Control Reduction Theories: the Benefit of Structural Substitution

Zena M. Ariola∗ Hugo Herbelin
University of Oregon INRIA-Futurs

with a Historical Note by Matthias Felleisen

Abstract

The historical design of the call-by-value theory of control relies on the reification of evaluation contexts as regular functions and on the use of ordinary term application for jumping to a continuation. To the contrary, the $\lambda_{c_{\mathsf{tp}}}$ control calculus, developed by the authors, distinguishes between *jumps* and *terms*. This alternative calculus, which derives from Parigot's $\lambda\mu$-calculus, works by direct *structural substitution* of evaluation contexts. We review and revisit the legacy theories of control and argue that $\lambda_{c_{\mathsf{tp}}}$ provides an observationally equivalent but smoother theory. In an additional note contributed by Matthias Felleisen, we review the story of the birth of control calculi during the mid to late eighties at Indiana University.

1 Introduction

To reason about Scheme programs (Felleisen *et al.*, 1987) introduced the λ_c reduction theory. This theory was not pure in the sense that one of the rules was applicable only at the top of a program. To address this issue, Felleisen and Hieb introduced the λ_c revised reduction theory (Felleisen & Hieb, 1992) that was exclusively made of contextually valid equations. Both theories suffer a few weaknesses:

- Both reduction theories do not directly express the operational semantics of $\mathcal{C}$: reduction and operational semantics coincide only at the observational level.
- To simulate the operational semantics, the reduction semantics has to accommodate the following reduction rule:

$$\mathcal{C}_E: \quad E[\mathcal{C}\,M] \quad \rightarrow \quad \mathcal{C}\,(\lambda k.\,M\,(\lambda x.\,\mathcal{A}\,(k\,E[x])))$$

  However, it turns out that both reduction semantics are not confluent when extended with this rule.
- The revised theory has a complex notion of answers: An evaluation may simply yield a value, or produce an answer of the shape $\mathcal{C}\,(\lambda k.\,V)$ (with V possibly containing k) or produce an answer of the shape $\mathcal{C}\,(\lambda k.\,k\,V)$ (again with V possibly containing k). In the latter case, when V does not contain k, one

∗ Supported by National Science Foundation grant number CCR-0204389

would expect an additional reduction that eliminates the superfluous $\mathcal{C}$ application:

$$\mathcal{C}_{elim}: \quad \mathcal{C}\,(\lambda k.\, k\,M) \quad \to \quad M \quad k \text{ not free in } M$$

However, it turns out that the addition of this rule to the revised λ_c reduction semantics breaks confluence. On this observation, and on the previous one, Felleisen and Hieb in (Felleisen & Hieb, 1992) write: "We leave unsolved the problem of finding an extended theory that includes $\mathcal{C}_E$ or $\mathcal{C}_{elim}$ and still satisfies the classical properties of reduction theories".

- λ_c is not as expressive as one might expect. For instance, Scheme's `call/cc` operational semantics

$$E[\texttt{call/cc}\,(\lambda k.\, M)] \quad \mapsto \quad E[M[\lambda x.\, \mathcal{A}\,E[x]/k]]$$

cannot be simulated. Indeed, if we encode `call/cc` as usual as $\lambda x.\, \mathcal{C}\,(\lambda k.\, k\,(x\,k))$, one gets

$$E[\texttt{call/cc}\,(\lambda k.\, M)] \quad \mapsto\!\!\to \quad (\lambda x.\, \mathcal{A}\,E[x])\,(M[\lambda x.\, \mathcal{A}\,E[x]/k])$$

which does not converge to $E[M[\lambda x.\, \mathcal{A}\,E[x]/k]]$.

The calculus $\lambda_{c\mathtt{tp}}$ provides a solution to the above problems, and thus can be seen as a replacement of λ_c. The calculus $\lambda_{c\mathtt{tp}}$ is a call-by-value reformulation of Parigot's $\lambda\mu$ (Parigot, 1992), where μ is renamed into $\mathcal{C}$. It also contains a special constant called `tp` which denotes the top-level continuation, making explicit the abortive capabilities of λ_c. The essential design differences between λ_c and $\lambda_{c\mathtt{tp}}$ are the following:

- $\lambda_{c\mathtt{tp}}$ has specific variables for contexts while λ_c does not,
- λ_c reifies contexts as functions and moves them around using the standard substitution of λ-calculus while $\lambda_{c\mathtt{tp}}$ uses a specific notion of *structural substitution* of contexts,
- $\lambda_{c\mathtt{tp}}$ syntax forces calls to continuations to be abortive while λ_c uses a specific reduction rule for this purpose,
- λ_c does not have a special constant for the top-level continuation.

The calculus $\lambda_{c\mathtt{tp}}$ comes with a simple operational semantics expressive enough to simulate the semantics of `call/cc`, as described above. It is also expressive enough to simulate the operational semantics of λ_c, while the converse is false.

The calculus $\lambda_{c\mathtt{tp}}$ comes with a confluent reduction semantics which, to the contrary of λ_c, can simulate its own operational semantics. It also remains confluent when extended with a rule equivalent to $\mathcal{C}_E$.

Since $\lambda_{c\mathtt{tp}}$ reduction semantics simulates $\lambda_{c\mathtt{tp}}$ operational semantics, which itself can simulate λ_c operational semantics, which itself cannot be simulated by λ_c reduction semantics, it follows that the reduction theories of λ_c and $\lambda_{c\mathtt{tp}}$ do not simulate each other, as already observed in (Ong & Stewart, 1997). However, since λ_c operational semantics and reduction semantics are equivalent with respect to the observational behavior of a program, the same holds for the reduction semantics of λ_c and $\lambda_{c\mathtt{tp}}$. In short: A λ_c program reduces to an answer if and only if the corresponding $\lambda_{c\mathtt{tp}}$ program reduces to an answer.

$x, a, v, f, c, k \in \mathit{Vars}$		
$M, N \in \mathit{Terms}$	$::=$	$x \mid \lambda x.\, M \mid M\ N \mid \mathcal{A}\mathit{bort}\ M \mid \mathcal{C}\ M$
$V \in \mathit{Values}$	$::=$	$x \mid \lambda x.\, M$
$E \in \mathit{EvCtxt}$	$::=$	$\Box \mid E\ M \mid V\ E$

Fig. 1. Syntax of λ_c

The reduction theory of $\lambda_{c\mathsf{tp}}$ can be formulated either on terms or on jumps. If one formulates it on terms, it shares with the λ_c revised reduction theory the complexity of the notion of answer. However, if we formulate it on jumps (and we execute jumps of the form $\mathsf{tp}\,M$), the evaluation produces results of the unique shape $\mathsf{tp}\,V$. In fact, a similar approach can be done in λ_c too: By considering evaluation in an abortive context, all three forms of answers collapse to a single one.

The paper is organized as follows: Section 2 introduces λ_c, reviews its main properties and individuates its shortcomings. Section 3 introduces $\lambda_{c\mathsf{tp}}$ and shows how it solves λ_c's defects. All along these two sections, the relations between the different notions of operational and reduction semantics for the two calculi are investigated. Section 4 summarizes the agreement on the observational behaviors of λ_c and $\lambda_{c\mathsf{tp}}$ (Figure 8) and the discrepancies regarding the operational semantics (Figure 9). We conclude in Section 5 together with a historical note by Matthias Felleisen.

2 Indiana's Theory of Control

We start with the syntax of λ_c and its operational semantics. We present the *computational* reduction semantics given in (Felleisen *et al.*, 1987). This theory has two weaknesses:

- it contains one rule, called a *computational* rule, which is only applicable at the top of a program;
- the rules are not complete with respect to the operational semantics.

Next, we give the revised reduction semantics from (Felleisen & Hieb, 1992). This theory characterizes the computational rule in terms of two compatible rules (*i.e.* applicable in any context). Thus, solving one problem with the original theory at the expenses of complicating the correspondence with the operational semantics. We discuss how this relationship could be simplified by reducing a program in a particular context, which intuitively captures the execution of a program at the top-level prompt. As discussed in (Plotkin, 1975), the relationship between the reduction theory and an evaluator should be mediated by a *standardization theorem*. To that end, for both theories we define a notion of standard reduction and of weak-head reduction (*i.e.* a notion of standard reduction that stops at values).

2.1 Syntax and Operational Semantics

Figure 1 introduces the syntax of a call-by-value calculus extended with the unary operators $\mathcal{A}\mathit{bort}$ and $\mathcal{C}$. Variables and lambda-abstractions are called values.

The operational semantics of such a language can be described most concisely using the following operational rules, which rewrite complete programs:

$$\begin{array}{rrcl} \beta_v: & E[(\lambda x.\, M)\, V] & \mapsto_{\lambda_c} & E[M\,[V/x]] \\ \mathcal{A}bort_{T_E}: & E[\mathcal{A}bort\, M] & \mapsto_{\lambda_c} & M \\ \mathcal{C}_{T_E}^{\mathcal{A}bort}: & E[\mathcal{C}\, M] & \mapsto_{\lambda_c} & M\,(\lambda x.\, \mathcal{A}bort\, E[x]) \end{array}$$

The reflexive-transitive closure of $\mapsto_{\lambda_c}$ is denoted by $\twoheadmapsto_{\lambda_c}$. In each of the rules, the entire program is split into an evaluation context E and a current redex to rewrite. The evaluation context E is a term with exactly one hole, written as $\Box$, in it. It represents what to do after the execution of the redex and is referred to as the *continuation*. The first rule expresses what to do when a function is applied to a value: the argument is substituted for each free occurrence of the bound variable in the function's body. According to the second operational rule, the application of $\mathcal{A}bort$ to a term M aborts the current continuation (*i.e.* E) and returns M to the top-level. For example, one has:

$$1 + \mathcal{A}bort\, M + 3 \mapsto_{\lambda_c} M$$

where in this case the abandoned context is $1 + \Box + 3$. According to the last rule, the application of $\mathcal{C}$ to a term M abandons the current evaluation context and applies M to a procedural abstraction of that context. Note the presence of the abort operation in the abstracted context, which is $(\lambda x.\, \mathcal{A}bort\, E[x])$ and not $(\lambda x.\, E[x])$. This distinguishes continuations from regular functions. A function returns to the caller once completed, whereas the invocation of a continuation causes the context of the application to be discarded.

Through the paper, we will use the λ_c-term $\mathcal{C}\,(\lambda c.\, 1 + c\, 2 + (1 + 1)) + 3$ as our running example.

Example 2.1 (*Evaluation of* $\mathcal{C}\,(\lambda c.\, 1 + c\, 2 + (1 + 1)) + 3$)
The term $\mathcal{C}\,(\lambda c.\, 1 + c\, 2 + (1 + 1)) + 3$ is split into the evaluation context $\Box + 3$ and the redex $\mathcal{C}\,(\lambda c.\, 1 + c\, 2 + (1 + 1))$. The current evaluation context $\Box + 3$ is abandoned and the argument of $\mathcal{C}$ is applied to a procedural abstraction of that context:

$$\mathcal{C}\,(\lambda c.\, 1 + c\, 2 + (1 + 1)) + 3 \mapsto_{\lambda_c} (\lambda c.\, 1 + c\, 2 + (1 + 1))\,(\lambda x.\, \mathcal{A}bort\,(x + 3))$$

Continuing with the evaluation:

$$(\lambda c.\, 1 + c\, 2 + (1 + 1))\,(\lambda x.\, \mathcal{A}bort\,(x + 3)) \mapsto_{\lambda_c} 1 + (\lambda x.\, \mathcal{A}bort\,(x + 3))\, 2 + (1 + 1)$$

The invocation of the continuation abandons the calling context $1 + \Box + (1 + 1)$:

$$1 + (\lambda x.\, \mathcal{A}bort\,(x + 3))\, 2 + (1 + 1) \mapsto_{\lambda_c} 1 + \mathcal{A}bort\,(2 + 3) + (1 + 1)) \twoheadmapsto_{\lambda_c} 5$$

∎

$\mathcal{C}$ is at least as expressive as $\mathcal{A}bort$; it can be used to define an operator $\mathcal{A}$ equivalent to $\mathcal{A}bort$:

$$\mathcal{A}\, M \triangleq \mathcal{C}\,(\lambda k.\, M) \quad \text{where } k \text{ does not occur free in } M \quad \text{(Abbrev. 1)}$$

β_v :	$(\lambda x.\, M)\, V$	$\rightarrow_c$	$M\,[V/x]$
$\mathcal{C}_L$:	$(\mathcal{C} M)\, N$	$\rightarrow_c$	$\mathcal{C}\,(\lambda c.\, M\,(\lambda f.\, \mathcal{A}\,(c\,(f\, N))))$
$\mathcal{C}_R$:	$V\,(\mathcal{C}\, M)$	$\rightarrow_c$	$\mathcal{C}\,(\lambda c.\, M\,(\lambda x.\, \mathcal{A}\,(c\,(V\, x))))$
$\mathcal{C}_T$:	$\mathcal{C} M$	$\triangleright_{\mathcal{C}_T}$	$M\,(\lambda x.\, \mathcal{A}\, x)$

Fig. 2. Reduction and computation rules of call-by-value λ_c (Felleisen-Friedman-Kohlbecker-Duba)

To capture the proviso we often use _ which refers to an anonymous variable, and write $\mathcal{A}\, M$ as $\mathcal{C}\,(\lambda\_.\, M)$. If we replace $\mathcal{C}_{T_E}^{\mathcal{A}bort}$ by

$$\mathcal{C}_{T_E} : E[\mathcal{C}\, M] \mapsto_{\lambda_\mathcal{C}} M\,(\lambda x.\, \mathcal{A}\, E[x])\,,$$

then $\mathcal{A}bort_{T_E}$, where $\mathcal{A}bort$ has been replaced by $\mathcal{A}$, becomes derivable:

$$E[\mathcal{A}\, M] \mapsto_{\lambda_\mathcal{C}} (\lambda\_.\, M)\,(\lambda x.\, \mathcal{A}\, E[x]) \mapsto_{\lambda_\mathcal{C}} M\,.$$

Henceforth, we have the following easy result:

Proposition 2.2
For M with no occurrences of $\mathcal{A}bort$,

$M \mapsto\!\!\!\!\rightarrow_{\lambda_\mathcal{C}} V$ with rules β_v, $\mathcal{A}bort_{T_E}$ and $\mathcal{C}_{T_E}^{\mathcal{A}bort}$ iff $M \mapsto\!\!\!\!\rightarrow_{\lambda_\mathcal{C}} V'$ with rules β_v and $\mathcal{C}_{T_E}$

where V' is V where each $\mathcal{A}bort$ has been replaced by $\mathcal{A}$.

We will therefore focus on $\mathcal{C}$ in the remainder of the paper, and, unless stated otherwise, use $\mathcal{A}$ and $\mathcal{C}_{T_E}$ instead of $\mathcal{A}bort$, $\mathcal{A}bort_{T_E}$ and $\mathcal{C}_{T_E}^{\mathcal{A}bort}$.

2.2 Felleisen-Friedman-Kohlbecker-Duba Reduction Semantics

2.2.1 Reduction Rules

The initial reduction semantics of λ_c in (Felleisen *et al.*, 1987) is characterized by a combination of *congruent* reduction rules (written $\rightarrow_c$) applicable at any place of an expression and of a so-called *computational* rule (written $\triangleright_{\mathcal{C}_T}$) applicable only at the top-level of a computation. The rules are on Figure 2.

The local reduction rules are intuitively related to the operational rules as follows. Instead of capturing the entire evaluation context surrounding an invocation of $\mathcal{C}$ in one step, the rules $\mathcal{C}_L$ and $\mathcal{C}_R$ allow one to *lift* the control operation step-by-step until it reaches the top-level. At that point rule $\mathcal{C}_T$ applies the abort continuation. The C-reduction $\twoheadrightarrow_c$ is defined as the reflexive-transitive closure of $\rightarrow_c$. The C-computation $\triangleright_c$ is defined as the union of $\twoheadrightarrow_c$ and $\triangleright_{\mathcal{C}_T}$. Its reflexive-transitive closure is written $\triangleright_c^*$. Its reflexive-symmetric-transitive closure is written $\overset{\triangleright}{=}_c$. The C-computation $\triangleright_c$ is proved to satisfy the diamond property.

Example 2.3 (*Reduction of* $\mathcal{C}\,(\lambda c.\, 1 + c\, 2 + (1 + 1)) + 3$)

$$
\begin{array}{lll}
\mathcal{C}\,(\lambda c.\,1 + c\,2 + (1 + 1)) + 3 & \to_c & \mathcal{C}_L \\
\mathcal{C}\,(\lambda c'.\,(\lambda c.\,1 + c\,2 + (1 + 1))\,(\lambda x.\,\mathcal{A}\,(c'\,(x + 3)))) & \to_c & \beta_v \\
\mathcal{C}\,(\lambda c'.\,1 + ((\lambda x.\,\mathcal{A}\,(c'\,(x + 3)))\,2) + (1 + 1)) & \to_c & \beta_v \\
\mathcal{C}\,(\lambda c'.\,1 + \mathcal{A}\,(c'(\,2 + 3)) + (1 + 1)) & \twoheadrightarrow & \\
\mathcal{C}\,(\lambda c'.\,\mathcal{A}\,(c'\,(2 + 3))) & \rhd_{\mathcal{C}_T} & \\
(\lambda c'.\,\mathcal{A}\,(c'\,(2 + 3)))\,(\lambda x.\,\mathcal{A}\,x) & \twoheadrightarrow_c & \\
\mathcal{A}\,(\mathcal{A}\,5) & \rhd_{\mathcal{C}_T} & \\
(\lambda\_.\,\mathcal{A}\,5)\,(\lambda x.\,\mathcal{A}\,x) & \to_c & \beta_v \\
\mathcal{A}\,5 & \rhd_{\mathcal{C}_T} & \\
(\lambda\_.\,5)\,(\lambda x.\,\mathcal{A}\,x) & \to_c & \beta_v \\
5 &&
\end{array}
$$

2.2.2 Weak-Head Reduction

A part from the $\rhd_{\mathcal{C}_T}$ rule, the other rules can be applied in any order, including under a lambda-abstraction and a $\mathcal{C}$-abstraction. However, to use the reduction theory to reason about evaluation, it is important to define a notion of reduction which mimics the evaluator. To that end, one defines the notion of *weak-head* reduction. The C-computation has a natural notion of weak-head reduction (called standard reduction function in (Felleisen *et al.*, 1987), following Plotkin's terminology (Plotkin, 1975)). We say that M *weakly head reduces* to N for $\to_c$, written $M \overset{wh}{\to}_c N$, iff M has the form $E[P]$, where P is a β_v, $\mathcal{C}_L$ or $\mathcal{C}_R$ redex that reduces to Q, and N is $E[Q]$ (*i.e.*reduction occurs in an evaluation context position). The notation $\overset{wh}{\twoheadrightarrow}_c$ stands for the reflexive-transitive closure of $\overset{wh}{\to}_c$. We say that M *weakly head reduces* to N for $\rhd_c$, written $M \overset{wh}{\rhd}_c N$, iff $M \overset{wh}{\to}_c N$ or $M \rhd_{\mathcal{C}_T} N$. The notation $\overset{wh}{\rhd}{}^*_c$ stands for the reflexive-transitive closure of $\overset{wh}{\rhd}_c$.

Example 2.4 (*Weak-head reduction of* $\mathcal{C}\,(\lambda c.\,1 + c\,2 + (1 + 1)) + 3$)

We write $\mathcal{A}_x$ for the abort continuation $\lambda x.\,\mathcal{A}\,x$. We divide the reductions in different

groups separated by a blank line. Each group will collapse into a single step shortly.

$$
\begin{array}{ll}
\mathcal{C}\,(\lambda c.\,1\,+\,c\,2\,+\,(1\,+\,1))\,+\,3 & \overset{wh}{\rightarrow}_c \\
\mathcal{C}\,(\lambda c'.\,(\lambda c.\,1\,+c\,2\,+\,(1\,+\,1))\,(\lambda x.\,\mathcal{A}\,(c'\,(x\,+\,3)))) & \triangleright_{\mathcal{C}_T} \\
(\lambda c'.\,(\lambda c.\,1\,+c\,2\,+\,(1\,+\,1))\,(\lambda x.\,\mathcal{A}\,(c'\,(x\,+\,3))))\,\mathcal{A}_x & \overset{wh}{\rightarrow}_c \\
 & \\
(\lambda c.\,1\,+c\,2\,+\,(1\,+\,1))\,(\lambda x.\,\mathcal{A}\,(\mathcal{A}_x\,(x\,+\,3))) & \overset{wh}{\rightarrow}_c \\
 & \\
1\,+\,(\lambda x.\,\mathcal{A}\,(\mathcal{A}_x\,(x\,+\,3)))\,2\,+\,(1\,+\,1) & \overset{wh}{\rightarrow}_c \\
 & \\
1\,+\,\mathcal{A}\,(\mathcal{A}_x\,(2\,+\,3))\,+\,(1\,+\,1) & \overset{wh}{\rightarrow}_c \\
1\,+\mathcal{C}\,(\lambda q.\,(\lambda\_.\,\mathcal{A}_x\,(2\,+\,3))\,(\lambda z.\,\mathcal{A}\,(q\,(z\,+\,(1\,+\,1))))) & \overset{wh}{\rightarrow}_c \\
\mathcal{C}\,(\lambda r.\,(\lambda q.\,(\lambda\_.\,\mathcal{A}_x\,(2\,+\,3))\,(\lambda z.\,\mathcal{A}\,(q\,(z\,+\,(1\,+\,1)))))\,(\lambda w.\,\mathcal{A}\,(r\,(1\,+\,w)))) & \triangleright_{\mathcal{C}_T} \\
(\lambda r.\,(\lambda q.\,(\lambda\_.\,\mathcal{A}_x\,(2\,+\,3))\,(\lambda z.\,\mathcal{A}\,(q\,(z\,+\,(1\,+\,1)))))\,(\lambda w.\,\mathcal{A}\,(r\,(1\,+\,w))))\,\mathcal{A}_x & \overset{wh}{\rightarrow}_c \\
(\lambda q.\,(\lambda\_.\,\mathcal{A}_x\,(2\,+\,3))\,(\lambda z.\,\mathcal{A}\,(q\,(z\,+\,(1\,+\,1)))))\,(\lambda w.\,\mathcal{A}\,(\mathcal{A}_x\,(1\,+\,w))) & \overset{wh}{\rightarrow}_c \\
 & \\
(\lambda\_.\,\mathcal{A}_x\,(2\,+\,3))\,(\lambda z.\,\mathcal{A}\,((\lambda w.\,\mathcal{A}\,(\mathcal{A}_x\,(1\,+\,w)))\,(z\,+\,(1\,+\,1)))) & \overset{wh}{\rightarrow}_c \\
 & \\
(\lambda x.\,\mathcal{A}\,x)\,(2\,+\,3) & \overset{wh}{\rightarrow}_c \\
 & \\
(\lambda x.\,\mathcal{A}\,x)\,5 & \overset{wh}{\rightarrow}_c \\
 & \\
\mathcal{A}\,5 & \triangleright_{\mathcal{C}_T} \\
 & \\
(\lambda\_.\,5)\,(\lambda x.\,\mathcal{A}\,x) & \overset{wh}{\rightarrow}_c \\
 & \\
5 &
\end{array}
$$

∎

The following proposition extends the unique context lemma in (Felleisen & Friedman, 1986) to terms with free variables:

Proposition 2.5 (*Unique context lemma for* $\overset{wh}{\triangleright}{}^*_c$)
Let M be a term in λ_c. Exactly one of the following cases happens:

- M is a value V (we also say that M is an *answer*).
- M has a unique decomposition under the form $E[P]$ where P is a β_v, $\mathcal{C}_L$ or $\mathcal{C}_R$ redex.
- M has the form $\mathcal{C}\,N$ which is a $\triangleright_{\mathcal{C}_T}$ redex.
- M has a unique decomposition under the form $E[x\,V]$ in which case M is said *to have its weak-head reduction stopped*.

Especially, a weak-head redex, if it exists, is unique.

