A Quick Survey on Bisimulations for Delimited-Control Operators

Dariusz Biernacki Sergueï Lenglet

April 12th 2015

Abstract

We present a survey of the behavioral theory of the delimited-control operators *shift* and *reset*. We first define a notion of contextual equivalence, that we then aim to characterize with bisimilarities. We consider several styles of bisimilarities, namely normal form, applicative, and environmental. Each style has its strengths and weaknesses, and we provide several examples to allow comparisons between the different kinds of equivalence proofs.

Contents

List of Figures

1 Introduction

Control operators for delimited continuations [9, 11] provide elegant means for expressing advanced control mechanisms [9, 15]. Moreover, they play a fundamental role in the semantics of computational effects [12], normalization by evaluation [3] and as a crucial refinement of abortive control operators such as *callcc* [11, 24]. Of special interest are control operators *shift* and *reset* [9] due to their origins in continuation-passing style (CPS) and their connection with computational monads [12]. The control delimiter reset delimits the current continuation and the control operator shift abstracts the current delimited continuation as a first class value that when resumed is composed with the then-current continuation.

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* [20] is widely considered as the most natural equivalence on terms in languages similar to 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 [23], 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 executed 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 context 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 *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., is included into and 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, and also because we do not have to consider

all forms of contexts. Applicative bisimilarity is usually sound and complete w.r.t. contextual equivalence, at least for deterministic languages such as the plain λ-calculus [1].

In contrast with applicative bisimilarity, *normal form* bisimilarity [18] does not contain any quantification over arguments or contexts in its definition. The principle is to reduce the compared terms to values (if possible), and then to decompose the resulting values into sub-components that have to be themselves bisimilar. Unlike applicative 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.

Finally, *environmental bisimilarity* [22] 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 (like normal form bisimilarity) to simplify its equivalence proofs. In contrast, the definition of up-to techniques for applicative bisimilarity remains an open problem.

In this article, we propose a survey of our previously published work [6, 5, 7] on the behavioral theory of a λ-calculus extended with the operators shift and reset. We first define a notion of contextual equivalence, that we aim to characterize with the three styles of bisimilarities discussed above. We provide several examples to show how to prove that two terms are equivalent with each bisimulation style.

Section 2 presents the syntax and semantics of the calculus $\lambda_{\mathcal{S}}$ with shift and reset we use in this paper. In this section, we also remind the definition of CPS equivalence, a CPS-based equivalence between terms, and discusse the definition of a contextual equivalence for $\lambda_{\mathcal{S}}$. We look for (at least sound) alternatives of this contextual equivalence by considering several styles of bisimilarities: normal form in Section 3, applicative in Section 4, and environmental in Section 5. Section 6 concludes this paper. Section 3 summarizes results presented in [6], Section 4 results in [5], and Section 5 results in [7].

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

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

2.1 Syntax

The language $\lambda_{\mathcal{S}}$ extends the call-by-value λ-calculus with the delimited-control operators *shift* and *reset* [9]. We assume we have a set of term variables, ranged over by x, y, z, and k. We use the metavariable k for term variables representing

a continuation (e.g., when bound with a shift), while x, y, and z stand for any values; we believe such distinction helps to understand examples and reduction rules. The syntax of terms and values is given by the following grammars:

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

The operator *shift* ($\mathcal{S}k.t$) is a capture operator, the extent of which is determined by the delimiter *reset* ($\langle\cdot\rangle$). A λ-abstraction $\lambda x.t$ binds x in t and a shift construct $\mathcal{S}k.t$ binds k in t; terms are equated up to α-conversion of their bound variables. The set of free variables of t is written $\mathsf{fv}(t)$; a term is *closed* if $\mathsf{fv}(t) = \emptyset$.

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

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

Regular contexts are ranged over by C. The pure evaluation contexts[1] (abbreviated as pure contexts), ranged over by E, represent delimited continuations and can be captured by the shift operator. The call-by-value evaluation contexts, ranged over by F, represent arbitrary continuations and encode the chosen reduction strategy. Filling a context C (respectively E, F) with a term t produces a term, written $C[t]$ (respectively $E[t]$, $F[t]$); the free variables of t may be captured in the process. We extend the notion of free variables to contexts (with $\mathsf{fv}(\Box) = \emptyset$), and we say a context C (respectively E, F) is *closed* if $\mathsf{fv}(C) = \emptyset$ (respectively $\mathsf{fv}(E) = \emptyset$, $\mathsf{fv}(F) = \emptyset$). In any definitions or proofs, we say a variable is *fresh* if it does not occur free in the terms or contexts under consideration.

2.2 Reduction Semantics

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

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

The term $(\lambda x.t)\ v$ is the usual call-by-value redex for β-reduction (rule (β_v)). The operator $\mathcal{S}k.t$ captures its surrounding context E up to the dynamically nearest enclosing reset, and substitutes $\lambda x.\langle E[x] \rangle$ for k in t (rule (*shift*)). If a reset is enclosing a value, then it has no purpose as a delimiter for a potential capture, and it can be safely removed (rule (*reset*)). All these reductions may occur within a metalevel context F. The chosen call-by-value evaluation strategy is encoded in the grammar of the evaluation contexts. Furthermore, the reduction relation $\to_{\mathrm{v}}$ is compatible with evaluation contexts F, i.e., $F[t] \to_{\mathrm{v}} F[t']$ whenever $t \to_{\mathrm{v}} t'$.

[1] This terminology comes from Kameyama (e.g., in [17]).

Example 2.1 (fixed-point combinators)**.** We remind 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 $\Theta \stackrel{\mathsf{def}}{=} \theta\ \theta$ is Turing's call-by-value fixed-point combinator, and $\Delta \stackrel{\mathsf{def}}{=} \lambda x.\delta_x\ \delta_x$ is Curry's call-by-value fixed-point combinator. In [8], 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 $\lambda x.\langle \delta_x\ \mathcal{S}k.k\ k\rangle$. We use the combinators and their delimited-control variants as examples throghout the paper, and we study in particular the equivalences between them in Example 3.3.

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

Definition 2.2. 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 2.3. *A term t is stuck iff $t = E[\mathcal{S}k.t']$ for some E, k, and t' or $t = F[x\ v]$ for some F, x, and v.*

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

Definition 2.4. 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 2.5. *For all terms t, either t is a normal form, or there exist a unique redex r and a unique context F such that $t = F[r]$.*

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

Definition 2.6. We write $t \Downarrow_{\mathrm{v}} t'$ if $t \to_{\mathrm{v}}^* t'$ and t' cannot reduce further.

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

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

Proposition 2.7. *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.)*

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

Figure 1: Kameyama and Hasegawa's axiomatization of $\lambda_{\mathcal{S}}$

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 [10, 12], but also in some other studies of these operators [2, 16]. This is in contrast to the original reduction semantics for shift and reset [4] that has been obtained from the 2-layered continuation-passing-style (CPS) semantics [9], 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.

The two semantics, therefore, differ in that in the original semantics there are no 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 behaviour.

The top-level reset requirement, imposed by the original semantics, does not lend itself naturally to the normal-form and applicative bismulation techniques that we propose for the relaxed semantics in Sections 3 and 4. We show, however, that the requirement can be successfully treated in the framework of environmental bisimulations, presented in Section 5.

2.4 CPS Equivalence

The operators shift and reset have been originally defined by a translation into continuation-passing style [9]. This CPS translation induces the following notion of equivalence on $\lambda_{\mathcal{S}}$ terms:

Definition 2.8. Terms t and t' are CPS equivalent if their CPS translations

are $\beta\eta$-convertible.

