Mathematical Structures in Computer Science (2022), **32**, pp. 934–981
doi:10.1017/S0960129522000433

PAPER

On reduction and normalization in the computational core

Claudia Faggian[1], Giulio Guerrieri[2], Ugo de' Liguoro[3] and Riccardo Treglia[4]

[1]Université de Paris Cité, IRIF, CNRS, F-75013 Paris, France, [2]Aix Marseille University, CNRS, LIS UMR 7020, Marseille, France, [3]Università di Torino, Department of Computer Science, Turin, Italy and [4]Università di Bologna, DISI, Bologna, Italy

*Corresponding author. Email: riccardo.treglia@unibo.it

(Received 23 April 2021; revised 29 November 2022; accepted 10 December 2022; first published online 16 January 2023)

Abstract

We study the reduction in a λ-calculus derived from Moggi's computational one, which we call the computational core. The reduction relation consists of rules obtained by orienting three monadic laws. Such laws, in particular associativity and identity, introduce intricacies in the operational analysis. We investigate the central notions of returning a value versus having a normal form and address the question of normalizing strategies. Our analysis relies on factorization results.

Keywords: Computational lambda-calculus; rewriting; factorization; normalization

1. Introduction

The λ-calculus has been historically conceived as an equational theory of functions, so that reduction had an ancillary role in Church's view, and it was a tool for studying the theory β, see Barendregt (1984, Ch. 3). The development of functional programming languages like Lisp and ML, and of proof assistants like LCF, has brought a new, different interest in the λ-calculus and its *reduction* theory.

The cornerstone of this change in perspective is Plotkin's (1975), where the functional parameter passing mechanism is formalized by the *call-by-value rewrite rule* β_v, allowing reduction only if the argument term is a *value*, that is a variable or an abstraction. In Plotkin (1975), it is also introduced the notion of *weak evaluation*, namely no reduction in the body of a function (i.e., of an abstraction).

This is now the standard evaluation implemented by functional programming languages, where *values* are the terms of interest (and the normal forms for weak evaluation in the closed case). Full β_v reduction is instead the basis of proof assistants like Coq, where *normal forms* are the result of interest. More generally, the computational perspective on λ-calculus has given a central role to reduction, whose theory provides a sound framework for reasoning about program transformations, such as compiler optimizations or parallel implementations.

The rich variety of computational effects in actual implementations of functional programming languages brings further challenges. This dramatically affects the theory of reduction of the calculi formalizing such features, whose proliferation makes it difficult to focus on suitably general issues. A major change here is the discovery by Moggi (1988, 1989, 1991) of a whole family of calculi that are based on a few common traits, combining call-by-value with the abstract notion of effectful computation represented by a *monad*, which has shown to be quite successful. But Moggi's *computational λ-calculus* is an equational theory in the broader sense; much less is known of the reduction theory of such calculi: this is the focus of our paper.

The Computational Calculus. Since Moggi's seminal work, computational λ-calculi have been developed as a foundation of programming languages, formalizing both functional and non-functional features, see e.g. Wadler and Thiemann (2003), Benton et al. (2002), starting a thread in the literature that is still growing. The basic idea of computational λ-calculi is to distinguish *values* and *computations*, so that programs, represented by closed terms, are thought of as functions from values to computations. Intuitively, computations embody a richer structure than values and do form a larger set in which values can be embedded. On the other hand, the essence of programming is composition; to compose functions from values to computations we need a mechanism to uniformly extend them to functions of computations, while preserving their original behavior over the (image of) values.

To model these concepts, Moggi used the categorical notion of *monad*, abstractly representing the extension of the space of values to that of computations, and the associated Kleisli category, whose morphisms are functions from values to computations, which are the denotations of programs. Syntactically, following Wadler (1995), we can express these ideas by means of a call-by-value λ-calculus with two sorts of terms: *values*, ranged over by V, W, namely variables or abstractions, and *computations* denoted by L, M, N. Computations are formed by means of two operators: values are embedded into computations by means of the operator *unit* written `return` in Haskell programming language, whose name refers to the unit of a monad in categorical terms; a computation $M \star (\lambda x.N)$ is formed by the binary operator $\star$, called *bind* (»= in Haskell), representing the application to M of the extension to computations of the function $\lambda x.N$.

The Monadic Laws. The operational understanding of these new operators is that evaluating $M \star (\lambda x.N)$, which in Moggi's notation reads $\mathsf{let}\ x{:=}M\ \mathsf{in}\ N$, amounts to first evaluating M until a computation of the form *unit* V is reached, representing the trivial computation that returns the value V. Then V is passed to N by binding x to V, as expressed by the identity:

$$(unit\ V) \star \lambda x.N = N[V/x] \tag{1}$$

This is the first of the three *monadic laws* in Wadler (1995). The remaining laws are

$$M \star \lambda x.unit\, x = M \tag{2}$$

$$(L \star \lambda x.M) \star \lambda y.N = L \star \lambda x.(M \star \lambda y.N) \quad \text{with } x \notin \mathsf{fv}(N) \tag{3}$$

To understand these two last rules, let us define the composition (named Kleisli composition in category theory) of the functions $\lambda x.M$ and $\lambda y.N$ as

$$(\lambda x.M) \bullet (\lambda y.N) := \lambda x.(M \star (\lambda y.N))$$

where we can freely assume that x is not free in N.

Equality (2) (*identity*) implies that $(\lambda z.M) \bullet (\lambda x.unit\, x) = \lambda z.M$, which paired with the instance of (1): $(\lambda x.unit\, x) \bullet (\lambda y.N) = \lambda x.N[x/y] =_\alpha \lambda y.N$ (where $=_\alpha$ is the usual congruence generated by the renaming of bound variables), tells that $\lambda x.unit\, x$ is the identity of composition $\bullet$.

Equality (3) (*associativity*) implies:

$$((\lambda z.L) \bullet (\lambda z.M)) \bullet (\lambda y.N) = (\lambda z.L) \bullet ((\lambda z.M) \bullet (\lambda y.N))$$

namely that composition $\bullet$ is associative.

The monadic laws correspond to the three equalities in the definition of a Kleisli triple (Moggi 1991), which is an equivalent presentation of monads (MacLane 1997). Indeed, Moggi's calculus is the internal language of a suitable category equipped with a (strong) monad T, and with enough structure to internalize the morphisms of the respective Kleisli category. As such, it is a simply typed λ-calculus, where T is the type constructor associating with each type A the type TA

of computations over A. Therefore, *unit* and $\star$ are polymorphic operators with respective types (Wadler 1992, 1995):

$$unit : A \to TA \qquad \star : TA \to (A \to TB) \to TB \tag{4}$$

The Computational Core. The dynamics of λ-calculi is usually defined as a reduction relation on untyped terms. Moggi's preliminary report (Moggi 1988) specifies both an equational and, in Section 6, a *reduction* system even if only the former is thoroughly investigated and appears in Moggi (1989, 1991), while reduction is briefly treated for an untyped fragment of the calculus. However, when stepping from the typed calculus to the untyped one, we need to be careful by avoiding meaningless terms to creep into the syntax, so jeopardizing the calculus theory. For example: what should be the meaning of $M \star N$ where both M and N are computations? What about $(\lambda x.N) \star V$ for any V? Shall we have functional applications of any kind?

To answer these questions, in de' Liguoro and Treglia (2020) typability is taken as syntactic counterpart of being meaningful: inspired by ideas in Scott (1980), the untyped computational λ-calculus is a special case of the typed one, where there are just two types D and TD, related by the type equation $D = D \to TD$, that is Moggi's isomorphism of the call-by-value reflexive object (see Moggi 1988, Section 5). With such a proviso, we get the following syntax:

$$\begin{aligned} V, W &::= x \mid \lambda x.M && (Val) \\ M, N, L &::= unit\ V \mid M \star V && (Com) \end{aligned}$$

If we assume that all variables have type D, then it is easy to see that all terms in *Val* have type $D = D \to TD$, which is consistent with the substitution of variables with values in (1). On the other hand, considering the typing of *unit* and $\star$ in (4), terms in *Com* have type *TD*. As we have touched above, there is some variety in notation among computational λ-calculi; we choose the above syntax because it explicitly embodies the essential constructs of a λ-calculus with monads, but for functional application, which is definable: see Section 3 for further explanations. We dub the calculus *computational core*, noted $\lambda_©$.

From Equalities to Reduction. Similarly to Moggi (1988) and Sabry and Wadler (1997), the reduction rules in the computational core $\lambda_©$ are the relation obtained by orienting the monadic laws from left to right. We indicate by β_c, id, and σ the rules corresponding to (1), (2), and (3), respectively. The contextual closure of these rules, noted $\to_©$, has been proved confluent in de' Liguoro and Treglia (2020), which implies that equal terms have a common reduct and the uniqueness of normal forms.

In Plotkin (1975) call-by-value reduction $\to_{\beta_v}$ is an intermediate concept between the equational theory and the evaluation relation $\underset{\mathsf{w}}{\to}_{\beta_v}$, that models an abstract machine. Evaluation consists of persistently choosing the leftmost β_v-redex that is not in the scope of an abstraction, i.e., evaluation is *weak*.

The following crucial result bridges reduction (hence, the foundational calculus) with evaluation (implemented by an ideal programming language):

$$M \to^*_{\beta_v} V \text{ (for some value } V\text{) if and only if } M \underset{\mathsf{w}}{\to}^*_{\beta_v} V' \text{ (for some value } V'\text{)} \tag{5}$$

Such a result (Corollary 1 in Plotkin 1975) comes from an analysis of the *reduction* properties of $\to_{\beta_v}$, namely standardization.

As we will see, the rules induced by associativity and identity make the behavior of the reduction in $\lambda_©$ – and the study of its operational properties – *nontrivial* in the setting of any monadic λ-calculus. The issues are inherent to the rules coming from the monadic laws (2) and (3), independently of the syntactic representation of the calculus that internalizes them. The difficulty appears clearly if we want to follow a similar route to Plotkin (1975), as we discuss next.

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

Reduction vs. Evaluation. Following Felleisen (1988), reduction $\to_{\circledcirc}$ and evaluation $\underset{\mathsf{w}}{\to}_{\circledcirc}$ of $\lambda_{\circledcirc}$ can be defined as the closure of the reduction rules under arbitrary and evaluation contexts, respectively. Consider the following grammars:

$$
\begin{array}{ll}
\mathsf{C} ::= \langle\,\rangle \mid \mathit{unit}\,(\lambda x.\mathsf{C}) \mid \mathsf{C} \star V \mid M \star (\lambda x.\mathsf{C}) & \text{(arbitrary) contexts} \\
\mathsf{E} ::= \langle\,\rangle \mid \mathsf{E} \star V & \text{evaluation contexts}
\end{array}
$$

where the hole $\langle\,\rangle$ can be filled by terms in *Com*, only. Observe that the closure under evaluation context E is precisely weak reduction.

Weak reduction of $\lambda_{\circledcirc}$, however, turns out to be nondeterministic, nonconfluent, and its normal forms are *not unique*. The following is a counterexample to all such properties – see Section 5 for further examples.

$$
\begin{array}{ccc}
((\mathit{unit}\, z \star z) \star \lambda x.\, M) \star \lambda y.\, \mathit{unit}\, y & \xrightarrow[\mathsf{w}]{} & (\mathit{unit}\, z \star z) \star \lambda x.\, (M \star \lambda y.\, \mathit{unit}\, y) \\
\downarrow \mathsf{w} & & \\
(\mathit{unit}\, z \star z) \star \lambda x.\, M & &
\end{array}
$$

Such an issue is not specific to the syntax of the computational core. The same phenomena show up with the *let*-notation, more commonly used in computational calculi. Here, evaluation, usually called *sequencing*, is the reduction defined by the following contexts (Filinski 1996; Jones et al. 1998; Levy et al. 2003):

$$\mathsf{E}_{let} ::= \langle\,\rangle \mid \mathsf{let}\; x{:=}\mathsf{E}_{let}\; \mathsf{in}\; N.$$

Examples similar to the one above can be reproduced. We give the details in Example 5.3.

1.1 Content and contributions

The focus of this paper is an *operational* analysis of two crucial properties of a term M:

(i) M returns a *value* (i.e. $M \to_{\circledcirc}^{*} \mathit{unit}\, V$, for some V value).
(ii) M has a *normal form* (i.e. $M \to_{\circledcirc}^{*} N$, for some N $\circledcirc$-normal).

As in Accattoli et al. (2019), the cornerstone of our analysis are *factorization* results (also called *semi-standardization* in the literature): any reduction sequence can be reorganized so as to first performing specific steps and then everything else.

Via factorization, we show the key result (6), analogous to (5), relating reduction and *evaluation*:

$$M \to_{\circledcirc}^{*} \mathit{unit}\, V \text{ (for some value } V) \iff M \underset{\mathsf{w}}{\to}_{\beta_c}^{*} \mathit{unit}\, V' \text{ (for some value } V') \tag{6}$$

We then analyze the property of having a normal form (*normalization*), and define a family of *normalizing strategies*, i.e., subreductions that are guaranteed to reach a normal form, if any exists.

On the Rewrite Theory of Computational Calculi. In this paper, we study the rewrite theory of a specific computational calculus, namely $\lambda_{\circledcirc}$. We expose a number of issues, which we argue to be intrinsic to the monadic rules of computational calculi, namely associativity and identity. Indeed, the same issues which we expose in $\lambda_{\circledcirc}$, also appear in other computational calculi, as we discuss in Section 5, where we take as reference the calculus in Sabry and Wadler (1997), which we recall in Section 3.1. We expect that the solutions we propose for $\lambda_{\circledcirc}$ could be adapted also there.

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

Surface Reduction. The form of weak reduction that we defined in the previous section (*sequencing*) is standard in the literature. In this paper, we study also a less strict form of weak reduction, namely *surface reduction*, which is less constrained and better behaved then sequencing. Surface reduction disallows reduction under the *unit* operator only, and not under abstractions. Intuitively, weak reduction does not act in the body of a function, while surface reduction does not act in the scope of `return`. As we discuss in Section 3.1, it can also be seen as a more natural extension of call-by-value weak reduction to a computational calculus.

Surface reduction is well studied in the literature because it naturally arises when interpreting λ-calculus into linear logic, and indeed the name surface (which we take from Simpson 2005) is reminiscent of a similar notion in calculi based on linear logic (Ehrhard and Guerrieri 2016; Simpson 2005). In Section 4, we will make explicit the correspondence with such calculi, showing that the *unit* operator (from the computational core) behaves exactly like a bang ! (from linear logic).

Identity and Associativity. Our analysis exposes the operational role of the rules associated to the monadic laws of identity and associativity.

(i) To compute a *value*, only β_c steps are necessary.
(ii) To compute a *normal form*, β_c steps do not suffice: *associativity* (i.e., σ steps) is *necessary*.

Hence, the rule associated to the identity law turns out to be operationally *irrelevant*.

Normalization. The study of normalization is more complex than that of evaluation and requires some sophisticated techniques. We highlight some specific contributions.

- We define two families of *normalizing strategies* in $\lambda_©$. The first one, quite constrained, relies on an *iteration of weak reduction* $\underset{w}{\rightarrow}_{\lambda_©}$. The second one, more liberal, is based on an *iteration of surface reduction* $\underset{s}{\rightarrow}_{\lambda_©}$. The definition and proof of normalization is *parametric* on either.
- The technical *difficulty* in the proofs for normalization comes from the fact that neither weak nor surface reduction is deterministic. To deal with that we rely on a fine *quantitative analysis* of the number of β_c steps, which we carry-on when we study factorization in Section 6.

The most challenging proofs in the paper are those related to normalization via surface reduction. The effort is justified by the interest in a larger and *more versatile* strategy, which then does not induce a single abstract machine but *subsumes* several ones, each following a different reduction policy. It thus facilitates reasoning about optimization techniques and parallel implementation.

A Roadmap. Let us summarize the structure of the paper.

Section 2 contains the background notions which are relevant to our paper.

Section 3 gives the formal definition of the computational core $\lambda_©$ and its reduction.

In Sections 4 and 5, we analyze the properties of weak and surface reduction. We first study $\rightarrow_{\beta_c}$, and then we move to the whole $\lambda_©$, where associativity and identity also come to play, and issues appear.

In Section 6 we study several factorization results. The cornerstone of our construction is surface factorization (Theorem 6.1). We then further refine this result, first by postponing the id steps which are not β_c steps, and then with a form of weak factorization.

In Section 7, we study evaluation and analyze some relevant consequences of this result. We actually provide two different ways to deterministically compute a value. The first way is the one given by (6), via an *evaluation context*. The second way requires no contextual closure at all: simply applying β_c- and σ-*rules* will return a value, if possible.

In Section 8 we study normalization and normalizing strategies.

Section 9 concludes with final discussions and related work.

2. Preliminaries

2.1 Basics on rewriting

We recall here some standard definitions and notations in rewriting that we shall use in this paper (see for instance Terese 2003 or Baader and Nipkow 1998 for details).

Rewriting System. An *abstract rewriting system (ARS)* is a pair $(A, \to)$ consisting of a set A and a binary relation $\to$ on A whose pairs are written $t \to s$ and called *steps*. A $\to$*-sequence* from $t \in A$ is a sequence $(t_i \to t_{i+1})_{i \in I}$ of $\to$ steps, where $I = \mathbb{N}$ or $I = \{0, 1, \ldots, n-1\}$ for some $n \in \mathbb{N}$, $t_i \in A$ for all $i \in I$ and $t_0 = t$ (in particular, the sequence is empty for $I = \emptyset$, i.e. $n = 0$). We denote by $\to^*$ (resp. $\to^=$; $\to^+$) the transitive-reflexive (resp. reflexive; transitive) closure of $\to$, and $\leftarrow$ stands for the *transpose* of $\to$, that is, $u \leftarrow t$ if $t \to u$. We write $t \to^k s$ for a $\to$-sequence $t \to t_1 \to \ldots \to t_k = s$ of $k \in \mathbb{N}$ steps. If $\to_1, \to_2$ are binary relations on A then $\to_1 \cdot \to_2$ denotes their composition, i.e. $t \to_1 \cdot \to_2 s$ if there exists $u \in A$ such that $t \to_1 u \to_2 s$. We often set $\to_{12} := \to_1 \cup \to_2$.

A relation $\to$ is *deterministic* if for each $t \in A$ there is at most one $s \in A$ such that $t \to s$. It is *confluent* if $\leftarrow^* \cdot \to^* \subseteq \to^* \cdot \leftarrow^*$.

We say that $u \in A$ is $\to$*-normal* (or a $\to$*-normal form*, noted $u \not\to$) if $u \not\to t$ for all $t \in A$, that is, there is no $t \in A$ such that $u \to t$; and we say that $t \in A$ *has a normal form* u if $t \to^* u$ with u $\to$-normal. Confluence implies that each $t \in A$ has *unique normal form*, if any exists.

Normalization. Let $(A, \to)$ be an ARS. In general, a term may or may not reduce to a normal form. And if it does, not all reduction sequences necessarily lead to a normal form. A term is *weakly* or *strongly normalizing*, depending on if it may or must reduce to normal form. If a term t is strongly normalizing, any choice of steps will eventually lead to a normal form. However, if t is weakly normalizing, how do we compute a normal form? This is the problem tackled by *normalization* and *normalizing strategies*: by repeatedly performing *only specific steps*, a normal form will be computed, provided that t can reduce to any. We recall two important notions of normalization.

Definition 2.1 (Normalizing)**.** Let $(A, \to)$ be an ARS and $t \in A$.

(1) t is *strongly* $\to$*-normalizing* (or terminating) if every maximal $\to$-sequence from t ends in a normal form (i.e., t has no infinite $\to$-sequence).
(2) t is *weakly* $\to$*-normalizing* (or just normalizing) if there exists a $\to$-sequence from t that ends in a $\to$-normal form (i.e., t has a $\to$-normal form).

Reduction $\to$ is *strongly* (resp. *weakly*) *normalizing* if so is every $t \in A$. Reduction $\to$ is uniformly normalizing if every weakly $\to$-normalizing $t \in A$ is also strongly $\to$-normalizing.

Clearly, strong normalization implies weak normalization, and any deterministic reduction is uniformly normalizing.

A *normalizing strategy* for $\to$ is a reduction strategy which, given a term t, is guaranteed to reach its $\to$-normal form, if any exists.

Definition 2.2 (Normalizing strategies)**.** A subreduction $\underset{e}{\to} \subseteq \to$ is a normalizing strategy for $\to$ if $\underset{e}{\to}$ has the same normal forms as $\to$, and for all $t \in A$, if t has a $\to$-normal form, then *every* maximal $\underset{e}{\to}$-sequence from t ends in a $\to$-normal form.

Note that in Definition 2.2, $\underset{e}{\to}$ need not be deterministic, and $\underset{e}{\to}$ and $\to$ need not be confluent.

Factorization. In this paper, we will extensively use factorization results.

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

Definition 2.3 (Factorization, postponement)**.** Let $(A, \to)$ be an ARS with $\to \ = \ \underset{e}{\to} \cup \underset{i}{\to}$.

Relation $\to$ satisfies e-*factorization*, written $\mathtt{Fact}(\underset{e}{\to}, \underset{i}{\to})$, if

$$\mathtt{Fact}(\underset{e}{\to}, \underset{i}{\to}): \quad (\underset{e}{\to} \cup \underset{i}{\to})^* \subseteq \underset{e}{\to}^* \cdot \underset{i}{\to}^* \qquad \textbf{(Factorization)}$$

Relation $\underset{i}{\to}$ postpones after $\underset{e}{\to}$, written $\mathtt{PP}(\underset{e}{\to}, \underset{i}{\to})$, if

$$\mathtt{PP}(\underset{e}{\to}, \underset{i}{\to}): \quad \underset{i}{\to}^* \cdot \underset{e}{\to}^* \subseteq \underset{e}{\to}^* \cdot \underset{i}{\to}^* \qquad \textbf{(Postponement)}$$

It is an easy result that e-factorization is equivalent to postponement, which is a more convenient way to express it.

Lemma 2.4. *The following are equivalent (for any two relations* $\underset{e}{\to}, \underset{i}{\to}$*):*

(1) Postponement*:* $\mathtt{PP}(\underset{e}{\to}, \underset{i}{\to})$.
(2) Factorization*:* $\mathtt{Fact}(\underset{e}{\to}, \underset{i}{\to})$.

Hindley (1964) first noted that a local property implies factorization. Let $\to = \underset{e}{\to} \cup \underset{i}{\to}$. We say that $\underset{i}{\to}$ *strongly postpones* after $\underset{e}{\to}$, if

$$\mathtt{SP}(\underset{e}{\to}, \underset{i}{\to}): \quad \underset{i}{\to} \cdot \underset{e}{\to} \subseteq \underset{e}{\to}^* \cdot \underset{i}{\to}^= \qquad \textbf{(Strong Postponement)}$$

Lemma 2.5 (Hindley 1964)**.** $\mathtt{SP}(\underset{e}{\to}, \underset{i}{\to})$ *implies* $\mathtt{Fact}(\underset{e}{\to}, \underset{i}{\to})$.

Observe that the following are special cases of strong postponement. The first one is *linear* in $\underset{e}{\to}$; we refer to it as *linear postponement*. In the second one, recall that $\to \ = \ \underset{e}{\to} \cup \underset{i}{\to}$.

(1) $\underset{i}{\to} \cdot \underset{e}{\to} \subseteq \underset{e}{\to} \cdot \underset{i}{\to}^=$.
(2) $\underset{i}{\to} \cdot \underset{e}{\to} \subseteq \underset{e}{\to} \cdot \to$.

Linear variants of postponement can easily be adapted to *quantitative* variants, which allow us to "count the steps" and are useful to establish termination properties. We do this in Section 6.3.

Diamonds. We recall also another *quantitative* result, which we will use.

Fact 2.6 (Newman 1842)**.** *In an ARS* $(A, \to)$*, if* $\to$ *is quasi-diamond, then it has random descent, where quasi-diamond and random descent are defined below.*

(1) ***Quasi-Diamond****: For all* $t \in A$*, if* $t_1 \leftarrow t \to t_2$*, then* $t_1 = t_2$ *or* $t_1 \to u \leftarrow t_2$ *for some* u*.*
(2) ***Random Descent****: For all* $t \in A$*, all maximal* $\to$*-sequences from* t *have the* same number of steps, *and all end in the same normal form, if any exists.*

Clearly, if $\to$ is quasi-diamond then it is confluent and uniformly normalizing.

Postponement, Confluence and Commutation. Both postponement and confluence are commutation properties. Two relations $\triangleright$ and $\blacktriangleright$ on A *commute* if

$$\triangleleft^* \cdot \blacktriangleright^* \subseteq \blacktriangleright^* \cdot \triangleleft^* . \qquad \textbf{(Commutation)}$$

So, a relation $\to$ is confluent if and only if it commutes with itself. Postponement and commutation can be defined in terms of each other, simply taking $\underset{i}{\to}$ for $\triangleleft$ and $\underset{e}{\to}$ for $\blacktriangleright$ ($\underset{i}{\to}$ postpones

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

after $\underset{e}{\rightarrow}$ if and only if $\underset{i}{\leftarrow}$ commutes with $\underset{e}{\rightarrow}$). As propounded in van Oostrom (2020*b*), this fact allows for proving postponement by means of *decreasing diagrams* (van Oostrom 1994, 2008). This is a powerful and general technique to prove commutation properties: it reduces the problem of showing commutation to a *local* test; in exchange for localization, diagrams need to be decreasing with respect to a labeling.

Definition 2.7 (Decreasing). Let $\triangleright := \bigcup_{k\in K} \triangleright_k$ and $\blacktriangleright := \bigcup_{j\in J} \blacktriangleright_j$. The pair of relations $\triangleright, \blacktriangleright$ is *decreasing* if for some well-founded strict order $<$ on the set of labels $K \cup J$ the following holds:

$$\triangleleft_k \cdot \blacktriangleright_j \subseteq (\blacktriangleright^*_{\langle k\rangle} \cdot \blacktriangleright^=_j \cdot \blacktriangleright^*_{\langle k,j\rangle}) \cdot (\triangleleft^*_{\langle j\rangle} \cdot \triangleleft^=_k \cdot \triangleleft^*_{\langle k,j\rangle}) \qquad \text{for every } k \in K, j \in J$$

where $\langle L\rangle = \{i \in K \cup J \mid \exists l \in L.\, l > i\}$ for any $L \subseteq K \cup J$, and $\langle i_1, \ldots, i_n\rangle = \langle\{i_1, \ldots, i_n\}\rangle$.

Theorem 2.8 (Decreasing diagram van Oostrom 1994). *A pair of relations* $\triangleright, \blacktriangleright$ *commutes if it is decreasing.*

Modularizing Confluence. A classic tool to modularize a proof of confluence is Hindley–Rosen lemma: the union of confluent reductions is itself confluent if they all commute with each other.

Lemma 2.9 (Hindley–Rosen). *Let* $\rightarrow_1$ *and* $\rightarrow_2$ *be relations on a set A. If* $\rightarrow_1$ *and* $\rightarrow_2$ *are confluent and commute with each other, then* $\rightarrow_1 \cup \rightarrow_2$ *is confluent.*

Like for postponement, strong commutation implies commutation.

Lemma 2.10 (Strong commutation Hindley 1964). *Strong commutation* ($\leftarrow_1 \cdot \rightarrow_2 \;\subseteq\; \rightarrow_2{}^* \cdot \leftarrow_1{}^=$) *implies commutation.*

2.2 Basics on the λ-calculus

We recall the syntax and some relevant notions of the λ-calculus, taking Plotkin's call-by-value (*CbV*, for short) λ-calculus (Plotkin 1975) as a concrete example.

Terms and values are mutually generated by the grammars below.

$$V ::= x \mid \lambda x.T \quad (\textit{values}; \text{ set: } \mathit{Val}) \qquad T, S, R ::= V \mid TS \quad (\textit{terms}; \text{ set: } \Lambda)$$

where x ranges over a countably infinite set *Var* of *variables*. Terms of shape TS and $\lambda x.T$ are called *applications* and *abstractions*, respectively. In $\lambda x.T$, λx binds the occurrences of x in T. The set of *free* (i.e. non-bound) variables of a term T is denoted by $\mathsf{fv}(T)$. Terms are identified up to (clash-avoiding) renaming of their bound variables (*α-congruence*).

Reduction.

