

$$x^{\Downarrow} = x$$

$$(\lambda x.M)^{\Downarrow} = \lambda x.M^{\Downarrow}$$

$$(M\ N)^{\Downarrow} = M^{\Downarrow}\ N^{\Downarrow}$$

$$(\text{let } x = M \text{ in } N)^{\Downarrow} = N^{\Downarrow}[x \mapsto M^{\Downarrow}]$$

- Expands **let**
- Equivalent to [Maraist+ 1998]  
(except “marked redexes”)

# Main theorem: correspondence of call-by-need with call-by-name

Theorem (soundness of  $\xrightarrow{\text{need}}$ )

If  $M \xrightarrow{\text{need}} A$ , then  $M^{\flat} \xrightarrow{\text{name}} V$  for some  $V$

Theorem (completeness of  $\xrightarrow{\text{need}}$ )

If  $M^{\flat} \xrightarrow{\text{name}} V$ , then  $M \xrightarrow{\text{need}} A$  for some  $A$

(Correspondence between  $A$  and  $V$  also holds)

## Cf. previous researches

- Ariola and Felleisen [1997]
  - Based on informally defined term graphs and their correspondence
- Maraist et al. [1998]
  - Complicated “marked reduction” and explicit treatment of reduction position

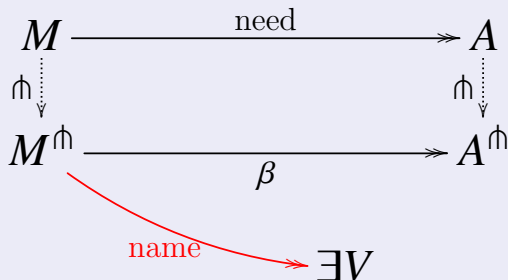
We give a **simpler proof!**

# Our proof

## Lemma (single-step correspondence)

- $A^{\dagger}$  is a value
- For any  $E$  and  $x$ , there exists  $E_n$  such that  $E[x]^{\dagger} = E_n[x]$
- If  $M \xrightarrow{\text{VCA}} N$  then  $M^{\dagger} = N^{\dagger}$
- If  $M \xrightarrow{\text{I}} N$  then  $M^{\dagger} \xrightarrow{\text{name}} \circ \xrightarrow{\beta} N^{\dagger}$

# Proof of soundness



Since  $A$  is an answer,  $A^{\text{♯}}$  is a value

Hence  $M^{\text{♯}} \xrightarrow{\text{name}} V$  by the termination of  
 $\xrightarrow{\text{name}}$



# Completeness

If  $M^\dagger \xrightarrow{\text{name}} V$  then  
 $M \xrightarrow{\text{need}} A$

Difficulties:

- Administrative reductions might not terminate
  - If  $M \xrightarrow{\text{VCA}} N$  then  $M^\dagger = N^\dagger$
- Redexes shared by **let** are reduced at once
  - $M \xrightarrow{\text{I}} N$  implies  $M^\dagger \xrightarrow{\text{name}} \circ \xrightarrow{\beta} N^\dagger$ 
    - Bodies of  $\lambda$ -abstraction can be reduced

Lemma (normalization of  $\xrightarrow{\text{VCA}}$ )

By a variant of [Maraist+ 1998]'s weighting:

$$\begin{aligned}\|x\|_s &= s(x) \\ \|\lambda x.M\|_s &= \|M\|_{s \circ [x \mapsto 1]} \\ \|MN\|_s &= 2\|M\|_s + 2\|N\|_s \\ \|\text{let } x = M \text{ in } N\|_s &= 2\|M\|_s + \|N\|_{s \circ [x \mapsto 1 + \|M\|_s]}\end{aligned}$$

$$M \xrightarrow{\text{VCA}} N \text{ implies } \|M\|_s > \|N\|_s$$

## Proof (completeness of call-by-need) (1/2).

Assume  $M \multimap \xrightarrow{\text{name}} V$ , show  $M \xrightarrow{\text{need}} A$

First, we show call-by-need reduction of  $M$  is normalizing

$$M \xrightarrow{\text{need}} \text{---} \xrightarrow{\text{need}}$$



## Proof (completeness of call-by-need) (1/2).

Assume  $M \multimap \xrightarrow{\text{name}} V$ , show  $M \xrightarrow{\text{need}} A$

