How to Prove Similarity a Precongruence in Non-Deterministic Call-by-Need Lambda Calculi

Matthias Mann and Manfred Schmidt-Schauß

Institut für Informatik
Johann Wolfgang Goethe-Universität
Postfach 11 19 32
D-60054 Frankfurt, Germany
schauss@cs.uni-frankfurt.de

Technical Report Frank-22
Research Group for Artificial Intelligence and Software Technology,
Institut für Informatik,
J.W.Goethe-Universität Frankfurt,

22.1.2006

Abstract. Extending the method of Howe, we establish a large class of untyped higher-order calculi, in particular such with call-by-need evaluation, where similarity, also called applicative simulation, can be used as a proof tool for showing contextual preorder. The paper also demonstrates that Mann's approach using an intermediate "approximation" calculus scales up well from a basic call-by-need non-deterministic lambda-calculus to more expressive lambda calculi. I.e., it is demonstrated, that after transferring the contextual preorder of a non-deterministic call-by-need lambda calculus to its corresponding approximation calculus, it is possible to apply Howe's method to show that similarity is a precongruence. The transfer is not treated in this paper.
The paper also proposes an optimization of the similarity-test by cutting off redundant computations.
Our results also applies to deterministic or non-deterministic call-by-value lambda-calculi, and improves upon previous work insofar as it is proved that only closed values are required as arguments for similarity-testing instead of all closed expressions.

Keywords: Similarity –Precongruence –Contextual Equivalence –Non-determinism –Call-by-need Lambda Calculus

1 Introduction

1.1 Introduction and Motivation

Higher-order calculi, in particular extended lambda-calculi, are of increasing importance as a foundation for programming languages, parallel programming lan-

guages, and concurrent modeling languages and their respective (operational) semantics. Functional programming languages, strict and non-strict ones, are rather close to these calculi and often directly use results from the field of lambda calculi and their semantic properties, e.g. by gaining insight into allowed program transformations and optimizations and into the correctness of program analyses. This also holds for non-deterministic lambda calculi, which are treated in papers in call-by-name, call-by-value and call-by-need variants, and also using different non-deterministic primitives. We address call-by-need variants of non-deterministic lambda calculi which in our opinion are most appropriate for applications in programming languages, since sharing avoids unnecessary copying of expressions having several potential values, in particular `amb`-expressions.

Equality of expressions and correctness of program transformations can be defined in different ways. Of high practical relevance are contextual preorder and contextual equivalence, defined as $s \lesssim_c t$ for two expressions, iff for all contexts C (i.e. programs), whenever $C[s]$ evaluates to a value, then also $C[t]$ evaluates to a value, and $s \simeq_c t$, iff $s \lesssim_c t$ and $t \lesssim_c s$. This is the maximal equality and the right one for practical applications of deterministic calculi. It is also used in non-deterministic calculi, though there are also other variants [MSC99], which also take non-termination into account, or employ combinations of may- and must-convergence. In this paper we only consider $\lesssim_c$ and $\simeq_c$ based on the existence of an evaluation to a value.

The disadvantage of contextual preorder is that it is hard to determine whether $s \lesssim_c t$ for given expressions s, t holds. Of course it is an undecidable question, nevertheless, it is of high importance to find tools that are able to prove $s \lesssim_c t$ for as much pairs of terms as possible. There are different methods for providing such tools. One is to restrict the number of necessary contexts by providing context lemmata [Mil71]. Another direction is to use applicative (bi)simulation, or similarity [Abr90]. This works as follows: there are no contexts involved, just reduce the given terms s, t at top level to a value. If the values are constructed by data constructors, then compare the subexpressions; if the values are abstractions, then test them applied to all possible arguments, asf. This has a chance to terminate for interesting classes of term pairs. The correctness proof for this tool is nontrivial and was given for a minimal calculus by Abramsky [Abr90], and then for large classes of lazy calculi by Howe ([How89,How96]). There is further work to extend the scope of this result to different classes of calculi (see [KvOvR93,San97,Gor99]).

The first result for non-deterministic call-by-need calculi is in [Man05b,Man05a] showing that in a minimal call-by-need lambda calculus L similarity is a sufficient criterion for contextual equivalence. The new method is to consider a lambda calculus L_A, which is an approximation variant of L with the same contextual equivalence but values instead of WHNFs as successful end-points of reductions, and to define similarity within L_A to which Howe's method can be applied. This requires a slight modification of Howe's method by restricting the set of substitutable terms to be pseudo-values, i.e. values or $\circledcirc$, which can be seen as a representative of non-terminating terms. Note, however, that this paper does not

treat the relation between a lambda calculus L and its approximating lambda calculus L_A, which is beyond the scope of this paper.

This paper develops Mann's method further to a larger class of calculi, permitting constructors, case-expressions and the corresponding reductions, and also different non-deterministic primitives. This will broaden the scope of the proof tool of similarity to other variants of non-deterministic call-by-need calculi, provided an appropriate approximation calculus is defined and proved to be adequate.

The main contribution of this paper is to define schematic higher-order computation systems (SHOCS), based on small-step reduction. Covering a large class of the aforementioned types of calculi, we provide a concise proof along the lines of Howe that similarity implies contextual preorder. A further contribution of this paper is that the restriction to pseudo-values also strengthens the original results on the relation between similarity and contextual equivalence insofar as only closed values as arguments have to be tested.

1.2 Applying the Results of this Paper to Call-By-Need Calculi

For deterministic or non-deterministic call-by-need lambda-calculi, the paper provides an interface for the similarity-check, which is free from internal conditions on the precongruence candidate relation (for more information see section 3). Assume such a lambda-calculus L that allows a non-recursive `let` and has a normal-order reduction as evaluation. Usually, there will be rules for shuffling the `let`-environments to the top, e.g.: $((\texttt{let}\ x = s_x\ \texttt{in}\ t)\ r) \to (\texttt{let}\ x = s_x\ \texttt{in}\ (t\ r))$, and $(\texttt{let}\ x = (\texttt{let}\ y = s_y\ \texttt{in}\ s_x)\ \texttt{in}\ t) \to (\texttt{let}\ x = s_x\ \texttt{in}\ (\texttt{let}\ y = s_y\ \texttt{in}\ t))$. Experience shows that these rules are in conflict with Howe's method, so our method proposes the following circumvention. To apply our result, an intermediate approximation calculus L_A has to be constructed, where one must assure that $s \leq_{c,L} t \iff s \leq_{c,L_A} t$ for all L-terms s, t. Moreover, the rules for let-shuffling above have to be eliminated in L_A, since their format does not fit the reductions of a SHOCS. The calculus L_A must first evaluate the term s in the term $(\texttt{let}\ x = s\ \texttt{in}\ t)$, giving a pseudo-value s_v or the constant $\circledcirc$ indicating a stopped evaluation, and then copying s_v, giving $t[s_v/x]$. Note that in a lambda-calculus with λ, `case` and constructors, a pseudo-value is an expression constructed from constructors, abstractions, perhaps variables, and the symbol $\circledcirc$, where unevaluated expressions are only allowed within abstractions. The interface also expects that a symbol $\circledcirc$ is introduced in L_A that has no reductions, and that a stop-rule $t \to \circledcirc$ is permitted for all terms t that are not already pseudo-values. The similarity-check has to be applied for terms in this calculus L_A, and the results can be carried back into the original calculus L. This transfer of the problem into L_A comes at the price that the reduction in L_A is non-deterministic at a rather high degree, since in general several possibilities for reductions have to be taken into account, and since there is no notion of a normal-order reduction or a standard reduction. The additional non-determinism can be reduced by restricting the reduction such that redundant pseudo-values need not be reached (see section 7).

The similarity check (see algorithm 4.7) is a procedure as follows: To check $s \lesssim_b t$ for closed terms s, t, compute the set of pseudo-values $\texttt{ans}(s), \texttt{ans}(t)$ that can be reached from s or t by reduction in L_A, respectively. If $\texttt{ans}(s) \subseteq \texttt{ans}(t)$, then $s \lesssim_b t$ holds. Otherwise, for every $\theta(s'_1, \ldots, s'_n) \in \texttt{ans}(s)$, there must exist a term $\theta(t'_1, \ldots, t'_n)) \in \texttt{ans}(t)$, such that for all substitutions σ instantiating closed pseudo-values (i.e. closed values or the constant $\circledcirc$) and for all i: $\sigma(s'_i) \lesssim_b \sigma(t'_i)$. Clearly, this test is nonterminating in general. Also the set of closed pseudo-values is infinite, but in many cases the structure of the terms is simple enough, so that at least the computation of $\texttt{ans}(\cdot)$ terminates. Often, a generalization or an abstraction or other ad-hoc methods may help to keep reasoning finite.

The permitted rules are e.g. copy-reductions, of which beta-reduction is an instance: $(\lambda x.s)\ v \to s[v/x]$, and projection rules, where the amb-rules are particular instances: $\texttt{amb}\ v\ t \to v$ and $\texttt{amb}\ t\ v \to v$, if v is a value in the language.

As an example, using the similarity check, is is easy to prove that $\texttt{amb}\ \Omega\ t\ \simeq_b\ t$. This easily follows from the fact that both terms have the same set of values as result. Hence by our result that similarity implies contextual preorder, also $\texttt{amb}\ \Omega\ t \simeq_c t$ holds.

1.3 Applying the Results of this Paper to Call-By-Value Calculi

For a deterministic or non-deterministic call-by-value calculus, the application of our results is similar as for call-by-need, however, a bit simpler and more direct. It is not necessary to use an intermediate approximation calculus nor is it necessary to add a $\circledcirc$-symbol. The call-by-value calculus must reduce to values without top `let`-environments. E.g. a list of numbers or a list of abstractions may be a value, but not a list of terms that are `let`-expressions. The test-procedure for similarity has to take into account all closed values, which is an improvement over previous work.

1.4 Structure of this Paper

The goal of this paper is to prove that (applicative) similarity implies contextual preorder for a large class of lambda calculi, with an emphasis on call-by-need non-deterministic lambda calculi, and therefore to extend the techniques of Howe. The development of the sections, definitions, lemmas and proofs is a bit more general than really necessary for the final result, the gain is that it provides more insight into the mechanism of Howe's proof technique and also gives different entry points if someone wants to extend the techniques for calculi not covered in this paper. A principle is to keep all the parts rather general in order to ease a reuse of the different steps in the development.

The other inclusion, i.e., that contextual preorder implies applicative similarity, is not true in general. For non-deterministic call-by-need calculi see [Man05b] for a counterexample. There are some (natural) criteria for equivalence in deterministic calculi (see Theorems 6.14 and 6.15).

The structure of the paper is as follows. In section 2 the abstract syntax for higher-order calculi is introduced following Howe ([How89,How96]). In section 3

the first extension to Howe is to restrict the substitutable terms, called pseudo-values, and to adapt the technique by defining the open extension of closed relations w.r.t. pseudo-values. From now on, the pseudo-values are a parameter for the hole development. In subsection 3.2 the so-called precongruence candidate relation is defined and a criterion for a relation η^o on terms to be a precongruence is proved. In section 4, a big-step operational semantics is considered, and necessary conditions are made explicit in the definition of a higher-order computation system. The similarity (applicative simulation) $\lesssim_b$ and the contextual preorder $\lesssim_c$ are defined on this basis. The most important condition implying that ${\lesssim_b}^o \subseteq \lesssim_c$ is the stability criterion in Theorem 4.15. However, at this level of abstraction, it is not possible to verify stability, hence in section 5, a small-step reduction and further conditions are specified. These conditions are summarized in the definition of a HOCS+SR (Def. 5.2). The conditions allow to show the right-stability part of the stability criteria. In section 6 we further specialize and define several formats for rules, which comprise all the reductions in an approximation calculus of a call-by-need non-deterministic calculus with (non-recursive) `let`, and also in call-by-value calculi with strict `let`. This is summarized in the definition of schematic higher order computation systems (SHOCS) (Def. 6.11). For SHOCS, it is then possible to show the left-stability for every kind of reduction, i.e., the remaining part of the stability criteria. This finally shows that in every lambda calculus that matches the definition of a SHOCS, similarity implies contextual preorder. In section 7, we show that it is permitted to replace the non-deterministic reduction relation defined in section 6 by an optimized reduction relation that may be more deterministic and also avoids reductions to redundant values. Section 8 contains some example calculi, and how our results may be applied to example terms.

