Normal Form Bisimulations for Delimited-Control Operators

Dariusz Biernacki, Sergueï Lenglet

Normal Form Bisimulations for Delimited-Control Operators

Dariusz Biernacki and Sergueï Lenglet

University of Wrocław

Abstract. We define a notion of normal form bisimilarity for the untyped call-by-value λ-calculus extended with the delimited-control operators shift and reset. Normal form bisimilarities are simple, easy-to-use behavioral equivalences which relate terms without having to test them within all contexts (like contextual equivalence), or by applying them to function arguments (like applicative bisimilarity). We prove that the normal form bisimilarity for shift and reset is sound but not complete w.r.t. contextual equivalence and we define up-to techniques that aim at simplifying bisimulation proofs. Finally, we illustrate the simplicity of the techniques we develop by proving several equivalences on terms.

1 Introduction

Morris-style contextual equivalence [16] is usually considered as the most natural behavioral equivalence for functional languages based on λ-calculi. Roughly, two terms are equivalent if we can exchange one for the other in a bigger program without affecting its behavior (i.e., whether it terminates or not). The quantification over program contexts makes contextual equivalence hard to use in practice and, therefore, it is common to look for easier-to-use behavioral equivalences, such as *bisimilarities*.

Several kinds of bisimilarity relations have been defined so far, such as *applicative bisimilarity* [1], *normal form bisimilarity* [12] (originally defined in [17], where it was called *open bisimilarity*), and *environmental bisimilarity* [18]. Applicative and environmental bisimilarities usually compare terms by applying them to function arguments; as a result, we obtain relations which completely characterize contextual equivalence, but still contain a universal quantification over arguments in their definitions. In contrast, normal form bisimilarity does not need such quantification; it equates terms by reducing them to normal form, and by requiring the sub-terms of these normal forms to be bisimilar. Normal form relations are convenient in practice, but they are usually not *complete* w.r.t. contextual equivalence, i.e., there exist contextually equivalent terms that are not normal form bisimilar.

A notion of normal form bisimulation has been defined in various calculi, including the pure λ-calculus [11, 12], the λ-calculus with ambiguous choice [13], the $\lambda\mu$-calculus [14], and the $\lambda\mu\rho$-calculus [20], where normal form bisimilarity completely characterizes contextual equivalence. However, it has not yet been

defined for calculi with *delimited-control* operators, such as shift and reset [7]—programming constructs rapidly gaining currency in the recent years. Unlike abortive control operators (such as call/cc), delimited-control operators allow to delimit access to the current continuation and to compose continuations. The operators shift and reset were introduced as a direct-style realization of the traditional success/failure continuation model of backtracking otherwise expressible only in continuation-passing style [7]. The numerous theoretical and practical applications of shift and reset (see, e.g., [2] for an extensive list) include the seminal result by Filinski showing that a programming language endowed with shift and reset is monadically complete [8].

Up to now, only an applicative bisimilarity has been defined for a calculus with shift and reset [4]. In this paper, we define several notions of normal form bisimilarity for such a calculus, more tractable than contextual equivalence or applicative bisimilarity. We prove they are *sound* w.r.t. contextual equivalence (i.e., included in contextual equivalence), but fail to be complete. We also develop *up-to techniques* that are helpful when proving equivalences with normal form bisimulations.

In Section 2, we define the λ-calculus with delimited control that we use in this paper, and we recall the definition of contextual equivalence of [4] for this calculus. We then define in Section 3 the main notion of normal form bisimilarity and we prove its properties. In Section 4, we refine the definition of normal form bisimilarity to relate more contextually equivalent terms, at the cost of extra complexity in bisimulation proofs. We also propose several up-to techniques which simplify the proofs of equivalence of terms. In Section 5, we illustrate the simplicity of use (compared to applicative bisimilarity) of the notions we define by employing them in the proofs of several equivalences of terms. Section 6 concludes the paper. The extended version of this article [5] contains the congruence proofs of the considered normal form bisimilarities.

2 The Calculus $\lambda_{\mathcal{S}}$

In this section, we present the syntax, reduction semantics, and contextual equivalence for the language $\lambda_{\mathcal{S}}$ studied throughout this article.

2.1 Syntax

The language $\lambda_{\mathcal{S}}$ extends the call-by-value λ-calculus with the delimited-control operators *shift* and *reset* [7]. We assume we have a set of term variables, ranged over by x, y, z, and k. We use the metavariable k for term variables representing a continuation (e.g., when bound with a shift), while x, y, and z stand for any values; we believe such distinction helps to understand examples and reduction rules. The syntax of terms and values is given by the following grammars:

$$\begin{array}{ll} \text{Terms: } & t ::= x \mid \lambda x.t \mid t\,t \mid \mathcal{S}k.t \mid \langle t \rangle \\ \text{Values: } & v ::= \lambda x.t \mid x \end{array}$$

The operator *shift* ($\mathcal{S}k.t$) is a capture operator, the extent of which is determined by the delimiter *reset* ($\langle\cdot\rangle$). A λ-abstraction $\lambda x.t$ binds x in t and a shift construct $\mathcal{S}k.t$ binds k in t; terms are equated up to α-conversion of their bound variables. The set of free variables of t is written $\mathsf{fv}(t)$; a term is *closed* if it does not contain free variables.

We distinguish several kinds of contexts, as follows.

$$\begin{array}{lll} \text{Pure contexts:} & E ::= \Box \mid v\ E \mid E\ t \\ \text{Evaluation contexts:} & F ::= \Box \mid v\ F \mid F\ t \mid \langle F \rangle \\ \text{Contexts:} & C ::= \Box \mid \lambda x.C \mid t\ C \mid C\ t \mid \mathcal{S}k.C \mid \langle C \rangle \end{array}$$

Regular contexts are ranged over by C. The pure evaluation contexts[1] (abbreviated as pure contexts), ranged over by E, represent delimited continuations and can be captured by the shift operator. The call-by-value evaluation contexts, ranged over by F, represent arbitrary continuations and encode the chosen reduction strategy. Filling a context C (respectively E, F) with a term t produces a term, written $C[t]$ (respectively $E[t]$, $F[t]$); the free variables of t may be captured in the process. A context is *closed* if it contains only closed terms.

2.2 Reduction Semantics

Before we present the reduction semantics for $\lambda_{\mathcal{S}}$, let us briefly describe an intuitive semantics of shift and reset by means of an example written in SML, using Filinski's implementation of shift and reset [8].

Example 1. The following function copies a list [3], where the SML expression `shift (fn k => t)` corresponds to $\mathcal{S}k.t$ and `reset (fn () => t)` corresponds to $\langle t \rangle$:

```
fun copy xs =
  let fun visit nil = nil
        | visit (x::xs) = visit (shift (fn k => x :: (k xs)))
  in reset (fn () => visit xs) end
```

This simple function illustrates the main ideas of programming with shift and reset:

- The control delimiter reset delimits continuations. Any control effects occurring in the subsequent calls to function `visit` are local to function `copy`.
- The control operator shift captures delimited continuations. Each but last recursive call to `visit` abstracts the continuation that can be represented as a function `fn v => reset (fn () => visit v)` and binds it to `k`.
- Captured continuations are composed statically. When applied, in the expression `x :: (k xs)`, the captured continuation becomes the current delimited continuation that is isolated from the rest of the program, and in particular from the expression `x ::`, by a control delimiter—witness the control delimiter in the expression `fn v => reset (fn () => visit v)` representing the captured continuation.

