Logical Methods in Computer Science
Volume 15, Issue 2, 2019, pp. 18:1–18:57
https://lmcs.episciences.org/

Submitted Apr. 24, 2018
Published May 24, 2019

BISIMULATIONS FOR DELIMITED-CONTROL OPERATORS

DARIUSZ BIERNACKI [a], SERGUEÏ LENGLET [b], AND PIOTR POLESIUK [a]

[a] University of Wrocław
e-mail address: {dabi,ppolesiuk}@cs.uni.wroc.pl

[b] Université de Lorraine
e-mail address: serguei.lenglet@univ-lorraine.fr

Abstract. We present a comprehensive study of the behavioral theory of an untyped λ-calculus extended with the delimited-control operators shift and reset. To that end, we define a contextual equivalence for this calculus, that we then aim to characterize with coinductively defined relations, called *bisimilarities*. We consider different styles of bisimilarities (namely applicative, normal-form, and environmental) within a unifying framework, and we give several examples to illustrate their respective strengths and weaknesses. We also discuss how to extend this work to other delimited-control operators.

1. Introduction

Delimited-control operators. Control operators for delimited continuations enrich a programming language with the ability to delimit the current continuation, to capture such a delimited continuation, and to compose delimited continuations. Such operators have been originally proposed independently by Felleisen [26] and by Danvy and Filinski [21], with numerous variants designed subsequently [34, 66, 32, 25]. The applications of delimited-control operators range from non-deterministic programming [21, 45], partial evaluation [58, 19], and normalization by evaluation [24] to concurrency [34], mobile code [89], linguistics [84], operating systems [43], and probabilistic programming [44]. Several variants of delimited-control operators are nowadays available in mainstream functional languages such as Haskell [25], OCaml [42], Racket [29], and Scala [76].

The control operators shift and reset [21] were designed to account for the traditional model of non-deterministic programming based on success and failure continuations [96], and their semantics as well as pragmatics take advantage of an extended continuation-passing style (CPS), where the continuation of the computation is represented by the current delimited continuation (the success continuation) and a metacontinuation (the failure continuation). The control delimiter reset resets the current continuation, whereas the control operator

Key words and phrases: Delimited-control operators, behavioral equivalences, bisimulations.
This work was supported by PHC Polonium and by National Science Centre, Poland, grant no. 2014/15/B/ST6/00619.

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.23638/LMCS-15(2:18)2019

shift captures the current continuation, which then can be either discarded (expressing failure in a backtracking search) or duplicated (expressing a backtracking point creation). When a captured continuation is resumed, the then-current continuation is pushed on the metacontinuation (representing a list of pending delimited continuations). For this reason shift and reset are known as static delimited-control operators, as opposed, e.g., to control and prompt [26] that are dynamic, in that they require an actual stack concatenation to compose continuations, and for this reason go beyond the standard CPS [12] [1].

The static delimited-control operators have been surrounded by an array of CPS-based semantic artifacts that greatly support programming and reasoning about code, by making it possible to interpret effectful programs in a purely functional language. As a matter of fact, most of the applications of delimited-control listed above have been presented using shift and reset. But the connection with CPS is even more intimate—in his seminal article [27], Filinski showed that because the continuation monad can express any other monad, shift and reset can express any monadic effect (such as exceptions or non-determinism) in direct style. Furthermore, iterating the CPS transformation for a language with shift and reset leads to a CPS hierarchy [21] which in turn allows one to express layered computational effects in direct style [28]. These results establish a special position of shift and reset among all the delimited-control operators considered in the literature, even though from an operational standpoint, shift and reset can be easily expressed in terms of dynamic control operators [10]. Interestingly, the abortive control operator call/cc known from Scheme and SML of New Jersey requires the presence of mutable state to obtain the expressive power of shift and reset [27] (and of other delimited-control operators).

Relying on the CPS translation to a pure language is helpful and inspiring when programming with shift and reset, but it is arguably more convenient to reason directly about the code with control operators. To facilitate such reasoning, Kameyama et al. devised direct-style axiomatizations for a number of delimited-control calculi [39, 38, 40] that are sound and complete with respect to the corresponding CPS translations. Numerous other results concerning equational reasoning in various calculi for delimited continuations [77, 2, 33, 62] show that it has been a topic of active research.

While the CPS-based equational theories are a natural consequence of the denotational or translational semantics of control operators such as shift and reset, they are not strong enough to verify the equivalences of programs that have unrelated images through the CPS translation, but that operationally cannot be distinguished (e.g., take two different fixed-point combinators). In order to build a stronger theory of program equivalence for delimited control, we turn to the operational foundations of shift and reset, and consider operationally-phrased criteria for program equivalence. The original operational semantics of shift and reset, in the form of an abstract machine and a corresponding context-sensitive reduction semantics, has been derived through defunctionalization [7]. Here the concepts of the delimited continuation and the metacontinuation are materialized as a stack and a metastack of the machine, and as a context and a metacontext in the reduction semantics. A direct consequence of this semantics is that no "missing reset" error can occur in the course of program evaluation—a reset guarding the current delimited continuation is always present. A *relaxed* version of the semantics, where a delimiter surrounding the context is not statically ensured has also been considered in the literature [39] and in some implementations [27].

[1] Expressing control and prompt, as well as other dynamic control operators, in terms of shift and reset and, therefore, in CPS is possible, but it requires an involved continuation answer type relying on recursion [11, 41, 85].

This less-structured approach sacrifices the direct correspondence with CPS for flexibility and it scales better to other delimited-control operators.

Behavioral equivalences. Because of the complex nature of control effects, it can be difficult to determine if two programs that use shift and reset are equivalent (i.e., behave in the same way) or not. *Contextual equivalence* [67] is widely considered as the most natural equivalence on terms in languages based on the λ-calculus. The intuition behind this relation is that two programs are equivalent if replacing one by the other in a bigger program does not change the behavior of this bigger program. The behavior of a program has to be made formal by defining the *observable actions* we want to take into account for the calculus we consider. It can be, e.g., inputs and outputs for communicating systems [82], memory reads and writes, etc. For the plain λ-calculus [1], it is usually whether the term terminates or not. The "bigger program" can be seen as a *context* (a term with a hole) and, therefore, two terms t_0 and t_1 are contextually equivalent if we cannot tell them apart when evaluated within any context C, i.e., if $C[t_0]$ and $C[t_1]$ produce the same observable actions.

The latter quantification over contexts C makes contextual equivalence hard to use in practice to prove that two given terms are equivalent. As a result, one usually looks for more tractable alternatives to contextual equivalence, such as logical relations (see, e.g., [71]), axiomatizations (see, e.g., [50]), or *bisimulations*. A bisimulation relates two terms t_0 and t_1 by asking them to mimic each other in a coinductive way, e.g., if t_0 reduces to a term t'_0, then t_1 has to reduce to a term t'_1 so that t'_0 and t'_1 are still in the bisimulation, and conversely for the reductions of t_1. An equivalence on terms, called *bisimilarity* can be derived from a notion of bisimulation: two terms are bisimilar if there exists a bisimulation which relates them. Finding an appropriate notion of bisimulation consists in finding the conditions on which two terms are related, so that the resulting notion of bisimilarity is *sound* and *complete* w.r.t. contextual equivalence, (i.e., it is included in and it contains contextual equivalence, respectively).

Different styles of bisimulations have been proposed for calculi similar to the λ-calculus. For example, *applicative* bisimilarity [1] relates terms by reducing them to values (if possible), and the resulting values have to be themselves applicative bisimilar when applied to an arbitrary argument. As we can see, applicative bisimilarity still contains some quantification over arguments to compare values, but is nevertheless easier to use than contextual equivalence because of its coinductive nature—bisimulation relations are constructed incrementally, following a step-by-step analysis of the possible interactions of the program with its environment, and also because we do not have to consider all forms of contexts. When sound, applicative bisimilarity is usually also complete w.r.t. contextual equivalence, at least for deterministic languages such as the plain λ-calculus [1].

Environmental bisimilarity [80, 81] is quite similar to applicative bisimilarity, as it compares terms by reducing them to values, and then requires the resulting values to be bisimilar when applied to some arguments. However, the arguments are no longer arbitrary, but built using an *environment*, which represents the knowledge accumulated so far by an outside observer on the tested terms. Like applicative bisimilarity, environmental bisimilarity is usually sound and complete, but it also allows for *up-to techniques* to simplify its equivalence proofs. The idea behind up-to techniques is to define relations that are not exactly bisimulations but are included in bisimulations. Finding an up-to relation equating two given terms is usually simpler than finding a regular bisimulation relating these terms.

Unlike for environmental bisimilarity, the definition of useful up-to techniques for applicative bisimilarity remains an open problem.

In contrast to applicative and environmental bisimilarity, *normal-form* bisimilarity [53] (also called *open* bisimilarity in [78]) does not contain any quantification over arguments or contexts in its definition. The principle is to reduce the compared terms to normal forms (if possible), and then to decompose the resulting normal forms into sub-components that have to be themselves bisimilar. Unlike applicative or environmental bisimilarity, normal-form bisimilarity is usually not complete, i.e., there exist contextually equivalent terms that are not normal-form bisimilar. But because of the lack of quantification over contexts, proving that two terms are normal-form bisimilar is usually quite simple, and the proofs can be further simplified with the help of up-to techniques (like with environmental bisimilarity).

This work. In this article, we present a comprehensive study of the behavioral theory of a λ-calculus extended with the operators shift and reset, called $\lambda_{\mathcal{S}}$. In previous works, we defined applicative [13], normal-form [14, 17], and environmental [15, 3] bisimilarities for this calculus. Here we present these results in a systematic and uniform way, with examples allowing for comparisons between the different styles of bisimulation. In particular, we compare bisimilarities to Kameyama and Hasegawa's direct style axiomatization of $\lambda_{\mathcal{S}}$ [39], and we use these axioms as examples throughout the paper. We consider two semantics for $\lambda_{\mathcal{S}}$, one that is faithful to its defining CPS translation, where terms are evaluated within an outermost reset (we call it the "original semantics"), and another one where this requirement is lifted (we call it the "relaxed semantics"). Finally, we discuss how this work can be extended to other delimited-control operators.

Structure of the article. Section 2 presents the syntax and semantics of the calculus $\lambda_{\mathcal{S}}$ with shift and reset that we use in this paper. We also recall the definition of CPS equivalence, a CPS-based equivalence between terms, and its axiomatization. Section 3 discusses the definition of a contextual equivalence for $\lambda_{\mathcal{S}}$, and its relationship with CPS equivalence. We look for (at least sound) alternatives of this contextual equivalence by considering several styles of bisimilarities: applicative in Section 4, environmental in Section 5, and normal-form in Section 6. Section 7 discusses the possible extensions of our work to other semantics and other calculi with delimited control, and Section 8 concludes this paper. In particular, we summarize in Figure 11 the relationships between all the behavioral equivalences defined in this paper. We discuss related work—in particular, our own previous work—in the relevant sections, e.g., related work on applicative bisimilarities for control operators is discussed at the beginning of Section 4.

Notations and basic definitions. We use the following notations frequently throughout the paper. We write $\stackrel{\mathsf{def}}{=}$ for a defining equality, i.e., $m \stackrel{\mathsf{def}}{=} e$ means that m is defined as the expression e. Given a metavariable m, we write $\overrightarrow{m}$ for a sequence of entities denoted by m. Given a binary relation $\mathcal{R}$, we write $m \mathrel{\mathcal{R}} m'$ for $(m, m') \in \mathcal{R}$, $\mathcal{R}^{-1}$ for its inverse, defined as $\mathcal{R}^{-1} \stackrel{\mathsf{def}}{=} \{(m', m) \mid m \mathrel{\mathcal{R}} m'\}$, and $\mathcal{R}^*$ for its transitive and reflexive closure, defined as

$$\mathcal{R}^* \stackrel{\mathsf{def}}{=} \{(m, m') \mid \exists k, m_1, \ldots, m_k, k \geq 0 \wedge m = m_0 \wedge m_k = m' \wedge \forall 0 \leq i < k, m_i \mathrel{\mathcal{R}} m_{i+1}\}.$$

Further, given two binary relations $\mathcal{R}$ and $\mathcal{S}$ we use juxtaposition $\mathcal{R}\mathcal{S}$ for their composition, defined as $\mathcal{R}\mathcal{S} \stackrel{\mathsf{def}}{=} \{(m, m') \mid \exists m'', m \mathrel{\mathcal{R}} m'' \wedge m'' \mathrel{\mathcal{S}} m'\}$. Finally, a relation $\mathcal{R}$ is *compatible* if it is preserved by all the operators of the language, e.g., $t_0 \mathcal{R} t_1$ implies $\lambda x.t_0 \mathcal{R} \lambda x.t_1$; a relation is a *congruence* if it is a compatible equivalence relation.

2. The Calculus

In this section, we present the syntax, reduction semantics, and CPS equivalence for the language $\lambda_{\mathcal{S}}$ studied throughout this article. The operators shift and reset have been originally defined and have then been usually studied and implemented with a call-by-value semantics; e.g., almost all the references we give in Section 1 use such a semantics. We therefore choose to work with call by value in the main devolpments of this article, and only briefly discuss call by name in Section 7.2.

2.1. Syntax.

The language $\lambda_{\mathcal{S}}$ extends the call-by-value λ-calculus with the delimited-control operators shift and reset [21]. We assume we have a set of term variables, ranged over by x, y, z, and k. We use the metavariable k for shift-bound variables representing a continuation, while x, y, and z stand for the usual lambda-bound variables representing any values; we believe such a distinction helps to understand examples and reduction rules.

The syntax of terms ($\mathcal{T}$) and values ($\mathcal{V}$) is given by the following grammars:

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

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 t is *closed* if $\mathsf{fv}(t) = \emptyset$. The set of closed terms (values) is noted $\mathcal{T}_c$ ($\mathcal{V}_c$, respectively).

We distinguish several kinds of contexts, represented outside-in, as follows:

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

Regular contexts are ranged over by C. The pure evaluation contexts ($\mathcal{PC}$) (abbreviated as pure contexts),[2] 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 (E, F) with a term t produces a term, written $C[t]$ ($E[t]$, $F[t]$, respectively); the free variables of t may be captured in the process. We extend the notion of free variables to contexts (with $\mathsf{fv}(\square) = \emptyset$), and we say a context C (E, F) is *closed* if $\mathsf{fv}(C) = \emptyset$ ($\mathsf{fv}(E) = \emptyset$, $\mathsf{fv}(F) = \emptyset$, respectively). The set of closed pure contexts is noted $\mathcal{PC}_c$. In any definitions or proofs, we say a variable is *fresh* if it does not occur free in the terms or contexts under consideration.

[2] This terminology comes from Kameyama (e.g., in [39]); note that we use the metavariables of [7] for evaluation contexts, which are reversed compared to [39].

2.2. **Reduction Semantics.** The reduction semantics of $\lambda_{\mathcal{S}}$ is defined by the following rules, where $t\{v/x\}$ is the usual capture-avoiding substitution of v for x in t:

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

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. Furthermore, the reduction relation $\to_{\mathrm{v}}$ is compatible with evaluation contexts F, i.e., $F[t] \to_{\mathrm{v}} F[t']$ whenever $t \to_{\mathrm{v}} t'$. We write $t \to_{\mathrm{v}}$ when there is a t' such that $t \to_{\mathrm{v}} t'$ and we write $t \not\to_{\mathrm{v}}$ when no such t' exists.

All along the article, we use the terms $i \stackrel{\mathsf{def}}{=} \lambda x.x$, $\omega \stackrel{\mathsf{def}}{=} \lambda x.x\, x$, and $\Omega \stackrel{\mathsf{def}}{=} \omega\, \omega$ to build examples, starting with the next one.

Example 2.1. We present the sequence of reductions initiated by $\langle((\mathcal{S}k_1.i\ (k_1\ i))\, \mathcal{S}k_2.\omega)\, \Omega\rangle$. The term $\mathcal{S}k_1.i\ (k_1\ i)$ is within the pure context $E \stackrel{\mathsf{def}}{=} (\square\ \mathcal{S}k_2.\omega)\ \Omega$, enclosed in a delimiter $\langle\cdot\rangle$, so E is captured according to rule (*shift*):

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

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

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

When the captured context E is reactivated, it is not simply *concatenated* with the context $i\ \square$, but *composed* thanks to the reset enclosing E. (This operation corresponds to continuation composition in the CPS semantics of shift and reset, and it is crucially different from context concatenation [12].) As a result, the capture triggered by $\mathcal{S}k_2.\omega$ leaves the term i outside the first enclosing reset intact:

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

Because k_2 does not occur in ω, the context $(i\ \square)\ \Omega$ 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 Ω in this example.

Example 2.2 (fixed-point combinators)**.** We recall the definition of Turing's and Curry's fixed-point combinators. Let $\theta \stackrel{\mathsf{def}}{=} \lambda xy.y\,(\lambda z.x\,x\,y\,z)$ and $\delta_x \stackrel{\mathsf{def}}{=} \lambda y.x\,(\lambda z.y\,y\,z)$; then $\Theta_{\mathsf{v}} \stackrel{\mathsf{def}}{=} \theta\,\theta$ is Turing's call-by-value fixed-point combinator, and $\Delta_{\mathsf{v}} \stackrel{\mathsf{def}}{=} \lambda x.\delta_x\,\delta_x$ is Curry's call-by-value fixed-point combinator. In [20], the authors propose variants of these combinators using shift and reset. They write Turing's combinator as $\langle\theta\ \mathcal{S}k.k\ k\rangle$ and Curry's combinator as $\lambda x.\langle\delta_x\ \mathcal{S}k.k\ k\rangle$. For an example, the following reduction sequence demonstrates the behavior of the former:

$$\langle\theta\ \mathcal{S}k.k\ k\rangle \to_{\mathrm{v}} \langle(\lambda x.\langle\theta\ x\rangle)\ (\lambda x.\langle\theta\ x\rangle)\rangle \to_{\mathrm{v}}^{*} \lambda y.y\ (\lambda z.(\lambda x.\langle\theta\ x\rangle)\ (\lambda x.\langle\theta\ x\rangle)\ y\ z)$$

We use the combinators and their delimited-control variants as examples for the equivalence proof techniques we define throughout the paper.

Remark 2.3. The context capture can also be written using local reduction rules [26], where the context is consumed piece by piece. We discuss these reduction rules and their consequences on the results of this article in Section 7.1.

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

Definition 2.4. 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.

Proposition 2.5. *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.*

Sketch. The "if" part is straightforward. The "only if" part is by induction on t; we detail the application case $t_0\, t_1$. If t_0 is stuck, we can conclude with the induction hypothesis. Otherwise, t_0 is a value. If t_1 is stuck, then we can conclude with the induction hypothesis again. If t_1 is a value v_1, then t_0 is not a λ-abstraction, so we have an open-stuck term $x\, v_1$ for some x. □

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

Definition 2.6. 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.

Proposition 2.7. *For all terms t, either t is a normal form, or there exist a unique redex r and a unique context F such that $t = F[r]$.*

Sketch. By induction on t. For example, in the case $t = \langle t'\rangle$, either t' itself is reducing, and we can conclude with the induction hypothesis, or t' is a value v, and we have a redex $\langle v\rangle$ in the empty context. □

Finally, we define the evaluation relation of $\lambda_{\mathcal{S}}$ as follows.

Definition 2.8. We write $t \Downarrow_{\mathrm{v}} t'$ if $t \to_{\mathrm{v}}^{*} t'$ and t' is a normal form.

If a term t admits an infinite reduction sequence, like Ω, we say it *diverges*, written $t \Uparrow_{\mathrm{v}}$.

In the rest of the paper, we use the following results on the reduction (or evaluation) of terms: a control stuck term cannot be obtained from a term of the form $\langle t\rangle$, and reduction is preserved by substitution.

Proposition 2.9. *If $\langle t\rangle \Downarrow_{\mathrm{v}} t'$ then t' is a value or an open stuck term of the form $\langle F[x\, v]\rangle$. (If t is closed then t' can only be a closed value.)*

Sketch. By case analysis on the reduction rules, $\langle t\rangle \to_{\mathrm{v}} t'$ implies t' is a value or $t' = \langle t''\rangle$ for some t''. So if $\langle t\rangle \Downarrow_{\mathrm{v}} t'$ then t' is a value, or a normal form $\langle t''\rangle$ for some t''. By Proposition 2.5, if t' is not a value, it is either control-stuck or open-stuck, but a control-stuck term cannot have an outermost reset, so t' is necessarily open-stuck. □

Proposition 2.10. *If $t \rightarrow_v t'$, then $t\{v/x\} \rightarrow_v t'\{v/x\}$.*

Sketch. By case analysis on the reduction rules. □

2.3. **The original reduction semantics.** Let us notice that the reduction semantics we have introduced does not require terms to be evaluated within a top-level reset—a requirement that is commonly relaxed in practical implementations of shift and reset [25, 27], but also in some other studies of these operators [5, 38]. This is in contrast to the original reduction semantics for shift and reset [7] that has been obtained from the 2-layered continuation-passing-style (CPS) semantics [21], discussed in Section 2.4. A consequence of the correspondence with the CPS-based semantics is that terms in the original reduction semantics are treated as complete programs and are decomposed into triples consisting of a subterm (a value or a redex), a delimited context, and a meta-context (a list of delimited contexts), resembling abstract machine configurations. Such a decomposition imposes the existence of an implicit top-level reset, hard-wired in the decomposition, surrounding any term to be evaluated.

While the relaxed semantics scales better to calculi with multiple prompts [25], the original one lends itself to a generalization to a hierarchy of delimited-control operators [7]; see Section 7.6 for more details about these extensions. The two semantics differ in that the original semantics does not allow for control-stuck terms. However, it can be easily seen that operationally the difference is not essential—they are equivalent when it comes to terms of the form $\langle t \rangle$. In the rest of the article we call such terms *delimited terms* and we use the relaxed semantics when analyzing their behavior.

The top-level reset requirement, imposed by the original semantics, does not lend itself naturally to the notion of applicative bisimulation that we propose for the relaxed semantics in Section 4. We show, however, that the requirement can be successfully treated in the framework of environmental and normal-form bisimulations, presented in Sections 5.4 and 6.4.

2.4. **CPS Equivalence.** The operators shift and reset have been originally defined by a translation into continuation-passing style [21] that we present in Figure 1. Translated terms expect two continuations: the delimited continuation representing the rest of the computation up to the dynamically nearest enclosing delimiter, and the metacontinuation representing the rest of the computation beyond this delimiter. In the first three equations the metacontinuation k_2 could be η-reduced, yielding Plotkin's familiar CBV CPS translation [72]. In the equation for reset, the current delimited continuation k_1 is moved to the metacontinuation and the delimited term receives the initial delimited continuation. In the equation for shift, the current continuation is captured (and reinitialized) as a lambda abstraction that when applied pushes the then-current delimited continuation on the metacontinuation, and applies the captured continuation to the argument. A CPS-transformed program is run with the initial delimited continuation γ and the identity metacontinuation.

The CPS translation for $\lambda_{\mathcal{S}}$ induces the following notion of equivalence on $\lambda_{\mathcal{S}}$ terms.

Definition 2.11. Two terms t and t' are CPS equivalent, written $t \equiv t'$, if their CPS translations are $\beta\eta$-convertible, where $\beta\eta$-convertibility is the smallest congruence containing the relations $\rightarrow_\beta$ and $\rightarrow_\eta$:

$$\begin{array}{rcll} (\lambda x.t)\, t' & \rightarrow_\beta & t\{t'/x\} & \\ \lambda x.t\, x & \rightarrow_\eta & t & \text{if } x \notin \mathsf{fv}(t) \end{array}$$

$$
\begin{array}{rcl}
\overline{x} & = & \lambda k_1 k_2 . k_1\, x\, k_2 \\
\overline{\lambda x.t} & = & \lambda k_1 k_2 . k_1\, (\lambda x.\overline{t})\, k_2 \\
\overline{t_0\, t_1} & = & \lambda k_1 k_2 . \overline{t_0}\, (\lambda x_0 k_2' . \overline{t_1}\, (\lambda x_1 k_2'' . x_0\, x_1\, k_1\, k_2'')\, k_2')\, k_2 \\
\overline{\langle t \rangle} & = & \lambda k_1 k_2 . \overline{t}\, \gamma\, (\lambda x . k_1\, x\, k_2) \\
\overline{\mathcal{S} k.t} & = & \lambda k_1 k_2 . \overline{t}\{(\lambda x_1 k_1' k_2' . k_1\, x_1\, (\lambda x_2 . k_1'\, x_2\, k_2'))/k\}\, \gamma\, k_2 \\
 & & \text{with } \gamma = \lambda x k_2 . k_2\, x
\end{array}
$$

Figure 1: Definitional CPS translation of $\lambda_{\mathcal{S}}$

$$
\begin{array}{rcll}
(\lambda x.t)\, v & = & t\{v/x\} & \beta_{\mathtt{v}} \\
(\lambda x.E[x])\, t & = & E[t] \text{ if } x \notin \mathsf{fv}(E) & \beta_{\Omega} \\
\langle E[\mathcal{S}k.t] \rangle & = & \langle t\{\lambda x.\langle E[x] \rangle / k\} \rangle \text{ if } x \notin \mathsf{fv}(E) & \langle \cdot \rangle_{\mathcal{S}} \\
\langle (\lambda x.t_0)\, \langle t_1 \rangle \rangle & = & (\lambda x.\langle t_0 \rangle)\, \langle t_1 \rangle & \langle \cdot \rangle_{\mathtt{lift}} \\
\langle v \rangle & = & v & \langle \cdot \rangle_{\mathtt{val}} \\
\mathcal{S}k.\langle t \rangle & = & \mathcal{S}k.t & \mathcal{S}_{\langle \cdot \rangle} \\
\lambda x.v\, x & = & v \text{ if } x \notin \mathsf{fv}(v) & \eta_{\mathtt{v}} \\
\mathcal{S}k.k\, t & = & t \text{ if } k \notin \mathsf{fv}(t) & \mathcal{S}_{\mathtt{elim}}
\end{array}
$$

Figure 2: Kameyama and Hasegawa's axiomatization of $\lambda_{\mathcal{S}}$

For example, the reduction rules $t \rightarrow_{\mathrm{v}} t'$ given in Section 2.2 are sound w.r.t. CPS because CPS translating t and t' yields $\beta\eta$-convertible terms in the λ-calculus. The CPS equivalence has been characterized in terms of direct-style equations by Kameyama and Hasegawa, who developed a sound and complete axiomatization of shift and reset [39]: two terms are CPS equivalent iff one can derive their equality using the equations of Figure 2.

The axiomatization is a source of examples for the bisimulation techniques that we study in Sections 4, 5, and 6, and it allows us to relate the notion of CPS equivalence to the notions of contextual equivalence that we introduce in Section 3. In particular, we show that all but one axiom are validated by the bisimilarities for the relaxed semantics, and that all the axioms are validated by the equivalences of the original semantics. The discriminating axiom that confirms the discrepancy between the two semantics is $\mathcal{S}_{\mathtt{elim}}$—the only equation that hinges on the existence of the top-level reset.

It might be possible to consider alternative CPS translations for shift and reset, e.g., as given in [63], that correspond to the relaxed semantics. Such CPS translations require a recursive structure of continuations, which makes it hard to reason about the image of the translations, and, moreover, the operational correspondence between the relaxed semantics and such CPS translations is not as tight as between the original semantics and the original CPS translation considered in this section. Devising a respective axiomatization to be validated by the bisimilarity theories presented in this work is a research path beyond the scope of the present article.

3. Contextual Equivalence

Studying the behavioral theory of a calculus usually starts by the definition of a Morris-style contextual equivalence [67]. As usual, the idea is to express that two terms are equivalent if and only if they cannot be distinguished when put in an arbitrary context. The question is then which behaviors to observe in $\lambda_{\mathcal{S}}$ for each of the two semantics considered in this paper.

3.1. Definition for the Relaxed Semantics.

We first discuss the definition of contextual equivalence for closed terms, before extending it to open terms. As in the regular λ-calculus, we could observe only if a term reduces to a value or not, leading to the following relation.

Definition 3.1. Let t_0, t_1 be closed terms. We write $t_0 \mathrel{\mathscr{C}^1} t_1$ if for all closed C, $C[t_0] \Downarrow_{\mathrm{v}} v_0$ for some v_0 implies $C[t_1] \Downarrow_{\mathrm{v}} v_1$ for some v_1, and conversely for $C[t_1]$.

But in $\lambda_{\mathcal{S}}$, the evaluation of closed terms generates not only values, but also control stuck terms. Taking this into account, a more fine-grained definition of contextual equivalence would be as follows.

Definition 3.2. Let t_0, t_1 be closed terms. We write $t_0 \mathrel{\mathscr{C}^2} t_1$ if for all closed C,
- $C[t_0] \Downarrow_{\mathrm{v}} v_0$ for some v_0 iff $C[t_1] \Downarrow_{\mathrm{v}} v_1$ for some v_1;
- $C[t_0] \Downarrow_{\mathrm{v}} t_0'$ for some control stuck term t_0' iff $C[t_1] \Downarrow_{\mathrm{v}} t_1'$ for some control stuck term t_1'.

This definition can actually be formulated in a simpler way, where we do not distinguish cases based on the possible normal forms.

Proposition 3.3. *We have $t_0 \mathrel{\mathscr{C}^2} t_1$ iff for all closed C, $C[t_0] \Downarrow_{\mathrm{v}}$ iff $C[t_1] \Downarrow_{\mathrm{v}}$.*

Proof. Suppose that $C[t_0] \Downarrow_{\mathrm{v}}$ iff $C[t_1] \Downarrow_{\mathrm{v}}$ holds. We prove that we have $t_0 \mathrel{\mathscr{C}^2} t_1$ (the reverse implication is immediate). Assume there exists C such that $C[t_0] \Downarrow_{\mathrm{v}} t_0'$ with t_0' control stuck, and $C[t_1] \Downarrow_{\mathrm{v}} v_1$. Then $C[t_0]\,\Omega \Downarrow_{\mathrm{v}} t_0'\,\Omega$ ($t_0'\,\Omega$ is control stuck), and $C[t_1]\,\Omega \to_{\mathrm{v}}^* v_1\,\Omega \Uparrow_{\mathrm{v}}$. The context $C\,\Omega$ distinguishes t_0 and t_1, hence a contradiction. Therefore, if $C[t_0]$ evaluates to a control stuck term, then so does $C[t_1]$, and similarly for evaluation to values. $\square$

By the definitions, it is clear that $\mathscr{C}^2 \subseteq \mathscr{C}^1$. The inclusion is strict, because of terms such as $\mathcal{S}k.\Omega$, which are control-stuck terms but diverge when unstuck. Indeed, we have $\mathcal{S}k.\Omega \mathrel{\not\mathscr{C}^2} \Omega$, because $\mathcal{S}k.\Omega$ is a stuck term, but not Ω and, therefore, the second item of Definition 3.2 is violated. However, they are related by $\mathscr{C}^1$.

Proposition 3.4. *We have $\mathcal{S}k.\Omega \mathrel{\mathscr{C}^1} \Omega$.*

Proof. Let C be such that $C[\mathcal{S}k.\Omega] \Downarrow_{\mathrm{v}} v_0$ for some v_0. Then we prove that $C[\Omega]$ reduces to a value as well; in fact, C does not evaluate the term that fills its hole. We define multi-holes contexts H by the following grammar

$$H ::= \square \mid x \mid \lambda x.H \mid H\ H \mid \mathcal{S}k.H \mid \langle H \rangle$$

and we write $H[t]$ for the plugging of t in all the holes of H. We show that $(*)$ for all H and v, $H[\mathcal{S}k.\Omega] \Downarrow_{\mathrm{v}} v$ implies that there exists H' such that $v = \lambda x.H'[\mathcal{S}k.\Omega]$ and $H[\Omega] \Downarrow_{\mathrm{v}} \lambda x.H'[\Omega]$.

We proceed by induction on the number of steps n in the evaluation; the proof is straightforward if $n = 0$. Suppose $n > 0$; then $H[\mathcal{S}k.\Omega] \to_{\mathrm{v}} t' \Downarrow_{\mathrm{v}} v$ for some t'. Suppose a copy of $\mathcal{S}k.\Omega$ is in an evaluation context F in $H[\mathcal{S}k.\Omega]$. The context F cannot be pure, because $H[\mathcal{S}k.\Omega]$ reduces, so $F = F'[\langle E \rangle]$, which implies $t' = F'[\langle \Omega \rangle]$; this contradicts $t' \Downarrow_{\mathrm{v}} v$ (the calculus is deterministic). Consequently, the copies of $\mathcal{S}k.\Omega$ are not in an evaluation context in $H[\mathcal{S}k.\Omega]$, and the reduction $H[\mathcal{S}k.\Omega] \rightarrow_{\mathrm{v}} t'$ can be written $H[\mathcal{S}k.\Omega] \rightarrow_{\mathrm{v}} H''[\mathcal{S}k.\Omega]$ for some H'', and we also have $H[\Omega] \rightarrow_{\mathrm{v}} H''[\Omega]$. We can then conclude by applying the induction hypothesis on $H''[\mathcal{S}k.\Omega] \Downarrow_{\mathrm{v}} v$.