- Contexts (with exactly one *hole* $\langle\,\rangle$) are generated by the grammar

  $$\mathsf{C} ::= \langle\,\rangle \mid T\mathsf{C} \mid \mathsf{C}T \mid \lambda x.\mathsf{C} \qquad (\textit{Contexts})$$

  $\mathsf{C}\langle T\rangle$ stands for the term obtained from C by replacing the hole with the term T (possibly capturing some free variables of T).
- A rule ρ is a binary relation on Λ, also noted $\mapsto_\rho$, writing $R \mapsto_\rho R'$; R is called a *ρ-redex*.
- A reduction step $\rightarrow_\rho$ is the closure of ρ under contexts C. Explicitly, if $T, S \in \Lambda$ then $T \rightarrow_\rho S$ if $T = \mathsf{C}\langle R\rangle$ and $S = \mathsf{C}\langle R'\rangle$, for some context C and some $R \mapsto_\rho R'$.

The Call-by-Value λ-calculus. The CbV λ-calculus is the rewrite system $(\Lambda, \rightarrow_{\beta_v})$, the set of terms Λ equipped with *β_v-reduction* $\rightarrow_{\beta_v}$, that is, the contextual closure of the rule $\mapsto_{\beta_v}$:

$$(\lambda x.T)V \mapsto_{\beta_v} T[V/x] \quad (V \in \mathit{Val})$$

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

where $T[V/x]$ is the term obtained from T by capture-avoiding substitution of V for the free occurrences of x in T. Notice that here β-redexes can be fired only when the argument is a *value*.

Weak evaluation (which does not reduce in the body of a function) evaluates closed terms to values. In the literature of CbV, there are three main weak schemes: reducing from left to right, as defined by Plotkin (1975), from right to left (Leroy 1990), or in an arbitrary order (Lago and Martini 2008). *Left* contexts L, *right* contexts R, and (arbitrary order) *weak* contexts W are, respectively, defined by

$$\mathsf{L} ::= \langle\,\rangle \mid \mathsf{L}\,T \mid V\mathsf{L} \qquad \mathsf{R} ::= \langle\,\rangle \mid T\mathsf{R} \mid \mathsf{R}V \qquad \mathsf{W} ::= \langle\,\rangle \mid \mathsf{W}T \mid T\mathsf{W}$$

Given a rule $\mapsto_\rho$ on Λ, *weak reduction* $\underset{\mathsf{w}}{\rightarrow}_\rho$ is the closure of $\mapsto_\rho$ under weak contexts W; *non-weak reduction* $\underset{\neg\mathsf{w}}{\rightarrow}_\rho$ is the closure of $\mapsto_\rho$ under contexts C that are not weak. *Left* and *non-left* reductions ($\underset{\mathsf{l}}{\rightarrow}_\rho$ and $\underset{\neg\mathsf{l}}{\rightarrow}_\rho$), *right* and *non-right* reductions ($\underset{\mathsf{r}}{\rightarrow}_\rho$ and $\underset{\neg\mathsf{r}}{\rightarrow}_\rho$) are defined analogously.

Note that $\underset{\mathsf{l}}{\rightarrow}_{\beta_v}$ and $\underset{\mathsf{r}}{\rightarrow}_{\beta_v}$ are deterministic, whereas $\underset{\mathsf{w}}{\rightarrow}_{\beta_v}$ is not.

CbV Weak Factorization. Factorization of $\rightarrow_{\beta_v}$ allows for a characterization of the terms which reduce to a value. Convergence below is a remarkable consequence of factorization.

Theorem 2.11 (Weak left factorization Plotkin 1975)**.**

(1) Left Factorization of $\rightarrow_{\beta_v}$: $\rightarrow^*_{\beta_v} \subseteq \underset{\mathsf{l}}{\rightarrow}_{\beta_v}{}^* \cdot \underset{\neg\mathsf{l}}{\rightarrow}_{\beta_v}{}^*$.

(2) Value Convergence: $T \rightarrow^*_{\beta_v} V$ *for some value* V *if and only if* $T \underset{\mathsf{l}}{\rightarrow}_{\beta_v}{}^* V'$ *for some value* V'.

The same results hold for $\underset{\mathsf{w}}{\rightarrow}_{\beta_v}$ *and* $\underset{\mathsf{r}}{\rightarrow}_{\beta_v}$ *in place of* $\underset{\mathsf{l}}{\rightarrow}_{\beta_v}$.

Since the $\underset{\mathsf{l}}{\rightarrow}_{\beta_v}$-normal forms of closed terms are exactly closed values, Theorem 2.11.2 means that every closed term T β_v-reduces to a value if and only if $\underset{\mathsf{l}}{\rightarrow}_{\beta_v}$-reduction from T terminates.

3. The Computational Core $\lambda_©$

We recall the syntax and the reduction of the *computational core*, shortly $\lambda_©$, introduced in de' Liguoro and Treglia (2020).

We use a notation slightly different from the one used in de' Liguoro and Treglia (2020) (and recalled in Section 1). Such a syntactical change is convenient both to present the calculus in a more familiar fashion, and to establish useful connections between $\lambda_©$ and two well-known calculi, namely Simpson's calculus (Simpson 2005) and Plotkin's call-by-value λ-calculus (Plotkin 1975).

The equivalence between the current presentation of $\lambda_©$ and de' Liguoro and Treglia (2020) is detailed in Appendix E.

Definition 3.1 (Terms of $\lambda_©$)**.** Terms of the computational core consist of two sorts of expressions:

$$\begin{array}{lll} \textit{Val}: & V, W ::= x \mid \lambda x.M & \text{(values)} \\ \textit{Com}: & M, N ::= !V \mid VM & \text{(computations)} \end{array}$$

where x ranges over a countably infinite set *Var* of variables. We set $\textit{Term} := \textit{Val} \cup \textit{Com}$; $\mathsf{fv}(V)$ and $\mathsf{fv}(M)$ are the sets of free variables occurring in V and M, respectively, and are defined as usual. Terms are identified up to clash-avoiding renaming of bound variables (α*-congruence*).

The unary operator ! is just another notation for *unit* as presented in Section 1: it coerces a value V into a computation $!V$, sometimes called *returned value*.

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

Remark 3.2 (Application)**.** A computation VM is a *restricted* form of application, corresponding to the term $M \star V$ in Wadler (1995) (see Section 1) where there is no functional application. The reason is that the bind $\star$ represents an effectful form of application, such that by redefining the unit and bind one obtains an actual evaluator for the desired computational effects (Wadler 1995). This restriction may seem a strong limitation because we apparently cannot express iterated applications: $(VM)N$ is not well formed in $\lambda_{©}$. However, application among computations is definable in $\lambda_{©}$:

$$MN := (\lambda z.zN)M \qquad \text{where } z \notin \mathsf{fv}(N).$$

Reduction. The operational semantics of $\lambda_{©}$ puts together rules corresponding to the monad laws.

Definition 3.3 (Reduction)**.** Relation $\mapsto_{©} = \mapsto_{\beta_c} \cup \mapsto_{\mathsf{id}} \cup \mapsto_{\sigma}$ is the union of the following rules:

$$\begin{array}{lll} \beta_c) & (\lambda x.M)(!V) \mapsto_{\beta_c} M[V/x] & \\ \mathsf{id}\,) & (\lambda x.!x)M \mapsto_{\mathsf{id}} M & \\ \sigma\,) & (\lambda y.N)((\lambda x.M)L) \mapsto_{\sigma} (\lambda x.(\lambda y.N)M)L & \text{for } x \notin \mathsf{fv}(N) \end{array}$$

For every $\rho \in \{\beta_c, \sigma, \mathsf{id}, ©\}$, *reduction* $\to_\rho$ is the contextual closure of $\mapsto_\rho$, where *contexts* are defined as follows:

$$\mathsf{C} ::= \langle\,\rangle \mid !(\lambda x.\mathsf{C}) \mid V\mathsf{C} \mid (\lambda x.\mathsf{C})M \qquad \text{Contexts}$$

All reductions in Definition 3.3 are binary relations on *Com*, thanks to the proposition below.

Proposition 3.4. *The set of computations Com is closed under substitution and reduction:*

(1) If $M \in Com$ and $V \in Val$, then $M[V/x] \in Com$.
(2) For every $\rho \in \{\beta_c, \sigma, \mathsf{id}, ©\}$, if $N \to_\rho N'$, then: $N \in Com$ if and only if $N' \in Com$.

Proof. Point 1 (formally proved by induction on M) holds because $M[V/x]$ just replaces a value, x, with another value, V. Point 2 is proved by induction on the context for $N \to_\rho N'$, using Point 1. □

The *computational core* $\lambda_{©}$ is the rewriting system $(Com, \to_{©})$.

Proposition 3.5 (Confluence, de' Liguoro and Treglia 2020)**.** *Reduction $\to_{©}$ is confluent.*

Remark 3.6 (β_c and β_v)**.** The relation between $\to_{\beta_c}$ of the computational core and $\to_{\beta_v}$ of Plotkin's CbV λ-calculus is investigated in Appendix F. To give a taste of it, we show with an example how β_v-reduction is simulated by β_c-reduction, possibly with more steps. Since $\to_{©}$ is a relation on *Com* (Proposition 3.4), no computation N will ever reduce to any value V; however, reduction to values is represented by a reduction $N \to_{©}^{*} !V$, where $!V$ is the coercion of value V into a computation. Let us assume that $M \to_{\beta_c}^{*} !\lambda x.M'$ and $N \to_{\beta_c}^{*} !V$. We have:

$$\begin{array}{rcll} MN & = & (\lambda z.zN)M & \text{(by the encoding in Remark 3.2)} \\ & \to_{\beta_c}^{*} & (\lambda z.z!V)(!\lambda x.M') & \text{(where } z \notin \mathsf{fv}(V) \text{ since } z \notin \mathsf{fv}(N)) \\ & \to_{\beta_c} & (\lambda x.M')!V & \\ & \to_{\beta_c} & M'[V/x] & \end{array}$$

Similarly, in Plotkin's CbV λ-calculus, if $M \to_{\beta_v}^{*} \lambda x.M'$ and $N \to_{\beta_v}^{*} V$, then $MN \to_{\beta_v}^{*} M'[V/x]$.

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

Surface and Weak Reduction. As we shall see in the next sections, there are two natural restrictions of $\to_{©}$: *weak reduction* $\underset{\mathsf{w}}{\to}_{©}$ which does not fire in the scope of λ, and *surface reduction* $\underset{\mathsf{s}}{\to}_{©}$, which does not fire in the scope of !. The former is the evaluation usually studied in CbV λ-calculus (Theorem 2.11). The latter is the natural evaluation in linear logic, and in Simpson's calculus, whose relation with $\lambda_{©}$ we discuss in Section 4.

Surface and *weak contexts* are, respectively, defined by the grammars

$$\mathsf{S} ::= \langle\,\rangle \mid V\mathsf{S} \mid (\lambda x.\mathsf{S})M \qquad \text{Surface Contexts}$$
$$\mathsf{W} ::= \langle\,\rangle \mid V\mathsf{W} \qquad \text{Weak Contexts}$$

For $\rho \in \{\beta_c, \mathsf{id}, \sigma, ©\}$, *weak* reduction $\underset{\mathsf{w}}{\to}_{\rho}$ is the closure of ρ under weak contexts W, *surface* reduction $\underset{\mathsf{s}}{\to}_{\rho}$ is its closure under surface contexts S. *non-surface* reduction $\underset{\neg\mathsf{s}}{\to}_{\rho}$ is the closure of ρ under contexts C that are not surface. Similarly, *nonweak* reduction $\underset{\neg\mathsf{w}}{\to}_{\rho}$ is the closure of ρ under contexts C that are not weak.

Clearly, $\underset{\mathsf{w}}{\to}_{\rho} \subsetneq \underset{\mathsf{s}}{\to}_{\rho} \subsetneq \to_{\rho}$. Note that $\underset{\mathsf{w}}{\to}_{\beta_c}$ is a *deterministic* relation, while $\underset{\mathsf{s}}{\to}_{\beta_c}$ is not.

Example 3.7. *To clarify the difference between surface and weak, let us consider the term* $(\lambda x.\boldsymbol{I}!x)!\lambda y.\boldsymbol{I}!y$, *where* $\boldsymbol{I} = \lambda z.!z$, *and two different* $\to_{©}$ *steps from it. We underline the fired redex.*

$$(\lambda x.\underline{\boldsymbol{I}!x})!\lambda y.\boldsymbol{I}!y \underset{\mathsf{s}}{\to}_{©} (\lambda x.!x)!\lambda y.\boldsymbol{I}!y \qquad (\lambda x.\boldsymbol{I}!x)!\lambda y.\underline{\boldsymbol{I}!y} \underset{\mathsf{s}}{\not\to}_{©} (\lambda x.\boldsymbol{I}!x)!\lambda y.!y$$
$$(\lambda x.\underline{\boldsymbol{I}!x})!\lambda y.\boldsymbol{I}!y \underset{\mathsf{w}}{\not\to}_{©} (\lambda x.!x)!\lambda y.\boldsymbol{I}!y \qquad (\lambda x.\boldsymbol{I}!x)!\lambda y.\underline{\boldsymbol{I}!y} \underset{\mathsf{w}}{\not\to}_{©} (\lambda x.\boldsymbol{I}!x)!\lambda y.!y.$$

Surface reduction can be seen as the natural counterpart of weak reduction in calculi with *let*-constructors or explicit substitutions, as we show in Section 3.1.

Remark 3.8 (Weak contexts)**.** In the CbV λ-calculus (see Section 2.2,), weak contexts can be given in three forms, according to the order in which redexes that are not in the scope of abstractions are fired: L, R, W. When the grammar of terms is restricted to computations, the three coincide. So, in $\lambda_{©}$ there is *only one* definition of weak context, and weak, left and right reductions coincide.

In Sections 4 and 5, we analyze the *properties* of weak and surface reduction. We first study $\to_{\beta_c}$, and then we move to the whole $\lambda_{©}$, where σ and id also come to play.

Notation. In the rest of the paper, we adopt the following notation:

$$\mathbf{I} := \lambda x.!x \quad \text{and} \quad \boldsymbol{\Delta} := \lambda x.x!x.$$

3.1 The computational core vs. computational calculi with let-notation

It is natural to compare the computational core $\lambda_{©}$ with other untyped computational calculi, and wonder if the analysis of the rewriting theory of $\lambda_{©}$ we present in this paper applies to them. There is indeed a rich literature on computational calculi refining Moggi's λ_c (Moggi 1988, 1989, 1991), most of them use the *let* -constructor. A standard reference is Sabry and Wadler's λ_{ml^*} (Sabry and Wadler 1997, Section 5), which we display in Figure 1.

λ_{ml^*} has a two sorted syntax that separates *values* (i.e., variables and abstractions) and *computations*. The latter are either *let*-expressions (aka explicit substitutions, capturing monadic binding), or applications (of values to values), or coercions $[V]$ of values V into computations ($[V]$ is the notation for *unit* V in Sabry and Wadler (1997), so it corresponds to $!V$ in $\lambda_{©}$).

- The *reduction rules* in λ_{ml^*} are the usual β and η from Plotkin's *call-by-value* λ-calculus (Plotkin 1975), plus the oriented version of three *monad laws*: let.β, let.η, let.*ass* (see Figure 1).
- *Reduction* $\to_{ml^*}$ is the contextual closure of the union of these rules.

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

$$\textit{Values:}\quad V, W ::= x \mid \lambda x.M \qquad\qquad \textit{Computations:}\quad M, N ::= [V] \mid \mathsf{let}\ x{:=}\ M\ \mathsf{in}\ N \mid VW$$

$$\begin{array}{rll}
(c.\beta) & (\lambda x.M)V \to M[V/x] & \\
(c.\eta) & \lambda x.Vx \to V & x \notin \mathsf{fv}(V) \\
(c.\mathsf{let}.\beta) & \mathsf{let}\ x := [V]\ \mathsf{in}\ N \to N[V/x] & \\
(c.\mathsf{let}.\eta) & \mathsf{let}\ x := M\ \mathsf{in}\ [x] \to M & x \notin \mathsf{fv}(M) \\
(c.\mathsf{let}.ass) & \mathsf{let}\ y{:=}(\mathsf{let}\ x := L\ \mathsf{in}\ M)\ \mathsf{in}\ N \to \mathsf{let}\ x{:=}\ L\ \mathsf{in}\ (\mathsf{let}\ y{:=}\ M\ \mathsf{in}\ N) & x \notin \mathsf{fv}(N)
\end{array}$$

Figure 1. λ_{ml^*}: Syntax and reduction.

$$(\cdot)^\bullet : \lambda_{ml^*} \to \lambda_{©} \qquad\qquad (\cdot)^\circ : \lambda_{©} \to \lambda_{ml^*}$$

$$\begin{array}{rcl}
(x)^\dagger &:=& x \\
(\lambda x.M)^\dagger &:=& \lambda x.(M)^\bullet \\
([V])^\bullet &:=& !(V^\dagger) \\
(VW)^\bullet &:=& V^\dagger\ !(W^\dagger) \\
(\mathsf{let}\ x := M\ \mathsf{in}\ N)^\bullet &:=& (\lambda x.N^\bullet)M^\bullet
\end{array}
\qquad
\begin{array}{rcll}
(x)^\ddagger &:=& x & \\
(\lambda x.M)^\ddagger &:=& \lambda x.(M)^\circ & \\
(!V)^\circ &:=& [V^\ddagger] & \\
(V\ !W)^\circ &:=& V^\ddagger\, W^\ddagger & \\
(xM)^\circ &:=& \mathsf{let}\ y := M^\circ\ \mathsf{in}\ xy & \text{if } y \notin \mathsf{fv}(M),\ M \neq\, !V \text{ for any value } V \\
((\lambda x.N)M)^\circ &:=& \mathsf{let}\ x := M^\circ\ \mathsf{in}\ N^\circ & \text{if } M \neq\, !V \text{ for any value } V
\end{array}$$

Figure 2. Translations between λ_{ml^*} and $\lambda_{©}$.

To state a correspondence between $\lambda_{©}$ and λ_{ml^*}, consider the translations in Figure 2: translation $(\cdot)^\bullet$ from λ_{ml^*} to $\lambda_{©}$ (resp. $(\cdot)^\circ$ from $\lambda_{©}$ to λ_{ml^*}) is defined via the auxiliary encoding $(\cdot)^\dagger$ (resp. $(\cdot)^\ddagger$) for values. The translations induce an equational correspondence by adding η-equality to $\lambda_{©}$. More precisely, let $\to_\eta$ be the closure of the rule $\mapsto_\eta$ (below left) under contexts G (below right).

$$\lambda x.(V!x) \mapsto_\eta V \qquad\qquad \mathsf{V} ::= \langle\, \rangle \mid \lambda x.\mathsf{G} \qquad \mathsf{G} ::= !\mathsf{V} \mid \mathsf{V}M \mid V\mathsf{G}$$

Let $=_{©\eta}$ be the reflexive transitive and symmetric closure of the reduction $\to_{©\eta} = \to_{©} \cup \to_\eta$, and similarly for $=_{ml^*}$ with respect to $\to_{ml^*}$.

Proposition 3.9. *The following hold:*

(1) $M =_{©\eta} (M^\circ)^\bullet$ for every computation M in $\lambda_{©}$;
(2) $(P^\bullet)^\circ =_{ml^*} P$ for every computation P in λ_{ml^*};
(3) $M =_{©\eta} N$ implies $M^\circ =_{ml^*} N^\circ$, for every computations M, N in $\lambda_{©}$;
(4) $P =_{ml^*} Q$ implies $P^\bullet =_{©\eta} Q^\bullet$, for every computations P, Q in λ_{ml^*}.

Proof. (1) By induction on M in $\lambda_{©}$.
(2) By induction on P in λ_{ml^*}.
(3) We prove that $M \to_{©\eta} N$ implies $M^\circ =_{ml^*} N^\circ$, by induction on the definition of $M \to_{©\eta} N$.
(4) We prove that $P \to_{ml^*} Q$ implies $P^\bullet \to_{©\eta} Q^\bullet$, by induction on the definition of $P \to_{ml^*} Q$. □

Proposition 3.9 establishes a precise correspondence between the *equational* theories of $\lambda_{©}$ (including η-conversion) and λ_{ml^*}. We had to consider $=_{©\eta}$ since

$$xM \neq_{©} ((xM)^\circ)^\bullet \;=\; (\lambda y.x!y)(M^\circ)^\bullet \qquad \text{when } M \neq\, !V \text{ for any value } V \tag{7}$$

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

(where $=_{©}$ is the reflexive transitive and symmetric closure of $\to_{©}$) and so condition (1) in Proposition 3.9 would not hold if we replace $=_{©\eta}$ with $=_{©}$.

Remark 3.10 (Some intricacies). The correspondence between the *reduction* theories of $\lambda_{©}$ (possibly including η) and λ_{ml^*} is not immediate and demands further investigations since, according to the terminology in Sabry and Wadler (1997), there is no Galois connection: Proposition 3.9 where we replace $=_{©\eta}$ with $\to^*_{©\eta}$, and $=_{ml^*}$ with $\to^*_{ml^*}$ does not hold. More precisely:

- the condition corresponding to Point 1, namely $M \to^*_{©\eta} (M^\circ)^\bullet$, fails since $xM \not\to^*_{©\eta} ((xM)^\circ)^\bullet$ when $M \neq !V$ for any value V, see (7) above;
- the condition corresponding to Point 3, namely $M \to^*_{©\eta} N$ implies $M^\circ \to^*_{ml^*} N^\circ$, fails because $(\lambda y.N)((\lambda x.!V)!W) \to_\sigma (\lambda x.(\lambda y.N)!V)!W$ but $((\lambda y.N)((\lambda x.!V)!W))^\circ \not\to^*_{ml^*} ((\lambda x.(\lambda y.N)!V)!W)^\circ$.

Surface vs. Weak Reduction. In this paper, we study not only weak but also surface reduction, as the latter has better rewriting properties than the former. *Surface reduction in* $\lambda_{©}$ can be seen as a natural counterpart to Plotkin's *weak reduction in calculi with the* let-*constructor*, such as λ_{ml^*}. Intuitively, a term of the form $\mathsf{let}\, x{:=}M \,\mathsf{in}\, N$ can be interpreted as syntactic sugar for $(\lambda x.N)M$, however, in the expression $\mathsf{let}\, x{:=}M \,\mathsf{in}\, N$, it is not obvious that weak reduction should avoid firing redexes in N. The distinction between λ and *let* allows for a clean interpretation of weak reduction: it forbids reduction under λ, but not under *let*, which is compatible with surface reduction in $\lambda_{©}$.

Technically, when we embed λ_{ml^*} in the computational core via the translation $(\cdot)^\bullet$ in Figure 2, weak reduction $\underset{\mathsf{w}}{\to}_{ml^*}$ in λ_{ml^*} (defined as the restriction of $\to_{ml^*}$ that does not fire under λ) corresponds to surface reduction $\underset{\mathsf{s}}{\to}_{©}$ in $\lambda_{©}$: if $P \underset{\mathsf{w}}{\to}_{ml^*} P'$ then $P^\bullet \underset{\mathsf{s}}{\to}_{©} (P')^\bullet$ but not necessarily $P^\bullet \underset{\mathsf{w}}{\to}_{©} (P')^\bullet$. Indeed, consider $P = \mathsf{let}\, x{:=}R \,\mathsf{in}\, Q \underset{\mathsf{w}}{\to}_{ml^*} \mathsf{let}\, x{:=}R \,\mathsf{in}\, Q' = P'$ in λ_{ml^*}, with $Q \underset{\mathsf{w}}{\to}_{ml^*} Q'$; then $P^\bullet = (\lambda x.Q)R \underset{\mathsf{s}}{\to}_{©} (\lambda x.Q')R = (P')^\bullet$, which is not a weak step, in $\lambda_{©}$.

4. The Operational Properties of β_c

Since β-reduction is the engine of any λ-calculus, we start our analysis of the rewriting theory of $\lambda_{©}$ by studying the properties of $\to_{\beta_c}$ and its surface restriction $\underset{\mathsf{s}}{\to}_{\beta_c}$. As we show in this section, $\to_{\beta_c}$ and $\underset{\mathsf{s}}{\to}_{\beta_c}$ have already been studied in the literature: there is an exact correspondence with Simpson's calculus (Simpson 2005), which stems from Girard's linear logic (Girard 1987). Indeed, the operator ! in Simpson (2005) (modeling the *bang* operator from linear logic) behaves exactly as the the operator ! in $\lambda_{©}$ (modeling *unit* in computational calculi). It is easily seen that $(Com, \to_{\beta_c})$, that is, $\lambda_{©}$ when considering only β_c reduction, is nothing but the restriction of the bang calculus to computations. Thus, $\to_{\beta_c}$ has the same operational properties as $\to_{\beta_!}$. In particular, surface factorization and confluence for $\to_{\beta_c}$ are inherited from the corresponding properties of $\to_{\beta_!}$ in Simpson's calculus.

The Bang Calculus. We call *bang calculus* the fragment of Simpson's linear λ-calculus (Simpson 2005) without linear abstraction. It has also been studied in Ehrhard and Guerrieri (2016), Guerrieri and Manzonetto (2019), Faggian and Guerrieri (2021), Guerrieri and Olimpieri (2021) (with the name bang calculus, which we adopt), and it is closely related to Levy's Call-by-Push-Value (Levy 1999).

We briefly recall the bang calculus $(\Lambda^!, \to_{\beta_!})$. *Terms* $\Lambda^!$ are defined by

$$T, S, R ::= x \mid TS \mid \lambda x.T \mid !T \qquad (\textbf{terms}, \text{ set: } \Lambda^!)$$

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

Contexts (C) and *surface contexts* (S) are generated by the grammars:

$$\mathsf{C} ::= \langle\,\rangle \mid T\mathsf{C} \mid \mathsf{C}T \mid \lambda x.\mathsf{C} \mid !\mathsf{C} \qquad \textbf{(contexts)}$$
$$\mathsf{S} ::= \langle\,\rangle \mid T\mathsf{S} \mid \mathsf{S}T \mid \lambda x.\mathsf{S} \qquad \textbf{(surface contexts)}$$

The reduction $\to_{\beta_!}$ is the closure under context C of the rule

$$(\lambda x.R)!T \mapsto_{\beta_!} R[T/x]$$

Surface reduction $\underset{\mathsf{s}}{\to}_{\beta_!}$ is the closure of the rule $\mapsto_{\beta_!}$ under surface contexts S. *non-surface* reduction $\underset{\neg\mathsf{s}}{\to}_{\beta_!}$ is the closure of the rule $\mapsto_{\beta_!}$ under contexts C that are not surface. Surface reduction *factorizes* $\to_{\beta_!}$.

Theorem 4.1 (Surface factorization Simpson 2005)**.** *In* $\Lambda^!$*:*

(1) Surface factorization of $\to_{\beta_!}$: $\to^*_{\beta_!} \subseteq \underset{\mathsf{s}}{\to}^*_{\beta_!} \cdot \underset{\neg\mathsf{s}}{\to}^*_{\beta_!}$.
(2) Bang convergence: $T \to^*_{\beta_!} !R$ *for some term* R *if and only if* $T \underset{\mathsf{s}}{\to}^*_{\beta_!} !S$ *for some term* S.

Surface reduction is nondeterministic, but satisfies the diamond property of Fact 2.6.

Theorem 4.2 (Confluence and diamond Simpson 2005)**.** *In* $\Lambda^!$*:*

- *reduction* $\to_{\beta_!}$ *is confluent;*
- *reduction* $\underset{\mathsf{s}}{\to}_{\beta_!}$ *is quasi-diamond (and hence confluent).*

Restriction to Computations. The restriction of the bang calculus to computations, i.e., $(Com, \to_{\beta_!})$ is *exactly the same* as the fragment of $\lambda_©$ with β_c-rule as unique reduction rule, i.e. $(Com, \to_{\beta_c})$.

First, observe that the set of computations Com defined in Section 3 is a subset of the terms $\Lambda^!$, and moreover it is closed under $\to_{\beta_c}$ reduction (exactly as in Proposition 3.4). Second, observe that the restriction of contexts and surface contexts to computations, gives exactly the grammar defined in Section 3. Then $(Com, \to_{\beta_!})$ and $(Com, \to_{\beta_c})$ are in fact *the same*, and for every $M, N \in Com$:

- $M \to_{\beta_!} N$ if and only if $M \to_{\beta_c} N$.
- $M \underset{\mathsf{s}}{\to}_{\beta_!} N$ if and only if $M \underset{\mathsf{s}}{\to}_{\beta_c} N$.