[1] This terminology comes from Kameyama (e.g., in [9]).

Formally, the call-by-value reduction semantics of $\lambda_{\mathcal{S}}$ is defined as follows, where $t\{v/x\}$ is the usual capture-avoiding substitution of v for x in t:

$$\begin{array}{ll} (\beta_v) & F[(\lambda x.t)\, v] \to_{\mathrm{v}} F[t\{v/x\}] \\ (\mathit{shift}) & F[\langle E[\mathcal{S}k.t]\rangle] \to_{\mathrm{v}} F[\langle t\{\lambda x.\langle E[x]\rangle/k\}\rangle] \text{ with } x \notin \mathsf{fv}(E) \\ (\mathit{reset}) & F[\langle v\rangle] \to_{\mathrm{v}} F[v] \end{array}$$

The term $(\lambda x.t)\, v$ is the usual call-by-value redex for β-reduction (rule (β_v)). The operator $\mathcal{S}k.t$ captures its surrounding context E up to the dynamically nearest enclosing reset, and substitutes $\lambda x.\langle E[x]\rangle$ for k in t (rule (*shift*)). If a reset is enclosing a value, then it has no purpose as a delimiter for a potential capture, and it can be safely removed (rule (*reset*)). All these reductions may occur within a metalevel context F. The chosen call-by-value evaluation strategy is encoded in the grammar of the evaluation contexts.

Example 2. Let $i = \lambda x.x$ and $\omega = \lambda x.x\, x$. We present the sequence of reductions initiated by $\langle((\mathcal{S}k_1.i\, (k_1\, i))\, \mathcal{S}k_2.\omega)\, (\omega\, \omega)\rangle$. The term $\mathcal{S}k_1.i\, (k_1\, i)$ is within the pure context $E = (\Box\ \mathcal{S}k_2.\omega)\ (\omega\, \omega)$, enclosed in a delimiter $\langle\cdot\rangle$, so E is captured according to rule (*shift*).

$$\langle((\mathcal{S}k_1.i\, (k_1\, i))\, \mathcal{S}k_2.\omega)\, (\omega\, \omega)\rangle \to_{\mathrm{v}} \langle i\, ((\lambda x.\langle(x\, \mathcal{S}k_2.\omega)\, (\omega\, \omega)\rangle)\, i)\rangle$$

The role of reset in $\lambda x.\langle E[x]\rangle$ is more clear after reduction of the β_v-redex $(\lambda x.\langle E[x]\rangle)\, i$.

$$\langle i\, ((\lambda x.\langle(x\, \mathcal{S}k_2.\omega)\, (\omega\, \omega)\rangle)\, i)\rangle \to_{\mathrm{v}} \langle i\, \langle(i\, \mathcal{S}k_2.\omega)\, (\omega\, \omega)\rangle\rangle$$

When the captured context E is reactivated, it is not *merged* with the context $i\ \Box$, but *composed* thanks to the reset enclosing E. As a result, the capture triggered by $\mathcal{S}k_2.\omega$ leaves the term i outside the first enclosing reset untouched.

$$\langle i\, \langle(i\, \mathcal{S}k_2.\omega)\, (\omega\, \omega)\rangle\rangle \to_{\mathrm{v}} \langle i\, \langle\omega\rangle\rangle$$

Because k_2 does not occur in ω, the context $(i\ \Box)\ (\omega\ \omega)$ is discarded when captured by $\mathcal{S}k_2.\omega$. Finally, we remove the useless delimiter $\langle i\, \langle\omega\rangle\rangle \to_{\mathrm{v}} \langle i\, \omega\rangle$ with rule (*reset*), and we then β_v-reduce and remove the last delimiter $\langle i\, \omega\rangle \to_{\mathrm{v}} \langle\omega\rangle \to_{\mathrm{v}} \omega$. Note that while the reduction strategy is call-by-value, some function arguments are not evaluated, like the non-terminating term $\omega\, \omega$ in this example.

There exist terms which are not values and which cannot be reduced any further; these are called *stuck terms*.

Definition 1. *A term t is stuck if t is not a value and $t \not\to_{\mathrm{v}}$.*

For example, the term $E[\mathcal{S}k.t]$ is stuck because there is no enclosing reset; the capture of E by the shift operator cannot be triggered. In fact, stuck terms are easy to characterize.

Lemma 1. *A term t is stuck iff $t = E[\mathcal{S}k.t']$ for some E, k, and t' or $t = F[x\, v]$ for some F, x, and v.*

We call *control stuck terms* terms of the form $E[\mathcal{S}k.t]$ and *open stuck terms* the terms of the form $F[x\ v]$.

Definition 2. *A term t is a normal form, if t is a value or a stuck term.*

We call *redexes* (ranged over by r) terms of the form $(\lambda x.t)\ v$, $\langle E[\mathcal{S}k.t]\rangle$, and $\langle v\rangle$. Thanks to the following unique-decomposition property, the reduction relation $\to_{\mathrm{v}}$ is deterministic.

Lemma 2. *For all terms t, either t is a normal form, or there exist a unique redex r and a unique context F such that $t = F[r]$.*

Finally, we write $\to_{\mathrm{v}}^*$ for the transitive and reflexive closure of $\to_{\mathrm{v}}$, and we define the evaluation relation of $\lambda_{\mathcal{S}}$ as follows.

Definition 3. *We write $t \Downarrow_{\mathrm{v}} t'$ if $t \to_{\mathrm{v}}^* t'$ and $t' \not\to_{\mathrm{v}}$.*

The result of the evaluation of a term, if it exists, is a normal form. If a term t admits an infinite reduction sequence, we say it *diverges*, written $t \Uparrow_{\mathrm{v}}$. In the rest of the article, we use extensively $\Omega = (\lambda x.x\ x)\ (\lambda x.x\ x)$ as an example of such a term.

2.3 Contextual Equivalence

In this paper, we use the same contextual equivalence as in [4], where control stuck terms can be observed. Note that this relation is a bit more discriminative than simply observing termination, as pointed out in [4].

Definition 4. *Let t_0, t_1 be terms. We write $t_0 \approx_c t_1$ if for all C such that $C[t_0]$ and $C[t_1]$ are closed, the following hold:*

- *$C[t_0] \Downarrow_{\mathrm{v}} v_0$ implies $C[t_1] \Downarrow_{\mathrm{v}} v_1$;*
- *$C[t_0] \Downarrow_{\mathrm{v}} t_0'$, where t_0' is control stuck, implies $C[t_1] \Downarrow_{\mathrm{v}} t_1'$, with t_1' control stuck as well;*

and conversely for $C[t_1]$.

We can simplify the proofs of contextual equivalence of terms by relying on the following context lemma [15] for $\lambda_{\mathcal{S}}$ (for a proof see Definition 5 and Section 3.4 in [4]). Instead of testing terms with (free-variables capturing) general contexts, we can simply first close them (using closed values) and then put them within (closed) evaluation contexts.

Lemma 3 (Context Lemma). *We have $t_0 \approx_c t_1$ iff for all closed contexts F and for all substitutions σ (mapping variables to closed values) such that $t_0\sigma$ and $t_1\sigma$ are closed, the following hold:*

- *$F[t_0\sigma] \Downarrow_{\mathrm{v}} v_0$ implies $F[t_1\sigma] \Downarrow_{\mathrm{v}} v_1$;*
- *$F[t_0\sigma] \Downarrow_{\mathrm{v}} t_0'$, where t_0' is control stuck, implies $F[t_1\sigma] \Downarrow_{\mathrm{v}} t_1'$, with t_1' control stuck as well;*

and conversely for $F[t_1\sigma]$.