Applying the property $(*)$ with $H = C$, we get that $C[\mathcal{S}k.\Omega] \Downarrow_{\mathrm{v}} v_0$ implies $C[\Omega] \Downarrow_{\mathrm{v}} v$ for some v. Conversely, if $C[\Omega] \Downarrow_{\mathrm{v}} v_1$, we can prove that $C[\mathcal{S}k.\Omega] \Downarrow_{\mathrm{v}} v$ for some v using the same reasoning. Therefore, we have $\mathcal{S}k.\Omega \mathrel{\mathscr{C}^1} \Omega$. $\square$

We work with $\mathscr{C}^2$ as the main contextual equivalence for the relaxed semantics, since it corresponds to the usual definition of contextual equivalence in languages similar to the λ-calculus, where we simply observe termination [1] (see Proposition 3.3). Henceforth, we simply write $\mathscr{C}$ for $\mathscr{C}^2$.

We extend $\mathscr{C}$ to open terms using closing substitutions: we say σ closes t if it maps the free variables of t to closed values. We define the *open extension* of a relation as follows.

Definition 3.5. Let $\mathcal{R}$ be a relation on closed terms, and t_0 and t_1 be open terms. We write $t_0 \mathrel{\mathcal{R}^\circ} t_1$ if for every substitution σ which closes t_0 and t_1, $t_0\sigma \mathrel{\mathcal{R}} t_1\sigma$ holds.

Remark 3.6. Contextual equivalence can be defined directly on open terms by requiring that the context C binds the free variables of the related terms. We prove the resulting relation is equal to $\mathscr{C}^\circ$ in Section 4.2.

To prove completeness of bisimilarities, we use a variant of $\mathscr{C}$ which takes into account only evaluation contexts to compare terms.

Definition 3.7. Let t_0, t_1 be closed terms. We write $t_0 \mathrel{\mathscr{D}} t_1$ if for all closed F,

- $F[t_0] \Downarrow_{\mathrm{v}} v_0$ for some v_0 iff $F[t_1] \Downarrow_{\mathrm{v}} v_1$ for some v_1;
- $F[t_0] \Downarrow_{\mathrm{v}} t'_0$ for some control stuck term t'_0 iff $F[t_1] \Downarrow_{\mathrm{v}} t'_1$ for some control stuck term t'_1.

The definitions imply $\mathscr{C} \subseteq \mathscr{D}$. While proving completeness of applicative bisimilarity in Section 4, we also prove $\mathscr{D} = \mathscr{C}$, which means that testing with evaluation contexts is as discriminating as testing with any contexts. Such a simplification result is similar to Milner's context lemma [65].

The relations $\mathscr{C}^1$ and $\mathscr{C}^2$ are not suitable for the original semantics, because they distinguish terms that should be equated according to Kameyama and Hasegawa's axiomatization. Indeed, according to these relations, $\mathcal{S}k.k\,v$ (where $k \notin \mathsf{fv}(v)$) cannot be related to v (axiom $\mathcal{S}_{\texttt{elim}}$ in Figure 2), because a stuck term cannot be related to a value. In the next section, we discuss a definition of contextual equivalence for the original semantics.

3.2. Definition for the Original Semantics.

Terms are evaluated in the original semantics within an enclosing reset, so the corresponding contextual equivalence should test terms in contexts of the form $\langle C \rangle$ only. Because delimited terms cannot reduce to stuck terms (Proposition 2.9), the only possible observable action is evaluation to values. We, therefore, define contextual equivalence for the original semantics as follows.

Definition 3.8. Let t_0, t_1 be closed terms. We write $t_0 \mathrel{\mathscr{P}} t_1$ if for all closed C, $\langle C[t_0] \rangle \Downarrow_{\mathrm{v}} v_0$ for some v_0 iff $\langle C[t_1] \rangle \Downarrow_{\mathrm{v}} v_1$ for some v_1.

The relation $\mathscr{P}$ is defined on all (closed) terms, not just delimited ones. The resulting relation is less discriminating than $\mathscr{C}$, because $\mathscr{P}$ uses contexts of a particular form, while $\mathscr{C}$ tests with all contexts.

Proposition 3.9. *We have* $\mathscr{C} \subsetneq \mathscr{P}$.

As a result, any equivalence between terms we prove for the relaxed semantics also holds in the original semantics, and any bisimilarity sound w.r.t. $\mathscr{C}$ (like the bisimilarities we define in Sections 4, 5.3, and 6) is also sound w.r.t. $\mathscr{P}$. However, to reach completeness, we have to design a bisimilarity suitable for delimited terms (see Section 5.4). As for the relaxed semantics, we extend $\mathscr{P}$ to open terms using Definition 3.5.

The inclusion of Proposition 3.9 is strict because, e.g., $\mathscr{P}$ verifies the axiom $\mathcal{S}_{\mathtt{elim}}$, while $\mathscr{C}$ does not. In fact, we prove in Section 5.7 that $\mathscr{P}$ contains the CPS equivalence $\equiv$. The reverse inclusion holds neither for $\mathscr{P}$ nor $\mathscr{C}$: there exist contextually equivalent terms that are not CPS equivalent.

Proposition 3.10. (1) *We have* $\Omega \mathrel{\mathscr{P}} \Omega\,\Omega$ *(respectively* $\Omega \mathrel{\mathscr{C}} \Omega\,\Omega$*), but* $\Omega \not\equiv \Omega\,\Omega$.
(2) *We have* $\Theta_{\mathsf{v}} \mathrel{\mathscr{P}} \Delta_{\mathsf{v}}$ *(respectively* $\Theta_{\mathsf{v}} \mathrel{\mathscr{C}} \Delta_{\mathsf{v}}$*), but* $\Theta_{\mathsf{v}} \not\equiv \Delta_{\mathsf{v}}$.

The contextual equivalences $\mathscr{C}$ and $\mathscr{P}$ put all diverging terms in one equivalence class, while CPS equivalence is more discriminating. Furthermore, as is usual with equational theories for λ-calculi, CPS equivalence is not strong enough to equate Turing's and Curry's (call-by-value) fixed-point combinators.

As explained in the introduction, contextual equivalence is difficult to prove in practice for two given terms because of the quantification over contexts. We look for a suitable replacement (that is, an equivalence that is at least sound w.r.t. $\mathscr{C}$ or $\mathscr{P}$) by studying different styles of bisimulation in the next sections.

4. Applicative Bisimilarity

Applicative bisimilarity has been originally defined for the lazy λ-calculus [1]. The main idea is to reduce (closed) terms to values, and then compare the resulting λ-abstractions by applying them to an arbitrary argument. When sound, applicative bisimilarity for deterministic languages is usually also complete (see, e.g., [31, 95, 30]), and soundness is proved thanks to a systematic technique called Howe's method [35, 30]. However, defining and proving sound the most powerful up-to techniques, such as bisimulation up to context, remain an open issue for applicative bisimilarity.

Very few works study applicative bisimilarity in a calculus with control. Merro [64] defines an applicative bisimilarity which characterizes contextual equivalence in the *CPS calculus* [94], a minimal calculus which models the control features of functional languages with imperative jumps. In the $\lambda\mu$-calculus, Lassen [51] proposes a sound but not complete applicative bisimilarity in call-by-name. We improve this result [16] by defining sound and complete applicative bisimilarities in both call-by-name and call-by-value.

In this section, we define a sound and complete applicative bisimilarity for the relaxed semantics of $\lambda_{\mathcal{S}}$. Our definition of applicative bisimilarity relies on a labeled transition system, introduced first (Section 4.1). We then prove its soundness and completeness in Section 4.2, before showing how it can be used on the $\lambda_{\mathcal{S}}$ axiomatization (Section 4.3). We cover results that have been originally presented in [13].

$$\frac{}{(\lambda x.t)\, v \xrightarrow{\tau} t\{v/x\}}\ (\beta_{\mathsf{v}}) \qquad \frac{t_0 \xrightarrow{\tau} t_0'}{t_0\, t_1 \xrightarrow{\tau} t_0'\, t_1}\ (\mathit{left}_\tau) \qquad \frac{t \xrightarrow{\tau} t'}{v\, t \xrightarrow{\tau} v\, t'}\ (\mathit{right}_\tau) \qquad \frac{}{\langle v \rangle \xrightarrow{\tau} v}\ (\mathit{reset})$$

$$\frac{t \xrightarrow{\tau} t'}{\langle t \rangle \xrightarrow{\tau} \langle t' \rangle}\ (\langle\cdot\rangle_\tau) \qquad \frac{t \xrightarrow{\Box} t'}{\langle t \rangle \xrightarrow{\tau} t'}\ (\langle\cdot\rangle_{\mathcal{S}}) \qquad \frac{}{\lambda x.t \xrightarrow{v} t\{v/x\}}\ (\mathit{val})$$

$$\frac{x \notin \mathsf{fv}(E)}{\mathcal{S}k.t \xrightarrow{E} \langle t\{\lambda x.\langle E[x]\rangle/k\}\rangle}\ (\mathit{shift}) \qquad \frac{t_0 \xrightarrow{E[\Box\, t_1]} t_0'}{t_0\, t_1 \xrightarrow{E} t_0'}\ (\mathit{left}_{\mathcal{S}}) \qquad \frac{t \xrightarrow{E[v\, \Box]} t'}{v\, t \xrightarrow{E} t'}\ (\mathit{right}_{\mathcal{S}})$$

Figure 3: Labeled Transition System

4.1. **Applicative Bisimilarity.** One possible way to define an applicative bisimilarity is to rely on a labeled transition system (LTS), where the possible interactions of a term with its environment are encoded in the labels (see, e.g., [31, 30]). Using a LTS simplifies the definition of the bisimilarity and makes it easier to use some techniques in proofs, such as diagram chasing. In Figure 3, we define a LTS $t_0 \xrightarrow{\alpha} t_1$ with three kinds of transitions, where we assume all the terms to be closed. An *internal action* $t \xrightarrow{\tau} t'$ is an evolution from t to t' without any help from the surrounding context; it corresponds to a reduction step from t to t'. The transition $v_0 \xrightarrow{v_1} t$ expresses the fact that v_0 needs to be applied to another value v_1 to evolve, reducing to t. Finally, the transition $t \xrightarrow{E} t'$ means that t is control stuck, and when t is put in a context E enclosed in a reset, the capture can be triggered, the result of which being t'.

Most rules for internal actions (Figure 3) are straightforward; the rules (β_{v}) and (*reset*) mimic the corresponding reduction rules, and the compositional rules (right_τ), (left_τ), and ($\langle\cdot\rangle_\tau$) allow internal actions to happen within any evaluation context. The rule ($\langle\cdot\rangle_{\mathcal{S}}$) for context capture is explained later. Rule (*val*) defines the only possible transition for values. While both rules (β_{v}) and (*val*) encode β-reduction, they are quite different in nature; in the former, the term $(\lambda x.t)\, v$ can evolve by itself, without any help from the surrounding context, while the latter expresses the possibility for $\lambda x.t$ to evolve only if a value v is provided by the environment.

The rules for context capture are built following the principles of complementary semantics developed in [59]. The label of the transition $t \xrightarrow{E} t'$ contains what the environment needs to provide (a context E, but also an enclosing reset, left implicit) for the control stuck term t to reduce to t'. Hence, the transition $t \xrightarrow{E} t'$ means that we have $\langle E[t]\rangle \xrightarrow{\tau} t'$ by context capture. For example, in the rule (*shift*), the result of the capture of E by $\mathcal{S}k.t$ is $\langle t\{\lambda x.\langle E[x]\rangle/k\}\rangle$.

In rule ($\mathit{left}_{\mathcal{S}}$), we want to know the result of the capture of E by the term $t_0\, t_1$, assuming t_0 contains a shift ready to perform the capture. Under this hypothesis, the capture of E by $t_0\, t_1$ comes from the capture of $E[\Box\, t_1]$ by t_0. Therefore, as a premise of the rule ($\mathit{left}_{\mathcal{S}}$), we check that t_0 is able to capture $E[\Box\, t_1]$, and the result t_0' of this transition is exactly the result we want for the capture of E by $t_0\, t_1$. The rule ($\mathit{right}_{\mathcal{S}}$) follows the same pattern. Finally, a control stuck term t enclosed in a reset is able to perform an internal action (rule $(\langle\cdot\rangle_{\mathcal{S}})$); we obtain the result t' of the transition $\langle t\rangle \xrightarrow{\tau} t'$ by letting t capture the empty context, i.e., by considering the transition $t \xrightarrow{\Box} t'$.

Example 4.1. We illustrate how the LTS handles capture by considering the transition from $\langle (i\ \mathcal{S}k.\omega)\ \Omega\rangle$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathcal{S}k.\omega \xrightarrow{(i\,\Box)\ \Omega} \langle\omega\rangle}\ (shift)}{i\ \mathcal{S}k.\omega \xrightarrow{\Box\ \Omega} \langle\omega\rangle}\ (right_{\mathcal{S}})}{(i\ \mathcal{S}k.\omega)\ \Omega \xrightarrow{\Box} \langle\omega\rangle}\ (left_{\mathcal{S}})}{\langle (i\ \mathcal{S}k.\omega)\ \Omega\rangle \xrightarrow{\tau} \langle\omega\rangle}\ (\langle\cdot\rangle_{\mathcal{S}})$$

Reading the tree from bottom to top, we see that the rules $(\langle\cdot\rangle_{\mathcal{S}})$, $(left_{\mathcal{S}})$, and $(right_{\mathcal{S}})$ build the captured context in the label by deconstructing the initial term. Indeed, the rule $(\langle\cdot\rangle_{\mathcal{S}})$ removes the outermost reset and initiates the context in the label with $\Box$. The rules $(left_{\mathcal{S}})$ and $(right_{\mathcal{S}})$ then successively remove the outermost application and store it in the context. The process continues until a shift operator is found; then we know the captured context is completed, and the rule $(shift)$ computes the result of the capture. This result is then simply propagated from top to bottom by the other rules.

The LTS corresponds to the reduction semantics $\to_{\mathrm{v}}$ and exhibits the observable terms (values and control stuck terms) of the language. The only difficulty is in the treatment of control stuck terms. The next lemma makes the correspondence between $\xrightarrow{E}$ and control stuck terms explicit.

Lemma 4.2. *If $t \xrightarrow{E} t'$, then there exist E', k, and s such that $t = E'[\mathcal{S}k.s]$ and $t' = \langle s\{\lambda x.\langle E[E'[x]]\rangle/k\}\rangle$.*

The proof is by induction on $t \xrightarrow{E} t'$. From this lemma, we can deduce the correspondence between $\xrightarrow{\tau}$ and $\to_{\mathrm{v}}$, and between $\xrightarrow{\alpha}$ (for $\alpha \neq \tau$) and the observable actions of the language.

Proposition 4.3. *The following hold:*

- *We have $\xrightarrow{\tau} = \to_{v}$.*
- *If $t \xrightarrow{E} t'$, then t is a stuck term, and $\langle E[t]\rangle \xrightarrow{\tau} t'$.*
- *If $t \xrightarrow{v} t'$, then t is a value, and $t\ v \xrightarrow{\tau} t'$.*

We write $\Rightarrow$ for the reflexive and transitive closure of $\xrightarrow{\tau}$. We define the weak delay transition[3] $\overset{\alpha}{\Rightarrow}$ as $\Rightarrow$ if $\alpha = \tau$ and as $\Rightarrow\xrightarrow{\alpha}$ otherwise. The definition of (weak delay) bisimilarity is then straightforward.

Definition 4.4. A relation $\mathcal{R}$ on closed terms is an applicative simulation if $t_0\ \mathcal{R}\ t_1$ implies that for all $t_0 \xrightarrow{\alpha} t_0'$, there exists t_1' such that $t_1 \overset{\alpha}{\Rightarrow} t_1'$ and $t_0'\ \mathcal{R}\ t_1'$. A relation $\mathcal{R}$ on closed terms is an applicative bisimulation if $\mathcal{R}$ and $\mathcal{R}^{-1}$ are applicative simulations. Applicative bisimilarity $\mathscr{A}$ is the largest applicative bisimulation.

In words, two terms are equivalent if any transition from one is matched by a weak transition with the same label from the other. Because the calculus is deterministic, it is not mandatory to test the internal steps when proving that two terms are bisimilar.

[3]A transition where internal steps are allowed before, but not after a visible action.

Proposition 4.5. *If $t \xrightarrow{\tau} t'$ (respectively $t \Downarrow_{\mathrm{v}} t'$) then $t \mathrel{\mathscr{A}} t'$.*

As a result, applicative bisimulation can be defined in terms of big-step transitions.

Definition 4.6. A relation $\mathcal{R}$ on closed terms is a big-step applicative simulation if $t_0 \mathrel{\mathcal{R}} t_1$ implies that for all $t_0 \overset{\alpha}{\Rightarrow} t'_0$ with $\alpha \neq \tau$, there exists t'_1 such that $t_1 \overset{\alpha}{\Rightarrow} t'_1$ and $t'_0 \mathrel{\mathcal{R}} t'_1$. A relation $\mathcal{R}$ on closed terms is a big-step applicative bisimulation if $\mathcal{R}$ and $\mathcal{R}^{-1}$ are big-step applicative simulations.

Proposition 4.7. *If $\mathcal{R}$ is a big-step applicative bisimulation, then $\mathcal{R} \subseteq \mathscr{A}$.*

Sketch. By showing that $\{(t_0, t_1) \mid (t_0, t_1) \in \exists t'_0, t'_1, t'_0 \to^*_{\mathrm{v}} t_0 \wedge t'_1 \to^*_{\mathrm{v}} t_1 \wedge t'_0 \mathrel{\mathcal{R}} t'_1\}$ is an applicative bisimulation. □

In this section, we drop the adjective "applicative" and refer to the two kinds of relations simply as "bisimulation" and "big-step bisimulation" where it does not cause confusion.

Example 4.8 (double reset)**.** For all closed terms t, we show that $\langle\langle t\rangle\rangle \mathrel{\mathscr{A}} \langle t\rangle$ holds by proving that $\mathcal{R} \overset{\mathsf{def}}{=} \{(\langle t\rangle, \langle\langle t\rangle\rangle) \mid t \in \mathcal{T}_c\} \cup \{(t, t) \mid t \in \mathcal{T}_c\}$ is an applicative bisimulation. First, $\langle t\rangle$ cannot be a value or a control-stuck term, so we only have to consider $\xrightarrow{\tau}$-transition. By case analysis on the reduction rules, we can see that $\langle t\rangle \xrightarrow{\tau} t'$ iff $t' = \langle t''\rangle$ for some t'', or t' is a value v.

If $\langle t\rangle \xrightarrow{\tau} \langle t''\rangle$, then $\langle\langle t\rangle\rangle \xrightarrow{\tau} \langle\langle t''\rangle\rangle$, and the resulting terms are in $\mathcal{R}$. Otherwise, if $\langle t\rangle \xrightarrow{\tau} v$, then $\langle\langle t\rangle\rangle \xrightarrow{\tau}\xrightarrow{\tau} v$ and we get identical terms. Conversely, if $\langle\langle t\rangle\rangle \xrightarrow{\tau} t'$, then we can show that either $t' = \langle\langle t''\rangle\rangle$ for some t'' and $\langle t\rangle \xrightarrow{\tau} \langle t''\rangle$, or $t' = \langle v\rangle$ for some v and $\langle t\rangle = \langle v\rangle$. This concludes the proof for the terms in the first set of $\mathcal{R}$, and checking the bisimulation game for identical terms (the second set of $\mathcal{R}$) is straightforward.

Example 4.9 (Turing's combinator)**.** We study here the relationships between Turing's and Curry's fixed-point combinator and their respective variants with delimited control [20] (see Example 2.2 for the definitions). We start with Turing's combinator Θ_{v} and its variant $\Theta_{\mathcal{S}} \overset{\mathsf{def}}{=} \langle \theta\, \mathcal{S}k.k\, k\rangle$. The two terms can perform the following transitions:

$$\Theta_{\mathsf{v}} \overset{v}{\Rightarrow} v\ (\lambda z.\theta\, \theta\, v\, z)$$
$$\Theta_{\mathcal{S}} \overset{v}{\Rightarrow} v\ (\lambda z.(\lambda x.\langle \theta\, x\rangle)\, (\lambda x.\langle \theta\, x\rangle)\, v\, z)$$

Taking $v = \lambda x.t$, we have to study $t\{(\lambda z.\theta\, \theta\, v\, z)/x\}$, and $t\{(\lambda z.(\lambda x.\langle \theta\, x\rangle)\, (\lambda x.\langle \theta\, x\rangle)\, v\, z)/x\}$. A way to proceed is by case analysis on t, the interesting case being $t = F[x\, v']$. If it is possible to conclude using applicative bisimulation, the needed candidate relation is much more complex than with environmental (Example 5.16) or normal-form (Example 6.3) bisimulations, so we refer to these examples for a complete proof.

In contrast, Curry's combinator Δ_{v} is not bisimilar to its delimited-control variant $\Delta_{\mathcal{S}} \overset{\mathsf{def}}{=} \lambda x.\langle \delta_x\, \mathcal{S}k.k\, k\rangle$. Indeed, after applying these values to an argument v, we obtain respectively $v\ (\lambda z.\delta_v\, \delta_v\, z)$ and $\langle\langle v\ (\lambda z.(\lambda y.\langle \delta_v\, y\rangle)\, (\lambda y.\langle \delta_v\, y\rangle)\, z)\rangle\rangle$, and these terms are not bisimilar if $v = \lambda x.\mathcal{S}k.\Omega$, as the first one reduces to a control-stuck term while the second one diverges.

Remark 4.10. Applicative simulation can be formulated in a more classic, but equivalent, way (without labeled transitions), as follows. A relation $\mathcal{R}$ on closed terms is an applicative simulation if $t_0 \mathrel{\mathcal{R}} t_1$ implies:

- if $t_0 \to_{\mathrm{v}} t'_0$, then there exists t'_1 such that $t_1 \to^*_{\mathrm{v}} t'_1$ and $t'_0 \mathrel{\mathcal{R}} t'_1$;
- if t_0 is a value $\lambda x.t'_0$, then there exists t'_1 such that $t_1 \to^*_{\mathrm{v}} \lambda x.t'_1$, and for all closed v, we have $t'_0\{v/x\} \mathrel{\mathcal{R}} t'_1\{v/x\}$;
- if t_0 is a stuck term $E_0[\mathcal{S}k.t'_0]$, then there exist t'_1 and E_1 such that $t_1 \to^*_{\mathrm{v}} E_1[\mathcal{S}k.t'_1]$, and for all closed E, we have $\langle t'_0\{\lambda x.\langle E[E_0[x]]\rangle/k\}\rangle \mathrel{\mathcal{R}} \langle t'_1\{\lambda x.\langle E[E_1[x]]\rangle/k\}\rangle$.

The correspondence between this formulation and Definition 4.4 is a direct consequence of Proposition 4.3.

4.2. **Soundness and Completeness.** To prove the soundness of $\mathscr{A}$ w.r.t. the contextual equivalence $\mathscr{C}$, we show that $\mathscr{A}$ is a congruence using *Howe's method*, a well-known congruence proof method initially developed for the λ-calculus [35, 30]. The idea of the method is as follows: first, define the *Howe's closure* of $\mathscr{A}$, written $\mathscr{A}^\bullet$, a relation which contains $\mathscr{A}$ and is compatible by construction. Then, prove a simulation-like property for $\mathscr{A}^\bullet$; from this result, prove that $\mathscr{A}^\bullet$ and $\mathscr{A}$ coincide on closed terms. Because $\mathscr{A}^\bullet$ is compatible, it shows that $\mathscr{A}$ is compatible as well, and therefore a congruence.

The definition of $\mathscr{A}^\bullet$ relies on the notion of *compatible refinement*; given a relation $\mathcal{R}$ on open terms, the compatible refinement $\widehat{\mathcal{R}}$ relates two terms iff they have the same outermost operator and their immediate subterms are related by $\mathcal{R}$. Formally, it is inductively defined by the following rules:

$$\frac{}{x \mathrel{\widehat{\mathcal{R}}} x} \qquad \frac{t_0 \mathrel{\mathcal{R}} t_1}{\lambda x.t_0 \mathrel{\widehat{\mathcal{R}}} \lambda x.t_1} \qquad \frac{t_0 \mathrel{\mathcal{R}} t_1 \qquad t'_0 \mathrel{\mathcal{R}} t'_1}{t_0\, t'_0 \mathrel{\widehat{\mathcal{R}}} t_1\, t'_1} \qquad \frac{t_0 \mathrel{\mathcal{R}} t_1}{\mathcal{S}k.t_0 \mathrel{\widehat{\mathcal{R}}} \mathcal{S}k.t_1} \qquad \frac{t_0 \mathrel{\mathcal{R}} t_1}{\langle t_0\rangle \mathrel{\widehat{\mathcal{R}}} \langle t_1\rangle}$$

Howe's closure $\mathscr{A}^\bullet$ is inductively defined as the smallest compatible relation containing $\mathscr{A}^\circ$ and closed under right composition with $\mathscr{A}^\circ$.

Definition 4.11. Howe's closure $\mathscr{A}^\bullet$ is the smallest relation satisfying:

$$\frac{t_0 \mathrel{\mathscr{A}^\circ} t_1}{t_0 \mathrel{\mathscr{A}^\bullet} t_1} \qquad \frac{t_0 \mathrel{\mathscr{A}^\bullet \mathscr{A}^\circ} t_1}{t_0 \mathrel{\mathscr{A}^\bullet} t_1} \qquad \frac{t_0 \mathrel{\widehat{\mathscr{A}^\bullet}} t_1}{t_0 \mathrel{\mathscr{A}^\bullet} t_1}$$

By construction, $\mathscr{A}^\bullet$ is compatible (by the third rule of the definition), and composing on the right with $\mathscr{A}^\circ$ gives some transitivity properties to $\mathscr{A}^\bullet$. In particular, we can prove that $\mathscr{A}^\bullet$ is *substitutive*: if $t_0 \mathrel{\mathscr{A}^\bullet} t_1$ and $v_0 \mathrel{\mathscr{A}^\bullet} v_1$, then $t_0\{v_0/x\} \mathrel{\mathscr{A}^\bullet} t_1\{v_1/x\}$.

Let $(\mathscr{A}^\bullet)^c$ be the restriction of $\mathscr{A}^\bullet$ to closed terms. We cannot prove directly that $(\mathscr{A}^\bullet)^c$ is a bisimulation, so we prove a stronger result. Suppose we have $t_0 \mathrel{(\mathscr{A}^\bullet)^c} t_1$; instead of simply requiring $t_0 \xrightarrow{\alpha} t'_0$ to be matched by t_1 with the same label α, we ask t_1 to be able to respond for any label α' related to α by $(\mathscr{A}^\bullet)^c$. We, therefore, extend $\mathscr{A}^\bullet$ to all labels, by adding the relation $\tau \mathrel{\mathscr{A}^\bullet} \tau$, and by defining $E \mathrel{\mathscr{A}^\bullet} E'$ as follows:

$$\frac{}{\square \mathrel{\mathscr{A}^\bullet} \square} \qquad \frac{E_0 \mathrel{\mathscr{A}^\bullet} E_1 \qquad t_0 \mathrel{\mathscr{A}^\bullet} t_1}{E_0\, t_0 \mathrel{\mathscr{A}^\bullet} E_1\, t_1} \qquad \frac{E_0 \mathrel{\mathscr{A}^\bullet} E_1 \qquad v_0 \mathrel{\mathscr{A}^\bullet} v_1}{v_0\, E_0 \mathrel{\mathscr{A}^\bullet} v_1\, E_1}$$

Lemma 4.12 (Simulation-like property)**.** *If $t_0 \mathrel{(\mathscr{A}^\bullet)^c} t_1$ and $t_0 \xrightarrow{\alpha} t'_0$, then for all $\alpha \mathrel{(\mathscr{A}^\bullet)^c} \alpha'$, there exists t'_1 such that $t_1 \xRightarrow{\alpha'} t'_1$ and $t'_0 \mathrel{(\mathscr{A}^\bullet)^c} t'_1$.*

The main difficulty when applying Howe's method is to prove this simulation-like property. The proof [13] is by induction on $t_0 \mathrel{(\mathscr{A}^\bullet)^c} t_1$, and then by case analysis on the transition $t_0 \xrightarrow{\alpha} t'_0$. Lemma 4.12 allows us to prove that $(\mathscr{A}^\bullet)^c$ is a simulation, by choosing

$\alpha' = \alpha$. We cannot directly deduce that $(\mathscr{A}^\bullet)^c$ is a bisimulation, however we can prove that its transitive and reflexive closure $((\mathscr{A}^\bullet)^c)^*$ is a bisimulation, because of the following classical property of the Howe's closure [30].

Lemma 4.13. *The relation $(\mathscr{A}^\bullet)^*$ is symmetric.*

Sketch. The proof is by induction on the definition of the reflexive and transitive closure. The inductive case is straightforward. For the base case, we show that $t_0 \mathrel{\mathscr{A}^\bullet} t_1$ implies $t_1 (\mathscr{A}^\bullet)^* t_0$ by induction on the definition of Howe's closure. Most cases are straightforward using the induction hypothesis. The interesting case is when $t_0 \mathrel{\mathscr{A}^\bullet} t \mathrel{\mathscr{A}^\circ} t_1$ for some t. By the induction hypothesis, we have $t (\mathscr{A}^\bullet)^* t_0$. Because $\mathscr{A}$ itself is symmetric, we also have $t_1 \mathrel{\mathscr{A}^\circ} t$, which implies $t_1 \mathrel{\mathscr{A}^\bullet} t$, which when combined with $t (\mathscr{A}^\bullet)^* t_0$ gives the required result. □

The fact that $((\mathscr{A}^\bullet)^c)^*$ is a bisimulation implies that $((\mathscr{A}^\bullet)^c)^* \subseteq \mathscr{A}$. Because $\mathscr{A} \subseteq (\mathscr{A}^\bullet)^c \subseteq ((\mathscr{A}^\bullet)^c)^*$ holds by construction, we can deduce $\mathscr{A} = (\mathscr{A}^\bullet)^c$. Since $(\mathscr{A}^\bullet)^c$ is compatible, and we can easily show that $\mathscr{A}$ is transitive and reflexive, we have the following result.

Theorem 4.14. *The relation $\mathscr{A}$ is a congruence.*

Combined with the fact that labels correspond to observable actions (Proposition 4.3), Theorem 4.14 entails that $\mathscr{A}$ is sound w.r.t. contextual equivalence.

Corollary 4.15. *We have $\mathscr{A} \subseteq \mathscr{C}$.*

Completeness and context lemma. For the reverse inclusion, we use $\mathscr{D}$, the contextual equivalence which tests with contexts F only (see Definition 3.7). We can prove that $\mathscr{A}$ is complete w.r.t. $\mathscr{D}$, by showing that $\mathscr{D}$ is an applicative bisimulation [13].

Theorem 4.16. *We have $\mathscr{D} \subseteq \mathscr{A}$.*

Sketch. We show that $\mathscr{D}$ is an applicative bisimulation. Let $t_0 \mathrel{\mathscr{D}} t_1$. If $t_0 \xrightarrow{\tau} t'_0$, it is easy to check that we still have $t'_0 \mathrel{\mathscr{D}} t_1$. If $t_0 \xrightarrow{v} t'_0$, then by Proposition 4.3, t_0 is a value and $t_0\, v \rightarrow_{\mathrm{v}} t'_0$. Because $t_0 \mathrel{\mathscr{D}} t_1$, there exists v_1 such that $t_1 \rightarrow_{\mathrm{v}}^* v_1$, therefore $t_1 \overset{v}{\Rightarrow} t'_1$ for t'_1 such that $v_1\, v \rightarrow_{\mathrm{v}} t'_1$. What is left to prove is that $t'_0 \mathrel{\mathscr{D}} t'_1$, i.e., for all F, $F[t'_0]$ behaves like $F[t'_1]$ (i.e., one evaluates to respectively a value or stuck term iff the other do so as well). But from $t_0 \mathrel{\mathscr{D}} t_1$, we get that $F'[t_0]$ behaves like $F'[t_1]$ for all F', so in particular for $F' = F[\Box\, v]$. In the end, $F[t_0\, v]$ behaves like $F[t_1\, v]$, but these terms reduces to respectively $F[t'_0]$ and $F[t'_1]$, so we can conclude from there. The reasoning is the same for $t_0 \xrightarrow{E} t'_0$. □

As a result, the relations $\mathscr{C}$, $\mathscr{D}$, and $\mathscr{A}$ coincide, which means that $\mathscr{A}$ is complete w.r.t. $\mathscr{C}$

Corollary 4.17. *We have $\mathscr{C} = \mathscr{D} = \mathscr{A}$.*

Indeed, we have $\mathscr{D} \subseteq \mathscr{A}$ (Theorem 4.16), $\mathscr{A} \subseteq \mathscr{C}$ (Corollary 4.15), and $\mathscr{C} \subseteq \mathscr{D}$ (by definition).

This equality also allows us to prove that we can formulate the open extension of $\mathscr{C}$ using capturing contexts.

Proposition 4.18. *We have $t_0 \mathrel{\mathscr{C}^\circ} t_1$ iff for all C capturing the variables of t_0 and t_1, the following holds:*

- $C[t_0] \Downarrow_{\mathrm{v}} v_0$ *iff* $C[t_1] \Downarrow_{\mathrm{v}} v_1$*;*
- $C[t_0] \Downarrow_{\mathrm{v}} t'_0$*, where* t'_0 *is control stuck, iff* $C[t_1] \Downarrow_{\mathrm{v}} t'_1$*, with* t'_1 *control stuck as well.*

Proof. Suppose $t_0 \mathscr{C}^\circ t_1$. Then $t_0 \mathscr{A}^\circ t_1$, and because $\mathscr{A}^\circ$ is a congruence, for all C capturing the variables of t_0 and t_1, we have $C[t_0] \mathscr{A} C[t_1]$. We have $C[t_0] \Downarrow_{\mathrm{v}} v_0$ iff $C[t_1] \Downarrow_{\mathrm{v}} v_1$ by bisimilarity definition, and similarly with $C[t_0] \Downarrow_{\mathrm{v}} t_0'$, where t_0' is control stuck.