Intermezzo 2.6

Observe that if M weakly head reduces to N by $\mathcal{C}_L$ or $\mathcal{C}_R$, then it is necessarily weakly head reducible further by a sequence (possibly empty) of $\mathcal{C}_L$ or $\mathcal{C}_R$, ended by $\triangleright_{\mathcal{C}_T}$ and by as many β_v as the number of $\mathcal{C}_L$ or $\mathcal{C}_R$. We write $\triangleright_{\mathcal{C}_{T_{E^*}}}$ for such a combination of rules (which generalizes $\triangleright_{\mathcal{C}_T}$):

$$\mathcal{C}_{T_{E^*}}: \quad E[\mathcal{C}\,M] \quad \triangleright_{\mathcal{C}_{T_{E^*}}} \quad M\,E^*$$

where E^* is defined as:

$$\begin{array}{lcl} \Box^* & = & \lambda x.\,\mathcal{A}\,x \\ E[V\,\Box]^* & = & \lambda x.\,\mathcal{A}\,(E^*\,(V\,x)) \\ E[\Box\,N]^* & = & \lambda x.\,\mathcal{A}\,(E^*\,(x\,N)) \end{array}$$

Example 2.7 (*Alternative weak-head reduction of* $\mathcal{C}(\lambda c.\,1 + c\,2 + (1 + 1)) + 3$)

$$\begin{array}{ll} \mathcal{C}\,(\lambda c.\,1 + c\,2 + (1 + 1)) + 3 & \triangleright_{\mathcal{C}_{T_{E^*}}} \\ (\lambda c.\,1 + c\,2 + (1 + 1))\,(\lambda x.\,\mathcal{A}\,((\lambda x.\,\mathcal{A}\,x)\,(x + 3))) & \overset{wh}{\rightarrow}_c \\ 1 + (\lambda x.\,\mathcal{A}\,((\lambda x.\,\mathcal{A}\,x)\,(x + 3)))\,2 + (1 + 1) & \overset{wh}{\rightarrow}_c \\ 1 + \mathcal{A}\,((\lambda x.\,\mathcal{A}\,x)\,(2 + 3)) + (1 + 1) & \triangleright_{\mathcal{C}_{T_{E^*}}} \\ (\lambda\_.\,(\lambda x.\,\mathcal{A}\,x)\,(2 + 3))\,(\lambda z.\,\mathcal{A}\,((\lambda w.\,\mathcal{A}\,((\lambda x.\,\mathcal{A}\,x)\,(1 + w)))\,(z + (1 + 1)))) & \overset{wh}{\rightarrow}_c \\ (\lambda x.\,\mathcal{A}\,x)\,(2 + 3) & \overset{wh}{\rightarrow}_c \\ (\lambda x.\,\mathcal{A}\,x)\,5 & \overset{wh}{\rightarrow}_c \\ \mathcal{A}\,5 & \triangleright_{\mathcal{C}_{T_{E^*}}} \\ (\lambda\_.\,5)\,(\lambda x.\,\mathcal{A}\,x) & \overset{wh}{\rightarrow}_c \\ 5 & \end{array}$$

Comparing it with the reduction in Example 2.4, one has that the first $\mathcal{C}_{T_{E^*}}$ step corresponds to one lifting step, one $\triangleright_{\mathcal{C}_T}$ step and one β_v step. The second $\mathcal{C}_{T_{E^*}}$ corresponds to two lifting steps, one $\triangleright_{\mathcal{C}_T}$ step and two β_v steps. Whereas the last $\mathcal{C}_{T_{E^*}}$ corresponds to one $\triangleright_{\mathcal{C}_T}$ step. ∎

Moreover, if M is of the form $\lambda k.\,N$, then $E[\mathcal{C}\,(\lambda k.\,N)]$ weakly head reduces further to $N[E^*/k]$. This leads to the following variant of $\triangleright_{\mathcal{C}_{T_{E^*}}}$:

$$\mathcal{C}'_{T_{E^*}}: \quad E[\mathcal{C}\,(\lambda k.\,N)] \quad \triangleright_{\mathcal{C}'_{T_{E^*}}} \quad N[E^*/k]$$

Let $\mathcal{C}_L^-$, $\mathcal{C}_R^-$ and $\mathcal{C}_T^-$ be the restrictions of $\mathcal{C}_L$, $\mathcal{C}_R$ and $\mathcal{C}_T$ that apply only when the body of $\mathcal{C}$ is not an abstraction. Writing $\overset{wh}{\triangleright}_{\mathcal{C}_{T_{E^*}}\beta_v}$ for the union of $\triangleright_{\mathcal{C}_{T_{E^*}}}$ and weak-head β_v, and $\overset{wh}{\triangleright}_{\mathcal{C}'_{T_{E^*}}\mathcal{C}_T^-\mathcal{C}_L^-\mathcal{C}_R^-\beta_v}$ for the union of $\triangleright_{\mathcal{C}'_{T_{E^*}}}$ and weak-head reduction of $\mathcal{C}_T^-$, $\mathcal{C}_L^-$, $\mathcal{C}_R^-$ and β_v redexes, we get the following equivalence:

Proposition 2.8
$M \overset{wh}{\triangleright}_c^* V$ iff $M \overset{wh}{\triangleright}^*_{\mathcal{C}_{T_{E^*}}\beta_v} V$ iff $M \overset{wh}{\triangleright}^*_{\mathcal{C}'_{T_{E^*}}\mathcal{C}_T^-\mathcal{C}_L^-\mathcal{C}_R^-\beta_v} V$. Moreover, the Unique Context Lemma still holds by replacing items 2 and 3 in its statement by the rules composing $\overset{wh}{\triangleright}^*_{\mathcal{C}_{T_{E^*}}\beta_v}$ or by the rules composing $\overset{wh}{\triangleright}^*_{\mathcal{C}'_{T_{E^*}}\mathcal{C}_T^-\mathcal{C}_L^-\mathcal{C}_R^-\beta_v}$.

2.2.3 Operational Semantics vs Weak-Head Reduction

The formulation of weak-head reduction in terms of $\mathcal{C}_{T_{E^*}}$ and β_v allows one to compare it to the operational semantics: β_v steps match but $\mathcal{C}_{T_{E^*}}$ steps do not. Indeed, the weak-head reduction reduces $E[\mathcal{C}\,M]$ to $M\,E^*$ while the operational semantics reduces it to $(M\,(\lambda x.\,\mathcal{A}\,E[x]))$. Consider our example term, the operational semantics binds continuation variable c to $\lambda x.\,\mathcal{A}\,(x\,+\,3)$, whereas the weak-head reduction binds c to $(\lambda x.\,\mathcal{A}\,((\lambda x.\,\mathcal{A}\,x)\,(x\,+\,3)))$. In general, the problem is that the operational semantics lifts the context at once, whereas the reduction theory lifts the control operation step-by-step. Unfortunately, each lifting introduces a new λ-abstraction to represent its partial continuation. The applications of these partial continuations, like the application

$$(\lambda x.\,\mathcal{A}\,x)\,(x\,+\,3)$$

above, cannot be simplified because the argument is not a value. The relation between $\lambda x.\,\mathcal{A}\,E[x]$ and E^* has been investigated in (Felleisen *et al.*, 1987). This relation, written $\approx_p$ in (Felleisen *et al.*, 1987), turns out to be expressible from β_v and the following two additional rules:

$$\begin{array}{lll} \beta_\Omega: & (\lambda x.\,\mathcal{A}\,E[x])\,M & \to \quad \mathcal{A}\,E[M] \\ \mathcal{C}_{idem}: & \mathcal{C}\,(\lambda c.\,\mathcal{C}\,M) & \to \quad \mathcal{C}\,(\lambda c.\,M\,(\lambda x.\,\mathcal{A}\,x)) \end{array}$$

Both rules are observationally sound (especially, the rule $\mathcal{C}_{idem}$ will be discussed in Section 2.3). This leads to the following reformulation of Theorem 4.7 in (Felleisen *et al.*, 1987) (we need Proposition 2.2 as the original result is stated for $\mapsto_{\lambda_\mathcal{C}}$ with $\mathcal{A}bort$, i.e. with the operational rules $\mathcal{C}_{T_E}^{\mathcal{A}bort}$ and $\mathcal{A}bort_{T_E}$):

Theorem 2.9 (*Simul. of oper. sem. by weak-head red. for initial theory*)
$M \mapsto\!\!\!\!\to_{\lambda_\mathcal{C}} V$ iff $M \overset{wh}{\triangleright}{}_c^* V'$ for some V' such that $V' \to\!\!\!\!\to_{\beta_\Omega \mathcal{C}_{idem} \beta_v} V$.
Especially, if V is $\mathcal{C}$-free, $M \mapsto\!\!\!\!\to_{\lambda_\mathcal{C}} V$ iff $M \overset{wh}{\triangleright}{}_c^* V$.

Example 2.10 (*A $\lambda_\mathcal{C}$-term and its evaluation and weak-head reduction*)
Weak-head reduction of our example term is able to reach the value produced by the operational semantics. Consider instead the term $\mathcal{C}\,(\lambda k.\,k\,(\lambda x.\,k))\,z$. According to the operational semantics one has:

$$\mathcal{C}\,(\lambda k.\,k\,(\lambda x.\,k))\,z \mapsto\!\!\!\!\to_{\lambda_\mathcal{C}} \lambda f.\,\mathcal{A}\,(f\,z)$$

With respect to the weak-head reduction for $\triangleright_c$ one has:

$$\begin{array}{ll} \mathcal{C}\,(\lambda k.\,k\,(\lambda x.\,k))\,z & \overset{wh}{\to}_c \quad \mathcal{C}_L \\ \mathcal{C}\,(\lambda c.\,(\lambda k.\,k\,(\lambda x.\,k))\,(\lambda f.\,\mathcal{A}\,(c\,(f\,z)))) & \triangleright_{\mathcal{C}_T} \\ (\lambda c.\,(\lambda k.\,k\,(\lambda x.\,k))\,(\lambda f.\,\mathcal{A}\,(c\,(f\,z))))\,(\lambda x.\,\mathcal{A}\,x) & \overset{wh}{\triangleright}{}_c^* \\ \lambda f.\,\mathcal{A}\,((\lambda x.\,\mathcal{A}\,x)\,(f\,z)) & \end{array}$$

To obtain the value of the evaluator one proceeds with the additional rules:

$$\begin{array}{lcl} \lambda f.\,\mathcal{A}\,((\lambda x.\,\mathcal{A}\,x)\,(f\,z)) & \to & \beta_\Omega \\ \lambda f.\,\mathcal{A}\,(\mathcal{A}\,(f\,z)) & \to & \mathcal{C}_{idem} \\ \lambda f.\,\mathcal{A}\,((\lambda\_.\,f\,z)\,(\lambda x.\,\mathcal{A}\,x)) & \to & \beta_v \\ \lambda f.\,\mathcal{A}\,(f\,z) & & \end{array}$$

Note that $\overset{wh}{\to}_c$ (i.e. without $\triangleright_c$) does not reduce the above term to a value. ∎

2.2.4 Weak-Head Standardization

Theorem 3.10 in (Felleisen *et al.*, 1986) gives a general standardization result for $\triangleright_c^*$. We give below its restriction to the case of reduction to a value.

Theorem 2.11 (*Weak-head standardization for* $\triangleright_c^*$)
$M \triangleright_c^* V$ iff $M \overset{wh}{\triangleright_c^*} V'$, where $V' \twoheadrightarrow_c V$.

Proof
From the general standardization theorem in (Felleisen *et al.*, 1986) and the assumption that a standard reduction leading to a value strictly extends weak-head reduction. Note that in general, for this latter assumption to be true it requires some redesign of the notion of standardization. See the remark below. □

Remark 2.12
There is a small flaw in the definition of standard reduction used in (Felleisen *et al.*, 1986). This flaw actually already occurs in Plotkin's definition of standard reduction (Plotkin, 1975) on which (Felleisen *et al.*, 1986) relies. Plotkin's notion of standard reduction is not deterministic and it does not satisfy the property that a standard reduction necessarily extends weak-head reduction. Assume for instance that $M \overset{wh}{\to}_c M'$ and $N \overset{wh}{\to}_c N'$. Then, the two following distinct reduction paths are standard with respect to Plotkin-style definition of standardization:

$$\begin{array}{llllll} (\lambda y.M)\,N & \to_c & (\lambda y.M)\,N' & \to_c & (\lambda y.M')\,N' \\ (\lambda y.M)\,N & \to_c & (\lambda y.M')\,N & \to_c & (\lambda y.M')\,N' \end{array}$$

The first derivation is standard because it reduces first a weak-head redex and the second is standard by congruence of standardization with respect to application. Only the first one extends weak-head reduction. A solution to the problem is to restrict congruence with respect to application to congruence with respect to evaluation contexts.

2.3 Felleisen and Hieb's Reduction Semantics

The revised λ_c theory in (Felleisen & Hieb, 1992) characterizes the uses of $\mathcal{C}_T$ that are valid in any evaluation context. These uses are captured by two new rules called $\mathcal{C}_{idem}$ and $\mathcal{C}_{top}$. This leads to the new context-compatible reduction system $\to$ presented in Figure 3. We write $\twoheadrightarrow$ for its reflexive-transitive closure and $=$ for its reflexive-symmetric-transitive closure.

$$
\begin{array}{lllll}
\beta_v: & (\lambda x.\, M)\, V & \to & M\,[V/x] \\
\mathcal{C}_L: & (\mathcal{C}\, M)\, N & \to & \mathcal{C}\,(\lambda c.\, M\,(\lambda f.\, \mathcal{A}\,(c\,(f\, N)))) \\
\mathcal{C}_R: & V\,(\mathcal{C}\, M) & \to & \mathcal{C}\,(\lambda c.\, M\,(\lambda x.\, \mathcal{A}\,(c\,(V\, x)))) \\
\mathcal{C}_{idem}: & \mathcal{C}\,(\lambda c.\, \mathcal{C}\, M) & \to & \mathcal{C}\,(\lambda c.\, M\,(\lambda x.\, \mathcal{A}\, x)) \\
\mathcal{C}_{top}: & \mathcal{C} M & \to & \mathcal{C}\,(\lambda c.\, M\,(\lambda x.\, \mathcal{A}\,(c\, x)))
\end{array}
$$

Fig. 3. Reduction rules of call-by-value λ_c (Felleisen and Hieb)

If after some uses of the rules $\mathcal{C}_L$ and $\mathcal{C}_R$, another control operator is reached, $\mathcal{C}_{idem}$ applies the abort continuation. At any point, it is possible to use $\mathcal{C}_{top}$ to start applying M to part of the captured context and then continue lifting the outer $\mathcal{C}$ to accumulate more of the context. As for the operational rules, the right-hand sides of the reduction rules contain the abort operation. Indeed, the main use of rule $\mathcal{C}_{top}$ is to surround each invocation of a continuation with the abort operation. $\mathcal{C}_{top}$ turns what looks like a regular function call into a continuation's invocation. For example, in the term $\mathcal{C}\,(\lambda c.\, 1 + c\; 2 + 3)$ continuation c is invoked using the normal syntax for function application. However, after $\mathcal{C}_{top}$, the application of the continuation is surrounded by the abort operation:

$$\mathcal{C}\,(\lambda c.\, 1 + c\; 2 + 3) \to \mathcal{C}\,(\lambda k.\, (\lambda c.\, 1 + c\; 2 + 3)\,(\lambda x.\, \mathcal{A}\,(k\, x))) \twoheadrightarrow \mathcal{C}\,(\lambda k.\, 1 + \mathcal{A}\,(k\, 2) + 3)$$

Example 2.13 (*Reduction of* $\mathcal{C}\,(\lambda c.\, 1 + c\; 2 + (1 + 1)) + 3$)

$$
\begin{array}{lll}
\mathcal{C}\,(\lambda c.\, 1 + c\; 2 + (1 + 1)) + 3 & \to & \mathcal{C}_L \\
\mathcal{C}\,(\lambda c'.\, (\lambda c.\, 1 + c\; 2 + (1 + 1))\,(\lambda x.\, \mathcal{A}\,(c'\,(x + 3)))) & \to & \beta_v \\
\mathcal{C}\,(\lambda c'.\, 1 + ((\lambda x.\, \mathcal{A}\,(c'\,(x + 3)))\; 2) + (1 + 1)) & \to & \beta_v \\
\mathcal{C}\,(\lambda c'.\, 1 + \mathcal{A}\,(c'(2 + 3)) + (1 + 1)) & \twoheadrightarrow & \\
\mathcal{C}\,(\lambda c'.\, \mathcal{A}\,(c'\,(2 + 3))) & \to & \mathcal{C}_{idem} \\
\mathcal{C}\,(\lambda c'.\, (\lambda \_.\, c'\,(2 + 3))\,(\lambda x.\, \mathcal{A}\, x)) & \twoheadrightarrow & \\
\mathcal{C}\,(\lambda c'.\, c'\, 5) & &
\end{array}
$$

∎

Notice that there is no reduction rule that allows one to reduce the above term to 5, as it happens according to the operational semantics and the original theory. Applications of rule $\mathcal{C}_{top}$ does not help:

$$\mathcal{C}\,(\lambda c'.\, c'\, 5) \;\twoheadrightarrow\; \mathcal{C}\,(\lambda c.\, \mathcal{A}\,(c\, 5))$$

Remark 2.14
The problem with rule $\mathcal{C}_{top}$ is that even in the simply-typed case, it makes the reduction system not strongly normalizable:

$$\mathcal{C}\, y \to \mathcal{C}\,(\lambda c.\, y\,(\lambda x.\, \mathcal{A}\,(c\, x))) \to \mathcal{C}\,(\lambda c'.\, (\lambda c.\, y\,(\lambda x.\, \mathcal{A}\,(c\, x)))\,(\lambda x.\, \mathcal{A}\,(c'\, x))) \to \cdots$$

Theorem 2.15
The λ_c-calculus is confluent.

Proof
This is proved in Theorem 3.14 of (Felleisen & Hieb, 1992) by first showing the

confluence of the following reduction system (called $\lambda_{c'}$):

$$\begin{array}{lll}
\beta_v: & (\lambda x.\, M)\, V & \to \quad M\,[V/x] \\
\mathcal{C}'_L: & (\mathcal{C}\,(\lambda k.\, M))\, N & \to \quad \mathcal{C}\,(\lambda c.\, M\,[(\lambda f.\, \mathcal{A}\,(c\,(f\,N)))/k]) \\
\mathcal{C}'_R: & V\,(\mathcal{C}\,(\lambda k.\, M)) & \to \quad \mathcal{C}\,(\lambda c.\, M\,[\lambda x.\, \mathcal{A}\,(c\,(V\,x))/k]) \\
\mathcal{C}'_{idem}: & \mathcal{C}\,(\lambda c.\, \mathcal{C}\,(\lambda k.\, M)) & \to \quad \mathcal{C}\,(\lambda c.\, M\,[(\lambda x.\, \mathcal{A}\,x)/k]) \\
\mathcal{C}'_{top}: & \mathcal{C}\,(\lambda k.\, M) & \to \quad \mathcal{C}\,(\lambda c.\, M\,[(\lambda x.\, \mathcal{A}\,(c\,x))/k]) \\
\mathcal{C}_{top}: & \mathcal{C}\,M & \to \quad \mathcal{C}\,(\lambda c.\, M\,(\lambda x.\, \mathcal{A}\,(c\,x)))
\end{array}$$

$\lambda_{c'}$ has the same reflexive-transitive closure of λ_c, therefore confluence of λ_c follows. □

Remark 2.16
Even though the reduction rules can be applied in any context, they do have a strategy embedded in them. For example, one cannot reduce the following term

$$(\mathcal{A}\,2)\;(\mathcal{A}\,5)$$

to both $\mathcal{A}\,2$ and $\mathcal{A}\,5$, thus contradicting the confluence result. The above term reduces to $\mathcal{A}\,2$ but cannot reduce to $\mathcal{A}\,5$. According to the $\mathcal{C}_R$ rule, the argument $\mathcal{A}\,5$ can only be lifted after the function part is reduced to a value. This reflects a left-to-right evaluation strategy. Reduction rules which enforce a right-to-left evaluation order are as follows:

$$\begin{array}{lll}
M\,(\mathcal{C}\,N) & \to & \mathcal{C}\,(\lambda c.\, N\,(\lambda x.\, \mathcal{A}\,(c\,(M\,x)))) \\
(\mathcal{C}\,M)\,V & \to & \mathcal{C}\,(\lambda c.\, M\,(\lambda f.\, \mathcal{A}\,(c\,(f\,V))))
\end{array}$$

2.3.1 Weak-Head Reduction

Felleisen and Hieb give a definition of weak-head reduction for the revised theory[1]. This is not as obvious as for Felleisen-Friedman-Kohlbecker-Duba's theory because the removal of $\mathcal{C}_T$ makes the connection with the operational semantics less tight. Felleisen and Hieb give the following ad hoc definition that mimics Felleisen-Friedman-Kohlbecker-Duba weak-head reduction. The definition is in two steps. First, we say that M *$\mathcal{C}$-weakly head reduces* to N, written $M \overset{\mathcal{C}\text{-}wh}{\to} N$, in the following cases:

- M has the form $\mathcal{C}\,(\lambda k.\, E[P])$, where P is a β_v, $\mathcal{C}_L$ or $\mathcal{C}_R$ redex that reduces to Q, and N is $\mathcal{C}\,(\lambda k.\, E[Q])$
- M has the form $\mathcal{C}\,(\lambda k.\, \mathcal{C}\,P)$ which is a $\mathcal{C}_{idem}$ redex and N is $\mathcal{C}\,(\lambda k.\, P\,\lambda x.\, \mathcal{A}\,x)$.

Note that the $\mathcal{C}$-weakly head reduction never applies $\mathcal{C}_{top}$ but it does reduce the top-level $\mathcal{C}_{idem}$ redex. Moreover, it reduces under a $\mathcal{C}$-abstraction. We write $\overset{\mathcal{C}\text{-}wh}{\twoheadrightarrow}$ for the reflexive-transitive closure of $\overset{\mathcal{C}\text{-}wh}{\to}$. Then, M is said to *iteratively weakly head reduce* to N, written $M \overset{wh}{\twoheadrightarrow} N$, when either $M \overset{wh}{\twoheadrightarrow}_c N$ or, for some P, $M \overset{wh}{\twoheadrightarrow}_c \mathcal{C}\,P \to_{\mathcal{C}_{top}} \mathcal{C}\,(\lambda k.\, P\,\lambda x.\, \mathcal{A}\,(k\,x)) \overset{\mathcal{C}\text{-}wh}{\to} N$, where $\overset{wh}{\twoheadrightarrow}_c$ is as in Section 2.2.2.

[1] Weak-head reduction is called standard reduction function in Felleisen and Hieb (Felleisen & Hieb, 1992).

