Computational Soundness of Non-Confluent Calculi

Elena Machkasova

Wellesley College

About this talk

Reasons for this talk:

- discussion of some interesting properties of calculi.
- looking for "customers" for the new technique. Candidates: calculi with state, calculi with explicit substitution.

Computational soundness: intuition

Two calculus relations:

- *Evaluation* defines the meaning of a term with respect to the small-step operational semantics (what is the result of evaluating the term on the computer).
- *Calculus Rewrite rules* define equivalence of terms in the calculus. Correspond to *local* program transformations (e.g. function inlining, constant propagation, some loop optimizations).

Computational soundness relates the two: calculus relation preserves the meaning of a term. Hence local transformations preserve meaning.

Disclaimer: global transformations (such as closure conversion, function specialization) require different proof techniques.

2 main examples

- "Good" case: call-by-value λ-calculus with constants.
  - confluent
  - finite (bounded) confluent developments
- "Challenging" case: calculus of records with mutually recursive components.
  - non-confluent
  - developments are not finite and non-confluent

Call-by-value λ-calculus (CBV)

Includes numeric constants and operations.

$$M, N, L \in \text{Term} \ ::= \ c \mid x \mid (\lambda x.M) \mid M_1 \mathbin{@} M_2 \mid M_1 + M_2$$
$$V \in \textbf{Value} \ ::= \ c \mid x \mid \lambda x.M$$

Notion of reduction = basic computational step.

$$(\lambda x.M) \mathbin{@} V \quad \leadsto \quad M[x := V]$$

$$c_1 + c_2 \quad \leadsto \quad \overline{c_1 + c_2} \qquad \text{(the result of addition)}$$

Left-hand side of $\leadsto$ is called *redex*. R ranges over redexes, Q ranges over the right-hand sides of $\leadsto$.

Examples of evaluation in CBV

Evaluation $\Longrightarrow$ finds a unique evaluation redex in a term (if it exists). $\Longrightarrow$ does not reduce redexes under a λ.

the whole term:

$(\lambda x.x) @ (\lambda y.2+3) \quad \Longrightarrow \quad \lambda y.2+3$

left-to-right:

$((\lambda x.x) @ (\lambda y.y)) @ (2+3) \quad \Longrightarrow \quad (\lambda y.y) @ (2+3)$

operand after operator:

$(\lambda y.y) @$ (2 + 3) $\quad \Longrightarrow \quad (\lambda y.y) @ 5$

Gray box shows which redex was reduced in the reduction.

Examples of calculus relation in CBV

Calculus relation $\longrightarrow$ can reduce any redex in a term.

$$((\lambda x.x) \mathbin{@} (\lambda y.y)) \mathbin{@} \boxed{\text{2+3}} \quad \longrightarrow \quad ((\lambda x.x) \mathbin{@} (\lambda y.y)) \mathbin{@} 5$$

$$\boxed{((\lambda x.x) \mathbin{@} (\lambda y.y))} \mathbin{@} (2+3) \quad \longrightarrow \quad (\lambda y.y) \mathbin{@} (2+3)$$

- $\Longrightarrow$ is a function, $\longrightarrow$ is not.
- $\Longrightarrow \subset \longrightarrow$
- Notation: $\longrightarrow^*, \Longrightarrow^*$, etc. denote reflexive transitive closure of the respective relations.

Non-evaluation relation (denoted $\circ\!\!\longrightarrow$)

A *non-evaluation* relation $\circ\!\!\longrightarrow$ is defined as $\circ\!\!\longrightarrow = \longrightarrow \setminus \Longrightarrow$.

Example of different relations in CBV:

$$
\begin{array}{rl}
((\lambda x.x) \ @ \ (\lambda y.\lambda z.y + 1)) \ @ \ (\boxed{3 + 4}) & \circ\!\!\longrightarrow \\
\boxed{((\lambda x.x) \ @ \ (\lambda y.\lambda z.y + 1))} \ @ \ 7 & \Longrightarrow \\
\boxed{(\lambda y.\lambda z.y + 1) \ @ \ 7} & \Longrightarrow \\
\lambda z.\boxed{7 + 1} & \circ\!\!\longrightarrow \\
\lambda z.8 &
\end{array}
$$

Normal forms:

M is an *evaluation n. f.* if there is no N s.t. $M \Longrightarrow N$.
Examples: $\lambda z.7 + 1, \lambda z.8$.