For example, the reduction rules $t \to_{\mathrm{v}} t'$ given in Section 2.2 are sound w.r.t. the 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 [17]: two $\lambda_{\mathcal{S}}$ terms are CPS equivalent iff one can derive their equality using the equations of Figure 1.

The axiomatization is a source of examples for the bisimulation techniques that we study in Sections 3, 4 and 5, and it allows us to relate the notion of CPS equivalence to the notions of contextual equivalence that we introduce in the next section. 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 environmental bisimilarity for 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.

2.5 Contextual Equivalence

In this section, we discuss the possible definitions of a Morris-style contextual equivalence for the calculus $\lambda_{\mathcal{S}}$. As usual, the idea is to express that two terms are equivalent iff they cannot be distinguished when put in an arbitrary context. The question is then what kind of behavior we want to observe. In $\lambda_{\mathcal{S}}$, the evaluation of closed terms generates not only values, but also control stuck terms. Taking this into account, we obtain the following definition of contextual equivalence.

Definition 2.9. Let t_0, t_1 be closed terms. We write $t_0 \mathbb{C} t_1$ if for all closed C,

- $C[t_0] \Downarrow_{\mathrm{v}} v_0$ 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.

The relation $\mathbb{C}$ is defined on closed terms, but can be extended to open terms using closing substitutions: we say σ closes t if it maps the free variables of t to closed values. The *open extension* of a relation, written $\mathcal{R}^\circ$, is defined as follows.

Definition 2.10. Let $\mathcal{R}$ be a relation on closed terms, and t_0 and t_1 be open terms. We write $t_0 \ \mathcal{R}^\circ \ t_1$ if for every substitution σ which closes t_0 and t_1, $t_0\sigma \ \mathcal{R} \ t_1\sigma$ holds.

The relation $\mathbb{C}$ is 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 $\mathcal{S}_{\mathtt{elim}}$ in Figure 1), 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.

2.6 Contextual Equivalence for the Original Semantics

To reflect the fact that in the original semantics terms are evaluated within an enclosing reset, the contextual equivalence we consider for the original semantics tests terms in contexts of the form $\langle C \rangle$ only. Because delimited terms cannot reduce to stuck terms (Proposition 2.7), the only possible observable action is evaluation to values. We therefore define contextual equivalence for delimited terms as follows.

Definition 2.11. Let t_0, t_1 be closed terms. We write $t_0 \mathbb{P} t_1$ if for all closed C, $\langle C[t_0] \rangle \Downarrow_{\mathrm{v}} v_0$ iff $\langle C[t_1] \rangle \Downarrow_{\mathrm{v}} v_1$.

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

Proposition 2.12. *We have* $\mathbb{C} \subseteq \mathbb{P}$.

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. $\mathbb{C}$ (like the bisimilarities we define in Sections 3, 4, and 5.1) is also sound w.r.t. $\mathbb{P}$. However, to reach completeness, we have to design a bisimilarity suitable for delimited terms (see Section 5.2).

The inclusion of Proposition 2.12 is strict; in particular, $\mathbb{P}$ verifies the axiom $\mathcal{S}_{\mathtt{elim}}$, while $\mathbb{C}$ does not. In fact, we prove in Section 5.2 that $\mathbb{P}$ contains the CPS equivalence $\equiv$. The reverse inclusion does not hold (for $\mathbb{P}$ as well as for $\mathbb{C}$): there exists contextually equivalent terms that are not CPS equivalent.

Proposition 2.13.
1. *We have* $\Omega \mathbb{P} \Omega\Omega$ *(respectively* $\Omega \mathbb{C} \Omega\Omega$*), but* $\Omega \not\equiv \Omega\Omega$.
2. *We have* $\Theta \mathbb{P} \Delta$ *(respectively* $\Theta \mathbb{C} \Delta$*), but* $\Theta \not\equiv \Delta$.

The contextual equivalences $\mathbb{C}$ and $\mathbb{P}$ 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. $\mathbb{C}$ or $\mathbb{P}$) by studying different styles of bisimulation in the next sections.

3 Normal Form Bisimilarity

Normal form bisimilarity [18] 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 (studied in the next sections), normal form bisimilarity usually does not contain a universal quantification over testing terms or contexts in its definition, and is therefore easier

Definition of $\star$ on values:

$$x \star y \stackrel{\mathsf{def}}{=} x\ y \qquad \lambda x.t \star y \stackrel{\mathsf{def}}{=} t\{y/x\}$$

Definition of $\mathcal{R}^{\mathsf{NF}\eta}$ on normal forms and contexts

$$\frac{E[x]\ \mathcal{R}\ E'[x] \qquad x \text{ fresh}}{E\ \mathcal{R}^{\mathsf{NF}\eta}\ E'} \qquad \frac{\langle E[x]\rangle\ \mathcal{R}\ \langle E'[x]\rangle \qquad F[x]\ \mathcal{R}\ F'[x] \qquad x \text{ fresh}}{F[\langle E\rangle]\ \mathcal{R}^{\mathsf{NF}\eta}\ F'[\langle E'\rangle]}$$

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

$$\frac{F_0\ \mathcal{R}^{\mathsf{NF}\eta}\ F_1 \qquad v_0\ \mathcal{R}^{\mathsf{NF}\eta}\ v_1}{F_0[x\ v_0]\ \mathcal{R}^{\mathsf{NF}\eta}\ F_1[x\ v_1]}$$

Figure 2: Definitions of the operator $\star$ and the relation $\mathcal{R}^{\mathsf{NF}\eta}$

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. This section summarizes the results in [6].

3.1 Definition

In the λ-calculus [21, 18], the definition of normal form bisimilarity has to take into account only values and open stuck terms. In $\lambda_{\mathcal{S}}$ with the relaxed semantics, we have to relate also control stuck terms; we propose here a first way to deal with these terms, that will be refined in the next subsection. Deconstructing normal forms leads to comparing contexts as well as terms. Given a relation $\mathcal{R}$ on terms, we define in Fig. 2 an extension of $\mathcal{R}$ to normal forms and contexts, written $\mathcal{R}^{\mathsf{NF}\eta}$, which relies on an application operator for values $\star$. The rationale behind the definitions of $\star$ and $\mathcal{R}^{\mathsf{NF}\eta}$ becomes clear when we explain our notion of normal form bisimilarity, defined below.

Definition 3.1. A relation $\mathcal{R}$ on terms is a normal form simulation if $t_0\ \mathcal{R}\ t_1$ and $t_0 \Downarrow_{\mathrm{v}} t_0'$ implies $t_1 \Downarrow_{\mathrm{v}} t_1'$ and $t_0'\ \mathcal{R}^{\mathsf{NF}\eta}\ t_1'$. A relation $\mathcal{R}$ is a normal form bisimulation if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are normal form simulations. Normal form bisimilarity, written $\mathbb{N}$, is the largest normal form bisimulation.

In this section, we often drop the "normal form" attribute when it does not cause confusion. Two terms t_0 and t_1 are bisimilar if their evaluations lead to matching normal forms (e.g., if t_0 evaluates to a control stuck term, then so does t_1) with bisimilar sub-components. We now detail the different cases.

Normal form bisimilarity does not distinguish between evaluation to a variable and evaluation to a λ-abstraction. Instead, we relate terms evaluating to

any values v_0 and v_1 by comparing $v_0 \star x$ and $v_1 \star x$, where x is fresh. As originally pointed out by Lassen [18], this is necessary for the bisimilarity to be sound w.r.t. η-expansion; otherwise it would distinguish η-equivalent terms such as $\lambda y.x\, y$ and x. Using $\star$ instead of regular application avoids the introduction of unnecessary β-redexes, which could reveal themselves problematic in proofs.