Hence, $\to_{\beta_c}$ in $\lambda_©$ *inherits the operational properties of* $\to_{\beta_!}$, in particular *surface factorization* and the *quasi-diamond* property of $\underset{\mathsf{s}}{\to}_{\beta_!}$ (Theorems 4.1 and 4.2). We will use both extensively.

Fact 4.3 (Properties of β_c and of its *surface* restriction)**.** *In* $\lambda_©$*:*

- *reduction* $\to_{\beta_c}$ *is nondeterministic and confluent;*
- *reduction* $\underset{\mathsf{s}}{\to}_{\beta_c}$ *is quasi-diamond (and hence confluent);*
- *reduction* $\to_{\beta_c}$ *satisfies surface factorization:* $\to^*_{\beta_c} \subseteq \underset{\mathsf{s}}{\to}_{\beta_c}{}^* \cdot \underset{\neg\mathsf{s}}{\to}_{\beta_c}{}^*$.
- $M \to^*_{\beta_c} !V$ *for some value* V *if and only if* $T \underset{\mathsf{s}}{\to}^*_{\beta_c} !W$ *for some value* W.

In Sections 6 and 7, we shall generalize and refine the last two points, respectively, to reduction $\to_©$ instead of $\to_{\beta_c}$.

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

5. Operational Properties of $\lambda_{©}$, Weak and Surface Reduction

We study evaluation and normalization in $\lambda_{©}$ via *factorization* theorems (Section 6), which are based on both weak and surface reductions. The construction we develop in the next sections demands more work than one may expect. This is due to the fact that the rules induced by the monadic laws of associativity and identity make the analysis of the reduction properties nontrivial. In particular – as anticipated in the introduction – *weak reduction* does not factorize $\to_{©}$, and has severe drawbacks, which we explain next. *Surface reduction* behaves better, but still present difficulties. In the rest of this section, we examine their respective properties.

5.1 Weak reduction: the impact of associativity and identity

Weak (left) reduction (Section 2.2) is one of the most common and studied way to implement evaluation in CbV, and more generally in calculi with effects.

Weak β_c reduction $\underset{\mathsf{w}}{\to}_{\beta_c}$, that is, the closure of β_c under weak contexts is a *deterministic* relation. However, when including the rules induced by the monadic equation of associativity and identity, the reduction is *nondeterministic*, *nonconfluent*, and normal forms are *not unique*.

This is somehow surprising, given the prominent role of such a reduction in the literature of calculi with effects. Notice that the issues only come from σ and id, not from β_c. To resume:

(1) Reductions $\underset{\mathsf{w}}{\to}_{\mathsf{id}}$ and $\underset{\mathsf{w}}{\to}_{\beta_c\mathsf{id}}$ are nondeterministic, but are both confluent.

(2) $\underset{\mathsf{w}}{\to}_{\sigma}$, $\underset{\mathsf{w}}{\to}_{\beta_c\sigma}$, $\underset{\mathsf{w}}{\to}_{\sigma\mathsf{id}}$, and $\underset{\mathsf{w}}{\to}_{©}$ are nondeterministic, nonconfluent, and their normal forms are not unique, i.e., adding σ, weak reductions lose confluence and uniqueness of normal forms.

Example 5.1 (Non-confluence). *An example of the nondeterminism of* $\underset{\mathsf{w}}{\to}_{\mathsf{id}}$ *is the following:*

$$V((\lambda y.!y)N) \;\underset{\mathsf{w}}{\leftarrow}\; \mathsf{id}(\lambda x.!x)(V((\lambda y.!y)N)) \underset{\mathsf{w}}{\to}_{\mathsf{id}} (\lambda x.!x)(VN).$$

Because of the σ *rule, weak reductions* $\underset{\mathsf{w}}{\to}_{\sigma}$, $\underset{\mathsf{w}}{\to}_{\beta_c\sigma}$, $\underset{\mathsf{w}}{\to}_{\sigma\mathsf{id}}$ *and* $\underset{\mathsf{w}}{\to}_{©}$ *are not confluent and their normal forms are not unique (Point 2 above). Indeed, consider* $T = V((\lambda x.P)((\lambda y.Q)L))$ *where* $V = \lambda z.z!z$ *and* $P = Q = L = z!z$. *Then,*

$$M_1 = (\lambda x.VP)((\lambda y.Q)L) \;\underset{\mathsf{w}}{\leftarrow}_{\sigma}\; T \underset{\mathsf{w}}{\to}_{\sigma} V((\lambda y.(\lambda x.P)Q)L) = N_1$$

$$M_1 \underset{\mathsf{w}}{\to}_{\sigma} M_2 := (\lambda y.(\lambda x.VP)Q)L \neq (\lambda y.V((\lambda x.P)Q))L =: N_2 \;\underset{\mathsf{w}}{\leftarrow}_{\sigma}\; N_1$$

where M_2 *and* N_2 *are different* $\underset{\mathsf{w}}{\to}_{©}$*-normal forms (clearly,* $N_2 \underset{\neg\mathsf{w}}{\to}_{\sigma} M_2$*).*

Reduction $\to_{\beta_c}$ (like $\to_{\beta_v}$, Theorem 2.11.1) admits weak factorization (Fact F.2 in Appendix F)

$$\to^*_{\beta_c} \subseteq \underset{\mathsf{w}}{\to}^*_{\beta_c} \cdot \underset{\neg\mathsf{w}}{\to}^*_{\beta_c}$$

This is not the case for $\to_{©}$. The following counterexample is due to van Oostrom (2020a).

Example 5.2 (Nonfactorization van Oostrom 2020*a*). *Reduction* $\to_{©}$ *does not admit weak factorization. Consider the reduction sequence*

$$M := (\lambda y.\boldsymbol{I}!y)(z!z) \underset{\neg\mathsf{w}}{\to}_{\beta_c} (\lambda y.!y)(z!z) \underset{\mathsf{w}}{\to}_{\mathsf{id}} z!z$$

M is $\underset{\mathsf{w}}{\to}_{©}$*-normal and cannot reduce to* $z!z$ *by only performing* $\underset{\neg\mathsf{w}}{\to}_{©}$ *steps (note that* $(\lambda y.!y)(z!z)$ *is* $\underset{\neg\mathsf{w}}{\to}_{©}$*-normal), hence it is* impossible *to factorize the sequence from M to* $z!z$ *as* $M \underset{\mathsf{w}}{\to}^*_{©} \cdot \underset{\neg\mathsf{w}}{\to}^*_{©} z!z$.

Let: Different Notation, Same Issues. We stress that the issues are inherent to the associativity and identity rules, not to the specific syntax of $\lambda_{©}$. Exactly the same issues appear in Sabry and Wadler's λ_{ml^*} (Sabry and Wadler 1997) (see our Figure 1), as we show in the example below.

Example 5.3 (Evaluation context in let-notation). *In* let*-notation, the standard evaluation is* sequencing *(Filinski 1996; Jones et al. 1998; Levy et al. 2003), which exactly corresponds to weak reduction in $\lambda_{©}$. The evaluation context for sequencing is*

$$\mathsf{E_{let}} ::= \langle\,\rangle \mid \mathsf{let}\, x{:=}\mathsf{E_{let}}\, \mathsf{in}\, M.$$

*We write $\underset{\mathsf{e}}{\to}_{ml^*}$ for the closure of the λ_{ml^*} rules (in Figure 1) under contexts $\mathsf{E_{let}}$. We observe two problems, the first one due to the rule c.*let*.ass, the second one to the rule c.*let*.η.*

(1) ***Non-confluence.*** *Because of the associative rule c.*let*.ass, reduction $\underset{\mathsf{e}}{\to}_{ml^*}$ is* nondeterministic, nonconfluent, *and* normal forms are not unique. *Consider the following term*

$$T := \overline{\mathsf{let}\, z{:=}\, \underline{(\mathsf{let}\, x{:=}\, (\mathsf{let}\, y{:=}L\, \mathsf{in}\, Q)\, \mathsf{in}\, P)}\, \mathsf{in}\, R} \qquad \textit{with } R = P = Q = L = zz.$$

There are two weak redexes in T, the overlined and the underlined one. Therefore,

$$\begin{aligned} T &\underset{\mathsf{e}}{\to}_{ml^*} \mathsf{let}\, x{:=}\, (\mathsf{let}\, y{:=}L\, \mathsf{in}\, Q)\, \mathsf{in}\, (\mathsf{let}\, z{:=}P\, \mathsf{in}\, R) \\ &\underset{\mathsf{e}}{\to}_{ml^*} \mathsf{let}\, y{:=}L\, \mathsf{in}\, (\mathsf{let}\, x{:=}Q\, \mathsf{in}\, (\mathsf{let}\, z{:=}P\, \mathsf{in}\, R)) =: T' \\ T &\underset{\mathsf{e}}{\to}_{ml^*} \mathsf{let}\, z{:=}\, (\mathsf{let}\, y{:=}L\, \mathsf{in}\, (\mathsf{let}\, x{:=}Q\, \mathsf{in}\, P))\, \mathsf{in}\, R \\ &\underset{\mathsf{e}}{\to}_{ml^*} \mathsf{let}\, y{:=}L\, \mathsf{in}\, (\mathsf{let}\, z{:=}(\mathsf{let}\, x{:=}Q\, \mathsf{in}\, P)\, \mathsf{in}\, R) =: T'' \end{aligned}$$

where T' are T'' different $\underset{\mathsf{e}}{\to}_{ml^*}$-normal forms.

(2) ***Non-factorization.*** *Because of the c.*let*.η-rule,* factorization w.r.t. sequencing *does not hold. That is, a reduction sequence $M \to^*_{ml^*} N$ cannot be reorganized as weak steps followed by non-weak steps. Consider the following variation on van Oostrom's Example 5.2:*

$$M := \mathsf{let}\, y{:=}zz\, \mathsf{in}\, (\mathsf{let}\, x{:=}[y]\, \mathsf{in}\, [x]) \underset{\neg\mathsf{w}}{\to}_{c.\mathsf{let}.\eta} \mathsf{let}\, y{:=}zz\, \mathsf{in}\, [y] \underset{\mathsf{w}}{\to}_{c.\mathsf{let}.\eta} zz$$

M is $\underset{\mathsf{w}}{\to}_{ml^*}$-normal and cannot reduce to zz by only performing $\underset{\neg\mathsf{w}}{\to}_{ml^*}$ steps (note that $\mathsf{let}\, y{:=}zz\, \mathsf{in}\, [y]$ *is $\underset{\neg\mathsf{w}}{\to}_{ml^*}$-normal), so it is* impossible *to factorize the sequence form M to zz as $M \underset{\mathsf{w}}{\to}^*_{ml^*} \cdot \underset{\neg\mathsf{w}}{\to}^*_{ml^*} zz$.*

5.2 Surface reduction

In $\lambda_{©}$, surface reduction is nondeterministic, but confluent, and well-behaving.

Fact 5.4 (Nondeterminism). *For $\rho \in \{©, \beta_c, \sigma, \mathsf{id}, \beta_c\sigma, \beta\mathsf{id}, \sigma\mathsf{id}\}$, $\underset{\mathsf{s}}{\to}_{\rho}$ is nondeterministic (because in general more than one surface redex can be fired).*

We now analyze confluence of surface reduction. We will use confluence of $\underset{\mathsf{s}}{\to}_{\beta_c\sigma}$ (Point 2 below) in Section 8 (Theorem 8.13).

Proposition 5.5 (Confluence of surface reductions).

(1) *Each of the reductions $\underset{\mathsf{s}}{\to}_{\beta_c}$, $\underset{\mathsf{s}}{\to}_{\mathsf{id}}$, $\underset{\mathsf{s}}{\to}_{\sigma}$ is confluent.*
(2) *Reductions $\underset{\mathsf{s}}{\to}_{\beta_c\mathsf{id}} = (\underset{\mathsf{s}}{\to}_{\beta_c} \cup \underset{\mathsf{s}}{\to}_{\mathsf{id}})$ and $\underset{\mathsf{s}}{\to}_{\beta_c\sigma} = (\underset{\mathsf{s}}{\to}_{\beta_c} \cup \underset{\mathsf{s}}{\to}_{\sigma})$ are confluent.*

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

(3) *Reduction* $\underset{\mathsf{s}}{\to}_{©} = (\underset{\mathsf{s}}{\to}_{\beta_c} \cup \underset{\mathsf{s}}{\to}_{\sigma} \cup \underset{\mathsf{s}}{\to}_{\mathsf{id}})$ *is confluent.*

(4) *Reduction* $\underset{\mathsf{s}}{\to}_{\sigma\mathsf{id}} = (\underset{\mathsf{s}}{\to}_{\sigma} \cup \underset{\mathsf{s}}{\to}_{\mathsf{id}})$ *is* not *confluent.*

Proof. We rely on confluence of $\underset{\mathsf{s}}{\to}_{\beta_c}$ (by Theorem 4.2), and on Hindley–Rosen Lemma (Lemma 2.9). We prove commutation via strong commutation (Lemma 2.10). The only delicate point is the commutation of $\underset{\mathsf{s}}{\to}_{\sigma}$ with $\underset{\mathsf{s}}{\to}_{\mathsf{id}}$ (Points 3 and 4).

(1) $\underset{\mathsf{s}}{\to}_{\sigma}$ is locally confluent and terminating, and so confluent; $\underset{\mathsf{s}}{\to}_{\mathsf{id}}$ is quasi-diamond (in the sense of Fact 2.6), and hence also confluent.

(2) It is easily verified that $\underset{\mathsf{s}}{\to}_{\beta_c}$ and $\underset{\mathsf{s}}{\to}_{\mathsf{id}}$ strongly commute and similarly for $\underset{\mathsf{s}}{\to}_{\beta_c}$ and $\underset{\mathsf{s}}{\to}_{\sigma}$. The claim then follows by Hindley–Rosen Lemma.

(3) $\underset{\mathsf{s}}{\to}_{\beta_c} \cup \underset{\mathsf{s}}{\to}_{\mathsf{id}}$ strongly commutes with $\underset{\mathsf{s}}{\to}_{\sigma}$. This point is delicate because to close a diagram of the shape $\underset{\mathsf{s}}{\leftarrow}_{\sigma} \cdot \underset{\mathsf{s}}{\to}_{\mathsf{id}}$ may require a $\underset{\mathsf{s}}{\to}_{\beta_c}$ step, see Example 5.7 below. The claim then follows by Hindley–Rosen Lemma.

(4) A counterexample is provided by the same diagram mentioned in the previous point (Example 5.7), requiring a $\underset{\mathsf{s}}{\to}_{\beta_c}$ step to close. □

Example 5.6. *Let us give an example for nondeterminism and confluence of surface reduction.*

(1) Non-determinism: *Consider the term* $(\lambda x.R)((\lambda y.R')N)$ *where* R *and* R' *are any redexes.*

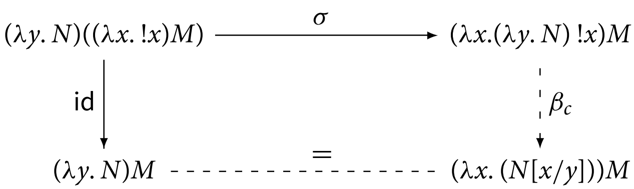
(2) Confluence: *Consider the same term as in Example 5.1:* $T = V((\lambda x.P)((\lambda y.Q)L))$*. Then,*

$$M_2 \underset{\mathsf{s}}{\leftarrow}_{\sigma} M_1 \underset{\mathsf{s}}{\leftarrow}_{\sigma} T \underset{\mathsf{s}}{\to}_{\sigma} N_1 \underset{\mathsf{s}}{\to}_{\sigma} N_2$$

Now we can close the diagram:

$$M_2 = (\lambda y.(\lambda x.VP)Q)L \underset{\mathsf{s}}{\leftarrow}_{\sigma} (\lambda y.V((\lambda x.P)Q))L = N_2.$$

Example 5.7. *In the following counterexample to confluence of* $\underset{\mathsf{s}}{\to}_{\sigma} \cup \underset{\mathsf{s}}{\to}_{\mathsf{id}}$*, where* $M = N = z!z$*, the* σ*-redex overlaps with the* id*-redex. The corresponding steps are surface (and even weak) and the only way to close the diagram is by means of a* β_c *step, which is also surface (but not weak).*



Note that $x \notin \mathsf{fv}(N)$*, and so* $\lambda x.(N[x/y]) = \lambda y.N$*, since* $\lambda x.(N[x/y])$ *is the term obtained from* $\lambda y.N$ *by renaming its bound variable* y *to* x*.*

This is also a counterexample to confluence of $(\to_{\sigma} \cup \to_{\mathsf{id}})$ *and of* $(\underset{\mathsf{w}}{\to}_{\sigma} \cup \underset{\mathsf{w}}{\to}_{\mathsf{id}})$*.*

In Section 6, we prove that surface reduction *does factorize* $\to_{©}$, similarly to what happens for Simpson's calculus (Theorem 4.1). Surface reduction also has a drawback: it does not allows us to separate $\to_{\beta_c}$ and $\to_{\sigma}$ steps. This fact makes it difficult to reason about returning a value.

Example 5.8 (An issue with surface reduction)**.** *Consider the term* $\Delta((\lambda x.!\Delta)(xx))$*, which is normal for* $\to_{\beta_c}$ *and in particular for* $\underset{\mathsf{s}}{\to}_{\beta_c}$*, but*

$$\Delta((\lambda x.!\Delta)(x!x)) \underset{\mathsf{s}}{\to}_{\sigma} (\lambda x.\Delta!\Delta)(x!x) \underset{\mathsf{s}}{\to}_{\beta_c} (\lambda x.\Delta!\Delta)(x!x)$$

Here it is not possible to postpone a step $\underset{\mathsf{s}}{\to}_{\sigma}$ *after a step* $\underset{\mathsf{s}}{\to}_{\beta_c}$*.*

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

5.3 Confluence properties of β_c, σ and id

Finally, we briefly revisit the confluence of $\lambda_{©}$, already established in de' Liguoro and Treglia (2020), in order to analyze the confluence properties of the different subsystems too. This completes the analysis given in Proposition 5.5. In Section 8, we will use confluence of $\to_{\beta_c\sigma}$ (Theorem 8.14).

Proposition 5.9 (Confluence of β_c, σ and id)**.**

(1) *Each of the reductions* $\to_{\beta_c}$, $\to_{\mathsf{id}}$, $\to_\sigma$ *is confluent.*
(2) *Reductions* $\to_{\beta_c\mathsf{id}} = (\to_{\beta_c} \cup \to_{\mathsf{id}})$ *and* $\to_{\beta_c\sigma} = (\to_{\beta_c} \cup \to_\sigma)$ *are confluent.*
(3) *Reduction* $\to_{©} = (\to_{\beta_c} \cup \to_\sigma \cup \to_{\mathsf{id}})$ *is confluent.*
(4) *Reduction* $\to_{\sigma\mathsf{id}} = (\to_\sigma \cup \to_{\mathsf{id}})$ *is* not *confluent.*

Proof. We rely on confluence of $\to_{\beta_c}$ (by Theorem 4.2), and on Hindley–Rosen Lemma (Lemma 2.9). We prove commutation via strong commutation (Lemma 2.10). The only delicate point is again the commutation of $\to_\sigma$ with $\to_{\mathsf{id}}$ (Points 3 and 4).

(1) $\to_\sigma$ is locally confluent and terminating, and so confluent. $\to_{\mathsf{id}}$ is quasi-diamond in the sense of Fact 2.6, and hence confluent.
(2) It is easily verified that $\to_{\beta_c}$ and $\to_{\mathsf{id}}$ strongly commute, and $\to_{\beta_c}$ and $\to_\sigma$ do as well. The claim then follows by Hindley–Rosen Lemma.
(3) $\to_{\beta_c} \cup \to_{\mathsf{id}}$ strongly commutes with $\to_\sigma$. This point is delicate because to close a diagram of the shape $\leftarrow_\sigma \cdot \to_{\mathsf{id}}$ may require a $\to_{\beta_c}$ step (see Example 5.7). The claim then follows by Hindley–Rosen Lemma.
(4) A counterexample is provided by the same diagram mentioned in the previous point (Example 5.7), requiring a $\to_{\beta_c}$ step to close. □

6. Surface and Weak Factorization

In this section, we prove several factorization results for $\lambda_{©}$. Surface factorization is the cornerstone for the subsequent development. It is proved in Section 6.1.

Theorem 6.1. *[Surface factorization in $\lambda_{©}$] Reduction $\to_{©}$ admits surface factorization:*

$$M \to_{©}^{*} N \text{ implies } M \underset{\mathsf{s}}{\to}_{©}^{*} \cdot \underset{\neg\mathsf{s}}{\to}_{©}^{*} N.$$

We then refine this result first by *postponing* id steps that are not also β_c steps, and then by means of *weak factorization* (on surface steps). This further phases serve two purposes:

(1) to postpone non-weak $\beta_c\sigma$ steps after weak $\beta_c\sigma$ steps, and
(2) to separate weak β_c and σ steps, by postponing $\underset{\mathsf{w}}{\to}_\sigma$ steps after $\underset{\mathsf{w}}{\to}_{\beta_c}$ steps, and
(3) to perform a fine analysis of quantitative properties, namely the number of β_c steps.

We will need Points 1 and 2 to define *evaluation* relations (Section 7) and Point 3 to define *normalizing strategies* (Section 8).

Technical Lemmas. We shall often exploit some basic properties of contextual closure, which we collect here. In any variant of the λ-calculus, if a step $T \to_\rho T'$ is obtained by the closure of a rule $\mapsto_\rho$ under a *non-empty context* (i.e., a context other than the hole), then T and T' have the *same shape*, that is, they are both applications or both abstractions or both variables or both !-terms.

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

Fact 6.2 (Shape preservation). *Let $\mapsto_\rho$ be a rule and $\to_\rho$ be its contextual closure. Assume $T = \mathsf{C}\langle R\rangle \to_\rho \mathsf{C}\langle R'\rangle = T'$ where $R \mapsto_\rho R'$ and $\mathsf{C} \neq \langle\,\rangle$. Then T and T' have the same shape.*

An easy-to-verify consequence of Fact 6.2 in $\lambda_{\copyright}$ is the following.

Lemma 6.3 (Redexes preservation). *Let $M \underset{\neg\mathsf{s}}{\to}_{\copyright} N$ and $\gamma \in \{\beta_c, \sigma, \mathsf{id}\}$: M is a γ-redex if and only if N is a γ-redex.*

Proof. See the Appendix, namely Corollary B.2 for $\gamma \in \{\sigma, \beta_c\}$, and Lemma C.1 for $\gamma = \mathsf{id}$. □

Lemma 6.3 is false if we replace the hypothesis $M \underset{\neg\mathsf{s}}{\to}_{\copyright} N$ with $M \underset{\neg\mathsf{w}}{\to}_{\copyright} N$. Indeed, consider $M = (\lambda x.(\lambda y.!y)!x)L \underset{\neg\mathsf{w}}{\to}_{\copyright} (\lambda x.!x)L = N$: N is a id-redex but M is not.

Notice the following inclusions, which we will use freely.

Fact 6.4. $\underset{\mathsf{w}}{\to}_{\copyright} \subsetneq \underset{\mathsf{s}}{\to}_{\copyright}$ *and* $\underset{\neg\mathsf{s}}{\to}_{\copyright} \subsetneq \underset{\neg\mathsf{w}}{\to}_{\copyright}$, *because a weak context is necessarily a surface context (but a surface context need not be a weak context, e.g. the surface context $\mathsf{S} = (\lambda x.\langle\,\rangle)M$ is not weak).*

6.1 Surface factorizations in $\lambda_{\copyright}$, modularly

We prove surface factorization in $\lambda_{\copyright}$. We already know that surface factorization holds for $\to_{\beta_c}$ (Fact 4.3), so we can rely on it, and work modularly, following the approach proposed in Accattoli et al. (2021). The tests for call-by-name head factorization and call-by-value weak factorization in Accattoli et al. (2021) easily adapt to surface factorization in $\lambda_{\copyright}$, yielding the following convenient test. It modularly establishes surface factorization of a reduction $\to_{\beta_c} \cup \to_\gamma$, where $\to_\gamma$ is a new reduction added to $\to_{\beta_c}$. Details of the proof are in Appendices B.2 and B.3.

Proposition 6.5 (A modular test for surface factorization with β_c). *Let $\to_\gamma$ be the contextual closure of a rule $\mapsto_\gamma$. Reduction $\to_{\beta_c} \cup \to_\gamma$ satisfies surface factorization (that is, $(\to_{\beta_c} \cup \to_\gamma)^* \subseteq (\underset{\mathsf{s}}{\to}_{\beta_c} \cup \underset{\mathsf{s}}{\to}_\gamma)^* \cdot (\underset{\neg\mathsf{s}}{\to}_{\beta_c} \cup \underset{\neg\mathsf{s}}{\to}_\gamma)^*$) if:*

(1) Surface factorization of $\to_\gamma$*:* $\to_\gamma^* \subseteq \underset{\mathsf{s}}{\to}_\gamma^* \cdot \underset{\neg\mathsf{s}}{\to}_\gamma^*$.
(2) $\mapsto_\gamma$ *is* substitutive*:* $R \mapsto_\gamma R'$ *implies* $R[M/x] \mapsto_\gamma R'[M/x]$.
(3) Root linear swap*:* $\underset{\neg\mathsf{s}}{\to}_{\beta_c} \cdot \mapsto_\gamma \subseteq \mapsto_\gamma \cdot \to_{\beta_c}^*$.

We will use the following easy property (an instance of Lemma B.4 in the Appendix).

Lemma 6.6. *Let $\to_\xi, \to_\gamma$ be the contextual closure of rules $\mapsto_\xi, \mapsto_\gamma$. Then, $\underset{\neg\mathsf{s}}{\to}_\xi \cdot \mapsto_\gamma \subseteq \underset{\mathsf{s}}{\to}_\gamma \cdot \to_\xi^=$ implies $\underset{\neg\mathsf{s}}{\to}_\xi \cdot \underset{\mathsf{s}}{\to}_\gamma \subseteq \underset{\mathsf{s}}{\to}_\gamma \cdot \to_\xi^=$.*

Root Lemmas. Lemmas 6.7 and 6.8 below provide *everything we need* to verify the conditions of the modular test in Proposition 6.5, and so to establish surface factorization in $\lambda_{\copyright}$.

Lemma 6.7 (σ-Roots). *Let $\gamma \in \{\sigma, \mathsf{id}, \beta_c\}$. The following holds:*

$$M \underset{\neg\mathsf{s}}{\to}_\gamma L \mapsto_\sigma N \text{ implies } M \mapsto_\sigma \cdot \to_\gamma N.$$

Proof. We have $L = (\lambda x.L_1)((\lambda y.L_2)L_3) \mapsto_\sigma (\lambda y.(\lambda x.L_1)L_2)L_3 = N$. Since L is a σ-redex, M also is a σ-redex (Lemma 6.3). So $M = (\lambda x.M_1)((\lambda y.M_2)M_3) \underset{\neg\mathsf{s}}{\to}_\gamma (\lambda x.L_1)((\lambda y.L_2)L_3) = L$, where for only one $i \in \{1, 2, 3\}$ $M_i \underset{\neg\mathsf{s}}{\to}_\gamma L_i$ and otherwise $M_j = L_j$, for $j \neq i$ (by Fact B.1 in the Appendix). Therefore, $M = (\lambda x.M_1)((\lambda y.M_2)M_3) \mapsto_\sigma (\lambda y.(\lambda x.M_1)M_2)M_3 \underset{\neg\mathsf{s}}{\to}_\gamma (\lambda y.(\lambda x.L_1)L_2)L_3 = N$. □

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

Lemma 6.8 (id-Roots). *Let $\gamma \in \{\sigma, \mathsf{id}, \beta_c\}$. The following holds:*

$$M \underset{\neg s}{\rightarrow}_{\gamma} L \mapsto_{\mathsf{id}} N \text{ implies } M \mapsto_{\mathsf{id}} \cdot \rightarrow_{\gamma} N.$$

Proof. We have $L = \mathbf{I}N \mapsto_{\mathsf{id}} N$. Since L is an id-redex, M also is (Lemma 6.3). So, $M = \mathbf{I}P \underset{\neg s}{\rightarrow}_{\gamma} \mathbf{I}N$ for some $P \underset{\neg s}{\rightarrow}_{\gamma} N$. Therefore, $M = \mathbf{I}P \mapsto_{\mathsf{id}} P \underset{\neg s}{\rightarrow}_{\gamma} N$. □

Let us make explicit the content of the two lemmas above. By instantiating $\gamma \in \{\sigma, \mathsf{id}, \beta_c\}$ in Lemmas 6.7 and 6.8, and combining them with Lemma 6.6, we obtain the following facts:

Fact 6.9. *(1) $M \underset{\neg s}{\rightarrow}_{\sigma} \cdot \mapsto_{\sigma} N$ implies $M \mapsto_{\sigma} \cdot \rightarrow_{\sigma}^{=} N$ and so (Lemma 6.6) $M \underset{\neg s}{\rightarrow}_{\sigma} \cdot \underset{s}{\rightarrow}_{\sigma} N$ implies $M \underset{s}{\rightarrow}_{\sigma} \cdot \rightarrow_{\sigma}^{=} N$ (i.e. strong postponement holds).*

(2) $M \underset{\neg s}{\rightarrow}_{\mathsf{id}} \cdot \mapsto_{\sigma} N$ implies $M \mapsto_{\sigma} \cdot \rightarrow_{\mathsf{id}}^{=} N$ and so (Lemma 6.6) $M \underset{\neg s}{\rightarrow}_{\mathsf{id}} \cdot \underset{s}{\rightarrow}_{\sigma} N$ implies $M \underset{s}{\rightarrow}_{\sigma} \cdot \rightarrow_{\mathsf{id}}^{=} N$.