M is a *calculus n. f.* if there is no N s.t. $M \longrightarrow N$.
Example: $\lambda z.8$.

Classification of terms

Classification is a total function from terms to a set of tokens.

$$\textit{Cl}(M) = \begin{cases} \textbf{evaluatable} \text{ if there is } N \text{ s.t. } M \Longrightarrow N \\ \textbf{const}(c) \text{ if } M = c \text{ (a constant)} \\ \textbf{abs} \text{ if } M = \lambda x.N \\ \textbf{error} \text{ otherwise} \end{cases}$$

Evaluatable terms: $(\lambda x.x) @ (\lambda y.y)$, $(\lambda x.x) @ (2+3)$, $1+5$.
Errors: $2 @ 3$, $(\lambda x.7+1)+5$.

- constants, abstractions are meaningful evaluation normal forms.
- errors are meaningless (“bad”) evaluation normal forms.

Class preservation: **if** $M \circ\!\!\longrightarrow N$**, then** *Cl*$(M) =$ *Cl*(N)**.**

Outcome: Meaning of a Term

- *Classification*: characterizes term at a particular time.
- *Outcome*: characterizes the ultimate fate of term.

$$\textit{Outcome}(M) = \begin{cases} \textit{Cl}(N) \text{ if } N \text{ is the eval. normal form of } M, \\ \bot \text{ if } M \text{ diverges} \end{cases}$$

Examples:

1. $\textit{Outcome}((\lambda x.x + 1) \ @ \ (3 + 4) = \mathbf{const}(8)$
2. $\textit{Outcome}((2 + 3) + (\lambda x.x)) = \mathbf{error}$
3. $\textit{Outcome}((\lambda w.w \ @ \ w) \ @ \ (\lambda w.w \ @ \ w)) = \bot$

Computational Soundness (formally)

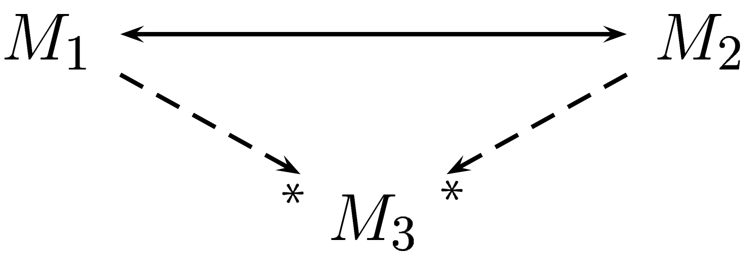
A calculus is computationally sound if $M \longrightarrow N$ implies *Outcome*$(M) =$ *Outcome*(N).

- Consequence of computational soundness: any program transformation represented as a sequence of calculus steps (forward and backward) is meaning-preserving.

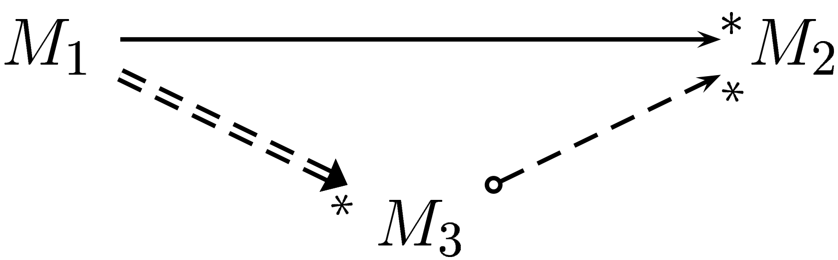
Traditional proof of comp. soundness

Ingredients of the proof:

Confluence:



Standardization:

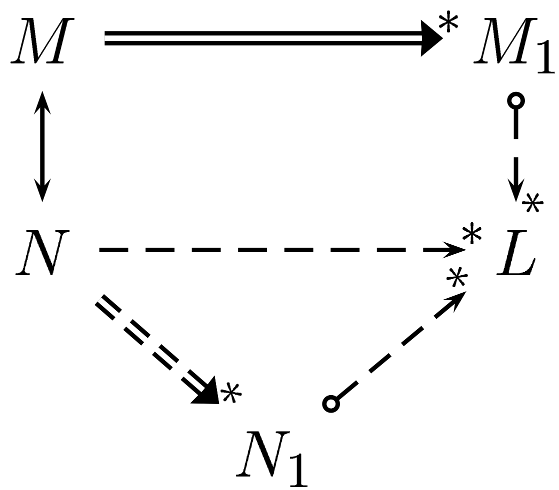


Class Preservation:

if $M \circ\!\!\longrightarrow M$ then *Cl*$(M) =$ *Cl*(N)

The proof:

Assume M_1 is eval. n.f.



Cl$(M_1) =$ *Cl*$(L) =$ *Cl*(N_1)

N_1 is eval. n.f.

Calculus of recursively-scoped records

- Record = unordered collection of uniquely labeled terms.
- Components may reference labels of other components.
- These dependencies may be mutually recursive.

Example (A, B, C, D are labels):

$$[A \mapsto B \mathbin{@} D, B \mapsto \lambda x.C, C \mapsto \lambda y.B, D \mapsto \lambda z.3]$$

Reductions on records include:

- reduction of a component
- substitution of a labeled value into a label reference.

Relations on records (example)

All the reductions below are examples of $\rightarrow$:

$$[A \mapsto 2+3, B \mapsto \text{C} \; @ \; A, C \mapsto \lambda x.x + A] \quad \Longrightarrow$$
$$[A \mapsto \text{2+3}, B \mapsto (\lambda x.x + A) \; @ \; A, C \mapsto \lambda x.x + A] \quad \Longrightarrow$$
$$[A \mapsto 5, B \mapsto (\lambda x.x + \text{A}) \; @ \; A, C \mapsto \lambda x.x + A] \quad \circ\!\!\!\longrightarrow$$
$$[A \mapsto 5, B \mapsto (\lambda x.x + 5) \; @ \; \text{A}, C \mapsto \lambda x.x + A] \quad \Longrightarrow$$
$$[A \mapsto 5, B \mapsto (\lambda x.x + 5) \; @ \; 5, C \mapsto \lambda x.x + A] \quad \Longrightarrow$$
$$[A \mapsto 5, B \mapsto 5 + 5, C \mapsto \lambda x.x + A] \quad \Longrightarrow$$
$$[A \mapsto 5, B \mapsto 10, C \mapsto \lambda x.x + \text{A}] \quad \circ\!\!\!\longrightarrow$$
$$[A \mapsto 5, B \mapsto 10, C \mapsto \lambda x.x + 5]$$

Note:

$[A \mapsto 5, B \mapsto 10, C \mapsto \lambda x.x + A]$ is an eval. n.f.

$[A \mapsto 5, B \mapsto 10, C \mapsto \lambda x.x + 5]$ is a calculus n.f.

Calculus of records is non-confluent

Example (along the lines of Ariola and Klop, 1997):