For the reverse implication, suppose that for all C capturing the variables of t_0 and t_1, the two items of the proposition hold. Let $\sigma = \{v_1/x_1 \dots v_n/x_n\}$ be a substitution closing t_0 and t_1. Let C be a closed context. We want to prove that $C[t_0\sigma] \to_{\mathrm{v}}^* v$ for some v iff $C[t_1\sigma] \to_{\mathrm{v}}^* v'$ for some v', and similarly for control stuck terms. But the context $C' \stackrel{\mathsf{def}}{=} C[(\lambda x_1 \dots x_n.\Box)\, v_1 \, \dots v_n]$ is a context capturing the variables of t_0 and t_1, and we have $C'[t_0] \to_{\mathrm{v}}^* C[t_0\sigma]$ and $C'[t_1] \to_{\mathrm{v}}^* C[t_1\sigma]$. Consequently, $C[t_0\sigma] \to_{\mathrm{v}}^* v$ iff $C'[t_0] \to_{\mathrm{v}}^* v$ iff $C'[t_0] \to_{\mathrm{v}}^* v'$ (first item of the proposition) iff $C[t_1\sigma] \to_{\mathrm{v}}^* v'$ for some v and v'. The reasoning is the same for control stuck terms. □

The next example is used as a counter-example to show that normal-form bisimilarity is not complete (Proposition 6.8): the two terms below are not normal-form bisimilar, but they can be proved applicative bisimilar quite easily.

Proposition 4.19. *We have $\langle x\, i\rangle \mathscr{C}^\circ (\lambda y.\langle x\, i\rangle)\, \langle x\, i\rangle$.*

Proof. We prove that $\mathcal{R} \stackrel{\mathsf{def}}{=} \{(\langle t\rangle, (\lambda y.\langle t\rangle)\, \langle t\rangle) \mid t \in \mathcal{T}_c, y \notin \mathsf{fv}(t)\} \cup \{(t,t) \mid t \in \mathcal{T}_c\}$ is a big-step bisimulation. The term $\langle t\rangle$ can either diverge or reduce to a value (according to Proposition 2.9). If it diverges, then both $\langle t\rangle$ and $(\lambda y.\langle t\rangle)\, \langle t\rangle$ diverge, otherwise, they both evaluate to the same value v. For all v', we, therefore, have $\langle t\rangle \stackrel{v'}{\Rightarrow} t'$ iff $(\lambda y.\langle t\rangle)\, \langle t\rangle \stackrel{v'}{\Rightarrow} t'$, and $t' \mathcal{R} t'$ holds, as wished. □

4.3. **Proving the Axioms.**

We show how to prove Kameyama and Hasegawa's axioms (Section 2.4) except for $\mathcal{S}_{\mathtt{elim}}$ using applicative bisimulation. In the following propositions, we assume the terms to be closed, since the proofs for open terms can be deduced directly from the results for closed terms. First, note that the $\beta_{\mathtt{v}}$, $\langle\cdot\rangle_{\mathcal{S}}$, and $\langle\cdot\rangle_{\mathtt{val}}$ axioms are direct consequences of Proposition 4.5.

Proposition 4.20 ($\eta_{\mathtt{v}}$ axiom)**.** *If $x \notin \mathsf{fv}(v)$, then $\lambda x.v\, x \mathscr{A} v$.*

Proof. We prove that $\mathcal{R} \stackrel{\mathsf{def}}{=} \{(\lambda x.(\lambda y.t)\, x, \lambda y.t) \mid t \in \mathcal{T}, \mathsf{fv}(t) \subseteq \{y\}\} \cup \mathscr{A}$ is a bisimulation. To this end, we have to check that $\lambda x.(\lambda y.t)\, x \xrightarrow{v_0} (\lambda y.t)\, v_0$ is matched by $\lambda y.t \xrightarrow{v_0} t\{v_0/y\}$, i.e., that $(\lambda y.t)\, v_0 \mathcal{R} t\{v_0/y\}$ holds for all v_0. We have $(\lambda y.t)\, v_0 \xrightarrow{\tau} t\{v_0/y\}$, and because $\xrightarrow{\tau} \subseteq \mathscr{A} \subseteq \mathcal{R}$, we have the required result. □

Proposition 4.21 ($\mathcal{S}_{\langle\cdot\rangle}$ axiom)**.** *We have $\mathcal{S}k.\langle t\rangle \mathscr{A} \mathcal{S}k.t$.*

Proof. We have $\mathcal{S}k.\langle t\rangle \xrightarrow{E} \langle\langle t\{\lambda x.\langle E[x]\rangle/k\}\rangle\rangle$ and $\mathcal{S}k.t \xrightarrow{E} \langle t\{\lambda x.\langle E[x]\rangle/k\}\rangle$ for all E. We obtain terms of the form $\langle\langle t'\rangle\rangle$ and $\langle t'\rangle$, and we have proved in Example 4.8 that $\langle\langle t'\rangle\rangle \mathscr{A} \langle t'\rangle$ holds for all t'. □

Proposition 4.22 ($\langle\cdot\rangle_{\mathtt{lift}}$ axiom)**.** *We have $\langle(\lambda x.t_0)\, \langle t_1\rangle\rangle \mathscr{A} (\lambda x.\langle t_0\rangle)\, \langle t_1\rangle$.*

Proof. A transition $\langle(\lambda x.t_0)\, \langle t_1\rangle\rangle \stackrel{\alpha}{\Rightarrow} t'$ (with $\alpha \neq \tau$) is possible only if $\langle t_1\rangle$ evaluates to some value v (evaluation to a control stuck terms is not possible according to Proposition 2.9). In this case, we have $\langle(\lambda x.t_0)\, \langle t_1\rangle\rangle \stackrel{\tau}{\Rightarrow} \langle(\lambda x.t_0)\, v\rangle \xrightarrow{\tau} \langle t_0\{v/x\}\rangle$ and $(\lambda x.\langle t_0\rangle)\, \langle t_1\rangle \stackrel{\tau}{\Rightarrow} \langle t_0\{v/x\}\rangle$. Therefore, we have $\langle(\lambda x.t_0)\, \langle t_1\rangle\rangle \stackrel{\alpha}{\Rightarrow} t'$ (with $\alpha \neq \tau$) iff $(\lambda x.\langle t_0\rangle)\, \langle t_1\rangle \stackrel{\alpha}{\Rightarrow} t'$. From there, it is easy to conclude. □

Proposition 4.23 (β_Ω axiom)**.** *If* $x \notin \mathsf{fv}(E)$*, then* $(\lambda x.E[x])\, t \mathrel{\mathscr{A}} E[t]$*.*

Sketch. We first give some intuitions on why the proof of this result is hard with applicative bisimulation. The difficult case is when t in the initial terms $(\lambda x.E[x])\, t$ and $E[t]$ is a control stuck term $E_0[\mathcal{S}k.t']$. Then we have the following transitions:

$$(\lambda x.E[x])\, t \xrightarrow{E_1} \langle t'\{\lambda y.\langle E_1[(\lambda x.E[x])\; E_0[y]]\rangle/k\}\rangle$$
$$E[t] \xrightarrow{E_1} \langle t'\{\lambda y.\langle E_1[E[E_0[y]]]\rangle/k\}\rangle$$

We obtain terms of the form $\langle t'\rangle\sigma$ and $\langle t'\rangle\sigma'$ (where σ and σ' are the above substitutions). We now have to consider the transitions from these terms, and the interesting case is when $\langle t'\rangle = F[k\, v]$.

$$\langle t'\rangle\sigma \xrightarrow{\tau} F\sigma[\langle E_1[(\lambda x.E[x])\; E_0[v\sigma]]\rangle] \stackrel{\mathsf{def}}{=} t_0$$
$$\langle t'\rangle\sigma' \xrightarrow{\tau} F\sigma'[\langle E_1[E[E_0[v\sigma']]]\rangle] \stackrel{\mathsf{def}}{=} t_1$$

We obtain terms that are similar to the initial terms $(\lambda x.E[x])\, t$ and $E[t]$, except for the extra contexts F and E_1, and the substitutions σ and σ'. Again, the interesting cases are when $E_0[v]$ is either a control stuck term, or a term of the form $F'[k\, v']$. Looking at these cases, we see that the bisimulation we have to define has to relate terms similar to t_0 and t_1, except with an arbitrary number of contexts F' and substitutions similar to σ and σ'.

Formally, given a sequence of (continuation) variables $k_1, \dots, k_n$ and a sequence $\overrightarrow{E_k}$ of triples of contexts E_i, E_i', E_i'' such that

$$\mathsf{fv}(E_i) \cup \mathsf{fv}(E_i') \cup \mathsf{fv}(E_i'') \subseteq \{k_1, \dots, k_{i-1}\} \qquad \text{for } 1 \leq i \leq n \tag{$\star$}$$

we define two families of sequences of substitutions as follows:

$$\begin{array}{rcl} \sigma_i^{\overrightarrow{E}} & = & \{\lambda x.\langle E_i''[(\lambda y.E_i[y])\; E_i'[x]]\rangle/k_i\} \\ \delta_i^{\overrightarrow{E}} & = & \{\lambda x.\langle E_i''[E_i[E_i'[x]]]\rangle/k_i\} \end{array}$$

Additionally, given a term t, a sequence of pure contexts $\overrightarrow{E} = E_1, \dots, E_m$ and a sequence of evaluation contexts $\overrightarrow{F} = F_1, \dots, F_m$, we inductively define two sequences of terms, $s_0, \dots, s_m$ and $u_0, \dots, u_m$, as follows:

$$\begin{array}{rclcrcl} s_0^{t,\overrightarrow{E},\overrightarrow{F}} & = & t & \qquad & u_0^{t,\overrightarrow{E},\overrightarrow{F}} & = & t \\ s_{i+1}^{t,\overrightarrow{E},\overrightarrow{F}} & = & F_{i+1}[(\lambda x.E_{i+1}[x])\; s_i] & & u_{i+1}^{t,\overrightarrow{E},\overrightarrow{F}} & = & F_{i+1}[E_{i+1}[u_i]] \end{array}$$

Then the following relation $\mathcal{R}$ is a bisimulation:

$$\begin{array}{rcl} \mathcal{R} & = & \{(s_i^{t,\overrightarrow{E},\overrightarrow{F}}\sigma_n^{\overrightarrow{E_k}} \dots \sigma_1^{\overrightarrow{E_k}}, u_i^{t,\overrightarrow{E},\overrightarrow{F}}\delta_n^{\overrightarrow{E_k}} \dots \delta_1^{\overrightarrow{E_k}}) \mid \\ & & \qquad \overrightarrow{k} = k_1, \dots, k_n, n \geq 0, \\ & & \qquad \overrightarrow{E_k} \text{ satisfies } (\star), \\ & & \qquad \overrightarrow{E} = E_1, \dots, E_m, \overrightarrow{F} = F_1, \dots, F_m, m \geq 0, \\ & & \qquad \mathsf{fv}(t) \cup \mathsf{fv}(\overrightarrow{E}) \cup \mathsf{fv}(\overrightarrow{F}) \subseteq \{k_1, \dots, k_n\}, \\ & & \qquad 0 \leq i \leq m\} \end{array}$$

We omit the complete bisimulation proof, as we provide much simpler proofs of this result with environmental or normal-form bisimilarities (see Propositions 5.21 and 6.29). $\square$

4.4. **Conclusion.** We define an applicative bisimilarity for the relaxed semantics of $\lambda_{\mathcal{S}}$ which extends the λ-calculus definition with a transition for control-stuck terms. Soundness can be proved by adapting Howe's method to this extra transition, and we can also show completeness w.r.t. $\mathscr{P}$ as well as a context lemma. However, we do not know how to extend these results to the original semantics of $\lambda_{\mathcal{S}}$. While we can think of an applicative bisimilarity for the original semantics by adapting the environmental bisimilarity we define in Section 5.4, we do not know how to prove it sound with Howe's technique. Roughly, Howe's technique fails because it requires the semantics to be preserved by all evaluation contexts, while the original semantics is preserved only by contexts with an outermost reset.

Another issue is that equivalence proofs with applicative bisimulation can be difficult, as witnessed by Example 4.9 or Proposition 4.23. We believe it is due to the lack of powerful up-to techniques, in particular the absence of bisimulation up to context, which reveals to be problematic in a calculus where context capture and manipulation is part of the semantics. As a result, applicative bisimulation seems suitable only for simple examples, such as Proposition 4.19.

5. Environmental Bisimilarity

Like applicative bisimilarity, environmental bisimilarity reduces closed terms to normal forms, which are then compared using some particular contexts (e.g., λ-abstractions are tested by passing them arguments). However, the testing contexts are not arbitrary, but built from an environment, which represents the knowledge acquired so far by an outside observer. The idea originally comes from languages with strict isolation or data abstraction [92, 93, 47, 48], where environments are used to handle information hiding. The term "environmental bisimulation" has then been introduced in [80, 81], and such a bisimilarity has been since defined in various higher-order languages (see, e.g., [83, 91, 69]), including the λ-calculus with first-class abortive continuations [97]. Environmental bisimilarity usually characterizes contextual equivalence, but is harder to use than applicative bisimilarity to prove that two given terms are equivalent. Nonetheless, one can define powerful up-to techniques [81] to simplify the equivalence proofs and deal with this extra difficulty. Besides, the authors of [46] argue that the additional complexity is necessary to handle more realistic features, like local state or exceptions.

Recently, the notion of environmental bisimilarity has been cast in a framework in which soundness proofs for the bisimilarity and its up-to techniques are factorized [60, 61]. We extended that framework to allow for more powerful up-to techniques that are better suited for delimited-control operators [3]. We informally explain in Section 5.1 why we need such an extension in $\lambda_{\mathcal{S}}$, before presenting the extended framework in Section 5.2 and the definition of the bisimilarity itself, first for the relaxed semantics in Section 5.3 and then the original one in Section 5.4. We improve the bisimilarities with up-to techniques (Section 5.5) that we apply to examples (Section 5.6), and in particular to the Kameyama and Hasegawa axiomatization (Section 5.7).

An older work [15] gives definitions of environmental bisimulations that are now completely obsolete. We revisit results originally published in a previous article [3], where the focus is more on a multi-prompted calculus. More precisely, Section 5.1 is rewritten for $\lambda_{\mathcal{S}}$ from [3, Section 4.1] Section 5.2 covers [3, Section 4.3], and Sections 5.3, 5.4, and 5.5 provide more details that [3, Section 5.2]. The examples of Sections 5.6 and 5.7 are a contribution of the present article.

5.1. **Informal Presentation.** In the original formulation of environmental bisimulation [81], two terms t_0 and t_1 are compared under some environment $\mathcal{E}$, which represents the knowledge of an external observer about t_0 and t_1. The definition of the bisimulation enforces some conditions on t_0 and t_1 as well as on $\mathcal{E}$. In Madiot et al.'s framework [60, 61], the conditions on t_0, t_1, and $\mathcal{E}$ are expressed using a LTS between *states* of the form $(\Gamma; t_0)$ and $(\Delta; t_1)$ as well as between states of the form Γ and Δ, where Γ and Δ are finite sequences of values corresponding to the first and second projection of the environment $\mathcal{E}$, respectively. Transitions from states of the form $(\Gamma; t_0)$ express conditions on t_0, while transitions from states of the form Γ explain how we compare environments. Henceforth, if m ranges over a sequence of entities, we write m_i for the $i^{\text{ th}}$ element of the sequence.

For the relaxed semantics of $\lambda_{\mathcal{S}}$, one could think of extending the LTS for the λ-calculus [61] (the first three rules below) with an extra transition for testing stuck terms.

$$\frac{t_0 \rightarrow_{\mathrm{v}} t_1}{(\Gamma; t_0) \xrightarrow{\tau} (\Gamma; t_1)} \qquad \frac{}{(\Gamma; v) \xrightarrow{\mathsf{v}} (\Gamma, v)} \qquad \frac{\Gamma_i = \lambda x.t}{\Gamma \xrightarrow{\lambda,i,\mathbb{V}} (\Gamma; t\{\mathbb{V}[\Gamma]/x\})}$$

$$\frac{t_0 \text{ is control-stuck} \qquad \langle \mathbb{E}[t_0; \Gamma] \rangle \rightarrow_{\mathrm{v}} t_1}{(\Gamma; t_0) \xrightarrow{\mathbb{E}} (\Gamma; t_1)}$$

We use *multi-hole contexts* $\mathbb{V}$ and $\mathbb{E}$ to build respectively values and pure evaluation contexts from an environment Γ; such contexts contain numbered holes $\Box_i$ to be filled with Γ_i. For example, $(\lambda x.(\Box_1\, \Box_3)\,(x\, \Box_3))[\Gamma] = \lambda x.(\Gamma_1\, \Gamma_3)\,(x\, \Gamma_3)$, assuming Γ is at least of size 3. Internal steps $\xrightarrow{\tau}$ correspond to reduction steps. The transition $\xrightarrow{\mathsf{v}}$ turns a state $(\Gamma; v)$ into a sequence of values; when we are done evaluating a term, we can add the newly acquired knowledge to the environment. Environments are tested with the transition $\xrightarrow{\lambda,i,\mathbb{V}}$, which means that the i^{th} element of Γ is tested by applying it to an argument built using $\mathbb{V}$. Finally, $\xrightarrow{\mathbb{E}}$ tests control-stuck terms by putting them in a context built from $\mathbb{E}$ to trigger the capture, where the notation $\mathbb{E}[t_0; \Gamma]$ means that the hole in the evaluation position in $\mathbb{E}$ is plugged with t_0 while the numbered holes are plugged with Γ_i.

The transitions $\xrightarrow{\tau}$, $\xrightarrow{\mathbb{E}}$, and $\xrightarrow{\lambda,i,\mathbb{V}}$ correspond to the transitions $\xrightarrow{\tau}$, $\xrightarrow{E}$, and $\xrightarrow{v}$ defining applicative bisimulation, except the testing arguments are built from the environment. As a result, plain environmental bisimulation proofs are harder than applicative ones, as witnessed by the following example.

Example 5.1 (Turing's combinator)**.** Following Example 4.9, we want to prove that Turing's combinator Θ_{v} is bisimilar to its variant $\Theta_{\mathcal{S}} \stackrel{\mathsf{def}}{=} \langle \theta\, \mathcal{S}k.k\, k \rangle$. We remind that

$$\Theta_{\mathsf{v}} \Downarrow_{\mathrm{v}} \lambda y.y\,(\lambda z.\theta\,\theta\,y\,z) \stackrel{\mathsf{def}}{=} v_0, \text{ and}$$

$$\Theta_{\mathcal{S}} \Downarrow_{\mathrm{v}} \lambda y.y\,(\lambda z.(\lambda x.\langle \theta\, x \rangle)\,(\lambda x.\langle \theta\, x \rangle)\,y\,z) \stackrel{\mathsf{def}}{=} v_1.$$

Let $\Gamma \stackrel{\mathsf{def}}{=} (v_0)$ and $\Delta \stackrel{\mathsf{def}}{=} (v_1)$; then

$$\Gamma \xrightarrow{\lambda,1,\mathbb{V}} (\Gamma; \mathbb{V}[\Gamma]\,(\lambda z.\theta\,\theta\,\mathbb{V}[\Gamma]\,z)) \text{ and}$$

$$\Delta \xrightarrow{\lambda,1,\mathbb{V}} (\Delta; \mathbb{V}[\Delta]\,(\lambda z.(\lambda x.\langle \theta\, x \rangle)\,(\lambda x.\langle \theta\, x \rangle)\,\mathbb{V}[\Delta]\,z)).$$

Because we have different terms $\mathbb{V}[\Gamma]$ and $\mathbb{V}[\Delta]$ and not a single value v, the case analysis suggested in Example 4.9 becomes much more complex, as we have to take into account how $\mathbb{V}$ uses Γ or Δ.

Up-to techniques are what makes environmental bisimulation tractable, in particular bisimulation up to context, which allows to factor out a common context: when comparing states of the form $(\Gamma; \mathbb{C}[\Gamma])$ and $(\Delta; \mathbb{C}[\Delta])$, where $\mathbb{C}$ is a multi-hole context, we can forget about $\mathbb{C}$ and focus on Γ and Δ. Similarly for $(\Gamma; \mathbb{F}[t_0; \Gamma])$ and $(\Delta; \mathbb{F}[t_1; \Delta])$, where $\mathbb{F}$ is a multi-hole evaluation context, we can consider only $(\Gamma; t_0)$ and $(\Delta; t_1)$; the restriction to evaluation contexts is necessary for the technique to be sound, as pointed out by Madiot [61, page 111]. Bisimulation up to context is unfortunately not powerful enough to be useful in $\lambda_{\mathcal{S}}$. Suppose we want to prove a variant of the β_Ω axiom, $\langle(\lambda x.E[x])\, t\rangle$ equivalent to $\langle E[t]\rangle$ if $x \notin \mathsf{fv}(E)$. If $t = E'[(\mathcal{S}k.k\, k)\, v]$ for some E' and v, then

$$(\emptyset; \langle(\lambda x.E[x])\, E'[(\mathcal{S}k.k\, k)\, v]\rangle) \xRightarrow{\tau} (\emptyset; \langle\langle(\lambda x.E[x])\, E'[\langle(\lambda x.E[x])\, E'[v\, v]\rangle]\rangle\rangle) \text{ and}$$
$$(\emptyset; \langle E[E'[(\mathcal{S}k.k\, k)\, v]]\rangle) \xRightarrow{\tau} (\emptyset; \langle\langle E[E'[\langle E[E'[v\, v]]\rangle]]\rangle\rangle).$$

The two resulting terms do not share a common evaluation context beyond $\langle\langle\Box\rangle\rangle$, so bisimulation up to context cannot simplify the proof from there.

Yet we can see that the two resulting terms have the same shape, except for the contexts $(\lambda x.E[x])\,\Box$ and E. Following this observation, in a previous work [3], we proposed a more expressive notion of bisimulation up to context where the common context $\mathbb{F}$ can be built out of *related evaluation contexts*. We do so by adding to the syntax of multi-hole contexts the constructs $\star_i[\mathbb{C}]$ and $\star_i[\mathbb{F}]$, where the hole $\star_i$ can be filled by an evaluation context F to produce respectively $F[\mathbb{C}]$ and $F[\mathbb{F}]$. We also include sequences of evaluation contexts Ψ or Φ in the LTS states $(\Psi; \Gamma; t)$ and $(\Psi; \Gamma)$. As a result, if $\Psi \stackrel{\mathsf{def}}{=} ((\lambda x.E[x])\,\Box)$, $\Phi \stackrel{\mathsf{def}}{=} (E)$, and $\mathbb{C} \stackrel{\mathsf{def}}{=} \langle\langle\star_1[E'[\star_1[E'[v\, v]]]]\rangle\rangle$, then

$$\mathbb{C}[\Psi; \emptyset] = \langle\langle(\lambda x.E[x])\, E'[\langle(\lambda x.E[x])\, E'[v\, v]\rangle]\rangle\rangle \text{ and}$$
$$\mathbb{C}[\Phi; \emptyset] = \langle\langle E[E'[\langle E[E'[v\, v]]\rangle]]\rangle\rangle$$

so $\mathbb{C}$ can be factored out using our notion of bisimulation up to related contexts.

Extending the state to include evaluation contexts means that these contexts have to be tested, by plugging them with an argument built from the environment.

$$\frac{}{(\Psi; \Gamma) \xrightarrow{\Box, j, \mathbb{V}} (\Psi; \Gamma; \Psi_j[\mathbb{V}[\Psi; \Gamma]])}$$

However, such a transition is problematic in conjunction with our notion of bisimulation up to related contexts. Indeed, for all F_0 and F_1, we have

$$(F_0; \emptyset) \xrightarrow{\Box, 1, \mathbb{V}} (F; \emptyset; F_0[\mathbb{V}[F_0; \emptyset]]) \text{ and}$$
$$(F_1; \emptyset) \xrightarrow{\Box, 1, \mathbb{V}} (F'; \emptyset; F_1[\mathbb{V}[F_1; \emptyset]]).$$

But the two resulting states are bisimilar up to related contexts, since for all F, $F[\mathbb{V}[F; \emptyset]] = (\star_1[\mathbb{V}])[F; \emptyset]$. If bisimulation up to related contexts is a valid up-to technique, it implies that $(F_0; \emptyset)$ and $(F_1; \emptyset)$ are bisimilar for any F_0 and F_1, which is obviously false (consider $F_0 = \Box$ and $F_1 = \Box\, \Omega$). To prevent this, we distinguish *passive* transitions (such as $\xrightarrow{\Box, i, \mathbb{V}}$) from the other ones (called *active*), so that only selected up-to techniques (referred to as *strong*) can be used after a passive transition. In contrast, any up-to technique (including bisimulation up to related contexts) can be used after an active transition. To formalize this idea, we extend Madiot et al.'s framework to allow such distinctions between transitions and between up-to techniques. We present the definitions in a general setting in Section 5.2, before illustrating them with environmental bisimilarity for $\lambda_{\mathcal{S}}$.

5.2. **Diacritical Progress and Up-to Techniques.** We recall the main definitions and results of the extended framework from our previous work [3]; see this paper for more details.

Diacritical progress. Let $\xrightarrow{\alpha}$ be a LTS defined on states ranged over by Σ or Θ, which contains an internal action labeled τ. Weak transitions $\xRightarrow{\alpha}$ are defined as $\xRightarrow{\tau} \stackrel{\mathsf{def}}{=} \xrightarrow{\tau}^*$ and $\xRightarrow{\alpha} \stackrel{\mathsf{def}}{=} \xRightarrow{\tau}\xrightarrow{\alpha}\xRightarrow{\tau}$ if $\alpha \neq \tau$. A (weak) bisimulation over this LTS can be defined using a notion of *progress*: a relation $\mathcal{R}$ progresses towards $\mathcal{S}$, written $\mathcal{R} \rightarrowtail \mathcal{S}$, if $\Sigma \mathrel{\mathcal{R}} \Theta$ implies that if $\Sigma \xrightarrow{\alpha} \Sigma'$, there exists Θ' such that $\Theta \xRightarrow{\alpha} \Theta'$ and $\Sigma' \mathrel{\mathcal{S}} \Theta'$, and conversely if $\Theta \xrightarrow{\alpha} \Theta'$. A bisimulation is then defined as a relation $\mathcal{R}$ verifying $\mathcal{R} \rightarrowtail \mathcal{R}$, and bisimilarity is the largest bisimulation.

In our extended framework, we suppose that the transitions of the LTS are partitioned into *passive* and *active* transitions, and we define *diacritical progress* as follows.

Definition 5.2. A relation $\mathcal{R}$ diacritically progresses to $\mathcal{S}, \mathcal{T}$ written $\mathcal{R} \rightarrowtail\!\!\!\rightarrow \mathcal{S}, \mathcal{T}$, if $\mathcal{R} \subseteq \mathcal{S}$, $\mathcal{S} \subseteq \mathcal{T}$, and $\Sigma \mathrel{\mathcal{R}} \Theta$ implies that

- if $\Sigma \xrightarrow{\alpha} \Sigma'$ and $\xrightarrow{\alpha}$ is passive, then there exists Θ' such that $\Theta \xRightarrow{\alpha} \Theta'$ and $\Sigma' \mathrel{\mathcal{S}} \Theta'$;
- if $\Sigma \xrightarrow{\alpha} \Sigma'$ and $\xrightarrow{\alpha}$ is active, then there exists Θ' such that $\Theta \xRightarrow{\alpha} \Theta'$ and $\Sigma' \mathrel{\mathcal{T}} \Theta'$;
- the converse of the above conditions on Θ.

A bisimulation is a relation $\mathcal{R}$ such that $\mathcal{R} \rightarrowtail\!\!\!\rightarrow \mathcal{R}, \mathcal{R}$, and bisimilarity $\mathscr{B}$ is the largest bisimulation. Since a bisimulation $\mathcal{R}$ progresses towards $\mathcal{R}$ after both passive and active transitions, the two notions of progress $\rightarrowtail$ and $\rightarrowtail\!\!\!\rightarrow$ in fact generate the same notions of bisimulation and bisimilarity; the distinction between active and passive transitions is interesting only when considering up-to techniques.

Up-to techniques. The goal of up-to techniques is to simplify bisimulation proofs: instead of proving that a relation $\mathcal{R}$ is a bisimulation, we show that $\mathcal{R}$ respects some looser constraints which still imply bisimilarity $\mathscr{B}$. In our setting, we distinguish the up-to techniques which can be used after a passive transition (called *strong* up-to techniques), from the ones which cannot. An up-to technique (resp. strong up-to technique) is a function f such that $\mathcal{R} \rightarrowtail\!\!\!\rightarrow \mathcal{R}, f(\mathcal{R})$ (resp. $\mathcal{R} \rightarrowtail\!\!\!\rightarrow f(\mathcal{R}), f(\mathcal{R})$) implies $\mathcal{R} \subseteq \mathscr{B}$. Proving that a given f is an up-to technique is difficult with this definition, so following Madiot, Pous, and Sangiorgi [75, 60], we rely on a notion of *respectfulness*, which gives sufficient conditions for f to be an up-to technique, and is easier to establish, as functions built out of respectful functions using composition and union remain respectful.

We first need some auxiliary notions on notations on functions on relations, ranged over by f, g, and h in what follows. We define $f \subseteq g$ and $f \cup g$ argument-wise, e.g., $(f \cup g)(\mathcal{R}) = f(\mathcal{R}) \cup g(\mathcal{R})$ for all $\mathcal{R}$. We define f^ω as $\bigcup_{n \in \mathbb{N}} f^n$. We write id for the identity function on relations, and $\widehat{f}$ for $f \cup \mathsf{id}$. Given a set $\mathfrak{F}$ of functions, we also write $\mathfrak{F}$ for the function defined as $\bigcup_{f \in \mathfrak{F}} f$. We say a function f is *generated from* $\mathfrak{F}$ if f can be built from functions in $\mathfrak{F}$ and id using union, composition, and $\cdot^\omega$. The largest function generated from $\mathfrak{F}$ is $\widehat{\mathfrak{F}}^\omega$. A function f is monotone if $\mathcal{R} \subseteq \mathcal{S}$ implies $f(\mathcal{R}) \subseteq f(\mathcal{S})$. We write $\mathcal{P}_{fin}(\mathcal{R})$

for the set of finite subsets of $\mathcal{R}$, and we say f is continuous if it can be defined by its image on these finite subsets, i.e., if $f(\mathcal{R}) \subseteq \bigcup_{\mathcal{S}\in\mathcal{P}_{fin}(\mathcal{R})} f(\mathcal{S})$. The up-to techniques of the present paper are defined by inference rules with a finite number of premises, so they are trivially continuous.

Definition 5.3. A function f evolves to g, h, written $f \rightsquigarrow g, h$, if for all $\mathcal{R} \rightarrowtail\!\!\!\!\!\twoheadrightarrow \mathcal{R}, \mathcal{T}$, we have $f(\mathcal{R}) \rightarrowtail\!\!\!\!\!\twoheadrightarrow g(\mathcal{R}), h(\mathcal{T})$. A function f *strongly* evolves to g, h, written $f \rightsquigarrow_{\mathsf{s}} g, h$, if for all $\mathcal{R} \rightarrowtail\!\!\!\!\!\twoheadrightarrow \mathcal{S}, \mathcal{T}$, we have $f(\mathcal{R}) \rightarrowtail\!\!\!\!\!\twoheadrightarrow g(\mathcal{S}), h(\mathcal{T})$.

Evolution can be seen as a notion of progress for functions on relations. Note that strong evolution does not put any condition on how $\mathcal{R}$ progresses, while regular evolution is more restricted, as it requires a relation $\mathcal{R}$ such that $\mathcal{R} \rightarrowtail\!\!\!\!\!\twoheadrightarrow \mathcal{R}, \mathcal{T}$.

Definition 5.4. A set $\mathfrak{F}$ of continuous functions is *diacritically respectful* if there exists $\mathfrak{S}$ such that $\mathfrak{S} \subseteq \mathfrak{F}$ and

- for all $f \in \mathfrak{S}$, we have $f \rightsquigarrow_{\mathsf{s}} \widehat{\mathfrak{S}}^{\omega}, \widehat{\mathfrak{F}}^{\omega}$;
- for all $f \in \mathfrak{F}$, we have $f \rightsquigarrow \widehat{\mathfrak{S}}^{\omega} \circ \widehat{\mathfrak{F}} \circ \widehat{\mathfrak{S}}^{\omega}, \widehat{\mathfrak{F}}^{\omega}$.

In words, a function is in a respectful set $\mathfrak{F}$ if it evolves towards a combination of functions in $\mathfrak{F}$. The (possibly empty) subset $\mathfrak{S}$ intuitively represents the strong up-to techniques of $\mathfrak{F}$. Any combination of functions can be used after an active transition. After a passive one, only strong functions can be used, except in the second case, where we progress from $f(\mathcal{R})$, with f not strong. In that case, it is expected to progress towards a combination that includes f; it is safe to do so, as long as f (or in fact, any non-strong function in $\mathfrak{F}$) is used at most once. If $\mathfrak{S}_1$ and $\mathfrak{S}_2$ are subsets of $\mathfrak{F}$ which verify the conditions of the definition, then $\mathfrak{S}_1 \cup \mathfrak{S}_2$ also does, so there exists the largest subset of $\mathfrak{F}$ which satisfies the conditions, written $\mathsf{strong}(\mathfrak{F})$.