For a control stuck term $E_0[\mathcal{S}k.t_0]$ to be executed, it has to be plugged into an evaluation context surrounded by a reset; by doing so, we obtain a term of the form $\langle t_0\{\lambda x.\langle E_0'[x]\rangle/k\}\rangle$ for some context E_0'. Notice that the resulting term is within a reset; similarly, when comparing $E_0[\mathcal{S}k.t_0]$ and $E_1[\mathcal{S}k.t_1]$, we ask for the shift bodies t_0 and t_1 to be related when surrounded by a reset. We also compare E_0 and E_1, which amounts to executing $E_0[x]$ and $E_1[x]$ for a fresh x, since the two contexts are pure. Comparing t_0' and t_1' without reset would be too discriminating, as it would distinguish contextually equivalent terms such as $\mathcal{S}k.\langle t\rangle$ and $\mathcal{S}k.t$ (axiom $\mathcal{S}_{\langle\cdot\rangle}$). Indeed, without reset, we would have to relate $\langle t\rangle$ and t, which are not equivalent in general (take $t = \mathcal{S}k'.v$ for some v), while Definition 3.1 requires $\langle\langle t\rangle\rangle$ and $\langle t\rangle$ to be related (which holds for all t; see Example 3.2).

The open stuck terms $F_0[x\, v_0]$ and $F_1[x\, v_1]$ are bisimilar if the values v_0 and v_1 as well as the contexts F_0 and F_1 are related. We have to be careful when defining bisimilarity on (possibly non pure) evaluation contexts. We cannot simply relate F_0 and F_1 by executing $F_0[y]$ and $F_1[y]$ for a fresh y. Such a definition would equate the contexts $\Box$ and $\langle\Box\rangle$, which in turn would relate the terms $x\, v$ and $\langle x\, v\rangle$, which are not contextually equivalent: they are distinguished by the context $(\lambda x.\Box)\, \lambda y.\mathcal{S}k.\Omega$. A context containing a reset enclosing the hole should be related only to contexts with the same property. However, we do not want to precisely count the number of delimiters around the hole; doing so would distinguish $\langle\Box\rangle$ and $\langle\langle\Box\rangle\rangle$, and therefore it would discriminate the contextually equivalent terms $\langle x\, v\rangle$ and $\langle\langle x\, v\rangle\rangle$. Hence, the definition of $\mathcal{R}^{\mathsf{NF}\eta}$ for contexts (Fig. 2) checks that if one of the contexts contains a reset surrounding the hole, then so does the other; then it compares the contexts beyond the first enclosing delimiter by simply evaluating them using a fresh variable. As a result, it rightfully distinguishes $\Box$ and $\langle\Box\rangle$, but it relates $\langle\Box\rangle$ and $\langle\langle\Box\rangle\rangle$.

We now give some examples to show how to prove equivalences using normal form bisimulation.

Example 3.2 (double reset)**.** We prove that $\langle t\rangle\ \mathbb{N}\ \langle\langle t\rangle\rangle$ by showing that $\mathcal{R} \stackrel{\mathsf{def}}{=} \{(\langle t\rangle, \langle\langle t\rangle\rangle)\} \cup \mathbb{N}$ is a bisimulation. First, note that the case $\langle t\rangle \Downarrow_{\mathrm{v}} E[\mathcal{S}k.t']$ is not possible because of Proposition 2.7. Suppose $\langle t\rangle \Downarrow_{\mathrm{v}} v$. we prove that $\langle t\rangle \Downarrow_{\mathrm{v}} v$ iff $\langle\langle t\rangle\rangle \Downarrow_{\mathrm{v}} v$. If $\langle t\rangle \Downarrow_{\mathrm{v}} v$, then $\langle\langle t\rangle\rangle \to_{\mathrm{v}}^* \langle v\rangle \to_{\mathrm{v}} v$. Conversely, if $\langle\langle t\rangle\rangle \Downarrow_{\mathrm{v}} v$, then $\langle t\rangle$ cannot diverge or cannot reduce to an open stuck term (otherwise, $\langle\langle t\rangle\rangle$ would also diverge or reduce to an open stuck term). Hence, we have $\langle t\rangle \Downarrow_{\mathrm{v}} v'$, which entails $\langle\langle t\rangle\rangle \to_{\mathrm{v}}^* \langle v'\rangle \to_{\mathrm{v}} v'$, which in turn implies $v = v'$ because normal forms are unique. Consequently, we have $\langle t\rangle \Downarrow_{\mathrm{v}} v$ iff $\langle\langle t\rangle\rangle \Downarrow_{\mathrm{v}} v$, and $v\ \mathbb{N}^{\mathsf{NF}\eta}\ v$ holds.

If $\langle t\rangle \Downarrow_{\mathrm{v}} F[x\, v]$, then there exists F' such that $t \Downarrow_{\mathrm{v}} F'[x\, v]$ and $F = \langle F'\rangle$.

Therefore, we have $\langle\langle t\rangle\rangle \Downarrow_{\mathrm{v}} \langle\langle F'[x\ v]\rangle\rangle$. We have $v\ \mathbb{N}^{\mathsf{NF}\eta}\ v$, and we have to prove that $\langle F'\rangle\ \mathcal{R}^{\mathsf{NF}\eta}\ \langle\langle F'\rangle\rangle$ to conclude. If F' is a pure context E, then we have to prove $\langle E[y]\rangle\ \mathcal{R}\ \langle E[y]\rangle$ and $y\ \mathcal{R}\ \langle y\rangle$ for a fresh y, which are both true because $\mathbb{N} \subseteq \mathcal{R}$. 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 $\mathbb{N} \subseteq \mathcal{R}$).

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

Example 3.3 (fixed-point combinators)**.** We study here the relationships between Turing's and Curry's fixed-point combinator and their respective variants with delimited control [8] (see Example 2.1 for the definitions). First, we prove that Turing's combinator Θ is bisimilar to its variant $\Theta_{\mathcal{S}} \stackrel{\mathsf{def}}{=} \langle\theta\ \mathcal{S}k.k\ k\rangle$. We build the candidate relation $\mathcal{R}$ incrementally, starting from $(\Theta, \Theta_{\mathcal{S}})$. Evaluating these two terms, we obtain

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

We therefore extend $\mathcal{R}$ with $(v_0 \star y, v_1 \star y)$, where y is fresh. These two new terms are open stuck, so we add there 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 $(v_0' \star z, v_1' \star z)$ and (z, z) for a fresh z to $\mathcal{R}$. Evaluating $v_0' \star z$ and $v_1' \star z$, we obtain respectively $y\ v_0'\ z$ and $y\ v_1'\ z$; to relate these two open stuck terms, we just need to add $(x\ z, x\ z)$ (for a fresh x) to $\mathcal{R}$, since we already have $v_0'\ \mathcal{R}^{\mathsf{NF}\eta}\ v_1'$. The constructed relation $\mathcal{R}$ we obtain is a normal form bisimulation.

In contrast, Curry's combinator Δ is not bisimilar to its delimited-control variant $\Delta_{\mathcal{S}} \stackrel{\mathsf{def}}{=} \lambda x.\langle\delta_x\ \mathcal{S}k.k\ k\rangle$. Indeed, evaluating the bodies of the two values, we obtain respectively $x\ (\lambda z.\delta_x\ \delta_x\ z)$ and $\langle\langle x\ (\lambda z.(\lambda y.\langle\delta_x\ y\rangle)\ (\lambda y.\langle\delta_x\ y\rangle)\ z)\rangle\rangle$, and these open stuck terms are not bisimilar, because $\Box$ is not related to $\langle\langle\Box\rangle\rangle$ by $\mathbb{N}^{\mathsf{NF}\eta}$. In fact, Δ and $\Delta_{\mathcal{S}}$ are distinguished by the context $\Box\ \lambda x.\mathcal{S}k.\Omega$. Finally, we can prove that the two original combinators Θ and Δ are bisimilar, using the same bisimulation as in [18].