2 Higher-Order Computation Language

The presentation of higher-order abstract syntax follows [How96].

Definition 2.1 (Higher-Order Computation Language). *Let $\mathcal{L} = (O, \alpha)$ be a signature, where O consists of symbols, called operators, and*

$$\alpha(\tau) \in \{\, \langle k_1, \ldots, k_n \rangle \mid \forall 1 \leq i \leq n \in \mathbb{N} : k_i \in \mathbb{N}_0 \,\} \text{ for every } \tau \in O.$$

Then $\mathcal{L}$ is called a higher-order computation language *(HOCL for short) and O its set of* operators *with α denoting their respective* arity.

Definition 2.2 (Terms and Operands). *Let $\mathcal{L} = (O, \alpha)$ be a higher-order computation language and V be a countable set of variables. Then the sets ${}^i\mathcal{T}(\mathcal{L})$ are inductively defined as follows:*

- $V \subseteq {}^0\mathcal{T}(\mathcal{L})$
- *If $t \in {}^0\mathcal{T}(\mathcal{L})$ and $x_1, \ldots, x_n \in V$ are distinct then $x_1, \ldots, x_n.t \in {}^n\mathcal{T}(\mathcal{L})$*
- *If $\tau \in O$ with arity $\alpha(\tau) = \langle k_1, \ldots, k_n \rangle$ and $t_j \in {}^{k_j}\mathcal{T}(\mathcal{L})$ for $j \in \{1, \ldots, n\}$ then $\tau(t_1, \ldots, t_n) \in {}^0\mathcal{T}(\mathcal{L})$*

The elements of ${}^n\mathcal{T}(\mathcal{L})$ *are called* operands*, the elements of* ${}^n\mathcal{T}(\mathcal{L})$ *for* $n > 0$ *are called* higher-order operands*, also denoted as* ${}^{(\geq 1)}\mathcal{T}(\mathcal{L})$*, and the elements of* $\mathcal{T}(\mathcal{L}) = {}^0\mathcal{T}(\mathcal{L})$ *are called* terms *of the language* $\mathcal{L}$*.*

Only the construct $x_1, \ldots, x_n.t$ can bind variables. Free and bound variables are defined as usual, $\mathcal{FV}(t)$ denotes the set of *free variables* of an operand t. An operand t is *closed* if all of its variables are bound, i.e. $\mathcal{FV}(t) = \emptyset$, otherwise it is called *open*. The set of closed terms will be referred to as $\mathcal{T}_0(\mathcal{L})$. Tuples of operands are sometimes denoted as $\overline{t}$, or $\overline{t_i}$. If the language $\mathcal{L}$ is clear from context we omit the parameter in the notation. Note that the abstract syntax has an implicit typing: whenever an operator τ and operands $\overline{a}_i$ are mentioned in some context, then $\tau(\overline{a}_i)$ is supposed to form a valid term.

The expression $s[t/x]$ stands for the application of the capture-free substitution of t for x. This notation is also used for tuples of terms and variables. We will also write $\{x \mapsto t\}$ for the substitution $[t/x]$ when it is convenient. In the following, we use the symbol $\equiv$ to denote syntactic equivalence up to renaming of bound variables, i.e. terms and operands will be considered syntactically equal modulo alpha-renaming. In particular for higher-order operands, this means that $\overline{x}.s \equiv \overline{y}.t$ holds if there are fresh variables $\overline{z}$ such that the syntactic equivalence $s[\overline{z}/\overline{x}] \equiv t[\overline{z}/\overline{y}]$ is true.

Variable Convention 1 *Throughout this paper, whenever some term* t *is referred to, the bound variables are chosen to be distinct from each other and the free variables. Furthermore, this convention extends to sets of terms as well as terms which result from other terms, e.g., by transformations.*

Example 2.3. With $O = \{\lambda, \texttt{let}, @, \texttt{Cons}, \texttt{Nil}, \texttt{case}\}$ a simple `let`-language with lists and a case is declared. The arities of $\lambda, @, \texttt{let}$ are $\alpha(\lambda) = \langle 1 \rangle$, $\alpha(@) = \langle 0, 0 \rangle$, $\alpha(\texttt{let}) = \langle 1, 0 \rangle$, $\alpha(\texttt{Cons}) = \langle 0, 0 \rangle$, $\alpha(\texttt{Nil}) = \langle\rangle$, and $\alpha(\texttt{case}) = \langle 0, 2, 0 \rangle$. A term like `case x of (Cons z1 z2) -> y; Nil -> Nil` would be expressed as $\texttt{case}(x, z_1 z_2.y, \texttt{Nil})$. For call-by-value calculi, there is a slight modification explained in Example 6.3.

The notion of a *context* will be introduced. Roughly, a context C is a term with a hole [ ] and $C[s]$ stands for the resulting term where s has been plugged into the hole of C. Contexts are defined analogously to Definition 2.2 where [ ] denotes the empty context.

Definition 2.4 (Contexts). *Let* $\mathcal{L}$ *be a HOCL and* $\mathcal{T}(\mathcal{L})$ *its set of terms. Then the set* $\mathcal{C}(\mathcal{L}) = {}^0\mathcal{C}(\mathcal{L})$ *of contexts over* $\mathcal{L}$ *is inductively defined as follows.*

- $[\,] \in {}^0\mathcal{C}(\mathcal{L})$
- *If* $t \in {}^0\mathcal{C}(\mathcal{L})$ *and* $x_1, \ldots, x_n \in V$ *are distinct then* $x_1, \ldots, x_n.t \in {}^n\mathcal{C}(\mathcal{L})$
- *If* $\tau \in O$ *has arity* $\alpha(\tau) = \langle k_1, \ldots, k_n \rangle$ *and* $C_i \in {}^{k_i}\mathcal{C}(\mathcal{L})$ *is true for some* $1 \leq i \leq n$ *as well as* $t_j \in {}^{k_j}\mathcal{T}(\mathcal{L})$ *for all* $j \in \{1, \ldots, i-1, i+1, \ldots, n\}$ *then* $\tau(t_1, \ldots, t_{i-1}, C_i, t_{i+1}, \ldots, t_n) \in {}^0\mathcal{C}(\mathcal{L})$ *holds.*

We may omit the language $\mathcal{L}$ whenever there is no risk of confusion and write $\mathcal{C}$ for the set of all such single-hole contexts. We sometimes also consider *multi-contexts*, i.e., contexts with multiple, distinguishable holes. If C, D are contexts then $C[D]$ denotes the context resulting from inserting D into C's hole. The essential feature of contexts is to possibly capture free variables of the term plugged into the hole. Therefore, contexts are *not* identified up-to renaming of bound variables.

Example 2.5. Regard the HOCL of example 2.3 again. Then $\texttt{let}(x.[\,], t)$ and $\texttt{let}(y.[\,], t)$ denote different contexts. Also $\lambda(x.@(s, [\,]))$ is a context.

3 Open Extension and the Precongruence Candidate

In this section we will develop the method for dealing with relations, open extensions and the so-called precongruence candidate introduced by Howe [How89,How96]. In this section we treat properties of a higher-order language equipped with a set of pseudo-values $\mathcal{PV} \subseteq \mathcal{T}$, which represents just the terms that may be copied and substituted. We require the following assumption.

Assumption 3.1. There is a set of *pseudo-values* $\mathcal{PV} \subseteq \mathcal{T}$. The condition $\mathcal{PV} \cap \mathcal{T}_0 \neq \emptyset$ must hold.

Definition 3.2. *A relation $\upsilon \subseteq \mathcal{T} \times \mathcal{T}$ is* admissible*, iff it is invariant under variable renamings, i.e. if for every variable permutation $\rho : s \upsilon t \iff \rho(s) \upsilon \rho(t)$.*

In this paper we tacitly assume that all relations on terms and operands are admissible.

Definition 3.3. *In general, a relation on higher-order operands is an extension of an (admissible) relation υ defined as follows*

$$\overline{x}.s \upsilon \overline{y}.t \stackrel{def}{\iff} s[\overline{z}/\overline{x}] \upsilon t[\overline{z}/\overline{y}] \text{ for fresh and different variables } \overline{z} \qquad (3.1)$$

For sequences of operands we use $\overline{s_i} \upsilon \overline{t_i} \stackrel{def}{\iff} \forall i : s_i \upsilon t_i$.

Definition 3.4. *For $\upsilon \subseteq \mathcal{T}^2$ a relation, $(\upsilon)_0 \stackrel{def}{=} \upsilon \cap {\mathcal{T}_0}^2$ defines its* restriction to closed terms*.*

Note that the operator $(\cdot)_0$ is monotone on relations.

3.1 Preorders and the Open Extension

A *preorder* is a reflexive and transitive relation and an equivalence relation is a symmetric preorder. A relation $\upsilon \subseteq \mathcal{T}^2$ is *compatible (with contexts)* if for every context $C \in \mathcal{C}$ from $s \upsilon t$ also $C[s] \upsilon C[t]$ follows. A *precongruence* is a compatible preorder, and a *congruence* is a compatible equivalence relation. A

relation $\upsilon \subseteq \mathcal{T}^2$ is said to be *operator-respecting* (see also [How89]), if and only if $\overline{a}_i \ \upsilon \ \overline{b}_i$ implies $\tau(\overline{a}_i) \ \upsilon \ \tau(\overline{b}_i)$ for all operands a_i, b_i and operators $\tau \in O$. By induction on the structure of contexts and due to our admissibility assumption for relations, it is easily shown that this implies compatibility. If the relation is transitive, then also the reverse holds.

A substitution σ with range $\mathcal{PV}$ will be called *pseudo-valued substitution* or $\mathcal{PV}$-substitution for short. The set of $\mathcal{PV}$-substitutions is denoted as $\mathcal{PVS}$. For an open term t a substitution σ is *closing* if $\sigma(t)$ is closed.

Definition 3.5. *Let $\eta \subseteq \mathcal{T}_0 \times \mathcal{T}_0$ be a relation on closed terms.*
Then for $s, t \in \mathcal{T}^2$: $s \ \eta^o \ t$, if and only if $\sigma(s) \ \eta \ \sigma(t)$ for all pseudo-valued closing substitutions σ holds.

It is obvious that the open extension of a relation on closed terms is admissible. It is also clear that $s \ \eta^o \ t$ implies $\rho(s) \ \eta^o \ \rho(t)$ for every pseudo-valued substitution $\rho = \{x \mapsto r\}$. Moreover, η^o is reflexive (transitive, symmetric, respectively) whenever η is. Further obvious properties of the open extension are:

Lemma 3.6. *Let $\eta, \nu \subseteq {\mathcal{T}_0}^2$ be relations. Then*

- $(\eta^o)_0 = \eta$.
- $\nu \subseteq \eta \implies \nu^o \subseteq \eta^o$.
- $\eta \subseteq \eta^o$
- *if η is a preorder, then $\eta \circ \eta^o \subseteq \eta^o$ holds.*

3.2 The Precongruence Candidate

In this subsection we define the precongruence candidate $\widehat{\eta}$ on the basis of a preorder η. The candidate relation will be operator-respecting by construction, but it is not known whether it is transitive. Note that for non-transitive relations, operator-respecting and compatible are different notions. Theorem 3.13 exhibits conditions when the relation $\widehat{\eta}$ is already a precongruence. The proofs in this section are from [How89], but slightly reworked and generalized.