Proposition 5.5. *Let $\mathfrak{F}$ be a diacritically compatible set.*

- *If $\mathcal{R} \rightarrowtail\!\!\!\!\!\twoheadrightarrow \widehat{\mathsf{strong}(\mathfrak{F})}^{\omega}(\mathcal{R}), \widehat{\mathfrak{F}}^{\omega}(\mathcal{R})$, then $\widehat{\mathfrak{F}}^{\omega}(\mathcal{R})$ is a bisimulation.*
- *any function generated from $\mathfrak{F}$ is an up-to technique, and any function generated from $\mathsf{strong}(\mathfrak{F})$ is a strong up-to technique.*
- *For all $f \in \mathfrak{F}$, we have $f(\approx) \subseteq \approx$.*

The second point implies that combining functions from a respectful set using union, composition, or $\cdot^{\omega}$ produces up-to techniques. In particular, if $f \in \mathfrak{F}$, then f is an up-to technique, and similarly, if $f \in \mathsf{strong}(\mathfrak{F})$, then f is a strong up-to technique. In practice, proving that f is in a respectful set $\mathfrak{F}$ is easier than proving it is an up-to technique. The last item states that bisimilarity is preserved by respectful functions, so proving that up to context is respectful implies that bisimilarity is preserved by contexts.

The first item suggests a more flexible notion of up-to technique, as it shows that given a respectful set $\mathfrak{F}$, a relation may progress towards different functions f and g, $\mathcal{R} \rightarrowtail\!\!\!\!\!\twoheadrightarrow f(\mathcal{R}), g(\mathcal{R})$, and still be included in the bisimilarity as long as f is generated from $\mathsf{strong}(\mathfrak{F})$ and g is generated from $\mathfrak{F}$. In what follows, we rely on that property in examples and say that in that case, $\mathcal{R}$ is a bisimulation up to $\mathfrak{F}$, or $\mathcal{R}$ is a bisimulation up to $f_1 \dots f_n$ if $\mathfrak{F} = \{f_1 \dots f_n\}$.

Remark 5.6 (respectful vs compatible functions)**.** The literature distinguishes between respectful [79] and *compatible* [73] functions: f is respectful if $\mathcal{R} \rightarrowtail \mathcal{S}$ and $\mathcal{R} \subseteq \mathcal{S}$ implies $f(\mathcal{R}) \rightarrowtail f(\mathcal{S})$, while f is compatible if $\mathcal{R} \rightarrowtail \mathcal{S}$ implies $f(\mathcal{R}) \rightarrowtail f(\mathcal{S})$. Some interesting up-to techniques are not compatible but are respectful thanks to the extra inclusion hypothesis.

Mimicking [60, 61], we use the term “compatible” instead of “respectful” in our previous work [3] for the definition with the extra inclusion hypothesis. We use “respectful” in this paper to be faithful to the original definitions, and because we use “compatible” for relations preserved by the operators of the language. Pous [74] argues that the difference between the two notions is of little importance anyway as they generate the same companion function.

Remark 5.7. As a matter of fact, the theory we present in this section does not require an underlying notion of LTS. In particular, Definition 5.3 and 5.4 as well as the proof of Proposition 5.5 do not depend on the notion of diacritical progress being defined in terms of a LTS. As long as the notion of progress satisfies the following (simple) conditions:

- if $R \rightarrowtail\!\!\!\twoheadrightarrow S, T$, $S \subseteq S'$, and $T \subseteq T'$, then $R \rightarrowtail\!\!\!\twoheadrightarrow S', T'$;
- if $\forall i \in I. R_i \rightarrowtail\!\!\!\twoheadrightarrow S, T$, then $\bigcup_{i \in I} R_i \rightarrowtail\!\!\!\twoheadrightarrow S, T$,

the presented theory is valid for such a notion of progress. We exploit this fact in Section 6.1, when defining a normal-form bisimilarity.

5.3. **Bisimilarity for the Relaxed Semantics.** We define environmental bisimulation using a LTS between states of the form $(\Psi; \Gamma; t)$ (called *term states*) or $(\Psi; \Gamma)$ (called *environment states*), where we denote by Γ or Δ a sequence of closed values, and by Ψ or Φ a sequence of closed evaluation contexts, and where t is a closed term. As explained in Section 5.1, the values are used to build testing arguments to compare λ-abstractions, while we store evaluation contexts to define bisimulation up to related contexts. We build the testing entities out of Ψ and Γ using *multi-hole contexts*, defined as follows.

Contexts:	$\mathbb{C} ::= \mathbb{V} \mid \mathbb{C}\,\mathbb{C} \mid \langle \mathbb{C} \rangle \mid \mathcal{S}k.\mathbb{C} \mid \star_j [\mathbb{C}]$
Value contexts:	$\mathbb{V} ::= x \mid \lambda x.\mathbb{C} \mid \square_i$
Evaluation contexts:	$\mathbb{F} ::= \square \mid \mathbb{F}\,\mathbb{C} \mid \mathbb{V}\,\mathbb{F} \mid \langle \mathbb{F} \rangle \mid \star_j [\mathbb{F}]$

We distinguish value holes $\square_i$ from context holes $\star_j$. These holes are indexed, unlike the special hole $\square$ of an evaluation context $\mathbb{F}$, which is in evaluation position (that is, filling the other holes of $\mathbb{F}$ gives a regular evaluation context F). Filling the holes of $\mathbb{C}$ and $\mathbb{V}$ with Ψ and Γ, written respectively $\mathbb{C}[\Psi; \Gamma]$ and $\mathbb{V}[\Psi; \Gamma]$, consists in replacing any subterm of the form $\star_j[\mathbb{C}']$ with $\Psi_j[\mathbb{C}']$ and any occurrence of $\square_i$ with Γ_i, assuming that j is smaller or equal than the size of Ψ and similarly for i w.r.t. Γ. We write $\mathbb{F}[t; \Psi; \Gamma]$ for the same operation with evaluation contexts, where we assume that t is put in $\square$. We extend the notion of free variables to multi-hole contexts as expected, and a multi-hole context is said closed if it has no free variables.

Figure 4 presents the LTS $\xrightarrow{\alpha}$ for the relaxed semantics of $\lambda_{\mathcal{S}}$, where the relation $t \xrightarrow{=} t'$ is defined as follows: if $t \rightarrow_{\mathrm{v}} t'$, then $t \xrightarrow{=} t'$, and if t is a normal form, then $t \xrightarrow{=} t$.[4] The multi-hole contexts $\mathbb{V}$ and $\mathbb{F}$ used in the transition $\xrightarrow{\mathbb{F}}$, $\xrightarrow{\lambda, i, \mathbb{V}}$, and $\xrightarrow{\square, j, \mathbb{V}}$ are supposed to be closed. The internal transition $\xrightarrow{\tau}$ corresponds to reduction. The transition $\xrightarrow{\lambda, i, \mathbb{V}}$ tests the λ-abstraction Γ_i by passing it an argument built with $\mathbb{V}$. The transition can be fired for any i smaller than the size of Γ, ensuring that all the values in Γ are tested. The transition $\xrightarrow{\mathsf{v}}$ turns a term state $(\Psi; \Gamma; v)$ into an environment state $(\Psi; \Gamma, v)$ since v cannot reduce further.

[4]The relation $\xrightarrow{=}$ is not exactly the reflexive closure of $\rightarrow_{\mathrm{v}}$, since an expression which is not a normal form *must* reduce.

$$\frac{t \rightarrow_{\mathrm{v}} t'}{(\Psi;\Gamma;t) \xrightarrow{\tau} (\Psi;\Gamma;t')} \qquad \frac{\Gamma_i = \lambda x.t}{(\Psi;\Gamma) \xrightarrow{\lambda,i,\mathbb{V}} (\Psi;\Gamma;t\{\mathbb{V}[\Psi;\Gamma]/x\})} \qquad \frac{}{(\Psi;\Gamma;v) \xrightarrow{\mathsf{v}} (\Psi;\Gamma,v)}$$

$$\frac{t \text{ is stuck} \qquad \mathbb{F}[t;\Psi;\Gamma] \xrightarrow{=} t'}{(\Psi;\Gamma;t) \xrightarrow{\mathbb{F}} (\Psi;\Gamma;t')} \qquad \frac{}{(\Psi;\Gamma) \xrightarrow{\square,j,\mathbb{V}} (\Psi;\Gamma;\Psi_j[\mathbb{V}[\Psi;\Gamma]])}$$

$$\frac{\Psi_j = E}{(\Psi;\Gamma) \xrightarrow{\square,j} (\Psi;\Gamma)} \qquad \frac{\Psi_j = F[\langle E\rangle]}{(\Psi;\Gamma) \xrightarrow{\langle\square\rangle,j} (\Psi, F[\langle\square\rangle], \langle E\rangle;\Gamma)}$$

Figure 4: LTS for the relaxed semantics

The transition $\xrightarrow{\mathbb{F}}$ compares stuck terms by putting them in an evaluation context $\mathbb{F}$ to trigger the capture, like the corresponding transition in applicative bisimulation. However, characterizing the capture-triggering contexts is more difficult than in Section 4, because of holes $\star_i$. Indeed, a context $\star_i[\mathbb{E}]$ may also provoke a capture if Ψ_i is an impure context: for example, we have $(\langle\square\rangle;\emptyset;\mathcal{S}k.t) \xrightarrow{\star_1[\square]} (\langle\square\rangle;\emptyset;\langle t\{\lambda x.\langle x\rangle/k\}\rangle)$. Instead of looking for a precise characterization, we simply test with all context $\mathbb{F}$, and then trigger the capture using $\xrightarrow{=}$ only when possible, i.e., when $\mathbb{F}[\Psi;\Gamma]$ contains a surrounding reset. A uninteresting transition $(\Psi;\Gamma;t) \xrightarrow{\mathbb{F}} (\Psi;\Gamma;\mathbb{F}[t;\Psi;\Gamma])$ where no capture happens will then be easily dealt with up-to techniques (see Example 5.14).

The remaining transitions $\xrightarrow{\square,j,\mathbb{V}}$, $\xrightarrow{\square,j}$, and $\xrightarrow{\langle\square\rangle,j}$ deal with the evaluation contexts in Ψ, and are therefore applied only in conjunction with bisimulation up to related contexts, as Ψ is not empty only in that case. The transition $\xrightarrow{\square,j,\mathbb{V}}$ tests the evaluation context Ψ_j by passing it a value built from $\mathbb{V}$, the same way λ-abstractions are tested with $\xrightarrow{\lambda,i,\mathbb{V}}$. Testing an evaluation context with a value is simpler than with any term, however it does not account for all the possible interactions of a term with an evaluation context. Indeed, a stuck term is able to distinguish a pure context from an impure one, and it can extract from $F[\langle E\rangle]$ the context up to the first enclosing reset $\langle E\rangle$. We use $\xrightarrow{\square,j}$ and $\xrightarrow{\langle\square\rangle,j}$ to mimic these behaviors. The transition $\xrightarrow{\square,j}$ simply states that Ψ_j is pure; in a bisimulation $\mathcal{R}$ such that $(\Psi;\Gamma)\ \mathcal{R}\ (\Phi;\Delta)$ and $(\Psi;\Gamma) \xrightarrow{\square,j} (\Psi;\Gamma)$, $(\Phi;\Delta)$ has to match with the same transition, meaning that Φ_j must also be pure. Similarly, $\xrightarrow{\langle\square\rangle,j}$ decomposes $\Psi_j = F[\langle E\rangle]$ into $F[\langle\square\rangle]$ and $\langle E\rangle$. Because the transition leaves a reset inside F, applying the same transition again to $F[\langle\square\rangle]$ does not decompose F further, but simply generates $F[\langle\square\rangle]$ again (and $\langle\square\rangle$). The duplicated contexts can then be ignored using up-to techniques.

To define environmental bisimulation using diacritical progress (Section 5.2), we distinguish the transitions $\xrightarrow{\square,j,\mathbb{V}}$ and $\xrightarrow{\mathsf{v}}$ as passive, while the remaining others are active. We consider a transition as passive if it can be inverted by an up-to technique, which is possible if no new information is generated between its source and target states. For example, $(\Psi;\Gamma;v) \xrightarrow{\mathsf{v}} (\Psi;\Gamma,v)$ is passive because we simply change the nature of the state (from term to environment). In contrast, the transition $(\Psi;\Gamma) \xrightarrow{\square,j} (\Psi;\Gamma)$ is active, as we gain some

$$\frac{p \to_{\mathsf{v}} p'}{(\Psi;\Gamma;p) \xrightarrow{\tau} (\Psi;\Gamma;p')} \qquad \frac{\Gamma_i = \lambda x.t \qquad \mathbb{F}[\Psi;\Gamma] \text{ is delimited}}{(\Psi;\Gamma) \xrightarrow{\lambda,i,\mathbb{V},\mathbb{F}} (\Psi;\Gamma;\mathbb{F}[t\{\mathbb{V}[\Psi;\Gamma]/x\};\Psi;\Gamma])}$$

$$\frac{}{(\Psi;\Gamma;v) \xrightarrow{\mathsf{v}} (\Psi;\Gamma,v)} \qquad \frac{\mathbb{F}[\Psi;\Gamma] \text{ is delimited}}{(\Psi;\Gamma) \xrightarrow{\Box,j,\mathbb{V},\mathbb{F}} (\Psi;\Gamma;\mathbb{F}[\Psi_j[\mathbb{V}[\Psi;\Gamma]];\Psi;\Gamma])}$$

$$\frac{\Psi_j = E}{(\Psi;\Gamma) \xrightarrow{\Box,j} (\Psi;\Gamma)} \qquad \frac{\Psi_j = F[\langle E\rangle]}{(\Psi;\Gamma) \xrightarrow{\langle\Box\rangle,j} (\Psi,F[\langle\Box\rangle],\langle E\rangle;\Gamma)}$$

Figure 5: LTS for the original semantics

information: Γ_j is a pure context. The transition $(\Psi;\Gamma) \xrightarrow{\Box,j,\mathbb{V}} (\Psi;\Gamma;t)$ is passive at it simply recombines existing information in Γ and Ψ to build Γ, without any reduction step taking place, and thus without generating new information. Some extra knowledge is produced only when $(\Psi;\Gamma;t)$ evolves (with active transitions), as it then tells us how the tested context Γ_j actually interacts with the value constructed from $\mathbb{V}$. Finally, $\xrightarrow{\lambda,i,\mathbb{C}_{\mathsf{v}}}$ and $\xrightarrow{\mathbb{E}}$ correspond to reduction steps and are therefore active, and $\xrightarrow{\langle\Box\rangle,j}$ is also active as it provides some information by telling us how to decompose a context.

Definition 5.8. A relation $\mathcal{R}$ on states is an environmental bisimulation if $\mathcal{R} \rightarrowtail\!\!\!\!\!\twoheadrightarrow \mathcal{R},\mathcal{R}$. Environmental bisimilarity $\mathscr{E}$ is the largest environmental bisimulation.

We extend $\mathscr{E}$ to open terms as follows: if $\overrightarrow{x} = \mathsf{fv}(t_0) \cup \mathsf{fv}(t_1)$, then we write $t_0 \; \mathscr{E}^\circ \; t_1$ if $(\emptyset;\lambda\overrightarrow{x}.t_0) \; \mathscr{E} \; (\emptyset;\lambda\overrightarrow{x}.t_1)$. We discuss the soudness and completeness of $\mathscr{E}$ in Section 5.5, after giving the definition of the bisimilarity for the original semantics.

5.4. **Definitions for the Original Semantics.** In the original semantics, terms are evaluated within a top-level reset. To follow that principle, the LTS for the original semantics is defined only on *pure terms*, i.e., terms without effects, defined as follows.

Pure terms: $\quad p ::= v \mid \langle t\rangle$

We remind that terms of the form $\langle t\rangle$ are called delimited; we extend this notion to contexts as well. A *pure state* is of the form $(\Psi;\Gamma;p)$, and the LTS operates either on pure or environment states. The problem is then how to build pure states out of terms that are not pure. A simple idea would be to relate two impure terms t_0 and t_1 by comparing $\langle t_0\rangle$ and $\langle t_1\rangle$. However, such a solution would not be sound, as it would relate $\mathcal{S}k.k\,y$ and $\mathcal{S}k.(\lambda z.z)\,y$, terms that can be distinguished by the context $\langle\Box\,\Omega\rangle$.

Instead, the transitions $\xrightarrow{\lambda,i,\mathbb{V},\mathbb{F}}$ and $\xrightarrow{\Box,j,\mathbb{V},\mathbb{F}}$ of the LTS for the original semantics (Figure 5) now include an extra argument $\mathbb{F}$ to build pure terms in their resulting state. Recall that we use any evaluation context and not a delimited context $\langle\mathbb{E}\rangle$ as it is possible to build a context of that shape from a context $\star_i[\mathbb{F}]$ assuming Ψ_i contains an enclosing reset. Besides, we can discard the non-interesting parts of a testing context $\mathbb{F}$ thanks to bisimulation up to context. The other main difference between the LTS for the original and relaxed semantics is the lack of rule for testing control-stuck terms, as pure terms cannot become stuck (see Proposition 2.9).

Techniques for both semantics

$$\frac{t_0 \to_{\mathrm{v}}^* t_0' \qquad t_1 \to_{\mathrm{v}}^* t_1' \qquad (\Psi; \Gamma; t_0') \;\mathcal{R}\; (\Phi; \Delta; t_1')}{(\Psi; \Gamma; t_0) \;\mathsf{red}(\mathcal{R})\; (\Phi; \Delta; t_1)} \qquad \frac{(\Psi, \overrightarrow{F}; \Gamma, \overrightarrow{v}; t_0) \;\mathcal{R}\; (\Phi, \overrightarrow{F'}; \Delta, \overrightarrow{w}; t_1)}{(\Psi; \Gamma; t_0) \;\mathsf{weak}(\mathcal{R})\; (\Phi; \Delta; t_1)}$$

Techniques specific to the relaxed semantics

$$\frac{(\Psi; \Gamma) \;\mathcal{R}\; (\Phi; \Delta)}{(\Psi, \overrightarrow{\mathbb{F}}[\Psi; \Gamma]; \Gamma, \overrightarrow{\mathbb{V}}[\Psi; \Gamma]; \mathbb{C}[\Psi; \Gamma]) \;\mathsf{rctx}(\mathcal{R})\; (\Phi, \overrightarrow{\mathbb{F}}[\Phi; \Delta]; \Delta, \overrightarrow{\mathbb{V}}[\Phi; \Delta]; \mathbb{C}[\Phi; \Delta])}$$

$$\frac{(\Psi; \Gamma; t_0) \;\mathcal{R}\; (\Phi; \Delta; t_1)}{(\Psi, \overrightarrow{\mathbb{F}}[\Psi; \Gamma]; \Gamma, \overrightarrow{\mathbb{V}}[\Psi; \Gamma]; \mathbb{F}[t_0; \Psi; \Gamma]) \;\mathsf{rectx}(\mathcal{R})\; (\Phi, \overrightarrow{\mathbb{F}}[\Phi; \Delta]; \Delta, \overrightarrow{\mathbb{V}}[\Phi; \Delta]; \mathbb{F}[t_1; \Phi; \Delta])}$$

Techniques specific to the original semantics

$$\frac{(\Psi; \Gamma) \;\mathcal{R}\; (\Phi; \Delta) \qquad \mathbb{C}[\Psi; \Gamma] \text{ and } \mathbb{C}[\Phi; \Delta] \text{ are delimited}}{(\Psi, \overrightarrow{\mathbb{F}}[\Psi; \Gamma]; \Gamma, \overrightarrow{\mathbb{V}}[\Psi; \Gamma]; \mathbb{C}[\Psi; \Gamma]) \;\mathsf{rctx}(\mathcal{R})\; (\Phi, \overrightarrow{\mathbb{F}}[\Phi; \Delta]; \Delta, \overrightarrow{\mathbb{V}}[\Phi; \Delta]; \mathbb{C}[\Phi; \Delta])}$$

$$\frac{(\Psi; \Gamma; p_0) \;\mathcal{R}\; (\Phi; \Delta; p_1) \qquad \mathbb{F}[\Psi; \Gamma] \text{ and } \mathbb{F}[\Phi; \Delta] \text{ are delimited}}{(\Psi, \overrightarrow{\mathbb{F}}[\Psi; \Gamma]; \Gamma, \overrightarrow{\mathbb{V}}[\Psi; \Gamma]; \mathbb{F}[p_0; \Psi; \Gamma]) \;\mathsf{rectx}(\mathcal{R})\; (\Phi, \overrightarrow{\mathbb{F}}[\Phi; \Delta]; \Delta, \overrightarrow{\mathbb{V}}[\Phi; \Delta]; \mathbb{F}[p_1; \Phi; \Delta])}$$

Figure 6: Up-to techniques in both semantics

As in the relaxed semantics, the transitions $\xrightarrow{\mathsf{v}}$ and $\xrightarrow{\square, j, \mathbb{V}, \mathbb{F}}$ are passive and the others are active; we write $\rightarrowtail\!\!\!\twoheadrightarrow_{\mathsf{p}}$ for the notion of progress based on the LTS of Figure 5.

Definition 5.9. A relation $\mathcal{R}$ on states is a pure environmental bisimulation if $\mathcal{R} \rightarrowtail\!\!\!\twoheadrightarrow_{\mathsf{p}} \mathcal{R}, \mathcal{R}$. Pure environmental bisimilarity $\mathscr{F}$ is the largest pure environmental bisimulation.

We define $\mathscr{F}^\circ$ on open terms as in the relaxed case, and we extend $\mathscr{F}$ to any terms as follows: we have $t_0 \;\mathscr{F}\; t_1$ if for all E, we have $(\emptyset; \emptyset; \langle E[t_0] \rangle) \;\mathscr{F}\; (\emptyset; \emptyset; \langle E[t_1] \rangle)$. As a simple example illustrating the differences between $\mathscr{E}$ and $\mathscr{F}$, we have the following result.

Proposition 5.10. *We have* $\Omega \;\mathscr{F}\; \mathcal{S}k.\Omega$.

The relation $\{((\emptyset; \emptyset; \langle E[\Omega] \rangle), (\emptyset; \emptyset; \langle E[\mathcal{S}k.\Omega] \rangle)), ((\emptyset; \emptyset; \langle E[\Omega] \rangle), (\emptyset; \emptyset; \langle \Omega \rangle))\}$ is a pure bisimulation. Proposition 5.10 does not hold with $\mathscr{E}$ because Ω is not stuck.

5.5. **Up-to Techniques.** Environmental bisimulation is meant to be used with up-to techniques, as doing bisimulation proofs with Definition 5.8 or 5.9 alone is tedious in practice. Figure 6 lists the up-to techniques we use for environmental bisimilarity in the two semantics. Bisimulation up to reduction red relates terms after some reduction steps, thus allowing a big-step reasoning even with a small step bisimulation. Bisimulation up to weakening weak, also called bisimulation up to environment in previous works [81], removes values and contexts from a state, thus diminishing its testing power, since less values and contexts means less arguments to build from with multi-hole contexts.

We define two kinds of bisimulations up to related contexts, depending whether we operate on environment states (rctx) or on term or pure states (rectx). As explained before, only evaluation contexts are allowed for term and pure states, while any context is valid with environment states. These up-to techniques differ from the usual bisimulation up to context in the syntax of the multi-hole contexts, which may include context holes $\star_i$. The definitions for the original semantics differ from the ones for the relaxed semantics in that only pure terms can be built in the case of the original semantics.

The definitions of rctx and rectx also allow the sequences of values and contexts to be extended. This operation opposite to weakening, known as *strengthening* [60], does not change the testing power of the states, since the added values and contexts are built out of the existing ones. We inline strengthening in the definitions of bisimulation up to related contexts for technical reason: a separate notion of bisimulation up to strengthening would be a regular up-to technique (not strong), like bisimulation up to related contexts, which entails that these up-to techniques could not be composed after a passive transition in a respectfulness proof.

The functions we define are indeed up-to techniques, are they form a respectful set in both semantics [3].

Lemma 5.11. $\mathfrak{F} \stackrel{\mathsf{def}}{=} \{\mathsf{red}, \mathsf{weak}, \mathsf{rctx}, \mathsf{rectx}\}$ *is diacritically respectful, with* $\mathsf{strong}(\mathfrak{F}) = \{\mathsf{red}, \mathsf{weak}\}$*.*

This lemma and the third property of Proposition 5.5 directly imply that $\mathscr{E}$ and $\mathscr{F}$ are compatible, from which we can deduce that they are sound w.r.t. respectively $\mathscr{C}$ and $\mathscr{P}$. We can also prove that they are complete [3].

Theorem 5.12. $t_0 \mathrel{\mathscr{C}} t_1$ *iff* $(\emptyset; \emptyset; t_0) \mathrel{\mathscr{E}} (\emptyset; \emptyset; t_1)$*, and* $t_0 \mathrel{\mathscr{P}} t_1$ *iff* $(\emptyset; \emptyset; t_0) \mathrel{\mathscr{F}} (\emptyset; \emptyset; t_1)$*.*

Sketch. Given two terms t_0, t_1, we write $t_0 \sim t_1$ if

- $t_0 \Downarrow_{\mathrm{v}} v_0$ for some v_0 iff $t_1 \Downarrow_{\mathrm{v}} v_1$ for some v_1, and
- $t_0 \Downarrow_{\mathrm{v}} t_0'$ for some control-stuck term t_0' iff $t_0 \Downarrow_{\mathrm{v}} t_1'$ for some control-stuck term t_1'.

For the relaxed semantics, we show that

$$\begin{aligned}\mathcal{R} \stackrel{\mathsf{def}}{=} &\{((\Psi; \Gamma; t_0), (\Phi; \Delta; t_1)) \mid \forall \mathbb{F}, \mathbb{F}[t_0; \Psi; \Gamma] \sim \mathbb{F}[t_1; \Phi; \Delta]\} \\ &\cup \{((\Psi; \Gamma), (\Phi; \Delta)) \mid \forall \mathbb{C}, \mathbb{C}[\Psi; \Gamma] \sim \mathbb{C}[\Phi; \Delta]\}\end{aligned}$$

is an environmental bisimulation. We define a similar candidate relation or the original semantics with the extra requirement that the contexts $\mathbb{F}$ and $\mathbb{C}$ should be delimited. The proof is then by case analysis on the possible transitions. For example, take $(\Psi; \Gamma) \mathrel{\mathcal{R}} (\Phi; \Delta)$ such that $(\Psi; \Gamma) \xrightarrow{\lambda, i, \mathbb{V}} (\Psi; \Gamma; t_0)$. Then $(\Phi; \Delta) \xrightarrow{\lambda, i, \mathbb{V}} (\Phi; \Delta; t_1)$ for some t_1. Let $\mathbb{F}$ such that $\mathbb{F}[t_0; \Psi; \Gamma]$ evaluates to a value or a control-stuck term. Consider $\mathbb{C} = \mathbb{F}[\Box_i\ \mathbb{V}]$; then $\mathbb{C}[\Psi; \Gamma] \rightarrow_{\mathrm{v}} \mathbb{F}[t_0; \Psi; \Gamma]$ and $\mathbb{C}[\Phi; \Delta] \rightarrow_{\mathrm{v}} \mathbb{F}[t_1; \Phi; \Delta]$. Because $\mathbb{C}[\Psi; \Gamma] \sim \mathbb{C}[\Phi; \Delta]$, we deduce $\mathbb{F}[t_0; \Psi; \Gamma] \sim \mathbb{F}[t_1; \Phi; \Gamma]$. □

5.6. **Examples.** We illustrate the usefulness of bisimulation up to related contexts, first on one of our running basic examples.

Example 5.13 (double reset)**.** The relation $\mathcal{R} \stackrel{\mathsf{def}}{=} \{(\emptyset; \emptyset; \langle\langle t \rangle\rangle), (\emptyset; \emptyset; \langle t \rangle) \mid t \in \mathcal{T}_c\}$ is a bisimulation up to context. Indeed, we show in Example 4.8 that either $\langle t \rangle \rightarrow_{\mathrm{v}} \langle t' \rangle$ for some t' or $\langle t \rangle \rightarrow_{\mathrm{v}} v$, and $\langle t \rangle \rightarrow_{\mathrm{v}} \langle t' \rangle$ iff $\langle\langle t \rangle\rangle \rightarrow_{\mathrm{v}} \langle\langle t' \rangle\rangle$ and $\langle t \rangle \rightarrow_{\mathrm{v}} v$ $\langle\langle t \rangle\rangle \rightarrow_{\mathrm{v}} \langle v \rangle \rightarrow_{\mathrm{v}} v$. After a $\xrightarrow{\tau}$ step, we either stay in $\mathcal{R}$, or we get identical terms, i.e., we are in $\mathsf{rctx}(\mathcal{R})$.

We then show how it helps when checking the $\xrightarrow{\mathbb{F}}$ transition.

Example 5.14. Let $\Sigma \stackrel{\mathsf{def}}{=} (\Psi;\Gamma;t_0)$ and $\Theta \stackrel{\mathsf{def}}{=} (\Phi;\Psi;t_1)$ so that t_0 and t_1 are stuck, and $\Sigma \mathrel{\mathcal{R}} \Theta$. If $\Sigma \xrightarrow{\mathbb{F}} (\Psi;\Gamma;\mathbb{F}[t_0;\Psi;\Gamma])$ and $\Theta \xrightarrow{\mathbb{F}} (\Phi;\Delta;\mathbb{F}[t_1;\Phi;\Delta])$, i.e., $\mathbb{F}$ does not trigger the capture in t_0 and t_1, then we can conclude directly since we have $(\Psi;\Gamma;\mathbb{F}[t_0;\Psi;\Gamma]) \mathrel{\mathsf{rectx}(\mathcal{R})} (\Psi;\Gamma;\mathbb{F}[t_1;\Phi;\Delta])$. Similarly, if $\mathbb{F} = \mathbb{F}'[\langle\mathbb{E}\rangle]$, then $\Sigma \xrightarrow{\mathbb{F}} (\Psi;\Gamma;\mathbb{F}'[\langle t_0'\rangle;\Psi;\Gamma])$ and $\Theta \xrightarrow{\mathbb{F}} (\Phi;\Delta;\mathbb{F}'[\langle t_1'\rangle;\Phi;\Delta])$ for some t_0' and t_1', so rectx allows us to forget about $\mathbb{F}'$ and to focus on $(\Psi;\Gamma;t_0')$ and $(\Phi;\Psi;t_1')$.

The next example is specific to the original semantics and illustrates the role of $\star_i$. It does not hold in the relaxed semantics, because the term on the right is control-stuck, but the one on the left may not evaluate to a control-stuck term if t_1 does not terminate.

Example 5.15. If $k \notin \mathsf{fv}(t_1)$, then $(\emptyset;\emptyset;(\lambda x.\mathcal{S}k.t_0)\ t_1) \mathrel{\mathscr{F}} (\emptyset;\emptyset;\mathcal{S}k.((\lambda x.t_0)\ t_1))$, as the relation

$$\begin{aligned}\mathcal{R} \stackrel{\mathsf{def}}{=} &\{((\emptyset;\emptyset;\langle E[(\lambda x.\mathcal{S}k.t_0)\ t_1]\rangle),(\emptyset;\emptyset;\langle E[\mathcal{S}k.((\lambda x.t_0)\ t_1)]\rangle))\}\\ &\cup\{((\langle E[(\lambda x.\mathcal{S}k.t_0)\ \Box]\rangle;\emptyset),(\langle(\lambda x.t_0\{\lambda y.\langle E[y]\rangle/k\})\ \Box\rangle;\emptyset))\}\end{aligned}$$

is a bisimulation up to reduction and related contexts. We start by analyzing the behavior of the first pair (Σ,Θ) in $\mathcal{R}$. If t_1 is a value v_1, then

$$\begin{aligned}\Sigma &\xrightarrow{\tau}\xrightarrow{\tau}(\emptyset;\emptyset;\langle t_0\{v_1/x\}\{\lambda z.\langle E[z]\rangle/k\}\rangle) \text{ and}\\ \Theta &\xrightarrow{\tau}\xrightarrow{\tau}(\emptyset;\emptyset;\langle t_0\{\lambda z.\langle E[z]\rangle/k\}\{v_1/x\}\rangle),\end{aligned}$$

but because $k \notin \mathsf{fv}(v_1)$, the resulting states are in fact equal, and therefore in $\mathsf{rctx}(\mathcal{R})$. If $t_1 \to_{\mathrm{v}} t_1'$, then

$$\begin{aligned}\Sigma &\xrightarrow{\tau}(\emptyset;\emptyset;\langle E[(\lambda x.\mathcal{S}k.t_0)\ t_1']\rangle) \text{ and}\\ \Theta &\xrightarrow{\tau}\xrightarrow{\tau}(\emptyset;\emptyset;\langle(\lambda x.t_0\{\lambda y.\langle E[y]\rangle/k\})\rangle\ t_1');\end{aligned}$$

the resulting states are in $\mathsf{rctx}(\mathcal{R})$, by considering the common context $\star_1[t_1']$. If t_1 is a control-stuck term $E'[\mathcal{S}k'.t_1']$, then

$$\begin{aligned}\Sigma &\xrightarrow{\tau}\langle t_1'\{\lambda z.\langle E[(\lambda x.\mathcal{S}k.t_0)\ E'[z]]\rangle/k'\}\rangle \text{ and}\\ \Theta &\xrightarrow{\tau}\xrightarrow{\tau}\langle t_1'\{\lambda z.\langle(\lambda x.t_0\{\lambda y.\langle E[y]\rangle/k\})\ E'[z]\rangle/k'\}\rangle.\end{aligned}$$

Again, the resulting states are in $\mathsf{rctx}(\mathcal{R})$, by considering the context $\langle t_1'\{\lambda z.\langle\star_1[E'[z]]\rangle/k'\}\rangle$. We have covered all the possible cases for (Σ,Θ).