Example 2.17 (*Iterative weak-head reduction of* $\mathcal{C}\,(\lambda c.\,1\,+\,c\,2\,+\,(1\,+\,1))\,+\,3$)
We write $\mathcal{A}_x^k$ and $\mathcal{A}_x$ for the continuations $\lambda x.\,\mathcal{A}\,(k\,x)$ and $\lambda x.\,\mathcal{A}\,x$, respectively. First, one lifts the control operator to the top-level:

$$\begin{array}{l}\mathcal{C}\,(\lambda c.\,1\,+\,c\,2\,+\,(1\,+\,1))\,+\,3 \overset{wh}{\to}_c\\ \mathcal{C}\,(\lambda c'.\,(\lambda c.\,1\,+c\,2\,+\,(1\,+\,1))\,(\lambda x.\,\mathcal{A}\,(c'\,(x\,+\,3))))\end{array}$$

$\mathcal{C}_{top}$ is applied next:

$$\begin{array}{l}\mathcal{C}\,(\lambda c'.\,(\lambda c.\,1\,+c\,2\,+\,(1\,+\,1))\,(\lambda x.\,\mathcal{A}\,(c'\,(x\,+\,3)))) \to_{\mathcal{C}_{top}}\\ \mathcal{C}\,(\lambda k.\,(\lambda c'.\,(\lambda c.\,1\,+c\,2\,+\,(1\,+\,1))\,(\lambda x.\,\mathcal{A}\,(c'\,(x\,+\,3))))\,\mathcal{A}_x^k)\end{array}$$

From this point on $\mathcal{C}_{top}$ is disallowed. One continues with the application of either β_v, $\mathcal{C}_L$ or $\mathcal{C}_R$ under a $\mathcal{C}$-abstraction:

$$\begin{array}{l}\mathcal{C}\,(\lambda k.\,(\lambda c'.\,(\lambda c.\,1\,+c\,2\,+\,(1\,+\,1))\,(\lambda x.\,\mathcal{A}\,(c'\,(x\,+\,3))))\,\mathcal{A}_x^k) \overset{\mathcal{C}\text{-}wh}{\twoheadrightarrow}\\ \mathcal{C}\,(\lambda k.\,\mathcal{C}\,(\lambda r.\,(\lambda q.\,(\lambda\_.\,\mathcal{A}_x^k\,(2\,+\,3))\,(\lambda z.\,\mathcal{A}\,(q\,(z\,+\,(1\,+\,1)))))\,(\lambda w.\,\mathcal{A}\,(r\,(1\,+\,w)))))\end{array}$$

At this point, $\mathcal{C}_{idem}$ is applied:

$$\begin{array}{l}\mathcal{C}\,(\lambda k.\,\mathcal{C}\,(\lambda r.\,(\lambda q.\,(\lambda\_.\,\mathcal{A}_x^k\,(2\,+\,3))\,(\lambda z.\,\mathcal{A}\,(q\,(z\,+\,(1\,+\,1)))))\,(\lambda w.\,\mathcal{A}\,(r\,(1\,+\,w))))) \overset{\mathcal{C}\text{-}wh}{\to}\\ \mathcal{C}\,(\lambda k.\,(\lambda r.\,(\lambda q.\,(\lambda\_.\,\mathcal{A}_x^k\,(2\,+\,3))\,(\lambda z.\,\mathcal{A}\,(q\,(z\,+\,(1\,+\,1)))))\,(\lambda w.\,\mathcal{A}\,(r\,(1\,+\,w))))\,\mathcal{A}_x)\end{array}$$

We continue the weak-head reduction under a $\mathcal{C}$-abstraction:

$$\begin{array}{l}\mathcal{C}\,(\lambda k.\,(\lambda r.\,(\lambda q.\,(\lambda\_.\,\mathcal{A}_x^k\,(2\,+\,3))\,(\lambda z.\,\mathcal{A}\,(q\,(z\,+\,(1\,+\,1)))))\,(\lambda w.\,\mathcal{A}\,(r\,(1\,+\,w))))\,\mathcal{A}_x) \overset{\mathcal{C}\text{-}wh}{\twoheadrightarrow}\\ \mathcal{C}\,(\lambda k.\,\mathcal{A}\,(k\,5))\end{array}$$

One last $\mathcal{C}_{idem}$ application leads to the answer:

$$\begin{array}{ll}\mathcal{C}\,(\lambda k.\,\mathcal{A}\,(k\,5)) & \overset{\mathcal{C}\text{-}wh}{\to}\\ \mathcal{C}\,(\lambda k.\,(\lambda\_.\,k\,5)\,(\lambda x.\,\mathcal{A}\,x)) & \overset{\mathcal{C}\text{-}wh}{\to}\\ \mathcal{C}\,(\lambda k.\,k\,5) & \end{array}$$

Comparing this reduction with the one in Example 2.4, notice how the first $\triangleright_{\mathcal{C}_T}$ corresponds to a $\mathcal{C}_{top}$ step, whereas the other two occurrences correspond to $\mathcal{C}_{idem}$ steps. ∎

As pointed out earlier, the iterative weak-head reduction does not produce the value that the evaluator would produce. The problem is that there is no way to get rid of the outermost $\mathcal{C}$. To that end, we introduce the following notion: M is said to *evaluate* to a value V iff

- $M \overset{wh}{\twoheadrightarrow} V$; or
- $M \overset{wh}{\twoheadrightarrow} \mathcal{C}\,(\lambda k.\,k\,(V_k[\lambda x.\,\mathcal{A}\,(k\,x)/k]))$ and $V \equiv V_k[\lambda x.\,\mathcal{A}\,x/k]$; or
- $M \overset{wh}{\twoheadrightarrow} \mathcal{C}\,(\lambda k.\,V_k[\lambda x.\,\mathcal{A}\,(k\,x)/k])$ and $V \equiv V_k[\lambda x.\,\mathcal{A}\,x/k]$.

Example 2.18
We would say that our running example evaluates to 5. We also say that $\mathcal{C}\,(\lambda k.\,k)$ evaluates to $\lambda x.\,\mathcal{A}\,x$ since:

$$\mathcal{C}\,(\lambda k.\,k) \to_{\mathcal{C}_{top}} \mathcal{C}\,(\lambda k.\,(\lambda k.\,k)\,(\lambda x.\,\mathcal{A}\,(k\,x))) \overset{\mathcal{C}\text{-}wh}{\to} \mathcal{C}\,(\lambda k.\,\lambda x.\,\mathcal{A}\,(k\,x))$$

and $\lambda x.\,\mathcal{A}\,(k\,x) \equiv k\,[\lambda x.\,\mathcal{A}\,(k\,x)/k]$ and $\lambda x.\,\mathcal{A}\,x \equiv k\,[\lambda x.\,\mathcal{A}\,x/k]$. ∎

The theorem below follows from Theorem 3.9 in (Felleisen & Hieb, 1992). Note that the mapping of the reduction sequences is one-to-one: the unique $\mathcal{C}_{top}$ step, if any, maps to a $\mathcal{C}_T$ step and all $\mathcal{C}_{idem}$ steps map to $\mathcal{C}_T$ steps too.

Theorem 2.19 (*Corresp. between initial and revised weak-head reduction*)
$M \overset{wh}{\triangleright}{}^*_c V$ iff M evaluates to V.

Combined with Theorem 2.9, we finally get the following simulation of the operational semantics:

Corollary 2.20 (*Simul. of oper. sem. by weak-head red. for the revised theory*)
$M \mapsto\!\!\!\!\rightarrow_{\lambda_C} V$ iff one of the following cases happens:

- $M \overset{wh}{\rightarrow\!\!\!\!\rightarrow} V'$ where $V' \rightarrow\!\!\!\!\rightarrow_{\beta_\Omega \mathcal{C}_{idem} \beta_v} V$
- $M \overset{wh}{\rightarrow\!\!\!\!\rightarrow} \mathcal{C}\,(\lambda k.\, k\, V_k[\lambda x.\, \mathcal{A}\,(k\,x)/k])$ where $V_k[\lambda x.\, \mathcal{A}\,x/k] \rightarrow\!\!\!\!\rightarrow_{\beta_\Omega \mathcal{C}_{idem} \beta_v} V$
- $M \overset{wh}{\rightarrow\!\!\!\!\rightarrow} \mathcal{C}\,(\lambda k.\, V_k[\lambda x.\, \mathcal{A}\,(k\,x)/k])$ where $V_k[\lambda x.\, \mathcal{A}\,x/k] \rightarrow\!\!\!\!\rightarrow_{\beta_\Omega \mathcal{C}_{idem} \beta_v} V$.

Example 2.21

- Consider the term $\mathcal{C}\,(\lambda k.\, k\,(\lambda z.\, k))$, one has:

$$\mathcal{C}\,(\lambda k.\, k\,(\lambda z.\, k)) \mapsto\!\!\!\!\rightarrow_{\lambda_C} (\lambda z.\, \lambda x.\, A\,x)$$

  Whereas, with respect to the reduction semantics:

$$\mathcal{C}\,(\lambda k.\, k\,(\lambda z.\, k)) \overset{wh}{\rightarrow\!\!\!\!\rightarrow} \mathcal{C}\,(\lambda k.\, k\,(\lambda z.\, \lambda x.\, \mathcal{A}\,(k\,x))) \equiv \mathcal{C}\,(\lambda k.\, k\,((\lambda z.\, k)[\lambda x.\, \mathcal{A}\,(k\,x)/k]))$$

  and

$$(\lambda z.\, \lambda x.\, A\,x) \equiv (\lambda z.\, k)[\lambda x.\, \mathcal{A}\,x/k]$$

- Consider the term $\mathcal{C}\,(\lambda k.\, k\,(\lambda x.\, k))\,z$ of Example 2.10, one has:

$$\begin{array}{ll} \mathcal{C}\,(\lambda k.\, k\,(\lambda x.\, k))\,z & \overset{wh}{\rightarrow}_c \\ \mathcal{C}\,(\lambda c.\, (\lambda k.\, k\,(\lambda x.\, k))\,(\lambda f.\, \mathcal{A}\,(c\,(f\,z)))) & \rightarrow_{\mathcal{C}_{top}} \\ \mathcal{C}\,(\lambda c.\, (\lambda c.\, (\lambda k.\, k\,(\lambda x.\, k))\,(\lambda f.\, \mathcal{A}\,(c\,(f\,z))))\,(\lambda x.\, \mathcal{A}\,(c\,x))) & \overset{\mathcal{C}\text{-}wh}{\rightarrow\!\!\!\!\rightarrow} \\ \mathcal{C}\,(\lambda c.\, c\,(\lambda f.\, \mathcal{A}\,((\lambda x.\, \mathcal{A}\,(c\,x))\,(f\,z)))) & \end{array}$$

  Where:

$$\lambda f.\, \mathcal{A}\,((\lambda x.\, \mathcal{A}\,(c\,x))\,(f\,z)) \equiv \lambda f.\, \mathcal{A}\,(c\,(f\,z))[\lambda x.\, \mathcal{A}\,(c\,x)/c]$$

  and

$$\lambda f.\, \mathcal{A}\,(c\,(f\,z))[\lambda x.\, \mathcal{A}\,x/c] \equiv \lambda f.\, \mathcal{A}\,((\lambda x.\, \mathcal{A}\,x)\,(f\,z)) \rightarrow\!\!\!\!\rightarrow_{\beta_\Omega \mathcal{C}_{idem} \beta_v} \lambda f.\, \mathcal{A}\,(f\,z)$$

  That answers are not only values is the return consequence of the removal of the computational rule $\mathcal{C}_T$.

∎

Intermezzo 2.22
To simplify the correspondence between the reduction and operational semantics, in (Felleisen & Hieb, 1992) two additional rules were proposed:

$$\begin{array}{lllll} \mathcal{C}_{elim}: & \mathcal{C}\,(\lambda k.\, k\,M) & \rightarrow & M & k \text{ not free in } M \\ \mathcal{C}_E: & E[\mathcal{C}\,M] & \rightarrow & \mathcal{C}\,(\lambda k.\, M\,(\lambda x.\, \mathcal{A}\,(k\,E[x]))) & \end{array}$$

Rule $\mathcal{C}_{elim}$ allows one to reduce our example term $\mathcal{C}\,(\lambda c.\,1 + c\,2 + (1 + 1)) + 3$ to the final value 5. The addition of the rule however breaks the confluence of λ_c:

$$\begin{array}{ccc} \mathcal{C}\,(\lambda k.\,k\,(x\,y)) & \longrightarrow & x\,y \\ \Big\downarrow & & \\ \mathcal{C}\,(\lambda k.\,(\lambda x.\,\mathcal{A}\,(k\,x))\,(x\,y)) & & \end{array}$$

The two diverging computations cannot be brought together. Using $\mathcal{C}_E$ one can naturally express that *any* part of the evaluation context outside an application of $\mathcal{C}$ can be captured and reified as a partial continuation. However, it destroys the confluence of λ_c since one cannot complete the following diagram:

$$\begin{array}{ccc} \mathcal{C}\,(\lambda k.\,k)\,x\,y & \longrightarrow\!\!\!\!\!\rightarrow & \mathcal{C}\,(\lambda q.\,\lambda z.\,\mathcal{A}\,(q\,(z\,x\,y))) \\ \Big\downarrow & & \\ \mathcal{C}\,(\lambda q.\,(\lambda z.\,\mathcal{A}\,((\lambda w.\,\mathcal{A}(q\,(w\,y)))\,(z\,x)))) & & \end{array}$$

Notice that $\mathcal{C}_{T_{E^*}}$ is derivable in Felleisen and Hieb reduction theory extended with β_Ω.

Weak head reduction $\overset{wh}{\twoheadrightarrow}$ is defined globally without relying on a uniform one-step notion of weak-head reduction. Especially, when the answer is not a value, a $\mathcal{C}_{top}$ step has necessarily to occur in between the (possibly empty) $\overset{wh}{\twoheadrightarrow}_c$ path and the (possibly empty) $\overset{\mathcal{C}\text{-}wh}{\twoheadrightarrow}$ path. The following unique context lemma for Felleisen and Hieb's reduction shows when exactly $\mathcal{C}_{top}$ is needed.

Proposition 2.23 (*Unique context lemma for* $\rightarrow_{\lambda c}$)
Let M be a term in λ_c. Exactly one of the following cases happens:

- M has the form V or $\mathcal{C}\,(\lambda k.\,k\,V)$ or $\mathcal{C}\,(\lambda k.\,V)$, in which case we say that M is an *answer*.
- M has a unique decomposition under the form $E[x\,V]$ or $\mathcal{C}\,(\lambda k.\,E[x\,V])$ (with $x \neq k$) or $\mathcal{C}\,(E[x\,V])$ or $\mathcal{C}\,x$ in which case M is said *to have its weak-head reduction stopped*.
- M has a unique decomposition under the form $E[P]$ or $\mathcal{C}\,(\lambda k.\,E[P])$ where P is a β_v, $\mathcal{C}_L$ or $\mathcal{C}_R$ redex.
- M has the form $\mathcal{C}\,(\lambda k.\,\mathcal{C}\,P)$ which is a $\mathcal{C}_{idem}$ redex.
- M has a unique decomposition under the form $\mathcal{C}\,(\lambda k.\,E[k\,V])$ with E non empty in which case only a $\mathcal{C}_{top}$ applies. No other $\mathcal{C}_{top}$ step is further needed.

2.3.2 Weak-Head Reduction in an Abortive Context

Felleisen and Hieb's notion of weak-head reduction has the following weaknesses:

- The notion of final answer is complex.
- The definition of weak-head reduction is not local and requires an explicit $\mathcal{C}_{top}$ step which is not strictly necessary from the point of view of head reduction.

To circumvent these weaknesses, we restate the previous results for terms explicitly evaluated in an abortive context, i.e. for expressions of the form $\mathcal{A}\,M$. Note that in this case, the weak-head reduction is restricted to a $\overset{\mathcal{C}\text{-}wh}{\twoheadrightarrow}$ path and it does not require $\mathcal{C}_{top}$.

Example 2.24 (*Weak-head reduction in an abortive context*)
We will reduce our running term as follows:

$$\begin{array}{ll}
\mathcal{A}\,(\mathcal{C}\,(\lambda c.\,1\,+\,c\,2\,+\,(1\,+\,1))\,+\,3) & \overset{\mathcal{C}\text{-}wh}{\rightarrow} \\
\mathcal{A}\,(\mathcal{C}\,(\lambda c'.\,(\lambda c.\,1\,+c\,2\,+\,(1\,+\,1))\,(\lambda x.\,\mathcal{A}\,(c'\,(x\,+\,3)))))) & \overset{\mathcal{C}\text{-}wh}{\rightarrow}\quad \mathcal{C}_{idem} \\
\mathcal{A}\,((\lambda c'.\,(\lambda c.\,1\,+c\,2\,+\,(1\,+\,1))\,(\lambda x.\,\mathcal{A}\,(c'\,(x\,+\,3))))\,\mathcal{A}_x) & \overset{\mathcal{C}\text{-}wh}{\twoheadrightarrow} \\
\mathcal{A}\,5 &
\end{array}$$

∎

We then get a tighter connection with the initial theory of control (C_T steps map one-to-one to $\mathcal{C}_{idem}$ steps) and hence, thanks to Theorem 2.9, a tighter correspondence with the operational semantics.

Theorem 2.25 (*Corresp. betw. initial and revised w.-h. red. in abortive context*)
$M \overset{wh}{\triangleright_c^*} N$ iff $\mathcal{A}\,M \overset{\mathcal{C}\text{-}wh}{\twoheadrightarrow} \mathcal{A}\,N$.

Corollary 2.26 (*Simul. of oper. sem. by weak-head red. in abortive context*)
$M \mapsto\!\!\twoheadrightarrow_{\lambda\mathcal{C}} V$ iff $\mathcal{A}\,M \overset{\mathcal{C}\text{-}wh}{\twoheadrightarrow} \mathcal{A}\,V'$ where $V' \twoheadrightarrow_{\beta_\Omega \mathcal{C}_{idem} \beta_v} V$.
Especially, if V is $\mathcal{C}$-free, $M \mapsto\!\!\twoheadrightarrow_{\lambda\mathcal{C}} V$ iff $\mathcal{A}\,M \overset{\mathcal{C}\text{-}wh}{\twoheadrightarrow} \mathcal{A}\,V$.

Remark 2.27
To emphasize the role of reasoning in an abortive context, we show that if $M \twoheadrightarrow A$ for A an answer, then $\mathcal{A}\,M \twoheadrightarrow \mathcal{A}\,V$ for some value V:

$$\begin{array}{lll}
\mathcal{A}\,\mathcal{C}\,(\lambda k.\,k\,V) & \rightarrow & \mathcal{C}_{idem} \\
\mathcal{A}\,((\lambda k.\,k\,V)\,(\lambda x.\,\mathcal{A}\,x)) & \rightarrow & \beta_v \\
\mathcal{A}\,((\lambda x.\,\mathcal{A}\,x)\,V\,[\lambda x.\,\mathcal{A}\,x/k]) & \rightarrow & \beta_v \\
\mathcal{A}\,(\mathcal{A}\,V[\lambda x.\,\mathcal{A}\,x/k]) & \rightarrow & \mathcal{C}_{idem} \\
\mathcal{A}\,((\lambda_.\,V\,[\lambda x.\,\mathcal{A}\,x/k])\,(\lambda x.\,\mathcal{A}\,x)) & \rightarrow & \beta_v \\
\mathcal{A}\,(V[\lambda x.\,\mathcal{A}\,x/k]) & &
\end{array}$$

$$\begin{array}{lll}
\mathcal{A}\,\mathcal{C}\,(\lambda k.\,V) & \rightarrow & \mathcal{C}_{idem} \\
\mathcal{A}\,((\lambda k.\,V)(\lambda x.\,\mathcal{A}\,x)) & \rightarrow & \beta_v \\
\mathcal{A}\,(V[\lambda x.\,\mathcal{A}\,x/k]) & &
\end{array}$$

∎

We restate the unique context lemma.

Proposition 2.28 (*Unique context lemma for $\rightarrow_{\lambda\mathcal{C}}$ in abortive context*)
Let M be a term in λ_c. Exactly one of the following cases happens:

- $\mathcal{A}\,M$ has the form $\mathcal{A}\,V$.

- $\mathcal{A}\,M$ has a unique decomposition under the form $\mathcal{A}\,E[P]$ where P is a β_v, $\mathcal{C}_L$ or $\mathcal{C}_R$ redex.
- $\mathcal{A}\,M$ has the form $\mathcal{A}\,(\mathcal{C}\,N)$ which is a $\mathcal{C}_{idem}$ redex.
- $\mathcal{A}\,M$ has the form $\mathcal{A}\,E[x\,V]$ in which case M is said *to have its weak-head reduction in abortive context stopped*.

Intermezzo 2.29
As in Section 2.2.2, one can observe that if $\mathcal{A}\,M$ $\mathcal{C}$-weakly-head reduces to $\mathcal{A}\,N$ by $\mathcal{C}_L$ or $\mathcal{C}_R$, then $\mathcal{A}\,N$ necessarily $\mathcal{C}$-weakly-head reduces further by a sequence (possibly empty) of $\mathcal{C}_L$ or $\mathcal{C}_R$, ended by $\mathcal{C}_{idem}$ and by as many β_v as the number of $\mathcal{C}_L$ or $\mathcal{C}_R$. We write $\rightarrow_{\mathcal{C}^{\mathcal{A}}_{E^*}}$ for such a combination of rules (which generalizes $\mathcal{C}_{idem}$):

$$\mathcal{C}^{\mathcal{A}}_{E^*}: \quad \mathcal{A}\,E[\mathcal{C}\,M] \quad \rightarrow_{\mathcal{C}^{\mathcal{A}}_{E^*}} \quad \mathcal{A}\,(M\,E^*)$$

where E^* is defined as in Section 2.2.2. If moreover M is of the form $\lambda k.\,N$ then $\mathcal{A}\,(M\,E^*)$ reduces further to $\mathcal{A}\,N[E^*/k]$. This leads to the following variant of $\rightarrow_{\mathcal{C}^{\mathcal{A}}_{E^*}}$:

$$\mathcal{C}'^{\mathcal{A}}_{E^*}: \quad \mathcal{A}\,E[\mathcal{C}\,(\lambda k.\,N)] \quad \rightarrow_{\mathcal{C}'^{\mathcal{A}}_{E^*}} \quad \mathcal{A}\,N[E^*/k]$$

Let $\mathcal{C}^-_L$ and $\mathcal{C}^-_R$ be as in Section 2.2.2 and $\mathcal{C}^-_{idem}$ be the restriction of $\mathcal{C}_{idem}$ that applies only when the body of the innermost $\mathcal{C}$ is not an abstraction. Writing $\overset{C\text{-}wh}{\rightarrow}_{\mathcal{C}^{\mathcal{A}}_{E^*}\beta_v}$ for the union of weak-head $\mathcal{C}^{\mathcal{A}}_{E^*}$ and β_v, and $\overset{C\text{-}wh}{\rightarrow}_{\mathcal{C}'^{\mathcal{A}}_{E^*}\mathcal{C}^-_{idem}\mathcal{C}^-_L\mathcal{C}^-_R\beta_v}$ for the union of weak-head $\mathcal{C}'^{\mathcal{A}}_{E^*}$, $\mathcal{C}^-_{idem}$, $\mathcal{C}^-_L$, $\mathcal{C}^-_R$ and β_v, we get the following equivalence:

Proposition 2.30
$\mathcal{A}\,M \overset{C\text{-}wh}{\twoheadrightarrow} \mathcal{A}\,V$ iff $\mathcal{A}\,M \overset{C\text{-}wh}{\rightarrow}_{\mathcal{C}^{\mathcal{A}}_{E^*}\beta_v} \mathcal{A}\,V$ iff $\mathcal{A}\,M \overset{C\text{-}wh}{\twoheadrightarrow}_{\mathcal{C}'^{\mathcal{A}}_{E^*}\mathcal{C}^-_{idem}\mathcal{C}^-_L\mathcal{C}^-_R\beta_v} \mathcal{A}\,V$.

2.4 The λ_c-calculus without the $\mathcal{C}_{top}$ rule: the $\lambda_{c^\star}$-calculus

As observed previosuly, if one reduces terms of the form $\mathcal{A}\,M$ then rule $\mathcal{C}_{top}$ is not needed, its effect is subsumed by the $\mathcal{C}_{idem}$ rule. We let $\lambda_{c^\star}$ stand for the reduction theory without rule $\mathcal{C}_{top}$.

Theorem 2.31
The $\lambda_{c^\star}$-calculus is confluent.

Proof
As pointed out in the proof of confluence for λ_c (Theorem 2.15), Felleisen and Heib prove confluence of an equivalent reduction system, the $\lambda_{c'}$ calculus. In addition, they also state the confluence of $\lambda_{c'}$ without the $\mathcal{C}_{top}$ and $\mathcal{C}'_{top}$ rules. However, we cannot rely on this result to show confluence of $\lambda_{c^\star}$, since the two reduction systems are not equivalent. To simulate a $\mathcal{C}_L$ reduction in $\lambda_{c'}$ one actually needs the $\mathcal{C}_{top}$ rule. Consider the $\lambda_{c^\star}$ reduction:

$$(\mathcal{C}\,x)\,y \rightarrow \mathcal{C}\,(\lambda c.\,x\,\lambda f.\,\mathcal{A}\,(c\,(f\,y)))$$

The simulation in $\lambda_{c'}$ is:

$$\begin{array}{lcl}
(\mathcal{C}\,x)\,y & \to & \mathcal{C}_{top} \\
(\mathcal{C}\,(\lambda c.\,x\,\lambda z.\,\mathcal{A}\,(c\,z)))\,y & \to & \mathcal{C}'_L \\
\mathcal{C}\,(\lambda c.\,x\,\lambda z.\,\mathcal{A}\,((\lambda x.\,\mathcal{A}\,(c\,(x\,y)))\,z)) & \to & \beta_v \\
\mathcal{C}\,(\lambda c.\,x\,\lambda z.\,\mathcal{A}\,(\mathcal{A}\,(c\,(z\,y)))) & \to & \mathcal{C}'_{idem} \\
\mathcal{C}\,(\lambda c.\,x\,\lambda z.\,\mathcal{A}\,(c\,(z\,y)))
\end{array}$$

We therefore give a direct proof of confluence using van Oostrom's method of *decreasing diagrams* (See Appendix A).