Definition 3.7 (Precongruence Candidate). *Let $\eta \subseteq \mathcal{T}_0 \times \mathcal{T}_0$ be a preorder. Then define its* precongruence candidate *$\widehat{\eta} \subseteq \mathcal{T} \times \mathcal{T}$ inductively by*

- *$x \ \widehat{\eta} \ b$ if $x \in V$ is a variable and $x \ \eta^o \ b$.*
- *$\tau(\overline{a}_i) \ \widehat{\eta} \ b$ if there exists $\overline{a}'_i$ such that $\overline{a}_i \ \widehat{\eta} \ \overline{a}'_i$ and $\tau(\overline{a}'_i) \ \eta^o \ b$ hold.*

The relation $\widehat{\eta}$ can be seen as an operator-respecting closure of η, however losing some properties. As Howe puts it: *$a \ \widehat{\eta} \ b$ if b can be obtained from a via one bottom-up pass of replacements of subterms by terms that are larger under η^o.* Note that $\tau \ \widehat{\eta} \ t \iff \tau \ \eta^o \ t$ if τ is a nullary operator and t is an arbitrary term. Note also that the precongruence candidate is admissible.

A technical, but helpful catalogue of properties of $\widehat{\eta}$ is the following lemma:

Lemma 3.8. *Let $\eta \subseteq {\mathcal{T}_0}^2$ be a preorder. Then*

1. $\widehat{\eta}$ *is reflexive*
2. $\widehat{\eta}$ *and* $\widehat{\eta}_0$ *are operator-respecting*
3. $\eta^o \subseteq \widehat{\eta}$
4. $\widehat{\eta} \circ \eta^o \subseteq \widehat{\eta}$

Proof. Note that the extension η^o is a preorder.

1. Let $a \in \mathcal{T}$ be an arbitrary but fixed term, then we show $a \;\widehat{\eta}\; a$ by induction on the structure of a:
   - If $a \in V$ is a variable, we have $a \;\widehat{\eta}\; a$ from the reflexivity of η^o and the base case of Definition 3.7.
   - If $a \equiv \tau(\overline{a}_i)$ for some operator τ and operands a_i, we have $a_i \;\widehat{\eta}\; a_i$ from the induction hypothesis and $\tau(\overline{a}_i) \;\eta^o\; \tau(\overline{a}_i)$ from the reflexivity of η^o. So, by Definition 3.7, we may compose this to $\tau(\overline{a}_i) \;\widehat{\eta}\; \tau(\overline{a}_i)$.
2. We assume $\overline{a}_i \;\widehat{\eta}\; \overline{b_i}$ and have to show $\tau(\overline{a}_i) \;\widehat{\eta}\; \tau(\overline{b_i})$ for an arbitrary but fixed operator τ. By reflexivity of η^o we have $\tau(\overline{b_i}) \;\eta^o\; \tau(\overline{b_i})$ and from Definition 3.7 we conclude $\tau(\overline{a}_i) \;\widehat{\eta}\; \tau(\overline{b_i})$.
   As a consequence if $\tau(\overline{a}_i), \tau(\overline{b_i}) \in \mathcal{T}_0$ are closed, we also have $\tau(\overline{a}_i) \;\widehat{\eta}_0\; \tau(\overline{b_i})$ from $\overline{a}_i \;\widehat{\eta}\; \overline{b_i}$, which implies that $\widehat{\eta}_0$ is operator-respecting too.
3. Assume $a \;\eta^o\; b$ for arbitrary but fixed $a, b \in \mathcal{T}$ and show $a \;\widehat{\eta}\; b$ by induction on the structure of a:
   - If $a \in V$ is a variable, we have $a \;\widehat{\eta}\; b$ directly from $a \;\eta^o\; b$ and the base case of Definition 3.7.
   - If $a \equiv \tau(\overline{a}_i)$ for some operator τ and operands a_i, we have $a_i \;\widehat{\eta}\; a_i$ from property 1, the reflexivity of $\widehat{\eta}$. By Definition 3.7 then, we may conclude $\tau(\overline{a}_i) \;\widehat{\eta}\; b$.
4. Assume $a \;\widehat{\eta}\; b$ and $b \;\eta^o\; c$, so according to Definition 3.7 we have to distinguish the following two cases:
   - a is a variable, then for $a \;\widehat{\eta}\; b$ also $a \;\eta^o\; b$ must hold and thus the proposition by transitivity of η^o and property 3.
   - For a of the form $\tau(\overline{a}_i)$ there is $\tau(\overline{a}'_i)$ with $\overline{a}_i \;\widehat{\eta}\; \overline{a}'_i$ and $\tau(\overline{a}'_i) \;\eta^o\; b$. By transitivity of η^o we also have $\tau(\overline{a}'_i) \;\eta^o\; c$ thus $\tau(\overline{a}_i) \;\widehat{\eta}\; c$.

The vital Substitution Lemma (see also [How96, Lemma 3.2]) holds also for our different notion of open extension:

Lemma 3.9 (Substitution Lemma). *Let $\eta \subseteq {\mathcal{T}_0}^2$ be a preorder on closed terms, $b, b' \in \mathcal{T}$ be terms and $a, a' \in \mathcal{PV}$ be pseudo-values. Then for every variable x the implication $(a \;\widehat{\eta}\; a' \wedge b \;\widehat{\eta}\; b') \implies b[a/x] \;\widehat{\eta}\; b'[a'/x]$ is true.*

Proof. By induction on the structure of the term b.

- If $b \in V$ is a variable then $b \;\widehat{\eta}\; b'$ is equivalent to $b \;\eta^o\; b'$ by definition. If $b \equiv x$ then $b[a'/x] \equiv a' \;\eta^o\; b'[a'/x]$ holds by the definition of the open extension. Because of $a \;\widehat{\eta}\; a'$ we also have $a \equiv b[a/x] \;\widehat{\eta}\; b[a'/x] \equiv a'$ and thus $a \;\widehat{\eta}\; b'[a'/x]$ is implied by composition, i.e., property 4 of Lemma 3.8.
  If b is a variable different from x, i.e., $b \equiv y$, then $y[a'/x] \;\eta^o\; b'[a'/x]$ is valid . Therefore $y[a/x] \equiv y \equiv y[a'/x] \;\eta^o\; b'[a'/x]$ holds and $y[a/x] \;\widehat{\eta}\; b'[a'/x]$ follows from property 3 of Lemma 3.8.

– If b is of the form $b \equiv \tau(\overline{b_i})$ then $\tau(\overline{b_i}) \ \widehat{\eta} \ b'$ means that there are $\overline{b'_i}$ such that $\overline{b_i} \ \widehat{\eta} \ \overline{b'_i}$ and $\tau(\overline{b'_i}) \ \eta^o \ b'$ holds. Hence, for all $i : b_i[a/x] \ \widehat{\eta} \ b'_i[a'/x]$ by the induction hypothesis. Moreover, $\tau(\overline{b_i})[a/x] \equiv \tau(\overline{b_i[a/x]}) \ \widehat{\eta} \ \tau(\overline{b'_i[a'/x]}) \equiv \tau(\overline{b'_i})[a'/x] \ \eta^o \ b'[a'/x]$, since $\widehat{\eta}$ is operator-respecting, and because of the definition of the open extension. This implies $b[a/x] \ \widehat{\eta} \ b'[a'/x]$ by composition (see Lemma 3.8 (4)).

Corollary 3.10. *Let $\eta \subseteq {\mathcal{T}_0}^2$ be a preorder on closed terms. Then $\widehat{\eta} \ \subseteq (\widehat{\eta}_0)^o$ holds.*

Proof. Assume arbitrary terms $s, t \in \mathcal{T}$ such that $s \ \widehat{\eta} \ t$ is true. Then by the Substitution Lemma $\sigma(s) \ \widehat{\eta} \ \sigma(t)$ holds for every pseudo-valued closing substitution σ, since $\widehat{\eta}$ is reflexive. Since $\sigma(s), \sigma(t) \in \mathcal{T}_0$ are closed even $\sigma(s) \ \widehat{\eta}_0 \ \sigma(t)$ is valid. This is satisfied for every pseudo-valued closing substitution σ, therefore with $s \ ((\widehat{\eta})_0)^o \ t$ the claim is shown.

Lemma 3.11. *Let $\eta \subseteq {\mathcal{T}_0}^2$ be a preorder on closed terms, $a, b \in \mathcal{T}$ be terms. Then $a \ \widehat{\eta} \ b$ if one of the following holds:*

– *$a \equiv x$ for a variable $x \in V$ and $x \ \eta^o \ b$*
– *$a \equiv \tau(\overline{a_i})$ for some $\tau \in O$, operands a_i, and there are operands a'_i such that $\overline{a_i} \ \widehat{\eta} \ \overline{a'_i}$ and $\tau(\overline{a'_i}) \ \eta^o \ b$ hold with $\mathcal{FV}(a'_i) \subseteq \bigcup_j \mathcal{FV}(a_j) \cup \mathcal{FV}(b)$ for all i.*

Proof. It is sufficient to show the second case.
By the definition of $\widehat{\eta}$, if $a \equiv \tau(\overline{a_i}) \ \widehat{\eta} \ b$, then there is some $\overline{a''_i}$, such that $\overline{a_i} \ \widehat{\eta} \ \overline{a''_i}$ and $\tau(\overline{a''_i}) \ \eta^o \ b$. Let $\{x_1, \ldots, x_n\} = \bigcup_j \mathcal{FV}(a''_j) \setminus (\bigcup_j \mathcal{FV}(a_j) \cup \mathcal{FV}(b))$. We construct the desired operands $\overline{a'_i}$ as $\overline{a'_i} := \overline{a''_i}[c/x_1, \ldots, c/x_n]$, where c is a closed pseudo-value, which exists due to Assumption 3.1. The relation $\tau(\overline{a'_i}) \ \eta^o \ b$ holds, since x_k does not occur free in b. The relation $\overline{a_i} \ \widehat{\eta} \ \overline{a'_i}$ follows from n successive applications (for each of the variables $x_1, \ldots, x_n$) of the Substitution Lemma 3.9 to $\overline{a_i} \ \widehat{\eta} \ \overline{a''_i}$ and $c \ \widehat{\eta} \ c$. The latter relation holds, since $\widehat{\eta}$ is reflexive by Lemma 3.8.

Corollary 3.12. *Let $\eta \subseteq {\mathcal{T}_0}^2$ be a preorder on closed terms, $\tau(\overline{a_i}), b \in \mathcal{T}_0$ be closed terms, such that $\tau(\overline{a_i}) \ \widehat{\eta} \ b$ holds. Then there are operands a'_i, such that $\tau(\overline{a'_i})$ is closed, $\overline{a_i} \ \widehat{\eta} \ \overline{a'_i}$ and $\tau(\overline{a'_i}) \ \eta^o \ b$.*

Proof. Follows from Lemma 3.11.

Now we can establish our counterpart to [How96, Theorem 3.1].

Theorem 3.13. *Let $\eta \subseteq {\mathcal{T}_0}^2$ be a preorder. Then the following are equivalent.*

1. *η^o is a precongruence*
2. *$\widehat{\eta} \ \subseteq \eta^o$*
3. *$\widehat{\eta}_0 \subseteq \eta$*

Proof. The claim is shown by a chain of implications.