For the second set, if

$$(\Psi;\emptyset) \stackrel{\mathsf{def}}{=} (\langle E[(\lambda x.\mathcal{S}k.t_0)\ \Box]\rangle;\emptyset) \qquad \text{and} \qquad (\Phi;\emptyset) \stackrel{\mathsf{def}}{=} (\langle\lambda x.t_0\{\lambda y.\langle E[y]\rangle/k\}\ \Box\rangle;\emptyset),$$

then

$$\begin{aligned}(\Psi;\emptyset) &\xrightarrow{\Box,1,\mathbb{V},\mathbb{F}}\xrightarrow{\tau}\xrightarrow{\tau}(\Psi;\emptyset;\langle\mathbb{F}[\langle t_0\{\mathbb{V}[\Psi;\emptyset]/x\}\{\lambda y.\langle E[y]\rangle/k\}\rangle;\Psi;\emptyset]\rangle) \text{ and}\\ (\Phi;\emptyset) &\xrightarrow{\Box,1,\mathbb{V},\mathbb{F}}\xrightarrow{\tau}(\Phi;\emptyset;\langle\mathbb{F}[\langle t_0\{\mathbb{V}[\Phi;\emptyset]/x\}\{\lambda y.\langle E[y]\rangle/k\}\rangle;\Phi;\emptyset]\rangle)\end{aligned}$$

The resulting states are in $\mathsf{rctx}(\mathcal{R})$; note that we use the regular up-to technique rctx only after an active $\xrightarrow{\tau}$ transition, and not after the passive $\xrightarrow{\Box,1,\mathbb{V},\mathbb{F}}$ transition.

The next example shows the limits of bisimulation up to related contexts for environmental bisimilarity, as we have to define an infinite candidate relation for a simple example. However, it is still an improvement over the plain environmental (Example 5.1) or applicative (Example 4.9) proofs.

Example 5.16 (Turing's combinator)**.** Let $\theta' \stackrel{\mathsf{def}}{=} \lambda x.\langle \theta\, x\rangle$, $v_0 \stackrel{\mathsf{def}}{=} \lambda y.y\,(\lambda z.\theta\,\theta\,y\,z)$, $v_1 \stackrel{\mathsf{def}}{=} \lambda y.y\,(\lambda z.\theta'\,\theta'\,y\,z)$. We define $\mathcal{R}_0$ inductively as follows.

$$\frac{}{(\emptyset; v_0)\ \mathcal{R}_0\ (\emptyset; v_1)} \qquad \frac{(\emptyset;\Gamma)\ \mathcal{R}_0\ (\emptyset;\Delta)}{(\emptyset;\Gamma,\lambda z.\Theta_{\mathsf{v}}\ \mathbb{V}[\emptyset;\Gamma]\ z)\ \mathcal{R}_0\ (\emptyset; v_1, \lambda z.\theta'\,\theta'\ \mathbb{V}[\emptyset;\Delta]\ z)}$$

Then $\mathcal{R} \stackrel{\mathsf{def}}{=} \{((\emptyset;\emptyset;\Theta_{\mathsf{v}}),(\emptyset;\emptyset;\Theta_{\mathcal{S}})),((\emptyset;\Gamma;\Theta_{\mathsf{v}}),(\emptyset;\Delta;\theta'\,\theta'))\ \mid\ (\emptyset;\Gamma)\ \mathcal{R}_0\ (\emptyset;\Delta)\}\cup\mathcal{R}_0$ is a bisimulation up to related contexts. For the first two pairs, we have respectively $(\emptyset;\emptyset;\Theta_{\mathsf{v}}) \stackrel{\mathsf{v}}{\Rightarrow} (\emptyset; v_0)$ and $(\emptyset;\emptyset;\Theta_{\mathcal{S}}) \stackrel{\mathsf{v}}{\Rightarrow} (\emptyset; v_1)$, and $(\emptyset;\Gamma;\Theta_{\mathsf{v}}) \stackrel{\mathsf{v}}{\Rightarrow} (\emptyset;\Gamma, v_0)$ and $(\emptyset;\Delta;\theta'\,\theta') \stackrel{\mathsf{v}}{\Rightarrow} (\emptyset;\Delta, v_1)$ for some $(\emptyset;\Gamma)\ \mathcal{R}_0\ (\emptyset;\Delta)$, so we end up in $\mathcal{R}_0$ in both cases.

Let $(\emptyset;\Gamma)\ \mathcal{R}_0\ (\emptyset;\Delta)$, and suppose we want to test v_0 and v_1, i.e.,

$$(\emptyset;\Gamma) \xrightarrow{\lambda,1,\mathbb{V}} (\emptyset;\Gamma;\mathbb{V}[\emptyset;\Gamma]\ (\lambda z.\Theta_{\mathsf{v}}\ \mathbb{V}[\emptyset;\Gamma]\ z)) \text{ and}$$
$$(\emptyset;\Delta) \xrightarrow{\lambda,1,\mathbb{V}} (\emptyset;\Delta;\mathbb{V}[\emptyset;\Delta]\ (\lambda z.\theta'\,\theta'\ \mathbb{V}[\emptyset;\Delta]\ z))$$

At that point, we would like to relate Θ_{v} and $\theta'\theta'$ and conclude using bisimulation up context, however these terms are not in an evaluation contexts in the above resulting states. Similarly, we cannot isolate $\Theta_{\mathsf{v}}\,\square$ and $(\theta'\,\theta')\,\square$ using $\star$ and bisimulation up to related contexts, as these contexts are not evaluation contexts. Instead, $\mathcal{R}_0$ has been defined so that the resulting states are in $\mathsf{rctx}(\mathcal{R}_0)$.

Finally, suppose $i > 1$ and $\Gamma_i = \lambda z.\Theta_{\mathsf{v}}\ \mathbb{V}[\emptyset;\Gamma']\ z$, $\Delta_i = \lambda z.\theta'\,\theta'\ \mathbb{V}[\emptyset;\Delta']\ z$ for some $\mathbb{V}$ and $(\emptyset;\Gamma')\ \mathcal{R}_0\ (\emptyset;\Delta')$. Then $(\emptyset;\Gamma) \xrightarrow{\lambda,i,\mathbb{V}'} (\emptyset;\Gamma;\Theta_{\mathsf{v}}\ \mathbb{V}[\emptyset;\Gamma']\ \mathbb{V}'[\emptyset;\Gamma])$ and $(\emptyset;\Delta) \xrightarrow{\lambda,i,\mathbb{V}'} (\emptyset;\Delta;\theta'\,\theta'\ \mathbb{V}[\emptyset;\Delta']\ \mathbb{V}'[\emptyset;\Delta])$, and the resulting states are in $\mathsf{rectx}(\mathcal{R})$.

5.7. **Proving the Axioms.** We prove Kameyama and Hasegawa's axioms (Section 2.4) with environmental bisimulation. The β_{v}, $\langle\cdot\rangle_{\mathcal{S}}$, and $\langle\cdot\rangle_{\mathtt{val}}$ axioms are direct consequences of a more general result.

Proposition 5.17. *If $t \rightarrow_v t'$, then $(\emptyset;\emptyset;t)\ \mathscr{E}\ (\emptyset;\emptyset;t')$.*

Proof. It is easy to see that $\{((\emptyset;\emptyset;t),(\emptyset;\emptyset;t'))\}$ is a bisimulation up to context: after a transition, we get identical terms. □

Proposition 5.18 (η_{v} axiom)**.** *If $x \notin \mathsf{fv}(v)$, then $(\emptyset;\emptyset;\lambda x.v\,x)\ \mathscr{E}\ (\emptyset;\emptyset;v)$.*

Proof. The relation $\mathcal{R} \stackrel{\mathsf{def}}{=} \{((\emptyset;\lambda x.v\,x),(\emptyset;v))\}$ is a bisimulation up to context up to reduction. If $v = \lambda y.t$, then the $\xrightarrow{\lambda,1,\mathbb{V}}$ transition produces $(\emptyset;\lambda x.vx; v\mathbb{V}[\emptyset;\lambda x.vx])$ and $(\emptyset;v;t\{\mathbb{V}[\emptyset;v]/y\})$. Then $(\emptyset;\lambda x.v\,x;v\,\mathbb{V}[\emptyset;\lambda x.v\,x]) \xrightarrow{\tau} (\emptyset;\lambda x.v\,x;t\{\mathbb{V}[\emptyset;\lambda x.v\,x]/y\})\ \mathsf{rctx}(\mathcal{R})\ (\emptyset;v;t\{\mathbb{V}[\emptyset;v]/y\})$, as wished. □

Proposition 5.19 ($\mathcal{S}_{\langle\cdot\rangle}$ axiom)**.** *We have $(\emptyset;\emptyset;\mathcal{S}k.\langle t\rangle)\ \mathscr{E}\ (\emptyset;\emptyset;\mathcal{S}k.t)$.*

Proof. The relation $\mathcal{R} \stackrel{\mathsf{def}}{=} \{((\emptyset;\emptyset;\mathcal{S}k.\langle t\rangle),(\emptyset;\emptyset;\mathcal{S}k.t)),((\emptyset;\emptyset;\langle\langle t\rangle\rangle),(\emptyset;\emptyset;\langle t\rangle)) \mid t \in \mathcal{T}_c\}$ is a bisimulation up to context. Indeed, if $\mathbb{F} = F[\langle E\rangle]$ (the other case being trivial), then $(\emptyset;\emptyset;\mathcal{S}k.\langle t\rangle) \xrightarrow{\mathbb{F}} (\emptyset;\emptyset;F[\langle\langle t\{\lambda x.\langle E[x]\rangle/k\}\rangle\rangle])$ and $(\emptyset;\emptyset;\mathcal{S}k.t) \xrightarrow{\mathbb{F}} (\emptyset;\emptyset;F[\langle t\{\lambda x.\langle E[x]\rangle/k\}\rangle])$. We obtain states of the form $(\emptyset;\emptyset;F[\langle\langle t'\rangle\rangle])$ and $(\emptyset;\emptyset;F[\langle t'\rangle])$, which are in $\mathsf{rectx}(\mathcal{R})$. Example 5.13 concludes in the case of the pair $((\emptyset;\emptyset;\langle\langle t\rangle\rangle),(\emptyset;\emptyset;\langle t\rangle))$. □

Proposition 5.20 ($\langle\cdot\rangle_{\texttt{lift}}$ axiom)**.** *We have $(\emptyset;\emptyset;\langle(\lambda x.t_0)\ \langle t_1\rangle\rangle) \mathrel{\mathscr{E}} (\emptyset;\emptyset;(\lambda x.\langle t_0\rangle)\ \langle t_1\rangle)$.*

Proof. The relation $\{((\emptyset;\emptyset;\langle(\lambda x.t)\ \langle t'\rangle\rangle),(\emptyset;\emptyset;(\lambda x.\langle t\rangle)\ \langle t'\rangle)) \mid t' \in \mathcal{T}_c\}$ is a bisimulation up to context, with the same reasoning as in Example 5.13. □

Proposition 5.21 (β_Ω axiom)**.** *If $x \notin \mathsf{fv}(E)$, then $(\emptyset;\emptyset;(\lambda x.E[x])\ t) \mathrel{\mathscr{E}} (\emptyset;\emptyset;E[t])$.*

Proof. Define $\Psi \stackrel{\mathsf{def}}{=} ((\lambda x.E[x])\ \Box)$, $\Phi \stackrel{\mathsf{def}}{=} (E)$, and

$$\mathcal{R} \stackrel{\mathsf{def}}{=} \{((\Psi;\emptyset),(\Phi;\emptyset)),((\Psi;\emptyset;(\lambda x.E[x])\ \mathbb{V}[\Psi;\emptyset]),(\Phi;\emptyset;E[\mathbb{V}[\Psi;\emptyset]]))\}.$$

Then $(\emptyset;\emptyset;(\lambda x.E[x])\ t)$ $\mathsf{weak}(\mathsf{rctx}(\mathcal{R}))$ $(\emptyset;\emptyset;E[t])$ and $\mathcal{R}$ is a bisimulation up to context, since the sequence $(\Psi;\emptyset) \xrightarrow{\Box,1,\mathbb{V}} (\Psi;\emptyset;(\lambda x.E[x])\ \mathbb{V}[\Psi;\emptyset]) \xrightarrow{\tau} (\Psi;\emptyset;E[\mathbb{V}[\Psi;\emptyset]])$ fits $(\Phi;\emptyset) \xrightarrow{\Box,1,\mathbb{V}} (\Phi;\emptyset;E[\mathbb{V}[\Phi;\emptyset]]) \xRightarrow{\tau} (\Phi;\emptyset;E[\mathbb{V}[\Phi;\emptyset]])$, where the final states are in $\mathsf{rctx}(\mathcal{R})$. Notice we use rctx after $\xrightarrow{\tau}$, and not after the passive $\xrightarrow{\Box,1,\mathbb{V}}$ transition. The transition $\xrightarrow{\Box,1}$ is easy to check. □

Proposition 5.22. *If $k \notin \mathsf{fv}(t)$, then $(\emptyset;\emptyset;\mathcal{S}k.k\ t) \mathrel{\mathscr{F}} (\emptyset;\emptyset;t)$.*

Proof. Let $\Psi \stackrel{\mathsf{def}}{=} (\langle\lambda x.\langle E[x]\rangle\ \Box\rangle,\langle\langle\Box\rangle\rangle)$, $\Phi \stackrel{\mathsf{def}}{=} (\langle E\rangle,\langle\Box\rangle)$, and

$$\mathcal{R} \stackrel{\mathsf{def}}{=} \{((\emptyset;\emptyset;\langle E[\mathcal{S}k.k\ t]\rangle),(\emptyset;\emptyset;\langle E[t]\rangle)),((\Psi;\emptyset),(\Phi;\emptyset)) \mid x \notin \mathsf{fv}(E)\}.$$

For the first pair, we have $(\emptyset;\emptyset;\langle E[\mathcal{S}k.k\ t]\rangle) \xrightarrow{\tau} (\emptyset;\emptyset;\langle(\lambda x.\langle E[x]\rangle)\rangle\ t)$ and $(\emptyset;\emptyset;\langle E[t]\rangle) \xRightarrow{\tau} (\emptyset;\emptyset;\langle E[t]\rangle)$ so that the resulting states are in $\mathsf{rectx}(\mathcal{R})$, by considering the context $\star_1[t]$.

Otherwise, the sequence $(\Psi;\emptyset) \xrightarrow{\Box,1,\mathbb{V},\mathbb{F}}\xrightarrow{\tau} (\Psi;\emptyset;\langle\mathbb{F}[\langle\langle E[\mathbb{V}[\Psi;\emptyset]]\rangle\rangle;\Psi;\emptyset]\rangle)$ is matched by $(\Psi;\emptyset) \xRightarrow{\Box,1,\mathbb{V},\mathbb{F}} (\Psi;\emptyset;\langle\mathbb{F}[\langle E[\mathbb{V}[\Phi;\emptyset]]\rangle;\Phi;\emptyset]\rangle))$, since the resulting states are in $\mathsf{rctx}(\mathcal{R})$, and we use up to related contexts after a $\xrightarrow{\tau}$ transition. Finally, $(\Psi;\emptyset) \xrightarrow{\Box,2,\mathbb{V},\mathbb{F}}\xrightarrow{\tau}\xrightarrow{\tau} (\Psi;\emptyset;\langle\mathbb{F}[\mathbb{V}[\Psi;\emptyset];\Psi;\emptyset]\rangle)$ is matched by $(\Phi;\emptyset) \xrightarrow{\Box,2,\mathbb{V},\mathbb{F}}\xrightarrow{\tau} (\Phi;\emptyset;\langle\mathbb{F}[\mathbb{V}[\Phi;\emptyset];\Phi;\emptyset]\rangle)$, and the context splitting transitions $\xrightarrow{\langle\Box\rangle,i}$ are easy to check for $i \in \{1,2\}$. □

The bisimilarity $\mathscr{F}$ verifies all the axioms of $\equiv$, it is therefore complete w.r.t. this relation.

Corollary 5.23. *We have $\equiv \subseteq \mathscr{M}$.*

As a result, we can use $\equiv$ as a proof technique for $\mathscr{F}$ (and, therefore, for $\mathscr{P}$). For instance, Example 5.15 holds directly as it can be derived from the axioms [39].

5.8. **Conclusion.** We define environmental bisimilarities that are sound and complete in the relaxed and original semantics. Plain environmental bisimulation is harder to use than applicative bisimulation, but it is supposed to be used in conjunction with up-to techniques. In particular, bisimulation up to related contexts, which allows to forget about a common context built out of values and evaluation contexts in the environment, is what makes the proof technique tractable enough to prove the β_Ω axiom, an axiom which can hardly be proved with applicative bisimilarity (see Example 4.23). However, some equivalence proofs seem to be still unnecessary complex, as witnessed by Example 5.16.

Another issue is that the definition of $\mathscr{F}$ is only a small improvement over the definition of $\mathscr{P}$, as it contains quantifications over evaluation contexts, either when extending the definition from any terms to pure terms, or in the transitions $\xrightarrow{\lambda,i,\mathbb{V},\mathbb{F}}$ and $\xrightarrow{\square,j,\mathbb{V},\mathbb{F}}$. In practice, these contexts are not too problematic as many of them can be abstracted away in equivalence proofs thanks to up-to techniques (see Example 5.15 or Proposition 5.22), but we wonder if it is possible to still have a complete bisimilarity and quantify over less contexts or to restrict the class of terms on which such a quantification over contexts is necessary.

6. Normal-Form Bisimilarity

Normal-form bisimilarity [53] (originally defined in [78], where it was called *open bisimilarity*) equates (open) terms by reducing them to normal form, and then requiring the sub-terms of these normal forms to be bisimilar. Unlike applicative and environmental bisimilarities, normal-form bisimilarity usually does not contain a universal quantification over testing terms or contexts in its definition, and is therefore easier to use than the former two. However, it is also usually not complete w.r.t. contextual equivalence, meaning that 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 [52, 53], the λ-calculus with ambiguous choice [54], the $\lambda\mu$-calculus [55], and the $\lambda\mu\rho$-calculus [88], a calculus with control and store, where normal-form bisimilarity characterizes contextual equivalence. It has also been defined for typed languages [56, 57]. In a recent work [17], we recast normal-form bisimulation in the framework of diacritical progress (Section 5.2), to be able to define up-to techniques which respect η-expansion; we refer to this work for more details.

In Section 6.1, we propose a first definition of normal-form bisimilarity for the relaxed semantics, for which we define up-to techniques in Section 6.2. We then refine the definition in Section 6.3, to relate more contextually equivalent terms. We turn to the original semantics in Section 6.4, and we prove the axioms in Section 6.5. The material of Sections 6.1 and 6.2 comes from [17], where the proofs can be found, and supersedes [14]. Refined bisimilarity, originally defined in [14], is adapted to the framework of [17] in the present article. Normal-form bisimilarity for the original semantics is also a contribution of this article. The proofs for Sections 6.3 and 6.4 can be found in the appendix.

6.1. **Definition.**

The main idea behind the definition of normal-form bisimilarity is that 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. In the λ-calculus [78, 53], the possible normal forms are only values and open stuck terms. In the relaxed semantics of $\lambda_{\mathcal{S}}$, we need to relate also control-stuck terms; we propose here a first way to deal with these terms, that will be refined in a later subsection. Deconstructing normal forms leads to comparing contexts as well as terms. Given a relation $\mathcal{R}$ on terms, we define in Figure 7 the extensions of $\mathcal{R}$ to respectively values $\mathcal{R}^{\mathsf{v}}$, other normal forms $\mathcal{R}^{\mathsf{nf}}$, and contexts $\mathcal{R}^{\mathsf{c}}$.

The relation $\mathcal{R}^{\mathsf{v}}$ treats uniformly the different kinds of values by applying them to a fresh variable x. As originally pointed out by Lassen [53], this is necessary for the bisimilarity

$$\frac{E_0[x] \mathrel{\mathcal{R}} E_1[x] \qquad x \text{ fresh}}{E_0 \mathrel{\mathcal{R}^{\mathsf{c}}} E_1} \qquad \frac{\langle E_0[x]\rangle \mathrel{\mathcal{R}} \langle E_1[x]\rangle \qquad F_0[x] \mathrel{\mathcal{R}} F_1[x] \qquad x \text{ fresh}}{F_0[\langle E_0\rangle] \mathrel{\mathcal{R}^{\mathsf{c}}} F_1[\langle E_1\rangle]}$$

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

Figure 7: Extension of a relation to contexts and normal forms

to be sound w.r.t. η-expansion; otherwise it would distinguish η-equivalent terms such as $\lambda y.x\, y$ and x. However, unlike Lassen, we do not use a special application operator to get rid of administrative β-redexes when possible, as it is not necessary in our framework. The definition of $\mathcal{R}^{\mathsf{v}}$ also easily scales to some other kinds of values: for example, we consider shift as a value in a previous work [17] with the same definition.

A control-stuck term $E_0[\mathcal{S}k.t_0]$ can be executed if it is plugged into a pure 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'. The resulting term is within a reset; similarly, when $\mathcal{R}^{\mathsf{nf}}$ compares $E_0[\mathcal{S}k.t_0]$ and $E_1[\mathcal{S}k.t_1]$, it relates the shift bodies t_0 and t_1 within an enclosing reset. The pure contexts E_0 and E_1 are also tested by simply plugging a fresh variable into them. Comparing t_0' and t_1' without a surrounding reset would be too discriminating, as it would distinguish equivalent terms such as $\mathcal{S}k.\langle t\rangle$ and $\mathcal{S}k.t$ (axiom $\mathcal{S}_{\langle\cdot\rangle}$). 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 our definition requires $\langle\langle t\rangle\rangle$ and $\langle t\rangle$ to be related (which holds for all t; see Example 6.2).

Two open stuck terms $F_0[x\, v_0]$ and $F_1[x\, v_1]$ are related by $\mathcal{R}^{\mathsf{nf}}$ 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 compare 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 $\square$ and $\langle\square\rangle$, which in turn would relate the terms $x\, v$ and $\langle x\, v\rangle$, which are distinguished by the context $(\lambda x.\square)\, \lambda y.\mathcal{S}k.\Omega$. 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 $\langle\square\rangle$ and $\langle\langle\square\rangle\rangle$, and, therefore, it would discriminate the contextually equivalent terms $\langle x\, v\rangle$ and $\langle\langle x\, v\rangle\rangle$. Hence, we check with $\mathcal{R}^{\mathsf{c}}$ (Figure 7) that if one of the contexts contains a reset surrounding the hole, then so does the other; then we compare the contexts beyond the first enclosing delimiter by simply evaluating them using a fresh variable. As a result, it rightfully distinguishes $\square$ and $\langle\square\rangle$, but it relates $\langle\square\rangle$ and $\langle\langle\square\rangle\rangle$.

With these auxiliary relations, we define normal-form bisimilarity using the notion of diacritical progress of Section 5.2. However, here we do not introduce an underlying LTS, but instead we refer directly to the reduction semantics of the calculus, which we find advantageous when working with open terms. One can check that the notion of progress defined below satisfies the conditions mentioned in Remark 5.7, and therefore we can still rely on the theory presented in Section 5.2.

Definition 6.1. A relation $\mathcal{R}$ on open terms diacritically progresses to $\mathcal{S}$, $\mathcal{T}$ written $\mathcal{R} \rightarrowtail \mathcal{S}, \mathcal{T}$, if $\mathcal{R} \subseteq \mathcal{S}$, $\mathcal{S} \subseteq \mathcal{T}$, and $t_0 \mathrel{\mathcal{R}} t_1$ implies:

- if $t_0 \to_{\mathrm{v}} t_0'$, then there exists t_1' such that $t_1 \to_{\mathrm{v}}^{*} t_1'$ and $t_0' \mathrel{\mathcal{T}} t_1'$;
- if t_0 is a value, then there exists v_1 such that $t_1 \Downarrow_{\mathrm{v}} v_1$, and $t_0 \mathrel{\mathcal{S}^{\mathsf{v}}} v_1$;

- if t_0 is a normal form but not a value, then there exist t'_1 such that $t_1 \Downarrow_{\mathrm{v}} t'_1$ and $t_0 \mathcal{T}^{\mathsf{nf}} t'_1$;
- the converse of the above conditions on t_1.

A normal-form bisimulation is a relation $\mathcal{R}$ such that $\mathcal{R} \rightarrowtail\!\!\!\!\twoheadrightarrow \mathcal{R}, \mathcal{R}$. Normal-form bisimilarity $\mathscr{N}$ is the largest normal-form bisimulation.

Testing values is passive, as we want to prevent the use of bisimulation up to context in that case (if $v_0 \mathcal{R} v_1$, then $v_0\, x$ and $v_1\, x$ are automatically in a bisimulation up to context). The remaining clauses of the bisimulation are active.

We show how to prove equivalences with normal-form bisimulation with our running examples.

Example 6.2 (double reset)**.** We prove that $\langle t\rangle \mathscr{N} \langle\langle t\rangle\rangle$ by showing that $\mathcal{R} \stackrel{\mathsf{def}}{=} \{(\langle\langle t\rangle, \langle\langle t\rangle\rangle) \mid t \in \mathcal{T}\} \cup \mathscr{N}$ is a normal-form bisimulation. On top of reduction steps, for which we can conclude as in Example 4.8, we have to consider the case $\langle t\rangle = F[xv]$. Then by Proposition 2.9, there exists F' such that $F = \langle F'\rangle$. Therefore, we have $\langle\langle t\rangle\rangle = \langle\langle F'[x\, v]\rangle\rangle$. We have $v \mathscr{N}^{\mathsf{v}} v$, and we have to prove that $\langle F'\rangle \mathcal{R}^{\mathsf{c}} \langle\langle F'\rangle\rangle$ holds 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 $\mathscr{N} \subseteq \mathcal{R}$. 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 $\mathscr{N} \subseteq \mathcal{R}$).

Similarly, if $\langle\langle t\rangle\rangle = F[x\, v]$, then we can show that there exists F' such that $F = \langle\langle F'\rangle\rangle$ and $\langle t\rangle \Downarrow_{\mathrm{v}} \langle F'[x\, v]\rangle$, and we can conclude as in the previous case. As we can see, the proof is longer that with applicative (Example 4.8) or environmental (Example 5.13) bisimilarities, just because we have to consider open-stuck terms.

Example 6.3 (Turing's combinator)**.** We prove that Turing's combinator Θ_{v} is bisimilar to its variant $\Theta_{\mathcal{S}}$ by building the candidate relation $\mathcal{R}$ incrementally, starting from $(\Theta_{\mathsf{v}}, \Theta_{\mathcal{S}})$. Evaluating these two terms, we obtain

$$\Theta_{\mathsf{v}} \Downarrow_{\mathrm{v}} \lambda y.y\, (\lambda z.\theta\, \theta\, y\, z) \stackrel{\mathsf{def}}{=} v_0, \text{ and}$$

$$\Theta_{\mathcal{S}} \Downarrow_{\mathrm{v}} \lambda y.y\, (\lambda z.(\lambda x.\langle\theta\, x\rangle)\, (\lambda x.\langle\theta\, x\rangle)\, y\, z) \stackrel{\mathsf{def}}{=} v_1.$$

Evaluating $(v_0\, y, v_1\, y)$ for a fresh y, we obtain two open-stuck terms, so we add their decomposition to $\mathcal{R}$. Let $v'_0 \stackrel{\mathsf{def}}{=} \lambda z.\theta\, \theta\, y\, z$ and $v'_1 \stackrel{\mathsf{def}}{=} \lambda z.(\lambda x.\langle\theta\, x\rangle)\, (\lambda x.\langle\theta\, x\rangle)\, y\, z$; then we add $(v'_0\, z, v'_1\, z)$ and (z, z) for a fresh z to $\mathcal{R}$. Evaluating $v'_0\, z$ and $v'_1\, z$, we obtain respectively $y\, v'_0\, z$ and $y\, v'_1\, z$; to relate these two open stuck terms, we just need to add $(x\, z, x\, z)$ (for a fresh x) to $\mathcal{R}$, since we already have $v'_0 \mathcal{R}^{\mathsf{v}} v'_1$. The constructed relation $\mathcal{R}$ we obtain is a normal-form bisimulation.

As we can see, the proof is much simpler than with applicative (Example 4.9) or environmental (Example 5.16) bisimulations, even using the plain definition of normal-form bisimulation. We can further simplify the definition of the candidate relation thanks to up-to techniques.

6.2. **Up-to Techniques, soundness, and completeness.** The already quite tractable equivalence proofs based on normal-form bisimulation can be further simplified with up-to techniques. Unlike with environmental bisimilarity, we define smaller techniques in Figure 8 which, when combined together, correspond to the usual bisimulation up to related contexts.

$$\frac{}{t \;\mathsf{refl}(\mathcal{R})\; t} \qquad \frac{t_0 \;\mathcal{R}\; t_1}{\lambda x.t_0 \;\mathsf{lam}(\mathcal{R})\; \lambda x.t_1} \qquad \frac{t_0 \;\mathcal{R}\; t_1}{\mathcal{S}k.t_0 \;\mathsf{shift}(\mathcal{R})\; \mathcal{S}k.t_1} \qquad \frac{t_0 \;\mathcal{R}\; t_1 \qquad v_0 \;\mathcal{R}^{\mathsf{v}}\; v_1}{t_0\{v_0/x\} \;\mathsf{subst}(\mathcal{R})\; t_1\{v_1/x\}}$$

$$\frac{t_0 \rightarrow_{\mathrm{v}}^* t'_0 \qquad t_1 \rightarrow_{\mathrm{v}}^* t'_1 \qquad t'_0 \;\mathcal{R}\; t'_1}{t_0 \;\mathsf{red}(\mathcal{R})\; t_1} \qquad \frac{t_0 \;\mathcal{R}\; t_1 \qquad E_0 \;\mathcal{R}^{\mathsf{c}}\; E_1}{E_0[t_0] \;\mathsf{pctx}(\mathcal{R})\; E_1[t_1]} \qquad \frac{t_0 \;\mathcal{R}\; t_1 \qquad \langle E_0 \rangle \;\mathcal{R}^{\mathsf{c}}\; \langle E_1 \rangle}{\langle E_0[t_0] \rangle \;\mathsf{pctxrst}(\mathcal{R})\; \langle E_1[t_1] \rangle}$$

$$\frac{t_0 \;\mathcal{R}\; t_1 \qquad t_0, t_1 \text{ pure} \qquad F_0[x] \;\mathcal{R}\; F_1[x] \qquad x \text{ fresh}}{F_0[t_0] \;\mathsf{ectxpure}(\mathcal{R})\; F_1[t_1]}$$

Figure 8: Up-to techniques for normal-form bisimilarity

Such a fine-grained approach allows for a finer classification between strong and regular up-to techniques.[5]

The technique red is the usual bisimulation up-to reduction, lam and shift allow compatibility w.r.t. λ-abstraction and shift, while compatibility for variables is a consequence of refl, as we have $x \;\mathsf{refl}(\mathcal{R})\; x$ for all x. Bisimulation up to substitution subst is not uncommon for normal-form bisimilarity [52, 55]. The remaining techniques deal with evaluation contexts and behave the same way as the bisimulation up to related contexts of Section 5.5: each of them factors out related contexts, and not simply a common context. We compare contexts using $\cdot^{\mathsf{c}}$ except for ectxpure, which uses a more naive test, as this technique plugs contexts with only pure terms (values or delimited terms), which cannot decompose the contexts. The usual bisimulation up to related contexts can be obtained by composing the three up-to techniques about contexts.

Lemma 6.4. *If $t_0 \;\mathcal{R}\; t_1$ and $F_0 \;\mathcal{R}^{\mathsf{c}}\; F_1$ then $F_0[t_0] \;(\mathsf{pctx} \cup (\mathsf{ectxpure} \circ \mathsf{pctxrst}))(\mathcal{R})\; F_1[t_1]$.*

We can also derive compatibility w.r.t. application from refl and pctx.

Lemma 6.5. *If $t_0 \;\mathcal{R}\; t'_0$ and $t_1 \;\mathcal{R}\; t'_1$, then $t_0\, t_1 \;(\mathsf{pctx} \circ (\mathsf{id} \cup \mathsf{pctx} \circ (\mathsf{id} \cup \mathsf{refl})))(\mathcal{R})\; t'_0\, t'_1$.*

Proof. Let x be a fresh variable; then $x \,\square \;\mathsf{refl}(\mathcal{R})^{\mathsf{c}}\; x \,\square$. Combined with $t_1 \;\mathcal{R}\; t'_1$, it implies $x\, t_1 \;\mathsf{pctx}((\mathsf{id} \cup \mathsf{refl})(\mathcal{R}))\; x\, t'_1$, i.e., $\square\, t_1 \;\mathsf{pctx}((\mathsf{id} \cup \mathsf{refl})(\mathcal{R}))^{\mathsf{c}}\; \square\, t'_1$. This combined with $t_0 \;\mathcal{R}\; t'_0$ using pctx gives the required result. $\square$

Finally, compatibility w.r.t. reset can be deduced from pctxrst by taking the empty context. (Defining a dedicated up-to technique for reset would have some merit since it could be proved strong, unlike pctxrst [17].)