First, we show call-by-need reduction of  $M$  is normalizing

$$\xrightarrow{\text{need}} = (\xrightarrow{\text{I}} \cup \xrightarrow{\text{VCA}})^* = (\xrightarrow{\text{VCA}} \circ \xrightarrow{\text{I}})^* \text{ holds}$$

$$M \xrightarrow{\text{VCA}} \xrightarrow{\text{I}} \text{---} \xrightarrow{\text{VCA}} \xrightarrow{\text{I}}$$

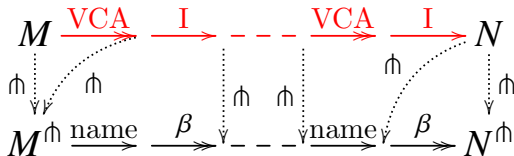
# Proof (completeness of call-by-need) (1/2).

Assume  $M \multimap \xrightarrow{\text{name}} V$ , show  $M \xrightarrow{\text{need}} A$

First, we show call-by-need reduction of  $M$  is normalizing

$\xrightarrow{\text{VCA}}$  is an administrative reduction

$\xrightarrow{\text{I}}$  corresponds  $\xrightarrow{\text{name}} \circ \xrightarrow{\beta}$

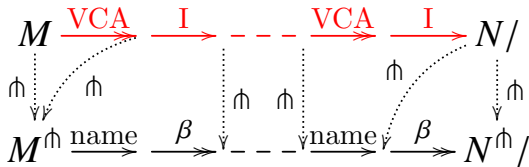


## Proof (completeness of call-by-need) (1/2).

Assume  $M^\heartsuit \xrightarrow{\text{name}} V$ , show  $M \xrightarrow{\text{need}} A$

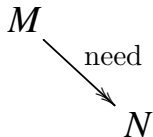
First, we show call-by-need reduction of  $M$  is normalizing

By  $M^\heartsuit \xrightarrow{\beta} V$ ,  $M^\heartsuit$  is terminating by  $\xrightarrow{\text{name}} \circ \xrightarrow{\beta}$  (= induction on derivation is available)



## Proof (completeness of call-by-need) (2/2).

Next, show normal form  $N$  of  $M$  is an answer



## Proof (completeness of call-by-need) (2/2).

Next, show normal form  $N$  of  $M$  is an answer

Normal form  $N$  is an answer or stuck state  $E[x]$

$$M \xrightarrow{\text{need}} N = A \vee N = E[x]$$

## Proof (completeness of call-by-need) (2/2).

Next, show normal form  $N$  of  $M$  is an answer

Assume  $N$  is stuck state, show it leads to contradiction

$$M \xrightarrow{\text{need}} N = E[x]$$

## Proof (completeness of call-by-need) (2/2).

Next, show normal form  $N$  of  $M$  is an answer

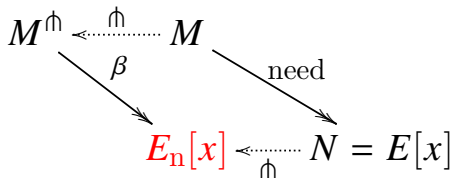
By single-step correspondence:

$$\begin{array}{ccc} M^\flat & \xleftarrow{\quad \flat \quad} & M \\ & \searrow \beta & \searrow \text{need} \\ & (E[x])^\flat & \xleftarrow{\quad \flat \quad} N = E[x] \end{array}$$

## Proof (completeness of call-by-need) (2/2).

Next, show normal form  $N$  of  $M$  is an answer

By single-step correspondence:

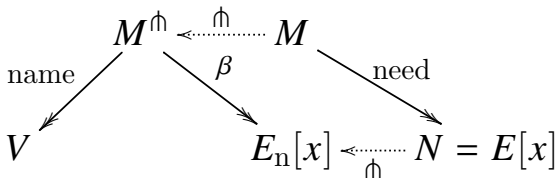




## Proof (completeness of call-by-need) (2/2).

Next, show normal form  $N$  of  $M$  is an answer

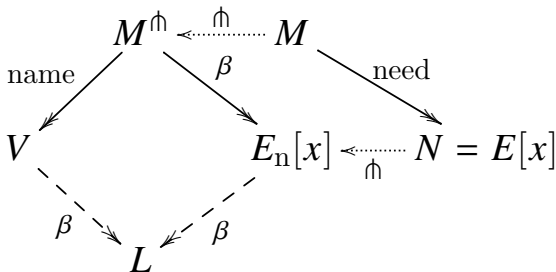
By assumption:



## Proof (completeness of call-by-need) (2/2).

Next, show normal form  $N$  of  $M$  is an answer

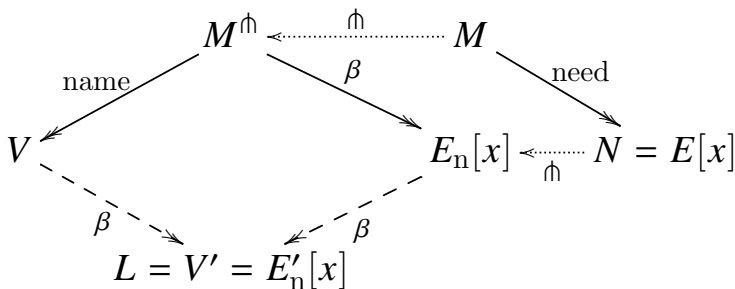
By confluence of  $\xrightarrow{\beta}$ :



## Proof (completeness of call-by-need) (2/2).

Next, show normal form  $N$  of  $M$  is an answer

$\xrightarrow{\beta}$  preserves valueness and stuckness in  
call-by-name



# Outline

- 1 Call-by-name and call-by-need  $\lambda$ -calculi
- 2 Simplified proof of the correspondence
- 3 Coq formalization**
- 4 Conclusion

## Coq formalization

Almost straightforward, expect treatment of evaluation contexts

```
Lemma answer_or_stuck_or_reducible M :  
  answer M  
    (    E x,  
      evalctx E      M = E.[tvar x]      bv E      x)  
    (    E L N,  
      evalctx E      M = E.[L]      reduceI L N)  
    (    E L N,  
      evalctx E      M = E.[L]      reduceVCA L N).
```

- Try induction on  $M$

## Case $M = x$

```
x : var
=====
answer (tvar x)
  (    E y, evalctx E      tvar x = E.[tvar y]
    bv E      x)
  (    E L N, evalctx E      tvar x = E.[L]
    reduceI L N)
  (    E L N, evalctx E      tvar x = E.[L]
    reduceVCA L N).
```

- Trivial from  $E = []$

## Case $M = x$

### ■ However, automated reasoning fails

```
Coq < eauto.

x : var
=====
answer (tvar x)
  (    E y, evalctx E      tvar x = E.[tvar y]
    bv E      x)
  (    E L N, evalctx E      tvar x = E.[L]
    reduceI L N)
  (    E L N, evalctx E      tvar x = E.[L]
    reduceVCA L N).
```

## Why fails?

To prove...

$$\begin{array}{l} E \ y, \text{ evalctx } E \\ \text{tvar } x = E.[\text{tvar } y] \qquad \text{bv } E \qquad x \end{array}$$

...we must find  $E$  such that

$$\text{tvar } x = E.[\text{tvar } y]$$



Higher order pattern matching required!



## Solution: eliminate evaluation contexts

- Expand evaluation contexts into reductions

- $\xrightarrow{\beta}, \xrightarrow{\text{name}}, \xrightarrow{I}$  and  $\xrightarrow{\text{VCA}}$

- Introduce stuckness predicate

- $\text{needs}_n(M, x) (\Leftrightarrow \exists E. M = E_n[x])$  and  
 $\text{needs}(M, x) (\Leftrightarrow \exists E. M = E[x])$

- Approximate

$$\text{let } x = V \text{ in } E[x] \rightarrow \text{let } x = V \text{ in } E[V]$$

by substitution

$$\text{let } x = V \text{ in } E[x] \rightarrow E[x][x \mapsto V]$$

(N.B. correspondence in original semantics is also proved)

# Automation succeeds!

```
Lemma answer_or_stuck_or_reducible M :  
  answer M \/  
    (exists x, needs M x) \/  
    (exists N, reduceI M N) \/  
    (exists N, reduceVCA M N).
```

Proof.

induction M as

[[|? [Hanswer|[[[]|[[[]|[[[]]]]]

||? [Hanswer|[[[]|[[[]|[[[]]]]]

? [|[[[]]]|[[[]|[[[]]]]]]; eauto 6;

inversion Hanswer; subst; eauto 6.

Qed.

# Outline

- 1 Call-by-name and call-by-need  $\lambda$ -calculi
- 2 Simplified proof of the correspondence
- 3 Coq formalization
- 4 Conclusion**

# Conclusion

- Formalized call-by-need  $\lambda$ -calculus [Ariola+ 1995] in the Coq proof assistant
- Gave **simplified proof** of correspondence with call-by-name, and verified in Coq
  - Using standardization theorem [Curry&Feys 1958]