"1 $\Longrightarrow$ 2": Assuming η^o to be a precongruence and $a\ \widehat{\eta}\ b$, we show $a\ \eta^o\ b$ by induction on the definition of $\widehat{\eta}$.
- If $a \in V$ is a variable, the only possibility is $a\ \eta^o\ b$.
- If $a \equiv \tau(\overline{a}_i)$ for some operator τ and operands a_i, there must have been operands a_i' such that $a_i\ \widehat{\eta}\ a_i'$ for every i and $\tau(\overline{a}_i')\ \eta^o\ b$. From the induction hypothesis we may conclude $\overline{a}_i\ \eta^o\ \overline{a}_i'$, which in turn means $\tau(\overline{a}_i)\ \eta^o\ \tau(\overline{a}_i')$ and furthermore $\tau(\overline{a}_i)\ \eta^o\ b$ since η^o is a precongruence.

"2 $\Longrightarrow$ 3": From $\widehat{\eta}\ \subseteq\ \eta^o$ we have $\widehat{\eta}_{\,0} \subseteq (\eta^o)_{\,0} = \eta$ because of Lemma 3.6.

"3 $\Longrightarrow$ 2": From $\widehat{\eta}_{\,0} \subseteq \eta$ we have $(\widehat{\eta}_{\,0})^o \subseteq \eta^o$ by monotonicity (see Lemma 3.6). In conjunction with Corollary 3.10, this becomes $\widehat{\eta} \subseteq\ (\widehat{\eta}_{\,0})^o \subseteq\ \eta^o$.

"2 $\Longrightarrow$ 1": Property (3) of Lemma 3.8 and $\widehat{\eta}\ \subseteq\ \eta^o$ together imply $\widehat{\eta} = \eta^o$, thus η^o is operator-respecting by Lemma 3.8 (2) and a precongruence. □

4 Evaluation, Similarity and Contextual Preorder

This section introduces a big-step operational semantics, and gives all conditions that are necessary for the precongruence proof to go through on this abstract level. This is close to the abstraction given by Howe [How89], however, slightly more general since we have a set of answers and a set of pseudo-values as parameters.

4.1 Higher-Order Computation Systems

Conditions on pseudo-values, answers, and the big-step reduction are formulated in the following definition. Note that in this section there is no connection between answers and pseudo-values, but in subsequent sections, answers will also be pseudo-values.

Definition 4.1 (Higher-Order Computation System). *A tuple* $(\mathcal{L}, \mathcal{PV},\ \mathcal{ANS},\ \Downarrow)$ *is called a* higher-order computation system (HOCS)*, iff the following conditions hold:*

1. $\mathcal{L} = (O, \alpha)$ *is a higher-order computation language.*
2. *There is a set of answer terms* $\mathcal{ANS} \subseteq \mathcal{T}$.
3. $\mathcal{PV} \subseteq \mathcal{T}$ *is a set of pseudo-values for* $\mathcal{L}$ *with* $\mathcal{PV} \cap \mathcal{T}_0 \neq \emptyset$ *(see also Assumption 3.1).*
4. $\Downarrow\ \subseteq \mathcal{T} \times \mathcal{ANS}$.
5. $\forall s \in \mathcal{T}_0, \forall v \in \mathcal{ANS} : s \Downarrow v\ \Longrightarrow\ v \in \mathcal{T}_0$.
6. $\forall v \in \mathcal{ANS} : v \Downarrow v$.
7. $\forall s \in \mathcal{T} : \exists s_c \in \mathcal{T}_0 : s[s_c/x] \Downarrow \Longleftrightarrow\ s \Downarrow$.

We then call $\Downarrow$ evaluation*. If* $s \Downarrow v$*, the term* v *is the* answer *to which* s *is said to* evaluate *to. We simply write* $s \Downarrow$ *if there exists some* v *such that* s *evaluates to* v*, and we say* s converges*. Otherwise, we write* $s \not\Downarrow$ *if there is no such* v.

Note that evaluation $\Downarrow$ may be non-deterministic, as $s \Downarrow v$ may hold for different v. Note also that the notion of answers (values) is rather abstract, e.g. in this section, answers are not further restricted, and there is no connection between answers and pseudo-values. Note also that it may be possible that a value v reduces to another value v'. There are no restrictions on the term s_c, however, in general the term s_c can be uniformly chosen as a non-terminating closed term (usually Ω). In section 5 below, we will treat big-step evaluation defined by small-step evaluation, i.e., when values can be reached by finitely many reduction steps.

Definition 4.2. *The* answer set *of a term s is defined by* $\mathtt{ans}(s) = \{\, t \mid s \Downarrow t \,\}$.

4.2 Simulations and Contextual Equivalence

By observing the termination behavior of terms in all possible contexts, we obtain contextual preorder and contextual equivalence. This is the semantics of terms w.r.t. evaluation. In other papers, contextual equivalence is also called *observational equivalence*.

Definition 4.3. *Given a higher-order computation system* $(\mathcal{L}, \mathcal{PV},\ \mathcal{ANS},\ \Downarrow)$. *The* contextual preorder $\lesssim_c\ \subseteq \mathcal{T}^2$ *is defined by*

$$s \lesssim_c t \iff^{def} (\forall C \in \mathcal{C} :\ C[s] \Downarrow \implies C[t] \Downarrow) \tag{4.1}$$

Contextual equivalence $\simeq_c\ \subseteq \mathcal{T}^2$ *is defined by*

$$s \simeq_c t \iff^{def} s \lesssim_c t \wedge t \lesssim_c s \tag{4.2}$$

Obviously, contextual preorder is a precongruence and contextual equivalence is a congruence.

Contextual preorder and equivalence are not effective, since the number of contexts is infinite, and in general, termination is undecidable. In most calculi the situation can be improved by providing context lemmata which reduce the number of necessary contexts. We will define applicative simulation and mutual similarity, which will be shown to be sufficient conditions for contextual preorder. It uses evaluation followed by an analysis of the subterms. For non-deterministic calculi the definition of $s \lesssim_b t$, it will also require that for choices within s, we have to fix the choices on the t-side. This leads to the notion of a *simulation*, which is based on a, not necessarily effective, *experiment*.

Definition 4.4. *Let $\mathcal{L}$ be a higher-order computation language and $\upsilon \subseteq \mathcal{T} \times \mathcal{T}$ be a relation. Then the* experiment $[\cdot]$ *with* $[\upsilon] \subseteq {\mathcal{T}_0}^2$ *is given by*

$$s\ [\upsilon]\ t \iff^{def} (\forall \theta(\overline{s}_i) :\ s \Downarrow \theta(\overline{s}_i) \implies (\exists \theta(\overline{t}_i) :\ t \Downarrow \theta(\overline{t}_i)\ \wedge\ \overline{s}_i\ \upsilon\ \overline{t}_i)) \tag{4.3}$$

Definition 4.5 (Simulation). *Let a higher-order computation system* $(\mathcal{L}, \mathcal{PV},\ \mathcal{ANS},\ \Downarrow)$ *be given. A relation $\eta \subseteq {\mathcal{T}_0}^2$ is called a* simulation *if and only if $\eta \subseteq [\eta^o]$ holds.*
Similarity $\lesssim_b\ \subseteq {\mathcal{T}_0}^2$ *is defined to be the largest simulation, i.e. as the greatest fixed point of* $[\cdot^o]$.

From Definition 4.4 and Lemma 3.6 it is clear that $[\cdot^o]$ is monotone, hence its greatest fixed point exists.

By $\eta \subseteq [\eta^o]$ a simulation constitutes a *post-fixed point* of the $[\cdot^o]$-operator. A relation η which meets this property is called $[\cdot^o]$-*dense* in [Gor94]. This is the base for the proof principle of *co-induction*, cf. [Gor99,NNH99].

Lemma 4.6. *There are two characterizations of* $\lesssim_b$*:*

- *Similarity* $\lesssim_b$ *is the greatest* $[\cdot^o]$*-dense set, i.e., it is itself* $[\cdot^o]$*-dense and can be characterized as the union of all* $[\cdot^o]$*-dense sets:*

$$\lesssim_b \; = \bigcup \{\eta \mid \eta \subseteq [\eta^o]\}$$

- *An operational characterization is*

$$\lesssim_b \; = \bigcap_i \left[(\mathcal{T}_0{}^2)^o \right]^i$$

This follows, since we are working on sets and relations (cf. [DP92,Gor94]). The technique of fixed points and co-induction has its history and is being used for different calculi ([Mil71,Par81,Abr90,Las98,Gor99]).

A procedure to test similarity is as follows:

Algorithm 4.7. Similarity-Test (s, t).
The input terms s, t must be closed terms.

1. Return True, if $s \equiv t$.
2. Compute the sets $\mathtt{ans}(s), \mathtt{ans}(t)$.
3. For every $\theta(\overline{s_i}) \in \mathtt{ans}(s)$, select a $\theta(\overline{t_i}) \in \mathtt{ans}(t)$. If there is no such term, then fail. Test the following property for all i:
   - If s_i, t_i are (closed) terms, the Similarity-Test for s_i, t_i must return True.
   - If s_i, t_i are higher-order operands, then for every pseudo-valued substitution σ, the Similarity-Test applied to $\sigma(s_i), \sigma(t_i)$ must return true.

The test is successful, if there is a run, using an appropriate selections at every level, which returns "True".

We define *mutual similarity* as a symmetric similarity. Other alternatives are too weak in the case of non-deterministic calculi (see [Gor99, p. 19] and in particular [Las98, p. 92] and [Man05b]).

Definition 4.8 (Mutual Similarity). *The largest equivalence relation contained in* $\lesssim_b$ *is defined by* $\simeq_b \stackrel{def}{=} \lesssim_b \cap \gtrsim_b$ *and called* mutual similarity.

Lemma 4.9. *The relation* $\lesssim_b$ *is a preorder and the relation* $\simeq_b$ *is an equivalence relation.*

Proof. To prove that $\lesssim_b$ is a preorder, we have to prove that it is reflexive and transitive. We give only the proof for transitivity, the case for reflexivity is simpler.

The proof uses that $\lesssim_b \; = \bigcap_i \left[({\mathcal{T}_0}^2)^o \right]^i$ (see Lemma 4.6). We show by induction on i that the relations $\nu_i := \left[({\mathcal{T}_0}^2)^o \right]^i$ are transitive. This is clear for $i = 0$. For $i > 0$, assume $s \; \nu_i \; r \; \nu_i \; t$. For every $\theta(\overline{s_j})$ with $s \Downarrow \theta(\overline{s_j})$, there is some $\theta(\overline{r_j})$ with $r \Downarrow \theta(\overline{r_j})$ and $\overline{s_j} \; {\nu_{i-1}}^o \; \overline{r_j}$. The relation $r \; \nu_i \; t$ implies that there exists $\theta(\overline{t_j})$ with $t \Downarrow \theta(\overline{t_j})$ and $\overline{r_j} \; {\nu_{i-1}}^o \; \overline{t_j}$. Since by induction ν_{i-1} is transitive, and also the open extension ${\nu_{i-1}}^o$ is transitive, hence $\overline{s_j} \; {\nu_{i-1}}^o \; \overline{t_j}$. This shows $s \; \nu_i \; t$. If $s \; \lesssim_b \; r \; \lesssim_b \; t$, then for all i: $s \; \nu_i \; r \; \nu_i \; t$, and since ν_i is transitive for all i, we obtain the claim.

Note that now $\widehat{\lesssim_b}$ can be defined and used, since $\lesssim_b$ is shown to be a preorder. Since $\lesssim_b \; = [{\lesssim_b}^o]$, we have the following property of $\lesssim_b$:

$$s \; \lesssim_b \; t \iff \forall \theta(\overline{a}_i) \in \mathtt{ans}(s) : \; \exists \theta(\overline{a}'_i) \in \mathtt{ans}(t) : \; \overline{a}_i \; {\lesssim_b}^o \; \overline{a}'_i \tag{4.4}$$

4.3 Stability as a Criterion for a Simulation to Imply Contextual Preorder

Contrary to [How89], our proof method emphasizes the use of reduction *rules* instead of performing a certain proof task for every language *operator* (see also [Gor99]). A preparation for this approach are the stability criteria.