Theorem 6.6. *The set $\mathfrak{F} \stackrel{\mathsf{def}}{=} \{\mathsf{refl}, \mathsf{lam}, \mathsf{shift}, \mathsf{subst}, \mathsf{pctx}, \mathsf{pctxrst}, \mathsf{ectxpure}, \mathsf{id}, \mathsf{red}\}$ is diacritically compatible, with $\mathsf{strong}(\mathfrak{F}) = \mathfrak{F} \setminus \{\mathsf{pctx}, \mathsf{pctxrst}, \mathsf{ectxpure}\}$.*

We explain what sets apart pctx, pctxrst, and ectxpure from the other techniques by sketching the progress proof for pctx.

Sketch. Let $\mathcal{R} \rightarrowtail \mathcal{R}, \mathcal{S}$, and $E_0[t_0] \;\mathsf{pctx}(\mathcal{R})\; E_1[t_1]$ such that $E_0 \;\mathcal{R}^{\mathsf{c}}\; E_1$ and $t_0 \;\mathcal{R}\; t_1$. We proceed by case analysis on E_0 and t_0. Most cases are straightforward; the problematic case is when t_0 is a variable x and $E_0 = \square\, v$. Because $t_0 \;\mathcal{R}\; t_1$, there exists v_1 such that

[5] We do not do the same with environmental bisimilarity, because unlike normal-form bisimilarity, it is defined primarily on closed terms, and therefore we do not consider, e.g., a bisimulation up to λ-abstraction with environmental bisimilarity.

$t_1 \Downarrow_{\mathsf{v}} v_1$ and $x \mathrel{\mathcal{R}^{\mathsf{v}}} v_1$. Because $E_0 \mathrel{\mathcal{R}^{\mathsf{c}}} E_1$, we have $E_0[y] \mathrel{\mathcal{R}} E_1[y]$ for a fresh y, and therefore $E_0[x] \mathrel{\mathsf{subst}(\mathcal{R})} E_1[v_1]$. We can conclude with subst, assuming it has been proved before: there exists t'_1 such that $E_1[t_1] \rightarrow^*_{\mathsf{v}} E_1[v_1] \rightarrow^*_{\mathsf{v}} t'_1$ and $x\, v \mathrel{\widehat{\mathfrak{F}}^{\omega}(\mathcal{S})^{\mathsf{nf}}} t'_1$.
If we try to prove pctx strong, we would have $\mathcal{R} \rightarrowtail\!\!\!\!\twoheadrightarrow \mathcal{S}, \mathcal{T}$ as a hypothesis. In the subcase sketched above, we would have $x \mathrel{\mathcal{S}^{\mathsf{v}}} v_1$ and $E_0[x] \mathrel{\mathsf{subst}(\mathcal{S})} E_1[v_1]$ instead of $\mathcal{R}$, and since there is no progress hypothesis on $\mathcal{S}$, we could not conclude. The techniques $\mathsf{ectxpure}$ and $\mathsf{pctxrst}$ have the same problematic subcase.

Since compatibility w.r.t. the operators of the language can be deduced from the techniques of Figure 8, we can conclude that $\mathscr{N}$ is compatible using Lemma 5.5. We can then show that $\mathscr{N}$ is sound w.r.t. $\mathscr{C}$.

Theorem 6.7. *We have $\mathscr{N} \subseteq \mathscr{C}$.*

The following counter-example, inspired by Lassen [53], shows that the inclusion is in fact strict; normal-form bisimilarity is not complete.

Proposition 6.8. *We have $\langle x\, i\rangle \mathrel{\mathscr{C}^\circ} (\lambda y.\langle x\, i\rangle)\, \langle x\, i\rangle$, but these terms are not normal-form bisimilar.*

Proof. We prove that $\langle x\, i\rangle \mathrel{\mathscr{C}^\circ} (\lambda y.\langle x\, i\rangle)\, \langle x\, i\rangle$ holds using applicative bisimilarity in Proposition 4.19. They are not normal-form bisimilar, because the terms $\langle z\rangle$ and $(\lambda y.\langle x\, i\rangle)\, \langle z\rangle$ (where z is fresh) are not bisimilar: the former evaluates to z while the latter evaluates to an open-stuck term. □

Lassen's other counter-example can also be adapted to $\lambda_{\mathcal{S}}$: we can show that $\langle x\, y\rangle\, \Omega$ and Ω are contextually equivalent but not normal-form bisimilar.

Remark 6.9. Following Filinski's simulation of shift and reset in terms of $\mathsf{call/cc}$ and a single reference cell [27], one can express the terms of the $\lambda_{\mathcal{S}}$-calculus in the $\lambda\mu\rho$-calculus [88], a calculus with store and a construct similar to $\mathsf{call/cc}$. Yet, Støvring and Lassen's normal-form bisimilarity is sound and complete with respect to the contextual equivalence of $\lambda\mu\rho$ [88], while our relation is only sound. It shows that $\lambda\mu\rho$ is more expressive and can distinguish more terms than $\lambda_{\mathcal{S}}$, mostly because of the state construct. For example, the encodings of the two terms of Proposition 6.8 in $\lambda\mu\rho$ would not be contextually equivalent in $\lambda\mu\rho$, since substituting for x a value that, e.g., increments a value of some reference cell, would lead to two different states that can be easily distinguished observationally.

We show how up-to techniques can simplify the definitions of candidate relations on several examples, starting with the double reset one.

Example 6.10 (double reset)**.** To relate $\langle t\rangle$ and $\langle\langle t\rangle\rangle$, we can avoid the case analysis of Example 6.2 by noticing that $\langle t\rangle \mathrel{\mathsf{ectxpure} \circ (\mathsf{id} \cup \mathsf{refl})(\mathcal{R})} \langle\langle t\rangle\rangle$ holds with $\mathcal{R} \overset{\mathsf{def}}{=} \{(x, \langle x\rangle)\}$, since $\langle t\rangle$ is pure. We then conclude by showing that $\mathcal{R}$ is a bisimulation (eventually up to something), which is straightforward.

Bisimulation up to related contexts is easier to use with normal-form bisimilarity compared to environmental bisimilarity, as we can plug any related terms into any contexts (thanks to lam and shift), without the restriction of Section 5.5 that non-value terms are limited to evaluation contexts. As a result, the equivalence proof for Turing's combinator and its variant can be greatly simplified, as we can see below.

Example 6.11 (Turing's combinator)**.** Let

$$v_0 \stackrel{\mathsf{def}}{=} \lambda y.y\,(\lambda z.\Theta_{\mathsf{v}}\,y\,z) \text{ and } v_1 \stackrel{\mathsf{def}}{=} \lambda y.y\,(\lambda z.(\lambda x.\langle\theta\,x\rangle)\,(\lambda x.\langle\theta\,x\rangle)\,y\,z).$$

The relation

$$\mathcal{R} \stackrel{\mathsf{def}}{=} \{(\Theta_{\mathsf{v}}, \Theta_{\mathcal{S}}), (v_0\,y, v_1\,y), (\Theta_{\mathsf{v}}, (\lambda x.\langle\theta\,x\rangle)\,(\lambda x.\langle\theta\,x\rangle)) \mid y \text{ fresh}\}$$

is a bisimulation up to context up to reduction. Indeed, we remind that $\Theta_{\mathsf{v}} \Downarrow_{\mathrm{v}} v_0$ and $\Theta_{\mathcal{S}} \Downarrow_{\mathrm{v}} v_1$. Reducing $v_0\,y$ and $v_1\,y$ for a fresh y, we get terms sharing the common context $y\,(\lambda z.\square\,y\,z)$, and the two terms filling the holes (respectively Θ_{v} and $(\lambda x.\langle\theta\,x\rangle)\,(\lambda x.\langle\theta\,x\rangle)$) are in $\mathcal{R}$. We can conclude using bisimulation up to related contexts, as we use it after an active reduction step. The terms Θ_{v} and $(\lambda x.\langle\theta\,x\rangle)\,(\lambda x.\langle\theta\,x\rangle)$ also reduce respectively to v_0 and v_1, so we can conclude in the same way.

As an extra example, we prove a variant of the β_Ω axiom; the axiom itself is proved in Section 6.5.

Example 6.12. If $x \notin \mathsf{fv}(E)$, then $\langle(\lambda x.\langle E[x]\rangle)\,t\rangle \mathrel{\mathscr{N}} \langle E[t]\rangle$. Indeed, if

$$\mathcal{R} \stackrel{\mathsf{def}}{=} \{(\langle\langle(\lambda x.\langle E[x]\rangle)\,y\rangle, \langle E[y]\rangle) \mid y \text{ fresh}\}.$$

then $\langle(\lambda x.\langle E[x]\rangle)\,t\rangle \mathrel{\mathsf{pctxrst} \circ (\mathsf{id} \cup \mathsf{refl})(\mathcal{R})} \langle E[t]\rangle$. Furthermore, $\langle(\lambda x.\langle E[x]\rangle)\,y\rangle \to_{\mathrm{v}} \langle\langle E[y]\rangle\rangle$, and we prove in Example 6.2 that $\langle\langle E[y]\rangle\rangle \mathrel{\mathscr{N}} \langle E[y]\rangle$; therefore $\mathcal{R} \subseteq \mathsf{red}(\mathscr{N})$, and we can conclude from here.

6.3. **Refined Normal-Form Bisimilarity.** The normal-form bisimulation of Definition 6.1 is too discriminating with control-stuck terms, as we can see with the following example.

Proposition 6.13. *We have $\mathcal{S}k.i \mathrel{\mathscr{C}} (\mathcal{S}k.i)\Omega$, but these terms are not normal-form bisimilar.*

Proof. We can easily prove that $\mathcal{S}k.i \mathrel{\mathscr{C}} (\mathcal{S}k.i)\,\Omega$ holds with applicative bisimilarity or Definition 6.15. They are not normal-form bisimilar, since the contexts $\square$ and $\square\,\Omega$ are not related by $\mathscr{N}^{\mathsf{nf}}$ (x converges while $x\,\Omega$ diverges). $\square$

When comparing two control-stuck terms $E_0[\mathcal{S}k.t_0]$ and $E_1[\mathcal{S}k.t_1]$, normal-form bisimilarity considers the contexts E_0, E_1 and the shift bodies t_0, t_1 separately, while they are combined if the control-stuck terms are put under a reset and the capture goes through. We propose a more refined definition of normal-form bisimulation which tests stuck terms by simulating the capture, while taking into account the fact that a context bigger than E_0 and E_1 can be captured. We do so by introducing a *context variable* to represent the context beyond E_0 and E_1. We let c range over a set of context variables. We introduce such a variable when simulating a capture, where the context is always captured with its reset. To simulate this, we suppose that c stands for a pure context surrounded by a delimiter. As a result, the definition of $\mathcal{R}^{\mathsf{nf}}$ on control-stuck terms becomes as in Figure 9.

Remark 6.14. We could try to use a regular variable k' to play the role of a context variable, and define the extension $\mathcal{R}^{\mathsf{nf}}$ on control-stuck terms as follows:

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

However, such variables are substituted with contexts and not with values, and so they have to be treated separately from regular variables.

$$\frac{D_0[x] \mathrel{\mathcal{R}} D_1[x] \qquad F_0[x] \mathrel{\mathcal{R}} F_1[x] \qquad x \text{ fresh}}{F_0[D_0] \mathrel{\mathcal{R}^{\mathsf{c}}} F_1[D_1]}$$

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

Up-to techniques specific to refined bisimilarity

$$\frac{t_0 \mathrel{\mathcal{R}} t_1}{c[t_0] \mathrel{\mathsf{cvar}(\mathcal{R})} c[t_1]} \qquad\qquad \frac{t_0 \mathrel{\mathcal{R}} t_1 \qquad D_0 \mathrel{\mathcal{R}^{\mathsf{c}}} D_1}{t_0\{D_0/c\} \mathrel{\mathsf{csubst}(\mathcal{R})} t_1\{D_1/c\}}$$

Figure 9: Extension to normal forms and up-to techniques for the refined bisimilarity

Formally we extend the syntax of terms and evaluation contexts F, and we introduce a new kind of *delimited contexts* ranged over by D.

Terms:	$t ::= \ldots \mid c[t]$
Evaluation contexts:	$F ::= \ldots \mid c[F]$
Delimited contexts:	$D ::= \langle E\rangle \mid c[E]$

We write $t\{D/c\}$ for the context substitution of c by D in t, defined so that $(c[t])\{D/c\} \stackrel{\mathsf{def}}{=} D[t\{D/c\}]$, $(c'[t])\{D/c\} \stackrel{\mathsf{def}}{=} c'[t\{D/c\}]$ if $c' \neq c$, and the substitution is propagated recursively on subterms in the other cases. The capture reduction rule is changed to take delimited contexts into account.

$$F[D[\mathcal{S}k.t]] \rightarrow_{\mathrm{v}} F[\langle t\{\lambda x.D[x]/k\}\rangle] \text{ with } x \notin \mathsf{fv}(D) \quad (\textit{shift})$$

Given a relation $\mathcal{R}$ on extended open terms, we keep the definitions of $\mathcal{R}^{\mathsf{v}}$, $\mathcal{R}^{\mathsf{nf}}$ on open-stuck terms, and $\mathcal{R}^{\mathsf{c}}$ on pure contexts as in Figure 7, and we change $\mathcal{R}^{\mathsf{nf}}$ on control-stuck terms and $\mathcal{R}^{\mathsf{c}}$ on any contexts as in Figure 9. The latter change is to account for delimited contexts. The extended calculus also features a new kind of normal forms, of the shape $F[c[v]]$, called *context-stuck terms*. They are similar to open-stuck terms but are tested differently, as we can see in the definition of progress.

Definition 6.15. A relation $\mathcal{R}$ on extended open terms diacritically progresses to $\mathcal{S}$, $\mathcal{T}$ written $\mathcal{R} \rightarrowtail\!\!\!\!\rightarrow_{\mathsf{r}} \mathcal{S}, \mathcal{T}$, if $\mathcal{R} \subseteq \mathcal{S}$, $\mathcal{S} \subseteq \mathcal{T}$, and $t_0 \mathrel{\mathcal{R}} t_1$ implies:

- if $t_0 \rightarrow_{\mathrm{v}} t_0'$, then there exists t_1' such that $t_1 \rightarrow_{\mathrm{v}}^{*} t_1'$ and $t_0' \mathrel{\mathcal{T}} t_1'$;
- if t_0 is a value, then there exists v_1 such that $t_1 \Downarrow_{\mathrm{v}} v_1$, and $t_0 \mathrel{\mathcal{S}^{\mathsf{v}}} v_1$;
- if $t_0 = F_0[c[v_0]]$, then there exists F_1, v_1 such that $t_1 \Downarrow_{\mathrm{v}} F_1[c[v_1]]$, $F_0[\langle\Box\rangle] \mathrel{\mathcal{T}^{\mathsf{c}}} F_1[\langle\Box\rangle]$, and $v_0 \mathrel{\mathcal{S}^{\mathsf{v}}} v_1$;
- if t_0 is an open-stuck or control-stuck term, then there exists t_1' such that $t_1 \Downarrow_{\mathrm{v}} t_1'$ and $t_0 \mathrel{\mathcal{T}^{\mathsf{nf}}} t_1'$;
- the converse of the above conditions on t_1.

A refined normal-form bisimulation is a relation $\mathcal{R}$ such that $\mathcal{R} \rightarrowtail\!\!\!\!\rightarrow_{\mathsf{r}} \mathcal{R}, \mathcal{R}$. Refined normal-form bisimilarity $\mathscr{R}$ is the largest refined normal-form bisimulation.

The clause for context-stuck terms relates the terms $F_0[c[v_0]]$ and $F_1[c[v_1]]$ by comparing the contexts $F_0[\langle\Box\rangle]$ and $F_1[\langle\Box\rangle]$ because c implicitly includes a reset. This essentially amounts to equate $F_0[x]$ and $F_1[x]$ for a fresh x. In contrast with open-stuck terms, we relate the contexts with $\mathcal{T}$ but the values with $\mathcal{S}$, thus forbidding the use of regular up-to techniques when comparing values. Our goal is to prevent the application of the new csubst technique in that case; we explain why after Theorem 6.18.

To compare refined bisimilarity to the other relations on $\lambda_{\mathcal{S}}$, we translate the terms of the extended calculus back to $\lambda_{\mathcal{S}}$. Given an injective mapping f from context variables to regular variables, we define the translation $[\![\cdot]\!]^f$ on extended terms so that $[\![c[t]]\!]^f \stackrel{\mathsf{def}}{=} \langle f(c)\ [\![t]\!]^f\rangle$ and so that it is recursively applied to subterms in the other cases. The translation is defined on contexts in a similar way. It is easy to see that if t is a plain $\lambda_{\mathcal{S}}$-term, then $[\![t]\!]^f = t$ for all f, and that reduction is preserved by the translation.

Proposition 6.16. *For all t, t', and f, $t \rightarrow_v^* t'$ iff $[\![t]\!]^f \rightarrow_v^* [\![t']\!]^f$.*

We can relate $\mathscr{N}$ and $\mathscr{R}$ thanks to the translation.

Proposition 6.17. *For all t_0, t_1, and f such that the image of f does not intersect $\mathsf{fv}(t_0)$ and $\mathsf{fv}(t_1)$, if $[\![t_0]\!]^f\ \mathscr{N}\ [\![t_1]\!]^f$, then $t_0\ \mathscr{R}\ t_1$.*

The condition on f allows for the distinction between the evaluations to context-stuck and open-stuck terms. If $[\![t_0]\!]^f \Downarrow_{\mathrm{v}} F[x\,v]$ for some F, x and v, then either $x \in \mathsf{fv}(t_0)$, $t_0 \Downarrow_{\mathrm{v}} F'[x\,v']$, $[\![F']\!]^f = F$, and $[\![v']\!]^f = v$, or $x = f(c)$ for some c, $t_0 \Downarrow_{\mathrm{v}} F'[c[v']]$, $[\![F'[\langle\Box\rangle]]\!]^f = F$, and $[\![v']\!]^f = v$.

Proof. We prove that $\mathcal{R} \stackrel{\mathsf{def}}{=} \{(t_0, t_1) \mid [\![t_0]\!]^f\ \mathscr{N}\ [\![t_1]\!]^f\}$ is a refined bisimulation. What needs to be checked are context-stuck terms and control-stuck terms. If $t_0 = F_0[c[v_0]]$, then $[\![t_0]\!]^f = [\![F_0]\!]^f[\langle f(c)\ [\![v_0]\!]^f\rangle]$, and there exists F_1, v_1 such that $[\![t_1]\!]^f \Downarrow_{\mathrm{v}} [\![F_1]\!]^f[\langle f(c)\ [\![v_1]\!]^f\rangle]$ with $[\![t_0]\!]^f\ \mathscr{N}^{\mathsf{nf}}\ [\![F_1]\!]^f[\langle f(c)\ [\![v_1]\!]^f\rangle]$, i.e., $[\![F_0[\langle\Box\rangle]]\!]^f\ \mathscr{N}^{\mathsf{c}}\ [\![F_1[\langle\Box\rangle]]\!]^f$ and $[\![v_0]\!]^f\ \mathscr{N}^{\mathsf{v}}\ [\![v_1]\!]^f$. Therefore we have $t_1 \Downarrow_{\mathrm{v}} F_1[c[v_1]]$, and the clause for context-stuck terms is verified.

If $t_0 = E_0[\mathcal{S}k.t_0']$, then $[\![t_0]\!]^f = [\![E_0]\!]^f[\mathcal{S}k.[\![t_0']\!]^f]$, and there exists E_1, t_1' such that $[\![t_1]\!]^f \Downarrow_{\mathrm{v}} [\![E_1]\!]^f[\mathcal{S}k.[\![t_1']\!]^f]$, $[\![E_0]\!]^f\ \mathscr{N}^{\mathsf{c}}\ [\![E_1]\!]^f$, and $\langle[\![t_0']\!]^f\rangle\ \mathscr{N}\ \langle[\![t_1']\!]^f\rangle$. But $\mathscr{N}$ is compatible and substitutive, therefore we have

$$\langle[\![t_0']\!]^f\{\lambda x.\langle f(c)\ [\![E_0]\!]^f[x]\rangle/k\}\rangle\ \mathscr{N}\ \langle[\![t_1']\!]^f\{\lambda x.\langle f(c)\ [\![E_1]\!]^f[x]\rangle/k\}\rangle$$

for some fresh x and c. Consequently, we have $\langle t_0'\{\lambda x.c[E_0[x]]/k\}\rangle\ \mathcal{R}\ \langle t_1'\{\lambda x.c[E_1[x]]/k\}\rangle$, as wished. □

A direct consequence of Proposition 6.17 is that $\mathscr{N} \subset \mathscr{R}$. The inclusion is strict, because $\mathscr{R}$ relates the terms of Proposition 6.13, while $\mathscr{N}$ does not.

Up-to techniques and soundness. The up-to techniques for refined bisimilarity are the same as for normal-form bisimilarity (Figure 8), except that we add techniques specific to context variables csubst and cvar (defined in Figure 9), and we remove $\mathsf{pctxrst}$, as it can be directly expressed in terms of the two new techniques. In fact, csubst is a bit more powerful than $\mathsf{pctxrst}$, as several copies of the same context can be abstracted away with csubst against only one for $\mathsf{pctxrst}$. For the $\mathsf{ectxpure}$ technique, pure terms now include terms of the form $c[t]$ in addition to values and delimited terms $\langle t\rangle$.

Theorem 6.18. *The set $\mathfrak{F} \stackrel{\mathsf{def}}{=} \{\mathsf{refl}, \mathsf{lam}, \mathsf{shift}, \mathsf{cvar}, \mathsf{subst}, \mathsf{csubst}, \mathsf{pctx}, \mathsf{ectxpure}, \mathsf{id}, \mathsf{red}\}$ is diacritically compatible, with $\mathsf{strong}(\mathfrak{F}) = \mathfrak{F} \setminus \{\mathsf{csubst}, \mathsf{pctx}, \mathsf{ectxpure}\}$.*

Unsurprisingly, the technique csubst is not strong as it behaves like $\mathsf{pctxrst}$. In particular, it exhibits the same problematic subcase as the one presented after Theorem 6.6, by taking $t_0 \stackrel{\mathsf{def}}{=} c[x]$ and $D_0 = \langle \Box\, v \rangle$. Then from $t_0 \;\mathcal{R}\; t_1$ and $\mathcal{R} \rightarrowtail\!\!\!\rightarrow \mathcal{R}, \mathcal{S}$, we know there exist F_1 and v_1 such that $t_1 \Downarrow_{\mathrm{v}} F_1[c[v_1]]$, $\Box\; \mathcal{S}^{\mathsf{c}}\; F_1$, and $x \;\mathcal{R}^{\mathsf{v}}\; v_1$. From there, we can conclude as in Section 6.2, using subst; more details are given in the appendix. To conclude in that case, it is important to have $x \;\mathcal{R}^{\mathsf{v}}\; v_1$ and not $x \;\mathcal{S}^{\mathsf{v}}\; v_1$, justifying why the test for values is passive for context-stuck terms.

From Theorem 6.18 and Proposition 5.5, we deduce that $\mathscr{R}$ is compatible, which we then use to show that $\mathscr{R}$ is sound w.r.t. $\mathscr{C}$ in the following sense.

Theorem 6.19. *For all* $(t_0, t_1) \in \mathcal{T}^2$*, if* $t_0 \;\mathscr{R}\; t_1$*, then* $t_0 \;\mathscr{C}\; t_1$*.*

As we restrict t_0 and t_1 to plain $\lambda_{\mathcal{S}}$ terms, we do not need to work up to the translation. The relation $\mathscr{R}$ is not complete because it still does not relate the terms of Proposition 6.8. We would like to stress that even though $\mathscr{R}$ equates more contextually equivalent terms than $\mathscr{N}$, the latter is still useful, since it leads to very simple proofs of equivalence, as we can see with the examples of Sections 6.2 and 6.5. Therefore, $\mathscr{R}$ does not disqualify $\mathscr{N}$ as a proof technique. In fact, they can be used together, as in the next example.

Example 6.20. If $k' \notin \mathsf{fv}(E) \cup \mathsf{fv}(t)$ and $x \notin \mathsf{fv}(E)$, then $E[\mathcal{S}k.t] \;\mathscr{R}\; \mathcal{S}k'.t\{\lambda x.\langle k'\; E[x]\rangle/k\}$. The two terms are control stuck, therefore we have to prove that $\langle t\{\lambda x.c[E[x]]/k\}\rangle \;\mathscr{R}\; \langle t\{\lambda x.\langle(\lambda y.c[y])\; E[x]\rangle/k\}\rangle$ holds for a fresh c. Let f be an injective mapping verifying the conditions of Proposition 6.17. We know that $\langle f(c)\; E[x]\rangle \;\mathscr{N}\; \langle(\lambda y.\langle f(c)\; y\rangle)\; E[x]\rangle$ holds by Example 6.12, so we have $c[E[x]] \;\mathscr{R}\; \langle(\lambda y.c[y])\; E[x]\rangle$ by Proposition 6.17. We can then conclude using refl, lam, and subst.

Proving this result using only the regular normal-form bisimulation would require us to equate $E[y]$ and y (where y is fresh), which is not true in general (take $E = (\lambda z.\Omega)\; \Box$).

6.4. **Normal-Form Bisimulation for the Original Semantics.** Any sound bisimilarity for the relaxed semantics, such as $\mathscr{N}$ or $\mathscr{R}$, is also sound for the original semantics. We define in this section a bisimilarity which, while being not complete w.r.t. $\mathscr{P}$, still relates more terms in the original semantics than $\mathscr{N}$ or $\mathscr{R}$. We follow the same principle as in Section 5.4, and define a bisimilarity which primarily compares pure terms. We then extend it to any terms by introducing a context variable which stands for a potential evaluation context, as with refined bisimilarity.

Formally, we work on the extended calculus of Section 6.3, and we let p range over pure terms, which are now of three possible shapes.

Pure terms: $\quad p ::= v \mid \langle t \rangle \mid c[t]$

We update the definition of $\cdot^{\mathsf{v}}$, $\cdot^{\mathsf{c}}$, and $\cdot^{\mathsf{nf}}$ in Figure 10. Because we work on pure terms, the control-stuck terms case has been removed; similarly, the evaluation contexts in the context-stuck and open-stuck terms cases are delimited, so the pure context case of $\cdot^{\mathsf{c}}$ is no longer useful. The definition of $\cdot^{\mathsf{v}}$ has been changed so that we compare pure terms in its premise.

Definition 6.21. A relation $\mathcal{R}$ on extended pure open terms diacritically progresses to $\mathcal{S}$, $\mathcal{T}$ written $\mathcal{R} \rightarrowtail\!\!\!\rightarrow_{\mathsf{o}} \mathcal{S}, \mathcal{T}$, if $\mathcal{R} \subseteq \mathcal{S}$, $\mathcal{S} \subseteq \mathcal{T}$, and $p_0 \;\mathcal{R}\; p_1$ implies:

- if $p_0 \rightarrow_{\mathrm{v}} p_0'$, then there exists p_1' such that $p_1 \rightarrow_{\mathrm{v}}^{*} p_1'$ and $p_0' \;\mathcal{T}\; p_1'$;

$$\frac{D_0[x] \mathrel{\mathcal{R}} D_1[x] \qquad F_0[x] \mathrel{\mathcal{R}} F_1[x] \qquad x \text{ fresh}}{F_0[D_0] \mathrel{\mathcal{R}^{\mathsf{c}}} F_1[D_1]} \qquad \frac{c[v_0\, x] \mathrel{\mathcal{R}} c[v_1\, x] \qquad x, c \text{ fresh}}{v_0 \mathrel{\mathcal{R}^{\mathsf{v}}} v_1}$$

$$\frac{F_0 \mathrel{\mathcal{R}^{\mathsf{c}}} F_1 \qquad v_0 \mathrel{\mathcal{R}^{\mathsf{v}}} v_1}{F_0[x\, v_0] \mathrel{\mathcal{R}^{\mathsf{nf}}} F_1[x\, v_1]}$$

Up-to techniques

$$\frac{}{p \mathrel{\mathsf{refl}(\mathcal{R})} p} \qquad \frac{p_0 \mathrel{\mathcal{R}} p_1}{c[\lambda x.p_0] \mathrel{\mathsf{lam}(\mathcal{R})} c[\lambda x.p_1]} \qquad \frac{p_0 \mathrel{\mathcal{R}} p_1}{c[\mathcal{S}k.p_0] \mathrel{\mathsf{shift}(\mathcal{R})} c[\mathcal{S}k.p_1]}$$

$$\frac{p_0 \mathrel{\mathcal{R}} p_1}{c[p_0] \mathrel{\mathsf{cvar}(\mathcal{R})} c[p_1]} \qquad \frac{p_0 \mathrel{\mathcal{R}} p_1 \qquad v_0 \mathrel{\mathcal{R}^{\mathsf{v}}} v_1}{p_0\{v_0/x\} \mathrel{\mathsf{subst}(\mathcal{R})} p_1\{v_1/x\}} \qquad \frac{p_0 \mathrel{\mathcal{R}} p_1 \qquad D_0 \mathrel{\mathcal{R}^{\mathsf{c}}} D_1}{p_0\{D_0/c\} \mathrel{\mathsf{csubst}(\mathcal{R})} p_1\{D_1/c\}}$$

$$\frac{p_0 \rightarrow_{\mathrm{v}}^* p_0' \qquad p_1 \rightarrow_{\mathrm{v}}^* p_1' \qquad p_0' \mathrel{\mathcal{R}} p_1'}{p_0 \mathrel{\mathsf{red}(\mathcal{R})} p_1}$$

$$\frac{p_0 \mathrel{\mathcal{R}} p_1 \qquad F_0[x] \mathrel{\mathcal{R}} F_1[x] \qquad F_0[x], F_1[x] \text{ pure} \qquad x \text{ fresh}}{F_0[p_0] \mathrel{\mathsf{ectxpure}(\mathcal{R})} F_1[p_1]}$$

Figure 10: Extension to normal forms and up-to techniques for the original semantics

- if p_0 is a value, then there exists v_1 such that $p_1 \Downarrow_{\mathrm{v}} v_1$, and $p_0 \mathrel{\mathcal{S}^{\mathsf{v}}} v_1$;
- if $p_0 = F_0[c[v_0]]$, then there exists F_1, v_1 such that $p_1 \Downarrow_{\mathrm{v}} F_1[c[v_1]]$, $F_0[\langle\Box\rangle] \mathrel{\mathcal{T}^{\mathsf{c}}} F_1[\langle\Box\rangle]$, and $v_0 \mathrel{\mathcal{S}^{\mathsf{v}}} v_1$;
- if p_0 is an open-stuck term, then there exist p_1' such that $p_1 \Downarrow_{\mathrm{v}} p_1'$ and $p_0 \mathrel{\mathcal{T}^{\mathsf{nf}}} p_1'$;
- the converse of the above conditions on p_1.

A pure normal-form bisimulation is a relation $\mathcal{R}$ such that $\mathcal{R} \rightarrowtail\!\!\twoheadrightarrow_{\mathsf{o}} \mathcal{R}, \mathcal{R}$. Pure normal-form bisimilarity $\mathscr{M}$ is the largest pure normal-form bisimulation.

Again, testing values in the context-stuck terms case is passive, to prevent csubst to be used here; otherwise, from $c[v_0] \mathrel{\mathcal{R}} c[v_1]$, we could relate $c'[v_0\, x]$ and $c'[v_1\, x]$ directly for any v_0 and v_1.

We extend $\mathscr{M}$ to all terms as follows: $t_0 \mathrel{\mathscr{M}} t_1$ if $c[t_0] \mathrel{\mathscr{M}} c[t_1]$ for a fresh c. Pure bisimilarity relates more terms in the original semantics than the normal-form bisimilarities of the relaxed semantics.

Proposition 6.22. *We have $\mathscr{N} \subsetneq \mathscr{M}$ and $\mathscr{R} \subsetneq \mathscr{M}$.*

Proof. Because $\mathscr{N} \subseteq \mathscr{R}$, it is enough to show that $\mathscr{R} \subseteq \mathscr{M}$. Let $t_0 \mathrel{\mathscr{R}} t_1$; because $\mathscr{R}$ is compatible, we have $c[t_0] \mathrel{\mathscr{R}} c[t_1]$. We then prove that $\mathscr{R}$ is a pure bisimulation. On pure terms, the tests of the two notions of bisimulation differ only on values: we have $v_0\, x \mathrel{\mathscr{R}} v_1\, x$, and we need $c[v_0\, x] \mathrel{\mathscr{R}} c[v_1\, x]$. We can easily conclude using again the fact that $\mathscr{R}$ is compatible. □

The inclusions are strict, as we show in Proposition 6.30 that $\mathscr{M}$ verifies the $\mathcal{S}_{\mathtt{elim}}$ axiom while the others two do not.

The up-to techniques for $\mathscr{M}$ are defined in Figure 10; they are essentially the same as for refined bisimilarity with some minor adjustments to ensures that we relate pure terms in the premises as well as in the conclusion. We also remove the now useless pctx technique.