In the rest of the paper, when proving that terms are contextually equivalent, we implicitly use Lemma 3.

3 Normal Form Bisimilarity

In this section, we discuss a notion of bisimulation based on the evaluation of terms to normal forms. The difficulties are mainly in the handling of control stuck terms and in the definition of the relation on non-pure evaluation contexts. We propose here a first way to deal with control stuck terms, that will be refined in the next section. In any definitions or proofs, we say a variable is *fresh* if it does not occur free in the terms or contexts under consideration.

3.1 Definition

Following Lassen's approach [12], we define a normal form bisimulation where we relate terms by comparing the results of their evaluation (if they exist). As we need to compare terms as well as evaluation contexts, we extend a relation $\mathcal{R}$ on terms to contexts in the following way: we write $F_0 \ \mathcal{R} \ F_1$ if $F_0 = F_0'[\langle E_0 \rangle]$, $F_1 = F_1'[\langle E_1 \rangle]$, $F_0'[x] \ \mathcal{R} \ F_1'[x]$, and $\langle E_0[x] \rangle \ \mathcal{R} \ \langle E_1[x] \rangle$ for a fresh x, or if $F_0 = E_0$, $F_1 = E_1$, and $E_0[x] \ \mathcal{R} \ E_1[x]$ for a fresh x. The rationale behind this definition is explained later. Following [12], we define the application $v \star y$ as $x\ y$ if $v = x$, and as $t\{y/x\}$ if $v = \lambda x.t$. Finally, given a relation $\mathcal{R}$ on terms, we write $\mathcal{R}^{-1}$ for its inverse, and we inductively define a relation $\mathcal{R}^{\mathsf{NF}}$ on normal forms as follows:

$$\frac{v_0 \star x \ \mathcal{R} \ v_1 \star x \qquad x \text{ fresh}}{v_0 \ \mathcal{R}^{\mathsf{NF}} \ v_1} \qquad \frac{E_0 \ \mathcal{R} \ E_1 \qquad \langle t_0 \rangle \ \mathcal{R} \ \langle t_1 \rangle}{E_0[\mathcal{S}k.t_0] \ \mathcal{R}^{\mathsf{NF}} \ E_1[\mathcal{S}k.t_1]} \qquad \frac{F_0 \ \mathcal{R} \ F_1 \qquad v_0 \ \mathcal{R}^{\mathsf{NF}} \ v_1}{F_0[x\ v_0] \ \mathcal{R}^{\mathsf{NF}} \ F_1[x\ v_1]}$$

Definition 5. *A relation $\mathcal{R}$ on terms is a normal form simulation if $t_0 \ \mathcal{R} \ t_1$ and $t_0 \Downarrow_{\mathrm{v}} t_0'$ implies $t_1 \Downarrow_{\mathrm{v}} t_1'$ and $t_0' \ \mathcal{R}^{\mathsf{NF}} \ t_1'$. A relation $\mathcal{R}$ is a normal form bisimulation if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are normal form simulations. Normal form bisimilarity, written $\approx$, is the largest normal form bisimulation.*

Henceforth, we often drop the "normal form" attribute when talking about bisimulations for brevity. Two terms t_0 and t_1 are bisimilar if their evaluations lead to matching normal forms (e.g., if t_0 evaluates to a control stuck term, then so does t_1) with bisimilar sub-components. We now detail the different cases.

Normal form bisimilarity does not distinguish between evaluation to a variable and evaluation to a λ-abstraction. Instead, we relate terms evaluating to any values v_0 and v_1 by comparing $v_0 \star x$ and $v_1 \star x$, where x is fresh. As originally pointed out by Lassen [12], this is necessary for the bisimilarity to be sound w.r.t. η-expansion; otherwise it would distinguish η-equivalent terms such as $\lambda y.x\ y$ and x. Using $\star$ instead of regular application avoids the introduction of unnecessary β-redexes, which could reveal themselves problematic in proofs.

For a control stuck term $E_0[\mathcal{S}k.t_0]$ to be executed, it has to be plugged into an evaluation context surrounded by a reset; by doing so, we obtain a term of the form $\langle t_0\{\lambda x.\langle E_0'[x] \rangle / k\} \rangle$ for some context E_0'. Notice that the resulting term is within a reset; similarly, when comparing $E_0[\mathcal{S}k.t_0]$ and $E_1[\mathcal{S}k.t_1]$, we ask for the shift bodies t_0 and t_1 to be related when surrounded by a reset. We also

compare E_0 and E_1, which amounts to executing $E_0[x]$ and $E_1[x]$ for a fresh x, since the two contexts are pure. Comparing t'_0 and t'_1 without reset would be too discriminating, as it would distinguish the two contextually equivalent terms $\mathcal{S}k.\langle t\rangle$ and $\mathcal{S}k.t$.[2] Indeed, without reset, we would have to relate $\langle t\rangle$ and t, which are not equivalent in general (take $t = \mathcal{S}k'.v$ for some v), while Definition 5 requires $\langle\langle t\rangle\rangle$ and $\langle t\rangle$ to be related (which holds for all t; see Example 3).

Two normal forms $F_0[x\,v_0]$ and $F_1[x\,v_1]$ are bisimilar if the values v_0 and v_1 as well as the contexts F_0 and F_1 are related. We have to be careful when defining bisimilarity on (possibly non pure) evaluation contexts. We cannot simply relate F_0 and F_1 by executing $F_0[y]$ and $F_1[y]$ for a fresh y. Such a definition would equate the contexts $\Box$ and $\langle\Box\rangle$, which in turn would relate the terms $x\,v$ and $\langle x\,v\rangle$, which are not contextually equivalent: they are distinguished by the context $(\lambda x.\Box)\,\lambda y.\mathcal{S}k.\Omega$. A context containing a reset enclosing the hole should be related only to contexts with the same property. However, we do not want to precisely count the number of delimiters around the hole; doing so would distinguish $\langle\Box\rangle$ and $\langle\langle\Box\rangle\rangle$, and therefore it would discriminate the contextually equivalent terms $\langle x\,v\rangle$ and $\langle\langle x\,v\rangle\rangle$. Hence, the extension of bisimulation to contexts (given before Definition 5) checks that if one of the contexts contains a reset surrounding the hole, then so does the other; then it compares the contexts beyond the first enclosing delimiter by simply evaluating them using a fresh variable. As a result, it rightfully distinguishes $\Box$ and $\langle\Box\rangle$, but it relates $\langle\Box\rangle$ and $\langle\langle\Box\rangle\rangle$.

Example 3. We prove that $\langle t\rangle \approx \langle\langle t\rangle\rangle$ by showing that $\mathcal{R}{=}\ \{(\langle t\rangle, \langle\langle t\rangle\rangle)\}\cup \approx$ is a bisimulation. If $\langle t\rangle \Downarrow_{\mathrm{v}} v$, then $\langle\langle t\rangle\rangle \Downarrow_{\mathrm{v}} v$, and $v \approx^{\mathsf{NF}} v$ holds. The case $\langle t\rangle \Downarrow_{\mathrm{v}} E[\mathcal{S}k.t']$ is not possible; one can check that if $\langle t\rangle \to_{\mathrm{v}} t'$, then t' is a value, or can be written $\langle t''\rangle$ for some t'' (and the same holds for $\langle t\rangle \Downarrow_{\mathrm{v}} t'$).