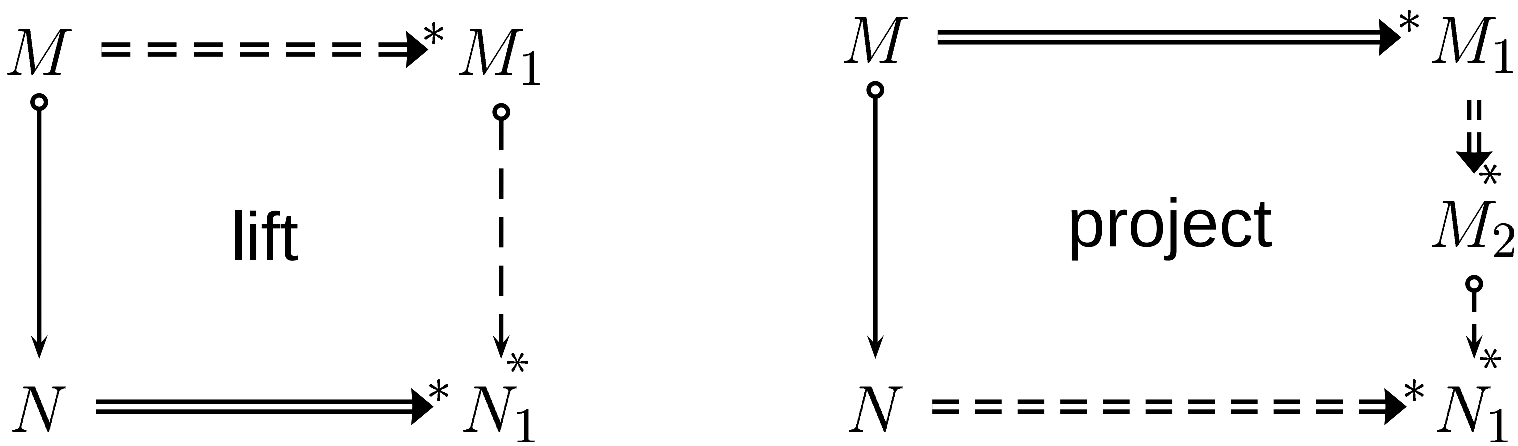
$$
\begin{array}{ccc}
[A \mapsto \lambda x.B, B \mapsto \lambda y.A] & \longrightarrow & [A \mapsto \lambda x.\lambda y.A, B \mapsto \lambda y.A] \\
\downarrow & & \vdots \\
[A \mapsto \lambda x.B, B \mapsto \lambda y.\lambda x.B] & \cdots\cdots\to & ?
\end{array}
$$

- in $[A \mapsto \lambda x.\lambda y.A, B \mapsto \lambda y.A]$ even number of λs in the first component, odd in the second.
- in $[A \mapsto \lambda x.B, B \mapsto \lambda y.\lambda x.B]$ odd number of λs in the first component, even in the second.

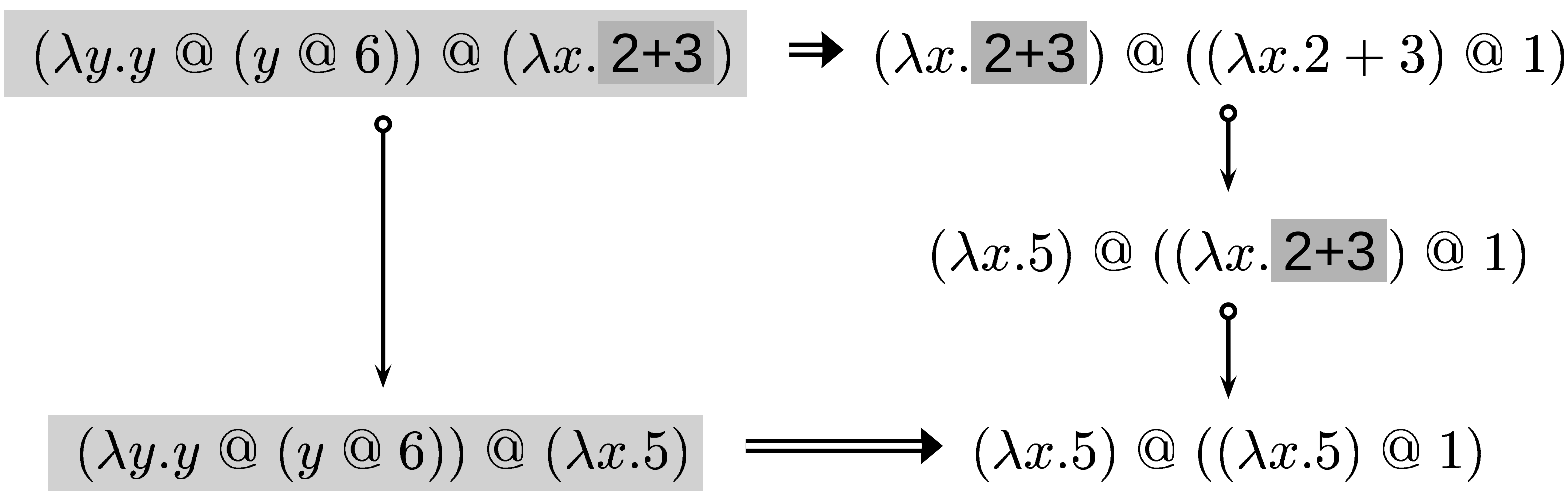
All reductions preserve this property, never arrive at the same term.