As pointed out in the appendix, to deal with the duplication caused by the β_v reduction one works with the notion of parallel reduction. There is an interference between a β_v reduction and a $\mathcal{C}_R$ redex, which as shown below is benign:

$$\begin{array}{ccc}
(\lambda k.\,k\,\mathcal{C}\,(\lambda q.\,q\,x))\,V & \xrightarrow{\beta_v} & V\,\mathcal{C}\,(\lambda q.\,q\,x) \\
\downarrow{\scriptstyle \mathcal{C}_R} & & \downarrow{\scriptstyle \mathcal{C}_R} \\
(\lambda k.\,\mathcal{C}\,(\lambda c.\,(\lambda q.\,q\,x)\,(\lambda x.\,\mathcal{A}\,(c\,(k\,x)))))\,V & \underset{\beta_v}{-\!\!-\!\!\succ} & \mathcal{C}\,(\lambda c.\,(\lambda q.\,q\,x)\,(\lambda x.\,\mathcal{A}\,(c\,(V\,x))))
\end{array}$$

The lifting rules do not interfere with themselves:

$$\begin{array}{ccc}
\mathcal{C}\,(\lambda k.\,k\,\mathcal{C}\,(\lambda q.\,q\,x))\,y & \xrightarrow{\mathcal{C}_L} & \mathcal{C}\,(\lambda c.\,(\lambda k.\,k\,\mathcal{C}\,(\lambda q.\,q\,x))\,(\lambda f.\,\mathcal{A}\,(c\,(f\,y)))) \\
\downarrow{\scriptstyle \mathcal{C}_R} & & \downarrow{\scriptstyle \mathcal{C}_R} \\
\mathcal{C}\,(\lambda k.\,\mathcal{C}\,(\lambda c.\,(\lambda q.\,q\,x)\,(\lambda x.\,\mathcal{A}\,(c\,(k\,x)))))\,y & \underset{\mathcal{C}_L}{-\!\!-\!\!\succ} & M
\end{array}$$

where the common term M is

$$\mathcal{C}\,(\lambda c.\,(\lambda k.\,\mathcal{C}\,(\lambda c.\,(\lambda q.\,q\,x)\,(\lambda x.\,\mathcal{A}\,(c\,(k\,x)))))\,(\lambda f.\,\mathcal{A}\,(c\,(f\,y))))$$

However, the lifting rules interfere with a $\mathcal{C}_{idem}$ reduction:

$$\begin{array}{ccccc}
\mathcal{C}\,(\lambda c.\,\mathcal{C}\,M)\,N & & \xrightarrow{\mathcal{C}_{idem}} & & \mathcal{C}\,(\lambda c.\,M\,\lambda x.\mathcal{A}\,x)\,N \\
\downarrow{\scriptstyle \mathcal{C}_L} & & & & \downarrow{\scriptstyle \mathcal{C}_L} \\
 & & & & M_2 \\
 & & & & \downarrow{\scriptstyle \beta_v} \\
\mathcal{C}\,(\lambda q.\,(\lambda c.\,\mathcal{C}\,M)\,(\lambda f.\,\mathcal{A}\,(q\,(f\,N)))) & \underset{\beta_v}{-\!\!-\!\!\succ} & M_1 & -\!\!-\!\!-\!\!-\!\!-\!\!\succ & M_3
\end{array}$$

where M_1 is

$$\mathcal{C}\,(\lambda q.\,\mathcal{C}\,M\,[(\lambda f.\,\mathcal{A}\,(q\,(f\,N)))/c])$$

M_2 is

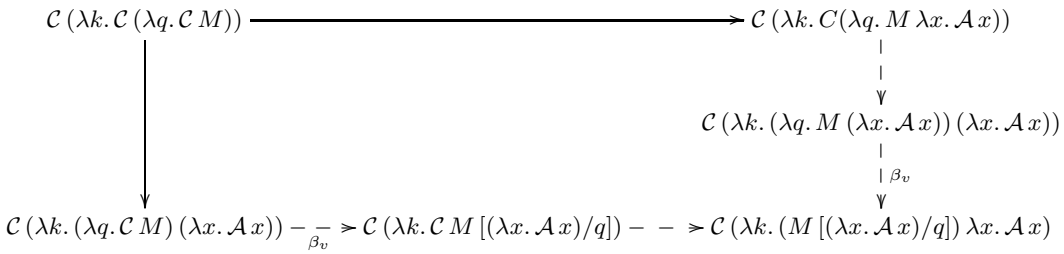
$$\mathcal{C}\,(\lambda q.\,(\lambda c.\,M\,\lambda x.\mathcal{A}\,x)\,(\lambda f.\,\mathcal{A}\,(q\,(f\,N))))$$

and M_3 is

$$\mathcal{C}\,(\lambda q.\,M\,[(\lambda f.\,\mathcal{A}\,(q\,(f\,N)))/c]\,(\lambda x.\,\mathcal{A}\,x))$$

To solve the problem we take the $\mathcal{C}_L, \mathcal{C}_R > \beta_v$.

$\mathcal{C}_{idem}$ interferes with itself:



To make the above diagram decreasing we take $\mathcal{C}_{idem} > \beta_v$. □

2.4.1 Weak-Head Standardization in an Abortive Context

The purpose of this section is to prove a weak-head standardization theorem for the revised notion of control. In (Felleisen & Hieb, 1992) there is a notion of standardization, which however is non-deterministic. We first recall Felleisen and Hieb's results and deduce from it that $\mathcal{C}_{top}$ is useless for weak-head standardization in an abortive context. Our own deterministic weak-head standardization theorem comes next.

Based on (Felleisen & Hieb, 1992), we say that M FH-weakly head reduces to M', written $M \overset{FH\text{-}wh}{\rightarrow} M'$, if there exists an evaluation context E^d such that $M \equiv E^d[N]$ and $M' \equiv E^d[N']$ for N and N' a redex and its contractum, respectively. The evaluation context E^d is defined as follows:

$$E^d ::= E \mid \mathcal{C}\,(\lambda k.\,E)$$

Note that the decomposition of an evaluation context and a redex is not unique. In fact, the term $\mathcal{A}\,\mathcal{C}\,(\lambda k.\,\mathcal{C}\,N)$ contains four standard redexes:

$E^d \equiv \square$	and a $\mathcal{C}_{top}$ redex
$E^d \equiv \square$	and a $\mathcal{C}_{idem}$ redex
$E^d \equiv \mathcal{A}\square$	and a $\mathcal{C}_{top}$ redex
$E^d \equiv \mathcal{A}\square$	and a $\mathcal{C}_{idem}$ redex

Any reduction path can be factorized through a FH-weak-head reduction:

Theorem 2.32 (*FH-standardization*)
$\mathcal{A}\,M \twoheadrightarrow_{\lambda_\mathcal{C}} \mathcal{A}\,V$ iff $\mathcal{A}\,M \overset{FH\text{-}wh}{\twoheadrightarrow}_{\lambda_\mathcal{C}} \mathcal{A}\,V'$ for some V'.

Proof
We rely on the standardization theorem (Theorem 3.16) in (Felleisen & Hieb, 1992), which itself directly relies on Plotkin (Plotkin, 1975) for its proof. Felleisen and Hieb's standardization theorem states that $M \twoheadrightarrow_{\lambda_\mathcal{C}} N$ iff $M \overset{s}{\twoheadrightarrow} N$, where $M \overset{s}{\twoheadrightarrow} N$ is defined by the following clauses:

- $M \overset{FH\text{-}wh}{\twoheadrightarrow} N$ implies $M \overset{s}{\twoheadrightarrow} N$
- $M \overset{s}{\twoheadrightarrow} N$ and $M' \overset{s}{\twoheadrightarrow} N'$ implies $M\,M' \overset{s}{\twoheadrightarrow} N\,M' \overset{s}{\twoheadrightarrow} N\,N'$
- $M \overset{s}{\twoheadrightarrow} N$ implies $\lambda x.M \overset{s}{\twoheadrightarrow} \lambda x.N$ and $\mathcal{C}\,M \overset{s}{\twoheadrightarrow} \mathcal{C}\,N$

From Felleisen and Hieb's standardization theorem we obtain $\mathcal{A}\,M \overset{FH\text{-}wh}{\twoheadrightarrow}_{\lambda_\mathcal{C}} \mathcal{A}\,V$ iff $\mathcal{A}\,M \overset{FH\text{-}wh}{\twoheadrightarrow}_{\lambda_\mathcal{C}} \mathcal{A}\,N \overset{s}{\twoheadrightarrow} \mathcal{A}\,V' \overset{s}{\twoheadrightarrow} \mathcal{A}\,V$ with $N \overset{FH\text{-}wh}{\twoheadrightarrow} V'$. By definition of $\overset{FH\text{-}wh}{\twoheadrightarrow}$, $\mathcal{A}\,N \overset{FH\text{-}wh}{\twoheadrightarrow} \mathcal{A}\,V'$ hence the result. □

Proposition 2.33

If $\mathcal{A}\,M \twoheadrightarrow_{\lambda_\mathcal{C}} \mathcal{A}\,V$ then $\mathcal{A}\,M \twoheadrightarrow_{\lambda_{\mathcal{C}^\star}} \mathcal{A}\,V'$.

Proof

From the standardization of $\lambda_\mathcal{C}$ (Theorem 2.32), $\mathcal{A}\,M \overset{FH\text{-}wh}{\twoheadrightarrow}_{\lambda_\mathcal{C}} \mathcal{A}\,V'$. Next, we prove the following diagram:

$$\begin{array}{ccc} \mathcal{A}\,M & \xrightarrow[\mathcal{C}_{top}]{FH\text{-}wh} & M' \\ & \searrow^{\lambda_{\mathcal{C}^\star}} & \downarrow{\scriptstyle \lambda_{\mathcal{C}^\star}} \\ & & \mathcal{A}\,M'' \end{array} \tag{2}$$

If E^d is empty, one has:

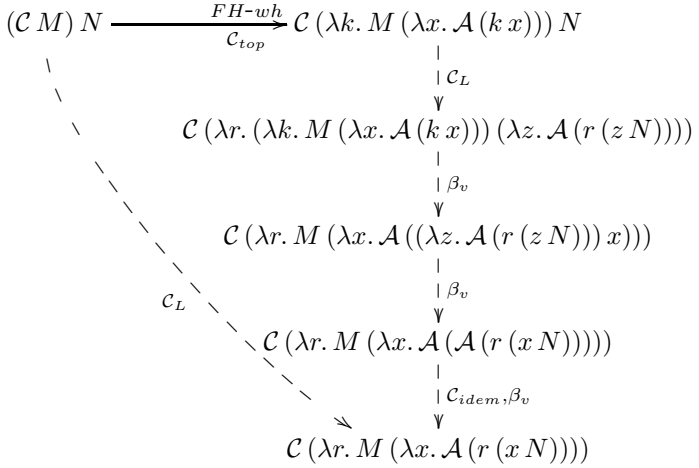
$$\begin{array}{ccc} \mathcal{A}\,M & \longrightarrow & \mathcal{C}\,(\lambda k.\,(\lambda\_.\,M)(\lambda x.\,\mathcal{A}\,(k\,x))) \\ & & \downarrow{\scriptstyle \beta_v} \\ & & \mathcal{A}\,M \end{array}$$

Since the top-level term is of the form $\mathcal{A}\,M$, if E^d is non-empty it must be of the form $\mathcal{A}\,E$. If E is empty:

$$\begin{array}{ccc} \mathcal{A}\,(\mathcal{C}\,M) & \xrightarrow[\mathcal{C}_{top}]{FH\text{-}wh} & \mathcal{A}\,(\mathcal{C}\,(\lambda k.\,M\,(\lambda x.\,\mathcal{A}\,(k\,x)))) \\ & \searrow^{\mathcal{C}_{idem}} & \downarrow{\scriptstyle \mathcal{C}_{idem}} \\ & & \mathcal{A}\,((\lambda k.\,M\,(\lambda x.\,\mathcal{A}\,(k\,x)))(\lambda x.\,\mathcal{A}\,x)) \\ & & \downarrow{\scriptstyle \beta_v} \\ & & \mathcal{A}\,(M\,(\lambda x.\,\mathcal{A}\,((\lambda x.\,\mathcal{A}\,x)\,x))) \\ & & \downarrow{\scriptstyle \beta_v} \\ & \searrow^{\mathcal{C}_{idem}} & \mathcal{A}\,(M\,(\lambda x.\,\mathcal{A}\,(\mathcal{A}\,x))) \\ & & \downarrow{\scriptstyle \mathcal{C}_{idem,\beta_v}} \\ & & \mathcal{A}\,(M\,(\lambda x.\,\mathcal{A}\,x)) \end{array}$$

Otherwise, let the top-level term be of the form $\mathcal{A}\,E[E'[\mathcal{C}\,M]]$ where E' is either

$\Box\, N$ or $V\, N$. If E' is $\Box\, N$ we have:



A similar diagram can be constructed if E' is $V\,\Box$.

From diagram 2 one concludes $\mathcal{A}\, M =_{\lambda_{\mathcal{C}^\star}} \mathcal{A}\, V$. The result then follows from confluence of $\lambda_{\mathcal{C}^\star}$ and the fact that values are stable with respect to $\lambda_{\mathcal{C}^\star}$ reductions. □

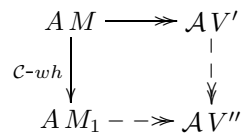
Note that diagram 2 does not hold if the $\mathcal{C}_{top}$ reduction is not standard. For example, with respect to the following reduction:

$$\mathcal{A}\,(\Omega\,(C\, M)) \to \mathcal{A}\,(\Omega\,(\mathcal{C}\,(\lambda k.\, M\, \lambda x.\, \mathcal{A}\,(k\, x))))$$

where Ω stands for a non-terminating computation, one cannot find a common term N such that $\mathcal{A}\,(\Omega\,(C\, M)) \twoheadrightarrow_{\lambda_{\mathcal{C}^\star}} N$ and $\mathcal{A}\,(\Omega\,(\mathcal{C}\,(\lambda k.\, M\, \lambda x.\, \mathcal{A}\,(k\, x)))) \twoheadrightarrow_{\lambda_{\mathcal{C}^\star}} N$.

Theorem 2.34 (*Weak-head standardization for $\twoheadrightarrow_{\lambda_{\mathcal{C}}}$ in an abortive context*)
$\mathcal{A}\, M \twoheadrightarrow_{\lambda_{\mathcal{C}}} \mathcal{A}\, V$ iff $\mathcal{A}\, M \overset{\mathcal{C}\text{-}wh}{\twoheadrightarrow} \mathcal{A}\, V'$, where $V' \twoheadrightarrow_{\lambda_{\mathcal{C}}} V$.

Proof
From Proposition 2.33, $\mathcal{A}\, M \twoheadrightarrow_{\lambda_{\mathcal{C}^\star}} \mathcal{A}\, V'$. We follow the proof technique in (Huet & Lévy, 1991). Let B be the reduction $\mathcal{A}\, M \twoheadrightarrow_{\lambda_{\mathcal{C}^\star}} \mathcal{A}\, V'$. First one shows that the reduction B contracts the descendant of the weak-head redex, say U_1, occurring in $\mathcal{A}\, M$. Then one constructs the projection of the reduction B with respect to the U_1-reduction, *i.e.*, one closes the diagram below



We denote the reduction $A\, M_1 \twoheadrightarrow \mathcal{A}\, V''$ as B/U_1. Since the reduction B/U_1 also leads to an answer, one can proceed by performing the projection $(B/U_1)/U_2$, where U_2 is the weak-head redex contracted by the reduction B/U_1. As before, also $(B/U_1)/U_2$ leads to an answer. To guarantee the termination of such a process one has to show that at each step the weight associated to each reduction decreases.

We explain the weight associated to a reduction through an example. To the following reduction:

$$\begin{array}{ll}
\mathcal{A}\,((\lambda x.\,(x\,z)\;+\;(x\,z))\,(\lambda x.\,2\;+\;2)) & \to \\
\mathcal{A}\,((\lambda x.\,(x\,z)\;+\;(x\,z))\,(\lambda x.\,4)) & \to \\
\mathcal{A}\,((\lambda x.\,4)\,z\;+\;(\lambda x.\,4)\,z) & \to \\
\mathcal{A}\,(4+(\lambda x.\,4)\,z) & \to \\
\mathcal{A}\,(4+4) & \to \\
\mathcal{A}\,8 &
\end{array}$$

we associate the measure $\langle 1,1,1,1,1\rangle$. The projection of the above reduction with respect to the weak-head redex (*i.e.* the outermost β_v redex) is:

$$\begin{array}{ll}
\mathcal{A}\,(((\lambda x.\,2+2)\,z)\;+\;((\lambda x.\,2\;+\;2)\,z)) & \to\to \\
\mathcal{A}\,(((\lambda x.\,4)\,z)\;+\;((\lambda x.\,4)\,z)) & \equiv \\
\mathcal{A}\,(((\lambda x.\,4)\,z)\;+\;((\lambda x.\,4)\,z)) & \to \\
\mathcal{A}\,(4\;+(\lambda x.\,4)\,z) & \to \\
\mathcal{A}\,(4\;+\;4) & \to \\
\mathcal{A}\,8 &
\end{array}$$

The weight associated to the above reduction is $\langle 1,1,1,0,2\rangle$. In other words, the tuple represents the number of times each redex of the original sequence has been duplicated. Using the lexicographic order on tuples we have $\langle 1,1,1,1,1\rangle > \langle 1,1,1,0,2\rangle$. Notice how we count the steps from the answer up to the original term, otherwise, due to duplication of redexes the weight will not decrease. Other than the usual duplication caused by the β_v rule, a duplication in the horizontal line can be caused by the interference between $\mathcal{C}_L$ and $\mathcal{C}_{idem}$, and $\mathcal{C}_{idem}$ and itself, as shown in the proof of confluence of $\lambda_{c^\star}$ (Theorem 2.31). This however can be taken care of by working with $\mathcal{C}'^{\mathcal{A}}_{E^*}$, $\mathcal{C}^-_{idem}$, $\mathcal{C}^-_L$, $\mathcal{C}^-_R$ and β_v, as in Proposition 2.30. The projection of B with respect to a $\mathcal{C}^-_{idem}$, $\mathcal{C}^-_L$ or $\mathcal{C}^-_R$ redex is easy because none of them interfere with $\mathcal{C}_{idem}$. The projection of B with respect to a $\mathcal{C}'^{\mathcal{A}}_{E^*}$ redex is defined as follows. If B does not start with a weak-head redex, this first redex is projected and the rest of B is recursively projected with respect to the $\mathcal{C}'^{\mathcal{A}}_{E^*}$ redex. If B starts with a weak-head redex then the $\mathcal{C}'^{\mathcal{A}}_{E^*}$ reduction necessarily starts with the same weak-head redex (see Proposition 2.28). This redex is removed in B and the projection process continues with the rest of B and the rest of $\mathcal{C}'^{\mathcal{A}}_{E^*}$, i.e. $\mathcal{C}'^{\mathcal{A}}_{E^*}$ with its weak-head redex omitted. If this weak-head redex is $\mathcal{C}_L$ or $\mathcal{C}_R$, omitting it in $\mathcal{C}'^{\mathcal{A}}_{E^*}$ still leaves us with a (shorter) $\mathcal{C}'^{\mathcal{A}}_{E^*}$ redex. If this weak-head redex is $\mathcal{C}_{idem}$ then the $\mathcal{C}'^{\mathcal{A}}_{E^*}$ redex collapses into a sequence of β_v redexes and each of them is recursively removed from B. □

2.5 The Impact of Continuations as Regular Functions

In addition to losing strong normalization (see Remark 2.14), treating continuations as regular functions means that continuations follow the call-by-value discipline: their arguments must be reduced to values before the actual invocation is performed. Consider the following λ_c evaluation:

$$\mathcal{C}\,(\lambda c.\,c\,(2\;+\;1))\mapsto\!\!\!\!\to_{\lambda_c}\;(\lambda c.\,c\,(2\;+\;1))\,(\lambda x.\,\mathcal{A}\,x)\mapsto\!\!\!\!\to_{\lambda_c}\;(\lambda x.\,\mathcal{A}\,x)\,(2\;+\;1)$$

The next evaluation step is to apply the reified continuation $(\lambda x.\, \mathcal{A}\, x)$ to the argument $2 + 1$. However, $2 + 1$ must be simplified to a value first which is wasteful. Indeed, this behavior has a well-known space leak which is demonstrated by the following example:

$$\begin{array}{lcl} \mathit{loop}\ 0 & = & 0 \\ \mathit{loop}\ n & = & \mathcal{C}\,(\lambda c.\, c\,(\mathit{loop}\,(n\text{-}1))) \end{array}$$

When the recursive call to $\mathit{loop}\,(n\text{-}1)$ returns, the continuation c is invoked, which abandons the entire current stack. So the recursive call to *loop* takes place on top of a stack which will never be used. If the recursive call increases the size of the stack before looping, as is the case here, the result is that the stack grows proportional to the depth of recursion, as shown below:

$$\begin{array}{ll} & \mathit{loop}\,3 \\ \mapsto\!\!\!\!\rightarrow_{\lambda_{\mathcal{C}}} & (\lambda x.\, \mathcal{A}\, x)\,(\mathit{loop}\,2) \\ \mapsto\!\!\!\!\rightarrow_{\lambda_{\mathcal{C}}} & (\lambda x.\, \mathcal{A}\,(\lambda x.\, \mathcal{A}\, x)\, x)\,(\mathit{loop}\,1) \\ \mapsto\!\!\!\!\rightarrow_{\lambda_{\mathcal{C}}} & (\lambda x.\, \mathcal{A}\,((\lambda x.\, \mathcal{A}\,((\lambda x.\, \mathcal{A}\, x)\, x))\, x))\,(\mathit{loop}\,0) \end{array}$$

Requiring that the argument of a continuation be a value forces one to evaluate the argument in some continuation and then erase this continuation, instead of the equivalent but more efficient choice of first erasing the continuation and then evaluating the argument (Ganz *et al.*, 1999). One could imagine treating a continuation invocation differently from a regular function call, allowing one to perform the invocation even though the argument is not a value. This would avoid the space leak alluded to above:

$$\mathit{loop}\,3 \mapsto\!\!\!\!\rightarrow (\lambda x.\, \mathcal{A}\, x)\,(\mathit{loop}\,(3\text{-}1)) \mapsto\!\!\!\!\rightarrow \mathcal{A}\,(\mathit{loop}\,(3\text{-}1))$$

Notice how the continuation is invoked instead of reducing the argument. We address these issues together with the lack of strong normalization in the context of the $\lambda_{\mathsf{c_{tp}}}$-calculus, which we introduce in the next section.

Intermezzo 2.35
Matthias Felleisen and his colleagues studied and designed other control operators. In a historical note starting on page 41, Matthias reviews the story of their discovery. In here, we briefly explain `call/cc` and $\mathcal{F}$; their operational rules are as follows:

$$\begin{array}{lcl} E[\texttt{call/cc}\, M] & \mapsto & E[M\,(\lambda x.\, \mathcal{A}\, E[x])] \\ E[\mathcal{F}\, M] & \mapsto & M\,(\lambda x.\, E[x]) \end{array}$$

The rules show that `call/cc` differs from $\mathcal{C}$ in that `call/cc` duplicates the evaluation context. If the captured continuation is not invoked, control goes back to the context surrounding the `call/cc`. For example, with E being the context $\Box + 1$, one has:

$$\texttt{call/cc}\,(\lambda c.\, 4) + 1 \mapsto E[(\lambda c.\, 4)(\lambda x.\, \mathcal{A}\, E[x])] \mapsto E[4] \mapsto 5$$

Whereas, if `call/cc` is replaced with $\mathcal{C}$ one has:

$$\mathcal{C}\,(\lambda c.\, 4) + 1 \mapsto\!\!\!\!\rightarrow 4$$

$\mathcal{F}$ differs from $\mathcal{C}$ in that the invocation of the continuation does not abort the calling

$$
\begin{array}{rcl}
x, a, v, f \in \mathit{Vars} & & \\
k, c \in \mathit{KVars} & & \\
\mathit{KConsts} & = & \{\ \mathsf{tp}\ \} \\
q \in \mathit{KAtoms} & ::= & k \mid \mathsf{tp} \\
M, N \in \mathit{Terms} & ::= & x \mid \lambda x.\, M \mid M\, N \mid \mathcal{C}\,(\lambda k.\, J) \\
V \in \mathit{Values} & ::= & x \mid \lambda x.\, M \\
J \in \mathit{Jumps} & ::= & q\, M \\
E \in \mathit{EvCtxt} & ::= & \Box \mid E\, M \mid V\, E
\end{array}
$$

Fig. 4. Syntax of $\lambda_{c\mathsf{tp}}$

context. In fact, the body of captured continuation contains $E[x]$ instead of $\mathcal{A}\,E[x]$:

$$
\begin{array}{ll}
\mathcal{F}\,(\lambda c.\, 1 + c\,2 + (1 + 1)) + 3 & \mapsto \\
(\lambda c.\, 1 + c\,2 + (1 + 1))\,(\lambda x.\, x + 3) & \mapsto\!\!\!\!\mapsto \\
1 + 5 + (1 + 1) & \mapsto\!\!\!\!\mapsto \\
8 &
\end{array}
$$

3 A Revised Theory of Control: The $\lambda_{c\mathsf{tp}}$-calculus

The $\lambda_{c\mathsf{tp}}$-calculus was presented in a previous work (Ariola & Herbelin, 2003; Ariola *et al.*, 2004). It is basically a call-by-value version of Parigot's $\lambda\mu$-calculus (Parigot, 1992), where μ is renamed into $\mathcal{C}$. It also contains a special constant tp to denote the top-level continuation. The distinguishing feature of the $\lambda_{c\mathsf{tp}}$ calculus is that it reserves a special treatment for the invocation of a continuation, which we refer to as a *jump*.