If $\langle t\rangle \Downarrow_{\mathrm{v}} F[x\,v]$, then there exists F' such that $t \Downarrow_{\mathrm{v}} F'[x\,v]$ and $F = \langle F'\rangle$. Therefore, we have $\langle\langle t\rangle\rangle \Downarrow_{\mathrm{v}} \langle\langle F'[x\,v]\rangle\rangle$. We have $v \approx^{\mathsf{NF}} v$, and we have to prove that $\langle F'\rangle\ \mathcal{R}\ \langle\langle F'\rangle\rangle$ to conclude. If F' is a pure context E, then we have to prove $\langle E[y]\rangle\ \mathcal{R}\ \langle E[y]\rangle$ and $y\ \mathcal{R}\ \langle y\rangle$ for a fresh y, which are both true because $\approx\subseteq\mathcal{R}$. If $F' = F''[\langle E\rangle]$, then given a fresh y, we have to prove $\langle F''[y]\rangle\ \mathcal{R}\ \langle\langle F''[y]\rangle\rangle$ (clear by the definition of $\mathcal{R}$), and $\langle E[y]\rangle\ \mathcal{R}\ \langle E[y]\rangle$ (true because $\approx\subseteq\mathcal{R}$).

Similarly, it is easy to check that the evaluations of $\langle\langle t\rangle\rangle$ are matched by $\langle t\rangle$.

Example 4. In [6], the authors propose variants of Curry's and Turing's call-by-value fixed point combinators using shift and reset. Let $\theta = \lambda xy.y\,(\lambda z.x\,x\,y\,z)$. We prove that Turing's combinator $t_0 = \theta\,\theta$ is bisimilar to its shift and reset variant $t_1 = \langle\theta\,\mathcal{S}k.k\,k\rangle$. We build the candidate relation $\mathcal{R}$ incrementally, starting from (t_0, t_1). Evaluating t_0 and t_1, we obtain $t_0 \Downarrow_{\mathrm{v}} \lambda y.y\,(\lambda z.\theta\,\theta\,y\,z) = v_0$ and $t_1 \Downarrow_{\mathrm{v}} \lambda y.y\,(\lambda z.(\lambda x.\langle\theta\,x\rangle)\,(\lambda x.\langle\theta\,x\rangle)\,y\,z) = v_1$; we have to add $(v_0 \star y, v_1 \star y)$ (for a fresh y) to $\mathcal{R}$. To relate these terms, we must add $(v'_0 \star z, v'_1 \star z)$ and (z, z) for a fresh z to $\mathcal{R}$, where $v'_0 = \lambda z.\theta\,\theta\,y\,z$ and $v'_1 = \lambda z.(\lambda x.\langle\theta\,x\rangle)\,(\lambda x.\langle\theta\,x\rangle)\,y\,z$. Evaluating $v'_0 \star z$ and $v'_1 \star z$, we obtain respectively $y\,v'_0\,z$ and $y\,v'_1\,z$; to relate these two normal forms, we just need to add $(x\,z, x\,z)$ (for a fresh x) to $\mathcal{R}$, since

[2] The equivalence $\mathcal{S}k.\langle t\rangle \equiv \mathcal{S}k.t$ comes from Kameyama and Hasegawa's axiomatization of shift and reset [9] and has been proved using applicative bisimilarity in [4].

we already have $v'_0 \; \mathcal{R}^{\mathsf{NF}} \; v'_1$. One can check that the constructed relation $\mathcal{R}$ is a normal form bisimulation.

In contrast, Curry's combinator $t'_0 = \lambda x.\delta_x \; \delta_x$, where $\delta_x = \lambda y.x \; (\lambda z.y \; y \; z)$, is not bisimilar to its delimited-control variant $t'_1 = \lambda x.\langle \delta_x \; \mathcal{S}k.k \; k \rangle$. Indeed, evaluating the bodies of the two values, we obtain respectively $x \; (\lambda z.\delta_x \; \delta_x \; z)$ and $\langle\langle x \; (\lambda z.(\lambda y.\langle \delta_x \; y\rangle) \; (\lambda y.\langle \delta_x \; y\rangle) \; z)\rangle\rangle$, and these open stuck terms are not bisimilar, because $\Box \not\approx \langle\langle \Box \rangle\rangle$. In fact, t'_0 and t'_1 are distinguished by the context $\Box \lambda x.\mathcal{S}k.\Omega$. Finally, we can prove that the two original combinators $\theta \; \theta$ and $\lambda x.\delta_x \; \delta_x$ are bisimilar, using the same bisimulation as in [12].

3.2 Soundness and Completeness

Usual congruence proofs for normal form bisimilarities include direct proofs, where a context and/or substitutive closure of the bisimilarity is proved to be itself a bisimulation [11, 13, 20], and proofs based on continuation-passing style (CPS) translations [12, 14]. The CPS approach consists in proving a CPS-based correspondence between the bisimilarity $\mathcal{R}_1$ we want to prove sound and a relation $\mathcal{R}_2$ that we already know is a congruence. Because CPS translations are usually themselves compatible, we can then conclude that $\mathcal{R}_1$ is a congruence. For example, for the λ-calculus, Lassen proved a CPS-correspondence between the eager normal form bisimilarity and the Böhm trees equivalence [12].

Because shift and reset have been originally defined in terms of CPS [7], one can expect the CPS approach to be successful. However, the CPS translation of shift and reset assumes that $\lambda_{\mathcal{S}}$ terms are executed within an outermost reset, and therefore they cannot evaluate to a control stuck term. For the normal form bisimilarity to be sound w.r.t. CPS, we would have to restrict its definition to terms of the form $\langle t \rangle$. This does not seem possible while keeping Definition 5 without quantification over contexts. For example, to relate values v_0 and v_1, we would have to execute $v_0 \star x$ and $v_1 \star x$ (where x is fresh) under reset. However, requiring simply $\langle v_0 \star x \rangle$ and $\langle v_1 \star x \rangle$ to be related would be unsound; such a definition would relate $\lambda y.\mathcal{S}k.k \; y$ and $\lambda y.\mathcal{S}k.(\lambda z.z) \; y$, which can be distinguished by the context $\langle \Box \; (\lambda z.z) \; \Omega \rangle$. To be sound, we would have to require $\langle E[v_0 \star x] \rangle$ to be related to $\langle E[v_1 \star x] \rangle$ *for every* E; we then introduce a quantification over contexts that we want to avoid in the first place. Because normal forms may contain control stuck terms as sub-terms, normal form bisimilarity has to be able to handle them, and, therefore, it cannot be restricted to terms of the form $\langle t \rangle$ only.