The bisimilarity $\mathbb{N}$ is sound w.r.t. contextual equivalence.

Theorem 3.4. *We have* $\mathbb{N} \subseteq \mathbb{C}$.

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

Proposition 3.5. *Let* $i \stackrel{\mathsf{def}}{=} \lambda y.y$. *We have* $\langle\langle x\ i\rangle\ \mathcal{S}k.i\rangle\ \mathbb{C}^{\circ}\ \langle\langle x\ i\rangle\ (\langle x\ i\rangle\ \mathcal{S}k.i)\rangle$, *but* $\langle\langle x\ i\rangle\ \mathcal{S}k.i\rangle\ \not{\mathbb{N}}\ \langle\langle x\ i\rangle\ (\langle x\ i\rangle\ \mathcal{S}k.i)\rangle$.

3.2 Proving the Axioms

We now show how the axioms can be proved using normal form bisimulation. Because we work with the relaxed semantics in this section, we remind that the

$\mathcal{S}_{\mathtt{elim}}$ axiom does not hold, as discussed in Section 2.5. The η-equivalence axiom (η_{v} axiom) holds by definition of $\mathbb{N}^{\mathsf{NF}\eta}$.

Proposition 3.6 ($\mathcal{S}_{\langle\cdot\rangle}$ axiom)**.** *We have* $\mathcal{S}k.\langle t\rangle\ \mathbb{N}\ \mathcal{S}k.t$.

Proof. We want to relate two stuck terms, so using normal form bisimulation, we have to show $\langle\langle t\rangle\rangle\ \mathbb{N}\ \langle t\rangle$ (proved in Example 3.2) and $\Box\ \mathbb{N}^{\mathsf{NF}\eta}\ \Box$ (a consequence of the fact that $\mathbb{N}$ is reflexive). □

Proposition 3.7 ($\langle\cdot\rangle_{\mathtt{lift}}$ axiom)**.** *We have* $\langle(\lambda x.t_0)\ \langle t_1\rangle\rangle\ \mathbb{N}\ (\lambda x.\langle t_0\rangle)\ \langle t_1\rangle$.

Proof. We prove that $\mathcal{R} \stackrel{\mathsf{def}}{=} \{(\langle(\lambda x.t_0)\ \langle t_1\rangle\rangle, (\lambda x.\langle t_0\rangle)\ \langle t_1\rangle)\} \cup \{(t,t)\}$ is a normal form bisimulation. The terms $\langle(\lambda x.t_0)\ \langle t_1\rangle\rangle$ and $(\lambda x.\langle t_0\rangle)\ \langle t_1\rangle$ reduces to a normal form iff $\langle t_1\rangle$ reduces to a normal form, and according to Proposition 2.7, we have two cases.

If $\langle t_1\rangle \Downarrow_{\mathsf{v}} v$, then $\langle(\lambda x.t_0)\ \langle t_1\rangle\rangle \to_{\mathsf{v}}^* \langle t_0\{v/x\}\rangle$ and $(\lambda x.\langle t_0\rangle)\langle t_1\rangle \to_{\mathsf{v}}^* \langle t_0\{v/x\}\rangle$. Therefore, $\langle(\lambda x.t_0)\ \langle t_1\rangle\rangle \Downarrow_{\mathsf{v}} t''$ iff $(\lambda x.\langle t_0\rangle)\ \langle t_1\rangle \Downarrow_{\mathsf{v}} t''$, and we have $t''\ \mathcal{R}^{\mathsf{NF}\eta}\ t''$, as required.

If $\langle t_1\rangle$ reduces to an open stuck term, then $\langle t_1\rangle \Downarrow_{\mathsf{v}} \langle F[y\,v]\rangle$ by Proposition 2.7. In this case, we have $\langle(\lambda x.t_0)\ \langle t_1\rangle\rangle \Downarrow_{\mathsf{v}} \langle(\lambda x.t_0)\ \langle F[y\,v]\rangle\rangle$ and also $(\lambda x.\langle t_0\rangle)\ \langle t_1\rangle \Downarrow_{\mathsf{v}} (\lambda x.\langle t_0\rangle)\ \langle F[y\,v]\rangle$. We have $\langle(\lambda x.t_0)\ \langle F\rangle\rangle\ \mathcal{R}^{\mathsf{NF}\eta}\ (\lambda x.\langle t_0\rangle)\ \langle F\rangle$ and $v\ \mathcal{R}^{\mathsf{NF}\eta}\ v$ by definition of $\mathcal{R}$, as required.

□

Proposition 3.8 (β_Ω axiom)**.** *If* $x \notin \mathsf{fv}(E)$*, then* $(\lambda x.E[x])\ t\ \mathbb{N}\ E[t]$.

Proof. We prove that $\mathcal{R} \stackrel{\mathsf{def}}{=} \{((\lambda x.E[x])\ t, E[t]), x \notin \mathsf{fv}(E)\} \cup \{(t,t)\}$ is a normal form bisimulation. If $(\lambda x.E[x])t$ evaluates to some normal form, then t evaluates to some normal form as well. We distinguish three cases. If $t \Downarrow_{\mathsf{v}} v$, then $(\lambda x.E[x])\ t \to_{\mathsf{v}}^* E[v]$ (because $x \notin \mathsf{fv}(E)$), and $E[t] \to_{\mathsf{v}}^* E[v]$. We obtain the same term in both cases, and from there, it is easy to conclude.

If $t \Downarrow_{\mathsf{v}} F[y\,v]$, then $(\lambda x.E[x])\,t \Downarrow_{\mathsf{v}} (\lambda x.E[x])\,F[y\,v]$, and $E[t] \Downarrow_{\mathsf{v}} E[F[x\,v]]$. We have to prove $v\ \mathcal{R}^{\mathsf{NF}\eta}\ v$, which is obvious, and $(\lambda x.E[x])\,F\ \mathcal{R}^{\mathsf{NF}\eta}\ E[F]$. Let z be a fresh variable. If F is a pure context E', we have to prove $(\lambda x.E[x])\ E'[z]\ \mathcal{R}\ E[E'[z]]$, which is clearly true. Otherwise $F = F'[\langle E'\rangle]$, and we have to prove $(\lambda x.E[x])\ F'[z]\ \mathcal{R}\ E[F'[z]]$, which is clearly true, and $\langle E'[z]\rangle\ \mathcal{R}\ \langle E'[z]\rangle$, which is true as well because $\mathcal{R}$ contains the identity relation.

If $t \Downarrow_{\mathsf{v}} E'[\mathcal{S}k.t']$, then we have $(\lambda x.E[x])t \Downarrow_{\mathsf{v}} (\lambda x.E[x])\,E'[\mathcal{S}k.t']$, and $E[t] \Downarrow_{\mathsf{v}} E[E'[\mathcal{S}k.t']]$. Let y be a fresh variable. We have to prove $(\lambda x.E[x])\ E'[y]\ \mathcal{R}\ E[E'[y]]$, which is clearly true, and $\langle t'\rangle\ \mathcal{R}\ \langle t'\rangle$, which is true as well.