(3) $M \underset{\neg s}{\rightarrow}_{\beta_c} \cdot \mapsto_{\sigma} N$ implies $M \mapsto_{\sigma} \cdot \rightarrow_{\beta_c}^{=} N$.

Fact 6.10. *(1) $M \underset{\neg s}{\rightarrow}_{\mathsf{id}} \cdot \mapsto_{\mathsf{id}} N$ implies $M \mapsto_{\mathsf{id}} \cdot \rightarrow_{\mathsf{id}}^{=} N$, and so (Lemma 6.6) $M \underset{\neg s}{\rightarrow}_{\mathsf{id}} \cdot \underset{s}{\rightarrow}_{\mathsf{id}} N$ implies $M \underset{s}{\rightarrow}_{\mathsf{id}} \cdot \rightarrow_{\mathsf{id}}^{=} N$ (i.e. strong postponement holds).*

(2) $M \underset{\neg s}{\rightarrow}_{\sigma} \cdot \mapsto_{\mathsf{id}} N$ implies $M \mapsto_{\mathsf{id}} \cdot \rightarrow_{\sigma}^{=} N$, and so (Lemma 6.6) $M \underset{\neg s}{\rightarrow}_{\sigma} \cdot \underset{s}{\rightarrow}_{\mathsf{id}} N$ implies $M \underset{s}{\rightarrow}_{\mathsf{id}} \cdot \rightarrow_{\sigma}^{=} N$.

(3) $M \underset{\neg s}{\rightarrow}_{\beta_c} \cdot \mapsto_{\mathsf{id}} N$ implies $M \mapsto_{\mathsf{id}} \cdot \rightarrow_{\beta_c}^{=} N$.

Surface Factorization of $\rightarrow_{\mathsf{id}} \cup \rightarrow_{\sigma}$. We can now combine the facts above concerning σ and id steps, using the modular approach proposed in Accattoli et al. (2021) (see Theorem B.3 in the Appendix), to prove surface factorization of $\rightarrow_{\mathsf{id}\sigma} = \rightarrow_{\mathsf{id}} \cup \rightarrow_{\sigma}$.

Lemma 6.11 (Surface factorization of idσ). Surface factorization *of $\rightarrow_{\mathsf{id}} \cup \rightarrow_{\sigma}$ holds, because:*

(1) Surface factorization of $\rightarrow_{\sigma}$ holds (that is, $\rightarrow_{\sigma}^{*} \subseteq \underset{s}{\rightarrow}_{\sigma}^{*} \cdot \underset{\neg s}{\rightarrow}_{\sigma}^{*}$).

(2) Surface factorization of $\rightarrow_{\mathsf{id}}$ holds (that is, $\rightarrow_{\mathsf{id}}^{*} \subseteq \underset{s}{\rightarrow}_{\mathsf{id}}^{*} \cdot \underset{\neg s}{\rightarrow}_{\mathsf{id}}^{*}$).

(3) Linear swap: $\underset{\neg s}{\rightarrow}_{\mathsf{id}} \cdot \underset{s}{\rightarrow}_{\sigma} \subseteq \underset{s}{\rightarrow}_{\sigma} \cdot \rightarrow_{\mathsf{id}}^{*}$.

(4) Linear swap: $\underset{\neg s}{\rightarrow}_{\sigma} \cdot \underset{s}{\rightarrow}_{\mathsf{id}} \subseteq \underset{s}{\rightarrow}_{\mathsf{id}} \cdot \rightarrow_{\sigma}^{*}$.

Proof. Points 1 and 2 follow from Fact 6.9.1 and Fact 6.10.1, respectively, by (linear) strong postponement (Lemma 2.5). Point 3 is Fact 6.9.2. Point 4 is Fact 6.10.2. □

Surface Factorization of $\lambda_{©}$, Modularly. We are now ready to use the modular test for surface factorization with β_c (Proposition 6.5) to prove Theorem 6.1.

Theorem 6.1. *[Surface factorization in $\lambda_{©}$] Reduction $\rightarrow_{©}$ admits surface factorization:*

$$M \rightarrow_{©}^{*} N \text{ implies } M \underset{s}{\rightarrow}_{©}^{*} \cdot \underset{\neg s}{\rightarrow}_{©}^{*} N.$$

Proof. All conditions in Proposition 6.5 hold, namely

(1) *Surface factorization* of $\rightarrow_{\mathsf{id}} \cup \rightarrow_{\sigma}$ holds by Lemma 6.11.
(2) *Substitutivity*: $\mapsto_{\mathsf{id}}$ and $\mapsto_{\sigma}$ are substitutive (the proof is immediate).
(3) *Root linear swap*: for $\xi \in \{\mathsf{id}, \sigma\}$, $\underset{\neg s}{\rightarrow}_{\beta_c} \cdot \mapsto_{\xi} \subseteq \mapsto_{\xi} \cdot \rightarrow_{\beta_c}^{=}$ by Fact 6.9.3 and Fact 6.10.3. □

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

Interestingly, the same machinery can also be used to prove another surface factorization result, which says that surface factorization, when applied to $\rightarrow^*_{\beta_c\sigma}$ only, does not create $\rightarrow_{\mathsf{id}}$ steps.

Proposition 6.12 (Surface factorization of $\beta_c\sigma$). *Reduction* $\rightarrow_{\beta_c\sigma}$ *admits surface factorization:*

$$M \rightarrow^*_{\beta_c\sigma} N \text{ implies } M \underset{\mathsf{s}}{\rightarrow}^*_{\beta_c\sigma} \cdot \underset{\neg\mathsf{s}}{\rightarrow}^*_{\beta_c\sigma} N.$$

Proof. All conditions in Proposition 6.5 hold, namely

(1) *Surface factorization* of $\rightarrow_\sigma$ holds by Fact 6.9.1 and strong postponement (Lemma 2.5).
(2) *Substitutivity*: $\mapsto_\sigma$ is substitutive (the proof is immediate).
(3) *Root linear swap*: $\underset{\neg\mathsf{s}}{\rightarrow}_{\beta_c} \cdot \mapsto_\sigma \subseteq \mapsto_\sigma \cdot \rightarrow^=_{\beta_c}$ by Fact 6.9.3. □

The fact that two similar factorization results (Theorem 6.1 and Proposition 6.12) can be proven by means of the *same* modular test (Proposition 6.5) fed on *similar* lemmas shows one of the benefits of our modular approach: passing from one result to the other is smooth and effortless.

6.2 A closer look at id steps, via postponement

We show a postponement result for id steps on which evaluation (Section 7) and normalization (Section 8) rely. Note that $\rightarrow_{\mathsf{id}}$ overlaps with $\rightarrow_{\beta_c}$. We define $\rightarrow_\iota$ as a $\rightarrow_{\mathsf{id}}$ step that is not $\rightarrow_{\beta_c}$.

$$\mapsto_\iota \coloneqq \mapsto_{\mathsf{id}} \setminus \mapsto_{\beta_c} \qquad (\iota\text{-rule})$$

Clearly, $\rightarrow_{©} = \rightarrow_{\beta_c} \cup \rightarrow_\sigma \cup \rightarrow_\iota$. In the proofs, it is convenient to split $\rightarrow_{\beta_c}$ into steps that are also $\rightarrow_{\mathsf{id}}$ steps, and those that are not.

$$\mapsto_{\beta 1} \coloneqq \mapsto_{\mathsf{id}} \cap \mapsto_{\beta_c} \qquad (\beta 1\text{-rule})$$
$$\mapsto_{\beta 2} \coloneqq \mapsto_{\beta_c} \setminus \mapsto_{\mathsf{id}} \qquad (\beta 2\text{-rule})$$

Notice that $\rightarrow_{\beta_c} = \rightarrow_{\beta 1} \cup \rightarrow_{\beta 2}$.

We prove that $\rightarrow_\iota$ steps can be postponed after both $\rightarrow_{\beta_c}$ and $\rightarrow_\sigma$ steps, by using van Oostrom's decreasing diagrams technique (van Oostrom 1994, 2008) (Theorem 2.8). The proof closely follows van Oostrom's proof of the postponement of η after β (van Oostrom 2020*b*).

Theorem 6.13 (Postponement of ι). *If* $M \rightarrow_{©}{}^* N$, *then* $M \rightarrow_{\beta_c\sigma}{}^* \cdot \rightarrow_\iota{}^* N$.

Proof. Let $\triangleleft = \rightarrow_\iota$ and let $\blacktriangleright = \rightarrow_{\beta 1} \cup \rightarrow_{\beta 2} \cup \rightarrow_\sigma$. We equip the labels $\{\beta 1, \beta 2, \sigma, \iota\}$ with the following (well-founded) order

$$\beta 2 < \iota \qquad \iota < \beta 1 \qquad \iota < \sigma$$

We prove that the pair of relations $\triangleright, \blacktriangleright$ is decreasing by checking the following three local commutations hold (the technical details are in the Appendix):

(1) $\rightarrow_\iota \cdot \rightarrow_{\beta 1} \subseteq \rightarrow_{\beta 1}{}^* \cdot \rightarrow^=_\iota$ (see Lemma C.3);
(2) $\rightarrow_\iota \cdot \rightarrow_{\beta 2} \subseteq \rightarrow_{\beta 1}{}^* \cdot \rightarrow_{\beta 2}{}^= \cdot \rightarrow_{\beta 1}{}^* \cdot \rightarrow^*_\iota$ (see Lemma C.4);
(3) $\rightarrow_\iota \cdot \rightarrow_\sigma \subseteq (\rightarrow_\sigma \cup \rightarrow_{\beta 1})^* \cdot \rightarrow^=_\iota$ (see Lemma C.5).

Hence, by Theorem 2.8, the relations $\triangleright$ and $\blacktriangleright$ commute. That is, $\rightarrow_\iota$ postpones after $\rightarrow_{\beta_c} \cup \rightarrow_\sigma$:

$$\rightarrow^*_\iota \cdot \rightarrow_{\beta_c\sigma}{}^* \subseteq \rightarrow_{\beta_c\sigma}{}^* \cdot \rightarrow^*_\iota$$

Or equivalently (Lemma 2.4), $\rightarrow^*_{©} \subseteq \rightarrow_{\beta_c\sigma}{}^* \cdot \rightarrow^*_\iota$. □

Corollary 6.14 (Surface factorization + ι postponement). *If* $M \rightarrow^*_{©} N$, *then* $M \underset{\mathsf{s}}{\rightarrow}^*_{\beta_c\sigma} \cdot \underset{\neg\mathsf{s}}{\rightarrow}^*_{\beta_c\sigma} \cdot \rightarrow_\iota{}^* N$.

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

Proof. Immediate consequence of ι-postponement (Theorem 6.13) and of surface factorization of the resulting initial $\to_{\beta_c\sigma}$-sequence (Proposition 6.12). □

6.3 Weak factorization

Thanks to surface factorization plus ι-postponement (Corollary 6.14), every $\to_{©}$-sequence can be rearranged so that it starts with an $\underset{\mathsf{s}}{\to}_{\beta_c\sigma}$-sequence. We show now that such an initial $\underset{\mathsf{s}}{\to}_{\beta_c\sigma}$-sequence can in turn be factorized into weak steps followed by non-weak steps. This *weak factorization* result will be used in Section 7 to obtain evaluation via weak β_c steps (Theorem 7.6).

Remarkably, weak factorization of an $\underset{\mathsf{s}}{\to}_{\beta_c\sigma}$-sequence *preserves* the number of β_c steps. This property has no role with respect to evaluation, but it will be crucial when we investigate normalizing strategies in Section 8. For this reason, we include it in the statement of Theorem 6.16.

Quantitative Linear Postponement. Let us take an abstract point of view. The condition in Lemma 2.5 – Hindley's strong postponement – can be refined into quantitative (linear) variants, which allow us to "count the steps" and are useful to establish termination properties.

Lemma 6.15 (Linear postponement). *Let $(A, \to)$ be an ARS with $\to\, = \,\underset{\mathsf{e}}{\to} \cup \underset{\mathsf{i}}{\to}$.*

- *If $\underset{\mathsf{i}}{\to} \cdot \underset{\mathsf{e}}{\to} \subseteq \underset{\mathsf{e}}{\to} \cdot \underset{\mathsf{i}}{\to}^{=}$, then $M \to^* N$ implies $M \underset{\mathsf{e}}{\to}^* \cdot \underset{\mathsf{i}}{\to}^* N$ and the two sequences have the same number of $\underset{\mathsf{e}}{\to}$ steps.*
- *For all $l \in L$ with L a set of indices, let $\to_l\, = \,\underset{\mathsf{e}}{\to}_l \cup \underset{\mathsf{i}}{\to}_l$ and $\underset{\mathsf{e}}{\to} = \bigcup_l \underset{\mathsf{e}}{\to}_l$ and $\underset{\mathsf{i}}{\to} = \bigcup_l \underset{\mathsf{i}}{\to}_l$. Assume*

$$\underset{\mathsf{i}}{\to}_j \cdot \underset{\mathsf{e}}{\to}_k \subseteq \underset{\mathsf{e}}{\to}_k \cdot \to_j \quad \textit{for all } j, k \in L \tag{8}$$

  Then, $M \to^* N$ implies $M \underset{\mathsf{e}}{\to}^* \cdot \underset{\mathsf{i}}{\to}^* N$ and the two sequences have the same number of $\to_l$ steps, for each $l \in L$.

Observe that in (8), the last step is $\to_j$, not necessarily $\underset{\mathsf{i}}{\to}_j$.

Weak Factorization. We show two kinds of *weak factorization*: a surface $\beta_c\sigma$ sequence can be reorganized, so that non-weak $\beta_c\sigma$ steps are postponed after the weak ones; and a weak $\beta_c\sigma$ sequence can in turn be rearranged so that weak β_c steps are before weak σ steps.

Theorem 6.16 (Weak factorization).

(1) If $M \underset{\mathsf{s}}{\to}^*_{\beta_c\sigma} N$ then $M \underset{\mathsf{w}}{\to}^*_{\beta_c\sigma} \cdot \underset{\neg\mathsf{w}}{\to}^*_{\beta_c\sigma} N$ where all steps are surface, and the two sequences have the same number of β_c steps.
(2) If $M \underset{\mathsf{w}}{\to}^*_{\beta_c\sigma} N$ then $M \underset{\mathsf{w}}{\to}^*_{\beta_c} \cdot \underset{\mathsf{w}}{\to}^*_{\sigma} N$, and the two sequences have the same number of β_c steps.

Proof. In both claims, we use Lemma 6.15. Its linearity allows us to count the β_c steps. In the proof, we write $\underset{\mathsf{w}}{\to}$ (resp. $\underset{\mathsf{s}}{\to}$) for $\underset{\mathsf{w}}{\to}_{\beta_c\sigma}$ (resp. $\underset{\mathsf{s}}{\to}_{\beta_c\sigma}$).

(1) Let $\underset{\mathsf{i}}{\to} \, = \, \underset{\mathsf{s}}{\to} \setminus \underset{\mathsf{w}}{\to}$ (i.e., $\underset{\mathsf{i}}{\to}$ is a surface step whose redex is in the scope of a λ). We prove linear postponement:

$$\underset{\mathsf{i}}{\to} \cdot \underset{\mathsf{w}}{\to} \subseteq \underset{\mathsf{w}}{\to} \cdot \underset{\mathsf{s}}{\to} \tag{9}$$

Assume $M \underset{\mathsf{i}}{\to} L \underset{\mathsf{w}}{\to} N$: M and L have the same shape, which is not $!U$, otherwise no weak or surface step from M is possible. We examine the cases.

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

– The step $L \underset{w}{\rightarrow} N$ has empty context:
  – $L \mapsto_{\beta_c} N$. Then, $L = (\lambda x.P')!V \mapsto_{\beta_c} P'[V/x] = N$, and $M = (\lambda x.P)!V \underset{i}{\rightarrow} (\lambda x.P')!V$ with $P \underset{s}{\rightarrow} P'$. Therefore, $(\lambda x.P)!V \mapsto_{\beta_c} P[V/x] \underset{s}{\rightarrow} P'[V/x] = N$.
  – $L \mapsto_{\sigma} N$. Then, $L = V((\lambda x.P)Q) \mapsto_{\sigma} (\lambda x.VP)Q = N$, and $M = V_0((\lambda x.P_0)Q_0) \underset{i}{\rightarrow} V((\lambda x.P)Q)$ where exactly one among V_0, P_0, Q_0 reduces to V, P, Q, respectively, the other two are unchanged. So, $M = V_0((\lambda x.P_0)Q_0) \mapsto_{\sigma} (\lambda x.V_0P_0)Q_0 \underset{i}{\rightarrow} (\lambda x.VP)Q = N$.

– The step $L \underset{w}{\rightarrow} N$ has nonempty context. Necessarily, we have $L = VQ \underset{w}{\rightarrow} VQ'$, with $Q \underset{w}{\rightarrow} Q'$:
  – Case $M = M_V Q \underset{i}{\rightarrow} VQ \underset{w}{\rightarrow} VQ' = N$ with $M_V \underset{i}{\rightarrow} V$ and $Q \underset{w}{\rightarrow} Q'$, then $M_V Q \underset{w}{\rightarrow} M_V Q' \underset{i}{\rightarrow} VQ'$.
  – Case $M = VM_Q \underset{i}{\rightarrow} VQ \underset{w}{\rightarrow} VQ' = N$ with $M_Q \underset{i}{\rightarrow} Q \underset{w}{\rightarrow} Q$. We conclude by *i.h.*.

Observe that we have proved more than (9), namely we proved

$$\underset{i}{\rightarrow}_j \cdot \underset{w}{\rightarrow}_k \subseteq \underset{w}{\rightarrow}_k \cdot \underset{s}{\rightarrow}_j \quad \text{(for all } j, k \in \{\beta_c, \sigma\})$$

So, we conclude that the two sequences have the same number of β_c steps, by Lemma 6.15.

(2) We prove $\underset{w}{\rightarrow}_\sigma \cdot \underset{w}{\rightarrow}_{\beta_c} \subseteq \underset{w}{\rightarrow}_{\beta_c} \cdot \underset{w}{\rightarrow}_\sigma^{=}$ similarly to Point 1 and conclude by Lemma 6.15. □

Combining Points 1 and 2 in Theorem 6.16, we deduce that

$$M \underset{s}{\rightarrow}^*_{\beta_c\sigma} !V \text{ implies } M \underset{w}{\rightarrow}^*_{\beta_c} \cdot \underset{w}{\rightarrow}^*_{\sigma} \cdot \underset{\neg w}{\rightarrow}^*_{\beta_c\sigma} !V$$

and the two sequences from M to $!V$ have the same number of β_c steps.

7. Returning a Value

In this section, we focus on *values*. They are the terms of interest in the CbV λ-calculus. Also, for weak reduction there, closed values are exactly the normal forms of closed terms, i.e., of *programs*.

In a computational setting such as $\lambda_©$, we are interested in knowing if a term M *returns* a value, i.e. if $M \rightarrow^*_© !V$ for some value V, noted $M \Downarrow$ (the computation $!V$ is sometimes called a *returned value*, in that it is the coercion of a value V to the computational level). Since a term may be reduced in several ways and so its reduction graph can become quite complicated, it is natural to search for deterministic reductions to return a value. Hence, the question is if M returns a value, is there a *deterministic* reduction (called *evaluation*) from M that is guaranteed to return a value? The answer is positive. In fact, there are two such reductions: $\underset{w}{\rightarrow}_{\beta_c}$ and $\mapsto_{\beta_c\sigma}$ (Theorem 7.4 below).

Recall that $\mapsto_{\beta_c\sigma} = (\mapsto_{\beta_c} \cup \mapsto_\sigma)$ is the union of two rules without any contextual closure. Thanks to their simple reduction graph, deterministic reductions are quite useful in particular for proving negative results such as showing that a computation cannot return a value.

Fact 7.1. *In $\lambda_©$, reductions $\underset{w}{\rightarrow}_{\beta_c}$, $\mapsto_{\beta_c}$, $\mapsto_\sigma$, and $\mapsto_{\beta_c\sigma} = \mapsto_{\beta_c} \cup \mapsto_\sigma$ are deterministic.*

In $\lambda_©$ one of the reasons for the interest in values is that, akin to the CbV λ-calculus, *closed* (i.e., without free variables) returned values are exactly the closed normal forms for weak reductions $\underset{w}{\rightarrow}_©$ and $\underset{w}{\rightarrow}_{\beta_c}$. This is a consequence of the following syntactic characterizations of normal forms.

Proposition 7.2. *A computation is normal for reduction $\underset{w}{\rightarrow}_©$ (resp. $\underset{s}{\rightarrow}_©$; $\rightarrow_©$) if and only if it is of the form $\mathsf{N_W}$ (resp. $\mathsf{N_S}$; N) defined below, where $\hat{M}$ denotes a computation $M \neq !x$ for any $x \in Var$.*

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

$$\mathsf{N_W} ::= !V \mid \mathsf{A_W} \mid (\lambda x.\hat{M})\mathsf{A_W} \qquad \mathsf{A_W} ::= x\mathsf{N_W}$$
$$\mathsf{N_S} ::= !V \mid \mathsf{A_S} \mid (\lambda x.\hat{\mathsf{N}}_\mathsf{S})\mathsf{A_S} \qquad \mathsf{A_S} ::= x\mathsf{N_S}$$
$$\mathsf{N} ::= !x \mid !\lambda.\mathsf{N} \mid \mathsf{A} \mid (\lambda x.\hat{\mathsf{N}})\mathsf{A} \qquad \mathsf{A} ::= x\mathsf{N}$$

Proof. The right-to-left part is proved by induction on $\mathsf{N_W}$ (resp. $\mathsf{N_S}$; N). The left-to-right part follows easily from the observation that every computation can be written in a unique way as $V_1(\ldots(V_n!V_0)\ldots)$ for some $n \geq 0$ and some values $V_0, \ldots, V_n$. □

Corollary 7.3 (Closed normal forms)**.** *Let* $\to \in \{\underset{\mathsf{w}}{\to}_{\beta_c}, \underset{\mathsf{w}}{\to}_{\circledcirc}, \underset{\mathsf{s}}{\to}_{\beta_c}, \underset{\mathsf{s}}{\to}_{\circledcirc}\}$. *A closed computation is* $\to$*-normal if and only if it is a returned value.*

Proof. Computations of shape $\mathsf{A_W}$ and $\mathsf{A_S}$ have a free variable. So, according to Proposition 7.2, closed returned values are all and only the closed normal forms for $\underset{\mathsf{w}}{\to}_{\circledcirc}$ and $\underset{\mathsf{s}}{\to}_{\circledcirc}$.

Moreover, since every closed computation M can be written in a unique way as $V_1(\ldots(V_n!V_0)\ldots)$ for some $n \geq 0$ and some closed values $V_0, \ldots, V_n$, if M is $\underset{\mathsf{w}}{\to}_{\beta_c}$-normal or $\underset{\mathsf{s}}{\to}_{\beta_c}$-normal then $n = 0$ (otherwise $V_n!V_0$ would be a β_c-redex), hence M is a returned value. □

Corollary 7.3 means that reductions $\underset{\mathsf{w}}{\to}_{\beta_c}, \underset{\mathsf{w}}{\to}_{\circledcirc}, \underset{\mathsf{s}}{\to}_{\beta_c}, \underset{\mathsf{s}}{\to}_{\circledcirc}$ behave differently only on *open* computations (that is, with at least one free variable).

We can now state the main result in this section. Sections 7.1 and 7.2 are devoted to prove it.

Theorem 7.4 (Returning a value)**.** *The following are equivalent:*

(1) M returns a value, i.e. $M \to_{\circledcirc}^{*} !V$.
(2) The maximal $\underset{\mathsf{w}}{\to}_{\beta_c}$*-sequence from M is* finite *and ends in a returned value* $!W$.
(3) The maximal $\mapsto_{\beta_c\sigma}$*-sequence from M is* finite *and ends in a returned value* $!W$.

Proof. (1) $\implies$ (2) is Theorem 7.6 below, which we prove in forthcoming Section 7.1.
(2) $\implies$ (3) is Proposition 7.10 below, which we prove in forthcoming Section 7.2.
(3) $\implies$ (1) is trivial. □

Note that Theorem 7.4 (and hence the analysis that will follow) is not restricted to closed terms. Indeed, an open term may well return a value. For example, $!(\lambda x.!z)$ or $!x$ or $(\lambda x.!x)!z$.

7.1 Values via weak β_c steps

Thanks to factorization, we can prove that $\underset{\mathsf{w}}{\to}_{\beta_c}$ steps suffice to return a value. This is an immediate consequence of surface factorization plus ι postponement (Corollary 6.14), and weak factorization (Theorem 6.16), and the fact that non-weak steps, ι steps, and σ steps cannot produce !-terms.

Lemma 7.5. *If* $M \to_{\circledcirc} !V$ *with a step that is* not $M \underset{\mathsf{w}}{\to}_{\beta_c} !V$, *then* $M = !W$ *for some value* W.

Proof. Indeed, one can easily check the following (recall that $\to_\iota = \to_{\mathsf{id}} \setminus \to_{\beta_c}$).

- If $M \to_\sigma !V$, then $M = !W$ for some value W (proof by induction on M).
- If $M \to_\iota !V$, then $M = !W$ for some value W (proof by induction on M).
- If $M \underset{\neg\mathsf{w}}{\to}_{\beta_c} !V$, then $M = !W$ for some value W (by shape preservation, Fact 6.2). □

Theorem 7.6 (Values via weak β_c steps)**.** *The following are equivalent:*

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

(1) $M \to_{\copyright}^{*} !V$ *for some* $V \in Val$*;*
(2) $M \underset{\mathsf{w}}{\to}_{\beta_c}^{*} !W$ *for some* $W \in Val$*.*

Proof. Point 2 trivially implies Point 1, as $\underset{\mathsf{w}}{\to}_{\beta_c} \subseteq \to_{\copyright}$. Let us show that Point 1 entails Point 2.

If $M \to_{\copyright}^{*} !V$ then $M \underset{\mathsf{s}}{\to}_{\beta_c\sigma}^{*} \cdot \underset{\neg\mathsf{s}}{\to}_{\beta_c\sigma}^{*} \cdot \to_{\iota}^{*} !V$ by surface factorization plus ι postponement (Corollary 6.14). By weak factorization (Theorem 6.16.1-2), we have

$$M \underset{\mathsf{w}}{\to}_{\beta_c}^{*} M' \underset{\mathsf{w}}{\to}_{\sigma}^{*} \cdot \underset{\neg\mathsf{w}}{\to}_{\beta_c\sigma}^{*} \cdot \underset{\neg\mathsf{s}}{\to}_{\beta_c\sigma}^{*} \cdot \to_{\iota}^{*} !V.$$

By iterating Lemma 7.5 from $!V$ backward (and since $\underset{\neg\mathsf{s}}{\to}_{\rho} \subseteq \underset{\neg\mathsf{w}}{\to}_{\rho}$), we have that all terms in the sequence from M' to $!V$ are !-terms. So in particular, M' has shape $!W$ for some value W. □

Remark 7.7. Theorem 7.6 was already claimed in de' Liguoro and Treglia (2020), for closed terms. However, the inductive argument there (which does not use any factorization) is fallacious, it does not suffice to produce a complete proof in the case where $M \underset{\neg\mathsf{w}}{\to}_{\copyright} \cdot \underset{\mathsf{w}}{\to}_{\copyright} !V$.