Since CPS cannot help us in proving congruence, we follow a more direct approach, by relying on a context closure. Given a relation $\mathcal{R}$, we define its substitutive, reflexive, and context closure $\widehat{\mathcal{R}}$ by the rules of Fig. 1. The main lemma of the congruence proof is then as follows:

Lemma 4. *If $\mathcal{R}$ is a normal form bisimulation, then so is $\widehat{\mathcal{R}}$.*

More precisely, we prove that if $t_0 \; \widehat{\mathcal{R}} \; t_1$ and t_0 evaluates to some normal form t'_0 in m steps, then t_1 evaluates to a normal form t'_1 such that $t'_0 \; \widehat{\mathcal{R}}^{\mathsf{NF}} \; t'_1$. The

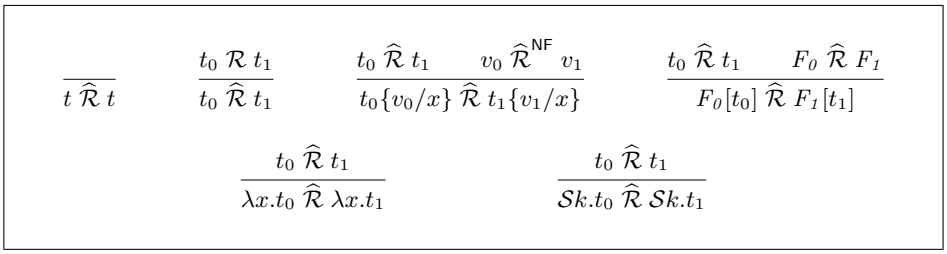

Fig. 1: Substitutive, reflexive, and context closure of a relation $\mathcal{R}$

proof is by nested induction on m and on the definition of $\widehat{\mathcal{R}}$; it can be found in [5]. Congruence of $\approx$ then follows immediately.

Corollary 1. *The relation $\approx$ is a congruence*

We can then easily prove that $\approx$ is sound w.r.t. contextual equivalence.

Theorem 1. *We have $\approx \subseteq \approx_c$.*

The following counter-example shows that the inclusion is in fact strict; normal form bisimilarity is not complete.

Proposition 1. *Let $i = \lambda y.y$. We have $\langle\langle x\, i\rangle\, \mathcal{S}k.i\rangle \approx_c \langle\langle x\, i\rangle\, (\langle x\, i\rangle\, \mathcal{S}k.i)\rangle$, but $\langle\langle x\, i\rangle\, \mathcal{S}k.i\rangle \not\approx \langle\langle x\, i\rangle\, (\langle x\, i\rangle\, \mathcal{S}k.i)\rangle$.*

Proof. Replacing x by a closed value v, we get $\langle\langle v\, i\rangle\, \mathcal{S}k.i\rangle$ and $\langle\langle v\, i\rangle\, (\langle v\, i\rangle\, \mathcal{S}k.i)\rangle$, which both evaluate to i if the evaluation of $\langle v\, i\rangle$ terminates (otherwise, they both diverge). With this observation, it is easy to prove that $\langle\langle x\, i\rangle\, \mathcal{S}k.i\rangle$ and $\langle\langle x\, i\rangle\, (\langle x\, i\rangle\, \mathcal{S}k.i)\rangle$ are contextually equivalent. They are not bisimilar, because the terms $\langle y\, \mathcal{S}k.i\rangle$ and $\langle y\, (\langle x\, i\rangle\, \mathcal{S}k.i)\rangle$ (where y is fresh) are not bisimilar: the former evaluates to i while the latter is in normal form (but is not a value). □

4 Refined Bisimilarity and Up-to Techniques

In this section, we propose an improvement of the definition of normal form bisimilarity, and we discuss some proof techniques which aim at simplifying equivalence proofs.

4.1 Refined Bisimilarity

Normal form bisimilarity could better deal with control stuck terms. To illustrate this, consider the following terms.

Proposition 2. *Let $i = \lambda x.x$. We have $\mathcal{S}k.i \approx_c (\mathcal{S}k.i)\ \Omega$, but $\mathcal{S}k.i \not\approx (\mathcal{S}k.i)\ \Omega$.*

Proof. If $\mathcal{S}k.i$ and $(\mathcal{S}k.i)\ \Omega$ are put within a pure context, then we obtain two control stuck terms, and if we put these two terms within a context $F[\langle E\rangle]$, then they both reduce to $F[i]$. Therefore, $\mathcal{S}k.i$ and $(\mathcal{S}k.i)\ \Omega$ are contextually equivalent. They are not normal form bisimilar, since the contexts $\Box$ and $\Box\ \Omega$ are not bisimilar (x converges while $x\ \Omega$ diverges). □

When comparing control stuck terms, normal form bisimilarity considers contexts and shift bodies separately, while they are combined if the control stuck terms are put under a reset and the capture goes through. To fix this issue, we consider another notion of bisimulation. Given a relation $\mathcal{R}$ on terms, we define $\mathcal{R}^{\mathsf{RNF}}$ on normal forms, which is defined the same way as $\mathcal{R}^{\mathsf{NF}}$ on values and open stuck terms, and is defined on control stuck terms as follows:

$$\frac{\langle t_0'\{\lambda x.\langle k'\ E_0[x]\rangle/k\}\rangle\ \mathcal{R}\ \langle t_1'\{\lambda x.\langle k'\ E_1[x]\rangle/k\}\rangle \qquad k', x \text{ fresh}}{E_0[\mathcal{S}k.t_0]\ \mathcal{R}^{\mathsf{RNF}}\ E_1[\mathcal{S}k.t_1]}$$

Definition 6. *A relation $\mathcal{R}$ on terms is a refined normal form simulation if $t_0\ \mathcal{R}\ t_1$ and $t_0 \Downarrow_{\mathrm{v}} t_0'$ implies $t_1 \Downarrow_{\mathrm{v}} t_1'$ and $t_0'\ \mathcal{R}^{\mathsf{RNF}}\ t_1'$. A relation $\mathcal{R}$ is a refined normal form bisimulation if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are refined normal form simulations. Refined normal form bisimilarity, written $\overset{\bullet}{\approx}$, is the largest refined normal form bisimulation.*

In the control stuck terms case, Definition 6 simulates the capture of E_0 (respectively E_1) by $\mathcal{S}k.t_0$ (respectively $\mathcal{S}k.t_1$). However, if t_0 is put into a context $\langle E\rangle$, then $\mathcal{S}k.t_0$ captures a context bigger than E_0, namely $E[E_0]$. We take such possibility into account by using a variable k' in the definition of $\mathcal{R}^{\mathsf{RNF}}$, which represents the context that can be captured beyond E_0 and E_1.

Refined bisimilarity contains the regular bisimilarity.

Proposition 3. *We have $\approx\ \subset\ \overset{\bullet}{\approx}$.*

Indeed, for control stuck terms, we have $t_0 \Downarrow_{\mathrm{v}} E_0[\mathcal{S}k.t_0']$, $t_1 \Downarrow_{\mathrm{v}} E_1[\mathcal{S}k.t_1']$, $E_0 \approx E_1$, and $\langle t_0'\rangle \approx \langle t_1'\rangle$. Because $\approx$ is a congruence (Corollary 1), it is easy to see that $\langle t_0'\{\lambda x.\langle k'\ E_0[x]\rangle/k\}\rangle \approx \langle t_1'\{\lambda x.\langle k'\ E_1[x]\rangle/k\}\rangle$ holds for fresh k' and x. Therefore, $\approx$ is a refined bisimulation, and is included in $\overset{\bullet}{\approx}$. The inclusion is strict, because $\overset{\bullet}{\approx}$ relates the terms of Proposition 2, while $\approx$ does not.