Theorem 6.23. *The set* $\mathfrak{F} \stackrel{\mathsf{def}}{=} \{\mathsf{refl}, \mathsf{lam}, \mathsf{shift}, \mathsf{reset}, \mathsf{subst}, \mathsf{csubst}, \mathsf{ectxpure}, \mathsf{id}, \mathsf{red}\}$ *is diacritically compatible, with* $\mathsf{strong}(\mathfrak{F}) = \mathfrak{F} \setminus \{\mathsf{csubst}, \mathsf{ectxpure}\}$.

We deduce that $\mathscr{M}$ is compatible on pure terms. For compatibility w.r.t. any terms, let $t_0 \mathrel{\mathscr{M}} t_1$; then by definition, $c[t_0] \mathrel{\mathscr{M}} c[t_1]$ for a fresh c. We can then deduce compatibility w.r.t. evaluation contexts (application and reset) with csubst. For the remaining constructs, we need separate proofs that $\lambda x.t_0 \mathrel{\mathscr{M}} \lambda x.t_1$ and $\mathcal{S}k.t_0 \mathrel{\mathscr{M}} \mathcal{S}k.t_1$, but these are straightforward. Consequently, $\mathscr{M}$ is compatible on all terms, and we can show it is sound w.r.t. $\mathscr{P}$.

Theorem 6.24. *For all* $(t_0, t_1) \in \mathcal{T}^2$, *if* $t_0 \mathrel{\mathscr{M}} t_1$, *then* $t_0 \mathrel{\mathscr{P}} t_1$.

Example 6.25. We prove Example 5.15 again with pure normal-form bisimulation: if $k \notin \mathsf{fv}(t_1)$, then $(\lambda x.\mathcal{S}k.t_0)\, t_1 \mathrel{\mathscr{M}} \mathcal{S}k.((\lambda x.t_0)\, t_1)$. We want to relate $c[((\lambda x.\mathcal{S}k.t_0)\, t_1)]$ with $c[\mathcal{S}k.((\lambda x.t_0)\, t_1)]$ for a fresh c. But $c[\mathcal{S}k.((\lambda x.t_0)\, t_1)] \rightarrow_{\mathrm{v}} \langle(\lambda x.t_0\{\lambda y.c[y]/k\})\, t_1\rangle$. Let

$$\mathcal{R} \stackrel{\mathsf{def}}{=} \{(c[((\lambda x.\mathcal{S}k.t_0)\, z)], \langle(\lambda x.t_0\{\lambda y.c[y]/k\})\, z\rangle) \mid z \text{ fresh}\};$$

then $c[((\lambda x.\mathcal{S}k.t_0)\, t_1)] \mathrel{\mathsf{csubst} \circ (\mathsf{id} \cup \mathsf{refl})(\mathcal{R})} \langle(\lambda x.t_0\{\lambda y.c[y]/k\})\, t_1\rangle$. The relation $\mathcal{R}$ is a bisimulation up to red and refl, since we have $c[((\lambda x.\mathcal{S}k.t_0)\, z)] \rightarrow_{\mathrm{v}}^2 \langle t_0\{\lambda y.c[y]/k\}\{z/x\}\rangle$ and $\langle(\lambda x.t_0\{\lambda y.c[y]/k\})\, z\rangle \rightarrow_{\mathrm{v}} \langle t_0\{\lambda y.c[y]/k\}\{z/x\}\rangle$; we obtain two identical terms.

6.5. **Proving the Axioms.** We provide further examples by proving the axioms with normal-form bisimilarities. As usual, the $\beta_{\mathtt{v}}$, $\langle\cdot\rangle_{\mathcal{S}}$, $\langle\cdot\rangle_{\mathtt{val}}$, and $\beta_{\mathtt{v}}$ axioms are consequences of the fact that reduction is included in the bisimilarity.

Proposition 6.26. *If* $t \rightarrow_{v} t'$, *then* $t \mathrel{\mathscr{N}} t'$.

Proof. The relation $\{(t, t') \mid t \rightarrow_{\mathrm{v}} t'\}$ is a normal-form bisimulation up to refl. □

Proposition 6.27 ($\mathcal{S}_{\langle\cdot\rangle}$ axiom)**.** *We have* $\mathcal{S}k.\langle t\rangle \mathrel{\mathscr{N}} \mathcal{S}k.t$.

Proof. These terms are stuck, so we have to show that $\langle\langle t\rangle\rangle \mathrel{\mathscr{N}} \langle t\rangle$ (proved in Example 6.2) and $\square \mathrel{\mathscr{N}^{\mathsf{c}}} \square$ (but $\mathscr{N}$ is reflexive). □

Proposition 6.28 ($\langle\cdot\rangle_{\mathtt{lift}}$ axiom)**.** *We have* $\langle(\lambda x.t_0)\, \langle t_1\rangle\rangle \mathrel{\mathscr{N}} (\lambda x.\langle t_0\rangle)\, \langle t_1\rangle$.

Proof. If $\mathcal{R} \stackrel{\mathsf{def}}{=} \{(\langle(\lambda x.t_0)\, y\rangle, (\lambda x.\langle t_0\rangle)\, y) \mid y \text{ fresh}\}$, then $\langle(\lambda x.t_0)\, \langle t_1\rangle\rangle \mathrel{\mathsf{ectxpure} \circ (\mathsf{id} \cup \mathsf{refl})(\mathcal{R})} (\lambda x.\langle t_0\rangle)\, \langle t_1\rangle$, and $\mathcal{R}$ is a bisimulation up to red and refl, since the two terms in $\mathcal{R}$ reduces to $\langle t_0\{y/x\}\rangle$. □

Proposition 6.29 (β_Ω axiom)**.** *If* $x \notin \mathsf{fv}(E)$, *then* $(\lambda x.E[x])\, t \mathrel{\mathscr{N}} E[t]$.

Proof. If $\mathcal{R} \stackrel{\mathsf{def}}{=} \{((\lambda x.E[x])\, y, E[y]) \mid y \text{ fresh}\}$, then $(\lambda x.E[x])\, t \mathrel{\mathsf{pctx} \circ (\mathsf{id} \cup \mathsf{refl})(\mathcal{R})} E[t]$, and $\mathcal{R}$ is a bisimulation up to red and refl, since the two terms in $\mathcal{R}$ reduces to $E[y]$. □

Proposition 6.30 ($\mathcal{S}_{\mathtt{elim}}$ axiom)**.** *If* $k \notin \mathsf{fv}(t)$, *then* $t \mathrel{\mathscr{M}} \mathcal{S}k.k\, t$.

Proof. We must relate $\langle c[\mathcal{S}k.k\, t]\rangle$ and $\langle c[t]\rangle$ for a fresh c, but $\langle c[\mathcal{S}k.k\, t]\rangle \rightarrow_{\mathrm{v}} \langle(\lambda x.\langle c[x]\rangle)\, t\rangle$, but we know that $\langle y\, t\rangle \mathrel{\mathscr{N}} \langle(\lambda x.\langle y\, x\rangle)\, t\rangle$ holds for all y (Example 6.12), so we can conclude with $[\![\cdot]\!]^f$ and Proposition 6.22. □

Consequently, $\mathscr{M}$ is complete w.r.t. $\equiv$, and $\equiv$ can be used as a proof technique for $\mathscr{M}$.

6.6. **Conclusion.** We propose several normal-form bisimilarities for the two semantics of $\lambda_{\mathcal{S}}$. For the relaxed semantics, we define normal-form and refined bisimilarities which differ in how they handle control-stuck terms; the former is easier to use but relates less contextually equivalent terms than the latter. Refined bisimilarity is defined on an extended calculus, where context variables represent unknown delimited contexts, the same way regular variables stand for unknown values. We follow the same idea for the original semantics, where the bisimilarity is defined on pure terms, and extended to any terms thanks to context variables.

Normal-form bisimulation is already tractable enough that we can prove complex equivalences with its plain definition (see Example 6.3). Proofs can be further simplified thanks to up-to techniques. Bisimulation up to related contexts is simpler to use than with environmental bisimilarity, as any term can be plugged in any context. As a result, the equivalence proof for Turing's combinator is simpler with normal-form than with environmental bisimilarity (compare Example 6.11 and Example 5.16). The downside of normal-form bisimilarity is that it is not complete w.r.t. contextual equivalence, and fails to relate terms that can be trivially related with applicative bisimilarity, as witnessed by the terms of Proposition 6.8.

7. Extensions

In this section, we discuss how our results are affected if we consider other semantics for $\lambda_{\mathcal{S}}$, or if we study other delimited-control operators, giving directions for future work in the process.

7.1. **Local Reduction Rules.** In the semantics of Section 2, contexts are captured in one reduction step. Another usual way of computing capture is to use local reduction rules, where the context is consumed piece by piece [26]. Formally, we introduce *elementary contexts*, defined as follows:

Elementary contexts: $G ::= v\,\square \mid \square\,t$

The reduction rule (*shift*) is then replaced with the next two rules.

$$F[G[\mathcal{S}k.t]] \rightarrow_{\mathrm{v}} F[\mathcal{S}k'.t\{\lambda x.\langle k'\; G[x]\rangle/k\}] \text{ with } x, k' \notin \mathsf{fv}(G) \cup \mathsf{fv}(t) \quad (\mathit{shift}_G)$$

$$F[\langle \mathcal{S}k.t\rangle] \rightarrow_{\mathrm{v}} F[\langle t\{\lambda x.x/k\}\rangle] \quad (\mathit{shift}_I)$$

As we can see in rule (shift_G), the capture of an elementary context does not require a reset, and it leaves the operator shift in place to continue the capture process. The process stops when a reset is encountered, in which case the rule (shift_I) applies: the shift operator is removed, and its variable k is replaced with the function representing the delimited empty context.

With local reduction rules, control stuck terms are of the form $\mathcal{S}k.t$ (without any surrounding context). This has major consequences on the definition of normal-form bisimulations, as it brings the regular definition (Definition 6.1) and the refined one (Definition 6.15) closer together, e.g., the terms of Proposition 6.13 can be proved contextually equivalent with the regular definition (when phrased in terms of the local reduction rules).

The resulting bisimulation proofs are arguably more difficult than with the semantics of Section 2, as we can see with the next example.

Example 7.1 (β_Ω axiom)**.** Assume we want to prove that $(\lambda x.E[x])\,t\;\mathscr{N}\;E[t]$ ($x \notin \mathsf{fv}(E)$) with local rules. If t is a control stuck term $\mathcal{S}k.t'$, we have to relate $\langle t'\{\lambda y.\langle k'\ (\lambda x.E[x])\ y\rangle/k\}\rangle$ (y, k' fresh) with $\langle t'\overrightarrow{\sigma}\rangle$, where $\overrightarrow{\sigma}$ are the substitutions we obtain as a result of the progressive capture of E by $\mathcal{S}k.t'$. We do not need sequences of substitutions with the semantics of Section 2.

The theory for applicative and environmental bisimulations is not affected by using local rules; in particular, we still have to compare control-stuck terms by putting them in a pure (multi-hole) context. However, a proof using a small-step bisimulation of any kind becomes tedious with local rules, as they introduce a lot of redexes (first to capture a whole pure context, and then to reduce all the produced β-redexes), and a reduction of each redex has to be matched in a small-step relation. We, therefore, believe that the reduction rules of Section 2 are better suited to proving the equivalence of two $\lambda_{\mathcal{S}}$ terms.

7.2. **Call-by-Name Reduction Semantics.** In call-by-name, arguments are not reduced to values before β-reduction takes place. Such a semantics can be achieved by changing the syntax of (pure) evaluation contexts as follows:

CBN pure contexts: $E ::= \Box \mid E\ t$

CBN evaluation contexts: $F ::= \Box \mid F\ t \mid \langle F\rangle$

and by turning the β-reduction rule into

$$F[(\lambda x.t_0)\,t_1] \quad \to_{\mathrm{n}} \quad F[t_0\{t_1/x\}] \qquad (\beta_n)$$

The rules (*shift*) and (*reset*) are the same as in call-by-value, but their meanings change because of the new syntax for call-by-name contexts. We still distinguish the relaxed semantics (without outermost enclosing reset) from the original semantics.

The results of this paper can be adapted to call-by-name by transforming values used as arguments into arbitrary terms, for example when comparing λ-abstractions with applicative bisimilarity, or when building testing terms from the environment in environmental bisimilarity. We can also relate the bisimilarities to the call-by-name CPS equivalence, which has been axiomatized by Kameyama and Tanaka [40]. The axioms for call-by-name are the same or simpler than in call-by-value: the axioms $\langle\cdot\rangle_{\mathcal{S}}$, $\langle\cdot\rangle_{\mathtt{val}}$, and $\mathcal{S}_{\langle\cdot\rangle}$ can be proved in call-by-name using bisimulations with the same proofs as in call-by-value. The call-by-value axioms $\beta_{\mathtt{v}}$, β_Ω, and $\langle\cdot\rangle_{\mathtt{lift}}$ are replaced by a single axiom for call-by-name β-reduction

$$(\lambda x.t_0)\,t_1 =_{\mathrm{KT}} t_0\{t_1/x\},$$

which is straightforward to prove since the three bisimilarities contain reduction. Finally, the axiom $\mathcal{S}_{\mathtt{elim}}$ still holds only for the original semantics.

7.3. **CPS-based Equivalences.** It is possible to go beyond CPS equivalence and use the CPS definition of shift and reset to define behavioral equivalences in terms of it: t_0 and t_1 are bisimilar in $\lambda_{\mathcal{S}}$ if their translations $\overline{t_0}$ and $\overline{t_1}$ are bisimilar in the plain λ-calculus. As an example, we can define CPS applicative bisimilarity $\mathscr{A}_{\mathsf{CPS}}$ as follows: given two closed terms t_0 and t_1 of $\lambda_{\mathcal{S}}$, we have $t_0\ \mathscr{A}_{\mathsf{CPS}}\ t_1$ if $\overline{t_0}$ and $\overline{t_1}$ are applicative bisimilar in the call-by-value λ-calculus [1]. We compare here this equivalence to the contextual equivalence $\mathscr{P}$ for the original semantics, since the CPS of Figure 1 is valid for that semantics only.

Even if $\mathscr{A}_{\mathsf{CPS}}$ is sound w.r.t. $\mathscr{P}$, we show it is not complete. A CPS translated term is of the form $\lambda k_1 k_2.t$, where k_1 and k_2 stand for, respectively, the continuation and the metacontinuation of the term, which are λ-abstractions of a special shape. But applicative bisimilarity in λ-calculus compares terms with any λ-abstraction, not just a continuation or metacontinuation, making $\mathscr{A}_{\mathsf{CPS}}$ over-discriminating compared to $\mathscr{P}$. Indeed, let $v_0 \stackrel{\mathsf{def}}{=} \lambda x.\Omega$ and $v_1 \stackrel{\mathsf{def}}{=} \lambda x.\langle x\ i\rangle\,\Omega$. We have $v_0\ \mathscr{P}\ v_1$, roughly because v_0 diverges as soon as it is applied to a value v, and so does v_1, either because $\langle v\ i\rangle$ diverges, or because $\langle v\ i\rangle$ converges and Ω then diverges (more formally, the relation $\{(\lambda x.\Omega, \lambda x.\langle x\ i\rangle\,\Omega)\} \cup \{(\Omega, \langle t\rangle\,\Omega) \mid t \in \mathcal{T}_c\} \cup \{(\Omega, v\,\Omega) \mid v \in \mathcal{V}_c\}$ is an applicative bisimulation, included in $\mathscr{C}$ and, therefore, in $\mathscr{P}$). The CPS translation of these terms, after some administrative reductions, yields

$$\overline{v_0} = \lambda k_1 k_2.k_1\ (\lambda x.\overline{\Omega})\ k_2, \text{ and}$$
$$\overline{v_1} = \lambda k_1 k_2.k_1\ (\lambda x k_1' k_2'.x\ (\lambda y.\overline{y})\ \gamma\ (\lambda z.\overline{\Omega}\ v'\ k_2))\ k_2$$

where γ is defined in Figure 1 and v' is some value, the precise definition of which is not important. If $v \stackrel{\mathsf{def}}{=} \lambda z k_2.z\ (\lambda x' k_1'' k_2''.i)\ i\ i$, then $\overline{v_0}\ v\ i \rightarrow_{\mathrm{v}}^* \overline{\Omega}$ and $\overline{v_1}\ v\ i \rightarrow_{\mathrm{v}}^* i$; the diverging part in $\overline{v_1}$, namely $\lambda z.\overline{\Omega}\,v'\,k_2$, is thrown away by v instead of being eventually applied, as it should be if v was a continuation (the term $\lambda x' k_1'' k_2''.i$ is not in the 2-layer CPS).

A possible way to get completeness for $\mathscr{A}_{\mathsf{CPS}}$ could be to restrict the target language of the CPS translation to a CPS calculus, i.e., a subcalculus where the grammar of terms enforces the correct shape of arguments passed as values, continuations, or metacontinuations (as in, e.g., [39]). However, even with completeness, we believe it is more tractable to work in direct style with the relations we define in this paper, than on CPS translations of terms: as we can see with v_0 and v_1 above, translating even relatively simple source terms leads to voluminous terms in CPS. Besides, $\mathscr{A}_{\mathsf{CPS}}$ compares all translated terms with a continuation (which corresponds to a context E) and a metacontinuation (which corresponds to a metacontext F), while bisimilarities in direct style need at most a context E to compare stuck terms.

Nonetheless, we believe that studying fully the relationship between CPS-based behavioral equivalences and direct-style equivalences is an interesting future work. We would like to consider other CPS translations, including a CPS translation for the relaxed semantics [63], or the 1-layer CPS translation for the original semantics [21]. We would also like to know if it is possible to obtain a CPS-based soundness proof for normal-form bisimilarity, as in λ-calculus [53], to have a complete picture of the interactions between CPS and behavioral equivalences.

7.4. **The $\lambda\mu\widehat{\mathsf{tp}}$-Calculus.** The $\lambda\mu$-calculus [68] contains a μ-construct that can be seen as an abortive control operator. In this calculus, we evaluate named terms of the form $[\alpha]t$, and the names α are used as placeholders for evaluation contexts. Roughly, a μ term $\mu\alpha.[\beta]t$ is able to capture its whole (named) evaluation context $[\gamma]E$, and substitutes α with $[\gamma]E$ in $[\beta]t$. Context substitution is the same as the one presented in Section 6.3. In particular, it is capture-free, e.g., in $[\beta]t\{[\gamma]E/\alpha\}$, the free names of $[\gamma]E$ (such as γ) cannot be bound by the μ constructs in t.

The $\lambda\mu\widehat{\mathsf{tp}}$-calculus [33] extends the $\lambda\mu$-calculus by adding a special name $\widehat{\mathsf{tp}}$ which can be dynamically bound during a context substitution. Besides, the μ-operator no longer captures the whole context, but only up to the nearest enclosing μ-binding of $\widehat{\mathsf{tp}}$. As a result, a μ-binding of $\widehat{\mathsf{tp}}$ can be seen as a delimiter, and in fact, the $\lambda\mu\widehat{\mathsf{tp}}$-calculus simulates $\lambda_{\mathcal{S}}$ [33]. In particular, their CPS equivalences coincide. However, defining bisimilarities in $\lambda\mu\widehat{\mathsf{tp}}$ may lead to relations similar to the $\lambda\mu$-calculus ones [16] because of names. Indeed, we have to compare named values $[\alpha]v$ in $\lambda\mu\widehat{\mathsf{tp}}$, which requires substituting α with some named context, as in the $\lambda\mu$-calculus [16]. Similarly, control stuck terms are of the form $[\alpha]F[\mu\widehat{\mathsf{tp}}.[\beta]v]$, and a way to relate them would be by replacing β with a context $[\gamma]E$ or $[\widehat{\mathsf{tp}}]E$. It would be interesting to compare the behavioral theories of $\lambda_{\mathcal{S}}$ and $\lambda\mu\widehat{\mathsf{tp}}$ to see if the encoding of the former into the latter is fully abstract (i.e., preserves contextual equivalence).

7.5. **Typed Setting.** A type system affects the semantics of a language by ruling out ill-typed terms, and thus restricts the possible behaviors compared to the untyped calculus. Applicative [30, 31], normal-form [56, 57], and environmental [93, 90] bisimilarities have been defined for various calculi and type systems. The type systems for shift and reset [20, 5] assign types not only to terms, but also to contexts. Pure contexts E are given types of the form $A \rhd B$, where A is the type of the hole and B is the answer type, and evaluation contexts (also called metacontexts) F are assigned types of the form $\neg A$, where A is the type of the hole. A typing judgment $\Gamma \mid B \ \vdash \ t \ : \ A \mid C$ roughly means that under the typing context Γ, the term t can be plugged into a pure context E of type $A \rhd B$ and a metacontext F of type $\neg C$, producing a well-typed term $F[\langle E[t]\rangle]$. In general, the evaluation of t may capture the surrounding context of type $A \rhd B$ to produce a value of type C, with $B \neq C$. Function types also contain extra information about the contexts the terms are plugged into: a term of type $A\ {}_C\!\to_D\ B$ can be applied to an argument of type A within a pure context of type $B \rhd C$ and a metacontext of type $\neg D$.

The complexity of the type systems for shift and reset (compared to, e.g., plain λ-calculus) may have some consequences on the definition of a typed bisimilarity for the language. In particular, we wonder how the extra type annotations for pure contexts and metacontexts should be factored in the bisimilarities. It seems natural to include types for the pure contexts for control stuck terms, since pure contexts already occur in the definitions of applicative and environmental bisimilarities in that case; it is not clear if and how the types for the metacontexts should be mentioned. The study of a typed $\lambda_{\mathcal{S}}$ can be interesting also to see how the types modify the equivalences between terms. We leave this as a future work.

A related and unexplored topic is defining logical relations to characterize contextual equivalence for typed calculi with delimited continuations. So far, Asai introduced logical relation to prove the correctness of a partial evaluator for shift and reset [4], whereas Biernacka et al. in a series of articles proposed logical predicates for proving termination of evaluation in several calculi of delimited control [6, 8, 9]. We expect such logical relations to exploit the notion of context and metacontext and, therefore, to be biorthogonal [49, 70]. Biorthogonal and step-indexed Kripke logical relations have been proposed for an ML-like language with $\mathsf{call/cc}$ by Dreyer et al. [23] and adapting this approach to a similar language based on Asai and Kameyama's polymorphic type system for shift and reset [5] presents itself as an interesting topic of future research. An alternative to step-indexed Kripke logical relations that also have been shown to account for abortive continuations are parametric bisimulations [37], built on relation transition systems of Hur et al. [36]. Whether such hybrids of logical relations and bisimulations can effectively support reasoning about delimited continuations is an open question.

7.6. **Other Delimited-Control Operators.**

CPS hierarchy. The operators shift and reset are just an instance of a more general construct called the *CPS hierarchy* [21]. As explained in Section 2.4, shift and reset have been originally defined by a translation into CPS. When iterated, the CPS translation leads to a hierarchy of continuations, in which it is possible to define a hierarchy of control operators shift_i and reset_i ($i \geq 1$) that generalizes shift and reset, and that makes possible to separate computational effects that should exist independently in a program. For example, in order to collect the solutions found by a backtracking algorithm implemented with shift_1 and reset_1, one has to employ shift_2 and reset_2, so that there is no interference between searching and emitting the results of the search. The CPS hierarchy was also envisaged to account for nested computations in hierarchical structures [7].

In the hierarchy, a shift operator of level i captures the context up to the first enclosing reset_i or reset_j with $j > i$. So for example in $\langle E_1[\langle E_0[\mathcal{S}_1 k.t]\rangle_2]\rangle_1$, the $\mathcal{S}_1$ captures only E_0, not E_1. We believe the results of this paper generalize to the CPS hierarchy without issues. The notions of pure context and control stuck term now depend on the hierarchy level: a pure context of level i does not contain a reset_j (for $j \geq i$) encompassing its hole, and can be captured by an operator shift_i. A control stuck term of level i is an operator shift_i in a pure context of level i. The definitions of bisimulations have to be generalized to deal with control stuck terms of level i the same way we treat stuck terms of level 1. For example, two control stuck terms of level i are applicative bisimilar if they are bisimilar when put in an arbitrary level i pure context surrounded by a reset_i. The proofs for $i = 1$ should carry through to any i.

Operator shift_0*.* The operator shift_0 ($\mathcal{S}_0$) allows a term to capture a pure context with its enclosing delimiter [21]. The capture reduction rule for this operator is thus as follows:

$$F[\langle E[\mathcal{S}_0 k.t]\rangle] \rightarrow_{\mathrm{v}} F[t\{\lambda x.\langle E[x]\rangle/k\}], \text{ with } x \notin \mathsf{fv}(E)$$

Note that there is no reset around t in $F[t\{\lambda x.\langle E[x]\rangle/k\}]$. Consequently, a term is able to directly decompose an evaluation context F into pure contexts through successive captures with $\mathcal{S}_0$; this is not possible in $\lambda_{\mathcal{S}}$.

The definitions of bisimilarities of this paper should extend to a calculus with shift_0 as far as the relaxed semantics is concerned. Since a term is able to access the context beyond the first enclosing reset, contextual equivalence is more discriminating with shift_0 than in $\lambda_{\mathcal{S}}$. For example, $\langle\langle t\rangle\rangle$ is no longer equivalent to $\langle t\rangle$, as we can see by taking $t = \mathcal{S}_0 k.\mathcal{S}_0 k.\Omega$.

For the original semantics (that in the case of shift_0 assumes a persistent top-level reset), the definitions have to take into account the fact that a delimited term $\langle t\rangle$ may evaluate to a control stuck term (like, e.g., $\langle \mathcal{S}_0 k.\mathcal{S}_0 k.t\rangle$ for any t) and that, therefore, it is not sufficient to compare values with values only. For instance, in order to validate the following equation taken from the axiomatization of shift_0 [62]:

$$\mathcal{S}_0 k.\langle(\lambda x.\mathcal{S}_0 k'.k\, x)\, t\rangle =_{\mathrm{M}} t, \text{ with } k \notin \mathsf{fv}(t)$$

we would have to be able to compare normal forms of different kinds, which can be achieved by putting the normal forms in a context $\langle E\rangle$ for any E.

Operators control *and* prompt. The control operator ($\mathcal{F}$) captures a pure context up to the first enclosing prompt ($\#$), but the captured context does not include the delimiter [26]. Formally, the capture reduction rule is as follows:

$$F[\#E[\mathcal{F}k.t]] \rightarrow_{\mathrm{v}} F[\#t\{\lambda x.E[x]/k\}], \text{ with } x \notin \mathsf{fv}(E)$$

Unlike with shift and reset where continuation composition is static, with control and prompt it is dynamic, in the sense that the extent of control operations in the captured context comprises the context of the resumption of the captured context [12]. A $\mathsf{control}_0$ variant also exists [85], where the delimiter is captured with the context but not kept: as a result, no delimiter is present in the right-hand side of the capture reduction rule.

The theory of this paper should extend to control and prompt with minor changes. However, studying this calculus would still be interesting to pinpoint the differences between the equivalences of shift/reset and control/prompt. For example, $\#\#t$ is equivalent to $\#t$, the same way $\langle\langle t\rangle\rangle$ is equivalent to $\langle t\rangle$. In fact, we conjecture the axioms can still be proved equivalent if we replace shift and reset with control and prompt (with the same restriction for $\mathcal{S}_{\mathtt{elim}}$). In contrast, $t_0 \stackrel{\mathsf{def}}{=} (\mathcal{S}k_1.k_1\ (\lambda x.\mathcal{S}k_2.t)\ \Omega)\ v$ (where $k_1, k_2, x \notin \mathsf{fv}(t)$) is equivalent to $\mathcal{S}k.\Omega$ (because $t_0 \xrightarrow{E}\rightarrow_{\mathrm{v}}^* \langle\langle E[\mathcal{S}k_2.t]\rangle\ \Omega\rangle$, and this term always diverges), but the term $t_0' \stackrel{\mathsf{def}}{=} (\mathcal{F}k_1.k_1\ (\lambda x.\mathcal{F}k_2.t)\ \Omega)\ v$ is equivalent to $\#t$ (because $t_0' \xrightarrow{E}\rightarrow_{\mathrm{v}}^* \#E[\mathcal{S}k_2.t]\ \Omega \rightarrow_{\mathrm{v}} \#t$). Maybe we can find (general enough) laws which hold with control and prompt but not with shift and reset, and conversely.

Multiple prompts. In languages with (named) multiple prompts [32, 25, 22] control delimiters (prompts) as well as control operators are tagged with names, so that the control operator captures the evaluation context up to the dynamically nearest delimiter with the matching name. In a calculus with tagged shift ($\mathcal{S}_a$) and reset ($\langle\cdot\rangle_a$) the operational semantics of shift is given by the following rule:

$$F'[\langle F[\mathcal{S}_a k.t]\rangle_a] \rightarrow_{\mathrm{v}} F'[\langle t\{\lambda x.\langle F[x]\rangle_a/k\}\rangle_a], \text{ with } a \notin \#(F) \text{ and } x \notin \mathsf{fv}(E)$$

where $\#(F)$ is the set of the prompts guarding the hole of F. Such calculi resemble the CPS hierarchy, already considered in this section, however there are differences in their semantics. In contrast to the CPS hierarchy, where evaluation contexts form a hierarchy[6] and the extent of control operations of level i is limited by control delimiters of any level $j \geq i$, in the calculus with multiple prompts the evaluation context is a list of the standard CBV evaluation contexts separated by named prompts and the control operations reach across any prompts up to a matching one. Moreover, the salient and unique feature of such calculi is dynamic name generation that allows one, e.g., to eliminate unwanted interactions between the control operations used to implement some control structure (e.g., coroutines) and the control operations of the code that uses the control structure.

Even without dynamic name generation, which gives an additional expressive power to such calculi, calculi with multiple prompts generalize, e.g., simple exceptions [32] and the catch/throw constructs [18]. The results of this article can be seamlessly adapted to these calculi and most, if not all, of the presented techniques should carry over without surprises.

[6] In the original semantics, the evaluation context of level $i+1$ is a list (a stack, really) of evaluation contexts of level i separated by control delimiters of level i (contexts of level 1 are just the standard CBV evaluation contexts.) and the number of context layers is fixed [7].

However, when dynamic name generation is included in the calculus, comparing two terms becomes more difficult, as prompts with the same purpose can be generated with different names. With environmental bisimilarity, we can use environments to remember the relationships between generated prompts. We do so in [3] and define sound and complete environmental bisimilarities and their up-to techniques for a calculus with dynamically generated prompt names. Resource generation makes the definition of a sound applicative bisimilarity difficult for such a calculus, as argued in [46].

7.7. **Other Constructs.** Here, we briefly discuss what happens when $\lambda_{\mathcal{S}}$ is extended with constructs that can be found in usual programming languages.

Constants. While adding constants (such as numerals, booleans, ...) to the language does not raise any issue for applicative [30] and environmental bisimilarities, defining a satisfactory normal-form bisimilarity in the presence of constants raises some difficulties [88]: e.g., it is not clear how to define a normal-form bisimulation which equates $x + y$ and $y + x$. Relying on encodings of constants into plain λ-calculus is not enough, as these encodings usually do not respect the properties of the constants, like, for example, commutativity of $+$. These problems are orthogonal to the presence of control operators though.

Store. Bisimilarities for languages with store are usually of the environmental kind [81, 48, 90], and [46] argues that the usual form of applicative bisimilarity is not sound in the presence of store. Støvring and Lassen define a sound and complete normal-form bisimilarity for $\lambda\mu\rho$ [88], a calculus with store and an abortive control construct inspired by Parigot's $\lambda\mu$ [68]. Their work largely relies on the fact that in $\lambda\mu$-calculus (and in $\lambda\mu\rho$ as well), terms are of the form $[a]t$, where the name a acts as a placeholder for an evaluation context. These names are also essential to be able to define a sound and complete applicative bisimilarity for $\lambda\mu$ [16]. Developing a sound behavioral theory of $\lambda_{\mathcal{S}}$ extended with higher-order store, potentially taking advantage of context variables, is of interest as a future work.

Exceptions. Like for store, Koutavas et al. [46] give examples showing that applicative bisimilarity is not sound for a calculus with exceptions, and environmental bisimilarity should instead be used. Studying an extension of $\lambda_{\mathcal{S}}$ with exceptions would be interesting to compare the encoding of exceptions using shift and reset [27] to the native constructs. We leave this as a future work.

8. Conclusion

In our study of the behavioral theory of a calculus with shift and reset, we consider two semantics: the original one, where terms are executed within an outermost reset, and the relaxed one, where this requirement is lifted. For each, we define a contextual equivalence (respectively $\mathscr{P}$ and $\mathscr{C}$), that we try to characterize with different kinds of bisimilarities (normal-form $\mathscr{N}$, $\mathscr{R}$, $\mathscr{M}$, applicative $\mathscr{A}$, and environmental $\mathscr{E}$, $\mathscr{F}$). We also compare our relations to CPS equivalence $\equiv$, a relation which equates terms with $\beta\eta$-equivalent CPS translations. We summarize in Figure 11 the relationships between these relations.