3.1 Syntax and Operational Semantics

The syntax of $\lambda_{c\mathsf{tp}}$ is in Figure 4. The use of $\mathcal{C}$ is restricted: the argument is always a λ-abstraction which binds a continuation variable. Thus, one cannot write a term such as $\mathcal{C}\,(\lambda k.\,(\lambda x.\,\mathcal{C}\,x)\,k)$. We refer to a term of the form $\mathcal{C}\,(\lambda k.\,J)$ as a $\mathcal{C}$-abstraction. The body of a $\mathcal{C}$-abstraction is restricted to a *jump*. There is a continuation constant tp which denotes the top-level continuation. For example, one would write the λ_c-term $\mathcal{C}\,(\lambda\_.\,5)$ as $\mathcal{C}\,(\lambda\_.\,\mathsf{tp}\,5)$, explicitly indicating the return to the top-level. Variables bound to continuations are distinct from other variables and can only occur in application position, thus one cannot write a term such as $\mathcal{C}\,(\lambda k.\,k)$. Moreover, the invocation of a continuation must be surrounded by a $\mathcal{C}$-abstraction. Instead of writing $(k\,2) + 1$ one is forced to write $\mathcal{C}\,(\lambda\_.\,k\,2) + 1$. This means that the abortive nature of continuations, instead of being reflected in the semantics, is captured in the syntax itself. The $\mathcal{C}$-abstraction surrounding the invocation of a continuation resembles the use of the ML throw construct (Duba *et al.*, 1991). To summarize, aborting a computation (*i.e.*, throwing to the top-level continuation) is written as:

$$\mathcal{A}\,M \quad \triangleq \quad \mathcal{C}\,(\lambda\_.\,\mathsf{tp}M) \qquad \text{(Abbrev. 3)}$$

$$
\begin{array}{rcl}
(x)^\circ & \triangleq & x \\
(\lambda x.\, M)^\circ & \triangleq & \lambda x.\, M^\circ \\
(M\, N)^\circ & \triangleq & M^\circ\, N^\circ \\
(\mathcal{C} M)^\circ & \triangleq & \mathcal{C}\,(\lambda k.\, \mathsf{tp}\,(M^\circ\,(\lambda x.\, \mathcal{T}h\ k\ x)))
\end{array}
$$

Fig. 5. Translation of λ_c in $\lambda_{c_{\mathsf{tp}}}$

$$
\begin{array}{rcl}
(x)^\bullet & \triangleq & x \\
(\lambda x.\, M)^\bullet & \triangleq & \lambda x.\, M^\bullet \\
(M\, N)^\bullet & \triangleq & M^\bullet\, N^\bullet \\
(\mathcal{C}\,(\lambda k.\, J))^\bullet & \triangleq & \mathcal{C}\,(\lambda k.\, J^\bullet) \\
 & & \\
\mathsf{tp}\, M^\bullet & \triangleq & M^\bullet \\
k\, M^\bullet & \triangleq & k\, M^\bullet
\end{array}
$$

Fig. 6. Translation of $\lambda_{c_{\mathsf{tp}}}$ in λ_c

and throwing to a user-defined continuation is written as:

$$\mathcal{T}h\ k\ M \quad \triangleq \quad \mathcal{C}\,(\lambda\_.\, k\, M) \qquad \text{(Abbrev. 4)}$$

The operational semantics of programs is given below:

$$
\begin{array}{lrcl}
\beta_v: & E[(\lambda x.\, M)\, V] & \mapsto_{\lambda_{c_{\mathsf{tp}}}} & E[M\,[V/x]] \\
\mathcal{C}_{T_E}: & E[\mathcal{C}\,(\lambda k.\, k\, M)] & \mapsto_{\lambda_{c_{\mathsf{tp}}}} & E[M\,[\mathsf{tp}\, E/k]] \\
{\mathcal{C}_{T_E}}': & E[\mathcal{C}\,(\lambda k.\, \mathsf{tp}\, M)] & \mapsto_{\lambda_{c_{\mathsf{tp}}}} & M\,[\mathsf{tp}\, E/k]
\end{array}
$$

Unlike the operational semantics for λ_c, these rules make use of a new notion of substitution, called *structural substitution*, which was first introduced in (Parigot, 1992). The general form of structural substitution is written $M\,[q\, E/k]$ (resp. $J\,[q\, E/k]$) and reads as: "replace every jump of the form $k\, N$ in M (resp. J) with the jump $(q\, E[N])$ (and recursively in N)". The substitutions $M\,[\mathsf{tp}\, E/k]$ and $J\,[\mathsf{tp}\, E/k]$ are defined similarly.

The structural substitution $M\,[q\, E/k]$ (resp. $J\,[q\, E/k]$) is inductively defined as follows:

$$
\begin{array}{lcll}
x\,[q\, E/k] & \equiv & x & \\
(\lambda x.\, M)\,[q\, E/k] & \equiv & \lambda x.(M\,[q\, E/k]) & \\
(M\, N)\,[q\, E/k] & \equiv & M\,[q\, E/k]\, N\,[q\, E/k] & \\
\mathcal{C}\,(\lambda k.\, J)\,[q\, E/k] & \equiv & \mathcal{C}\,(\lambda k.\, J) & \\
\mathcal{C}\,(\lambda k'.\, J)\,[q\, E/k] & \equiv & \mathcal{C}\,(\lambda k'.\, J\,[q\, E/k]) & k' \neq k \\
(k\, M)\,[q\, E/k] & \equiv & q\, E[M\,[q\, E/k]] & \\
(k'\, M)\,[q\, E/k] & \equiv & k'\, M\,[q\, E/k] & k' \neq k \\
(\mathsf{tp}\, M)\,[q\, E/k] & \equiv & \mathsf{tp}\, M\,[q\, E/k] &
\end{array}
$$

Note that this notion is not applicable to λ_c since continuations are not necessarily applied to an argument.

The translation of λ_c-terms into the $\lambda_{c_{\mathsf{tp}}}$-calculus is given in Figure 5. If E is a context, its compositional application on each component of the context is written

$$
\begin{array}{lrcl}
\beta_v: & (\lambda x.\, M)\, V & \to & M\,[V/x] \\
\mathcal{C}_L: & \mathcal{C}\,(\lambda k.\, J)\, N & \to & \mathcal{C}\,(\lambda k.\, J\, k\, (\Box\, N)/k]) \\
\mathcal{C}_R: & V\, \mathcal{C}\,(\lambda k.\, J) & \to & \mathcal{C}\,(\lambda k.\, J\,[k\, (V\, \Box)/k]) \\
\mathcal{C}_{idem}: & q\, \mathcal{C}\,(\lambda k.\, J) & \to & J\,[q\, \Box/k]
\end{array}
$$

Fig. 7. Reductions of call-by-value $\lambda_{c\mathsf{tp}}$

E°. **COMMENT by Zena:** The above notion of doverlineE is not clear **END**
Notice how in the $\mathcal{C}$-abstraction case three things are happening:

- the captured continuation is given a name k;
- the implicit jump to the top-level is made explicit;
- the implicit aborting of the context when k is applied is also made explicit.

Based on Abbrev. 1, we have:

$$(\mathcal{A}\, M)^\circ \to_{\beta_v} \mathcal{A}\, M^\circ \tag{5}$$

The translation from a $\lambda_{c\mathsf{tp}}$-term M to a λ_c-term is denoted by $M^\bullet$ and simply corresponds to dropping each reference to tp and interpreting each jump as a regular application. The formal definition is given in Figure 6.

There are two important differences between λ_c and the set of terms coming from the translation: First, for terms in the image of the translation, occurrences of $k\, N$ are necessarily surrounded by some "$\mathcal{C}\,(\lambda k$". Therefore, rule $\mathcal{C}_{top}$ is not needed to evaluate terms coming from $\lambda_{c\mathsf{tp}}$. Second, in the image of the translation each continuation is applied to an argument. This makes the use of structural substitution possible.

Example 3.1 (*The evaluation of our example term*)
The evaluation of the $\lambda_{c\mathsf{tp}}$-term corresponding to the λ_c-term $\mathcal{C}\,(\lambda c.\, 1 + c\, 2 + (1 + 1)) + 3$ is shown below:

$$
\begin{array}{ll}
(\mathcal{C}\,(\lambda c.\, 1 + c\, 2 + (1 + 1)) + 3)^\circ & \triangleq \\
C(\lambda k.\, \mathsf{tp}\,((\lambda c.\, 1 + c\, 2 + (1 + 1))\,(\lambda x.\, \mathit{Th}\ k\ x))) + 3 & \mapsto\!\!\!\!\to_{\lambda_{c\mathsf{tp}}} \\
((\lambda c.\, 1 + c\, 2 + (1 + 1))\,(\lambda x.\, \mathit{Th}\ k\ x))\,[\mathsf{tp}\,(\Box + 3)/k] & \equiv \\
(\lambda c.\, 1 + c\, 2 + (1 + 1))\,(\lambda x.\, \mathcal{A}\,(x + 3)) & \mapsto\!\!\!\!\to_{\lambda_{c\mathsf{tp}}} \\
1 + \mathcal{A}\,(2 + 3) + (1 + 1) & \mapsto\!\!\!\!\to_{\lambda_{c\mathsf{tp}}} \\
5 &
\end{array}
$$

∎

In spite of being defined on structural substitution, the operational semantics given for $\lambda_{c\mathsf{tp}}$ faithfully implements the operational semantics assigned to λ_c. We consider here λ_c with the primitive operator *Abort* and we let $(\mathit{Abort}\, M)^\circ \triangleq \mathcal{A}\, M^\circ$. We have:

Proposition 3.2 (*Simulation of λ_c oper. sem. in $\lambda_{c\mathsf{tp}}$*)
$M \mapsto_{\lambda_c} N$ in λ_c with primitive abort operator iff $M^\circ \mapsto_{\lambda_{c\mathsf{tp}}} N^\circ$ in $\lambda_{c\mathsf{tp}}$.

Proof
The first clause (β-reduction) of each operational semantics trivially correspond. The second clause for $\lambda_{c_{\mathsf{tp}}}$ does not occur by definition of M°. Finally, the second and third clauses for λ_c map to the third clause in $\lambda_{c_{\mathsf{tp}}}$ as shown below:

$$\begin{array}{lll} E[\mathcal{C}\,M]^\circ & \triangleq & E^\circ[\mathcal{C}\,(\lambda k.\,\mathsf{tp}\,(M^\circ\,(\lambda x.\,\mathcal{Th}\;k\;x)))] \\ & \mapsto_{\lambda_c} & (M^\circ\,(\lambda x.\,\mathcal{Th}\;k\;x))[\mathsf{tp}\,E^\circ/k] \\ & \equiv & M^\circ\,(\lambda x.\,\mathcal{A}\,E^\circ[x]) \\ & \triangleq & M\,(\lambda x.\,\mathcal{A}bort\,E[x])^\circ \\ \\ E[\mathcal{A}bort\,M]^\circ & \triangleq & E^\circ[\mathcal{A}\,M^\circ] \\ & \triangleq & E^\circ[\mathcal{C}\,(\lambda\_.\,\mathsf{tp}\,M^\circ)] \\ & \mapsto_{\lambda_c} & M^\circ \end{array}$$

□

By Proposition 2.2 and by iteration of the previous proposition, we get:

Proposition 3.3
$M \mapsto\!\!\!\!\rightarrow_{\lambda_c} V$ with or without primitive abort iff $M^\circ \mapsto\!\!\!\!\rightarrow_{\lambda_{c_{\mathsf{tp}}}} V^\circ$.

$\lambda_{c_{\mathsf{tp}}}$ faithfully simulates λ_c through $^\circ$ but the converse is not true. Compared to λ_c, the structural substitution of $\lambda_{c_{\mathsf{tp}}}$ "optimizes" the application to the continuation as it does not require that the argument of the continuation be evaluated first. Conversely, $\mapsto\!\!\!\!\rightarrow_{\lambda_c}$ "delays" the call to the continuation leading to a possible space leak as discussed in Section 2.5. By reasoning on non-terminating terms, one can show the following:

Proposition 3.4 (*Non simulation of $\lambda_{c_{tp}}$ oper. sem. in λ_c*)
We may have $M \mapsto_{\lambda_{c_{\mathsf{tp}}}} N$ without having $M^\bullet \mapsto\!\!\!\!\rightarrow_{\lambda_c} N^\bullet$

Proof
Consider $M \equiv E[\mathcal{C}\,(\lambda k.\,k\,\Omega)]$ where Ω stands for a non-terminating computation (with no occurrence of k). Then $M \mapsto_{\lambda_{c_{\mathsf{tp}}}} E[\Omega]$ and $M^\bullet \mapsto\!\!\!\!\rightarrow_{\lambda_c} ((\lambda x.\,\mathcal{A}\,(E[x]))\,\Omega)$. Since the evaluation of Ω is non-terminating, $((\lambda x.\,\mathcal{A}\,(E[x]))\,\Omega)$ will never reach $E[\Omega]$. Note that one could even get an irreversible space leak in λ_c when instead the evaluation in $\lambda_{c_{\mathsf{tp}}}$ is simply looping: take $\Omega \equiv Y\,(\lambda x.\,\mathcal{C}\,(\lambda k.\,k\,x))$, with Y some fixpoint operator of λ-calculus (*e.g.* $\lambda f.\,(\lambda y.\,(f\,(y\,y))\,\lambda y.\,(f\,(y\,y)))$). □

However, we have a simulation up to applications of β_Ω.

Proposition 3.5 (*Simulation of $\lambda_{c_{tp}}$ oper. sem. in λ_c up to β_Ω*)
$M \mapsto\!\!\!\!\rightarrow_{\lambda_{c_{\mathsf{tp}}}} V$ iff $M^\bullet \mapsto\!\!\!\!\rightarrow_{\lambda_c} V'$ where V' and V satisfy $V' \twoheadrightarrow_{\beta_\Omega \mathcal{C}_{idem} \beta_v} V^\bullet$

The next remark will allow to simplify the notations used in the proof of Proposition 3.5.

Remark 3.6

(On the ability to express states in the syntax) One motivation for the λ-calculus extended with control is to provide a framework to abstractly study the operational semantics of real languages. With a language like λ_c, the focus is on terms. Especially, the notion of state, though crucial in any actual implementation of a language handling continuations, is not representable in λ_c. With the explicit introduction of the top-level continuation tp, the situation changes. Indeed, tp can be identified with the "bottom of the stack" of stack-based computing devices. Especially, the operational semantics of $\lambda_{c\mathsf{tp}}$ defined above can be equally rewritten as follows:

$$\begin{array}{lllll}
\beta_v: & \mathsf{tp}\,E[(\lambda x.\,M)\,V] & \mapsto_{\lambda_{c\mathsf{tp}}} & \mathsf{tp}\,E[M\,[V/x]] \\
\mathcal{C}_{T_E}: & \mathsf{tp}\,E[\mathcal{C}\,(\lambda k.\,k\,M)] & \mapsto_{\lambda_{c\mathsf{tp}}} & \mathsf{tp}\,E\,[M\,[\mathsf{tp}\,E/k]] \\
{\mathcal{C}_{T_E}}': & \mathsf{tp}\,E[\mathcal{C}\,(\lambda k.\,\mathsf{tp}\,M)] & \mapsto_{\lambda_{c\mathsf{tp}}} & \mathsf{tp}\,M\,[\mathsf{tp}\,E/k]
\end{array}$$

or, more concisely, as:

$$\begin{array}{lllll}
\beta_v: & \mathsf{tp}\,E[(\lambda x.\,M)\,V] & \mapsto_{\lambda_{c\mathsf{tp}}} & \mathsf{tp}\,E[M\,V/x]] \\
\mathcal{C}_{T_E}: & \mathsf{tp}\,E[\mathcal{C}\,(\lambda k.\,J)] & \mapsto_{\lambda_{c\mathsf{tp}}} & J\,[tpcst\,E/k]]
\end{array}$$

More generally, the evaluation semantics could be extended to open computations as follows:

$$\begin{array}{lllll}
\beta_v: & q\,E[(\lambda x.\,M)\,V] & \mapsto_{\lambda_{c\mathsf{tp}}} & q\,E[M\,[V/x]] \\
\mathcal{C}_{T_E}: & q\,E[\mathcal{C}\,(\lambda k.\,J)] & \mapsto_{\lambda_{c\mathsf{tp}}} & J\,[q\,E/k]]
\end{array}$$

Proof of Proposition 3.5. The result is of the same kind as Theorem 2.9 (*i.e.* Th 4.7 of Felleisen-Friedman-Kohlbecker-Duba (Felleisen *et al.*, 1987)). Instead of exhibiting the relation characterizing how the two reduction paths differ, as done in (Felleisen *et al.*, 1987), we reason by nested induction. The only difficulty is to manage the slow down caused by the replacement of structural substitutions by substitutions of reified contexts.

We first prove that $M \mapsto\!\!\!\!\rightarrow_{\lambda_{c\mathsf{tp}}} V$ implies $M^\bullet \mapsto\!\!\!\!\rightarrow_{\lambda_c} V' \to\!\!\!\!\!\rightarrow_{\beta_\Omega \mathcal{C}_{idem} \beta_v} V^\bullet$. We reason by induction on the length of the reduction path. The case of an empty reduction is trivial so we can assume that $M \mapsto_{\lambda_{c\mathsf{tp}}} M' \mapsto\!\!\!\!\rightarrow_{\lambda_{c\mathsf{tp}}} V$ and by the induction hypothesis, we get $M'^\bullet \mapsto\!\!\!\!\rightarrow_{\lambda_c} V' \to\!\!\!\!\!\rightarrow_{\beta_\Omega \mathcal{C}_{idem} \beta_v} V^\bullet$. We focus on the reduction $M \mapsto_{\lambda_{c\mathsf{tp}}} M'$. The case of a β_v contraction is easy as it behaves the same in both $\mapsto\!\!\!\!\rightarrow_{\lambda_{c\mathsf{tp}}}$ and $\mapsto\!\!\!\!\rightarrow_{\lambda_c}$. Let's then assume that M is $E[\mathcal{C}\,(\lambda k.\,J)]$ and M' is $P\,[\mathsf{tp}\,E/k]$ (if J is $\mathsf{tp}\,P$) or M' is $E[P\,[\mathsf{tp}\,E/k]]$ (if J is $k\,P$). On the λ_c side, the reduction is simulated by $M^\bullet \mapsto_{\lambda_c} (\lambda k.\,J^\bullet)\,(\lambda x.\,\mathcal{A}\,E[x]^\bullet) \mapsto_{\lambda_c} J^\bullet\,[\lambda x.\,\mathcal{A}\,E[x]^\bullet/k]$. If moreover J has the form $k\,W$ with W a value, the reduction can progress even further with $J^\bullet\,[\lambda x.\,\mathcal{A}\,E[x]^\bullet/k] \mapsto_{\lambda_c} \mathcal{A}\,E[W]^\bullet\,[\lambda x.\,\mathcal{A}\,E[x]^\bullet/k] \mapsto\!\!\!\!\rightarrow_{\lambda_c} E[W]^\bullet\,[\lambda x.\,\mathcal{A}\,E[x]^\bullet/k]$. To get a uniform notation, we let J^+ be J if J has not the form $k\,W$ and $E[W]$ otherwise. We can then restate the reduction in λ_c as follows: $M^\bullet \mapsto\!\!\!\!\rightarrow_{\lambda_c} J^{+\bullet}\,[\lambda x.\,\mathcal{A}\,E[x]^\bullet/k]$. To use the induction hypothesis we need to lift the reduction $M'^\bullet \mapsto\!\!\!\!\rightarrow_{\lambda_c} V'$, where $M'^\bullet$ can be equally seen as $J^+\,[\mathsf{tp}\,E/k]^\bullet$, into some reduction starting from $J^{+\bullet}\,[\lambda x.\,\mathcal{A}\,E[x]^\bullet/k]$. To this aim, we show that $J^+\,[\mathsf{tp}\,E/k]^\bullet \mapsto\!\!\!\!\rightarrow_{\lambda_c} V'$ implies $J^{+\bullet}\,[\lambda x.\,\mathcal{A}\,E[x]^\bullet/k] \mapsto\!\!\!\!\rightarrow_{\lambda_c} V''^\bullet\,[\lambda x.\,\mathcal{A}\,E[x]^\bullet/k]$ where V' is $V''\,[\mathsf{tp}\,E/k]^\bullet$. Since

$$V''^\bullet\,[\lambda x.\,\mathcal{A}\,E[x]^\bullet/k] \to\!\!\!\!\!\rightarrow_{\beta_\Omega \mathcal{C}_{idem} \beta_v} V''\,[\mathsf{tp}\,E/k]^\bullet \equiv V'$$

the value $V^\bullet$ will eventually be reached.

The auxiliary result is by induction on the length of the reduction path from $J^+\,[\mathsf{tp}\,E/k]^\bullet$ to V'. The case of an empty reduction path is trivial. Otherwise $J^+\,[\mathsf{tp}\,E/k]^\bullet \mapsto_{\lambda_\mathcal{C}} P \twoheadrightarrow_{\lambda_\mathcal{C}} V'$. Necessarily, J^+ has the form $\mathsf{tp}\,E'[\mathcal{C}\,(\lambda k'.\,J')]$ or $\mathsf{tp}\,E'[(\lambda x.\,M)\,N]$ and it reduces to some J''. Hence P has the form $J''\,[\mathsf{tp}\,E/k]^\bullet$ and the same reduction step occurs in $J^{+\bullet}\,[\lambda x.\,\mathcal{A}\,E[x]^\bullet/k]$ leading to $J''^\bullet\,[\lambda x.\,\mathcal{A}\,E[x]^\bullet/k]$. If J'' has not the form $k\,W$, the subsidiary induction hypothesis is directly applicable. Otherwise, we need first to insert a few extra steps to release the context out of its reification: $k\,W^\bullet\,[\lambda x.\,\mathcal{A}\,E[x]^\bullet/k] \mapsto_{\lambda_\mathcal{C}} \mathcal{A}\,E[W]^\bullet\,[\lambda x.\,\mathcal{A}\,E[x]^\bullet/k] \twoheadrightarrow_{\lambda_\mathcal{C}} E[W]^\bullet\,[\lambda x.\,\mathcal{A}\,E[x]^\bullet/k]$.