Proving that $\overset{\bullet}{\approx}$ is sound requires some adjustments to the congruence proof of $\approx$. First, given a relation $\mathcal{R}$ on terms, we define its substitutive, bisimilar, and context closure $\widetilde{\mathcal{R}}$ by extending the rules of Fig. 1 with the following one.

$$\frac{t_0 \overset{\bullet}{\approx} t_0' \qquad t_0'\ \widetilde{\mathcal{R}}\ t_1' \qquad t_1' \overset{\bullet}{\approx} t_1}{t_0\ \widetilde{\mathcal{R}}\ t_1}$$

Henceforth, we simply write $\overset{\bullet}{\approx}\widetilde{\mathcal{R}}\overset{\bullet}{\approx}$ for the composition of the three relations. Our goal is to prove that $\widetilde{\overset{\bullet}{\approx}}$ is a refined bisimilarity. To this end, we need a few lemmas.

Lemma 5. *If $x \notin \mathsf{fv}(E)$, then $(\lambda x.E[x])\ t \approx E[t]$.*

One can prove that $\{((\lambda x.E[x])\,t, E[t]), x \notin \mathsf{fv}(E)\} \cup \{(t,t)\}$ is a bisimulation, by a straightforward case analysis on the result of the evaluation of t (if it exists). Note that Lemma 5, known as the β_Ω axiom in [9], has also been proved in [4]

using applicative bisimulation. We can see that the proof is much simpler using normal form bisimulation. With Lemma 5, congruence of $\approx$, and Proposition 3, we then prove the following result.

Lemma 6. *If* $x \notin \mathsf{fv}(E_0) \cup \mathsf{fv}(E_1)$ *and* $y \notin \mathsf{fv}(E_1)$ *then* $\langle t\{\lambda x.\langle E_1[E_0[x]]\rangle/k\}\rangle \mathrel{\dot{\approx}} \langle t\{\lambda x.\langle(\lambda y.E_1[y])\ E_0[x]\rangle/k\}\rangle$.

The main lemma of the congruence proof of $\dot{\approx}$ is as follows.

Lemma 7. *If* $\mathcal{R}$ *is a refined bisimulation, then so is* $\widetilde{\mathcal{R}}$.

The proof is an adaptation of the proof of Lemma 4. We sketch one sub-case of the proof, to illustrate why we need $\widetilde{\mathcal{R}}$ (instead of $\widehat{\mathcal{R}}$) and Lemma 6.

Proof (Sketch). Assume we are in the case where $E_0[t_0]\ \widetilde{\mathcal{R}}\ E_1[t_1]$ with $E_0[y]\ \widetilde{\mathcal{R}}\ E_1[y]$ for a fresh y, and $t_0\ \widetilde{\mathcal{R}}\ t_1$. Moreover, suppose $t_0 \Downarrow_{\mathrm{v}} E_0'[\mathcal{S}k.t_0']$. Then by the induction hypothesis, we know that there exist E_1', t_1' such that $t_1 \Downarrow_{\mathrm{v}} E_1'[\mathcal{S}k.t_1']$, and $\langle t_0'\{\lambda x.\langle k'\ E_0'[x]\rangle/k\}\rangle\ \widetilde{\mathcal{R}}\ \langle t_1'\{\lambda x.\langle k'\ E_1'[x]\rangle/k\}\rangle$ (*) for a fresh k'. Hence, we have $E_0[t_0] \Downarrow_{\mathrm{v}} E_0[E_0'[\mathcal{S}k.t_0']]$ and $t_1 \Downarrow_{\mathrm{v}} E_1[E_1'[\mathcal{S}k.t_1']]$, and we want to prove that $\langle t_0'\{\lambda x.\langle k'\ E_0[E_0'[x]]\rangle/k\}\rangle\ \widetilde{\mathcal{R}}\ \langle t_1'\{\lambda x.\langle k'\ E_1[E_1'[x]]\rangle/k\}\rangle$ holds. Because $E_0[y]\ \widetilde{\mathcal{R}}\ E_1[y]$, we have $\lambda y.k'\ E_0[y]\ \widetilde{\mathcal{R}}^{\mathsf{RNF}}\ \lambda y.k'\ E_1[y]$ (**). Using (*) and (**), we obtain

$$\langle t_0'\{\lambda x.\langle(\lambda y.k'\ E_0[y])\ E_0'[x]\rangle/k\}\rangle\ \widetilde{\mathcal{R}}\ \langle t_1'\{\lambda x.\langle(\lambda y.k'\ E_1[y])\ E_1'[x]\rangle/k\}\rangle,$$

because $\widetilde{\mathcal{R}}$ is substitutive. By Lemma 6, we know that

$$\langle t_0'\{\lambda x.\langle k'\ E_0[E_0'[x]]\rangle/k\}\rangle \mathrel{\dot{\approx}} \langle t_0'\{\lambda x.\langle(\lambda y.k'\ E_0[y])\ E_0'[x]\rangle/k\}\rangle$$
$$\langle t_1'\{\lambda x.\langle k'\ E_1[E_1'[x]]\rangle/k\}\rangle \mathrel{\dot{\approx}} \langle t_1'\{\lambda x.\langle(\lambda y.k'\ E_1[y])\ E_1'[x]\rangle/k\}\rangle,$$

which means that $\langle t_0'\{\lambda x.\langle k'\ E_0[E_0'[x]]\rangle/k\}\rangle \mathrel{\dot{\approx}\widetilde{\mathcal{R}}\dot{\approx}} \langle t_1'\{\lambda x.\langle k'\ E_1[E_1'[x]]\rangle/k\}\rangle$ holds. The required result then holds because $\dot{\approx}\widetilde{\mathcal{R}}\dot{\approx} \subseteq \widetilde{\mathcal{R}}$. □

We can then conclude that $\dot{\approx}$ is a congruence, and is sound w.r.t. $\approx_c$.

Corollary 2. *The relation* $\dot{\approx}$ *is a congruence.*

Theorem 2. *We have* $\dot{\approx} \subset \approx_c$.

The inclusion is strict, because the terms of Proposition 1 are still not related by $\dot{\approx}$.

We would like to stress that even though $\dot{\approx}$ equates more contextually equivalent terms than $\approx$, the latter is still useful, since it leads to very simple proofs of equivalence, as we can see with Lemma 5 (and with the examples of Section 5). Therefore, $\dot{\approx}$ does not disqualify $\approx$ as a proof technique.

4.2 Up-to Techniques

The idea behind up-to techniques [19, 10, 18] is to define relations that are not exactly bisimulations but are included in bisimulations. It usually leads to definitions of simpler candidate relations and to simpler bisimulation proofs. As pointed out in [10], using a direct approach to prove congruence of the normal form bisimilarity (as in Sections 3.2 and 4.1) makes up-to techniques based on the context closure easy to define and to prove valid. For example, we define bisimulation up to substitutive, reflexive, and context closure (in short, up to context) as follows.

Definition 7. *A relation $\mathcal{R}$ on terms is a simulation up to context if $t_0\ \mathcal{R}\ t_1$ and $t_0 \Downarrow_{\mathrm{v}} t'_0$ implies $t_1 \Downarrow_{\mathrm{v}} t'_1$ and $t'_0\ \widehat{\mathcal{R}}^{\mathsf{NF}}\ t'_1$. A relation $\mathcal{R}$ is a bisimulation up to context if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are simulations up to context.*

Similarly, we can define a notion of refined bisimulation up to context by replacing $\widehat{\mathcal{R}}^{\mathsf{NF}}$ by $\widetilde{\mathcal{R}}^{\mathsf{RNF}}$ in the above definition. The proofs of Lemmas 4 and 7 can easily be adapted to bisimulations up to context; a trivial change is needed only in the inductive case where $t_0\ \widehat{\mathcal{R}}\ t_1$ (respectively $t_0\ \widetilde{\mathcal{R}}\ t_1$) comes from $t_0\ \mathcal{R}\ t_1$.