□

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. In this

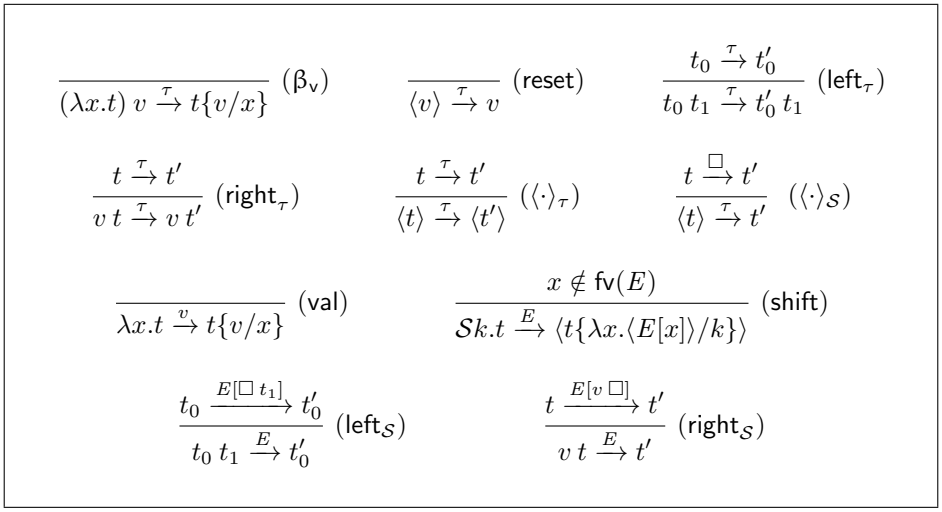

Figure 3: Labelled Transition System

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 labelled transition system, introduced first. We then define the relation itself, before showing how it can be used on some examples. This section summarizes the results in [5].

4.1 Labelled Transition System

One possible way to define an applicative bisimilarity is to rely on a labelled transition system (LTS), where the possible interactions of a term with its environment are encoded in the labels (see, e.g., [14, 13]). Using a LTS simplifies the definition of the bisimilarity and makes 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'. We do not have a case for open stuck terms, because we work with closed terms only.

Most rules for internal actions (Fig. 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. Note that 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 [19]. 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 ($\mathsf{left}_{\mathcal{S}}$), 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 premise of the rule ($\mathsf{left}_{\mathcal{S}}$), 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 ($\mathsf{right}_{\mathcal{S}}$) 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 $t \xrightarrow{\Box} t'$.

We now prove that the LTS corresponds to the reduction semantics $\rightarrow_{\mathrm{v}}$ 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 explicit the correspondence between $\xrightarrow{E}$ and control stuck terms.

Lemma 4.1. *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 direct by induction on $t \xrightarrow{E} t'$. From this lemma, we can deduce the correspondence between $\xrightarrow{\tau}$ and $\rightarrow_{\mathrm{v}}$, and between $\xrightarrow{\alpha}$ (for $\alpha \neq \tau$) and the observable actions of the language.

Proposition 4.2. *The following hold:*

- *We have $\xrightarrow{\tau} = \rightarrow_{\mathrm{v}}$.*
- *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 $t\ v \xrightarrow{\tau} t'$.*

4.2 Applicative Bisimilarity

We now define the notion of applicative bisimilarity for $\lambda_{\mathcal{S}}$. We write $\Rightarrow$ for the reflexive and transitive closure of $\xrightarrow{\tau}$. We define the weak delay[2] transition $\overset{\alpha}{\Rightarrow}$ as $\Rightarrow$ if $\alpha = \tau$ and as $\Rightarrow\xrightarrow{\alpha}$ otherwise. The definition of the (weak delay) bisimilarity is then straightforward.

[2]where internal steps are allowed before, but not after a visible action

Definition 4.3. 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 $t_1 \stackrel{\alpha}{\Rightarrow} t_1'$ 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 simulations. Applicative bisimilarity $\mathbb{A}$ 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. The relation $\mathbb{A}$ is sound and complete w.r.t. contextual equivalence.

Theorem 4.4. *We have* $\mathbb{A} = \mathbb{C}$.

We give some examples showing how applicative bisimulation can be used to prove the equivalence of terms.

Example 4.5 (double reset)**.** We show that $\langle\langle t\rangle\rangle \mathbb{A} \langle t\rangle$ holds by proving that $\mathcal{R} \stackrel{\mathsf{def}}{=} \{(\langle t\rangle, \langle\langle t\rangle\rangle)\} \cup \{(t,t)\}$ is a big-step applicative bisimulation. If $\langle t\rangle$ and/or $\langle\langle t\rangle\rangle$ is open, then $\langle t\rangle\sigma = \langle t\sigma\rangle$ (and similarly with $\langle\langle t\rangle\rangle$), for all closing substitution σ, so we still have terms in $\mathcal{R}$. With closed terms, the only possible (big-step) transition is $\langle t\rangle \stackrel{v}{\Rightarrow} t'$, which means $\langle t\rangle \Downarrow_{\mathrm{v}} v' \stackrel{v}{\rightarrow} t'$. But we have proved in Example 3.2 that $\langle t\rangle \Downarrow_{\mathrm{v}} v'$ iff $\langle\langle t\rangle\rangle \Downarrow_{\mathrm{v}} v'$. Consequently, we have $\langle t\rangle \stackrel{v}{\Rightarrow} t'$ iff $\langle\langle t\rangle\rangle \stackrel{v}{\Rightarrow} t'$, and we have $t' \mathcal{R} t'$, as wished. The proof is shorter than in Example 3.2 because we do not have to consider open stuck terms.

Example 4.6 (Turing's combinator)**.** We now consider Turing's combinator Θ and its variant $\Theta_{\mathcal{S}} \stackrel{\mathsf{def}}{=} \langle \theta\, \mathcal{S}k.k\, k\rangle$. The two terms can perform the following transitions.

$$\Theta \stackrel{v}{\Rightarrow} v\ (\lambda z.\theta\ \theta\ v\ z)$$
$$\Theta_{\mathcal{S}} \stackrel{v}{\Rightarrow} v\ (\lambda z.(\lambda x.\langle\theta\ x\rangle)\ (\lambda x.\langle\theta\ x\rangle)\ v\ z).$$

Assuming $v = \lambda x.t$, we have to study the behaviour of $t\{(\lambda z.\theta\ \theta\ v\ z)/x\}$, 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 $t = F[x\ v']$. The resulting applicative bisimulation one can write to relate Θ and $\Theta_{\mathcal{S}}$ is much more complex than the normal form bisimulation of Example 3.3.

4.3 Proving the Axioms

As with normal form bisimulation (Section 3.2), 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 deduce directly from the results with closed terms.

Proposition 4.7 (η_{v} axiom)**.** *If* $x \notin \mathsf{fv}(v)$*, then* $\lambda x.v\ x\ \mathbb{A}\ v$.

Proof. We prove that $\mathcal{R} \stackrel{\mathsf{def}}{=} \{(\lambda x.(\lambda y.t)\, x \mid \lambda y.t), x \notin \mathsf{fv}(t)\} \cup \mathbb{A}$ 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 $\xrightarrow{\tau} \subseteq \mathbb{A} \subseteq \mathcal{R}$, we have the required result. □

Proposition 4.8 ($\mathcal{S}_{\langle\cdot\rangle}$ axiom)**.** *We have* $\mathcal{S}k.\langle t\rangle\ \mathbb{A}\ \mathcal{S}k.t$.

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 4.5 that $\langle\langle t'\rangle\rangle\ \mathbb{A}\ \langle t'\rangle$ holds. □

Proposition 4.9 ($\langle\cdot\rangle_{\texttt{lift}}$ axiom)**.** *We have* $\langle(\lambda x.t_0)\ \langle t_1\rangle\rangle\ \mathbb{A}\ (\lambda x.\langle t_0\rangle)\ \langle t_1\rangle$.