Employing the precongruence property of a simulation η to show its inclusion within contextual preorder requires a means to transfer convergent behavior from closed to open terms. Such a condition holds for higher-order computation systems as the following lemma demonstrates.

Lemma 4.10. *Let a HOCS be given and let $\eta \subseteq {\mathcal{T}_0}^2$. Then*

$$(\forall s, t \in \mathcal{T}_0 : \; s \; \eta \; t \implies (s \Downarrow \implies t \Downarrow)) \implies (\forall s', t' \in \mathcal{T} : \; s' \; \eta^o \; t' \implies (s' \Downarrow \implies t' \Downarrow)) \tag{4.5}$$

Proof. Assume that $s' \; \eta^o \; t'$ and $s' \Downarrow \theta(\overline{s'_i})$ holds. Then let σ be the substitution which replaces the variables in $\mathcal{FV}(s', t')$ by (perhaps different) closed pseudo-values $s_{i,c}$ according to Definition 4.1 (7). Then $\sigma(s'), \sigma(t')$ are closed terms. The relation $\sigma(s') \; \eta \; \sigma(t')$ and $\sigma(s') \Downarrow$ implies $\sigma(t') \Downarrow$, and moreover $\sigma(t') \Downarrow$ implies that $t' \Downarrow$ by condition (7) in Definition 4.1.

Theorem 4.11. *Let $\eta \subseteq \mathcal{T}_0 \times \mathcal{T}_0$ be a simulation such that η^o is a precongruence. Then $\eta^o \subseteq \lesssim_c$ is true.*

Proof. Assume terms $s, t \in \mathcal{T}$ such that $s \; \eta^o \; t$ holds. Then $\forall C : \; C[s] \; \eta^o \; C[t]$ because η^o is a precongruence. Since η is a simulation, η matches the premise of Lemma 4.10. Thus, from $\forall C : \; C[s] \; \eta^o \; C[t]$ we may also infer $\forall C : \; C[s] \Downarrow \implies C[t] \Downarrow$ which establishes the claim.

Definition 4.12 (Stability). *Let a (HOCS) be given.*

- *The relation $\eta \subseteq {\mathcal{T}_0}^2$ is called* left-stable*, iff for all closed terms $s, s', t \in \mathcal{T}_0$ such that $s\ \eta\ t$ and $s \Downarrow s'$ hold, the relation $s'\ \eta\ t$ is valid as well.*
- *The relation $\eta \subseteq {\mathcal{T}_0}^2$ is* right-stable*, iff for all $t \in \mathcal{T}_0, \theta(\overline{s_i}) \in \mathcal{ANS} \cap \mathcal{T}_0$: if $\theta(\overline{s_i})\ \eta\ t$, then there is a $\theta(\overline{t_i}) \in \mathcal{ANS} \cap \mathcal{T}_0$, such that $t \Downarrow \theta(\overline{t_i})$ and $\overline{s_i}\ \eta^o\ \overline{t_i}$.*
- *The relation $\eta \subseteq {\mathcal{T}_0}^2$ is called* stable*, iff it is left-stable and right-stable.*

Proposition 4.13. *Let a (HOCS) be given. If $\eta \subseteq {\mathcal{T}_0}^2$ is stable, then η is a simulation, i.e. $\eta \subseteq [\eta^o]$ holds.*

Proof. Assume terms $s, t, \theta(\overline{s}_i) \in \mathcal{T}_0$ such that $s\ \eta\ t$ and $s \Downarrow \theta(\overline{s}_i)$ hold. Since η is left-stable, there exists a term $\theta(\overline{s}_i)$, such that $s \Downarrow \theta(\overline{s}_i)$ and $\theta(\overline{s}_i)\eta t$. Since η is right-stable, there exists $\theta(\overline{t}_i) \in \mathcal{ANS} \cap \mathcal{T}_0$, such that $t \Downarrow \theta(\overline{t}_i)$ and $\overline{(s_i)}\ \eta^o\ \overline{t}_i$ is satisfied. Thus $\overline{s}_i\ \eta^o\ \overline{t}_i$ holds. This shows $s\,[\eta^o]\,t$, and thus the claim is proved.

Definition 4.14. *Let a higher-order computation system $(\mathcal{L}, \mathcal{PV},\ \mathcal{ANS},\ \Downarrow)$ be given. Similarity $\lesssim_b$ is called* well-behaved*, iff*

- $(\widehat{\lesssim_b})_0 = \lesssim_b$,
- ${\lesssim_b}^o$ *is a precongruence, and*
- ${\lesssim_b}^o \subseteq \lesssim_c$.

Note that the properties are not independent, see Theorem 3.13.

Theorem 4.15. *Let a higher-order computation system $(\mathcal{L}, \mathcal{PV},\ \mathcal{ANS},\ \Downarrow)$ be given. If the closed precongruence candidate $(\widehat{\lesssim_b})_0 \subseteq {\mathcal{T}_0}^2$ is stable, then the similarity $\lesssim_b$ is well-behaved.*

Proof. Lemma 4.9 shows that $\lesssim_b$ is a preorder, hence the precongruence candidate can be defined. Proposition 4.13 applied to $(\widehat{\lesssim_b})_0$ shows that $(\widehat{\lesssim_b})_0$ is a simulation. Since $\lesssim_b$ contains all simulations by Lemma 4.6, we obtain $(\widehat{\lesssim_b})_0 \subseteq \lesssim_b$. Thus from Lemma 3.8 (3) we derive ${\lesssim_b}^o \subseteq \widehat{\lesssim_b}$, hence $\lesssim_b \subseteq (\widehat{\lesssim_b})_0$ and thus $\lesssim_b = (\widehat{\lesssim_b})_0$. Condition 3 of Theorem 3.13 is satisfied, hence ${\lesssim_b}^o$ is a precongruence. Now Theorem 4.11 shows that ${\lesssim_b}^o \subseteq \lesssim_c$.

We show a non-trivial property for non-deterministic calculi, and if the similarity $\lesssim_b$ is already known to be well-behaved.

Proposition 4.16. *Let a higher-order computation system $(\mathcal{L}, \mathcal{PV},\ \mathcal{ANS},\ \Downarrow)$ be given, such that the relation $\lesssim_b$ is well-behaved. Then for all closed terms $s, t \in \mathcal{T}_0$ the following holds:*

$$s \lesssim_b t \implies \forall \theta(\overline{s_i}) : s \Downarrow \theta(\overline{s_i}) \implies \exists \theta(\overline{t_i}) : t \Downarrow \theta(\overline{t_i}) \wedge \theta(\overline{s_i}) \lesssim_b \theta(\overline{t_i})$$

Proof. This follows, since $s \lesssim_b t \implies \forall \theta(\overline{s_i}) : s \Downarrow \theta(\overline{s_i})\quad \exists \theta(\overline{t_i}) : t \Downarrow \theta(\overline{t_i}) \wedge \overline{s_i} \lesssim_b{}^o\ \overline{t_i}$. Since ${\lesssim_b}^o$ is a precongruence, we also have $\theta(\overline{s_i}) \lesssim_b \theta(\overline{t_i})$.

5 Calculi with Small-Step Semantics

In order to apply the results to a given higher-order calculus and to prove stability of the closed precongruence candidate for several higher-order calculi in one blow, it is necessary to know more about the given higher-order calculi and their reduction rules. i.e., to formulate restrictions for the calculi.

5.1 HOCS built upon Small Step Reductions

In the following we define the notion of a HOCS+SR, where $\Downarrow$ is based on a *small step reduction relation* $\rightarrow$, and specify more conditions, but retain generality as much as possible. The gain is that the result is applicable to the approximation variants of call-by-need non-deterministic calculi, as well as to call-by-value calculi. A requirement is that the reduction is compatible using a small class of contexts, so-called surface contexts. This will make the reduction non-deterministic to a higher degree. In section 7 we show that also an optimized evaluation suffices (see Theorem 7.3).
The main technical task in this subsection will be to show that the right-stability part of the stability-condition is met.

Definition 5.1 (Surface Contexts). *Let a HOCL $\mathcal{L}$ be given. Then the set $\mathcal{S}$ of* surface contexts *over $\mathcal{L}$ is inductively defined as follows.*

- $[\,] \in \mathcal{S}$
- *If $\tau \in O$ with arity $\alpha(\tau) = \langle k_1, \ldots, k_n \rangle$, $S \in \mathcal{S}$ and $t_j \in {}^{k_j}\mathcal{T}(\mathcal{L})$ for $j \in \{1, \ldots, i-1, i+1, \ldots, n\}$ and $k_i = 0$, then $\tau(t_1, \ldots, t_{i-1}, S, t_{i+1}, \ldots, t_n) \in \mathcal{S}$.*

Note that this corresponds to non-closing surface contexts in [Man05b], i.e., `let` $x = [\cdot]$ `in` t is a surface context, but `let` $x = s$ `in` $[\cdot]$ is not a surface context.

Given a relation $\rightarrow \subseteq \mathcal{T}^2$, the notation $\rightarrow^+$ and $\rightarrow^*$ means the transitive and reflexive-transitive closure, respectively, of $\rightarrow$. Let $\mathcal{DIV}$ be the set of diverging terms, i.e. $\mathcal{DIV} := \{s \in \mathcal{T} \mid \sigma(s) \not\Downarrow \text{ for all pseudo-valued substitutions } \sigma\}$. It is obvious that $\mathcal{DIV} \cap \mathcal{T}_0 = \{s \in \mathcal{T}_0 \mid s \not\Downarrow\}$.

Definition 5.2. *A higher-order computation system with small step reduction (abbreviated as HOCS+SR) is a tuple $(H, \Theta, \rightarrow, sl)$ with a HOCS $H = (\mathcal{L}, \mathcal{PV}, \mathcal{ANS}, \Downarrow)$, such that the following holds:*

1. *Θ is a set of canonical operators with $\emptyset \neq \Theta \subseteq O$.*
2. *$\rightarrow \subseteq \mathcal{T}^2$ is a small step reduction that defines $\Downarrow \subseteq \mathcal{T} \times \mathcal{ANS}$, i.e., $\forall s \in \mathcal{T}, v \in \mathcal{ANS} : s \Downarrow v \iff s \rightarrow^* v$.*
3. *$sl \in \{lazy, strict\}$.*
4. *If $sl = lazy$, then there is a 0-ary constant $\circledcirc \in O \setminus \Theta$.*
5. *If $sl = strict$, then $\mathcal{DIV} \cap \mathcal{T}_0 \neq \emptyset$.*
6. *The set $\mathcal{PV}$ of pseudo-values is inductively defined as follows:*
   (a) *If $sl = lazy$, then*
   $\mathcal{PV} := \{\circledcirc\} \cup \{\theta(\overline{a}_i) \mid \theta$ *is canonical and either $a_i = \overline{x}.t$ for some t, or $a_i \in V$, or $a_i \in \mathcal{PV}$ }.*

(b) *If* $sl = strict$, *then*
$\mathcal{PV} ::= \{\theta(\overline{a}_i) \mid \theta$ *is canonical and either* $a_i = \overline{x}.t$ *for some* t, *or* $a_i \in \mathcal{PV}\}$.

7. *The set of* answers *is defined as:* $\mathcal{ANS} := \mathcal{PV} \setminus \{\circledcirc\}$
8. *For all* $s \in \mathcal{T}_0$: $s \to t \implies t \in \mathcal{T}_0$.
9. *The reduction* $\to$ *is compatible with surface contexts. I.e. for all terms* s, t *and surface context* S: $s \to t \implies S[s] \to S[t]$.
10. *For every term* s *there is a closing substitution* σ, *such that* $s\sigma \Downarrow \iff s \Downarrow$.
11. *If* $sl = lazy$, *then for every* $t \in \mathcal{T}$: $t \to \circledcirc$.
12. *If* $sl = lazy$, *then the constant* $\circledcirc$ *does not converge, i.e.,* $\circledcirc \not\Downarrow$.