Traditional proof requires confluence. We need new approach.

New technique: Lift and Project

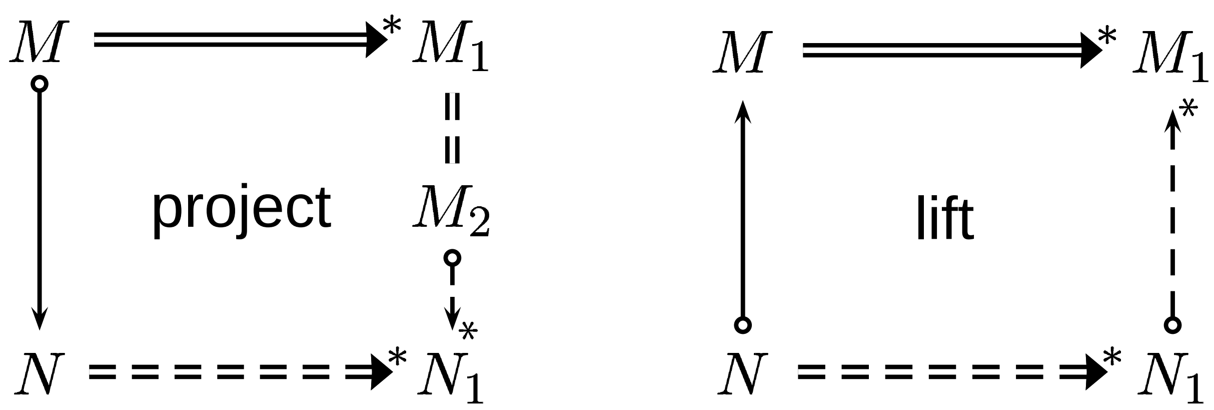


Example in CBV. Dark gray – redexes reduced by vertical arrows, light gray – redexes reduced by horizontal arrows.



New proof of computational soundness

Let M_1 be the evaluation normal form of M if it exists. We need to show that if $M \circ\!\!\rightarrow N$ or $N \circ\!\!\rightarrow M$ then *Outcome*$(M) =$ *Outcome*(N). Two cases:



Cl$(M_1) =$ *Cl*$(M_2) =$ *Cl*(N_1) *Cl*$(M_1) =$ *Cl*(N_1)

- Assume that class preservation holds.
- Assume that $\Longrightarrow$ is a function. In calculus of records $\Longrightarrow$ is not a function, but satisfies the diamond property. Proofs easily extend to this case.

Related work

- Computational soundness of confluent calculi: Plotkin 1975, Ariola, Felleisen, Maraist, Odersky, Wadler 1995, Taha 1999
- Proof techniques for confluence and/or standardization: Barendregt 1984, Huet, Levy 1991, Takahashi 1995, Gonthier, Levy, Mellies 1992, Wells, Muller 2000
- Related module calculi and recursive systems: Ariola, Klop 1997, Ariola, Blum 1997, Wells, Vestergaard 1999, Fisher, Reppy, Reicke 2000
- Applications to modules and linking: Machkasova, Turbak 2000, Machkasova 2002 (PhD thesis).

Future directions

- Applying the new technique to other non-confluent calculi, such as:
  - calculi with letrec.
  - calculi with state, side effects.
  - explicit substitution.
- Extending our technique to handle more calculi.
- Combining our technique with other program analyses (termination analysis).
- Considering other versions of classification.