7.2 Values via $\beta_c\sigma$ root steps

We also show an alternative way to evaluate a term in $\lambda_{\copyright}$. Let us call *root steps* the rules $\mapsto_{\beta_c}$, $\mapsto_{\sigma}$ and $\mapsto_{\mathsf{id}}$. The first two suffice to return a value, without the need for any contextual closure.

Note that this property holds only because terms are restricted to computations (for example, in Plotkin's CbV λ-calculus, $(II)(II)$ can be reduced, but it is not itself a redex, so $(II)(II) \not\mapsto_{\beta_v}$).

Looking closer at the proof of Corollary 7.3, we observe that any closed (i.e. without free variables) computation has the following property: it *is* either a returned value (when $n = 0$), or a β_c-redex (when $n = 1$) or a σ-redex (when $n > 1$). More generally, the same holds for any (possibly open) computation that returns a value (Corollary 7.9 below).

Lemma 7.8. *Assume* $M \underset{\mathsf{w}}{\to}_{\beta_c}^{*} !W$ *for some value* W*. Then,*

- *either* $M = !W$*,*
- *or* $M = (\lambda x.P)M'$ *and* $M' \underset{\mathsf{w}}{\to}_{\beta_c}^{*} !U$*, for some value* U*.*

Thus, $M = V_1(\ldots (V_n!U) \ldots)$*, where* $n \geq 0$ *and the* V_i*'s are abstractions, and if* $n > 0$ *then* $M = V_1 \ldots (V_{n-1}(\lambda x_n.P_n)!U) \ldots) \underset{\mathsf{w}}{\to}_{\beta_c} V_1(\ldots (V_{n-1}P_n[U/x_n]) \ldots)$*.*

Corollary 7.9 (Progression via root steps)**.** *If* M *returns a value (i.e.* $M \to_{\copyright}^{*} !W$ *for some value* W*), then* M *is either a* β_c*-redex, or a* σ*-redex, or it has shape* $!V$ *for some value* V*.*

Proof. By Theorem 7.6, $M \underset{\mathsf{w}}{\to}_{\beta_c}{}^{*} !W'$ for some value W'. By Lemma 7.8, we conclude. □

Corollary 7.9 states a *progression* results: a $\mapsto_{\beta_c\sigma}$-sequence from M may only end in a !-term. We still need to verify that such a sequence terminates.

Proposition 7.10 (Weak steps and root steps)**.** *If* $M \underset{\mathsf{w}}{\to}_{\beta_c}^{*} !W$ *then* $M \mapsto_{\beta_c\sigma}^{*} !W$*. Moreover, the two sequences have the same number of* β_c *steps.*

Proof. By induction on the number k of $\underset{\mathsf{w}}{\to}_{\beta_c}$ steps. If $k = 0$ the claim holds trivially. Otherwise, $M \underset{\mathsf{w}}{\to}_{\beta_c} M_1 \underset{\mathsf{w}}{\to}_{\beta_c}^{*} !W$ and by *i.h.*

$$M_1 \mapsto_{\beta_c\sigma}^{*} !W. \tag{10}$$

- If M is β_c-redex, then $M \mapsto_{\beta_c} M_1$ by determinism of $\underset{\mathsf{w}}{\rightarrow}_{\beta_c}$ (Fact 7.1), and the claim is proved.
- If M is a σ-redex, observe that by Lemma 7.8,
  - $M = (\lambda x_0.P_0)(\dots(\lambda x_{n-1}.P_{n-1})((\lambda x_n.P_n)!U)\dots)$, and
  - $M_1 = (\lambda x_0.P_0)(\dots(\lambda x_{n-1}.P_{n-1})(P_n[U/x_n])\dots)$.

  We apply all possible $\mapsto_\sigma$ steps starting from M, obtaining

$$M \mapsto_\sigma^* (\lambda x_{n-1}.(\dots(\lambda x_0.P_0)\dots)P_{n-1})((\lambda x_n.P_n)!U)$$
$$\mapsto_\sigma (\lambda x_n.(\lambda x_{n-1}.(\dots(\lambda x_0.P_0)\dots)P_{n-1})P_n)!U = M'$$

  which is a β_c-redex, so $M' \mapsto_{\beta_c} (\lambda x_{n-1}.(\dots(\lambda x_0.P_0)\dots)P_{n-1})(P_n[U/x_n]) =: N$ (note that we used the hypothesis on free variables of $\mapsto_\sigma$). We observe that $M_1 \mapsto_\sigma^* N$. We conclude, by using (10) and the fact that $\mapsto_{\beta_c\sigma}$ is deterministic (Fact 7.1). □

The converse of Proposition 7.10 is also true and immediate. We can finally prove that *root* steps $\mapsto_{\beta_c}$ and $\mapsto_\sigma$ suffice to return a value, without the need for any contextual closure.

Theorem 7.11 (Values via root $\beta_c\sigma$ steps)**.** *The following are equivalent:*

(1) $M \rightarrow_{\copyright}^* !V$ for some $V \in Val$;
(2) $M \mapsto_{\beta_c\sigma}^* !W$ for some $W \in Val$.

Proof. Trivially (2) $\implies$ (1). Conversely, (1) $\implies$ (2) by Proposition 7.10 and Theorem 7.6. □

7.3 Observational equivalence

We now adapt the notion of observational equivalence, introduced in Plotkin (1975) for the CbV λ-calculus, to $\lambda_{\copyright}$. Informally, two terms are observationally equivalent if they can be substituted for each other in all contexts without observing any difference in their behavior. For a computation M in $\lambda_{\copyright}$, the "behavior" of interest is *returning a value*: $M \rightarrow_{\copyright}^* !V$ for some value V, also noted $M \Downarrow$.

Definition 7.12 (Observational equivalence)**.** Let $M, N \in Com$. We say that M and N are *observationally equivalent*, noted $M \cong N$, if for every context C, $\mathsf{C}\langle M\rangle \Downarrow$ if and only if $\mathsf{C}\langle N\rangle \Downarrow$.

A consequence of Theorem 7.6 is that the behavior of interest in Definition 7.12 can be equivalently defined as $\mathsf{C}\langle M\rangle \underset{\mathsf{w}}{\rightarrow}_{\beta_c}{}^* !V$ for some value V, instead of $\mathsf{C}\langle M\rangle \Downarrow$: the resulting notion of observational equivalence would be exactly the same. The definition using $\underset{\mathsf{w}}{\rightarrow}_{\beta_c}$ instead of $\rightarrow_{\copyright}$ is more in the spirit of Plotkin's original one for the CbV λ-calculus (Plotkin 1975). Reduction $\underset{\mathsf{w}}{\rightarrow}_{\beta_c}$ is deterministic, and for closed terms it terminates if and only if it ends in a returned value (Corollary 7.3). Hence, for *closed* terms, returning a value amounts to say that their evaluation $\underset{\mathsf{w}}{\rightarrow}_{\beta_c}$ *halts*.

The advantage of our Definition 7.12 is that it allows us to prove an important property of observational equivalence – the fact that it contains the equational theory of $\lambda_{\copyright}$ (Corollary 7.15)—in a very easy way, thanks to the following obvious lemma and adequacy (Theorem 7.14).

Lemma 7.13 (Value persistence)**.** *For every value V, if $!V \rightarrow_{\copyright} M$ then $M = !W$ for some value W.*

Proof. In $\lambda_{\copyright}$, no redex has shape $!V$ for any value V, hence the step $!V \rightarrow_{\copyright} M$ is obtained via a non-empty contextual closure. By shape preservation (Fact 6.2), $M = !W$ for some value W. □

An easy argument, similar to that in Crary (2009) (which in turn simplifies the one in 1975) gives:

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

Theorem 7.14 (Adequacy). *If $M \to_{\circledcirc}^{*} N$ then $M \Downarrow$ if and only if $N \Downarrow$.*

Proof. Suppose $N \Downarrow$, that is, $N \to_{\circledcirc}^{*} !V$ for some value $!V$. Therefore, $M \to_{\circledcirc}^{*} N \to_{\circledcirc}^{*} !V$ and so $M \Downarrow$.

Conversely, suppose $M \Downarrow$, that is, $M \to_{\circledcirc}^{*} !V$ for some value V. By confluence of $\to_{\circledcirc}$ (Proposition 3.5), since $M \to_{\circledcirc}^{*} N$, there is $L \in Com$ such that $N \to_{\circledcirc}^{*} L$ and $!V \to_{\circledcirc}^{*} L$. Since $!V$ is a returned value, so is L by value persistence (Lemma 7.13). Therefore, $N \Downarrow$. □

Corollary 7.15 (Observational equivalence contains equational theory). *If $M =_{\circledcirc} N$ then $M \cong N$.*

Proof. As $M =_{\circledcirc} N$, there are $L_0, \ldots, L_n \in Com$ $(n \geq 0)$ such that $M = L_0 \leftrightarrow_{\circledcirc} L_1 \leftrightarrow_{\circledcirc} \cdots \leftrightarrow_{\circledcirc} L_n = N$, where $\leftrightarrow_{\circledcirc} := \to_{\circledcirc} \cup \leftarrow_{\circledcirc}$. Hence, for every context C, $\mathsf{C}\langle L_0\rangle \leftrightarrow_{\circledcirc} \mathsf{C}\langle L_1\rangle \leftrightarrow_{\circledcirc} \cdots \leftrightarrow_{\circledcirc} \mathsf{C}\langle L_n\rangle$. By adequacy (Theorem 7.14), $\mathsf{C}\langle L_i\rangle \Downarrow$ if and only if $\mathsf{C}\langle L_{i+1}\rangle \Downarrow$ for all $1 \leq i < n$. Thus, $M \cong N$. □

The converse of Corollary 7.15 fails. Indeed, $!\lambda x.!x \cong !\lambda x.!\lambda y.x!y$ but $!\lambda x.!x \neq_{\circledcirc} !\lambda x.!\lambda y.x!y$.

8. Normalization and Normalizing Strategies

In this section, we study normalization and normalizing strategies in $\lambda_{\circledcirc}$.

Reduction $\to_{\circledcirc}$ is obtained by adding $\to_{\iota}$ and $\to_{\sigma}$ to $\to_{\beta_c}$. What is the role of ι steps and σ steps with respect to normalization in $\lambda_{\circledcirc}$? Perhaps surprisingly, despite the fact that both $\to_{\iota}$ and $\to_{\sigma}$ are strongly normalizing (Proposition 8.6 below), their role is quite different.

(1) Unlike the case of terms returning a value we studied in Section 7, β_c steps do not suffice to capture $\circledcirc$-normalization, in that σ steps may turn a β_c-normalizing term into one that is not $\circledcirc$-normalizing. That is, σ steps are *essential* to normalization in $\lambda_{\circledcirc}$ (see Section 8.2).
(2) ι steps instead are *irrelevant* for normalization in $\lambda_{\circledcirc}$, in the sense that they play no role. Indeed, a term has a $\circledcirc$-normal form if and only if it has a $\beta_c\sigma$-normal form (see Section 8.1).

Taking into account both Points 1 and 2, in Section 8.3 we define two families of normalizing strategies in $\lambda_{\circledcirc}$. The first one, quite constrained, relies on an *iteration of weak reduction* $\underset{\mathsf{w}}{\to}_{\circledcirc}$. The second one, more liberal, is based on an *iteration of surface reduction* $\underset{\mathsf{s}}{\to}_{\circledcirc}$. The interest of a rather liberal strategy is that it provides *a more versatile framework* to reason about program transformations, or optimization techniques such as parallel implementation.

Technical Lemmas: Preservation of Normal Forms. We collect here some properties of preservation of (full, weak and surface) normal forms, which we will use along the section. The easy proofs are in Appendix D.

Lemma 8.1 *Assume $M \to_{\iota} N$.*

(1) M is β_c-normal if and only if N is β_c-normal.
(2) If M is σ-normal, so is N.

Lemma 8.2. *If $M \to_{\sigma} N$, then: M is $\underset{\mathsf{w}}{\to}_{\beta_c}$-normal if and only if so is N.*

Lemma 8.2 fails if we replace $\underset{\mathsf{w}}{\to}_{\beta_c}$ with $\underset{\mathsf{s}}{\to}_{\beta_c}$. Indeed, $M \to_{\sigma} N$ for some M $\underset{\mathsf{s}}{\to}_{\beta_c}$-normal does not imply that N is $\underset{\mathsf{s}}{\to}_{\beta_c}$-normal, as we will see in Example 8.8.

Lemma 8.3. *Let $\mathsf{e} \in \{\mathsf{w}, \mathsf{s}\}$. If $M \underset{\neg\mathsf{e}}{\to}_{\beta_c\sigma} N$ then: M is $\underset{\mathsf{e}}{\to}_{\beta_c\sigma}$-normal if and only if N is $\underset{\mathsf{e}}{\to}_{\beta_c\sigma}$-normal.*

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

8.1 Irrelevance of ι steps for normalization

We show that postponement of ι steps (Theorem 6.13) implies that $\to_\iota$ steps have no impact on normalization, i.e., whether a term M has or not a ©-normal form. Indeed, saying that M has a ©-normal form is equivalent to say that M has a $\beta_c\sigma$-normal form.

On the one hand, if $M \to^*_{\beta_c\sigma} N$ and N is $\beta_c\sigma$-normal, to reach a ©-normal form it suffices to extend the reduction with ι steps to a ι-normal form (since $\to_\iota$ is terminating, Proposition 8.6). Notice that here we use Lemma 8.1. On the other hand, the proof that ©-normalization implies $\beta_c\sigma$-normalization is trickier, because σ-normal forms are not preserved by performing a ι step backward (the converse of Lemma 8.1.2 is false). Here is a counterexample.

Example 8.4. *Consider $(\lambda x.x!x)(I(z!z)) \to_\iota (\lambda x.x!x)(z!z)$, where $(\lambda x.x!x)(z!z)$ is σ-normal (actually ©-normal) but $(\lambda x.x!x)(I(z!z))$ is not σ-normal.*

Consequently, the fact that M has a ©-normal form N means (by postponement of $\to_\iota$) that $M \to^*_{\beta_c\sigma} P \to^*_\iota N$ for some P that Lemma 8.1 guarantees to be β_c-normal only, not σ-normal. To prove that M has a $\beta_c\sigma$-normal form is not even enough to take the σ-normal form of P, because a σ step can create a β_c-redex. To solve the problem, we need the following technical lemma.

Lemma 8.5. *Assume $M \to^k_\iota N$, where $k > 0$, and N is $\sigma\iota$-normal. If M is not σ-normal, then there exist M' and N' such that either $M \to_\sigma M' \to_\iota N' \to^{k-1}_\iota N$ or $M \to_\sigma M' \to_{\beta_c} N' \to^{k-1}_\iota N$.*

We also use that $\to_\sigma$ and $\to_\iota$ are strongly normalizing (Proposition 8.6). Instead of proving that $\to_\sigma$ and $\to_\iota$ are – separately – so, we state a more general result (its proof is in Appendix D).

Proposition 8.6 (Termination of $\sigma\mathsf{id}$). *Reduction $\to_{\sigma\mathsf{id}} = (\to_\sigma \cup \to_{\mathsf{id}})$ is strongly normalizing.*

Now we have all the elements to prove the following.

Theorem 8.7 (Irrelevance of ι for normalization). *The following are equivalent:*

(1) M is ©-normalizing;
(2) M is $\beta_c\sigma$-normalizing.

Proof.

(1)⇒(2): If M is ©-normalizing, then $M \to^*_{©} N$ for some ©-normal N. By postponement of ι steps (Theorem 6.13), for some P we have

$$M \to^*_{\beta_c\sigma} P \to^*_\iota N \tag{11}$$

By Lemma 8.1.1, P is β_c-normal in (11).
For any sequence of the form (11), let $w(P) = (w_\iota(P), w_\sigma(P))$, where $w_\iota(P)$ and $w_\sigma(P)$ are the lengths of the maximal ι-sequence and of the maximal σ-sequence from P, respectively; they are well-defined because $\to_\iota$ and $\to_\sigma$ are strongly normalizing (Proposition 8.6).
We proceed by induction on $w(P)$ ordered lexicographically to prove that $M \to^*_{\beta_c\sigma} P' \to^*_\iota N$ for some P' $\beta_c\sigma$-normal (and so M is $\beta_c\sigma$-normalizing).
- If $w(P) = (0, h)$ then $P = N$, so P is σ-normal and hence $\beta_c\sigma$-normal.
- If $w(P) = (k, 0)$, then P is σ-normal and hence $\beta_c\sigma$-normal.
- Otherwise $w(P) = (k, h)$ with $k, h > 0$. By Lemma 8.5, $M \to^*_{\beta_c\sigma} P' \to^*_\iota N$ for some P' with $w(P') < w(P)$: indeed, $w(P') = (k, h-1)$ or $w(P') = (k-1, h)$. By *i.h.*, we can conclude.

(2)⇒(1): As M is $\beta_c\sigma$-normalizing, $M \to^*_{\beta_c\sigma} N$ for some $\beta_c\sigma$-normal N. As $\to_\iota$ is strongly normalizing (Proposition 8.6), $N \to^*_\iota P$ for some P ι-normal. By Lemma 8.1.1-2, P

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

is also β_c-normal and σ-normal. Summing up, $M \to_{©}^{*} P$ with P ©-normal, i.e., M is ©-normalizing. □

8.2 The essential role of σ steps for normalization

In $\lambda_{©}$, for normalization, σ steps play a crucial role, unlike ι steps. Indeed, σ steps can unveil "hidden" β_c-redexes in a term. Let us see this with an example, where we consider a term that is β_c-normal but diverging in $\lambda_{©}$ and this divergence is "unblocked" by a σ step.

Example 8.8. *[Normalization in $\lambda_{©}$] Let $\Delta = \lambda x.x!x$. Consider the σ step*

$$M_z = \Delta((\lambda y.!\Delta)(z!z)) \to_{\sigma} (\lambda y.\Delta!\Delta)(z!z) = N_z$$

M_z is β_c-normal, but not ©-normal. In fact, M_z is diverging in $\lambda_{©}$ (that is, it is not ©-normalizing):

$$M_z \to_{\sigma} N_z \to_{\beta_c} N_z \to_{\beta_c} \dots$$

Note that the σ step is weak and that N_z is normal for $\underset{\mathrm{w}}{\to}_{\beta_c}$ but not for $\underset{\mathrm{s}}{\to}_{\beta_c}$.

The fact that a σ step can unblock a hidden β_c-redex is not limited to open terms. Indeed, $!\lambda z.M_z$ is closed and β_c-normal, but divergent in $\lambda_{©}$:

$$!\lambda z.M_z \to_{\sigma} !\lambda z.N_z \to_{\beta_c} !\lambda z.N_z \to_{\beta_c} \dots$$

Example 8.8 shows that, contrary to ι steps, σ steps are essential to determine whether a term has or not a normal form in $\lambda_{©}$. This fact is in accordance with the semantics. First, it can be shown that the term M_z above and $\Delta!\Delta$ are observational equivalent. Second, the denotational models and type systems studied in Ehrhard (2012), de' Liguoro and Treglia (2020) (which are compatible with $\lambda_{©}$) interpret M_z in the same way as $\Delta!\Delta$, which is a β_c-divergent term. It is then reasonable to expect that the two terms have the same operational behavior in $\lambda_{©}$. Adding σ steps to β_c-reduction is a way to obtain this: both M_z and $\Delta!\Delta$ are divergent in $\lambda_{©}$. Said differently, σ-reduction restricts the set of ©-normal forms, so as to exclude some β_c-normal (but not $\beta_c\sigma$-normal) forms that are semantically meaningless.

Actually, σ-reduction *can only restrict* the set of terms having a normal form: it may turn a β_c-normal form into a term that diverges in $\lambda_{©}$, but it cannot turn a β_c-diverging term into a $\lambda_{©}$-normalizing one. To prove this (Proposition 8.10), we rely on the following lemma.

Lemma 8.9. *If M is not β_c-normal and $M \to_{\sigma} L$, then L is not β_c-normal and $L \to_{\beta_c} N$ implies $M \to_{\beta_c} \cdot \to_{\sigma}^{=} N$.*

Roughly, Lemma 8.9 says that a σ step on a term that is not β_c-normal cannot erase a β_c-redex, and hence it can be postponed. Lemma 8.9 does not contradict Example 8.8: the former talks about a σ step on a term that is not β_c-normal, whereas the start terms in Example 8.8 are β_c-normal.

Proposition 8.10. *If a term is $\beta_c\sigma$-normalizing (resp. strongly $\beta_c\sigma$-normalizing), then it is β_c-normalizing (resp. strongly β_c-normalizing).*

Proof. As $\to_{\beta_c} \subseteq \to_{\beta_c\sigma}$, any infinite β_c-sequence is an infinite $\beta_c\sigma$-sequence. So, if M is not strongly β_c-normalizing, it is not strongly $\beta_c\sigma$-normalizing.

We prove now the part of the statement about normalization. If M is $\beta_c\sigma$-normalizing, there exists a reduction sequence $\mathfrak{s} : M \to_{\beta_c\sigma}^{*} N$ with N $\beta_c\sigma$-normal. Let $|\mathfrak{s}|_{\sigma}$ be the number of steps in $\mathfrak{s}$, and let $|\mathfrak{s}|_{\beta_c}$ be the number of β_c steps after the last σ step in $\mathfrak{s}$ (when $|\mathfrak{s}|_{\sigma} = 0$, $|\mathfrak{s}|_{\beta_c}$ is just the length of $\mathfrak{s}$). We prove by induction on $(|\mathfrak{s}|_{\sigma}, |\mathfrak{s}|_{\beta_c})$ ordered lexicographically that M is β_c-normalizing. There are three cases.

(1) If $\mathfrak{s}$ contains only β_c steps ($|\mathfrak{s}|_{\sigma} = 0$), then $M \to_{\beta_c}^{*} N$ and we are done.

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

(2) If $\mathfrak{s} : M \to^*_{\beta_c\sigma} L \to^+_\sigma N$ ($\mathfrak{s}$ ends with a nonempty sequence of σ steps), then L is β_c-normal by Lemma 8.9, as N is β_c-normal; by *i.h.* applied to the sequence $\mathfrak{s}' : M \to^*_{\beta_c\sigma} L$ (as $|\mathfrak{s}'|_\sigma < |\mathfrak{s}|_\sigma$), M is β_c-normalizing.

(3) Otherwise $\mathfrak{s} : M \to^*_{\beta_c\sigma} L \to_\sigma P \to_{\beta_c} Q \to^*_{\beta_c} N$ ($L \to_\sigma P$ is the last σ step in $\mathfrak{s}$, followed by a β_c step). By Lemma 8.9, either there is a sequence $\mathfrak{s}' : M \to^*_{\beta_c\sigma} L \to_{\beta_c} R \to_\sigma Q \to^*_{\beta_c} N$, then $|\mathfrak{s}'|_\sigma = |\mathfrak{s}|_\sigma$ and $|\mathfrak{s}'|_{\beta_c} < |\mathfrak{s}|_{\beta_c}$; or $s' : M \to^*_{\beta_c\sigma} L \to_{\beta_c} Q \to^*_{\beta_c} N$ and then $|\mathfrak{s}'|_\sigma < |\mathfrak{s}|_\sigma$. In both cases $(|\mathfrak{s}'|_\sigma, |\mathfrak{s}'|_{\beta_c}) < (|\mathfrak{s}|_\sigma, |\mathfrak{s}|_{\beta_c})$, so by *i.h.* M is β_c-normalizing. □

8.3 Normalizing strategies

Irrelevance of ι steps (Theorem 8.7) implies that to define a normalizing strategy for $\lambda_©$, it suffices to define a normalizing strategy for $\beta_c\sigma$. We do so by iterating either *surface* or *weak* reduction. Our definition of $\beta_c\sigma$-normalizing strategy and the proof of normalization (Theorem 8.14) is *parametric* on either.

The difficulty here is that both weak and surface reduction are *nondeterministic*. The key property we need in the proof is that the reduction we iterate is *uniformly normalizing* (see Definition 2.1). We first establish that this holds for weak and surface reduction. While uniform normalization is easy to prove for the former, it is *nontrivial* for the latter, its proof is rather sophisticated. Here we reap the fruits of the careful analysis of the number of β_c steps in Section 6.3. Finally, we formalize the strategies and tackle normalization.

Notation. *Since we are now only concerned with $\beta_c\sigma$ steps, for the sake of readability in the rest of the section, we often write $\to$, $\underset{s}{\to}$ and $\underset{w}{\to}$ for $\to_{\beta_c\sigma}$, $\underset{s}{\to}_{\beta_c\sigma}$ and $\underset{w}{\to}_{\beta_c\sigma}$, respectively.*

Understanding Uniform Normalization. The fact that $\underset{s}{\to}$ and $\underset{w}{\to}$ are uniformly normalizing is key in the definition of normalizing strategy and deserves some discussion.

The heart of the normalization proof is that if M has a $\to$-normal form N, we can perform surface steps and reach a *surface normal form*. Note that surface factorization only guarantees that *there exists* a $\underset{s}{\to}$-sequence such that if $M \to^* N$ then $M \underset{s}{\to}^* L \underset{\neg s}{\to}^* N$, where L is $\underset{s}{\to}$-normal. The *existential* quantification is crucial here because $\underset{s}{\to}$ is *not* a deterministic reduction. *Uniform normalization* of $\underset{s}{\to}$ transforms the existential into a *universal* quantification: if M has a $\to$-normal form (and so a fortiori a surface normal form), then *every* sequence of $\underset{s}{\to}$ steps will terminate. The normalizing strategy then iterates this process, performing surface reduction on the subterms of a surface normal form, until we obtain a $\to$-normal form.

Uniform Normalization of Weak and Surface Reduction

We prove that both weak and surface reduction are uniformly normalizing, i.e., for $\mathsf{e} \in \{\mathsf{w}, \mathsf{s}\}$, if a term M is $\underset{e}{\to}$-normalizing, then it is strongly $\underset{e}{\to}$-normalizing. In both cases, the proof relies on the fact that all maximal $\underset{e}{\to}$-sequences from a given term M have the same number of β_c steps.

Fact 8.11 (Number of β_c steps)**.** *Given a $\to_{\beta_c\sigma}$-sequence $\mathfrak{s}$, the number of its β_c steps is finite if and only if $\mathfrak{s}$ is finite.*

Proof. The right-to-left implication is obvious. The left-to-right is an immediate consequence of the fact that $\to_\sigma$ is strongly normalizing (Proposition 8.6). □

A *maximal* $\underset{e}{\to}$-sequence from M is either infinite or ends in a e-normal form. Theorem 8.13 states that for $\mathsf{e} \in \{\mathsf{w}, \mathsf{s}\}$, all *maximal* $\underset{e}{\to}$-sequences from the same term M have the *same behavior*,

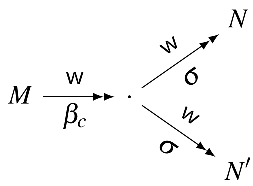


Figure 3. Weak reduction.

also quantitatively (with respect to the number of β_c steps). The proof relies on the following lemma. Recall that weak reduction is not confluent (Example 5.1); however, $\underset{w}{\rightarrow}_{\beta_c}$ is deterministic.

Lemma 8.12 (Invariant). *Given $M \in Com$,* every sequence $M \underset{w}{\rightarrow}^*_{\beta_c\sigma} N$ *where N is $\underset{w}{\rightarrow}_{\beta_c}$-normal has* the same number *k of β_c steps. Moreover,*

(1) the unique maximal $\underset{w}{\rightarrow}_{\beta_c}$-sequence from M has length k, and

(2) there exists a sequence $M \underset{w}{\rightarrow}_{\beta_c}{}^k L \underset{w}{\rightarrow}^*_{\sigma} N$ for some $L \in Com$.

Proof. The argument is illustrated in Figure 3. Let k be the number of β_c steps in a sequence $\mathfrak{s}\colon M \underset{w}{\rightarrow}^*_{\beta_c\sigma} N$ where N is $\underset{w}{\rightarrow}_{\beta_c}$-normal. By weak factorization (Theorem 6.16.2) there is a sequence $M \underset{w}{\rightarrow}_{\beta_c}{}^k L \underset{w}{\rightarrow}^*_{\sigma} N$ with the same number k of β_c steps. As N is $\underset{w}{\rightarrow}_{\beta_c}$-normal, so is L (Lemma 8.2). Thus, $M \underset{w}{\rightarrow}_{\beta_c}{}^k L$ is a maximal $\underset{w}{\rightarrow}_{\beta_c}$-sequence from M, and it is unique because $\underset{w}{\rightarrow}_{\beta_c}$ is deterministic. □

Theorem 8.13 (Uniform normalization).