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 2.7). 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 4.10 (β_Ω axiom)**.** *If* $x \notin \mathsf{fv}(E)$*, then* $(\lambda x.E[x])\ t\ \mathbb{A}\ E[t]$.

Sketch. We first give some intuitions on why the proof of this result is harder with applicative bisimulation than with normal form 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'.

□

5 Environmental Bisimilarity

Like applicative bisimilarity, environmental bisimilarity reduces closed terms to normal forms, which are then compared using some particular contexts (e.g.,

Term generating closure

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

Context generating closure

$$\frac{}{\Box\ \ddot{\mathcal{R}}\ \Box} \qquad \frac{F_0\ \ddot{\mathcal{R}}\ F_1 \qquad v_0\ \dot{\mathcal{R}}\ v_1}{v_0\ F_0\ \ddot{\mathcal{R}}\ v_1\ F_1} \qquad \frac{F_0\ \ddot{\mathcal{R}}\ F_1 \qquad t_0\ \dot{\mathcal{R}}\ t_1}{F_0\ t_0\ \ddot{\mathcal{R}}\ F_1\ t_1} \qquad \frac{F_0\ \ddot{\mathcal{R}}\ F_1}{\langle F_0\rangle\ \ddot{\mathcal{R}}\ \langle F_1\rangle}$$

Figure 4: Term and context generating closures

λ-abstractions are tested by passing them arguments). However, the testing contexts are not arbitrary, but built built from an environment, which represents the knowledge built so far by an outside observer. We give first the definition of environmental bisimilarity for the relaxed semantics. We then discuss a definition of environmental bisimilarity which we can prove complete for the original semantics. This section summarizes the results in [7].

5.1 Definition for the Relaxed Semantics

Environmental bisimulations use an environment $\mathcal{E}$ to accumulate knowledge about two tested terms. For the λ-calculus [22], $\mathcal{E}$ records the values (v_0, v_1) the tested terms reduce to, if they exist. We can then compare v_0 and v_1 at any time by passing them arguments built from $\mathcal{E}$. With the relaxed semantics of $\lambda_{\mathcal{S}}$, control stuck terms are also normal forms. To handle these, we allow environments to contain pairs of control stuck terms, and we test them by building pure contexts from $\mathcal{E}$. To build these testing arguments from $\mathcal{E}$, we define in Figure 4 two closures that generate respectively terms and evaluation contexts. Given a relation $\mathcal{R}$ on terms, we write $\dot{\mathcal{R}}$ for the term generating closure and $\ddot{\mathcal{R}}$ for the context generating closure. Even if $\mathcal{R}$ is defined only on closed terms, $\dot{\mathcal{R}}$ and $\ddot{\mathcal{R}}$ are defined on open terms and open contexts, respectively. In this section, we consider the restrictions of $\dot{\mathcal{R}}$ and $\ddot{\mathcal{R}}$ to respectively closed terms and closed contexts unless stated otherwise.

Formally, an environment $\mathcal{E}$ is a relation on normal forms which relates values with values and control stuck terms with control stuck terms; e.g., we define the identity environment $\mathcal{I}$ as $\{(t,t) \mid t \text{ is a normal form}\}$. An environmental relation $\mathcal{X}$ is a set of environments $\mathcal{E}$, and triples $(\mathcal{E}, t_0, t_1)$, where t_0 and t_1 are closed. We write $t_0\ \mathcal{X}_{\mathcal{E}}\ t_1$ as a shorthand for $(\mathcal{E}, t_0, t_1) \in \mathcal{X}$; roughly, it means that we test t_0 and t_1 with the knowledge $\mathcal{E}$. We define environmental bisimulation as follows.

Definition 5.1. A relation $\mathcal{X}$ is an environmental bisimulation if

1. $t_0\ \mathcal{X}_{\mathcal{E}}\ t_1$ implies:

(a) if $t_0 \to_{\mathrm{v}} t_0'$, then there exists t_1' such that $t_1 \to_{\mathrm{v}}^* t_1'$ and $t_0' \; \mathcal{X}_{\mathcal{E}} \; t_1'$;

(b) if t_0 is a normal form, then there exists a normal form t_1' of the same kind as t_0 such that $t_1 \to_{\mathrm{v}}^* t_1'$ and $\mathcal{E} \cup \{(t_0, t_1')\} \in \mathcal{X}$;

(c) the converse of the above conditions on t_1;

2. $\mathcal{E} \in \mathcal{X}$ implies:

(a) if $\lambda x.t_0 \; \mathcal{E} \; \lambda x.t_1$ and $v_0 \; \dot{\mathcal{E}} \; v_1$, then $t_0\{v_0/x\} \; \mathcal{X}_{\mathcal{E}} \; t_1\{v_1/x\}$;

(b) if $E_0[\mathcal{S}k.t_0] \; \mathcal{E} \; E_1[\mathcal{S}k.t_1]$ and $E_0' \; \ddot{\mathcal{E}} \; E_1'$, then $\langle t_0\{\lambda x.\langle E_0'[E_0[x]]\rangle/k\}\rangle \; \mathcal{X}_{\mathcal{E}} \; \langle t_1\{\lambda x.\langle E_1'[E_1[x]]\rangle/k\}\rangle$ for a fresh x.

Environmental bisimilarity, written $\approx$, is the largest environmental bisimulation. To prove that two terms t_0 and t_1 are equivalent, we want to relate them without any predefined knowledge, i.e., we want to prove that $t_0 \approx_\emptyset t_1$ holds; we also write $\mathbb{E}$ for $\approx_\emptyset$. The relation $\mathbb{E}$ will be the candidate to characterize contextual equivalence.

The first part of the definition makes the bisimulation game explicit for t_0 and t_1, while the second part focuses on environments $\mathcal{E}$. If t_0 is a normal form, then t_1 has to evaluate to a normal form of the same kind, and we extend the environment with the newly acquired knowledge. We then compare values in $\mathcal{E}$ (clause (2a)) by applying them to arguments built from $\mathcal{E}$, as in the λ-calculus [22]. Similarly, we test stuck terms in $\mathcal{E}$ by putting them within contexts $\langle E_0'\rangle$, $\langle E_1'\rangle$ built from $\mathcal{E}$ (clause (2b)) to trigger the capture. This is similar to the way we test values and stuck terms with applicative bisimilarity (Section 4), except that applicative bisimilarity tests both values or stuck terms with the same argument or context. Using different entities (as in Definition 5.1) makes bisimulation proofs harder, but it simplifies the proof of congruence of the environmental bisimilarity.

The relation we obtain is sound and complete w.r.t. contextual equivalence.

Theorem 5.2. *We have* $\mathbb{E} = \mathbb{C}$.

We now give some examples showing how the notion of environmental bisimulation can be used.

Example 5.3 (double reset)**.** We have $\langle\langle t\rangle\rangle \; \mathbb{E} \; \langle t\rangle$, because the relation

$$\{(\emptyset, \langle\langle t\rangle\rangle, \langle t\rangle)\} \cup \{(\mathcal{E}, t, t) \mid \mathcal{E} \subseteq \mathcal{I}\} \cup \{\mathcal{E} \mid \mathcal{E} \subseteq \mathcal{I}\}$$

is a big-step environmental bisimulation. Indeed, we know that $\langle\langle t\rangle\rangle \Downarrow_{\mathrm{v}} v$ iff $\langle t\rangle \Downarrow_{\mathrm{v}} v$, so we have to consider environments $\mathcal{E}$ of the form (v, v). Then, testing these $\mathcal{E}$ suppose to take $\lambda x.t \; \mathcal{E} \; \lambda x.t$ and some arguments $v_0 \; \dot{\mathcal{E}} \; v_1$, and relate $t\{v_0/x\}$ with $t\{v_1/x\}$. Since the terms related by $\mathcal{E}$ are the same, we have in fact $v_0 = v_1$, so we have to relate $t\{v_0/x\}$ with itself, hence the second set in the definition of the bisimulation.