Conversely, we reason on states and show that for J closed, $J^\bullet \twoheadrightarrow_{\lambda_\mathcal{C}} V'$ implies $J \twoheadrightarrow_{\lambda_{\mathcal{C}\mathsf{tp}}} \mathsf{tp}\,V$ for some value V such that $V' \twoheadrightarrow_{\beta_\Omega \mathcal{C}_{idem} \beta_v} V^\bullet$. This is by induction on the length of the reduction path from $J^\bullet$ to V'. Since J is closed, it has the form $\mathsf{tp}\,M$. The difficult case is when M is $E[\mathcal{C}\,(\lambda k.\,J)]$ in which case $J^\bullet \mapsto_{\lambda_\mathcal{C}} (\lambda k.\,J'^\bullet)\,(\lambda x.\,\mathcal{A}\,E[x]^\bullet) \mapsto_{\lambda_\mathcal{C}} J'^\bullet[\lambda x.\,\mathcal{A}\,E[x]^\bullet/k]$ while $\mathsf{tp}\,M \mapsto_{\lambda_{\mathcal{C}\mathsf{tp}}} J'\,[\mathsf{tp}\,E/k]$. Since the induction hypothesis only gives $J'[\lambda x.\,\mathcal{A}\,E[x]/k] \twoheadrightarrow_{\lambda_{\mathcal{C}\mathsf{tp}}} \mathsf{tp}\,V$ with $V' \twoheadrightarrow_{\beta_\Omega \mathcal{C}_{idem} \beta_v} V^\bullet$, we use a subsidiary induction to show that if $J'\,[\lambda x.\,\mathcal{A}\,E[x]/k] \twoheadrightarrow_{\lambda_{\mathcal{C}\mathsf{tp}}} \mathsf{tp}\,V$ then $J'\,[\mathsf{tp}\,E/k] \twoheadrightarrow_{\lambda_{\mathcal{C}\mathsf{tp}}} \mathsf{tp}\,W\,[\mathsf{tp}\,E/k]$ for some W such that V is $W\,[\lambda x.\,\mathcal{A}\,E[x]]$. The only case which does not directly commute is when J' is $k\,W'$ in which case $J'\,[\lambda x.\,\mathcal{A}\,E[x]/k] \twoheadrightarrow_{\lambda_{\mathcal{C}\mathsf{tp}}} \mathsf{tp}\,(\mathcal{A}\,E[W'])[\lambda x.\,\mathcal{A}\,E[x]/k] \twoheadrightarrow_{\lambda_{\mathcal{C}\mathsf{tp}}} \mathsf{tp}\,E[W']\,[\lambda x.\,\mathcal{A}\,E[x]/k]$ while on the other side we already have $J\,[\mathsf{tp}\,E/k] \equiv \mathsf{tp}\,E[W'][\,\mathsf{tp}\,E/k]$. It remains to observe again that $W[\lambda x.\mathcal{A}\,E[x]/k]^\bullet \twoheadrightarrow_{\beta_\Omega \mathcal{C}_{idem} \beta_v} W[\mathsf{tp}\,E/k]^\bullet$ to finally get $V' \twoheadrightarrow_{\beta_\Omega \mathcal{C}_{idem} \beta_v} W[\mathsf{tp}\,E/k]^\bullet$. ∎

3.2 Reduction Semantics

The reduction semantics is given in Figure 7. Like the original calculus, the rules $\mathcal{C}_L$ and $\mathcal{C}_R$ allow one to *lift* the control operation step-by-step until it reaches a point where it can no longer be lifted. When the control operator reaches a jump to the top-level (rule $\mathcal{C}_{idem}$ with q instantiated with tp), the captured continuation is the trivial continuation modeled by tp. Otherwise, if the control operator reaches a regular continuation variable k, the captured continuation becomes k.

3.2.1 Confluence

Remark 3.7
The $\lambda_{\mathcal{C}\mathsf{tp}}$ reduction rules are overlapping: a $\mathcal{C}_L$ reduction can destroy a $\mathcal{C}_{idem}$ redex, as shown below:

$$\begin{array}{ccc}
\mathcal{C}\,(\lambda k.\,k\,\mathcal{C}\,(\lambda q.\,q\,x))\,y & \xrightarrow[\mathcal{C}_{idem}]{} & \mathcal{C}\,(\lambda k.\,k\,x)\,y \\
\Big\downarrow{\scriptstyle \mathcal{C}_L} & & \Big\downarrow{\scriptstyle \mathcal{C}_L} \\
\mathcal{C}\,(\lambda k.\,k\,(\mathcal{C}\,(\lambda q.\,q\,x)\,y)) & \xrightarrow[\mathcal{C}_L]{} \mathcal{C}\,(\lambda k.\,k\,\mathcal{C}\,(\lambda q.\,q\,(x\,y))) & \xrightarrow[\mathcal{C}_{idem}]{} \mathcal{C}\,(\lambda k.\,k\,(x\,y))
\end{array}$$

To complete the above diagram the newly created $\mathcal{C}_L$ redex has to be reduced, as also observed by Baba *et al.* (Baba *et al.*, 2001) in the context of call-by-value

Parigot's $\lambda\mu$ calculus. This complicates the proof of confluence based on the method of parallel reductions of Tait and Martin-Löf, since the parallel reduction does not satisfy the diamond property. The solution in (Baba *et al.*, 2001) is to introduce the following generalization of $\mathcal{C}_{idem}$:

$$\mathcal{C}_E^J: \quad q\,E[\mathcal{C}\,(\lambda k.\,J)] \quad \to \quad J\,[q\,E/k]$$

The new rule allows one to close the above diagram in one step.

Theorem 3.8
$\lambda_{c_{\mathsf{tp}}}$ is confluent.

Proof
Follows the same steps as the proof of confluence of call-by-value $\lambda\mu$ (Baba *et al.*, 2001). Since $\lambda_{c_{\mathsf{tp}}}$ reductions rules are duplicating and interfering, one considers the alternative reduction system $\underline{\lambda_{c_{\mathsf{tp}}}}$. The calculus $\underline{\lambda_{c_{\mathsf{tp}}}}$ allows the reduction of multiple redexes in one step and contains the generalization of $\mathcal{C}_{idem}$ given in the above remark (see rule $\mathcal{C}_E^J$). The calculi $\lambda_{c_{\mathsf{tp}}}$ and $\underline{\lambda_{c_{\mathsf{tp}}}}$ have the same transitive closure, and $\underline{\lambda_{c_{\mathsf{tp}}}}$ has the diamond property. □

3.2.2 Robustness

The $\lambda_{c_{\mathsf{tp}}}$ reduction system can be also extended with the $\mathcal{C}_{elim}$ rule which eliminates a superfluous jump whose target is the current continuation:

$$\mathcal{C}_{elim}: \quad \mathcal{C}\,(\lambda k.\,k\,M) \to M \qquad k \text{ not free in } M$$

The counterpart of $\mathcal{C}_E$ in $\lambda_{c_{\mathsf{tp}}}$ is the following rule:

$$\mathcal{C}_E: \quad E[\mathcal{C}\,(\lambda k.\,J] \quad \to \quad \mathcal{C}\,(\lambda k.\,J\,[k\,E/k])$$

In contrast with λ_c, $\mathcal{C}_E$ is derivable from $\mathcal{C}_L$ and $\mathcal{C}_R$ in $\lambda_{c_{\mathsf{tp}}}$.

The fact that jumps never occur on the left- or right-hand-side of an application makes the need for a rule like $\mathcal{C}_{top}$ useless. As a consequence, no rule artificially breaks strong normalization (see *e.g.* (Ariola & Herbelin, 2003; Ariola *et al.*, 2005) for a proof of strong normalization in the simply-typed case).

The use of structural substitution avoids also the space leak discussed in Section 2.5. We have:

$$loop\,3 \mapsto_{\lambda_{c_{\mathsf{tp}}}} \mathcal{C}\,(\lambda c.\,c\,(loop\,(3\text{-}1))) \mapsto_{\lambda_{c_{\mathsf{tp}}}} loop\,3\text{-}1) \mapsto_{\lambda_{c_{\mathsf{tp}}}} \cdots$$

3.2.3 Comparison with Felleisen and Hieb's Reduction System

In λ_c, weak-head reduction simulates the operational semantics only up to some use of β_Ω, $\mathcal{C}_{idem}$ and β_v. Similarly, the operational semantics of $\lambda_{c_{\mathsf{tp}}}$ is simulated by the operational semantics of λ_c only up to some use of the same rules. The same kind of discrepancy shows up in the mutual simulation of the λ_c reduction rules by $\lambda_{c_{\mathsf{tp}}}$ reduction rules.

First, we need to define an equivalent of β_Ω on the $\lambda_{\mathcal{C}\mathsf{tp}}$ side,

$$\beta_\Omega : \quad (\lambda x.\mathcal{T}h\ k\ x)\, M \to \mathcal{T}h\ k\ M \ .$$

We denote with $=_{\lambda_{\mathcal{C}\mathsf{tp}},\,\beta_\Omega}$ the convertibility relation induced by the reduction relation $\lambda_{\mathcal{C}\mathsf{tp}}$ and the β_Ω axiom.

Proposition 3.9
Let M and N be $\lambda_{\mathcal{C}}$-terms. If $M =_{\lambda_{\mathcal{C}}} N$ then $M^\circ =_{\lambda_{\mathcal{C}\mathsf{tp}},\,\beta_\Omega} N^\circ$. More precisely, if $M \to_{\lambda_{\mathcal{C}}} N$ then there exists P such that $M^\circ \twoheadrightarrow_{\lambda_{\mathcal{C}\mathsf{tp}}} P \twoheadleftarrow_{\beta_\Omega,\,\beta_v,\,\mathcal{C}_{idem}} N^\circ$.

Proof
By cases:

($\mathcal{C}_L$)

$$\begin{array}{lll}
((\mathcal{C}\,M)\,N)^\circ & \triangleq & \mathcal{C}\,(\lambda k.\mathsf{tp}\,M^\circ\,(\lambda x.\,\mathcal{T}h\ k\ x)))\,N^\circ \\
& \to_{\mathcal{C}_L} & \mathcal{C}\,(\lambda k.\,\mathsf{tp}\,M^\circ\,(\lambda x.\,\mathcal{T}h\ k\ (x\,N^\circ))) \\
& \leftarrow_{\mathcal{C}_{idem}} & \mathcal{C}\,(\lambda k.\,\mathsf{tp}\,M^\circ\,(\lambda x.\,\mathcal{A}\,(\mathcal{T}h\ k\ (x\,N^\circ)))) \\
& \leftarrow_{\beta_\Omega} & \mathcal{C}\,(\lambda k.\mathsf{tp}\ (M^\circ\,(\lambda x.\,\mathcal{A}\,((\lambda z.\mathcal{T}h\ k\ z)\,(x\,N^\circ))))) \\
& \leftarrow_{\beta_v} & \mathcal{C}\,(\lambda k.\,\mathsf{tp}\,((\lambda c.\,M^\circ\,(\lambda x.\,\mathcal{A}\,(c\,(x\,N^\circ))))\,(\lambda z.\mathcal{T}h\ k\ z))) \\
\text{By (5)} & \leftarrow_{\beta_v} & \mathcal{C}\,(\lambda k.\mathsf{tp}\,((\lambda c.M^\circ\,(\lambda x.(\mathcal{A}\,(c\,(x\,N)))^\circ))\,(\lambda z.\,\mathcal{T}h\ k\ z))) \\
& \triangleq & \mathcal{C}\,(\lambda c.\,M\,(\lambda x.\,\mathcal{A}\,(c\,(x\,N))))^\circ
\end{array}$$

($\mathcal{C}_R$)

$$\begin{array}{lll}
(V\,(\mathcal{C}\,M))^\circ & \triangleq & V^\circ\,\mathcal{C}\,(\lambda k.\mathsf{tp}\,M^\circ\,(\lambda x.\,\mathcal{T}h\ k\ x)) \\
& \to_{\mathcal{C}_R} & \mathcal{C}\,(\lambda k.\,\mathsf{tp}\,(M^\circ\,(\lambda x.\,\mathcal{T}h\ k\ (V^\circ\,x)))) \\
& \leftarrow_{\mathcal{C}_{idem}} & \mathcal{C}\,(\lambda k.\mathsf{tp}\,(M^\circ\,(\lambda x.\,\mathcal{A}\,(\mathcal{T}h\ k\ (V^\circ\,x))))) \\
& \leftarrow_{\beta_\Omega} & \mathcal{C}\,(\lambda k.\mathsf{tp}\,(M^\circ\,(\lambda x.\,\mathcal{A}\,((\lambda z.\mathcal{T}h\ k\ z)\,(V^\circ\,x))))) \\
& \leftarrow_{\beta_v} & \mathcal{C}\,(\lambda k.\,\mathsf{tp}\,((\lambda c.\,M^\circ\,(\lambda x.\,\mathcal{A}\,(c\,(V^\circ\,x))))\,(\lambda z.\,\mathcal{T}h\ k\ z))) \\
\text{By (5)} & \leftarrow_{\beta_v} & \mathcal{C}\,(\lambda k.\,\mathsf{tp}\,((\lambda c.\,M^\circ\,(\lambda x.\,(\mathcal{A}\,(c\,(V\,x)))^\circ))\,(\lambda z.\,\mathcal{T}h\ k\ z))) \\
& \triangleq & \mathcal{C}\,(\lambda c.\,M\,(\lambda x.\,\mathcal{A}\,(c\,(V\,x))))^\circ
\end{array}$$

($\mathcal{C}_{idem}$)

$$\begin{array}{lll}
\mathcal{C}\,(\lambda c.\,\mathcal{C}\,M)^\circ & \triangleq & \mathcal{C}\,(\lambda k.\,\mathsf{tp}\,(\lambda c.\mathcal{C}\,(\lambda k'.\,\mathsf{tp}\,(M^\circ\,\lambda x.\,\mathcal{T}h\ k'\ x)))\,(\lambda x.\,\mathcal{T}h\ k\ x)) \\
& \to_{\beta_v} & \mathcal{C}\,(\lambda k.\,\mathsf{tp}\,\mathcal{C}\,(\lambda k'.\,\mathsf{tp}\,M^\circ\,[\lambda x.\,\mathcal{T}h\ k\ x/c]\,(\lambda x.\,\mathcal{T}h\ k'\ x))) \\
& \to_{\mathcal{C}_{idem}} & \mathcal{C}\,(\lambda k.\,\mathsf{tp}\,M^\circ\,[\lambda x.\,\mathcal{T}h\ k\ x/c]\,(\lambda x.\,\mathcal{A}\,x)) \\
& \leftarrow_{\beta_v} & \mathcal{C}\,(\lambda k.\,\mathsf{tp}\,((\lambda c.\,M^\circ\,(\lambda x.\,\mathcal{A}\,x))\,(\lambda x.\,\mathcal{T}h\ k\ x))) \\
\text{By (5)} & \leftarrow_{\beta_v} & \mathcal{C}\,(\lambda k.\,\mathsf{tp}\,((\lambda c.\,M^\circ\,(\lambda x.\,(\mathcal{A}\,x)^\circ))\,(\lambda x.\,\mathcal{T}h\ k\ x))) \\
& \triangleq & \mathcal{C}\,(\lambda c.\,M\,(\lambda x.\,\mathcal{A}\,x))^\circ
\end{array}$$

($\mathcal{C}_{top}$)

$$\begin{array}{lll}
(\mathcal{C}\,M)^\circ & \triangleq & \mathcal{C}\,(\lambda k.\,\mathsf{tp}\,(M^\circ\,\lambda x.\,\mathcal{T}h\ k\ x)) \\
& \leftarrow_{\mathcal{C}_{idem}} & \mathcal{C}\,(\lambda k.\,\mathsf{tp}\,(M^\circ\,\lambda x.\,\mathcal{A}\,(\mathcal{T}h\ k\ x))) \\
& \twoheadleftarrow_{\beta_v} & \mathcal{C}\,(\lambda k.\,\mathsf{tp}\,((\lambda c.\,(M^\circ\,\lambda x.\,\mathcal{A}\,(c\,x)))\,(\lambda x.\,\mathcal{T}h\ k\ x))) \\
\text{By (5)} & \leftarrow_{\beta_v} & \mathcal{C}\,(\lambda k.\,\mathsf{tp}\,((\lambda c.\,(M^\circ\,\lambda x.\,(\mathcal{A}\,(c\,x))^\circ))\,(\lambda x.\,\mathcal{T}h\ k\ x))) \\
& \triangleq & \mathcal{C}\,(\lambda c.\,M\,(\lambda x.\,\mathcal{A}\,(c\,x)))^\circ
\end{array}$$

□

Conversely, to simulate a $\lambda_{c_{\mathsf{tp}}}$ reduction in λ_c, we need β_Ω.

Proposition 3.10
Let M and N be closed $\lambda_{c_{\mathsf{tp}}}$-terms. If $M \to_{\lambda_{c_{\mathsf{tp}}}} N$ then we have that $M^\bullet \twoheadrightarrow_{\lambda_{c^\star}, \beta_\Omega} N^\bullet$.

Proof
In the following, $M\,[E/k]$ and $M\,[\mathcal{A}\,E/k]$ stand for structural substitution: each application of k to an argument N in M is replaced by $E\,[N\,[E/k]]$ and $\mathcal{A}\,E\,[N\,[\mathcal{A}\,E/k]]$, respectively. We remark that $M\,[\mathcal{A}\,E/k]$ reduces to $M\,[E/k]$ by $\mathcal{C}_{idem}$ and β_v.

We proceed by cases:

($\mathcal{C}_L$)

$$\begin{array}{lll}
(\mathcal{C}\,(\lambda k.\,J)\,M)^\bullet & \triangleq & \mathcal{C}\,(\lambda k.\,J^\bullet)\,M^\bullet \\
& \to_{\mathcal{C}_L} & \mathcal{C}\,(\lambda k.\,(\lambda k.\,J^\bullet)\,(\lambda f.\,\mathcal{A}\,(k\,(f\,M^\bullet)))) \\
& \to_{\beta_v} & \mathcal{C}\,(\lambda k.\,J^\bullet\,[\lambda f.\,\mathcal{A}\,(k\,(f\,M^\bullet))/k]) \\
& \twoheadrightarrow_{\beta_\Omega} & \mathcal{C}\,(\lambda k.\,J^\bullet\,[\mathcal{A}\,(k\,(\Box\,M))/k]) \\
& \twoheadrightarrow_{\mathcal{C}_{idem},\,\beta_v} & \mathcal{C}\,(\lambda k.\,J^\bullet\,[k\,(\Box\,M)/k]) \\
& \triangleq & \mathcal{C}\,(\lambda k.\,J[k\,(\Box\,M)/k])^\bullet
\end{array}$$

($\mathcal{C}_R$) As the previous case.
($\mathcal{C}_{idem}$) We have two cases:

$$\begin{array}{lll}
\mathcal{C}\,(\lambda k.\,\mathsf{tp}\,\mathcal{C}\,(\lambda k'.\,J))^\bullet & \triangleq & \mathcal{C}\,(\lambda k.\,\mathcal{C}\,(\lambda k'.\,J^\bullet)) \\
& \twoheadrightarrow_{\mathcal{C}_{idem},\,\beta_v} & \mathcal{C}\,(\lambda k.\,J^\bullet\,[\lambda x.\,\mathcal{A}\,x/k']) \\
& \twoheadrightarrow_{\beta_\Omega} & \mathcal{C}\,(\lambda k.\,J^\bullet\,[\mathcal{A}\,\Box/k']) \\
& \twoheadrightarrow_{\mathcal{C}_{idem},\,\beta_v} & \mathcal{C}\,(\lambda k.\,J^\bullet\,[\Box/k']) \\
& \triangleq & \mathcal{C}\,(\lambda k.\,J[\mathsf{tp}\,\Box/k'])^\bullet
\end{array}$$

$$\begin{array}{lll}
\mathcal{C}\,(\lambda k.\,k''\,\mathcal{C}\,(\lambda k'.\,J))^\bullet & \triangleq & \mathcal{C}\,(\lambda k.\,k''\,\mathcal{C}\,(\lambda k'.\,J^\bullet)) \\
& \twoheadrightarrow_{\mathcal{C}_R} & \mathcal{C}\,(\lambda k.\,\mathcal{C}\,(\lambda k'.\,J^\bullet\,[\lambda x.\,\mathcal{A}\,(k'\,(k''\,x))/k'])) \\
& \twoheadrightarrow_{\mathcal{C}_{idem},\,\beta_v} & \mathcal{C}\,(\lambda k.\,J^\bullet\,[\lambda x.\,\mathcal{A}\,((\lambda y.\,\mathcal{A}\,y)\,(k''\,x))/k']) \\
& \twoheadrightarrow_{\beta_\Omega} & \mathcal{C}\,(\lambda k.\,J^\bullet\,[\lambda x.\,\mathcal{A}\,(\mathcal{A}\,(k''\,x))/k']) \\
& \twoheadrightarrow_{\mathcal{C}_{idem},\,\beta_v} & \mathcal{C}\,(\lambda k.\,J^\bullet\,[\lambda x.\,\mathcal{A}\,(k''\,\Box)/k']) \\
& \twoheadrightarrow_{\beta_\Omega} & \mathcal{C}\,(\lambda k.\,J^\bullet\,[\mathcal{A}\,(k''\,\Box)/k']) \\
& \twoheadrightarrow_{\mathcal{C}_{idem},\,\beta_v} & \mathcal{C}\,(\lambda k.\,J^\bullet\,[k''\,\Box/k']) \\
& \triangleq & \mathcal{C}\,(\lambda k.\,J\,[k''\,\Box/k'])^\bullet
\end{array}$$

□

Remark incidentally that the composition of $^\bullet$ and $^\circ$ is not the identity in general.

Proposition 3.11
For all M in $\lambda_{c_{\mathsf{tp}}}$, $M^{\bullet\circ} \twoheadrightarrow_{\beta_\Omega \mathcal{C}_{idem}} M$. For all M in λ_c, $M^{\circ\bullet} \twoheadrightarrow_{\mathcal{C}_{top}} M$.

Due to the previous results and the use of β_Ω in the simulation, we cannot prove that in general $\lambda_{c_{\mathsf{tp}}}$ and λ_c simulate each other. For instance, $\mathcal{C}\,(\lambda k.\,k\,\mathcal{C}\,(\lambda k'.\,k'\,x))$ is convertible to $\mathcal{C}\,(\lambda k.\,k\,x)$ in $\lambda_{c_{\mathsf{tp}}}$ but is not in λ_c. This observation has been noted in (Ong & Stewart, 1997) and (de Groote, 1994) who have pointed out that the

relation between the λ_c-calculus and the call-by-value $\lambda\mu$-calculus does not preserve convertibility, even though such a correspondence of the convertibility relation holds in the case of call-by-name.

In order to relate λ_c and $\lambda_{c\mathsf{tp}}$, we focus on the observational behavior of the evaluation relation: a program (*i.e.*, a term without free variables) in λ_c produces an answer if and only if the evaluation of the related program in $\lambda_{c\mathsf{tp}}$ produces an answer. As shown in Example 2.27 and Section 2.3.2, the three distinct types of answers can be simplified if the program is reduced in a context representing the top-level. We thus formulate correctness as follows:

$$\text{Given a closed } \lambda_c\text{-term } M,\ \mathcal{A}\,M \twoheadrightarrow_{\lambda_c} \mathcal{A}\,V \text{ iff } \mathsf{tp}\,M^\circ \twoheadrightarrow_{\lambda_{c\mathsf{tp}}} \mathsf{tp}\,V' \ .$$

Before considering the general case, we focus on the weak-head reduction.

3.3 Weak-Head Reduction of Terms

Like λ_c, the reduction rules of $\lambda_{c\mathsf{tp}}$ are not complete with respect to the operational semantics when applied to terms. In particular, they cannot simulate the following evaluations:

$$\begin{array}{lcl} \mathcal{C}\,(\lambda k.\,k\,M) & \mapsto & M\,[\mathsf{tp}\,\Box/k] \\ \mathcal{C}\,(\lambda\_.\,\mathsf{tp}\,M) & \mapsto & M \end{array}$$

For example, the reduction rules cannot reduce the program

$$\mathcal{C}\,(\lambda k.\,k\,(\lambda x.\,\mathcal{T}h\ k\ (\lambda y.\,y)))$$

to $\lambda x.\,\mathcal{A}\,(\lambda y.\,y)$. Like the λ_c-calculus, the $\lambda_{c\mathsf{tp}}$-calculus can produce three kinds of answers: V, $\mathcal{C}\,(\lambda k.\ k\ V)$ or $\mathcal{C}\,(\lambda k.\,\mathsf{tp}\,V)$. The reason is that a computation involving control is dependent on its evaluation context. While the operational semantics implicitly works in an empty evaluation context, the reduction semantics cannot grant this assumption. The following proposition, reminiscent of Felleisen and Friedman's unique context lemma (Felleisen & Friedman, 1986), summarizes these observations.