We say a HOCS+SR is lazy (strict, respectively) if $sl = lazy$ *(or* $sl = strict$, *respectively).*

Note that the relation $\to$ is not deterministic in general, since it is allowed in arbitrary surface contexts. Furthermore, all canonical constants, i.e., 0-ary operators, are answers.

5.2 Reductions in HOCS+SR

Lemma 5.3. *Let* s, t *be closed terms. Then* $s \to^* t$ *implies that* $t \lesssim_b s$.

Proof. We use induction on the number of $\to$-reductions. The base is reflexivity of $\lesssim_b$. For $s \to t$, from $\mathtt{ans}(t) \subseteq \mathtt{ans}(s)$, $\lesssim_b = [\lesssim_b{}^o]$, and from reflexivity of $\lesssim_b$, we see that $t \lesssim_b s$.

Lemma 5.4. *Let* $s \in \mathcal{DIV} \cap \mathcal{T}_0$, *in particular* $s = \circledcirc$ *is possible. Then*

1. $s \lesssim_b t$ *for all closed terms* t.
2. $s \mathrel{\widehat{\lesssim_b}} t$ *for all closed terms* t.

Proof. The first relation holds, since $\lesssim_b = [\lesssim_b{}^o]$, and since s does not converge. The second relation holds, since for $s = \tau(\overline{s_i})$, the relation $s \mathrel{\widehat{\lesssim_b}} t$ follows from $\overline{s_i} \mathrel{\widehat{\lesssim_b}} \overline{s_i}$ and $s \lesssim_b t$, which holds by (1).

The following lemma prepares the proof that $(\widehat{\lesssim_b})_0$ is right-stable:

Lemma 5.5. *Let a (HOCS+SR) be given and let* $t \in \mathcal{T}_0, \theta(\overline{s_i}) \in \mathcal{ANS} \cap \mathcal{T}_0$. *If* $\theta(\overline{s_i}) \; (\widehat{\lesssim_b})_0 \; t$, *then there is a* $\theta(\overline{t_i}) \in \mathcal{ANS} \cap \mathcal{T}_0$, *such that* $t \Downarrow \theta(\overline{t_i})$ *and* $\overline{s_i} \mathrel{\widehat{\lesssim_b}} \overline{t_i}$.

Proof. The proof is by induction on the depth of $\theta(\overline{s_i})$. The definition of $(\widehat{\lesssim_b})_0$ and Lemma 3.12 imply that there is a closed $\theta(\overline{s'_i})$ with $\overline{s_i} \mathrel{\widehat{\lesssim_b}} \overline{s'_i} \wedge \theta(\overline{s'_i}) \lesssim_b t$. We define $\overline{s''_i}$ by looking at the operands s_i, s'_i for every i:

- If s_i is a higher-order operand, then let $s''_i := s'_i$. The relation $s_i \mathrel{\widehat{\lesssim_b}} s''_i$ holds.
- If $s_i = \circledcirc$ (i.e. this implies sl = lazy), then there are the following cases: If $s'_i \Downarrow$, then let s''_i be such that $s'_i \Downarrow s''_i$: otherwise, $s''_i := \circledcirc$. The relation $s_i \mathrel{\widehat{\lesssim_b}} s''_i$ holds by Lemma 5.4.

– If $s_i \in \mathcal{ANS}$, then $s_i := \theta_i(\overline{s_{i,j}})$ and s_i, s'_i must be closed as well. We apply the induction hypothesis to $s_i (\widehat{\lesssim_b})_0\ s'_i$, which results in a closed $\theta_i(\overline{s'_{i,j}})$ with $s'_i \Downarrow \theta_i(\overline{s'_{i,j}})$, and $\overline{s_{i,j}}\ \widehat{\lesssim_b}\ \overline{s'_{i,j}}$. We define $s''_i := \theta_i(\overline{s'_{i,j}})$. Since $\widehat{\lesssim_b}$ is operator-respecting, we also have $s_i \equiv \theta_i(\overline{s_{i,j}})\ (\widehat{\lesssim_b})_0\ \theta_i(\overline{s'_{i,j}}) \equiv s''_i$.

The construction shows that $\theta(\overline{s'_i}) \to^* \theta(\overline{s''_i})$ by Definition 5.2 (9) and that $\theta(\overline{s''_i})$ is closed by condition 5.2 (8), which implies $\theta(\overline{s''_i}) \lesssim_b t$ by Lemma 5.3. The condition in Definition 5.2 (6) on answer-terms for HOCS+SR now implies that $\theta(\overline{s''_i}) \in \mathcal{ANS}$ for a lazy as well as a strict HOCS+SR. Note that the case $s_i = \circledcirc$ above is not possible for a strict HOCS+SR.

Now $\theta(\overline{s''_i}) \lesssim_b t$ and $\lesssim_b\ =\ [\lesssim_b{}^o]$ imply that there is a closed $\theta(\overline{t_i}) \in \mathcal{ANS}$ with $t \Downarrow \theta(\overline{t_i})$ and $\overline{s''_i} \lesssim_b{}^o\ \overline{t_i}$. The two relations $\overline{s_i}\ \widehat{\lesssim_b}\ \overline{s''_i}$ and $\overline{s''_i} \lesssim_b{}^o\ \overline{t_i}$ can be combined to $\overline{s_i}\ \widehat{\lesssim_b}\ \overline{t_i}$ by Lemma 3.8, part 4.

Corollary 5.6. *Let a (HOCS+SR) be given. Let $s = \theta(\overline{s_i})$ and $t = \theta'(\overline{t_i})$ be closed answers with $s\ (\widehat{\lesssim_b})_0\ t$. Then $\theta = \theta'$ and for all $i : s_i\ \widehat{\lesssim_b}\ t_i$. In particular, for all i: if i is an index of a term in θ, then $s_i\ (\widehat{\lesssim_b})_0\ t_i$ and s_i, t_i are closed pseudovalues, and if $s_i \neq \circledcirc$, then s_i, t_i are closed answers.*

Proof. Follows from Lemma 5.5 and the conditions on answers in Definition 5.2.

Corollary 5.7. *Let a (HOCS+SR) be given. Then $(\widehat{\lesssim_b})_0$ is right-stable.*

Proof. Follows from Corollary 3.10, which shows $\widehat{\lesssim_b} \subseteq ((\widehat{\lesssim_b})_0)^o$, and from Lemma 5.5.

6 Schematic Higher-Order Computation Systems

This section introduces *schematic higher order computation systems* (SHOCS) and establishes the left-stability for every kind of its reductions. The consequence is, that in every lambda calculus which matches the definition of a SHOCS, similarity implies contextual preorder.

6.1 Schematic Reductions and Stability

In order to capture substitution within the reduction rules we need a notion of higher-order contexts.

Definition 6.1 (Redex Multi-Contexts). *Let a HOCL $\mathcal{L}$ be given. Then the set $\mathcal{PRC}$ of* potential redex multi-contexts *over $\mathcal{L}$ is inductively defined as follows.*

1. $[\,] \in \mathcal{PRC}$
2. *Let $\tau \in O$ with arity $\alpha(\tau) = \langle k_1, \ldots, k_n \rangle$. For $j \in \{1, \ldots, n\}$, there are the following possibilities: $a_j \equiv [\,]$, or $a_j \in \mathcal{PRC}$, provided $k_j = 0$. Then $\tau(a_1, \ldots, a_n) \in \mathcal{PRC}$.*

A redex multi-context *is a potential redex multi-context that in addition satisfies the following restrictions:*

- *The top level operator must be non canonical.*
- *All other operators τ mentioned in the second part of the construction must be canonical.*

The set of redex multi-contexts is denoted as $\mathcal{RC}$. A redex multi-context R may have several holes. For a redex multi-context R with n holes, we write $R[t_1, \ldots, t_n]$ for the operand constructed by replacing the n holes in R by $t_1, \ldots, t_n$, where the replacement is done in a left-to-right sequence. Our implicit assumption is that the operand construction is possible.

Note that the redex multi-context consists of one top non-canonical operator, further canonical operators, and holes, where the holes may be filled with terms or higher-order operands, however, for every hole, the "higher-order type" of the operand is fixed.

We define four types of reductions that can be treated in a schematic way.

Definition 6.2. *The four types of* schematic reductions *are of the form:*

1. stop-reduction
   $t \to \circledcirc$, *if t is a term.*
2. simple projection-reduction
   It has two parameters: $\tau \in O \setminus \Theta$ and k, which must be an index of a term-argument of τ, i.e. $\alpha(\tau)\ (k) = 0$
   A simple projection reduction for (τ, k) is of the form $\tau(t_1, \ldots, t_n) \to t_k$ for operands t_i.
3. conditional projection reduction
   It has three parameters: $\tau \in O \setminus \Theta$, k, which must be an index of a term-argument of τ, and a set $I \subset \{1, \ldots, n\}$, where n is the number of arguments of τ.
   For all $i \in I : \alpha(\tau)\ (i) = 0$. A conditional projection reduction for parameters (τ, k, I) is of the form $\tau(t_1, \ldots, t_n) \to t_k$, provided for all $i \in I : t_i \in \mathcal{ANS}$.
4. copy-reduction
   It has four parameters: A redex multi-context R, a set $I \subseteq \{1, \ldots, n\}$, where n is the number of holes of R, an index $k \in (\{1, \ldots, n\} \setminus I)$, and a function $\pi : I \to \{1, \ldots, n\}$. Conditions are: the k^{th} hole of R accepts only higher-order operands from ${}^{|I|}\mathcal{T}$, and for all $j \in I$: the arguments at index $\pi(j)$ of R must be terms.
   A copy-reduction for the parameters (R, I, k, π) is $R[\overline{s_i}] \to r[\overline{s_{\pi(j)}}/\overline{x_j}]$, where $\{s_j \mid j = 1, \ldots, n, j \neq k\} \subseteq \mathcal{PV} \cup {}^{(\geq 1)}\mathcal{T}$, and $s_k \equiv \overline{x_j}.r$ and $s_k \in {}^{|I|}\mathcal{T}$.

A projection reduction *is a simple projection reduction or a conditional projection reduction.*

Example 6.3. In [Man05a,Man05b] there is an example for a calculus with stop-reductions. A simple projection reduction is the reduction defined by a non-deterministic `choice`-rule (`choice` $s\ t$) $\to s$ and (`choice` $s\ t$) $\to t$, whereas the

local amb-rule and seq-rule with $(\texttt{amb}\ v\ t) \to v$ and $(\texttt{amb}\ s\ v) \to v$, $(\texttt{seq}\ v\ t) \to t$, where v are answers, are conditional projection rules. The call-by-value beta-reduction rule is a copy-reduction with $R = @(\lambda([\cdot]), [\cdot])$. The case-reduction has two different encodings, depending on whether the SHOCS is lazy or strict. The case-reduction $(\texttt{case}\ (\texttt{cons}\ s_1\ s_2)\ \texttt{of}\ (\texttt{cons}\ x_1\ x_2) \to t_1; \texttt{Nil} \to t_2) \to t[s_1/x_1, s_2/x_2]$ and the analogous case-reduction in the approximation calculus are also copy-reductions, where the redex multi-context is $\texttt{case}((\texttt{cons}([\cdot],[\cdot]),\ [\cdot],\ [\cdot])$. In the lazy variant, the arity of case is $\langle 0, 2, 0\rangle$, and the translation of the reduction is $\texttt{case}(\texttt{cons}(s_1, s_2), (x_1x_2.t_1), \circledcirc) \to t_1[s_1/x_1, s_2/x_2]$, and the reduction for Nil is as follows: $\texttt{case}((\texttt{Nil}),\ (x_1x_2.t_1),\ t_2) \to t_2$. In the strict (i.e., call-by-value) variant, the arity of case is $\langle 0, 2, 1\rangle$, and the translation of a reduction is $\texttt{case}(\texttt{cons}(s_1, s_2), (x_1x_2.t_1), z.t_2) \to t_1[s_1/x_1, s_2/x_2]$, and the reduction for Nil is as follows: $\texttt{case}((\texttt{Nil}),\ (x_1x_2.t_1),\ z.t_2) \to t_2[r/z] \equiv t_2$, where r is any term. The call-by-value encoding of the copy-reductions does not enforce evaluation of t_2 in the case that the scrutinized argument is a non-empty list. The reason for the slightly unusual encoding of the Nil-case is that in the naive encoding, a copy reduction would require the term t_2 to be reduced to a pseudo-value, which would prevent the proper encoding of call-by-value calculi.