(1) Reduction $\underset{w}{\rightarrow}_{\beta_c\sigma}$ is uniformly normalizing.

(2) Reduction $\underset{s}{\rightarrow}_{\beta_c\sigma}$ is uniformly normalizing.

Moreover, all maximal $\underset{w}{\rightarrow}_{\beta_c\sigma}$-sequences (resp. all maximal $\underset{s}{\rightarrow}_{\beta_c\sigma}$-sequences) from the same term M have the same number of β_c steps.

Proof. We write $\rightarrow$ (resp. $\underset{w}{\rightarrow}$, $\underset{s}{\rightarrow}$) for $\rightarrow_{\beta_c\sigma}$ (resp. $\underset{w}{\rightarrow}_{\beta_c\sigma}$, $\underset{s}{\rightarrow}_{\beta_c\sigma}$).

Claim 1. Let $M \underset{w}{\rightarrow}^* N$ where N is $\underset{w}{\rightarrow}$-normal, and so, in particular $\underset{w}{\rightarrow}_{\beta_c}$-normal. By Lemma 8.12, $M \underset{w}{\rightarrow}_{\beta_c}{}^k L \underset{w}{\rightarrow}^*_{\sigma} N$ where $M \underset{w}{\rightarrow}_{\beta_c}{}^k L$ is the (unique) maximal $\underset{w}{\rightarrow}_{\beta_c}$-sequence from M. We prove that no $\underset{w}{\rightarrow}$-sequence from M may have more than k β_c steps. Indeed, every sequence $\mathfrak{s}\colon M \underset{w}{\rightarrow}^* N'$ can be factorized (Theorem 6.16.2) as $M \underset{w}{\rightarrow}^*_{\beta_c} L' \underset{w}{\rightarrow}^*_{\sigma} N'$ with the same number of β_c steps as $\mathfrak{s}$, and $M \underset{w}{\rightarrow}^*_{\beta_c} L'$ is a prefix of the maximal $\underset{w}{\rightarrow}_{\beta_c}$-sequence $M \underset{w}{\rightarrow}_{\beta_c}{}^k L$ from M (since $\underset{w}{\rightarrow}_{\beta_c}$ is deterministic).

We deduce that no infinite $\underset{w}{\rightarrow}$-sequence from M is possible (by Fact 8.11).

Claim 2. Assume that $M \underset{s}{\rightarrow}^* N$ with N $\underset{s}{\rightarrow}$-normal. Recall that $\underset{s}{\rightarrow}$ is confluent (Proposition 5.5.2), so N is the unique $\underset{s}{\rightarrow}$-normal form of M.

First, by induction on N, we prove that given a term M,

$$(\#) \text{ all sequences } M \underset{s}{\rightarrow}^* N \text{ have the same number of } \beta_c \text{ steps.}$$

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

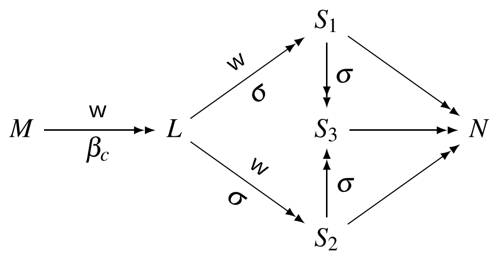


Figure 4. Surface reduction.

Let $\mathfrak{s}_1, \mathfrak{s}_2$ be two such sequences. Figure 4 illustrates the argument. By weak factorization (Theorem 6.16.1), there is a sequence $M \underset{w}{\to}^* S_1 \underset{\neg w}{\to}^* N$ (resp. $M \underset{w}{\to}^* S_2 \underset{\neg w}{\to}^* N$) with the same number of β_c steps as $\mathfrak{s}_1$ (resp. $\mathfrak{s}_2$), and whose steps are all surface. Note that S_1 and S_2 are $\underset{w}{\to}$-normal (by Lemma 8.3, because N is in particular $\underset{w}{\to}$-normal), and so in particular $\underset{w}{\to}_{\beta_c}$-normal. By Lemma 8.12, $M \underset{w}{\to}^* S_1$, $M \underset{w}{\to}^* S_2$ have the same number k of β_c steps, and so do the sequences $\mathfrak{s}'_1 : M \underset{w}{\to}_{\beta_c}{}^k L \underset{w}{\to}_{\sigma}^* S_1$ and $\mathfrak{s}'_2 : M \underset{w}{\to}_{\beta_c}{}^k L \underset{w}{\to}_{\sigma}^* S_2$.

To prove (#), we show that the sequences $\mathfrak{s}''_1 \colon S_1 \underset{\neg w}{\to}^* N$ and $\mathfrak{s}''_2 \colon S_2 \underset{\neg w}{\to}^* N$ have the same number of β_c steps.

By confluence of $\underset{s}{\to}_{\sigma}$ (Proposition 5.5.1), $S_1 \underset{s}{\to}_{\sigma}^* S_3 \underset{s}{\leftarrow}_{\sigma}^* S_2$, for some S_3, and (by confluence of $\underset{s}{\to}$, Proposition 5.5.2) there is a sequence $\mathfrak{t} \colon S_3 \underset{s}{\to}^* N$. By Lemma 8.2, since S_1, S_2 are $\underset{w}{\to}$-normal, terms in these sequences are $\underset{w}{\to}$-normal, and so all steps are not only surface, but also $\underset{\neg w}{\to}$ steps. That is, $S_1 \underset{\neg w}{\to}_{\sigma}^* S_3$, $S_2 \underset{\neg w}{\to}_{\sigma}^* S_3$ and $\mathfrak{t} \colon S_3 \underset{\neg w}{\to}^* N$. Hence, S_1, S_2, S_3, N have the same shape by Fact 6.2.

We examine the shape of N, and prove claim (#) by showing that $\mathfrak{s}''_1$ and $\mathfrak{s}''_2$ have the same number of β_c steps as $\mathfrak{t}$ (note that the sequences $S_1 \underset{\neg w}{\to}_{\sigma}^* S_3$ and $S_2 \underset{\neg w}{\to}_{\sigma}^* S_3$ have no β_c steps).

- $N = !V$. In this case, $N = S_1 = S_2$, and the claim (#) is immediate.
- $N = (\lambda x.P)Q$, and $S_i = (\lambda x.P_i)Q_i$ (for $i \in \{1, 2, 3\}$). We have $P_i \underset{s}{\to}^* P$ and $Q_i \underset{s}{\to}^* Q$. Since P and Q are $\underset{s}{\to}$-normal, by *i.h.* we have:
  - the two sequences $P_1 \underset{s}{\to}^* P$ and $P_1 \underset{s}{\to}_{\sigma}^* P_3 \underset{s}{\to}^* P$ have the same number of β_c steps, and similarly $Q_1 \underset{s}{\to}^* Q$ and $Q_1 \underset{s}{\to}_{\sigma}^* Q_3 \underset{s}{\to}^* Q$. Hence, $\mathfrak{s}''_1$ and $\mathfrak{t}$ have the same number of β_c steps.
  - Similarly, $\mathfrak{s}''_2$ and $\mathfrak{t}$ have the same number of β_c steps.

This completes the proof of (#). We now can conclude that $\underset{s}{\to}$ is *uniformly normalizing*. If the term M has a sequence $\mathfrak{s} \colon M \underset{s}{\to}^* N$ where N is $\underset{s}{\to}$-normal, then no $\underset{s}{\to}$-sequence can have more β_c steps than $\mathfrak{s}$, because given any sequence $M \underset{s}{\to}^* T$ then (by confluence of $\underset{s}{\to}$) $T \underset{s}{\to}^* N$, and (by #) $M \underset{s}{\to}^* T \underset{s}{\to}^* N$ has the same number of β_c steps as $\mathfrak{s}$. Hence, all $\underset{s}{\to}$-sequences from M are finite. □

Normalizing strategies

We are ready to define and deal with normalizing strategies for $\lambda_{©}$. Our definition is inspired, and generalizes, the stratified strategy proposed in Guerrieri (2015), Guerrieri et al. (2017), which iterates weak reduction (there called head reduction) according to a more strict discipline.

Iterated e**-Reduction.** We define a family of normalizing strategies, parametrically on the reduction to iterate, which can be surface or weak. Let $\mathsf{e} \in \{\mathsf{w}, \mathsf{s}\}$. Reduction $\underset{\mathsf{le}}{\rightarrow}$ is defined as follows, by iterating $\underset{\mathsf{e}}{\rightarrow}$ in the left-to-right order (Theorem 8.14 then shows that $\underset{\mathsf{le}}{\rightarrow}$ is a normalizing strategy).

(1) If M is *not* $\underset{\mathsf{e}}{\rightarrow}$-normal:

$$\frac{M \underset{\mathsf{e}}{\rightarrow} M'}{M \underset{\mathsf{le}}{\rightarrow} M'}$$

(2) If M *is* $\underset{\mathsf{e}}{\rightarrow}$-normal (below, "$V$ $\beta_c\sigma$-normal" means $V \in Var$ or $V = \lambda x.L$ with L $\beta_c\sigma$-normal):

$$\frac{N \underset{\mathsf{le}}{\rightarrow} N'}{M := !(\lambda x.N) \underset{\mathsf{le}}{\rightarrow} !(\lambda x.N')} \qquad \frac{N \underset{\mathsf{le}}{\rightarrow} N'}{M := (\lambda x.N)L \underset{\mathsf{le}}{\rightarrow} (\lambda x.N')L} \qquad \frac{V\ \beta_c\sigma\text{-normal} \quad N \underset{\mathsf{le}}{\rightarrow} N'}{M := VN \underset{\mathsf{le}}{\rightarrow} VN'}$$

Theorem 8.14 (Normalization for $\beta_c\sigma$). *Assume* $M \rightarrow^*_{\beta_c\sigma} N$ *where* N *is* $\rightarrow_{\beta_c\sigma}$*-normal. Let* $\mathsf{e} \in \{\mathsf{w}, \mathsf{s}\}$. *Then, every maximal* $\underset{\mathsf{le}}{\rightarrow}_{\beta_c\sigma}$*-sequence from* M *ends in* N.

Proof. By induction on the term N. Let $\mathfrak{s} = M, M_1, M_2, \ldots$ be a maximal $\underset{\mathsf{le}}{\rightarrow}$-sequence from M. We write $\rightarrow$, $\underset{\mathsf{e}}{\rightarrow}$, $\underset{\neg\mathsf{e}}{\rightarrow}$ and $\underset{\mathsf{le}}{\rightarrow}$ for $\rightarrow_{\beta_c\sigma}$, $\underset{\mathsf{e}}{\rightarrow}_{\beta_c\sigma}$, $\underset{\neg\mathsf{e}}{\rightarrow}_{\beta_c\sigma}$ and $\underset{\mathsf{le}}{\rightarrow}_{\beta_c\sigma}$, respectively. We observe that

(**) every maximal $\underset{\mathsf{e}}{\rightarrow}$-sequence from M is finite.

Indeed, from $M \rightarrow^* N$, by e-factorization, we have that $M \underset{\mathsf{e}}{\rightarrow} L \underset{\neg\mathsf{e}}{\rightarrow}^* N$. Since N is $\underset{\mathsf{e}}{\rightarrow}$-normal, so is L (by Lemma 8.3) and (**) follows by uniform normalization of $\underset{\mathsf{e}}{\rightarrow}$ (Theorem 8.13).

Let $\mathfrak{s}' \sqsubseteq \mathfrak{s}$ be the maximal prefix of $\mathfrak{s}$ such that $M_i \underset{\mathsf{e}}{\rightarrow} M_{i+1}$. Since it is finite, $\mathfrak{s}'$ is $M, \ldots, M_k$, where M_k is e-normal. Let $\mathfrak{s}'' = M_k, M_{k+1} \ldots$ be the sequence such that $\mathfrak{s} = \mathfrak{s}'\mathfrak{s}''$.

Note that all terms in $\mathfrak{s}''$ are e-normal (by repeatedly using Lemma 8.3 from M_k), hence $M_k \underset{\neg\mathsf{e}}{\rightarrow} M_{k+1} \underset{\neg\mathsf{e}}{\rightarrow} \ldots$, and (by shape preservation, Fact 6.2) all terms in $\mathfrak{s}''$ have the same shape as M_k.

By confluence of $\rightarrow$ (Proposition 5.9.2), $M_k \rightarrow^* N$. Again, all terms in this sequence are e-normal, by repeatedly using Lemma 8.3 from M_k. So, $M_k \underset{\neg\mathsf{e}}{\rightarrow}^* N$, and (by shape preservation, Fact 6.2) M_k and N have the same shape.

We have established that M_k and all terms in $\mathfrak{s}'' : M_k, M_{k+1}, \ldots$ have the same shape as N. Now we examine the possible cases for N.

- $N = !x$, and $M_k = !x$. Trivially $M \underset{\mathsf{le}}{\rightarrow}^* M_k = N$.
- $N = !(\lambda x.N_P)$ and $M_k = !(\lambda x.P)$ with $P \rightarrow^* N_P$. Since N_P is $\beta_c\sigma$-normal, by *i.h.* every maximal $\underset{\mathsf{le}}{\rightarrow}$-sequence from P terminates in N_P, and so every maximal $\underset{\mathsf{le}}{\rightarrow}$-sequence from $!(\lambda x.P)$ terminates in $!(\lambda x.N_P) = N$. Since the sequence $\mathfrak{s}'' = M_k, M_{k+1}, \ldots$ is a maximal $\underset{\mathsf{le}}{\rightarrow}$-sequence, we have that $\mathfrak{s} = \mathfrak{s}'\mathfrak{s}''$ is as follows

$$M \underset{\mathsf{le}}{\rightarrow}^* M_k = !(\lambda x.P) \underset{\mathsf{le}}{\rightarrow}^* !(\lambda x.N_P) = N.$$

- $N = (\lambda x.N_P)N_Q$ and $M_k = (\lambda x.P)Q$, with $P \rightarrow^* N_P$ and $Q \rightarrow^* N_Q$. Since N_P and N_Q are both $\beta_c\sigma$-normal, by *i.h.*:

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

– every maximal $\underset{\mathsf{le}}{\rightarrow}$-sequence from P ends in N_P. So every $\underset{\mathsf{le}}{\rightarrow}$-sequence from $(\lambda x.P)Q$ eventually reaches $(\lambda x.N_P)Q$;
– every maximal $\underset{\mathsf{le}}{\rightarrow}$-sequence from Q ends in N_Q. So every $\underset{\mathsf{le}}{\rightarrow}$-sequence from $(\lambda x.N_P)Q$ eventually reaches $(\lambda x.N_P)N_Q = N$.

Therefore $\mathfrak{s}$ is as follows

$$M \underset{\mathsf{le}}{\rightarrow}^* M_k = (\lambda x.P)Q \underset{\mathsf{le}}{\rightarrow}^* (\lambda x.N_P)Q \underset{\mathsf{le}}{\rightarrow}^* (\lambda x.N_P)N_Q = N.$$

- $N = xN_Q$ and $M_k = xQ$. Similar to the previous one. □

From a normalizing strategy for $\beta_c\sigma$ (Theorem 8.14), we derive a normalizing strategy in $\lambda_©$.

Corollary 8.15. *(Normalization for $\lambda_©$) Let $\mathsf{e} \in \{\mathsf{w}, \mathsf{s}\}$. If M is ©-normalizing, then any maximal $\underset{\mathsf{le}}{\rightarrow}_{\beta_c\sigma}$-sequence from M followed by any maximal $\rightarrow_\iota$-sequence ends in the ©-normal form of M.*

Proof. By Theorem 8.14, every maximal $\underset{\mathsf{le}}{\rightarrow}_{\beta_c\sigma}$-sequence from M ends in a $\rightarrow_{\beta_c\sigma}$-normal form L. Since $\rightarrow_\iota \subseteq \rightarrow_{\sigma\mathsf{id}}$ is strongly normalizing (Proposition 8.6), every maximal $\rightarrow_\iota$-sequence from L ends in a $\rightarrow_\iota$-normal form N, which is also $\rightarrow_{\beta_c\sigma}$-normal by Lemma 8.1. Therefore, N is ©-normal and this is the unique ©-normal form of M since $\rightarrow_©$ is confluent (Proposition 5.9.3). □

9. Conclusions and Related Work

9.1 Discussion: reduction and evaluation

In computational calculi, it is standard practice to define evaluation as *weak reduction*, aka *sequencing* (Dal Lago et al. 2017; Filinski 1996; Jones et al. 1998; Levy et al. 2003). Despite the prominent role that weak reduction has in the literature, in particular for calculi with effects, what one discovers when analyzing the rewriting properties is somehow unexpected. As we observe in Section 5, where we consider both the computational core $\lambda_©$ (de' Liguoro and Treglia 2020), and a widely recognized reference such as the calculus λ_{ml^*} by Sabry and Wadler (1997) (in turn inspired by Moggi 1988, 1989, 1991), while full reduction is confluent, the closure of the rules under *evaluation contexts* turns out to be *non-deterministic*, *non-confluent*, and its *normal forms* are *not unique*. The issues come from the monadic rules of *identity* and *associativity*, hence they are common to *all* computational calculi.

A Bridge between Evaluation and Reduction. On the one hand, computational λ-calculi have an unrestricted *non-deterministic reduction* that generates the equational theory of the calculus, studied for foundational and semantic purposes. On the other hand, *weak reduction* models evaluation in an ideal programming language. It is then natural to wonder what is the relation between reduction and evaluation. This is the first contribution of this paper. We establish a bridge between evaluation and reduction via a factorization theorem stating that every reduction can be rearranged so as to bring forward weak reduction steps.

We focused on the rewriting theory of a specific computational calculus, namely the computational core $\lambda_©$ (de' Liguoro and Treglia 2020). We expect that our results and approach can be adapted also to other computational calculi such as λ_{ml^*}. This demands further investigations. Transferring the results is not immediate because the correspondence between the two calculi is not direct with respect to the *rewriting* (see Remark 3.10).

9.2 Technical contributions

We studied the rewriting theory of the computational core $\lambda_©$ introduced in de' Liguoro and Treglia (2020), a variant of Moggi's λ_c-calculus (Moggi 1988), focusing on two questions:

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

- how to reach *values*?
- how to reach *normal forms*?

For the first point, we show that weak β_c-reduction is enough (Section 7). For the second question, we define a family of normalizing strategies (Section 8).

We have faced the issues caused by identity and associativity rules (which internalize the monadic rules in the syntax) and dealt with them by means of factorization techniques.

We have investigated in depth the structure of normalizing reductions, and we assessed the role of the σ-rule (aka associativity) as computational and not merely structural. We found out that it plays at least three distinct, independent roles in $\lambda_{©}$:

- σ unblocks "premature" β_c-normal forms so as to guarantee that there are not ©-normalizing terms whose semantics is the same as diverging terms, as we have seen in Section 8.2;
- it internalizes the associativity of Kleisli composition into the calculus, as a syntactic reduction rule, as explained in Section 1 after Equation (3);
- it "simulates" the contextual closure of the β_c-rule for terms that reduce to a value, as we have seen in Theorem 7.4.

9.3 Related work

Relation with Moggi's Calculus. Since our focus is on operational properties and reduction theory, we chose the computational core $\lambda_{©}$ (de' Liguoro and Treglia 2020) among the different variants of computational calculi in the literature inspired by Moggi's seminal work (Moggi 1988, 1989, 1991). Indeed, the computational core $\lambda_{©}$ has a "minimal" syntax that internalizes Moggi's original idea of deriving a calculus from the categorical model consisting of the Kleisli category of a (strong) monad. For instance, $\lambda_{©}$ does not have to consider both a pure and a (potentially) effectful functional application. So, $\lambda_{©}$ has less syntactic constructors and less reductions rules with respect to other computational calculi, and this simplifies our operational study.

Let us discuss the difference between $\lambda_{©}$ and Moggi's λ_c. As observed in Sections 1 and 3, the first formulation of λ_c and of its reduction relation was introduced in Moggi (1988), where it is formalized by using *let*-constructor. Indeed, this operator is not just a syntactical sugar for the application of λ-abstraction. In fact, it represents the extension to computations of functions from values to computations, therefore interpreting Kleisli composition. Combining *let* with ordinary abstraction and application is at the origin of the complexity of the reduction rules in Moggi (1988). On the other hand, this allows extensionality to be internalized. Adding the η-rule to $\lambda_{©}$ breaks confluence, as shown in de' Liguoro and Treglia (2020).

Besides using *let* or not, a major difference of $\lambda_{©}$ with respect to λ_c is the neat distinction among the two syntactical sorts of terms, restricting the combination of values and non-values since the very definition of the grammar of the language. In spite of these differences, in de' Liguoro and Treglia (2019, Section 9) it has been proved that there exists an interpretation of λ_c into $\lambda_{©}$ that preserves the reduction, while there is a reverse translation that preserves convertibility, only.

Other Related Work. Sabry and Wadler (1997) is the first work on the computational calculus to put on center stage the reduction. Still the focus of the paper are the properties of the translation between that and the monadic metalanguage – the reduction theory itself is not investigated.

In Herbelin and Zimmermann (2009) a different refinement of λ_c has been proposed. Its reduction rules are divided into a purely operational, a structural and an observational system. It is proved that the purely operational system suffices to reduce any closed term to a value. This result is similar to our Theorem 7.6, with weak β_c steps corresponding to head reduction in Herbelin and Zimmermann (2009). Interestingly, the analogous of our rule σ is part of the structural system, while the rule corresponding to our id is generalized and considered as an observational rule. Unlike our work, normalization is not studied in Herbelin and Zimmermann (2009).

Surface reduction is a generalization of weak reduction that comes from linear logic. We inherit surface factorization from the linear λ-calculus in Simpson (2005). Such a reduction has been recently studied in several variants of the λ-calculus, especially for semantic purposes (Accattoli and Guerrieri 2016; Accattoli and Paolini 2012; Bucciarelli et al. 2020; Carraro and Guerrieri 2014; Ehrhard and Guerrieri 2016; Guerrieri and Manzonetto 2019; Guerrieri and Olimpieri 2021; Guerrieri 2019).

Regarding the σ-rule, in Carraro and Guerrieri (2014) two commutation rules are added to Plotkin's CbV λ-calculus in order to remove meaningless normal forms – the resulting calculus is called *shuffling*. The commutative rule there called σ_3 is literally the same as σ here. In the setting of the *shuffling calculus*, properties such as the fact that all maximal surface $\beta_v\sigma$-reduction sequences from the same term M have the same number of β_v steps, and so such a reduction is uniformly normalizing, were known via semantic tools (Carraro and Guerrieri 2014; Guerrieri 2019), namely non-idempotent intersection types. In this paper, we give the first syntactic proof of such a result.

A relation between the computational calculus, Simpson (2005) and other linear calculi are well known in the literature, see for example Egger et al. (2009), Sabry and Wadler (1997), Maraist et al. (1999).

In de' Liguoro and Treglia (2020), Theorem 8.4 states that any closed term returns a value if and only if it is convergent according to a big-step operational semantics. That proof is incomplete and needs a more complex argument via factorization, as we do here to prove Theorem 7.6 (from which that statement in de' Liguoro and Treglia 2020 easily follows).

Acknowledgements. We are in debt with Vincent van Oostrom, for the technical issues he pinpointed, and for his many insightful technical suggestions. We also thank the referees for their valuable comments. This work was partially supported by the ANR project PPS: ANR-19-CE48-0014.

Note

1 Precisely, Accattoli studies the relation between the kernel calculus λ_{vker} and the *value substitution calculus* λ_{vsub}, i.e. CbV and the kernel extended with explicit substitutions. The syntax is slightly different, but not in an essential way.

References

Accattoli, B. (2015). Proof nets and the call-by-value λ-calculus. *Theoretical Computer Science* **606** 2–24.

Accattoli, B., Faggian, C. and Guerrieri, G. (2019). Factorization and normalization, essentially. In: *APLAS 2019: Programming Languages and Systems*, Lecture Notes in Computer Science, vol. 11893, Springer Verlag, 159–180.

Accattoli, B., Faggian, C. and Guerrieri, G. (2021). Factorize factorization. In: *29th EACSL Annual Conference on Computer Science Logic, CSL 2021, January 25–28, 2021, Ljubljana, Slovenia (Virtual Conference)*, LIPIcs, vol. 183, Schloss Dagstuhl, 6:1–6:25.

Accattoli, B. and Guerrieri, G. (2016). Open call-by-value. In: *Programming Languages and Systems - 14th Asian Symposium, APLAS 2016, Hanoi, Vietnam, November 21–23, 2016, Proceedings*, Lecture Notes in Computer Science, vol. 10017, 206–226, Springer.

Accattoli, B. and Paolini, L. (2012). Call-by-value solvability, revisited. In: *Functional and Logic Programming - 11th International Symposium, FLOPS 2012*, Lecture Notes in Computer Science, vol. 7294, Springer, 4–16.

Baader, F. and Nipkow, T. (1998). *Term Rewriting and All that*, Cambridge University Press, USA.

Barendregt, H. (1984). *The Lambda Calculus: Its Syntax and Semantics*, revised edition, Studies in Logic and the Foundations of Mathematics, vol. 103, North-Holland.

Benton, N., Hughes, J. and Moggi, E. (2002). Monads and effects. In: *Applied Semantics, International Summer School, APPSEM 2000*, Lecture Notes in Computer Science, vol. 2395, Springer, 42–122.

Bucciarelli, A., Kesner, D., Ríos, A. and Viso, A. (2020). The bang calculus revisited. In: Nakano, K. and Sagonas, K. (eds.) *Functional and Logic Programming - 15th International Symposium, FLOPS 2020, Akita, Japan, September 14–16, 2020, Proceedings*, Lecture Notes in Computer Science, vol. 12073, Springer, 13–32.

Carraro, A. and Guerrieri, G. (2014). A semantical and operational account of call-by-value solvability. In: *Foundations of Software Science and Computation Structures - 17th International Conference, FOSSACS 2014, Proceedings*, Lecture Notes in Computer Science, vol. 8412, Springer, 103–118.

Crary, K. (2009). Simple proof of call-by-value standardization. Technical report, Carnegie Mellon University, Computer Science Department. Paper 474.

Dal Lago, U., Gavazzo, F. and Levy, P. B. (2017). Effectful applicative bisimilarity: Monads, relators, and Howe's method. In: *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017*, IEEE Computer Society, 1–12.

de' Liguoro, U. and Treglia, R. (2019). Intersection types for the computational lambda-calculus. CoRR, abs/1907.05706.

de' Liguoro, U. and Treglia, R. (2020). The untyped computational λ-calculus and its intersection type discipline. *Theoretical Computer Science* **846** 141–159.

Ehrhard, T. and Guerrieri, G. (2016). The bang calculus: An untyped lambda-calculus generalizing call-by-name and call-by-value. In: *Proceedings of the 18th International Symposium on Principles and Practice of Declarative Programming (PPDP 2016)*, ACM, 174–187.

Ehrhard, T. (2012). Collapsing non-idempotent intersection types. In: *Computer Science Logic (CSL'12) - 26th International Workshop/21st Annual Conference of the EACSL, CSL 2012*, LIPIcs, vol. 16, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 259–273.

Egger, J., Møgelberg, R. E. and Simpson, A. (2009). Enriching an effect calculus with linear types. In: Grädel, E. and Kahle, R. (eds.) *Computer Science Logic, 23rd international Workshop, CSL 2009, 18th Annual Conference of the EACSL. Proceedings*, Lecture Notes in Computer Science, vol. 5771, Springer, 240–254.

Faggian, C. and Guerrieri, G. (2021). Factorization in call-by-name and call-by-value calculi via linear logic. In: *Foundations of Software Science and Computation Structures - 24th International Conference, FOSSACS 2021, Proceedings*, Lecture Notes in Computer Science, vol. 12650, Springer, 205–225.

Felleisen, M. (1988). The theory and practice of first-class prompts. In: *Proceedings of the 15th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL'88, New York, NY, USA, Association for Computing Machinery, 180–190.

Filinski, A. (1996). *Controlling Effects*. Phd thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Girard, J.-Y. (1987). Linear logic. *Theoretical Computer Science* **50** 1–102.