Example 5.4 (Turing's combinator)**.** Proving that Turing's combinator Θ is bisimilar to its variant $\Theta_{\mathcal{S}} \stackrel{\mathsf{def}}{=} \langle \theta\, \mathcal{S}k.k\, k \rangle$ using the basic definition of environmental bisimulation is harder than with applicative bisimulation (Example 4.6). We remind that

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

Therefore, we have to put (v_0, v_1) in an environment $\mathcal{E}$. When we then test v_0 and v_1, we use arguments v_0' and v_1' such that $v_0'\ \dot{\mathcal{E}}\ v_1'$, and we compare $v_0'\ (\lambda z.\theta\ \theta\ v_0'\ z)$ with $v_1'\ (\lambda z.(\lambda x.\langle \theta\, x \rangle)\ (\lambda x.\langle \theta\, x \rangle)\ v_1'\ z)$. Because we have two different terms v_0' and v_1', we can no longer do a case analysis as suggested in Example 4.6. To conclude with environmental bisimulation, we need bisimulation up to context (see [7]).

5.2 Environmental Relations for the Original Semantics

The bisimilarities introduced so far are sound and complete w.r.t. the contextual equivalence $\mathbb{C}$ of the relaxed semantics, but only sound w.r.t. the contextual equivalence $\mathbb{P}$ of the original semantics (cf. Proposition 2.12). We now propose a definition of environmental bisimulation adapted to delimited terms (but defined on all terms, like $\mathbb{P}$). Because control stuck terms cannot be obtained from the evaluation of a delimited term, environments $\mathcal{E}$ henceforth relate only values. Similarly, we write $\mathcal{R}^{\,\mathrm{v}}$ for the restriction of a relation $\mathcal{R}$ on terms to pairs of closed values.

Definition 5.5. A relation $\mathcal{X}$ is a delimited environmental bisimulation if

1. if $t_0\ \mathcal{X}_{\mathcal{E}}\ t_1$ and t_0 and t_1 are not both delimited terms, then for all closed E_0, E_1 such that $E_0\ \ddot{\mathcal{E}}\ E_1$, we have $\langle E_0[t_0] \rangle\ \mathcal{X}_{\mathcal{E}}\ \langle E_1[t_1] \rangle$;
2. $p_0\ \mathcal{X}_{\mathcal{E}}\ p_1$ implies
   (a) if $p_0 \to_{\mathrm{v}} p_0'$, then there exists p_1' such that $p_1 \to_{\mathrm{v}}^* p_1'$ and $p_0'\ \mathcal{X}_{\mathcal{E}}\ p_1'$;
   (b) if $p_0 \to_{\mathrm{v}} v_0$, then there exists v_1 such that $p_1 \to_{\mathrm{v}}^* v_1$, and $\{(v_0, v_1)\} \cup \mathcal{E} \in \mathcal{X}$;
   (c) the converse of the above conditions on p_1;
3. for all $\mathcal{E} \in \mathcal{X}$, if $\lambda x.t_0\ \mathcal{E}\ \lambda x.t_1$ and $v_0\ \dot{\mathcal{E}}\ v_1$, then $t_0\{v_0/x\}\ \mathcal{X}_{\mathcal{E}}\ t_1\{v_1/x\}$.

Delimited environmental bisimilarity, written $\simeq$, is the largest delimited environmental bisimulation. As before, the relation $\simeq_{\emptyset}$, also written $\mathbb{F}$, is candidate to characterize $\mathbb{P}$.

Clauses (2) and (3) of Definition 5.5 deal with delimited terms and environments in a classical way (as in plain λ-calculus). The problematic case is when relating terms t_0 and t_1 that are not both delimited terms (clause (1)). Indeed, one of them may be control stuck, and therefore we have to test them

within some contexts $\langle E_0 \rangle$, $\langle E_1 \rangle$ (built from $\mathcal{E}$) to potentially trigger a capture that otherwise would not happen. We cannot require both terms to be control stuck, as in clause (2b) of Definition 5.1, because a control stuck term can be equivalent to a term free from control effect. E.g., we will see that $v \mathbb{F} \mathcal{S}k.k\, v$, provided that $k \notin \mathsf{fv}(v)$.

The next proposition shows that $\mathbb{E}$ is more discriminative than $\mathbb{F}$.

Proposition 5.6. *We have* $\mathbb{E} \subseteq \mathbb{F}$.

A consequence of Proposition 5.6 is that we can use Definition 5.1 as a proof technique for $\mathbb{F}$. E.g., we have directly $\langle (\lambda x.t_0)\, \langle t_1 \rangle \rangle \mathbb{F} (\lambda x.\langle t_0 \rangle)\, \langle t_1 \rangle$, because $\langle (\lambda x.t_0)\, \langle t_1 \rangle \rangle \mathbb{E} (\lambda x.\langle t_0 \rangle)\, \langle t_1 \rangle$. The bisimilarity we obtain is sound and complete w.r.t. $\mathbb{P}$.

Theorem 5.7. *We have* $\mathbb{F} = \mathbb{P}$.

5.3 Examples

We illustrate the differences between $\mathbb{E}$ and $\mathbb{F}$, by giving some examples of terms related by $\mathbb{F}$, but not by $\mathbb{E}$. First, note that $\mathbb{F}$ relates non-terminating terms with stuck non-terminating terms.

Proposition 5.8. *We have* $\Omega \mathbb{F} \mathcal{S}k.\Omega$.

The relation $\{(\emptyset, \Omega, \mathcal{S}k.\Omega), (\emptyset, \langle E[\Omega] \rangle, \langle E[\mathcal{S}k.\Omega] \rangle), (\emptyset, \langle E[\Omega] \rangle, \langle \Omega \rangle)\}$ is a delimited bisimulation. Proposition 5.8 does not hold with $\mathbb{E}$ because Ω is not stuck.

As wished, $\mathbb{F}$ satisfies the only axiom of [17] not satisfied by $\mathbb{E}$.

Proposition 5.9. *If* $k \notin \mathsf{fv}(t)$, *then* $t \mathbb{F}^\circ \mathcal{S}k.k\, t$.

Consequently, $\mathbb{F}^\circ$ is complete w.r.t. $\equiv$.

Corollary 5.10. *We have* $\equiv \subseteq \mathbb{F}^\circ$.

As a result, we can use $\equiv$ (restricted to closed terms) as a proof technique for $\mathbb{F}$. E.g., the following equivalence can be derived from the axioms [17].

Proposition 5.11. *If* $k \notin \mathsf{fv}(t_1)$, *then* $(\lambda x.\mathcal{S}k.t_0)\, t_1 \mathbb{F} \mathcal{S}k.((\lambda x.t_0)\, t_1)$.

This equivalence does not hold with $\mathbb{E}$, because the term on the right is stuck, but the term on the left may not evaluate to a stuck term (if t_1 does not terminate).