More complex reduction rules could also be accommodated by using combinations of simpler ones.

Note that in the following proofs, the compatibility of $\lesssim_b$ as well as the transitivity of $(\widehat{\lesssim_b})_0$ are unknown and must not be used in proofs.

Lemma 6.4. *If $S[s] \to S[t]$, where $s \to t$ is a schematic reduction, then $S[s] \in \mathcal{T}_0 \implies S[t] \in \mathcal{T}_0$*

Proof. This is obvious for stop- and projection reductions. For copy reductions with $s_k = \overline{x_i}.r$, the resulting term $r[\overline{s_{\pi(j)}}/\overline{x_j}]$ is closed, since $\overline{s_{\pi(j)}}$ and $\overline{x_i}.r$ are closed.

6.2 Stability of Schematic Reductions

In this subsection, we assume that a HOCS+SR is given.
For proving stability of $(\widehat{\lesssim_b})_0$ we first treat top-level reductions only. The results will be transferred into arbitrary surface contexts later.

Corollary 6.5. *The relation $(\widehat{\lesssim_b})_0$ is left-stable w.r.t. stop-reductions. I.e., if $s\ (\widehat{\lesssim_b})_0\ t$ for closed terms s, t and $s \to \circledcirc$, then $\circledcirc\ (\widehat{\lesssim_b})_0\ t$.*

Proof. Follows from Lemma 5.4.

Lemma 6.6. *Given the parameters (τ, k), let us assume that all simple projections reductions for (τ, k) are permitted. Then the relation $(\widehat{\lesssim_b})_0$ is left-stable w.r.t. a simple projection-reduction for parameter (τ, k). I.e., if $s\ (\widehat{\lesssim_b})_0\ t$ for closed terms s, t and $s \to r$ by a simple projection-reduction for (τ, k), then $r\ (\widehat{\lesssim_b})_0\ t$.*

Proof. For a closed term $\tau(\overline{s_i})$ let $\tau(\overline{s_i}) \to s_k$ be a simple projection reduction. The relation $\tau(\overline{s_i})\ (\widehat{\lesssim_b})_0\ t$ for closed terms $\tau(\overline{s_i}), t$ implies that there is a closed term $\tau(\overline{s'_i})$, such that $\overline{s_i}\ \widehat{\lesssim_b}\ \overline{s'_i}$, and $\tau(\overline{s'_i}) \lesssim_b t$. The projection reduction $\tau(\overline{s'_i}) \to s'_k$ is permitted and k denotes a term-argument, hence by Lemma 5.3 $s'_k \lesssim_b \tau(\overline{s'_i})$. Transitivity of $\lesssim_b$ implies $s'_k \lesssim_b t$. Together with $s_k\ (\widehat{\lesssim_b})_0\ s'_k$ this implies $s_k\ (\widehat{\lesssim_b})_0\ t$, and hence the claim holds.

Lemma 6.7. *Given the parameters (τ, k, I), let us assume that all conditional projections reductions for (τ, k, I) are permitted, and that reductions are compatible with surface contexts. Then the relation $(\widehat{\lesssim_b})_0$ is left-stable w.r.t. a conditional projection-reduction for (τ, k, I).*

Proof. For a closed term $\tau(\overline{s_i})$ let $\tau(\overline{s_i}) \to s_k$ be a conditional projection reduction. The relation $\tau(\overline{s_i})\ (\widehat{\lesssim_b})_0\ t$ for closed terms $\tau(\overline{s_i}), t$ implies that there is a closed term $\tau(\overline{s'_i})$, such that $\overline{s_i}\ \widehat{\lesssim_b}\ \overline{s'_i}$, and $\tau(\overline{s'_i}) \lesssim_b t$. Since the reduction is applicable, for all $i \in I : s_i \in \mathcal{ANS}$ must hold. For $i \in I$, the terms s_i, s'_i are closed, and the relation $s_i\ \widehat{\lesssim_b}\ s'_i$ implies that $s'_i \Downarrow s''_i \in \mathcal{ANS}$ with $s_i\ \widehat{\lesssim_b}\ s''_i$ by Lemma 5.5. For $i \notin I$, let $s''_i := s'_i$. Then $\tau(\overline{s'_i}) \to^* \tau(\overline{s''_i})$ by the assumption that reduction is compatible with surface contexts, and thus Lemma 5.3 implies $\tau(\overline{s''_i}) \lesssim_b \tau(\overline{s'_i})$, and by transitivity, $\tau(\overline{s''_i}) \lesssim_b t$. Since all conditional projection reductions for (τ, k, I) are permitted, we have $\tau(\overline{s''_i}) \to s''_k$, hence by Lemma 5.3 $s''_k \lesssim_b \tau(\overline{s''_i})$. Transitivity of $\lesssim_b$ implies $s''_k \lesssim_b t$. Together with $s_k\ (\widehat{\lesssim_b})_0\ s''_k$ this implies $s_k\ (\widehat{\lesssim_b})_0\ t$, and hence the claim holds.

Lemma 6.8. *If a copy-reduction is applicable to a term t at top-level, then $t \equiv \tau(\overline{t_i})$, where τ is a non-canonical operator, and either $t_i \in \mathcal{PV}$, or t_i is a higher-order operand.*

Proof. Since $R[\overline{r_i}] \to a$ implies that for $R[\overline{r_i}] \equiv \tau(\overline{t_i})$, we have $t_i \in \mathcal{PV}$, or t_i is a higher-order operand, and the conditions for $t_i \in \mathcal{PV}$ according to Definition 5.2 are satisfied. This holds for lazy as well as strict HOCS+SR.

Lemma 6.9. *Given the parameters (R, I, k, π), let us assume that all copy-reductions for (R, I, k, π) are permitted. Assume also that reductions are compatible with surface contexts. Then the relation $(\widehat{\lesssim_b})_0$ is left-stable w.r.t. reduction by copy-reductions for (R, I, k, π). I.e., if $R[\overline{s_i}]\ (\widehat{\lesssim_b})_0\ t$ for closed terms $R[\overline{s_i}], t$ and $R[\overline{s_i}] \to s'$ by a copy-reduction w.r.t. (R, I, k, π), then s' is a closed term and $s'\ (\widehat{\lesssim_b})_0\ t$.*

Proof. Let $R[\overline{s_i}]\ (\widehat{\lesssim_b})_0\ t$ for a closed term $R[\overline{s_i}]$, such that this term reduces by a copy-reduction for (R, I, k, π) to $r[\overline{s_{\pi(j)}}/\overline{x_j}]$, where $s_k \equiv \overline{x_j}.r$. Note that the definition of redex multi-contexts enforces that all operands s_i are closed. Let $R[\overline{s_i}] = \tau(\overline{a_i})$. Since the copy-reduction is applicable, from the structure of redex multi-contexts it follows that every a_i is either $\circledcirc$, an answer, or a higher-order operand. We obtain closed a'_i, such that $\overline{a_i}\ \widehat{\lesssim_b}\ \overline{a'_i}, \tau(\overline{a'_i}) \lesssim_b t$. For every i, there are the following possibilities:

- $a_i = \circledcirc$. This implies sl = lazy. If $a'_i \Downarrow$, then let a''_i be such that $a'_i \Downarrow a''_i$, otherwise, let $a''_i := \circledcirc$. We have $a_i \mathrel{\widehat{\lesssim_b}} a''_i$ in either case.
- a_i is a higher-order operand. Then a'_i is also a higher-order operand, hence let $a''_i := a'_i$. Obviously, $a_i \mathrel{\widehat{\lesssim_b}} a''_i$.
- $a_i \in \mathcal{ANS}$ with $a_i \equiv \theta_i(\overline{a_{i,j}})$. Then by Lemma 5.5, there is a term $\theta_i(\overline{a'_{i,j}})$, such that $a'_i \Downarrow \theta_i(\overline{a'_{i,j}})$, and $\overline{a_{i,j}} \mathrel{\widehat{\lesssim_b}} \overline{a'_{i,j}}$. Let $a''_i := \theta_i(\overline{a'_{i,j}})$. The relation $a_i \mathrel{\widehat{\lesssim_b}} a''_i$ holds, since $\mathrel{\widehat{\lesssim_b}}$ is operator-respecting.

By the conditions on reductions for a HOCS+SR, we have $\tau(\overline{a'_i}) \to^* \tau(\overline{a''_i})$, hence $\tau(\overline{a''_i}) \lesssim_b t$.

The structure of redex multi-contexts, Lemma 6.8 and Corollary 5.6 together imply, using induction on the depth of operands, that $\tau(\overline{a''_i}) \equiv R[\overline{s'_i}]$ and $\overline{s_i} \mathrel{\widehat{\lesssim_b}} \overline{s'_i}$ where the operands s'_i are higher-order operands or pseudo-values. This permits to reduce this term by the same copy-reduction: $\tau(\overline{a''_i}) \equiv R[\overline{s'_i}] \to r'[\overline{s'_{\pi(j)}}/\overline{x_j}]$. The Substitution Lemma 3.9 shows that $r[\overline{s_{\pi(j)}}/\overline{x_j}]\ (\mathrel{\widehat{\lesssim_b}})_0\ r'[\overline{s'_{\pi(j)}}/\overline{x_j}]$. Since $\tau(\overline{a'_i}) \to^* \tau(\overline{a''_i}) \to r'[\overline{s'_{\pi(j)}}/\overline{x_j}]$, we have $r'[\overline{s'_{\pi(j)}}/\overline{x_j}] \lesssim_b t$. Hence by composition, and since the terms are closed, we also have $r[\overline{s_{\pi(j)}}/\overline{x_j}]\ (\mathrel{\widehat{\lesssim_b}})_0\ t$

Now the stability results can be transferred from top-level reductions to arbitrary surface contexts.