Guerrieri, G. and Manzonetto, G. (2019). The bang calculus and the two Girard's translations. In: *Proceedings Joint International Workshop on Linearity & Trends in Linear Logic and Applications (Linearity-TLLA 2018)*, EPTCS, vol. 292, 15–30.

Guerrieri, G. and Olimpieri, F. (2021). Categorifying non-idempotent intersection types. In: *29th EACSL Annual Conference on Computer Science Logic, CSL 2021, January 25–28, 2021, Ljubljana, Slovenia (Virtual Conference)*, LIPIcs, vol. 183, Schloss Dagstuhl, 25:1–25:24.

Guerrieri, G., Paolini, L. and Ronchi Della Rocca, S. (2017). Standardization and conservativity of a refined call-by-value lambda-calculus. *Logical Methods in Computer Science* **13**(4) 4–29.

Guerrieri, G. (2015). Head reduction and normalization in a call-by-value lambda-calculus. In: *2nd International Workshop on Rewriting Techniques for Program Transformations and Evaluation, WPTE 2015*, OASICS, vol. 46, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 3–17.

Guerrieri, G. (2019). Towards a semantic measure of the execution time in call-by-value lambda-calculus. In: *Proceedings Twelfth Workshop on Developments in Computational Models and Ninth Workshop on Intersection Types and Related Systems, DCM/ITRS 2018*, EPTCS, vol. 293, 57–72.

Hindley, J. R. (1964). *The Church-Rosser Property and a Result in Combinatory Logic*. Phd thesis, University of Newcastle-upon-Tyne.

Herbelin, H. and Zimmermann, S. (2009). An operational account of call-by-value minimal and classical lambda-calculus in "natural deduction" form. In: Curien, P.-L. (ed.) *Typed Lambda Calculi and Applications, 9th International Conference, TLCA 2009. Proceedings*, Lecture Notes in Computer Science, vol. 5608, Springer, 142–156.

Jones, S. L. P., Shields, M., Launchbury, J. and Tolmach, A. P. (1998). Bridging the gulf: A common intermediate language for ML and haskell. In: MacQueen, D. B. and Cardelli, L. (eds.) *POPL'98, Proceedings of the 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Diego, CA, USA, January 19–21, 1998*, ACM, 49–61.

Lago, U. D. and Martini, S. (2008). The weak lambda calculus as a reasonable machine. *Theoretical Computer Science* **398** (1–3) 32–50.

Leroy, X. (1990). The ZINC experiment: An economical implementation of the ML language. Technical report 117, INRIA.

Levy, P. B. (1999). Call-by-push-value: A subsuming paradigm. In: *Typed Lambda Calculi and Applications, 4th International Conference (TLCA'99)*, Lecture Notes in Computer Science, vol. 1581, 228–242.

Levy, P. B., Power, J. and Thielecke, H. (2003). Modelling environments in call-by-value programming languages. *Information and Computation* **185** (2) 182–210.

MacLane, S. (1997). *Categories for the Working Mathematician*, 2nd ed., Graduate Texts in Mathematics, Springer, New York, NY.

Maraist, J., Odersky, M., Turner, D. N. and Wadler, P. (1999). Call-by-name, call-by-value, call-by-need and the linear lambda calculus. *Theoretical Computer Science* **228** (1–2) 175–210.

Moggi, E. (1988). Computational Lambda-calculus and Monads. Report ECS-LFCS-88-66, University of Edinburgh, Edinburgh, Scotland.
Moggi, E. (1989). Computational lambda-calculus and monads. In: *Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS'89)*, IEEE Computer Society, 14–23.
Moggi, E. (1991). Notions of computation and monads. *Information and Computation* **93** (1) 55–92.
Newman, M. H. A. (1942). On theories with a combinatorial definition of equivalence. *Annals of Mathematics* **43** (2).
Plotkin, G. D. (1975). Call-by-name, call-by-value and the lambda-calculus. *Theoretical Computer Science* **1** (2) 125–159.
Sabry, A. and Wadler, P. (1997). A reflection on call-by-value. *ACM Transactions on Programming Languages and Systems* **19** (6) 916–941.
Scott, D. (1980). Relating theories of the λ-calculus. In: Hindley, R. J. and Seldin, J. P. (eds.) *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*, Academic Press, 403–450.
Simpson, A. (2005). Reduction in a linear lambda-calculus with applications to operational semantics. In: Giesl, J. (ed.) *Term Rewriting and Applications*, Berlin, Heidelberg, Springer Berlin Heidelberg, 219–234.
Terese. (2003). *Term Rewriting Systems*, Cambridge Tracts in Theoretical Computer Science, vol. 55, Cambridge University Press.
van Oostrom, V. (1994). Confluence by decreasing diagrams. *Theoretical Computer Science* **126** (2) 259–280.
van Oostrom, V. (2008). Confluence by decreasing diagrams converted. In: *Rewriting Techniques and Applications, 19th International Conference, RTA 2008*, Lecture Notes in Computer Science, vol. 5117, Springer, 306–320.
van Oostrom, V. (2020a). Private communication via electronic mail.
van Oostrom, V. (2020b). Some symmetries of commutation diamonds. In: *Proceedings of the 9th International Workshop on Confluence (IWC 2020), Paris, France*, 1–5.
Wadler, P. (1992). The essence of functional programming. In: *Conference Record of the Nineteenth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 1992*, ACM Press, 1–14.
Wadler, P. (1995). Monads for functional programming. In: *Advanced Functional Programming, First International Spring School on Advanced Functional Programming Techniques*, Lecture Notes in Computer Science, vol. 925, Springer, 24–52.
Wadler, P. and Thiemann, P. (2003). The marriage of effects and monads. *ACM Transactions on Computational Logic* **4** (1) 1–32.

APPENDIX

Appendix A. General properties of the contextual closure

Shape Preservation. We start by recalling a basic but key property of contextual closure. If a step $\to_\gamma$ is obtained by closure under *non-empty context* of a rule $\mapsto_\gamma$, then it preserves the shape of the term. We say that T and T' have *the same shape* if both terms are an application (resp. an abstraction, an variable, a term of shape $!P$).

Fact 6.2. (Shape preservation). *Let $\mapsto_\rho$ be a rule and $\to_\rho$ be its contextual closure. Assume $T = \mathsf{C}\langle R\rangle \to_\rho \mathsf{C}\langle R'\rangle = T'$ where $R \mapsto_\rho R'$ and $\mathsf{C} \neq \langle\,\rangle$. Then T and T' have the same shape.*

Note that a root step $\mapsto$ is both a *weak* and a *surface* step.

The implication in the previous lemma cannot be reversed as the following example shows:

$$M = V(\mathbf{I}P) \to_\iota VP = N$$

M is a σ-redex, but N is not.

Substitutivity. A relation $\hookrightarrow$ on terms is *substitutive* if

$$R \hookrightarrow R' \text{ implies } R[Q/x] \hookrightarrow R'[Q/x]. \qquad \textbf{(substitutive)}$$

An obvious induction on the shape of terms shows the following Barendregt (1984, p. 54).

Fact A.1 (Substitutive). *Let $\to_\gamma$ be the contextual closure of $\mapsto_\gamma$.*

(1) If $\mapsto_\gamma$ is substitutive then $\to_\gamma$ is substitutive: $T \to_\gamma T'$ implies $T[Q/x] \to_\gamma T'[Q/x]$.
(2) If $Q \to_\gamma Q'$ then $T[Q/x] \to_\gamma^* T[Q'/x]$.

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

Appendix B. Properties of the syntax $\Lambda^!$

In this section, we consider the set of terms $\Lambda^!$ (the same syntax as the *full* bang calculus, as defined in Section 4), endowed with a generic reduction $\to_\rho$ (from a generic rule $\mapsto_\rho$). We study some properties that hold in general in $(\Lambda^!, \to_\rho)$.

Terms are generated by the grammar:

$$T, S, R ::= x \mid ST \mid \lambda x.T \mid !T \qquad \textbf{(terms } \Lambda^!\textbf{)}$$

Contexts (C), *surface contexts* (S) and *weak contexts* (W) are generated by the grammars:

$$\mathsf{C} ::= \langle\rangle \mid T\mathsf{C} \mid \mathsf{C}T \mid \lambda x.\mathsf{C} \mid !\mathsf{C} \qquad \textbf{(contexts)}$$
$$\mathsf{S} ::= \langle\rangle \mid T\mathsf{S} \mid \mathsf{S}T \mid \lambda x.\mathsf{S} \qquad \textbf{(surface contexts)}$$
$$\mathsf{W} ::= \langle\rangle \mid T\mathsf{W} \mid \mathsf{W}T \mid !\mathsf{W} \qquad \textbf{(weak contexts)}$$

If $\mapsto_\rho$ is a rule, the reduction $\to_\rho$ its the closure under context C. *Surface reduction* $\underset{\mathsf{s}}{\to}_\rho$ (resp. *weak reduction* $\underset{\mathsf{w}}{\to}_\rho$) is the closure of $\mapsto_\rho$ under surface contexts S (resp. weak contexts W). *Non-surface reduction* $\underset{\neg\mathsf{s}}{\to}_\rho$ (resp. *non-weak reduction* $\underset{\neg\mathsf{w}}{\to}_\rho$) is the closure of $\mapsto_\rho$ under contexts C that are not surface (resp. not weak).

B.1 Shape preservation for internal steps in $\Lambda^!$.

Fact 6.2 (p. 952) implies that $\underset{\neg\mathsf{s}}{\to}_\rho$ and $\underset{\neg\mathsf{w}}{\to}_\rho$ steps always preserve the shape of terms. We recall that we write $\mapsto_\rho$ to indicate the step $\to_\rho$ obtained by *empty contextual closure*. The following property immediately follows from Fact 6.2.

Fact B.1 (Internal Steps). *Let $\mapsto_\rho$ be a rule and $\to_\rho$ be its contextual closure. The following hold for $\underset{\mathsf{i}}{\to}_\rho \in \{\underset{\neg\mathsf{s}}{\to}_\rho, \underset{\neg\mathsf{w}}{\to}_\rho\}$.*

(1) Reduction $\underset{\mathsf{i}}{\to}_\rho$ preserves the shapes of terms.
(2) There is no T such that $T \underset{\mathsf{i}}{\to}_\rho x$, for any variable x.
(3) $T \underset{\mathsf{i}}{\to}_\rho\ !U_1$ implies $T = !T_1$ and $T_1 \to_\rho U_1$.
(4) $T \underset{\mathsf{i}}{\to}_\rho \lambda x.U_1$ implies $T = \lambda x.T_1$ and $T_1 \to_\rho U_1$.
(5) $T \underset{\mathsf{i}}{\to}_\rho U_1U_2$ implies $T = T_1T_2$, with either (i) $T_1 \underset{\mathsf{i}}{\to}_\rho U_1$ (and $T_2 = U_2$) or (ii) $T_2 \underset{\mathsf{i}}{\to}_\rho U_2$ (and $T_1 = U_1$). Moreover, T_1 and U_1 have the same shape, and so T_2 and U_2.

Corollary B.2. *Let $\mapsto_\rho$ be a rule and $\to_\rho$ be its contextual closure. Assume $T \underset{\neg\mathsf{s}}{\to}_\rho S$ or $T \underset{\neg\mathsf{w}}{\to}_\rho S$.*

- *T is a $\beta_!$-redex if and only if S is.*
- *T is a σ-redex if and only if S is.*

Proof. The left-to-right direction follows from Fact B.1.1. The right-to-left direction is obtained by repetitively applying Fact B.1.3–5. □

B.2 Surface Factorization, Modularly

In an abstract setting, let us consider a rewrite system $(A, \to)$ where $\to\, = \,\to_\xi \cup \to_\gamma$. Under which condition $\to$ admits factorization, assuming that both $\to_\xi$ and $\to_\gamma$ do? That is, if $\to_\xi = \underset{\mathsf{e}}{\to}_\xi \cup \underset{\mathsf{i}}{\to}_\xi$ and $\to_\gamma = \underset{\mathsf{e}}{\to}_\gamma \cup \underset{\mathsf{i}}{\to}_\gamma$ e-factorize (i.e. $\to_\xi^* \subseteq \underset{\mathsf{e}}{\to}_\xi^* \cup \underset{\mathsf{i}}{\to}_\xi^*$ and $\to_\xi^* \subseteq \underset{\mathsf{e}}{\to}_\xi^* \cup \underset{\mathsf{i}}{\to}_\xi^*$), is it the case that

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

$\to^* \subseteq \underset{e}{\to}^* \cup \underset{i}{\to}^*$ (where $\underset{e}{\to} := \underset{e}{\to}_\xi \cup \underset{e}{\to}_\gamma$ and $\underset{i}{\to} := \underset{i}{\to}_\xi \cup \underset{i}{\to}_\gamma$)? To deal with this question, a technique for proving factorization for *compound systems* in a *modular* way has been introduced in Accattoli et al. (2021). The approach can be seen as an analog – for factorization – of the classical technique for confluence based on Hindley–Rosen lemma: if $\to_\xi, \to_\gamma$ are e-factorizing reductions, their union $\to_\xi \cup \to_\gamma$ also is, provided that two *local* conditions of commutation hold.

Theorem B.3 (Modular factorization, abstractly Accattoli et al. 2021). *Let* $\to_\xi = (\underset{e}{\to}_\xi \cup \underset{i}{\to}_\xi)$ *and* $\to_\gamma = (\underset{e}{\to}_\gamma \cup \underset{i}{\to}_\gamma)$ *be* e*-factorizing reductions. Let* $\underset{e}{\to} := \underset{e}{\to}_\xi \cup \underset{e}{\to}_\gamma$, *and* $\underset{i}{\to} := \underset{i}{\to}_\xi \cup \underset{i}{\to}_\gamma$. *The union* $\to_\xi \cup \to_\gamma$ *satisfies factorization* $\mathtt{Fact}(\underset{e}{\to}, \underset{i}{\to})$ *if the following swaps hold*

$$\underset{i}{\to}_\xi \cdot \underset{e}{\to}_\gamma \subseteq \underset{e}{\to}_\gamma \cdot \to_\xi^* \quad \textit{and} \quad \underset{i}{\to}_\gamma \cdot \underset{e}{\to}_\xi \subseteq \underset{e}{\to}_\xi \cdot \to_\gamma^* \qquad \textbf{(Linear Swaps)}$$

Extensions of the bang calculus. Following Faggian and Guerrieri (2021), we now consider a calculus $(\Lambda^!, \to)$, where $\to = \to_{\beta_!} \cup \to_\gamma$ and $\to_\gamma$ is the contextual closure of a new rule $\mapsto_\gamma$. Theorem B.3 states that the compound system $\to_{\beta_!} \cup \to_\gamma$ satisfies surface factorization if $\mathtt{Fact}(\underset{s}{\to}_{\beta_!}, \underset{\neg s}{\to}_{\beta_!})$, $\mathtt{Fact}(\underset{s}{\to}_\gamma, \underset{\neg s}{\to}_\gamma)$, and the two linear swaps hold. We know that $\mathtt{Fact}(\underset{s}{\to}_{\beta_!}, \underset{\neg s}{\to}_{\beta_!})$ always hold. We now show that verifying the linear swaps reduces to a single simple test, leading to Proposition 6.5.

First, we observe that each linear swap condition can be tested by considering for the surface step only $\mapsto$, that is, only the closure of $\mapsto$ under *empty* context. This is expressed in the following lemma, where we include also a useful variant.

Lemma B.4 (Root linear swaps). *In* $\Lambda^!$, *let* $\to_\xi, \to_\gamma$ *be the contextual closure of rules* $\mapsto_\xi, \mapsto_\gamma$.

(1) $\underset{\neg s}{\to}_\xi \cdot \mapsto_\gamma \subseteq \underset{s}{\to}_\gamma \cdot \to_\xi^*$ *implies* $\underset{\neg s}{\to}_\xi \cdot \underset{s}{\to}_\gamma \subseteq \underset{s}{\to}_\gamma \cdot \to_\xi^*$.

(2) Similarly, $\underset{\neg s}{\to}_\xi \cdot \mapsto_\gamma \subseteq \underset{s}{\to}_\gamma \cdot \to_\xi^=$ *implies* $\underset{\neg s}{\to}_\xi \cdot \underset{s}{\to}_\gamma \subseteq \underset{s}{\to}_\gamma \cdot \to_\xi^=$.

Proof. Assume $M \underset{\neg s}{\to}_\xi U \underset{s}{\to}_\gamma N$. If U is the redex, the claim holds by assumption. Otherwise, we prove $M \underset{s}{\to}_\gamma \cdot \to_\xi^* N$, by induction on the structure of U. Observe that both M and N have the same shape as U (by Property 6.2).

- $U = U_1 U_2$ (hence $M = M_1 M_2$ and $N = N_1 N_2$). We have two cases.
  (1) Case $U_1 \underset{s}{\to}_\gamma N_1$. By Fact B.1, either $M_1 \to_\xi U_1$ or $M_2 \to_\xi U_2$.
    a. Assume $M := M_1 M_2 \underset{\neg s}{\to}_\xi U_1 M_2 \underset{s}{\to}_\gamma N_1 M_2 =: N$.
    We have $M_1 \underset{\neg s}{\to}_\xi U_1 \underset{s}{\to}_\gamma N_1$, and we conclude by *i.h.*.
    b. Assume $M := U_1 M_2 \underset{\neg s}{\to}_\xi U_1 U_2 \underset{s}{\to}_\gamma N_1 U_2 =: N$.
    Then $U_1 M_2 \underset{s}{\to}_\gamma N_1 M_2 \to_\xi N_1 U_2$.
  (2) Case $U_2 \underset{s}{\to}_\gamma N_2$. Similar to the above.

- $U = \lambda x.U_0$ (hence $M = \lambda x.M_0$ and $N = \lambda x.N_0$). We conclude by *i.h.*.

Cases $U = !U_0$ or $U = x$ do not apply. □

As we study $\to_{\beta_!} \cup \to_\gamma$, one of the linear swap is $\underset{\neg s}{\to}_\gamma \cdot \underset{s}{\to}_{\beta_!} \subseteq \underset{s}{\to}_{\beta_!} \cdot \to_\gamma^*$. We show that *any* $\to_\gamma$ linearly swaps after $\underset{s}{\to}_{\beta_!}$ as soon as $\mapsto_\gamma$ is *substitutive*.

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

Lemma B.5 (Swap with $\underset{s}{\rightarrow}_{\beta_!}$). *If* $\mapsto_\gamma$ *is substitutive, then* $\underset{\neg s}{\rightarrow}_\gamma \cdot \underset{s}{\rightarrow}_{\beta_!} \subseteq \underset{s}{\rightarrow}_{\beta_!} \cdot \rightarrow_\gamma^*$ *always holds.*

Proof. We prove $\underset{\neg s}{\rightarrow}_\gamma \cdot \mapsto_{\beta_!} \subseteq \underset{s}{\rightarrow}_{\beta_!} \cdot \rightarrow_\gamma^*$, and conclude by Lemma B.4.

Assume $M \underset{\neg s}{\rightarrow}_\gamma (\lambda x.P)!Q \mapsto_{\beta_!} P[Q/x]$. We want to prove $M \underset{s}{\rightarrow}_{\beta_!} \cdot \rightarrow_\gamma^* P[Q/x]$. By Fact B.1, $M = M_1 M_2$ and either $M_1 = \lambda x.P_0 \rightarrow_\gamma \lambda x.P$ or $M_2 = !Q_0 \rightarrow_\gamma !Q$.

- In the first case, $M = (\lambda x.P_0)!Q$, with $P_0 \rightarrow_\gamma P$. So, $M = (\lambda x.P_0)!Q \mapsto_{\beta_!} P_0[Q/x]$ and we conclude by substitutivity of $\rightarrow_\gamma$ (Fact A.1.1).
- In the second case, $M = (\lambda x.P)!Q_0$ with $Q_0 \rightarrow_\gamma Q$. Therefore, $M = (\lambda x.P)!Q_0 \mapsto_{\beta_!} P[Q_0/x]$, and we conclude by Fact A.1.2. □

Summing up, since surface factorization for $\beta_!$ is known, we obtain the following compact test for surface factorization in extensions of $\rightarrow_{\beta_!}$.

Proposition B.6 (A modular test for surface factorization). *Let* $\rightarrow_{\beta_!}$ *be* $\beta_!$*-reduction and* $\rightarrow_\gamma$ *be the contextual closure of a rule* $\mapsto_\gamma$. *The reduction* $\rightarrow_{\beta_!} \cup \rightarrow_\gamma$ *satisfies surface factorization if:*

(1) Surface factorization of $\rightarrow_\gamma$*:* $\rightarrow_\gamma^* \subseteq \underset{s}{\rightarrow}_\gamma^* \cdot \underset{\neg s}{\rightarrow}_\gamma^*$

(2) $\mapsto_\gamma$ *is* substitutive*:* $R \mapsto_\gamma R'$ *implies* $R[Q/x] \mapsto_\gamma R'[Q/x]$.

(3) Root linear swap*:* $\underset{\neg s}{\rightarrow}_{\beta_!} \cdot \mapsto_\gamma \subseteq \mapsto_\gamma \cdot \rightarrow_{\beta_!}^*$.

B.3 Restriction to computations

In *Com*, let $\mapsto_\rho$ be a rule and $\rightarrow_\rho$ be its contextual closure. The restriction of reduction to computations preserves $\rightarrow_\rho$, $\underset{s}{\rightarrow}_\rho$, $\underset{\neg s}{\rightarrow}_\rho$, $\underset{w}{\rightarrow}_\rho$, $\underset{\neg w}{\rightarrow}_\rho$ steps. Thus, all properties that hold for $(\Lambda^!, \rightarrow_\rho)$ (e.g. Fact B.1 and Corollary B.2) also hold for $(Com, \rightarrow_\rho)$.

In particular, Proposition 6.5 is immediate consequence of Proposition B.6.

Appendix C. Properties of reduction in $\lambda_©$

We now consider $\lambda_©$, that is $(Com, \rightarrow_©)$. As we have just seen above, the properties we have studied in Appendix B also hold when restricting reduction to computations. Moreover, $\lambda_©$ satisfies also specific properties that do not hold in general, as the following.

Lemma C.1. *Let* $M \in Com$ *and* $M \underset{\neg s}{\rightarrow}_© L$*:* M *is a* id*-redex (resp. a* ι*-redex) if and only if* L *is.*

Proof. If M is a id-redex, this means that $M = (\lambda z.!z)P \underset{\neg s}{\rightarrow}_\gamma (\lambda z.!z)N = L$ where $P \underset{\neg s}{\rightarrow}_\gamma N$, hence L is a id-redex. Moreover, if M is a ι-redex, then $P \neq !V$, hence by Fact B.1 $L \neq !V'$ for any V'. Thus L is a ι-redex.

Let us prove that if L is a id-redex, so is M. Since $L = (\lambda z.!z)N$, by Fact B.1, M is an application; we have the following cases:

(i) either $M = (\lambda z.P)N \underset{\neg s}{\rightarrow}_\gamma (\lambda z.!z)N$ where $P \underset{\neg s}{\rightarrow}_\gamma !z$;

(ii) or $M = (\lambda z.!z)P \underset{\neg s}{\rightarrow}_\gamma (\lambda z.!z)N$ where $P \underset{\neg s}{\rightarrow}_\gamma N$.

Case *(i.)* is impossible because otherwise $P = !V$ for some value V, by Fact B.1, such that $V \rightarrow_\gamma z$, but such a V does not exist. Therefore, we are necessarily in case *(ii.)*, i.e., M is a id-redex. Moreover, if L is a ι-redex, then $N \neq !V$, hence N is an application, and so is P by Fact B.1. □

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

Note that Lemma C.1 is false if we replace the hypothesis $M \underset{\neg s}{\to}_{©} L$ with $M \underset{\neg w}{\to}_{©} L$. Indeed, consider $M = (\lambda x.(\lambda y.!y)!x)N \underset{\neg w}{\to}_{©} (\lambda x.!x)N = L$: L is a id-redex but M is not.

Lemma C.2. *There is no $M \in Com$ such that $M \to_\iota !x$.*

Proof. By induction on $M \in Com$, proving that for every M such that $M \to_\iota N$, $N \neq !x$. □

C.1 Postponement of ι, Technical Lemmas

Lemma C.3 (ι vs. β_1)**.** *$M \to_\iota L \to_{\beta_1} N$ implies $M \to^*_{\beta_1} \cdot \to^=_\iota N$*

Proof. We set the notation $\Rightarrow_{①} := \to^*_{\beta_1} \cdot \to^=_\iota$.

The proof is by induction on L. Cases:

- $L = (\lambda x.!x)!V \mapsto_{\beta_1} !V = N$. Then, there are two possibilities.
  Either $M = (\lambda z.!z)L \mapsto_\iota L$ then

$$M = (\lambda z.!z)((\lambda x.!x)!V) \to_{\beta_1} (\lambda z.!z)!V \to_{\beta_1} !V = N.$$

  Or $M = (\lambda x.!x)!W$ with $!W \to_\iota !V$, and then

$$M = (\lambda x.!x)!W \to_{\beta_1} !W \to_\iota !V = N$$

  The case $M = (\lambda x.P)!V$ with $P \to_\iota !x$ is impossible by Lemma C.2.
- $L = !\lambda x.P \to_{\beta_1} !\lambda x.P' = N$ where $P \to_{\beta_1} P'$. In this case note that necessarily $M = !\lambda x.Q$ where $Q \to_\iota P$. Otherwise, it should have been $M = (\lambda z.!z)L \mapsto_\iota L$, but the ι step is impossible because $L = !\lambda x.P$. By *i.h.*, since $Q \to_\iota P \to_{\beta_1} P'$, we have $Q \Rightarrow_{①} P$; hence $M \Rightarrow_{①} N$.
- $L = VP \to_{\beta_1} V'P' = N$ where $\to_{\beta_1}$ is not root steps, that is:
  (a) either $V \to_{\beta_1} V'$ and $P = P'$;
  (b) or $V = V'$ and $P \to_{\beta_1} P'$.

  By Fact 6.2, M, L, N are applications. So, M has the following shape:
  (1) $M = VQ$ with $Q \to_\iota P$
  (2) $M = WP$ with $W \to_\iota V$
  (3) $M = (\lambda x.!x)(VP) \mapsto_\iota VP = L$

  We distinguish six sub-cases:

  Case a1 We have $M = VQ \to_{\beta_1} V'Q \to_\iota V'P = N$, switching the steps $\to_\iota$ and $\to_{\beta_1}$, directly.

  Case b1 $Q \to_\iota P \to_{\beta_1} P'$, then the thesis follows by i.h.: $Q \Rightarrow_{①} P'$ and then $M = VQ \Rightarrow_{①} VP' = N$.

  Case a2 $W \to_\iota V \to_{\beta_1} V'$, then the thesis follows by i.h.: $W \Rightarrow_{①} V'$ and then $M = WP \Rightarrow_{①} V'P = N$.

  Case b2 We have $M = WP \to_{\beta_1} WP' \to_\iota VP' = N$, switching the steps $\to_\iota$ and $\to_{\beta_1}$, directly.

  Case a3 $M = (\lambda x.!x)(VP) \to_{\beta_1} (\lambda x.!x)(V'P) \mapsto_\iota V'P = N$.

  Case b3 $M = (\lambda x.!x)(VP) \to_{\beta_1} (\lambda x.!x)(VP') \mapsto_\iota VP' = N$. □

Lemma C.4 (ι vs. β_2)**.** *$M \to_\iota L \to_{\beta_2} N$ implies $\to^*_{\beta_1} \cdot \to^=_{\beta_2} \cdot \to^*_{\beta_1} \cdot \to^*_\iota N$*

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

Proof. We set the notation $\Rightarrow_{②} ::= \to^*_{\beta_1} \cdot \to^=_{\beta_2} \cdot \to^*_{\beta_1} \cdot \to^*_{\iota}$. The proof is by induction on L. Note that if $L = !x$ there is no β_2 reduction from it, so this case is not in the scope of the induction. Cases:

- $L = (\lambda x.P')!V' \mapsto_{\beta_2} P'[V'/x] = N$. Then, there are three possibilities.
  (i) $M = (\lambda x.P)!V'$ with $P \to_\iota P'$
  (ii) $M = (\lambda x.P')!V$ with $V \to_\iota V'$
  (iii) $M = (\lambda x.!x)L \mapsto_\iota L$

  So by analyzing each of the three cases above, we can postpone the $\to_\iota$ step as follows:

Case i $M = (\lambda x.P)!V \mapsto_{\beta_2} P[V/x] \to_\iota P'[V/x]$ where the last reduction step is possible by Fact A.1.1. Note that $P \neq !x$ otherwise would not possible $P \to_\iota P'$, as assumed.