Lemma 8. *If $\mathcal{R}$ is a bisimulation up to context, then $\widehat{\mathcal{R}}$ is a bisimulation. If $\mathcal{R}$ is a refined bisimulation up to context, then $\widetilde{\mathcal{R}}$ is a refined bisimulation.*

Consequently, if $\mathcal{R}$ is a bisimulation up to context, and if $t_0\ \mathcal{R}\ t_1$, then $t_0 \approx t_1$, because $\mathcal{R} \subseteq \widehat{\mathcal{R}} \subseteq \approx$.

Example 5. We can simplify the proof of bisimilarity between Turing's fixed point combinator and its delimited-control variant (cf. Example 4); indeed, it is enough to prove that $\mathcal{R} = \{(\theta\ \theta, \langle \theta\ \mathcal{S}k.\, k\rangle), (\theta\ \theta, (\lambda x.\langle \theta\ x\rangle)\ (\lambda x.\langle \theta\ x\rangle))\}$ is a bisimulation up to context.

When proving equivalence of terms, it is sometimes easier to reason in a small-step fashion instead of trying to evaluate terms completely. To allow this kind of reasoning, we define the following small-step notion.

Definition 8. *A relation $\mathcal{R}$ on terms is a small-step simulation up to context if $t_0\ \mathcal{R}\ t_1$ implies:*

- *if $t_0 \to_{\mathrm{v}} t'_0$, then there exists t'_1 such that $t_1 \to_{\mathrm{v}}^* t'_1$ and $t'_0\ \widehat{\mathcal{R}}\ t'_1$;*
- *if t_0 is a normal form, then there exists t'_1 such that $t_1 \Downarrow_{\mathrm{v}} t'_1$ and $t_0\ \widehat{\mathcal{R}}^{\mathsf{NF}}\ t'_1$.*

A relation $\mathcal{R}$ is a small-step bisimulation up to context if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are small-step simulations up to context.

Similarly, we can define the refined variant. Again, it is easy to check the validity of these two proof techniques.

Lemma 9. *If $\mathcal{R}$ is a small-step bisimulation up to context, then $\widehat{\mathcal{R}}$ is a bisimulation. If $\mathcal{R}$ is a refined small-step bisimulation up to context, then $\widetilde{\mathcal{R}}$ is a refined bisimulation.*

In the next section we show how these relations can be used (Proposition 5).

5 Examples

We now illustrate the usefulness of the relations and techniques defined in this paper, by proving some terms equivalences derived from the axiomatization of $\lambda_{\mathcal{S}}$ [9]. The relationship between contextual equivalence and Kameyama and Hasegawa's axioms has been studied in [4], using applicative bisimilarity. In particular, we show that terms equated by all the axioms except for $\mathcal{S}$ `elim` ($\mathcal{S}k.k\,t = t$ if $k \notin \mathsf{fv}(t)$) are applicative bisimilar. The same result can be obtained for normal form bisimilarity, using the same candidate relations as for applicative bisimilarity (see Propositions 1 to 4 in [4]), except for the β_Ω axiom, where the equivalence proof becomes much simpler (see Lemma 5). The terms $\mathcal{S}k.k\,v$ and v (equated by $\mathcal{S}$ `elim`) are not (applicative or normal form) bisimilar, because the former is control stuck while the latter is not. Conversely, there exist bisimilar terms that are not related by the axiomatization, such as $\Omega\,\Omega$ and Ω, or Curry's and Turing's combinators (Example 4).

In this section, we propose several terms equivalences, the proofs of which are quite simple using normal form bisimulation, especially compared to applicative bisimulation. In the following, we write $\mathcal{I}$ for the identity bisimulation $\{(t,t)\}$.

Proposition 4. *If $x \notin \mathsf{fv}(E)$, then $E[(\lambda x.t_0)\,t_1] \approx (\lambda x.E[t_0])\,t_1$.*

Proof. By showing that $\{(E[(\lambda x.t_0)\,t_1], (\lambda x.E[t_0])\,t_1), x \notin \mathsf{fv}(E)\} \cup \mathcal{I}$ is a normal form bisimulation. The proof is straightforward by case analysis on the result of the evaluation of t_1 (if it exists). □

The next example demonstrates how useful small-step relations can be.

Proposition 5. *If $x \notin \mathsf{fv}(E)$, then $\langle(\lambda x.\langle E[x]\rangle)\,t\rangle \approx \langle E[t]\rangle$.*

Proof. Let $\mathcal{R} = \{(\langle(\lambda x.\langle E[x]\rangle)\,t\rangle, \langle E[t]\rangle), x \notin \mathsf{fv}(E)\}$. We prove that $\mathcal{R} \cup \approx$ is a small-step bisimulation up to context, by case analysis on t.

- If $t \to_{\mathrm{v}} t'$, then $\langle(\lambda x.\langle E[x]\rangle)\,t\rangle \to_{\mathrm{v}} \langle(\lambda x.\langle E[x]\rangle)\,t'\rangle$, $\langle E[t]\rangle \to_{\mathrm{v}} \langle E[t']\rangle$, and we have $\langle(\lambda x.\langle E[x]\rangle)\,t'\rangle\ \mathcal{R}\ \langle E[t']\rangle$, as required.
- If $t = v$, then $\langle(\lambda x.\langle E[x]\rangle)\,v\rangle \to_{\mathrm{v}} \langle\langle E[v]\rangle\rangle$. We have proved in Example 3 that $\langle\langle E[v]\rangle\rangle \approx \langle E[v]\rangle$.
- If $t = F[y\,v]$, then we have to relate $\langle(\lambda x.\langle E[x]\rangle)\,F\rangle$ and $\langle E[F]\rangle$ (we clearly have $v \approx^{\mathsf{NF}} v$). If $F = F'[\langle E'\rangle]$, then we have $\langle(\lambda x.\langle E[x]\rangle)\,F'[z]\rangle\ \mathcal{R}\ \langle E[F'[z]]\rangle$ and $\langle E'[z]\rangle \approx \langle E'[z]\rangle$ for a fresh z. If $F = E'$, then $\langle(\lambda x.\langle E[x]\rangle)\,E'[z]\rangle\ \mathcal{R}\ \langle E[E'[z]]\rangle$ holds for a fresh z.
- If $t = E'[\mathcal{S}k.t']$, then $\langle(\lambda x.\langle E[x]\rangle)\,t\rangle \to_{\mathrm{v}} \langle t'\{\lambda y.\langle(\lambda x.\langle E[x]\rangle)\,E'[y]\rangle/k\}\rangle$, and $\langle E[t]\rangle \to_{\mathrm{v}} \langle t'\{\lambda y.\langle E[E'[y]]\rangle/k\}\rangle$. We have $\langle(\lambda x.\langle E[x]\rangle)\,E'[y]\rangle\ \mathcal{R}\ \langle E[E'[y]]\rangle$, therefore $\langle t'\{\lambda y.\langle(\lambda x.\langle E[x]\rangle)\,E'[y]\rangle/k\}\rangle\ \widehat{\mathcal{R}}\ \langle t'\{\lambda y.\langle E[E'[y]]\rangle/k\}\rangle$ holds, as wished. □

Without using small-step bisimulation, the definition of $\mathcal{R}$ as well as the bisimulation proof would be much more complex, since we would have to compute the results of the evaluations of $\langle(\lambda x.\langle E[x]\rangle)\,t\rangle$ and of $\langle E[t]\rangle$, which is particularly difficult if t is a control stuck term.