6 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 $\mathbb{P}$ and $\mathbb{C}$), that we try to

$\curvearrowleft$		$\equiv$	$\mathbb{N}$	$\mathbb{A}$	$\mathbb{E}$	$\mathbb{F}$
relaxed semantics:	$\mathbb{C}$		$\subsetneq$	$=$	$=$	$\supsetneq$
original semantics:	$\mathbb{P}$	$\subsetneq$	$\subsetneq$	$\subsetneq$	$\subsetneq$	$=$

Figure 5: Relationships between the equivalences of $\lambda_{\mathcal{S}}$ (e.g., $\mathbb{N} \subsetneq \mathbb{C}$)

characterize with different kinds of bisimilarities (normal form $\mathbb{N}$, applicative $\mathbb{A}$, and environmental $\mathbb{E}$, $\mathbb{F}$). We also compare our relations to CPS equivalence $\equiv$, a relation which equates terms with $\beta\eta$-equivalent CPS translations. The relationship between all these relations is summarized in Figure 5.

When comparing term equivalence proofs, we can see that each bisimulation style has its strengths and weaknesses. Normal form bisimulation arguably leads to the simplest proofs of equivalence on average, as it does not contain any quantification over arguments or testing contexts in its definition. For example, the β_Ω axiom can be easily proved using normal form bisimulation (Proposition 3.8); the proof with applicative bisimulation is much more complex (Proposition 4.10), and we do not know how to prove it with environmental bisimulation.

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 Proposition 3.5. Besides, normal form bisimulation operates on open terms by definition, which requires to consider an extra normal form (open stuck terms) in the bisimulation proofs. Applicative and environmental bisimulations do not have these issues: their corresponding bisimilarities are complete, and they operate on closed terms. As a result, the proof that $\langle\langle t \rangle\rangle$ is equivalent to $\langle t \rangle$ is shorter with applicative bisimulation than with normal form bisimulation (compare Example 3.2 and Example 4.5). This is also true, e.g., for the $\langle\cdot\rangle_{\mathtt{lift}}$ axiom (compare Proposition 3.7 and 4.9).

To summarize, to prove that two given terms are equivalent, we would suggest to first try to use normal form bisimulation, and if it fails, try applicative bisimulation, and next, environmental bisimulation. This strategy holds for the relaxed as well as the original semantics, except if one wants to relate, e.g., a control stuck term with a value (like with the $\mathcal{S}_{\mathtt{elim}}$ axiom): it is possible only with the environmental bisimulation for the original semantics.

References

[1] Samson Abramsky and C.-H. Luke Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, 105:159–267, 1993. 1, 2, 11

[2] Kenichi Asai and Yukiyoshi Kameyama. Polymorphic delimited continuations. In Zhong Shao, editor, *APLAS'07*, number 4807 in LNCS, pages 239–254, Singapore, December 2007. Springer-Verlag. 5

[3] Vincent Balat, Roberto Di Cosmo, and Marcelo P. Fiore. Extensional normalisation and type-directed partial evaluation for typed lambda calculus with sums. In Xavier Leroy, editor, *POPL'04*, SIGPLAN Notices, Vol. 39, No. 1, pages 64–76, Venice, Italy, January 2004. ACM Press. 1

[4] Małgorzata Biernacka, Dariusz Biernacki, and Olivier Danvy. An operational foundation for delimited continuations in the CPS hierarchy. *Logical Methods in Computer Science*, 1(2:5):1–39, November 2005. 5

[5] Dariusz Biernacki and Sergueï Lenglet. Applicative bisimulations for delimited-control operators. In Lars Birkedal, editor, *FOSSACS'12*, number 7213 in LNCS, pages 119–134, Tallinn, Estonia, March 2012. Springer-Verlag. 2, 12

[6] Dariusz Biernacki and Sergueï Lenglet. Normal form bisimulations for delimited-control operators. In Tom Schrijvers and Peter Thiemann, editors, *FLOPS'12*, number 7294 in LNCS, pages 47–61, Kobe, Japan, May 2012. Springer-Verlag. 2, 8

[7] Dariusz Biernacki and Sergueï Lenglet. Environmental bisimulations for delimited-control operators. In Chung-chieh 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, December 2013. Springer. 2, 16, 18

[8] Olivier Danvy and Andrzej Filinski. A functional abstraction of typed contexts. DIKU Rapport 89/12, DIKU, Computer Science Department, University of Copenhagen, Copenhagen, Denmark, July 1989. 4, 10

[9] Olivier Danvy and Andrzej Filinski. Abstracting control. In Wand [25], pages 151–160. 1, 2, 5

[10] R. Kent Dybvig, Simon Peyton-Jones, and Amr Sabry. A monadic framework for delimited continuations. *Journal of Functional Programming*, 17(6):687–730, 2007. 5

[11] Matthias Felleisen. The theory and practice of first-class prompts. In Jeanne Ferrante and Peter Mager, editors, *POPL'88*, pages 180–190, San Diego, California, January 1988. ACM Press. 1

[12] Andrzej Filinski. Representing monads. In Hans-J. Boehm, editor, *POPL'94*, pages 446–457, Portland, Oregon, January 1994. ACM Press. 1, 5

[13] Andrew D. Gordon. Bisimilarity as a theory of functional programming. *Theoretical Computer Science*, 228(1-2):5–47, 1999. 12

[14] Andrew D. Gordon and Gareth D. Rees. Bisimilarity for a first-order calculus of objects with subtyping. In Guy L. Steele Jr., editor, *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Programming*

Languages, pages 386–395, St. Petersburg Beach, Florida, January 1996. ACM Press. 12

[15] Robert Hieb, R. Kent Dybvig, and Claude W. Anderson, III. Subcontinuations. *Lisp and Symbolic Computation*, 5(4):295–326, December 1993. 1

[16] Yukiyoshi Kameyama. Axioms for control operators in the CPS hierarchy. *Higher-Order and Symbolic Computation*, 20(4):339–369, 2007. 5

[17] Yukiyoshi Kameyama and Masahito Hasegawa. A sound and complete axiomatization of delimited continuations. In Olin Shivers, editor, *ICFP'03*, SIGPLAN Notices, Vol. 38, No. 9, pages 177–188, Uppsala, Sweden, August 2003. ACM Press. 3, 6, 19

[18] Søren B. Lassen. Eager normal form bisimulation. In Prakash Panangaden, editor, *LICS'05*, pages 345–354, Chicago, IL, June 2005. IEEE Computer Society Press. 2, 7, 8, 9, 10

[19] Sergueï Lenglet, Alan Schmitt, and Jean-Bernard Stefani. Howe's method for calculi with passivation. In Mario Bravetti and Gianluigi Zavattaro, editors, *CONCUR'09*, number 5710 in LNCS, pages 448–462, Bologna, Italy, July 2009. Springer. 13

[20] James H. Morris. *Lambda Calculus Models of Programming Languages*. PhD thesis, Massachusets Institute of Technology, 1968. 1

[21] Davide Sangiorgi. The lazy lambda calculus in a concurrency scenario. In Andre Scedrov, editor, *LICS'92*, pages 102–109, Santa Cruz, California, June 1992. IEEE Computer Society. 8

[22] Davide Sangiorgi, Naoki Kobayashi, and Eijiro Sumii. Environmental bisimulations for higher-order languages. *ACM Transactions on Programming Languages and Systems*, 33(1):1–69, January 2011. 2, 16, 17

[23] Davide Sangiorgi and David Walker. *The Pi-Calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001. 1

[24] Dorai Sitaram and Matthias Felleisen. Reasoning with continuations II: Full abstraction for models of control. In Wand [25], pages 161–175. 1

[25] Mitchell Wand, editor. *Proceedings of the 1990 ACM Conference on Lisp and Functional Programming*, Nice, France, June 1990. ACM Press. 21, 22