Case ii $M = (\lambda x.P)!V \mapsto_{\beta_2} P[V/x] \to^*_\iota P[V'/x]$ where the last reduction step is possible by Fact A.1.2.

Case iii $M = (\lambda x.!x)L \mapsto_{\beta_2} (\lambda x.!x)N \mapsto_\iota N$

- $L = !\lambda x.P \to_{\beta_2} !\lambda x.P' = N$ where $P \to_{\beta_2} P'$. In this case, note that M has necessary the shape $!\lambda x.Q$ where $Q \to_\iota P$. Otherwise, M should have been $(\lambda z.!z)L \mapsto_\iota L$, but it is impossible by definition of $\mapsto_\iota$ since $L = !\lambda x.P$. The thesis follows by induction, since we have $Q \to_\iota P \to_{\beta_2} P'$, $Q \Rightarrow_{②} P$.
- $L = VP \to_{\beta_2} V'P' = N$ where $\to_{\beta_2}$ is not root steps, that is:
  (a) either $V \to_{\beta_2} V'$ and $P = P'$;
  (b) or $V = V'$ and $P \to_{\beta_2} P'$.

  By Fact 6.2, M, L, N are applications. So, M has the following shape:
  (1) $M = VQ$ with $Q \to_\iota P$
  (2) $M = WP$ with $W \to_\iota V$
  (3) $M = (\lambda x.!x)(VP) \mapsto_\iota VP = L$

  We distinguish six subcases:

Case a1 We have $M = VQ \to_{\beta_2} V'Q \to_\iota V'P = N$, switching the steps $\to_\iota$ and $\to_{\beta_2}$, directly.

Case b1 $Q \to_\iota P \to_{\beta_2} P'$, then the thesis follows by i.h., that is: $Q \Rightarrow_{②} P'$ and then $M = VQ \Rightarrow_{②} VP' = N$.

Case a2 $W \to_\iota V \to_{\beta_2} V'$, then the thesis follows by i.h., that is: $W \Rightarrow_{②} V'$ and then $M = WP \Rightarrow_{②} V'P = N$.

Case b2 We have $M = WP \to_{\beta_2} WP' \to_\iota VP' = N$, switching the steps $\to_\iota$ and $\to_{\beta_2}$, directly.

Case a3 $M = (\lambda x.!x)(VP) \to_{\beta_2} (\lambda x.!x)(V'P) \mapsto_\iota V'P = N$

Case b3 $M = (\lambda x.!x)(VP) \to_{\beta_2} (\lambda x.!x)(VP') \mapsto_\iota VP' = N$

□

Lemma C.5 (ι vs. σ). *$M \to_\iota L \to_\sigma N$ implies $M \to^*_\sigma \cdot \to^=_\iota N$*

Proof. We set the notation $\Rightarrow_{③} ::= (\to_\sigma \cup \to_{\beta_1})^* \cdot \to^=_\iota$.
The proof is by induction on L. We distinguishing if the last $L \to_\sigma N$ is a root step or not.
If $L \mapsto_\sigma N$, then $L = V((\lambda x.P)Q)$ and $N = (\lambda x.VP)Q$. Thus, there are seven cases for M:

(i) $M = (\lambda z.!z)(V((\lambda x.P)Q))$;

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

(ii) $M = V((\lambda z.!z)((\lambda x.P)Q))$;
(iii) $M = V((\lambda x.(\lambda z.!z)P)Q)$;
(iv) $M = (V((\lambda x.P)((\lambda z.!z)Q)))$;
(v) $M = W((\lambda x.P)Q)$ with $W \to_\iota V$ and $\to_\iota$ is not a root step;
(vi) $M = V((\lambda x.R)Q)$ with $R \to_\iota P$ and $\to_\iota$ is not a root step;
(vii) $M = V((\lambda x.P)R)$ with $R \to_\iota Q$ and $\to_\iota$ is not a root step.

So by analyzing each of the seven cases above, we can postpone the $\to_\iota$ step as follows:

Case i $M = (\lambda z.!z)(V((\lambda x.P)Q)) \to_\sigma (\lambda z.!z)((\lambda x.VP)Q) \mapsto_\iota (\lambda x.VP)Q = N$.

Case ii $M = V((\lambda z.!z)((\lambda x.P)Q)) \to_\sigma V((\lambda x.(\lambda z.!z)P)Q) \to_\sigma (\lambda x.V((\lambda z.!z)P))Q \to_\gamma (\lambda x.VP)Q = N$ where in the last step γ is ι or β_1 depending on whether P is of the form $!W$ or not.

Case iii $M = V((\lambda x.(\lambda z.!z)P)Q) \to_\sigma (\lambda x.V((\lambda z.!z)P))Q \to_\gamma (\lambda x.VP)Q = N$ where in the last step γ is ι or β_1 depending on whether P is of the form $!W$ or not.

Case iv $M = V((\lambda x.P)((\lambda z.!z)Q)) \underset{s}{\to}_\sigma (\lambda x.VP)((\lambda z.!z)Q) \to_\gamma (\lambda x.VP)Q = N$ where in the last step γ is ι or β_1 depending on whether Q is of the form $!W$ or not.

Case v $M = W((\lambda x.P)Q) \mapsto_\sigma (\lambda x.WP)Q \to_\iota (\lambda x.VP)Q = N$.

Case vi $M = V((\lambda x.R)Q) \mapsto_\sigma (\lambda x.VR)Q \to_\iota (\lambda x.VP)Q = N$.

Case vii $M = V((\lambda x.P)R) \mapsto_\sigma (\lambda x.VP)R \to_\iota (\lambda x.VP)Q = N$.

Consider the case $L = !\lambda x.P \to_\sigma !\lambda x.P' = N$ with $P \to_\sigma P'$. So, note that M has necessary the shape $!\lambda x.Q$ where $Q \to_\iota P$. Otherwise, M should have been $(\lambda z.!z)L \mapsto_\iota L$, but it is impossible by definition of $\mapsto_\iota$ since $L = !\lambda x.P$. The thesis follows by *i.h.*, since $Q \to_\iota P \to_\sigma P'$, $Q \Rightarrow_{③} P$.

The last case to consider is $L = VP \to_\sigma V'P' = N$ where $\to_\sigma$ is not root steps, that is:

(a) either $V \to_\sigma V'$ and $P = P'$;
(b) or $V = V'$ and $P \to_\sigma P'$.

By Fact 6.2, M, L, N are applications. So, M has one of the following shapes:

(1) $M = (\lambda z.!z)L \mapsto_\iota VP = L$;
(2) $M = WP$ with $W \to_\iota V$;
(3) $M = VQ$ with $Q \to_\iota P$.

Hence, combining Points a and b with Points 1 to 3, we distinguish six subcases:

Case a1 $M = (\lambda x.!x)(VP) \to_\sigma (\lambda x.!x)(V'P) \mapsto_\iota V'P = N$.

Case b1 $M = (\lambda x.!x)(VP) \to_\sigma (\lambda x.!x)(VP') \mapsto_\iota VP' = N$.

Case a2 $W \to_\iota V \to_\sigma V'$, then the thesis follows by i.h: $W \Rightarrow_{③} V'$ and then $M = WP \Rightarrow_{③} V'P = N$.

Case b2 We have $M = WP \to_\sigma WP' \to_\iota VP' = N$, switching the steps $\to_\iota$ and $\to_\sigma$, directly.

Case a3 We have $M = VQ \to_\sigma V'Q \to_\iota V'P = N$, switching the steps $\to_\iota$ and $\to_\sigma$, directly.

Case b3 $Q \to_\iota P \to_\sigma P'$, then the thesis follows by i.h.: $Q \Rightarrow_{③} P'$ and then $M = VQ \Rightarrow_{③} VP' = N$. □

Appendix D. Normalization of $\lambda_{©}$

Lemma 8.3. *Let* $\mathsf{e} \in \{\mathsf{w}, \mathsf{s}\}$. *If* $M \underset{\neg e}{\rightarrow}_{\beta_c\sigma} N$ *then:* M *is* $\underset{e}{\rightarrow}_{\beta_c\sigma}$*-normal if and only if* N *is* $\underset{e}{\rightarrow}_{\beta_c\sigma}$*-normal.*

Proof. By easy induction on the shape of M. Observe that M and N have the same shape because the step $M \underset{\neg e}{\rightarrow} N$ is not a root step.

- $M = !V$ and $N = !V'$: the claim is trivial.
- $M = VP$ and $N = V'P'$. Either $V \underset{\neg e}{\rightarrow} V'$ (and $P' = P$) or $P \underset{\neg e}{\rightarrow} P'$ (and $V = V'$). Assume $M = VP$ is e-normal. Since V and P are e-normal, by *i.h.* so are V' and P'. Moreover, N is not a redex, by Corollary B.2, so N is normal. Assuming $N = V'P'$ normal is similar. □

Fact D.1. *The reduction* ι *is quasi-diamond. Therefore, if* $S \rightarrow_{\iota}^{k} N$ *where* N *is* ι*-normal, then any maximal* ι*-sequence from* S *ends in* N*, in* k *steps.*

Lemma 8.1. *Assume* $M \rightarrow_{\iota} N$.

(1) *M is β_c-normal if and only if N is β_c-normal.*
(2) *If M is σ-normal, so is N.*

Proof. Easy to prove by induction on the structure of terms. □

Fact D.2 (Shape preservation of ι-sequences)**.** *If* S *is not an* ι*-redex, and* $S \rightarrow_{\iota}^{k} N$ *then no term in the sequence is an* ι*-redex, and so* N *has the same shape as* S*:*

(1) *$S = !(\lambda x.Q)$ implies $N = !(\lambda x.N_Q)$. Moreover, $Q \rightarrow_{\iota}^{k} N_Q$.*
(2) *$S = xP$ implies $N = xN_P$. Moreover, $P \rightarrow_{\iota}^{k} N_P$.*
(3) *$S = (\lambda x.Q)P$ implies $N = (\lambda x.N_Q)N_P$. Moreover, $Q \rightarrow_{\iota}^{k_1} N_Q$, $P \rightarrow_{\iota}^{k_2} N_P$ and $k = k_1 + k_2$.*

Lemma 8.5. *Assume* $M \rightarrow_{\iota}^{k} N$*, where* $k > 0$*, and* N *is* $\sigma\iota$*-normal. If* M *is not* σ*-normal, then there exist* M' *and* N' *such that either* $M \rightarrow_{\sigma} M' \rightarrow_{\iota} N' \rightarrow_{\iota}^{k-1} N$ *or* $M \rightarrow_{\sigma} M' \rightarrow_{\beta_c} N' \rightarrow_{\iota}^{k-1} N$.

Proof. The proof is by induction on M.
Assume M is a is a σ-redex, i.e. $M = V((\lambda x.P)L)$ where V is an abstraction.

- If M is also an ι-redex, then $V = \mathbf{I}$, and:
  (1) If $P = !U$, then $M = \mathbf{I}((\lambda x.!U)L) \rightarrow_{\iota} (\lambda x.!U)L \rightarrow_{\iota}^{k-1} N$ and $M \rightarrow_{\sigma} (\lambda x.\mathbf{I}!U)L \rightarrow_{\beta} (\lambda x.!U)L$.
  (2) If $P \neq !U$, then $M = \mathbf{I}((\lambda x.P)L) \rightarrow_{\iota} (\lambda x.P)L \rightarrow_{\iota}^{k-1} N$ and $M \rightarrow_{\sigma} (\lambda x.\mathbf{I}P)L \rightarrow_{\iota} (\lambda x.P)L$.

- Otherwise, if $M = V(\mathbf{I}L)$, then $M \rightarrow_{\iota} VL \rightarrow_{\iota}^{k-1} N$ and $M \rightarrow_{\sigma} (\lambda z.V!z)L \rightarrow_{\beta} VL$.
- No other case is possible because $M = V((\lambda x.P)L)$ not an ι-redex implies (by Fact D.2) that $N = N_1N_2$, with $N_1 \in \mathtt{Abs}$. If $(\lambda x.P) \neq \mathbf{I}$, then N would be a σ-redex because again $N_2 = N_2'N_2''$ with $N_2' \in \mathtt{Abs}$ (by Fact D.2).

Assume M is not a σ-redex. We examine the shape of S and use Fact D.2.

- $M = !\lambda x.Q$. We have $N = !\lambda x.N_Q$, $Q \rightarrow_{\iota}^{k} N_Q$, where Q is not σ-normal, and N_Q is $\sigma\iota$-normal. We conclude by *i.h.*

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

- $M = \mathbf{I}P$. We have $\mathbf{I}P \to_\iota P \to_\iota^{k-1} N$. Since P is not σ-normal, we use the *i.h.* on $P \to_\iota^{k-1} N$, obtaining that $P \to_\iota N' \to_\iota^{k-2} N$ and $P \to_\sigma P' \to_{\beta_{c\iota}} N'$. Therefore, also $\mathbf{I}P \to_\sigma \mathbf{I}P' \to_{\beta_{c\iota}} \mathbf{I}N' \underset{\mathsf{i}}{\to} N' \underset{\mathsf{i}}{\to}^{k-1} N$.
- $M = (\lambda x.Q)P$ (M is not an ι-redex). We have $N = (\lambda x.N_Q)N_P$, where N_Q and N_P are $\sigma\iota$-normal. We distinguish two cases.
  - If Q is not σ-normal, we note that $Q \to_\iota^{k_1} N_Q$, and conclude by *i.h.* Indeed, by *i.h.*, we obtain that $Q \to_\iota N'_Q \to_\iota^{k_1-1} N_Q$ and $Q \to_\sigma Q' \to_{\beta_{c\iota}} N'_Q$. So, $(\lambda x.Q)P \to_\iota (\lambda x.N'_Q)P \to_\iota^{k_1-1} (\lambda x.N_Q)P \to_\iota^{k_2} (\lambda x.N_Q)N_P$ and $(\lambda x.Q)P \to_\sigma (\lambda x.Q')P \to_{\beta_{c\iota}} (\lambda x.N'_Q)P$.
  - If P is not σ-normal, we note that $P \to_\iota^{k_2} N_P$, and conclude by *i.h.* □

Proposition 8.6. (Termination of $\sigma\mathsf{id}$). *Reduction* $\to_{\sigma\mathsf{id}} = (\to_\sigma \cup \to_{\mathsf{id}})$ *is strongly normalizing.*

Proof. We define two sizes $\mathsf{s}(M)$ and $\mathsf{s}_\sigma(M)$ for any term M.

$$
\begin{aligned}
\mathsf{s}(x) &= 1 & \mathsf{s}_\sigma(x) &= 1 \\
\mathsf{s}(\lambda x.M) &= \mathsf{s}(M) + 1 & \mathsf{s}_\sigma(\lambda x.M) &= \mathsf{s}_\sigma(M) + \mathsf{s}(M) \\
\mathsf{s}(VM) &= \mathsf{s}(V) + \mathsf{s}(M) & \mathsf{s}_\sigma(VM) &= \mathsf{s}_\sigma(V) + \mathsf{s}_\sigma(M) + 2\mathsf{s}(V)\mathsf{s}(M) \\
\mathsf{s}(!M) &= \mathsf{s}(M) & \mathsf{s}_\sigma(!M) &= \mathsf{s}_\sigma(M)
\end{aligned}
$$

Note that $\mathsf{s}(M) > 0$ and $\mathsf{s}_\sigma(M) > 0$ for any term M. It easy to check that if $M \to_{\mathsf{id}} \cup \to_\sigma N$, then $(\mathsf{s}(N), \mathsf{s}_\sigma(N)) <_{\text{lex}} (\mathsf{s}(M), \mathsf{s}_\sigma(M))$, where $<_{\text{lex}}$ is the strict lexicographical order on $\mathbb{N}^2$. Indeed, if $M \to_{\mathsf{id}} N$, then $\mathsf{s}(M) > \mathsf{s}(N)$; and if $M \to_\sigma N$ then $\mathsf{s}(M) = \mathsf{s}(N)$ and $\mathsf{s}_\sigma(M) > \mathsf{s}_\sigma(N)$. The proof is by straightforward induction on M. We show only the root cases, the other cases follow from the *i.h.* immediately.

- If $(\lambda x.!x)M \mapsto_{\mathsf{id}} M$ then $\mathsf{s}((\lambda x.!x)M) = \mathsf{s}(\lambda x.!x) + \mathsf{s}(M) > \mathsf{s}(M)$.
- If $(\lambda x.M)((\lambda y.N)L) \mapsto_\sigma (\lambda y.(\lambda x.M)N)L$ then clearly $\mathsf{s}((\lambda x.M)((\lambda y.N)L)) = \mathsf{s}((\lambda y.(\lambda x.M)N)L)$ and

$$
\begin{aligned}
&\mathsf{s}_\sigma((\lambda x.M)((\lambda y.N)L)) \\
&= \mathsf{s}_\sigma(\lambda x.M) + \mathsf{s}_\sigma((\lambda y.N)L) + 2\mathsf{s}(\lambda x.M)\mathsf{s}((\lambda y.N)L) \\
&= \mathsf{s}_\sigma(\lambda x.M) + \mathsf{s}_\sigma(\lambda y.N) + \mathsf{s}_\sigma(L) + 2\mathsf{s}(\lambda y.N)\mathsf{s}(L) + 2\mathsf{s}(\lambda x.M)\mathsf{s}((\lambda y.N)L) \\
&= \mathsf{s}_\sigma(\lambda x.M) + \mathsf{s}_\sigma(N) + \mathsf{s}(N) + \mathsf{s}_\sigma(L) + 2\mathsf{s}(N)\mathsf{s}(L) + 2\mathsf{s}(L) + 2\mathsf{s}(\lambda x.M)\mathsf{s}(\lambda y.N) + 2\mathsf{s}(\lambda x.M)\mathsf{s}(L) \\
&= \mathsf{s}_\sigma(\lambda x.M) + \mathsf{s}_\sigma(N) + \mathsf{s}(N) + \mathsf{s}_\sigma(L) + 2\mathsf{s}(N)\mathsf{s}(L) + 2\mathsf{s}(L) + \underline{2\mathsf{s}(\lambda x.M)} + 2\mathsf{s}(\lambda x.M)\mathsf{s}(N) + 2\mathsf{s}(\lambda x.M)\mathsf{s}(L) \\
&> \mathsf{s}_\sigma(\lambda x.M) + \mathsf{s}_\sigma(N) + 2\mathsf{s}(\lambda x.M)\mathsf{s}(N) + \underline{\mathsf{s}(\lambda x.M)} + \mathsf{s}(N) + \mathsf{s}_\sigma(L) + 2\mathsf{s}(\lambda x.M)\mathsf{s}(L) + 2\mathsf{s}(N)\mathsf{s}(L) + 2\mathsf{s}(L) \\
&= \mathsf{s}_\sigma(\lambda x.M) + \mathsf{s}_\sigma(N) + 2\mathsf{s}(\lambda x.M)\mathsf{s}(N) + \mathsf{s}(\lambda x.M) + \mathsf{s}(N) + \mathsf{s}_\sigma(L) + 2\mathsf{s}((\lambda x.M)N)\mathsf{s}(L) + 2\mathsf{s}(L) \\
&= \mathsf{s}_\sigma((\lambda x.M)N) + \mathsf{s}((\lambda x.M)N) + \mathsf{s}_\sigma(L) + 2\mathsf{s}(\lambda y.(\lambda x.M)N)\mathsf{s}(L) \\
&= \mathsf{s}_\sigma(\lambda y.(\lambda x.M)N) + \mathsf{s}_\sigma(L) + 2\mathsf{s}(\lambda y.(\lambda x.M)N)\mathsf{s}(L) \\
&= \mathsf{s}_\sigma((\lambda y.(\lambda x.M)N)L)
\end{aligned}
$$

□

Appendix E. Notational Equivalence between $\lambda_©$ and de' Liguoro and Treglia (2020)

Here we recall the calculus introduced in de' Liguoro and Treglia (2020) (see also our Section 1), henceforth denoted by $\lambda_\star$, and formalize that $\lambda_\star$ is *isomorphic* to the computational core $\lambda_©$. In other words, $\lambda_©$ (as defined in Section 3) is nothing but another presentation of $\lambda_\star$, with just a different notation.

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

First, we recall the syntax of $\lambda_\star$, with *unit* and $\star$ operators:

$$Val^\star: \quad V, W ::= x \mid \lambda x.M \qquad \text{(values)}$$

$$Com^\star: \quad M, N ::= unit\ V \mid M \star V \qquad \text{(computations)}$$

We set $Term^\star := Val^\star \cup Com^\star$. Contexts are defined in Section 1. Reductions in $\lambda_\star$ are the contextual closures of the rules (1), (2), and (3) on p. 935, oriented left-to-right, giving rise to reductions $\to_{\beta_c}$, $\to_{\mathsf{id}}$, and $\to_\sigma$, respectively. We set $\to_{©} = \to_{\beta_c} \cup \to_{\mathsf{id}} \cup \to_\sigma$.

Consider the translation $(\cdot)^\bullet$ from $Term^\star$ to $Term$, and conversely, the translation $(\cdot)^\circ$ from $Term$ to $Term^\star$:

Table E1. Translations between $\lambda_©$ and $\lambda_\star$

	$(\cdot)^\bullet : Term^\star \to Term$	$(\cdot)^\circ : Term \to Term^\star$
variables	$(x)^\bullet = x$	$(x)^\circ = x$
abstraction	$(\lambda x.P)^\bullet = \lambda x.(P)^\bullet$	$(\lambda x.M)^\circ = \lambda x.(M)^\circ$
returned values	$(unit\ V)^\bullet = !(V^\bullet)$	$(!V)^\circ = unit(V^\circ)$
bind/application	$(P \star V)^\bullet = V^\bullet P^\bullet$	$(VM)^\circ = M^\circ \star V^\circ$

Essentially, translating terms of $\lambda_\star$ in terms of the computational core $\lambda_©$ rewrites *unit* as !, and reverts the order in $\star$.

Proposition E.1. *The following holds:*

(1) $(M^\circ)^\bullet = M$ *for every term M in Term;*
(2) $(P^\bullet)^\circ = P$ *for every term P in* $Term^\star$*;*
(3) for any $\gamma \in \{\beta_c, \mathsf{id}, \sigma, ©\}$*, if* $M \to_\gamma N$ *in* $\lambda_©$*, then* $M^\circ \to_\gamma N^\circ$ *in* $\lambda_\star$*;*
(4) for any $\gamma \in \{\beta_c, \mathsf{id}, \sigma, ©\}$*, if* $P \to_\gamma Q$ *in* $\lambda_\star$*, then* $P^\bullet \to_\gamma Q^\bullet$ *in* $\lambda_©$*.*

Proof. Immediate by definition unfolding. □

Appendix F. Computational versus Call-by-Value

Here we formalize the relation between a fragment of the computational core $\lambda_©$ and Plotkin's call-by-value (CbV, for short) λ-calculus (Plotkin 1975).

The fragment of $\lambda_©$ that includes just β_c as only reduction rule, i.e. $(Com, \to_{\beta_c})$, is also isomorphic to the *kernel* of Plotkin's CbV λ-calculus, which is the restriction of $\to_{\beta_v}$ (see Section 2.2) to the set of terms $Com^v \subseteq \Lambda$ defined as follows (note that $Val^v \subseteq Com^v$).

$$Val^v: \quad V, W ::= x \mid \lambda x.M$$

$$Com^v: \quad M, N, L ::= V \mid VM$$

To establish such an isomorphism, consider the translation $(\cdot)^\bullet$ from $Term$ to Com^v, and conversely, the translation $(\cdot)^\circ$ from Com^v to $Term$.

Essentially, the translation $(\cdot)^\bullet$ simply forgets the operator !, and dually the translation $(\cdot)^\circ$ adds a ! in front of each value that is not in the functional position of an application. These two translations form an isomorphism.

Proposition F.1.

(1) $(M^\circ)^\bullet = M$*, for every term* $M \in Com^v$*;*

https://doi.org/10.1017/S0960129522000433 Published online by Cambridge University Press

Table F1. Translations between the computational core $\lambda_{©}$ and the kernel of the Call-by-Value λ-calculus

	$(\cdot)^{\bullet}: \mathit{Term} \to \mathit{Com}^{v}$	$(\cdot)^{\circ}: \mathit{Com}^{v} \to \mathit{Term}$
variables	$(x)^{\bullet} = x$	$(x)^{\circ} = !x$
abstraction	$(\lambda x.P)^{\bullet} = \lambda x.(P)^{\bullet}$	$(\lambda x.M)^{\circ} = !\lambda x.(M)^{\circ}$
returned values	$(!V)^{\bullet} = V^{\bullet}$	
bind/application	$(VP)^{\bullet} = V^{\bullet}P^{\bullet}$	$(VP)^{\circ} = \begin{cases} xP^{\circ} & \text{if } V = x \\ (\lambda x.Q^{\circ})P^{\circ} & \text{if } V = \lambda x.Q \end{cases}$

(2) $(P^{\bullet})^{\circ} = P$, *for every term* $P \in \mathit{Term}$;
(3) $M \to_{\beta_v} N$ *implies* $M^{\circ} \to_{\beta_c} N^{\circ}$, *for every terms* $M, N \in \mathit{Com}^{v}$;
(4) $P \to_{\beta_c} Q$ *implies* $P^{\bullet} \to_{\beta_v} Q^{\bullet}$, *for every terms* $P, Q \in \mathit{Term}$.

Proof. Immediate by definition unfolding. □

Observe also that the restriction of *weak context* to Com^{v} give exactly the grammar defined in Section 3, and this for all three weak contexts (L, R, W), which all collapse in the same shape. Thus, Proposition F.1 also holds when $\to_{\beta_v}$ and $\to_{\beta_c}$ are replaced by $\underset{\mathsf{W}}{\to}_{\beta_v}$ and $\underset{\mathsf{W}}{\to}_{\beta_c}$, respectively. As a consequence, since $\underset{\mathsf{W}}{\to}_{\beta_v}$ is deterministic in Com^{v} and $\to_{\beta_v}$ weak factorizes (Theorem 2.11.1),

Fact F.2 (Properties of β_c and its *weak* restriction). *In* $\lambda_{©}$*:*

- *reduction* $\underset{\mathsf{W}}{\to}_{\beta_c}$ *is deterministic;*
- *reduction* $\to_{\beta_c}$ *satisfies weak factorization:* $\to^{*}_{\beta_c} \subseteq \underset{\mathsf{W}}{\to}_{\beta_c}{}^{*} \cdot \underset{\neg\mathsf{W}}{\to}_{\beta_c}{}^{*}$.

Call-by-Value versus its Kernel. The CbV kernel – and so $(\mathit{Com}, \to_{\beta_c})$ – is as expressive as the CbV λ-calculus, as we discuss below. This result was already shown by Accattoli (2015).[1]

With respect to its *kernel*, Plotkin's CbV λ-calculus is more liberal in that application is unrestricted (left-hand side need not be a value). The kernel has the same expressive power as CbV calculus because the full syntax of Plotkin's CbV can be encoded into the restricted one the CbV kernel and because the CbV kernel can simulate every reduction sequence of Plotkin's full CbV.

Formally, consider the translation $(\cdot)^{\dagger}$ from Plotkin's CbV λ-calculus to its kernel.

$$(x)^{\dagger} = x \qquad (\lambda x.P)^{\dagger} = \lambda x.P^{\dagger} \qquad (PQ)^{\dagger} = \begin{cases} P^{\dagger}Q^{\dagger} & \text{if } P \text{ is a value;} \\ (\lambda x.xQ^{\dagger})P^{\dagger} & \text{otherwise.} \end{cases}$$

Proposition F.3 (Simulation of the CbV λ-calculus into its kernel). *For every term P in Plotkin's CbV* λ*-calculus, if* $P \to_{\beta_v} Q$ *then* $P^{\dagger} \to^{+}_{\beta_v} Q^{\dagger}$ *and* $P^{\dagger\circ} \to^{+}_{\beta_c} Q^{\dagger\circ}$.

Cite this article: Faggian C, Guerrieri G, de' Liguoro U and Treglia R (2022). On reduction and normalization in the computational core. *Mathematical Structures in Computer Science* **32**, 934–981. https://doi.org/10.1017/S0960129522000433