For the next example, we have to use refined bisimilarity.

Proposition 6. *If* $k' \notin \mathsf{fv}(E) \cup \mathsf{fv}(t)$ *and* $x \notin \mathsf{fv}(E)$*, then we have* $E[\mathcal{S}k.t] \mathrel{\dot{\approx}} \mathcal{S}k'.t\{\lambda x.\langle k'\ E[x]\rangle/k\}$.

Proof. The two terms are control stuck terms, therefore, we have to prove $\langle t\{\lambda x.\langle k''\ E[x]\rangle/k\}\rangle \mathrel{\dot{\approx}} \langle t\{\lambda x.\langle(\lambda y.\langle k''\ y\rangle)\ E[x]\rangle/k\}\rangle$ for a fresh k''. We know that $\langle k''\ E[x]\rangle \approx \langle(\lambda y.\langle k''\ y\rangle)\ E[x]\rangle$ holds by Proposition 5. Consequently, we have $\langle k''\ E[x]\rangle \mathrel{\dot{\approx}} \langle(\lambda y.\langle k''\ y\rangle)\ E[x]\rangle$ by Proposition 3. We can then conclude by congruence of $\mathrel{\dot{\approx}}$. □

Without Proposition 5, we would have to prove $\langle k''\ E[x]\rangle \mathrel{\dot{\approx}} \langle(\lambda y.\langle k''\ y\rangle)\ E[x]\rangle$ directly, using a small-step refined bisimulation up to context. Proving Proposition 6 with the regular normal form bisimilarity would require us to equate $E[y]$ and y (where y is fresh), which is not possible if $E = (\lambda z.\Omega)\ \Box$.

6 Conclusion

In this paper, we propose several normal formal bisimilarities for a λ-calculus with shift and reset, and we demonstrate their usefulness on several examples. Proving equivalences with the regular normal form bisimilarity generates minimal proof obligations, especially when used with up-to techniques. If the regular bisimilarity fails to relate the tested terms, then the refined bisimilarity can be of help. If they both fail, then we may have to use the applicative bisimilarity [4], which, unlike the bisimilarities of this paper, is complete.

We believe this work can easily be adapted to other delimited-control operators as well as the CPS hierarchy [7]. It might also be interesting to extend this work to the typed setting. Another possible future work would be to define *environmental bisimulations* [18] for $\lambda_{\mathcal{S}}$. When comparing two terms, environmental relations use an additional component, the environment, which represents the current knowledge of the observer. E.g., in the pure λ-calculus, when two tested terms reduce to values, they become known to the observer and are added to the environment. The observer can then challenge two λ-abstractions by applying them to two related arguments built from the environment. Environmental bisimilarities are usually sound and complete, and also allow for up-to techniques.

Another issue is to find a characterization of contextual equivalence for λ-calculi with abortive control operators. Normal form bisimilarities have been defined for extensions of the $\lambda\mu$-calculus [14], but they are usually not complete, except in the presence of a store construct [20]. It might be possible to reach completeness with applicative or environmental bisimilarities.

Acknowledgments: We thank Małgorzata Biernacka and the anonymous referees for insightful comments on the presentation of this work.

References

1. S. Abramsky and C.-H. L. Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, 105:159–267, 1993.

2. M. Biernacka, D. Biernacki, and O. Danvy. An operational foundation for delimited continuations in the CPS hierarchy. *Logical Methods in Computer Science*, 1(2:5):1–39, Nov. 2005.
3. D. Biernacki, O. Danvy, and K. Millikin. A dynamic continuation-passing style for dynamic delimited continuations. Technical Report BRICS RS-05-16, DAIMI, Department of Computer Science, Aarhus University, Aarhus, Denmark, May 2005.
4. D. Biernacki and S. Lenglet. Applicative bisimulations for delimited-control operators. In L. Birkedal, editor, *FOSSACS'12*, number 7213 in Lecture Notes in Computer Science, pages 119–134, Tallinn, Estonia, Mar. 2012. Springer-Verlag.
5. D. Biernacki and S. Lenglet. Normal form bisimulations for delimited-control operators, Feb. 2012. Available at `http://arxiv.org/abs/1202.5959`.
6. O. Danvy and A. Filinski. A functional abstraction of typed contexts. DIKU Rapport 89/12, DIKU, Computer Science Department, University of Copenhagen, Copenhagen, Denmark, July 1989.
7. O. Danvy and A. Filinski. Abstracting control. In M. Wand, editor, *LFP'90*, pages 151–160, Nice, France, June 1990. ACM Press.
8. A. Filinski. Representing monads. In H.-J. Boehm, editor, *POPL'94*, pages 446–457, Portland, Oregon, Jan. 1994. ACM Press.
9. Y. Kameyama and M. Hasegawa. A sound and complete axiomatization of delimited continuations. In O. Shivers, editor, *ICFP'03*, SIGPLAN Notices, Vol. 38, No. 9, pages 177–188, Uppsala, Sweden, Aug. 2003. ACM Press.
10. S. B. Lassen. Relational reasoning about contexts. In A. D. Gordon and A. M. Pitts, editors, *Higher Order Operational Techniques in Semantics*. Cambridge University Press, 1998. 91-135.
11. S. B. Lassen. Bisimulation in untyped lambda calculus: Böhm trees and bisimulation up to context. In M. M. Stephen Brookes, Achim Jung and A. Scedrov, editors, *MFPS'99*, volume 20 of *ENTCS*, pages 346–374, New Orleans, LA, Apr. 1999. Elsevier Science.
12. S. B. Lassen. Eager normal form bisimulation. In P. Panangaden, editor, *LICS'05*, pages 345–354, Chicago, IL, June 2005. IEEE Computer Society Press.
13. S. B. Lassen. Normal form simulation for McCarthy's amb. In M. Escardó, A. Jung, and M. Mislove, editors, *MFPS'05*, volume 155 of *ENTCS*, pages 445–465, Birmingham, UK, May 2005. Elsevier Science Publishers.
14. S. B. Lassen. Head normal form bisimulation for pairs and the $\lambda\mu$-calculus. In R. Alur, editor, *LICS'06*, pages 297–306, Seattle, WA, Aug. 2006. IEEE Computer Society Press.
15. R. Milner. Fully abstract models of typed λ-calculi. *Theoretical Computer Science*, 4(1):1–22, 1977.
16. J. H. Morris. *Lambda Calculus Models of Programming Languages*. PhD thesis, Massachusets Institute of Technology, 1968.
17. D. Sangiorgi. The lazy lambda calculus in a concurrency scenario. In *LICS'92*, pages 102–109, Santa Cruz, California, June 1992. IEEE Computer Society.
18. D. Sangiorgi, N. Kobayashi, and E. Sumii. Environmental bisimulations for higher-order languages. In J. Marcinkowski, editor, *LICS'07*, pages 293–302, Wroclaw, Poland, July 2007. IEEE Computer Society Press.
19. D. Sangiorgi and D. Walker. *The Pi-Calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.
20. K. Støvring and S. B. Lassen. A complete, co-inductive syntactic theory of sequential control and state. In M. Felleisen, editor, *POPL'07*, SIGPLAN Notices, Vol. 42, No. 1, pages 161–172, New York, NY, USA, Jan. 2007. ACM Press.