Normal-form bisimulation arguably leads to the simplest equivalence proofs in most cases; by essence, the lack of quantifications on testing entities in its definition leads to simpler proof

$$\begin{array}{rl} \text{relaxed semantics:} & \mathscr{N} \subsetneq \mathscr{R} \subsetneq \mathscr{C} = \mathscr{A} = \mathscr{E} \\ & \quad\;\; \rotatebox{90}{$\subsetneq$} \quad\;\; \rotatebox{90}{$\subsetneq$} \\ \text{original semantics:} & \equiv \subsetneq \mathscr{M} \subsetneq \mathscr{P} = \mathscr{F} \end{array}$$

Figure 11: Relationships between the equivalences of $\lambda_{\mathcal{S}}$

obligations. More importantly, it benefits from up-to techniques which manipulate contexts, which is of prime importance in a calculus where context capture is part of the semantics. Environmental bisimilarity also allows for such context manipulating up-to techniques, albeit in a less general form. As a result, proving that Turing's fixed-point combinator is bisimilar to its shift/reset variant can be done using up-to techniques for normal-form bisimilarity with only three pairs (Example 6.11), while it requires an inductively defined candidate relation with environmental bisimulation (Example 5.16). The lack of such a powerful up-to technique for applicative bisimulation makes it clearly less tractable than the other styles, as witnessed by the proofs for Turing's combinator (Example 4.9), or for the β_Ω axiom (Proposition 4.23 vs Propositions 6.29 and 5.21).

However, normal-form bisimulation cannot be used to prove all equivalences, since its corresponding bisimilarity is not complete. It can be too discriminating to relate very simple terms, like those in Propositions 6.13 and 6.8, even though refined normal-form bisimulation (Section 6.3) can help. In contrast, applicative and environmental bisimilarities are complete, and can be used as alternatives when normal-form bisimulation fails.

To summarize, to prove that two given terms are equivalent, we would suggest to first try normal-form bisimulation, and if it fails, try next environmental bisimulation. Applicative bisimulation should be used only in the simplest cases, such as terms similar to those of Proposition 4.19. The relations for the relaxed semantics can also be used as proof techniques for the original semantics, except in cases similar to the $\mathcal{S}_{\mathtt{elim}}$ axiom, where only the equivalences dedicated to the original semantics can be used.

References

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

[2] Z. M. Ariola, H. Herbelin, and A. Sabry. A proof-theoretic foundation of abortive continuations. *Higher-Order and Symbolic Computation*, 20(4):403–429, 2007.

[3] A. Aristizábal, D. Biernacki, S. Lenglet, and P. Polesiuk. Environmental Bisimulations for Delimited-Control Operators with Dynamic Prompt Generation. *Logical Methods in Computer Science*, 13(3), 2017.

[4] K. Asai. Logical relations for call-by-value delimited continuations. In M. van Eekelen, editor, *Proceedings of the Sixth Symposium on Trends in Functional Programming (TFP 2005)*, pages 413–428, Tallinn, Estonia, Sept. 2005. Institute of Cybernetics at Tallinn Technical University. Extended version available as Technical Report of Department of Information Science, Ochanomizu University, OCHA-IS 06-1.

[5] K. Asai and Y. Kameyama. Polymorphic delimited continuations. In Shao [86], pages 239–254.

[6] M. Biernacka and D. Biernacki. Context-based proofs of termination for typed delimited-control operators. In F. J. López-Fraguas, editor, *PPDP'09*, Coimbra, Portugal, Sept. 2009. ACM Press.

[7] 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.

[8] M. Biernacka, D. Biernacki, and S. Lenglet. Typing control operators in the CPS hierarchy. In M. Hanus, editor, *Proceedings of the 13th ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming (PPDP'11)*, Odense, Denmark, July 2011. ACM Press.

[9] M. Biernacka, D. Biernacki, S. Lenglet, and M. Materzok. Proving termination of evaluation for system F with control operators. In U. de'Liguoro and A. Saurin, editors, *Proceedings of the 1st Workshop on Control Operators and their Semantics (COS 2013)*, volume 127 of *Electronic Proceedings in Theoretical Computer Science*, pages 15–27, Eindhoven, The Netherlands, June 2013.
[10] D. Biernacki and O. Danvy. A simple proof of a folklore theorem about delimited control. *Journal of Functional Programming*, 16(3):269–280, 2006.
[11] D. Biernacki, O. Danvy, and K. Millikin. A dynamic continuation-passing style for dynamic delimited continuations. *ACM Trans. Program. Lang. Syst.*, 38(1):2:1–2:25, 2015.
[12] D. Biernacki, O. Danvy, and C. Shan. On the static and dynamic extents of delimited continuations. *Science of Computer Programming*, 60(3):274–297, 2006.
[13] D. Biernacki and S. Lenglet. Applicative bisimulations for delimited-control operators. In L. Birkedal, editor, *FOSSACS'12*, number 7213 in LNCS, pages 119–134, Tallinn, Estonia, Mar. 2012. Springer-Verlag.
[14] D. Biernacki and S. Lenglet. Normal form bisimulations for delimited-control operators. In T. Schrijvers and P. Thiemann, editors, *FLOPS'12*, number 7294 in LNCS, pages 47–61, Kobe, Japan, May 2012. Springer-Verlag.
[15] D. Biernacki and S. Lenglet. Environmental bisimulations for delimited-control operators. In C. Shan, editor, *Programming Languages and Systems - 11th Asian Symposium, APLAS 2013, 2013. Proceedings*, volume 8301 of *Lecture Notes in Computer Science*, pages 333–348, Melbourne, VIC, Australia, Dec. 2013. Springer.
[16] D. Biernacki and S. Lenglet. Applicative bisimilarities for call-by-name and call-by-value $\lambda\mu$-calculus. In B. Jacobs, A. Silva, and S. Staton, editors, *Proceedings of the 30th Annual Conference on Mathematical Foundations of Programming Semantics(MFPS XXX)*, volume 308 of *Electronic Notes in Theoretical Computer Science*, pages 49–64, Ithaca, USA, June 2014.
[17] D. Biernacki, S. Lenglet, and P. Polesiuk. Proving soundness of extensional normal-form bisimilarities. In A. Silva, editor, *Proceedings of the 33th Annual Conference on Mathematical Foundations of Programming Semantics(MFPS XXXIII)*, volume 336 of *Electronic Notes in Theoretical Computer Science*, pages 41–56, Ljubljana, Slovenia, June 2017.
[18] T. Crolard. A confluent lambda-calculus with a catch/throw mechanism. *JFP*, 9(6):625–647, 1999.
[19] O. Danvy. Type-directed partial evaluation. In Steele Jr. [87], pages 242–257.
[20] 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.
[21] O. Danvy and A. Filinski. Abstracting control. In M. Wand, editor, *LFP'90*, pages 151–160, Nice, France, June 1990. ACM Press.
[22] P. Downen and Z. M. Ariola. A systematic approach to delimited control with multiple prompts. In H. Seidl, editor, *Programming Languages and Systems, 21st European Symposium on Programming, ESOP 2012*, Lecture Notes in Computer Science, pages 234–253, Tallinn, Estonia, Mar. 2012. Springer-Verlag.
[23] D. Dreyer, G. Neis, and L. Birkedal. The impact of higher-order state and control effects on local relational reasoning. *Journal of Functional Programming*, 22(4-5):477–528, 2012.
[24] P. Dybjer and A. Filinski. Normalization and partial evaluation. In G. Barthe, P. Dybjer, L. Pinto, and J. Saraiva, editors, *Applied Semantics – Advanced Lectures*, number 2395 in Lecture Notes in Computer Science, pages 137–192, Caminha, Portugal, Sept. 2000. Springer-Verlag.
[25] R. K. Dybvig, S. Peyton-Jones, and A. Sabry. A monadic framework for delimited continuations. *Journal of Functional Programming*, 17(6):687–730, 2007.
[26] M. Felleisen. The theory and practice of first-class prompts. In J. Ferrante and P. Mager, editors, *POPL'88*, pages 180–190, San Diego, California, Jan. 1988. ACM Press.
[27] A. Filinski. Representing monads. In H.-J. Boehm, editor, *POPL'94*, pages 446–457, Portland, Oregon, Jan. 1994. ACM Press.
[28] A. Filinski. Representing layered monads. In A. Aiken, editor, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Programming Languages*, pages 175–188, San Antonio, Texas, Jan. 1999. ACM Press.
[29] M. Flatt, G. Yu, R. B. Findler, and M. Felleisen. Adding delimited and composable control to a production programming environment. In N. Ramsey, editor, *Proceedings of the 2007 ACM SIGPLAN International Conference on Functional Programming (ICFP'07)*, SIGPLAN Notices, Vol. 42, No. 9, pages 165–176, Freiburg, Germany, Sept. 2007. ACM Press.

[30] A. D. Gordon. Bisimilarity as a theory of functional programming. *Theoretical Computer Science*, 228(1-2):5–47, 1999.

[31] A. D. Gordon and G. D. Rees. Bisimilarity for a first-order calculus of objects with subtyping. In Steele Jr. [87], pages 386–395.

[32] C. Gunter, D. Rémy, and J. G. Riecke. A generalization of exceptions and control in ML-like languages. In S. Peyton Jones, editor, *Proceedings of the Seventh ACM Conference on Functional Programming and Computer Architecture*, pages 12–23, La Jolla, California, June 1995. ACM Press.

[33] H. Herbelin and S. Ghilezan. An approach to call-by-name delimited continuations. In P. Wadler, editor, *Proceedings of the Thirty-Fifth Annual ACM Symposium on Principles of Programming Languages*, pages 383–394. ACM Press, Jan. 2008.

[34] R. Hieb, R. K. Dybvig, and C. W. Anderson, III. Subcontinuations. *Lisp and Symbolic Computation*, 5(4):295–326, Dec. 1993.

[35] D. J. Howe. Proving congruence of bisimulation in functional programming languages. *Information and Computation*, 124(2):103–112, 1996.

[36] C. Hur, D. Dreyer, G. Neis, and V. Vafeiadis. The marriage of bisimulations and Kripke logical relations. In J. Field and M. Hicks, editors, *Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012, Philadelphia, Pennsylvania, USA, January 22-28, 2012*, pages 59–72. ACM, 2012.

[37] C.-K. Hur, G. Neis, D. Dreyer, and V. Vafeiadis. A logical step forward in parametric bisimulations. Technical Report MPI-SWS-2014-003, Max Planck Institute for Software Systems (MPI-SWS), Saarbrcken, Germany, Jan. 2014.

[38] Y. Kameyama. Axioms for control operators in the CPS hierarchy. *Higher-Order and Symbolic Computation*, 20(4):339–369, 2007.

[39] 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.

[40] Y. Kameyama and A. Tanaka. Equational axiomatization of call-by-name delimited control. In T. Kutsia, W. Schreiner, and M. Fernández, editors, *Proceedings of the 12th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming,*, pages 77–86, Hagenberg, Austria, July 2010. ACM.

[41] O. Kiselyov. How to remove a dynamic prompt: Static and dynamic delimited continuation operators are equally expressible. Technical Report 611, Computer Science Department, Indiana University, Bloomington, Indiana, Mar. 2005.

[42] O. Kiselyov. Delimited control in OCaml, abstractly and concretely: System description. In M. Blume and G. Vidal, editors, *Functional and Logic Programming, 10th International Symposium, FLOPS 2010*, number 6009 in Lecture Notes in Computer Science, pages 304–320, Sendai, Japan, Apr. 2010. Springer.

[43] O. Kiselyov and C. Shan. Delimited continuations in operating systems. In B. Kokinov, D. C. Richardson, T. R. Roth-Berghofer, and L. Vieu, editors, *Modeling and Using Context, 6th International and Interdisciplinary Conference, CONTEXT 2007*, number 4635 in Lecture Notes in Artificial Intelligence, pages 291–302, Roskilde, Denmark, Aug. 2007. Springer.

[44] O. Kiselyov and C. Shan. Embedded probabilistic programming. In W. Taha, editor, *Domain-Specific Languages, DSL 2009*, number 5658 in Lecture Notes in Computer Science, pages 360–384, Oxford, UK, July 2009. Springer.

[45] O. Kiselyov, C. Shan, D. P. Friedman, and A. Sabry. Backtracking, interleaving, and terminating monad transformers. In B. Pierce, editor, *Proceedings of the 2005 ACM SIGPLAN International Conference on Functional Programming (ICFP'05)*, SIGPLAN Notices, Vol. 40, No. 9, pages 192–203, Tallinn, Estonia, Sept. 2005. ACM Press.

[46] V. Koutavas, P. B. Levy, and E. Sumii. From applicative to environmental bisimulation. In M. Mislove and J. Ouaknine, editors, *Proceedings of the 27th Annual Conference on Mathematical Foundations of Programming Semantics(MFPS XXVII)*, volume 276 of *ENTCS*, pages 215–235, Pittsburgh, PA, USA, May 2011.

[47] V. Koutavas and M. Wand. Bisimulations for untyped imperative objects. In P. Sestoft, editor, *ESOP'06*, volume 3924 of *Lecture Notes in Computer Science*, pages 146–161, Vienna, Austria, Mar. 2006. Springer.

[48] V. Koutavas and M. Wand. Small bisimulations for reasoning about higher-order imperative programs. In J. G. Morrisett and S. L. P. Jones, editors, *POPL'06*, pages 141–152, Charleston, SC, USA, Jan. 2006. ACM Press.

[49] J. Krivine. Classical logic, storage operators and second-order lambda-calculus. *Annals of Pure and Applied Logic*, 68(1):53–78, 1994.

[50] I. Lanese, J. A. Pérez, D. Sangiorgi, and A. Schmitt. On the expressiveness and decidability of higher-order process calculi. *Inf. Comput.*, 209(2):198–226, 2011.

[51] S. B. Lassen. Bisimulation for pure untyped $\lambda\mu$-caluclus (extended abstract). Unpublished note, Jan. 1999.

[52] 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.

[53] S. B. Lassen. Eager normal form bisimulation. In P. Panangaden, editor, *LICS'05*, pages 345–354, Chicago, IL, June 2005. IEEE Computer Society Press.

[54] 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.

[55] 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.

[56] S. B. Lassen and P. B. Levy. Typed normal form bisimulation. In J. Duparc and T. A. Henzinger, editors, *Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL Proceedings*, volume 4646 of *Lecture Notes in Computer Science*, pages 283–297, Lausanne, Switzerland, Sept. 2007. Springer.

[57] S. B. Lassen and P. B. Levy. Typed normal form bisimulation for parametric polymorphism. In F. Pfenning, editor, *LICS'08*, pages 341–352, Pittsburgh, Pennsylvania, June 2008. IEEE Computer Society Press.

[58] J. L. Lawall and O. Danvy. Continuation-based partial evaluation. In C. L. Talcott, editor, *Proceedings of the 1994 ACM Conference on Lisp and Functional Programming*, LISP Pointers, Vol. VII, No. 3, pages 227–238, Orlando, Florida, June 1994. ACM Press.

[59] S. Lenglet, A. Schmitt, and J.-B. Stefani. Howe's method for calculi with passivation. In M. Bravetti and G. Zavattaro, editors, *CONCUR'09*, number 5710 in LNCS, pages 448–462, Bologna, Italy, July 2009. Springer.

[60] J. Madiot, D. Pous, and D. Sangiorgi. Bisimulations up-to: Beyond first-order transition systems. In P. Baldan and D. Gorla, editors, *25th International Conference on Concurrency Theory*, volume 8704 of *Lecture Notes in Computer Science*, pages 93–108, Rome, Italy, Sept. 2014. Springer.

[61] J.-M. Madiot. *Higher-order languages: dualities and bisimulation enhancements*. PhD thesis, Université de Lyon and Università di Bologna, 2015.

[62] M. Materzok. Axiomatizing subtyped delimited continuations. In S. R. D. Rocca, editor, *Computer Science Logic 2013, CSL 2013*, volume 23 of *LIPIcs*, pages 521–539, Torino, Italy, Sept. 2013. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

[63] M. Materzok and D. Biernacki. Subtyping delimited continuations. In O. Danvy, editor, *Proceedings of the 2011 ACM SIGPLAN International Conference on Functional Programming (ICFP'11)*, pages 81–93, Tokyo, Japan, Sept. 2011. ACM Press.

[64] M. Merro. On the observational theory of the CPS-calculus. *Acta Informatica*, 47(2):111–132, 2010.

[65] R. Milner. Fully abstract models of typed λ-calculi. *Theoretical Computer Science*, 4(1):1–22, 1977.

[66] L. Moreau and C. Queinnec. Partial continuations as the difference of continuations, a duumvirate of control operators. In M. Hermenegildo and J. Penjam, editors, *Sixth International Symposium on Programming Language Implementation and Logic Programming*, number 844 in Lecture Notes in Computer Science, pages 182–197, Madrid, Spain, Sept. 1994. Springer-Verlag.

[67] J. H. Morris. *Lambda Calculus Models of Programming Languages*. PhD thesis, Massachusets Institute of Technology, 1968.

[68] M. Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In A. Voronkov, editor, *LPAR'92*, number 624 in LNAI, pages 190–201, St. Petersburg, Russia, July 1992. Springer-Verlag.

[69] A. Piérard and E. Sumii. A higher-order distributed calculus with name creation. In *LICS'12*, pages 531–540, Dubrovnik, Croatia, June 2012. IEEE Computer Society Press.

[70] A. Pitts and I. Stark. Operational reasoning for functions with local state. In A. Gordon and A. Pitts, editors, *Higher Order Operational Techniques in Semantics*, pages 227–273. Publications of the Newton Institute, Cambridge University Press, 1998.
[71] A. M. Pitts. Parametric polymorphism and operational equivalence. *Mathematical Structures in Computer Science*, 10(3):321–359, 2000.
[72] G. D. Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1:125–159, 1975.
[73] D. Pous. Complete lattices and up-to techniques. In Shao [86], pages 351–366.
[74] D. Pous. Coinduction all the way up. In M. Grohe, E. Koskinen, and N. Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16*, pages 307–316, New York, NY, USA, July 2016. ACM.
[75] D. Pous and D. Sangiorgi. Enhancements of the bisimulation proof method. In D. Sangiorgi and J. Rutten, editors, *Advanced Topics in Bisimulation and Coinduction*, chapter 6, pages 233–289. Cambridge University Press, 2011.
[76] T. Rompf, I. Maier, and M. Odersky. Implementing first-class polymorphic delimited continuations by a type-directed selective cps-transform. In A. Tolmach, editor, *Proceedings of the 2009 ACM SIGPLAN International Conference on Functional Programming (ICFP'09)*, pages 317–328, Edinburgh, UK, Aug. 2009. ACM Press.
[77] A. Sabry. Note on axiomatizing the semantics of control operators. Technical Report CIS-TR-96-03, Department of Computer and Information Science, University of Oregon, 1996.
[78] D. Sangiorgi. The lazy lambda calculus in a concurrency scenario. In A. Scedrov, editor, *LICS'92*, pages 102–109, Santa Cruz, California, June 1992. IEEE Computer Society.
[79] D. Sangiorgi. On the bisimulation proof method. *Mathematical Structures in Computer Science*, 8(5):447–479, Oct. 1998.
[80] 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.
[81] D. Sangiorgi, N. Kobayashi, and E. Sumii. Environmental bisimulations for higher-order languages. *ACM Transactions on Programming Languages and Systems*, 33(1):1–69, Jan. 2011.
[82] D. Sangiorgi and D. Walker. *The Pi-Calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.
[83] N. Sato and E. Sumii. The higher-order, call-by-value applied Pi-calculus. In Z. Hu, editor, *APLAS'09*, volume 5904 of *LNCS*, pages 311–326, Seoul, Korea, Dec. 2009. Springer-Verlag.
[84] C. Shan. Delimited continuations in natural language: quantification and polarity sensitivity. In H. Thielecke, editor, *Proceedings of the Fourth ACM SIGPLAN Workshop on Continuations (CW'04)*, Technical report CSR-04-1, Department of Computer Science, Queen Mary's College, pages 55–64, Venice, Italy, Jan. 2004.
[85] C. Shan. A static simulation of dynamic delimited control. *Higher-Order and Symbolic Computation*, 20(4):371–401, 2007.
[86] Z. Shao, editor. *Proceedings of the Fifth Asian Symposium on Programming Languages and Systems, APLAS'07*, number 4807 in LNCS, Singapore, Dec. 2007. Springer-Verlag.
[87] G. L. Steele Jr., editor. *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Programming Languages*, St. Petersburg Beach, Florida, Jan. 1996. ACM Press.
[88] 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, Nice, France, Jan. 2007. ACM Press.
[89] E. Sumii. An implementation of transparent migration on standard Scheme. In M. Felleisen, editor, *Proceedings of the Workshop on Scheme and Functional Programming*, Technical Report 00-368, Rice University, pages 61–64, Montréal, Canada, Sept. 2000.
[90] E. Sumii. A complete characterization of observational equivalence in polymorphic *lambda*-calculus with general references. In E. Grädel and R. Kahle, editors, *Computer Science Logic, 23rd international Workshop, CSL 2009, 18th Annual Conference of the EACSL, Proceedings*, volume 5771 of *Lecture Notes in Computer Science*, pages 455–469, Coimbra, Portugal, Sept. 2009. Springer.
[91] E. Sumii. A bisimulation-like proof method for contextual properties in untyped lambda-calculus with references and deallocation. *Theoretical Computer Science*, 411(51-52):4358–4378, 2010.

[92] E. Sumii and B. C. Pierce. A bisimulation for dynamic sealing. *Theoretical Computer Science*, 375(1-3):169–192, 2007.
[93] E. Sumii and B. C. Pierce. A bisimulation for type abstraction and recursion. *Journal of the ACM*, 54(5), 2007.
[94] H. Thielecke. *Categorical Structure of Continuation Passing Style*. PhD thesis, University of Edinburgh, Edinburgh, Scotland, 1997. ECS-LFCS-97-376.
[95] J. Tiuryn and M. Wand. Untyped lambda-calculus with input-output. In H. Kirchner, editor, *21st Colloquium on Trees in Algebra and Programming (CAAP'96)*, volume 1059 of *Lecture Notes in Computer Science*, pages 317–329, Linköping, Sweden, Apr. 1996. Springer-Verlag.
[96] M. Wand and D. Vaillancourt. Relating models of backtracking. In K. Fisher, editor, *Proceedings of the 2004 ACM SIGPLAN International Conference on Functional Programming (ICFP'04)*, SIGPLAN Notices, Vol. 39, No. 9, pages 54–65, Snowbird, Utah, Sept. 2004. ACM Press.
[97] T. Yachi and E. Sumii. A sound and complete bisimulation for contextual equivalence in \lambda -calculus with call/cc. In A. Igarashi, editor, *Programming Languages and Systems – 14th Asian Symposium, APLAS 2016, Hanoi, Vietnam, November 21-23, 2016, Proceedings*, volume 10017 of *Lecture Notes in Computer Science*, pages 171–186, 2016.

Appendix A. Proofs Sketches for Normal-Form Bisimilarity

We only sketch the progress proofs for the refined bisimilarity and for the original semantics as they are very similar to the proofs for the relaxed semantics [17]. We start with the proof sketches for the original semantics, which exhibits the most differences.

Lemma A.1. *For all* $f \in \mathsf{strong}(\mathfrak{F})$, $f \rightsquigarrow_{\mathsf{s}} \widehat{\mathsf{strong}(\mathfrak{F})}^{\omega}, \widehat{\mathfrak{F}}^{\omega}$.

Proof. Let $\mathcal{R} \rightarrowtail\!\!\!\rightarrow_{\mathsf{o}} \mathcal{S}, \mathcal{T}$. The refl case is straightforward.

Let $p_0 \mathrel{\mathsf{red}(\mathcal{R})} p_1$ with $p_0 \rightarrow_{\mathrm{v}}^* p_0'$, $p_1 \rightarrow_{\mathrm{v}}^* p_1'$, and $p_0' \mathrel{\mathcal{R}} p_1'$. If $p_0 = p_0'$ (in particular, if p_0 is a normal form), then any test on p_0 is matched by p_1: $p_1 \rightarrow_{\mathrm{v}}^* p_1'$ and $p_0 \mathrel{\mathcal{R}} p_1'$, so we can conclude with the progress hypothesis on $\mathcal{R}$. If $p_0 \neq p_0'$, then $p_0 \rightarrow_{\mathrm{v}} p_0''$ for some p_0''. Then $p_0'' \rightarrow_{\mathrm{v}}^* p_0'$, therefore we have $p_0'' \mathrel{\mathsf{red}(\mathcal{R})} p_1$, which implies $p_0'' \mathrel{\mathsf{red}(\mathcal{T})} p_1$ (because $\mathcal{R} \subseteq \mathcal{T}$ by definition of progress).

Let $c[\lambda x.p_0] \mathrel{\mathsf{lam}(\mathcal{R})} c[\lambda x.p_1]$ with $p_0 \mathrel{\mathcal{R}} p_1$. The terms are context-stuck, and we have $\langle \Box \rangle \mathrel{\mathsf{refl}(\mathcal{T})} \langle \Box \rangle$, and $c'[(\lambda x.p_0)\, y] \mathrel{\mathsf{red}(\mathsf{cvar}(\mathsf{subst}(\mathcal{T})))} c'[(\lambda x.p_1)\, y]$, for any fresh c' and y. The case $c[\mathcal{S}k.p_0] \mathrel{\mathsf{lam}(\mathcal{R})} c[\mathcal{S}k.p_1]$ with $p_0 \mathrel{\mathcal{R}} p_1$ is similar.

Let $c[p_0] \mathrel{\mathsf{cvar}(\mathcal{R})} c[p_1]$ with $p_0 \mathrel{\mathcal{R}} p_1$. Either $p_0 \rightarrow_{\mathrm{v}} p_0'$ and we progress to $\mathsf{cvar}(\mathcal{T})$, or p_0 is a normal form. Then $c[p_0]$ is also a normal form, and the result is easy to verify for each of them.

Let $p_0\{v_0/x\} \mathrel{\mathsf{subst}(\mathcal{R})} p_1\{v_1/x\}$ with $p_0 \mathrel{\mathcal{R}} p_1$ and $v_0 \mathrel{\mathcal{R}^{\mathsf{v}}} v_1$. The interesting case is when $p_0 = F_0[x\, w_0]$. Because p_0 is pure, in fact $F_0 = F_0'[\langle E_0 \rangle]$ for some F_0' and E_0. Therefore, $p_1 \Downarrow_{\mathrm{v}} F_1'[\langle E_1[x\, w_1] \rangle]$, with $F_0'[\langle E_0 \rangle] \mathrel{\mathcal{T}^{\mathsf{c}}} F_1'[\langle E_1 \rangle]$ and $w_0 \mathrel{\mathcal{T}^{\mathsf{v}}} w_1$. We have $c[v_0\, y] \mathrel{\mathcal{R}} c[v_1\, y]$ for fresh c and y; we distinguish two cases. If $c[v_0\, y] \rightarrow_{\mathrm{v}} p_0''$, then there exists p_1'' such that $c[v_1\, y] \rightarrow_{\mathrm{v}}^* p_1''$ and $p_0'' \mathrel{\mathcal{T}} p_1''$. Then $p_0\{v_0/x\} \rightarrow_{\mathrm{v}} F_0'\{v_0/x\}[p_0''\{w_0\{v_0/x\}/y\}\{\langle E_0\{v_0/x\}\rangle/c\}]$ and $p_1\{v_1/x\} \rightarrow_{\mathrm{v}}^* F_1'\{v_1/x\}[p_1''\{w_1\{v_1/x\}/y\}\{\langle E_1\{v_1/x\}\rangle/c\}]$; the resulting terms are in $\mathcal{T}$ up to csubst, subst, and $\mathsf{ectxpure}$. Otherwise, $c[v_0\, y]$ is an open-stuck term; then there exists p_1'' such that $c[v_1\, y] \Downarrow_{\mathrm{v}} p_1''$ and $c[v_0\, y] \mathrel{\mathcal{T}^{\mathsf{nf}}} p_1''$. We can conclude as in the first case. $\square$

Lemma A.2. $\mathsf{ectxpure} \rightsquigarrow \widehat{\mathsf{strong}(\mathfrak{F})}^{\omega} \circ \widehat{\mathfrak{F}} \circ \widehat{\mathsf{strong}(\mathfrak{F})}^{\omega}, \widehat{\mathfrak{F}}^{\omega}$

Sketch. Let $\mathcal{R} \rightarrowtail\!\!\!\rightarrow_{\mathsf{o}} \mathcal{R}, \mathcal{T}$. Let $F_0[p_0] \mathrel{\mathsf{ectxpure}(\mathcal{R})} F_1[p_1]$ with $p_0 \mathrel{\mathcal{R}} p_1$ and $F_0[x] \mathrel{\mathcal{R}} F_1[x]$ for a fresh x. The cases $p_0 \rightarrow_{\mathrm{v}} p_0'$, and p_0 is a context-stuck or an open-stuck term are easy to check. If $p_0 = v_0$, then there exists v_1 such that $p_1 \Downarrow_{\mathrm{v}} v_1$ and $v_0 \mathrel{\mathcal{R}^{\mathsf{v}}} v_1$. Then

$F_1[p_1] \rightarrow_{\text{v}}^{*} F_1[v_1]$, and $F_0[v_0] = F_0[x]\{v_0/x\}$ $\mathsf{subst}(\mathcal{R})$ $F_1[x]\{v_1/x\} = F_1[v_1]$, so we can conclude with Lemma A.1. $\square$

Lemma A.3. $\mathsf{csubst} \rightsquigarrow \widehat{\mathsf{strong}(\mathfrak{F})}^{\omega} \circ \widehat{\mathfrak{F}} \circ \widehat{\mathsf{strong}(\mathfrak{F})}^{\omega}, \widehat{\mathfrak{F}}^{\omega}$

Sketch. Let $\mathcal{R} \rightarrowtail\!\!\!\!\twoheadrightarrow_{\mathsf{o}} \mathcal{R}, \mathcal{T}$. Let $p_0\{D_0/c\}$ $\mathsf{csubst}(\mathcal{R})$ $p_1\{D_1/c\}$ with $p_0 \mathrel{\mathcal{R}} p_1$ and $D_0[x] \mathrel{\mathcal{R}} D_1[x]$ for a fresh x. The interesting case is when $p_0 = F_0[c[v_0]]$. There exists F_1, v_1 such that $p_1 \Downarrow_{\text{v}} F_1[c[v_1]]$, $F_0[y] \mathrel{\mathcal{T}} F_1[y]$ for a fresh y, and $v_0 \mathrel{\mathcal{R}^{\mathsf{v}}} v_1$. The possible reductions of $p_0\{D_0/c\}$ comes from $D_0[v_0]$, but we have $D_0[v_0]$ $\mathsf{subst}(\mathcal{R})$ $D_1[v_1]$. Suppose $D_0[v_0] \rightarrow_{\text{v}} p_0'$. By Lemma A.1, there exists p_1' such that $D_1[v_1] \rightarrow_{\text{v}} p_1'$ and $p_0' \mathrel{\widehat{\mathfrak{F}}^{\omega}(\mathcal{T})} p_1'$. Then $p_0\{D_0/c\} \rightarrow_{\text{v}} F_0[p_0']\{D_0/c\}$, $p_1\{D_1/c\} \rightarrow_{\text{v}}^{*} F_1[p_1']\{D_1/c\}$, and the resulting terms are in $\widehat{\mathfrak{F}}^{\omega}(\mathcal{T})$ up to $\mathsf{ectxpure}$ and csubst, i.e., in $\widehat{\widehat{\mathfrak{F}}}^{\omega}(\mathcal{T})$, as wished. $\square$

For the relaxed semantics, we discuss only the interesting cases where control-stuck terms can be produced, which are shift and pctx.

Lemma A.4. $\mathsf{shift} \rightsquigarrow_{\mathsf{s}} \widehat{\mathsf{strong}(\mathfrak{F})}^{\omega}, \widehat{\mathfrak{F}}^{\omega}$.

Proof. Let $\mathcal{R} \rightarrowtail\!\!\!\!\twoheadrightarrow_{\mathsf{r}} \mathcal{S}, \mathcal{T}$ and $t_0 \mathrel{\mathcal{R}} t_1$. For fresh c and x, we have to relate $t_0\{\lambda x.c[x]/k\}$ and $t_1\{\lambda x.c[x]/k\}$, which is direct with refl and subst. $\square$

Lemma A.5. $\mathsf{pctx} \rightsquigarrow \widehat{\mathsf{strong}(\mathfrak{F})}^{\omega} \circ \widehat{\mathfrak{F}} \circ \widehat{\mathsf{strong}(\mathfrak{F})}^{\omega}, \widehat{\mathfrak{F}}^{\omega}$

Sketch. Let $\mathcal{R} \rightarrowtail\!\!\!\!\twoheadrightarrow_{\mathsf{r}} \mathcal{R}, \mathcal{T}$. Let $E_0[t_0]$ $\mathsf{ectxpure}(\mathcal{R})$ $E_1[t_1]$ with $t_0 \mathrel{\mathcal{R}} t_1$, $t_0 = E_0'[\mathcal{S}k.t_0']$, and $E_0[x] \mathrel{\mathcal{R}} E_1[x]$ for a fresh x. Let c, c' be fresh context variables. There exist E_1' and t_1' such that $t_1 \Downarrow_{\text{v}} E_1'[\mathcal{S}k.t_1']$ and $\langle t_0'\{\lambda y.c'[E_0'[y]]/k\}\rangle \mathrel{\mathcal{T}} \langle t_1'\{\lambda y.c'[E_1'[y]]/k\}\rangle$. Then $\langle t_0'\{\lambda y.c'[E_0'[y]]/k\}\rangle\{c[E_0]/c'\}$ $\mathsf{csubst}(\mathcal{T})$ $\langle t_1'\{\lambda y.c'[E_1'[y]]/k\}\rangle\{c[E_1]/c'\}$, from which we get $\langle t_0'\{\lambda y.c[E_0[E_0'[y]]]/k\}\rangle \mathrel{\mathcal{T}} \langle t_1'\{\lambda y.c[E_1[E_1'[y]]]/k\}\rangle$, as wished. $\square$