Proposition 3.12 (*Unique context lemma for* $\rightarrow_{\lambda_{c\mathsf{tp}}}$ *on terms*)
Let M be a term in $\lambda_{c\mathsf{tp}}$. Exactly one of the following cases happens:

- M has the form V, $\mathcal{C}\,(\lambda k.\,k\,V)$ or $\mathcal{C}\,(\lambda k.\,tpcst\,V)$. In this case M is called an *answer*.
- M has one of the following form:
  - $E[P]$ where P is a β_v, $\mathcal{C}_L$ or $\mathcal{C}_R$ redex,
  - $\mathcal{C}\,(\lambda k.\,q\,E[P])$ where P is a β_v, $\mathcal{C}_L$ or $\mathcal{C}_R$ redex,
  - $\mathcal{C}\,(\lambda k.\,J)$ where J is a $\mathcal{C}_{idem}$ redex.

  In this case, M is called *weakly head reducible*. If the contraction of the given redex in M gives N we write $M \overset{wh}{\rightarrow} N$ and we say that M weakly head reduces to N.
- M has the form $E[x\,V]$, $\mathcal{C}\,(\lambda k.\,q\,E[x\,V])$ or $\mathcal{C}\,(\lambda k.\,k'\,V)$ $(k' \neq k)$. In this case M is said *to have its weak-head reduction stopped*. In the first two cases, it is *stopped by* x while in the third case it is *stopped by* k'.

Especially, a weak-head redex, if it exists, is unique.

We write $M \stackrel{wh}{\twoheadrightarrow} M'$, for the reflexive-transitive closure of $\stackrel{wh}{\rightarrow}$. We also say that M iteratively weakly head reduces to M' for $\stackrel{wh}{\rightarrow}$.

3.4 Weak-Head Reduction of Jumps

Fortunately, $\lambda_{c\mathsf{tp}}$ has the ability to express a fixed top-level evaluation context: it is the purpose of the constant tp. The operational semantics can then be simulated in $\lambda_{c\mathsf{tp}}$ by explicitly reasoning on expressions of the form $\mathsf{tp}\,M$ rather than on terms.

In fact, thanks to the notion of *jumps*, the $\lambda_{c\mathsf{tp}}$ calculus has the ability to lift in the calculus the notion of *state* that is often considered as a purely implementational issue in abstract evaluation machines.

The following proposition characterizes the possible forms of a jump.

Proposition 3.13 (*Unique context lemma for $\rightarrow_{\lambda_{c\mathsf{tp}}}$ on jumps*)
Let J be a jump in $\lambda_{c\mathsf{tp}}$. Exactly one of the following cases happens:

- J has the form $\mathsf{tp}\,V$
- J has one of the following form:
  - $q\,E[P]$ where P is a β_v, $\mathcal{C}_L$ or $\mathcal{C}_R$ redex,
  - $q\,C\,(\lambda k.\,J)$ which is a $\mathcal{C}_{idem}$ redex.

  In this case, J is said *weakly head reducible*. If the contraction of the given redex in J gives J' we write $J \stackrel{wh}{\rightarrow} J'$ and we say that J weakly head reduces to J'.
- J has the form $q\,E[x\,V]$ or $k\,V$. In this case J is said *to have its weak-head reduction stopped*. In the first case, it is *stopped by* x while in the second case, it is *stopped by* k.

Especially, a weak-head redex, if it exists, is unique.

We write $J \stackrel{wh}{\twoheadrightarrow} J'$, for the reflexive-transitive closure of $\stackrel{wh}{\rightarrow}$. We also say that J iteratively weakly head reduces to J'. Remark that when $M \stackrel{wh}{\rightarrow} N$ by executing a $\mathcal{C}_{idem}$ redex and $q\,M \stackrel{wh}{\rightarrow} q'\,N$ by also executing a $\mathcal{C}_{idem}$ redex, the two $\mathcal{C}_{idem}$ redexes are not the same redex. Take for example, $q\,C\,(\lambda k.\,k\,C\,(\lambda k.\,J)) \stackrel{wh}{\rightarrow} q\,C\,(\lambda k.\,J)$ and $\mathcal{C}\,(\lambda k.\,k\,C\,(\lambda k.\,J)) \stackrel{wh}{\rightarrow} \mathcal{C}\,(\lambda k.\,J)$. Remark also that $q\,M \stackrel{wh}{\rightarrow} q'\,M'$ iff either $q \equiv q'$ and $M \stackrel{wh}{\rightarrow}_{\beta_v\mathcal{C}_L\mathcal{C}_R} M'$, or $M \stackrel{wh}{\rightarrow}_{\beta_v\mathcal{C}_L\mathcal{C}_R} \mathcal{C}\,(\lambda k.q''\,N) \stackrel{wh}{\rightarrow} \mathcal{C}\,(\lambda k.q'''\,N')$ with $(q'''\,N')[q/k] \equiv q'\,M'$. Otherwise said, the weak-head reduction of $q\,M$ and M only differs in the possible insertion of $\mathcal{C}_{idem}$ at the time the weak-head reduction of M reaches a term starting with $\mathcal{C}$.

Comparing Proposition 3.12 to Proposition 3.13 makes it clear that reasoning on jumps rather than on terms allows for a uniform characterization of answers. For instance, reasoning on jumps also makes rule $\mathcal{C}_{elim}$ derivable. Indeed, as soon as it is ensured that any expression $\mathcal{C}\,(\lambda k.\,k\,M)$ occurs in a context of the form $q\,E[\mathcal{C}\,(\lambda k.k\,M)]$, its reduction to $q\,E[M]$, when k does not occur free in M, is a consequence of the other rules.

From now on, we focus on jumps. Thanks to the derivability of $\mathcal{C}_E$ in $\lambda_{c\mathsf{tp}}$, the following correspondence between the operational and standard reduction semantics of $\lambda_{c\mathsf{tp}}$ can be easily checked:

Proposition 3.14 (*Simul. of oper. sem. by weak-head red. in* $\lambda_{c\mathsf{tp}}$)
$M \mapsto\!\!\!\!\rightarrow_{\lambda_{c\mathsf{tp}}} V$ iff $\mathsf{tp}\, M \stackrel{wh}{\rightarrow\!\!\!\!\rightarrow} \mathsf{tp}\, V$.

Combined with Proposition 3.3 together, we get:

Corollary 3.15 (*Soundness of w.-h. red. in* $\lambda_{c\mathsf{tp}}$ *for the oper. sem. of* λ_c)
$M \mapsto\!\!\!\!\rightarrow_{\lambda_c} V$ in λ_c iff $\mathsf{tp}\, M^\circ \stackrel{wh}{\rightarrow\!\!\!\!\rightarrow} \mathsf{tp}\, V^\circ$ in $\lambda_{c\mathsf{tp}}$.

3.5 Weak-Head Standardization

Theorem 3.16 (*Weak-head standardization in* $\lambda_{c\mathsf{tp}}$)
$\mathsf{tp}\, M \rightarrow\!\!\!\!\rightarrow_{\lambda_{c\mathsf{tp}}} \mathsf{tp}\, V$ iff $\mathsf{tp}\, M \stackrel{wh}{\rightarrow\!\!\!\!\rightarrow} \mathsf{tp}\, V'$, where $V \rightarrow\!\!\!\!\rightarrow_{\lambda_{c\mathsf{tp}}} V'$.

Proof
One direction is obvious. For the other direction we proceed as in the proof of Theorem 2.34, we follow the proof technique in (Huet & Lévy, 1991). A complication in constructing the projection of a reduction is the interference between $\mathcal{C}_L$ and $\mathcal{C}_{idem}$. As shown in Remark 3.7, the projection of the $\mathcal{C}_{idem}$ reduction with respect to the weak-head $\mathcal{C}_L$ redex consists of the reduction of a newly created redex. To avoid this problem, one replaces each occurrence of $\mathcal{C}_{idem}$ in the reduction from $\mathsf{tp}\, M \rightarrow\!\!\!\!\rightarrow_{\lambda_{c\mathsf{tp}}} \mathsf{tp}\, V$ by its generalization $\mathcal{C}_E^J$, so that the projection preserves the structure of the original reduction. Conversely, at the time of projecting a non trivial weak-head $\mathcal{C}_E^J$ redex along a weak-head $\mathcal{C}_R$ or $\mathcal{C}_L$, one simply removes the leading $\mathcal{C}_R$ or $\mathcal{C}_L$ redex and still stays with a (shorter) weak-head $\mathcal{C}_E^J$ redex. □

4 Connecting λ_c and $\lambda_{c\mathsf{tp}}$

4.1 Observational Equivalence of λ_c *and* $\lambda_{c\mathsf{tp}}$

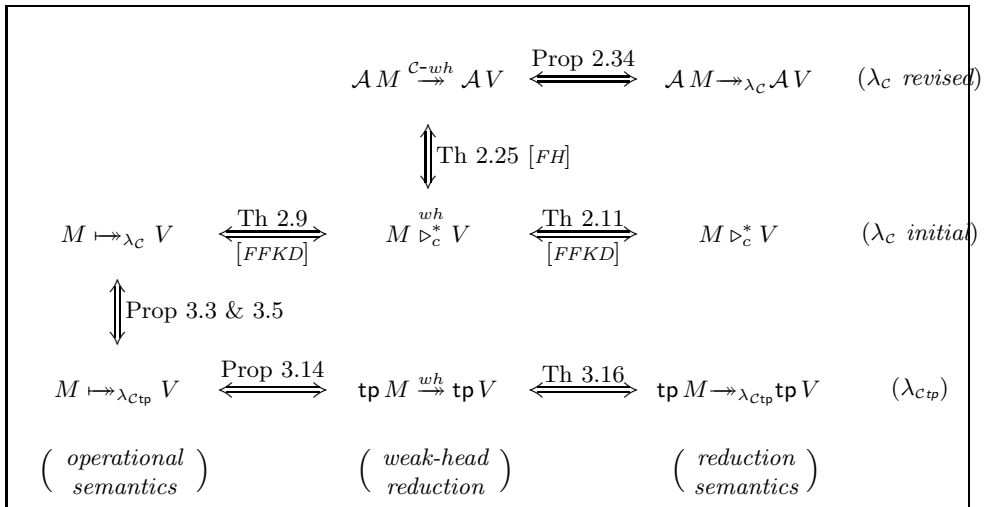
Figure 8 summarizes the equivalences shown in the paper. Especially, putting together Theorems 2.9 and 2.25 and Propositions 3.3, 3.5, and 3.14, we get:

Corollary 4.1 (*Correspondence between* λ_c *and* $\lambda_{c\mathsf{tp}}$ *weak-head reduction*)
$M \stackrel{wh}{\triangleright_c^*} V$ iff $\mathcal{A}\, M \stackrel{\mathcal{C}\text{-}wh}{\rightarrow\!\!\!\!\rightarrow} \mathcal{A}\, V$ iff $\mathsf{tp}\, M^\circ \stackrel{wh}{\rightarrow\!\!\!\!\rightarrow} \mathsf{tp}\, V'^\circ$ where $V \rightarrow\!\!\!\!\rightarrow_{\beta_\Omega \mathcal{C}_{idem} \beta_v} V'$.
$M \stackrel{wh}{\triangleright_c^*} V$ iff $\mathcal{A}\, M^\bullet \stackrel{\mathcal{C}\text{-}wh}{\rightarrow\!\!\!\!\rightarrow} \mathcal{A}\, V$ iff $\mathsf{tp}\, M \stackrel{wh}{\rightarrow\!\!\!\!\rightarrow} \mathsf{tp}\, V'$ where $V \rightarrow\!\!\!\!\rightarrow_{\beta_\Omega \mathcal{C}_{idem} \beta_v} V'^\bullet$.

Thanks to the standardization theorems, Theorems 2.34 and 3.16, we can then extend the correspondence to arbitrary reduction paths:

Corollary 4.2 (*Observational correspondence between* λ_c *and* $\lambda_{c\mathsf{tp}}$)
Let M be a closed λ_c-term. The evaluation of M converges iff the evaluation of M° converges:

$$M \triangleright_c^* V \text{ iff } \mathcal{A}\, M \rightarrow\!\!\!\!\rightarrow_{\lambda_c} \mathcal{A}\, V \text{ iff } \mathsf{tp}\, M^\circ \rightarrow\!\!\!\!\rightarrow_{\lambda_{c\mathsf{tp}}} \mathsf{tp}\, V' \ .$$



In each statement, V is a priori a different value (see the exact statement of the Propositions and Theorems for details). In the statements about $\lambda_\mathcal{C}$, M is a same $\lambda_\mathcal{C}$-term while it is a $\lambda_{\mathcal{C}\mathsf{tp}}$-term in the statements about $\lambda_{\mathcal{C}\mathsf{tp}}$. The equivalences hold both when $\lambda_\mathcal{C}$ is interpreted in $\lambda_{\mathcal{C}\mathsf{tp}}$ through $^\circ$ and when $\lambda_{\mathcal{C}\mathsf{tp}}$ is interpreted in $\lambda_\mathcal{C}$ though $^\bullet$.

Fig. 8. Summary of observational equivalences

Similarly, let M be a closed $\lambda_{\mathcal{C}\mathsf{tp}}$-term. The evaluation of M converges iff the evaluation of $M^\bullet$ converges:

$$M^\bullet \triangleright_c^* V \text{ iff } \mathcal{A}\,M^\bullet \twoheadrightarrow_{\lambda_\mathcal{C}} \mathcal{A}\,V \text{ iff } \mathsf{tp}\,M \twoheadrightarrow_{\lambda_{\mathcal{C}\mathsf{tp}}} \mathsf{tp}\,V' \ .$$

4.2 Distinguishing Features of the Different Operational and Reduction Semantics

Figure 9 summarizes how the different operational semantics and weak-head reduction semantics of $\lambda_\mathcal{C}$ and $\lambda_{\mathcal{C}\mathsf{tp}}$ behave. Since β_v is simulated the same in all cases, we focus on $\mathcal{C}_L$, $\mathcal{C}_R$, $\mathcal{C}_{idem}$ and $\triangleright_c$. To allow a full comparison, we consider terms that are in the image of $^\bullet$. The figure shows how some closed term $E[\mathcal{C}\,(\lambda k.M)]$ eventually captures the surrounding context of $\mathcal{C}$. The less efficient semantics is the reduction semantics of $\lambda_\mathcal{C}$, then comes the operational semantics of $\lambda_\mathcal{C}$ and its embedding in $\lambda_{\mathcal{C}\mathsf{tp}}$ when $\mathcal{C}$ is interpreted as an operator of reification of the context as a regular function. Finally, structural substitution is the most efficient. The results differ up to $\beta_\Omega\beta_v\mathcal{C}_{idem}$ contractions in the occurrences of the substituends. Note that all these contractions are non trivial unless E is empty in which case E^* is $\lambda x.\,\mathcal{A}\,x$ which is the same as $\lambda x.\,\mathcal{A}\,E[x]$.

4.3 Simulation of Structural Substitution in $\lambda_\mathcal{C}$

The mapping $^\circ$ interprets $\mathcal{C}$ as an operator that reifies its context into a regular function. Henceforth, it does not take advantage, as shown by Propositions 3.2 and

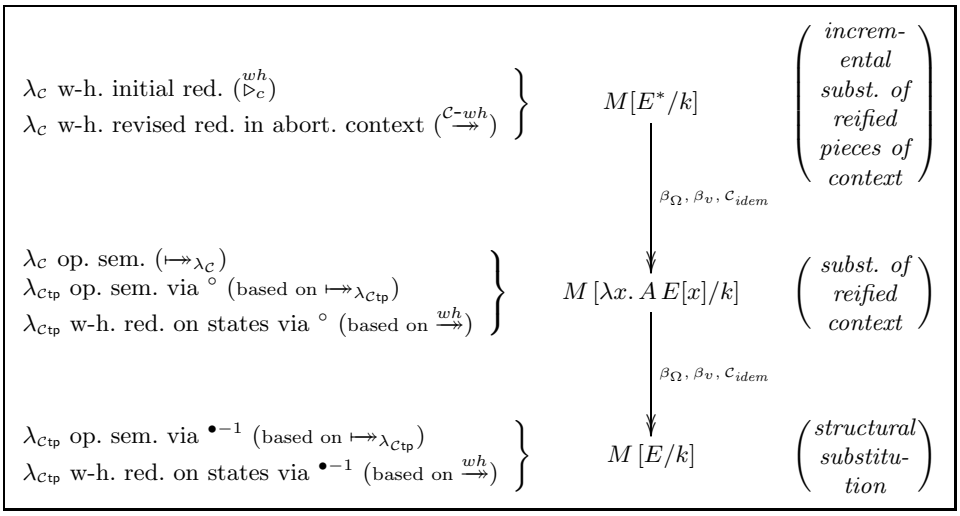


Fig. 9. How $E[\mathcal{C}\,(\lambda k.M)]$ eventually reduces for the different op. sem. and w.-h. reductions and how the respective results relates.

3.5, of the efficiency of structural substitution. We would get a better efficiency by directly interpreting λ_c into the image of $\lambda_{c_{\mathsf{tp}}}$ by $^\bullet$. Let's first focus on closed terms.

On closed terms, $^\bullet$ is injective and the characteristic feature of its image in λ_c is that $\mathcal{C}$ is necessarily applied to an abstraction of the form $\lambda k.M$, and every such k bound in the scope of $\mathcal{C}$ occurs applied under the form $k\,N$. Moreover, such a subterm $k\,N$ has to be itself the immediate subterm of some "$\mathcal{C}\,(\lambda k'$". Let's adopt the further convention that for every such subterm $k\,N$ surrounded by some "$\mathcal{C}\,(\lambda k'$", this "$\mathcal{C}\,(\lambda k'$" is omitted if k' does not occur free in $k\,N$. Otherwise said, if some $k\,N$ is surrounded by an $\mathcal{A}$, this $\mathcal{A}$ is left implicit. Let's call this restriction $\lambda_c^{S_0}$.

Now focusing on open terms, we observe that $^\bullet$ is not injective. The reason is that free variables, whether they are usual variables or continuation variables, are interpreted in the same and unique class of variables in λ_c. To remedy this non injectivity, we modify $\lambda_c^{S_0}$ so to introduce a distinct class of continuations variables. Let's call λ_c^S the resulting language. It is defined by the following grammar:

$$\begin{array}{l} x \in \mathit{Vars} \\ k \in \mathit{KVars} \\ M, N \in \mathit{Terms} \quad ::= \quad x \mid \lambda x.\,M \mid M\,N \mid k\,M \mid \mathcal{C}\,(\lambda k.M) \end{array}$$

If we restrict λ_c^S to the fragment with no free continuation variable, we fall back on a calculus which is essentially $\lambda_c^{S_0}$: the distinction between usual variables and continuation variables becomes unnecessary because it is enough to look at whether the variable is bound by some λ or by some $\mathcal{C}$ to know if it is an ordinary variable or a continuation variable. Otherwise said, $\lambda_c^{S_0}$ can be equivalently seen as a restriction of λ_c (where no distinction between usual and continuation variables is done) and as a restriction of λ_c^S.

Let † be the following interpretation of λ_c^S into $\lambda_{c\mathsf{tp}}$:

$$\begin{array}{rcll}
x^\dagger & = & x & \\
(\lambda x.\, M)^\dagger & = & \lambda x.\, M^\dagger & \\
(M\, N)^\dagger & = & M^\dagger\, N^\dagger & \text{if } M \text{ not some } k \\
(k\, M)^\dagger & = & \mathcal{T}h\ k\ M^\dagger & \\
\mathcal{C}\,(\lambda k.M)^\dagger & = & \mathcal{C}\,(\lambda k.\, k\, N^\dagger) & \text{if } M \text{ has the form } k\, N \\
\mathcal{C}\,(\lambda k.\, M)^\dagger & = & \mathcal{C}\,(\lambda k.\, \mathsf{tp}\, M^\dagger) & \text{otherwise}
\end{array}$$

This interpretation is not surjective ($k\, N$ and $\mathcal{A}\,(k\, N)$ have the same image) but this is sufficient to be able to transfer back structural reduction from $\lambda_{c\mathsf{tp}}$ to λ_c^S. The inherited reduction system for λ_c^S is the following:

$$\begin{array}{lrcl}
\beta_v: & (\lambda x.\, M)\, V & \to & M\, V/x] \\
\mathcal{C}_L: & \mathcal{C}\,(\lambda k.\, M)\, N & \to & \mathcal{C}\,(\lambda k.\, M\,[k\,(\Box\, N)/k]) \\
\mathcal{C}_R: & V\, \mathcal{C}\,(\lambda k.\, M) & \to & \mathcal{C}\,(\lambda k.\, M\,[k\,(V\, \Box)/k]) \\
\mathcal{A}_L: & (k\, M)\, N & \to & k\, M \\
\mathcal{A}_R: & V\,(k\, M) & \to & k\, M \\
\mathcal{C}_{idem}: & k'\, \mathcal{C}\,(\lambda k.\, M) & \to & M\,[k'/k] \\
\mathcal{C}_{idem}{}': & \mathcal{C}\,(\lambda k'.\, \mathcal{C}\,(\lambda k.\, M)) & \to & \mathcal{C}\,(\lambda k'.\, M\,[\mathcal{A}\,\Box/k]) \\
\mathcal{A}_{idem}: & k'\,(k.M) & \to & k\, M \\
\mathcal{A}_{idem}{}': & \mathcal{A}\,(k\, M) & \to & k\, M
\end{array}$$

Proposition 4.3 (*Simulation of $\lambda_{c\mathsf{tp}}$ within λ_c*)
For all M and N in λ_c^S, $M \to N$ in λ_c^S implies $M^\dagger \twoheadrightarrow N^\dagger$ in $\lambda_{c\mathsf{tp}}$. For all M and N in $\lambda_{c\mathsf{tp}}$, $M \to N$ in $\lambda_{c\mathsf{tp}}$ implies $M^\bullet \to N^\bullet$ in λ_c^S. Moreover, $M^{\bullet\dagger} \equiv M$ in $\lambda_{c\mathsf{tp}}$ and $M^{\dagger\bullet} \twoheadrightarrow M$ in λ_c^S. Since $\lambda_c^{S_0}$ is a subset of λ_c, this provides with a mutual simulation from this subset of λ_c with $\lambda_{c\mathsf{tp}}$ when the latter is restricted to the terms with no free continuation variable.

Compiling λ_c into $\lambda_c^{S_0}$ is now simple: each occurrence of k that is bound by some $\mathcal{C}\,(\lambda k.\, M)$ and that is not applied in M is replaced by $\lambda x.\, \mathcal{A}\,(k\, x)$ while each subterm $\mathcal{C}\, M$ where M is not of the form $\lambda k.\, N$ is replaced by $\mathcal{C}\,(\lambda k.\,(M\,(\lambda x.\, \mathcal{A}\,(k\, x))))$ (these transformations are known to be operationally sound). Of course, occurrences of $\mathcal{C}$ changed that way again behave as operators of functional reification of contexts.

5 Conclusion

We investigated the differences between the historical calculus of control λ_c and a calculus called $\lambda_{c\mathsf{tp}}$ that is derived from the interpretation of classical proofs as programs. Both calculi manipulate continuations but the former reifies them as regular functions and uses ordinary substitution to propagate continuations while the latter manipulates them directly as evaluation contexts and uses a specific notion of structural substitution.

We showed that the reduction systems of both calculi, though cannot simulate

each other, are observationally equivalent. We showed that control based on structural substitution provides smoother results than control based on context reification:

- Operational semantics and weak-head reduction match in the presence of structural substitution while they differ when contexts are incrementally reified.
- Reification of contexts expands the size of context, leading to possible space leaks, while structural substitution does not.

Thanks to the presence of a notation for the continuation of the top-level evaluation, the syntax of $\lambda_{c\mathsf{tp}}$ has a finer structure than the syntax of λ_c. In particular, the constructions of λ_c itself can be finely explained from the more elementary components of $\lambda_{c\mathsf{tp}}$.

We showed that making explicit the continuation of the top-level evaluation provides a way to uniformly manage the different kinds of answers that control theory reduction traditionally requires. It also clarifies the role of rules like $\mathcal{C}_{top}$ in λ_c or $\mathcal{C}_{elim}$ in the calculi inspired by $\lambda\mu$-calculus: these rules become operationally useless as soon as the continuation of the evaluation is made formal.

We incidentally proved weak-head standardization and confluence for $\lambda_{c\mathsf{tp}}$ and improved on previous results for λ_c. Especially, we provided a deterministic weak-head standardization for the revised theory of λ_c, we repaired a "deterministic leak" in Plotkin-style notion of standardization and we showed the confluence of the revised theory when $\mathcal{C}_{top}$ is omitted.

Scalability

We believe that our study would apply in a similar way to the call-by-name variant of λ_c in which β replaces β_v and C_R is removed. The main difference will be that β_Ω is a particular case of β in call-by-name.

We believe that our study would also directly apply to the extension of λ_c with a delimiter of continuation # (see the historical note) and the operational rules $C[E[\lambda x.\, M)\, V]] \to C[E[M\,[V/x]]]$, $C[E[\#\, V]] \to C[E[V]]$ and $C[E[\mathcal{C}\, M]]] \to C[M\ \lambda x.\mathcal{A}\, E[x]]$ with C being $\square$ or $C[E[\#\,\square]]$. The correspondence would then be with the $\lambda_{c\#\mathsf{tp}}$ calculus in (Ariola *et al.*, 2004; Ariola *et al.*, 2007).

Typing

A system of simple types for $\lambda_{c\mathsf{tp}}$, inherited from (Parigot, 1992), has been given in (Ariola & Herbelin, 2003; Ariola *et al.*, 2005). A peculiarity of this typing system is that the type of tp is a parameter of the system. Based on the definition of $(\mathcal{C}\, M)^\circ$, this typing system leads to naively type $\mathcal{C}$, seen as a stand-alone constant of λ_c, with type $((A \to B) \to T) \to A$ where T is the type of tp and $\mathcal{C}$ is polymorphic over A and B. This is quite constraining as this forces k to be used, in a given instance of $\mathcal{C}\,(\lambda k.M)$, only in contexts of type B. A more natural approach would be to force B to be the top-level type T and hence to have $\mathcal{C}$ of type $((A \to T) \to T) \to A$. With

this new constraint, each call to k would typically be surrounded by some $\mathcal{A}$ (itself of derived type $T \to A$ for any A) to be used in a contexts of arbitrary type. The system we obtain is strictly equivalent to, e.g., Murthy's parametric typing system $\vdash_T$ (Murthy, 1992) where Murthy's rule $abort_1$ is replaced by a dumb coercion from T to $\bot$. Indeed, Murthy's typing system, with this modification, can be seen as a system where the top-level type T and $\bot$ are interchangeable and $\mathcal{C}$ can freely have type $((A \to T) \to T) \to A$ or $((A \to \bot) \to \bot) \to A$ or any of the two other combination involving T and $\bot$.

A more interesting typing system is obtained by renouncing to the identification between $\bot$ and the top-level type T and by rather literally seeing $\bot$ as an empty type equipped with the rule $\Gamma \vdash_T M : \bot$ implies $\Gamma \vdash_T M : B$. Then constraining B to be $\bot$ in the naive type of $\mathcal{C}$, we get $\mathcal{C}$ of type $((A \to \bot) \to T) \to A$. By this approach, we obtain that calls to k in $\mathcal{C}\,(\lambda k.M)$ get usable in contexts of any type, consistently with the abortive nature of these calls. For instance, a term like $\mathcal{C}\,(\lambda k.$ if $\texttt{"foo"} = k\,3$ then 1 else 2) would be typable without needing to surround the call to k with $\mathcal{A}$.

In any case, we believe that typing $\mathcal{C}$ with type $((A \to \bot) \to \bot) \to A$ as in (Griffin, 1990) is overly restrictive for λ_c though the typing we obtain remains consistent with the observation that this is a relevant type for $\mathcal{C}$ *when the top-level type is itself* $\bot$. Alternatively, assigning the polymorphic type $((A \to \bot) \to \bot) \to A$ to $\mathcal{C}$, as in Griffin, forces us to reason in a top-level context of the form $\mathcal{C}\,(\lambda k.\, k\,\Box)$ where k, of type $T \to \bot$, has the role of an explicit top-level constant and `tp` is not needed any longer.

Implementation

One could ask which of λ_c or $\lambda_{c_{\texttt{tp}}}$ simulates at best real implementations of control operators. If we consider the `call/cc` operator that, among others, Scheme and SML provide, the common practice is to implement it as an operator that first duplicates the stack then pushes on the stack a closure that restores this stack. Formally, this corresponds to the rule

$$E[\texttt{call/cc}\,M] \mapsto E[M\,(\lambda x.\,\mathcal{A}\,E[x])]$$

where E schematizes the stack and $\lambda x.\mathcal{A}\,E[x]$ schematizes the restoring operator. If one try to model `call/cc` in λ_c or $\lambda_{c_{\texttt{tp}}}$ one observes that only $\lambda_{c_{\texttt{tp}}}$ is able to simulate the fact that the stack is kept in place by `call/cc`. If one takes the standard encoding of `call/cc` as $\lambda x.\,\mathcal{C}\,(\lambda k.\,k\,(x\,k))$, the derived operational rule is

$$E[\texttt{call/cc}\,M] \mapsto\!\!\!\!\to_{\lambda_c} (\lambda x.\,\mathcal{A}\,E[x])\,(M\,(\lambda x.\,\mathcal{A}\,E[x]))$$

and the discussion on the inefficiency of such an implementation applies (see Section 2.5). No other encoding of `call/cc` in λ_c can give the correct operational semantics because structural substitution is required and λ_c doesn't know about structural substitution.

To the contrary, $\lambda_{c\mathsf{tp}}$ supports the following encoding:

$$\begin{array}{lcl} \texttt{call/cc} & \triangleq & \mathcal{C}\,(\lambda k.\,k\,(M\,(\lambda x.\,\mathcal{T}h\ k\ x))) \\ \mathcal{A}\,M & \triangleq & \mathcal{T}h\ \mathsf{tp}\ M \end{array}$$

that exactly simulates the above operational rule of `call/cc`:

$$E[\texttt{call/cc}\,M] \mapsto\!\!\!\!\rightarrow_{\lambda_{c\mathsf{tp}}} E[M\,(\lambda x.\,\mathcal{T}h\ \mathsf{tp}\ E[x])] \equiv E[M\,(\lambda x.\,\mathcal{A}\,E[x])]\ .$$

In the absence of exception handling, we can in principle do more by implementing the calls to the continuation as special calls instead of regular call-by-value function calls. Consider for example the case of SML in which jumps are made explicit by calls to the operator *throw*. If *throw* k M were implemented as a function that first restores the stack encoded in its first argument before starting evaluating the second argument, one would directly obtain the efficiency of structural substitution. In short, in the absence of exceptions, we could safely assign to *throw* the following alternative semantics:

$$E'[\mathit{throw}\ (\lambda x.\,\mathcal{A}\,E[x])\ M] \mapsto E[M]\,.$$

COMMENT by Zena: do you that style for throw? END
Of course, if the evaluation of M later throws to another continuation, the restoring is a useless one, but in any case, it avoids keeping in place a stack that is definitely known to be useless. In the presence of exceptions though, this is not a conservative optimization as exceptions jump to the dynamically-closest handler (which, according to the semantics of SML, would become the one in E instead of the one in E').

Related Work

The purpose of this paper was to compare the reduction semantics of the λ_c and $\lambda_{c\mathsf{tp}}$ calculi which are both variants of usual λ-calculus with control. We deliberately do not study the connection with the $\overline{\lambda}\tilde{\mu}$-calculus (Curien & Herbelin, 2000) which is another promisingly "well-behaved" calculus for call-by-value control.

A comparison between a simply-typed call-by-name variant of λ_c and a variant of simply-typed Parigot's $\lambda\mu$-calculus similar to our calculus λ_c^S has been done by (de Groote, 1994). An interesting aspect of this work is that $\mathcal{A}$ is removed from $\mathcal{C}_L$ as it is the case in the lifting rule for $\mathcal{F}$ (see the historical note below). Using the lifting rules of $\mathcal{F}$ in the setting of λ_c, can indeed be seen as an improvement of λ_c since an occurrence of $\mathcal{A}$ is eventually anyway inserted by $\mathcal{C}_{idem}$. However, the simulation of $\mathcal{C}_{idem}$ is only marginally treated by de Groote and it strongly depends on the presence of types. From our point of view, this is because this study missed the notion of top-level continuation tp and that the only way to implicitly talk about it was to talk about terms of type $\bot$: in the simply-typed proof-as-program setting, $\bot$ is the type of tp (see (Ariola & Herbelin, 2003; Ariola *et al.*, 2005)).

A Historical Note:
On the Indiana Control Operators
by Matthias Felleisen

The births of $\mathcal{C}$, $\mathcal{F}$, and prompt took a long time. Indeed, prompt—the control delimiter—was "born" twice for radically different reasons.

The story begins with Daniel Friedman's famous "511" course. In the fall of 1984, a group of enthusiastic PhD students (including Bruce Duba, Eugene Kohlbecker, and myself) enrolled in this graduate seminar on programming language research. At the time, Dan Friedman focused on "coordinate computing," now known as concurrent and distributed computing (Filman & Friedman, 1984). Every week he asked us to implement a Scheme simulation of some coordinate computing language. In the process, we began to program with continuations because every simulation depended on implementing some form of threads.

After a few of those projects, I realized that capturing only a part of the current continuation would significantly simplify the programs and provide some protection of the kernel. In other words, while `call/cc` grabbed continuations between the current expression and the prompt, most simulations needed only a part of this continuation. Since I associated the activity of truncating the continuation with the visible Scheme prompt, I dubbed this new construct "first-class prompt." I used the term "first-class" because I wanted to place the prompt anywhere in my program, not just at the top of the main expression. My first crude implementation used Scheme 84's macros and engines (Haynes & Friedman, 1984).

During the following summer (1985), I worked at the MCC in Austin, and Dan Friedman came to visit me there in August. When he arrived, he was excited about a discovery he had made on the flight to Austin. He had understood that continuations and `call/cc` could be characterized by two equations:

$$\begin{aligned} f\ (\texttt{call/cc}\ g) &= \texttt{call/cc}\ (\lambda k.\ (g\ (\lambda x.k\ (f\ x)))) \\ (\texttt{call/cc}\ g)\ f &= \texttt{call/cc}\ (\lambda k.\ (g\ (\lambda x.k\ (x\ f)))) \end{aligned}$$

He liked the symmetry but he didn't know where to go from here. After I returned to Indiana later that month, Bruce, Eugene, Dan and I studied these equations in more depth. We realized that the `call/cc` of the equations wasn't the `call/cc` of Scheme and that the equations didn't capture `call/cc`'s behavior properly. So we dubbed this control operator $\mathcal{C}$ (after trying out some other TEX symbols) and continued our search of meaning in these equations.

By the end of the fall semester, I had understood how these equations fit in with the rest of Plotkin's framework on the λ_v-calculus (Plotkin,

1975), and we all had figured out the exact relationship between $\mathcal{C}$ and `call/cc`:

$$\begin{array}{rcl} \texttt{call/cc} & \triangleq & \lambda f.\ \mathcal{C}\,(\lambda k.\ k\ (f\ k)) \\ \mathcal{C} & \triangleq & \lambda f.\ \texttt{call/cc}\ (\lambda k.\ \mathcal{A}\,(f\ k)) \\ \mathcal{A}\,e & \triangleq & \mathcal{C}\,(\lambda\_.\ e) \end{array}$$

The result appeared as a conference paper (Felleisen *et al.*, 1986) and in a cleaned-up journal paper (Felleisen *et al.*, 1987). To establish the validity of the control calculus, I had to prove a Church-Rosser lemma and a Standard Reduction lemma. After some experimenting I discovered that a minor modification of the above equation worked much better:

$$\begin{array}{rcl} f\ (\mathcal{C}\,g) & = & \mathcal{C}\,(\lambda k.\ g\ (\lambda x.\ \mathcal{A}\,(k\ (f\ x)))) \\ (\mathcal{C}\,g)\ f & = & \mathcal{C}\,(\lambda k.\ g\ (\lambda x.\ \mathcal{A}\,(k\ (x\ f)))) \end{array}$$

A major blemish remained, however. We could not eliminate the special top-level rule from our calculus:

$$\mathcal{C}\,f \quad = \quad f\ (\lambda x.\ \mathcal{A}\,x) \quad \textbf{when } \mathcal{C}\,f \textbf{ is the entire program}$$

Physicists would call this a "major asymmetry," and I hated it. A minor blemish was that we had two different versions of these pairs of equations: one for calculating and one for meta-theorems.

Right after we had submitted the journal paper in 1986, I re-discovered my nearly forgotten prompt. More concretely, I realized that the condition "... is the entire program" in the above equation and "grabbing the current continuation of the program" (up to the prompt) posed the same problem. If I turned the "top" of the program into a separate, algebraically free construction, the calculus would become an ordinary calculus of control:

$$\#\ (\mathcal{C}\,f) \quad = \quad \#\ (f\ (\lambda x.\ \mathcal{A}\,x))$$

A quick check suggested that the revised theory would hold up, but now I had become curious as to whether I could simplify the calculus even more.

My search quickly showed that I could simplify the proofs of the meta-theorems even more if I threw out abort ($\mathcal{A}$) entirely. I knew I could remove $\mathcal{A}$, because it was just an abbreviation for $\mathcal{C}$ anyway. Of course, just like Dan Friedman's original equations didn't specify `call/cc`, these revised equations didn't specify $\mathcal{C}$ anymore. The next letter in the calligraphic alphabet that we hadn't used yet was $\mathcal{F}$ and so I arrived at these equations:

$$\begin{array}{rcl} e\ (\mathcal{F}\ g) & = & \mathcal{F}\ (\lambda k.\ g\ (\lambda x.\ k\ (e\ x))) \\ (\mathcal{F}\ g)\ e & = & \mathcal{F}\ (\lambda k.\ g\ (\lambda x.\ k\ (x\ e))) \\ \#\ (\mathcal{F}\ e) & = & \#\ (e\ (\lambda x.\ x)) \end{array}$$

and furthermore,

$$\begin{array}{rcl} \mathcal{C} & \triangleq & \lambda g.\ \mathcal{F}\ (\lambda k.\ g\ (\lambda x.\ \mathcal{A}(k\ x))) \\ \mathcal{A}\,e & \triangleq & \mathcal{F}\ (\lambda\_.\ e) \end{array}$$

Once I saw this set of equations, it was crystal clear that this was *the* calculus: it had simple equations, the equations described the calculations, they posed no problem for the meta-theorems, and the system introduced a powerful new control construct.

Naturally, we (that is, Bruce Duba and I) began to look for other control constructs that could be "derived" from calculi. Our most important insight was that we had a design choice concerning the behavior of $\mathcal{F}$ when it encountered a prompt:

- **it could do what it does now**
- **it could eliminate the prompt, and**
- **it could absorb it.**

We called these choices $\mathcal{F}$, $\mathcal{F}^+$, and $\mathcal{F}^-$ because $\mathcal{F}^+$ could simulate $\mathcal{F}$ and $\mathcal{F}$ could simulate $\mathcal{F}^-$. For all three, I sketched out proofs of the major meta-theorems, and they all worked out fine. At that point, I tried to use pragmatics to decide which of the three was important. I mostly used my examples from Dan Friedman's 1984 course, and those quickly showed that $\mathcal{F}$ was all I needed. That settled the question. When I finally submitted a paper to POPL 1988, I used $\mathcal{F}$ and prompt to introduce control delimiters into the programming language literature (Felleisen, 1988).

Note: Around the time I left Indiana, I invented my last control operator(s): $\mathcal{G}$. The standard reduction equation for this family of operators has this shape:

$$\#\ E[\mathcal{G}_{\boldsymbol{enc}}\ f] \quad = \quad \#\ f\ (\boldsymbol{enc}\{E\})$$

where *enc* is a meta-function that maps evaluation contexts to constructs inside the programming language. I never developed a theory or a practical framework for $\mathcal{G}$, but perhaps someone else will.

Acknowledgments

We thank Stefan Blom and Femke van Raamsdonk for answering numerous questions regarding rewriting. The paper also benefited from conversations with Amr Sabry.

References

Ariola, Z. M., & Herbelin, H. (2003). Minimal classical logic and control operators. *Pages 871–885 of: Thirtieth international colloquium on automata, languages and programming , ICALP'03, eindhoven, the netherlands, june 30 - july 4, 2003*, vol. 2719. Springer-Verlag, LNCS.

Ariola, Zena M., Herbelin, Hugo, & Sabry, Amr. (2004). A type-theoretic foundation of continuations and prompts. *Pages 40–53 of: Acm sigplan international conference on functional programming*. ACM Press, New York.

Ariola, Zena M., Herbelin, Hugo, & Sabry, Amr. (2005). A proof-theoretic foundation of abortive continuations. *Higher-order and symbolic computation*. To appear.

Ariola, Zena M., Herbelin, Hugo, & Sabry, Amr. (2007). A type-theoretic foundation of delimited continuations. *Higher-order and symbolic computation*. to appear.

Baba, Kensuke, Hirokawa, Sachio, & etsu Fujita, Ken. (2001). Parallel reduction in type free $\lambda\mu$-calculus. *Electronic notes in theoretical computer science*, **42**, 52–66.

Curien, Pierre-Louis, & Herbelin, Hugo. (2000). The duality of computation. *Pages 233–243 of: Acm sigplan international conference on functional programming*. ACM Press, New York.

de Groote, P. (1994). On the relation between the lambda-mu calculus and the syntactic theory of sequential control. *Pages 31–43 of:* Pfennig, F. (ed), *Logic programming and automated reasoning, proc. of the 5th international conference, lpar'94*. Berlin, Heidelberg: Springer.

Duba, B. F., Harper, R., & MacQueen, D. (1991). Typing first-class continuations in ML. *Pages 163–173 of: Conference record of the eighteenth annual ACM symposium on principles of programming languages*.

Felleisen, M. (1988). The theory and practice of first-class prompts. *Pages 180–190 of: Proceedings of the 15th acm symposium on principles of programming languages (popl '88)*. ACM Press, New York.

Felleisen, M., & Friedman, D. (1986). Control operators, the secd machine, and the lambda-calculus. *Pages 193–217 of: Formal description of programming concepts-iii*. North-Holland.

Felleisen, M., & Hieb, R. (1992). The revised report on the syntactic theories of sequential control and state. *Theoretical computer science*, **103**(2), 235–271.

Felleisen, M., Friedman, D., Kohlbecker, E., & Duba, B. (1986). Reasoning with continuations. *Pages 131–141 of: First symposium on logic and computer science*.

Felleisen, Matthias, Friedman, D. P., Kohlbecker, E., & Duba, B. (1987). A syntactic theory of sequential control. *Theoretical computer science*, **52**(3), 205–237. Preliminary version: Reasoning with Continuations, in Proceedings of the 1st IEEE Symposium on Logic in Computer Science, 1986.

Filman, Robert E., & Friedman, Daniel P. (1984). *Coordinated computing: Tools and techniques for distributed software*. New York: McGraw Hill.

Ganz, Steven E., Friedman, Daniel P., & Wand, Mitchell. (1999). Trampolined style. *Pages 18–27 of: Acm sigplan international conference on functional programming*. ACM Press, New York.

Griffin, T. G. (1990). The formulae-as-types notion of control. *Pages 47–57 of: Conf. record 17th annual ACM symp. on principles of programming languages, POPL'90, s an francisco, CA, USA, 17-19 jan 1990*. ACM Press, New York.

Haynes, Christopher T., & Friedman, Daniel P. (1984). Engines build process abstractions. *Pages 18–24 of: Conference record of the 1984 ACM symposium on lisp and functional programming*. ACM.

Huet, Gérard, & Lévy, Jean-Jacques. (1991). Computations in orthogonal rewriting systems 1 and 2. *Computational logic. essays in honor of alan robinson. ed. j.-l. lassez & g.d. plotkin*.

Murthy, C. (1992). Control operators, hierarchies, and pseudo-classical type systems: A-translation at work. *Pages 49–71 of: Acm workshop on continuations*.

Ong, C.-H. Luke, & Stewart, C. A. (1997). A Curry-Howard foundation for functional computation with control. *Pages 215–227 of: Conf. record 24th ACM SIGPLAN-SIGACT symp. on principles of programming languages, POPL'97, paris, france, 15-17 jan. 1997.* ACM Press, New York.

Parigot, M. (1992). Lambda-mu-calculus: An algorithmic interpretation of classical natural deduction. *Pages 190–201 of: Logic programming and automated reasoning: International conference lpar '92 proceedings, st. petersburg, russia.* Springer-Verlag.

Plotkin, Gordon D. (1975). Call-by-name, call-by-value, and the λ-calculus. *Theoretical computer science*, **1**, 125–159.

van Oostrom, Vincent. (1994). Confluence by decreasing diagrams. *Theoretical computer science*, **126**(1), 259–280.

A Decreasing diagrams

The problem with showing commutativity by means of a tiling argument is that one needs to show that the tiling process terminates. van Oostrom (van Oostrom, 1994) defined the notion of *decreasing diagrams* and showed that tiling with decreasing diagrams terminates. Decreasing diagrams are defined in the setting of labeled abstract reduction systems.

Definition A.1

An abstract rewriting system (ARS) is a structure $(A, \rightarrow)$ consisting of a set A and a binary relation on A. A labeled ARS is a structure $\langle A, (\underset{l}{\rightarrow})_{l \in L} \rangle$, where L is a set of labels and for each $l \in L$, $(A, \underset{l}{\rightarrow})$ is an ARS.

To define the notion of decreasing diagram we consider labeled diagrams and a well-founded order on the labels. The key to the notion is a measure $|.|$ defined on strings of labels. This measure is easily computed by following these steps:

- **Write down the string**
- **Erase every element in the string, such that a larger element occurs at an earlier position.**
- **Gather the remaining elements in a multiset.**

For example, using the natural numbers with their natural order, we have

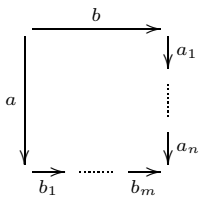
$$|121232| = |12\ \ 232| = |12\ \ 23\ \ | = \{\!\{1, 2, 2, 3\}\!\}\ .$$

Definition A.2

Given a set of labels A and a well-founded order $<$ on A, let $|.|$ be the measure from strings of labels to multisets of labels defined by:

$$|a_1 \dots a_n| = \{\!\{a_i | \text{ there is no } j < i \text{ with } a_j > a_i\}\!\}\ .$$

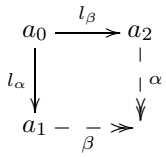
Then, the diagram



is *decreasing* with respect to $\leq$ if $\{\!\{a, b\}\!\} \geq |ab_1 \dots b_m|$ and $\{\!\{a, b\}\!\} \geq |ba_1 \dots a_n|$.

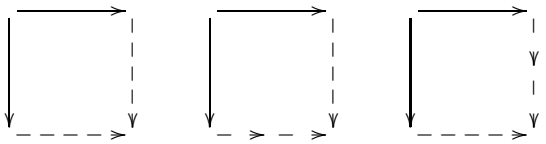
We can use the notion of decreasing diagrams to prove commutativity as follows. First, we prove the existence of enough diagrams to start a tiling process, then we check if all tiles are decreasing. By the following theorem we can then conclude commutativity.

Theorem A.3
Given a labeled ARS $\langle A, (\underset{l}{\rightarrow})_{l \in L_\alpha \cup L_\beta} \rangle$ and a well-founded order on $L_\alpha \cup L_\beta$. Define $\underset{\alpha}{\rightarrow} = \cup_{a \in L_\alpha} \underset{a}{\rightarrow}$ and $\underset{\beta}{\rightarrow} = \cup_{b \in L_\beta} \underset{b}{\rightarrow}$. If for every $a_0, a_1, a_2 \in A, l_\alpha \in L_\alpha, l_\beta \in L_\beta$, such that $a_0 \underset{l_\alpha}{\rightarrow} a_1$ and $a_0 \underset{l_\beta}{\rightarrow} a_2$ there exists a decreasing diagram

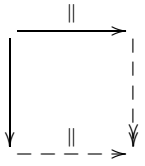


then we have that $\underset{\alpha}{\rightarrow}$ and $\underset{\beta}{\rightarrow}$ are commutative.

A special case arises when we take the sets L_α and L_β to be equal to the set of all labels L, then confluence of $\rightarrow_L$ can be concluded. A common case that decreasing diagrams cannot handle is duplication in both the horizontal and vertical direction, *e.g.* there is no possible labeling that makes the following diagrams all decreasing:



It is often possible to solve this problem by introducing a form of parallel reduction or complete development in, for example, the horizontal direction. With respect to parallel reduction the three diagrams should then collapse into the single diagram



which can be made decreasing by ordering the parallel reduction larger than the other reductions.