Lemma 6.10. *If $(\mathrel{\widehat{\lesssim_b}})_0$ is left-stable w.r.t. a reduction $s \to r$, then it is also left-stable w.r.t. $S[s] \to S[r]$, where we assume that reductions are compatible with surface contexts.*

Proof. We prove the claim by induction on the depth of the hole of S. Therefore, it is sufficient to prove that if $s \to r \wedge s\ (\mathrel{\widehat{\lesssim_b}})_0\ t' \implies r\ (\mathrel{\widehat{\lesssim_b}})_0\ t'$, then $\tau(s_1, \ldots s_{i-1}, s, s_{i+1}, \ldots, s_n)\ (\mathrel{\widehat{\lesssim_b}})_0\ t$ implies $\tau(s_1, \ldots s_{i-1}, r, s_{i+1}, \ldots, s_n)\ (\mathrel{\widehat{\lesssim_b}})_0\ t$. Assume $\tau(s_1, \ldots s_{i-1}, s, s_{i+1}, \ldots, s_n)\ (\mathrel{\widehat{\lesssim_b}})_0\ t$. Then there is a term $\tau(s'_1, \ldots s'_{i-1}, s', s'_{i+1}, \ldots, s'_n)$ such that $s_j \mathrel{\widehat{\lesssim_b}} s'_j$ for $j \neq i$, and $s\ (\mathrel{\widehat{\lesssim_b}})_0\ s'$, and $\tau(s'_1, \ldots s'_{i-1}, s', s'_{i+1}, \ldots, s'_n) \lesssim_b t$. By the induction hypothesis, $s\ (\mathrel{\widehat{\lesssim_b}})_0\ s'$ and $s \to r$ implies $r\ (\mathrel{\widehat{\lesssim_b}})_0\ s'$. Now we can use the definition of $\mathrel{\widehat{\lesssim_b}}$, since we have $s_j \mathrel{\widehat{\lesssim_b}} s'_j$ for $j \neq i$, and $r \mathrel{\widehat{\lesssim_b}} s'$, and obtain $\tau(s_1, \ldots, s_{i-1}, r, s_{i+1}, \ldots, s_n)\ (\mathrel{\widehat{\lesssim_b}})_0\ t$ because $\tau(s_1, \ldots, s_{i-1}, r, s_{i+1}, \ldots, s_n)$ and t are closed.

6.3 Schematic Higher Order Computation System

Now we show that the restriction to schematic rules and the appropriate definitions of answers is sufficient to use the similarity for proving contextual preorder.

Definition 6.11 (Schematic Higher Order Computation System (SHOCS)). *Given a HOCL $\mathcal{L}$, a canonical set of operators $\emptyset \neq \Theta \subseteq O$, and $sl \in \{lazy, strict\}$. All of the following conditions must hold:*

- *If $sl = lazy$, let there be a 0-ary symbol $\circledcirc$, and assume that*
- *All stop-reductions are permitted.*
- *There is a fixed set P, such that for every $p \in P$, all simple-projection reductions for p, or all conditional projection reductions for p, or all copy-reductions for p, respectively, are permitted.*
- *All small-step reductions above are also permitted in surface contexts. I.e., if $s \to t$, and S is a surface context, then also $S[s] \to S[t]$.*
- *Let the set of pseudo-values, the set of answer terms, and $\Downarrow$ be defined as in Definition 5.2.*
- *There is a closed pseudo-value.*
- *If $sl = strict$, there is a closed and diverging term Ω.*

Then we call this a Schematic Higher Order Computation System (SHOCS).

A trivial consequence of the definition of SHOCS is that a HOCS+SR, in which all non-stop-reductions are projection-reductions or copy-reductions, and reduction is compatible with surface-contexts, and whenever a simple projection reduction for p, a conditional projection reduction for p or a copy-reduction for p is permitted, then all simple projection reductions for p, all conditional projection reductions for p or all copy-reduction for p, respectively, are also permitted, then this HOCS+SR is also a SHOCS.

Theorem 6.12. *Every SHOCS is a HOCS+SR.*

Proof. It is easy to see that the conditions 4.1 (1) – (6) are satisfied.
We have to show that 4.1 (7) holds: The structure of the reductions shows that $s \to t$ implies $\sigma(s) \to \sigma(t)$ for any substitution of a term for free variables in s. This can easily be checked for all schematic reductions.
Now we show that $\forall s \in \mathcal{T} : \exists s_c \in \mathcal{T}_0 : s[s_c/x] \Downarrow \iff s \Downarrow$. For lazy SHOCS, we let $s_c := \circledcirc$, and for strict SHOCS, we let $s_c = \Omega$. If $s \Downarrow$, then by the argument above, also $s[s_c/x] \Downarrow$, since answers have a canonical top operator, which cannot be removed by a substitution. If $s[s_c/x] \Downarrow v$, then we slightly generalize, and assume that there is a term $s[r_1, \ldots, r_m] \Downarrow v$, where all r_i are closed diverging terms. By checking the cases for reductions, we see that the first reduction of $s[r_1, \ldots, r_m] \xrightarrow{*} s'$ is either a reduction within some $r_j \to r'_j$, or it is a reduction within s, independent of all r_i. In the first case, we can apply the induction hypothesis to show $s \Downarrow$. In the second case $s[r_1, \ldots, r_m] \to s'[r_1, \ldots, r_{m'}] \xrightarrow{*} v$, and we obtain that $s \to s' \Downarrow v$, and then we can use induction.

We have proved all conditions for a HOCS. All the remaining conditions for a HOCS+SR are satisfied either by definition, or are obvious consequences.

Theorem 6.13. *In every SHOCS, $\lesssim_b$ is well-behaved.*

Proof. Theorem 6.12 shows that the SHOCS is also a HOCS+SR, hence we can apply our machinery. The relation $(\widehat{\lesssim_b})_0$ is stable, which follows from Corollary 6.5 and Lemmas 6.6, 6.7, 6.9, and 6.10. Now Theorem 4.15 is applicable.

6.4 Equivalence of Similarity and Contextual Preorder

A counterexample in [Man05b] shows that generally ${\lesssim_b}^o \neq \lesssim_c$ for a non-deterministic calculus. This also holds in calculi if there are no selectors for some canonical operator. Analogously to Howe [How89], the following criterion is valid in SHOCS:

Theorem 6.14. *Let a HOCS be given, such that the following holds:*

1. *Evaluation $\Downarrow$ is deterministic,*
2. $\forall s : s \Downarrow s' \implies s \simeq_c s'$,
3. *similarity is well-behaved, and*
4. *for all canonical operators θ, θ' and all operands s_i, t_i:*

$$\theta(s_1, \ldots, s_n) \lesssim_c \theta'(t_1, \ldots, t_n) \implies \theta = \theta' \wedge s_i \; {\lesssim_c}^o \; t_i$$

Then $\lesssim_b \; = \; (\lesssim_c)_0$.

Proof. It is sufficient to show that $\lesssim_c$ is a simulation. Let s, t be closed terms with $s \; (\lesssim_c)_0 \; t$. If $s \Downarrow$, then $t \Downarrow$ by definition of $\lesssim_c$. Now let $s \Downarrow \theta(s_1, \ldots, s_n)$ and $t \Downarrow \theta'(t_1, \ldots, t_n)$ be the unique answers due to condition (1). By condition (2), $\theta(s_1, \ldots, s_n) \lesssim_c \theta'(t_1, \ldots, t_n)$. The above condition (4) implies $\theta = \theta'$, and $\forall i : s_i \; {\lesssim_c}^o \; t_i$. Hence $(\lesssim_c)_0$ is a simulation. Together with the condition (3) this implies that $\lesssim_b \; = \; (\lesssim_c)_0$.

The criteria in the theorem above are satisfied in most deterministic lambda-calculi, e.g. using the beta-rule and the case-rules for selection.

In non-deterministic lambda-calculi, the presence of the `seq`-operator and the reduction $(\texttt{seq}\ v\ s) \to s$ in the calculus and other conditions enforce that the contextual preorder implies $\lesssim_b$ in some useful cases.

Theorem 6.15. *Let a SHOCS be given, such that the following holds:*

1. *There is a* `seq`*-operator and the reduction $(\texttt{seq}\ v\ s) \to s$ is valid whenever $v \in \mathcal{ANS}$. Also $(\texttt{seq}\ r\ s) \Downarrow \implies r \Downarrow$ holds.*
2. *For all canonical operators θ, θ' and all operands s_i, t_i:*
   $\theta(s_1, \ldots, s_n) \lesssim_c \theta'(t_1, \ldots, t_n) \implies \theta = \theta'$ and $\forall i : s_i \lesssim_c t_i$.
3. *For every term t there is a surface context $D_t[\cdot]$, such that for all answers v: $D_t[v] \to^* t[v/x]$, and for all $r : D_t[r] \Downarrow \iff \exists v : r \Downarrow v \wedge t[v/x] \Downarrow$.*

Then for all closed terms s, t: if $\texttt{ans}(t)$ is finite, and every $t_i \in \texttt{ans}(t)$ is a closed term without higher-order operands as suboperands, then $s \lesssim_c t \implies s \lesssim_b t$.

Proof. We show that $\lesssim_c$ for these restricted term pairs is a simulation in this special case.
Let $s_0 \in \texttt{ans}(s)$ and let $\texttt{ans}(t) = \{t_1, \ldots, t_n\}$. We show that $s_0 \lesssim_c t_i$ for some $i = 1, \ldots, n$: Suppose this is false. Then for every i, there exists a context C_i, such that $C_i[s_0] \Downarrow$, but $C_i[t_i] \not\Downarrow$.
Then let $r := (\texttt{seq}\ C_1[x]\ (\texttt{seq}\ C_2[x] \ldots (\texttt{seq}\ C_{n-1}[x]\ C_n[x]) \ldots))$. Let D be

the surface context for t according to condition (3). Then $D[s] \to^* D[s_0] \to^* t[s_0/x] \Downarrow$, due to `seq`-reductions and the assumption that $\forall i : C_i[s_0] \Downarrow$. However, $D[t_i] \not\Downarrow$ for every i: Assume otherwise; then $D[t_j] \Downarrow$ for some j, and by the same reasoning as above, we see that this implies $C_j[t_j] \Downarrow$, which is not possible.
We have reached a contradiction by condition (3), and conclude that $s_0 \lesssim_c t_k$ for some k. Since s_0, t_k are answers, we also have $s_0 = \theta(\overline{s_{0,i}}), t_k = \theta(\overline{t_{k,i}})$, and by the condition on t, we also have $s_{0,i} \lesssim_c t_{k,i}$ for all i and $t_{k,i}$ is a closed answer without a higher-order suboperand. Hence $\lesssim_c$ on the restricted term pairs is a simulation, and the lemma holds.

The conditions in Theorem 6.15 are satisfied in most calculi, e.g. to satisfy the condition (3) often the contexts $\texttt{let } x = [\cdot] \texttt{ in } t$, or $(\lambda x.t\ [\cdot])$ can be used. Note that the counterexample in [Man05b] indeed has an infinite set $\texttt{ans}(t)$.

7 Optimizing the Procedure for Checking Similarity in SHOCS

The goal of this section is to give support for optimizing the similarity check procedure for SHOCS in making it more deterministic, i.e., by avoiding unnecessary computations of answers. This allows to avoid redundant answers which are equivalences due to Ω- or $\circledcirc$-related equivalences, e.g. the equivalences $\Omega \simeq_b @(\Omega, r)$ ($\circledcirc \simeq_b @(\circledcirc, r)$, respectively), which holds in all example calculi in section 8. An illustrating example for the use of this optimization is in subsection 8.1.

The following definition captures optimized reduction strategies that can avoid reductions that are redundant, e.g. may be more deterministic than $\to$, and may also avoid reductions to answers that are subsumed by other answers. We restrict the presentation to a modified big-step evaluation relation $\Downarrow_{std}$.

Definition 7.1. *Given a SHOCS H. Let $\Downarrow_{std} \subseteq \mathcal{T} \times \mathcal{ANS}$ be a relation with the following properties:*

1. $\forall s \in \mathcal{T}_0, v' \in \mathcal{ANS} : s \Downarrow_{std} v' \implies \exists v \in \mathcal{ANS} : s \Downarrow v \wedge v \simeq_b v'$.
2. $\forall s \in \mathcal{T}_0, s \Downarrow v : \exists v' \in \mathcal{T}_0 \cap \mathcal{ANS} : s \Downarrow_{std} v' \wedge v \lesssim_b v'$.

Then $\Downarrow_{std}$ is called a standardizing evaluation *for H.*

The first condition permits that $\Downarrow_{std}$ may lead to different, yet $\simeq_b$-equivalent, answers, than $\Downarrow$ leads to. The second condition enables $\Downarrow_{std}$ to choose from an answer set $\texttt{ans}(s)$ only the maximal ones w.r.t. $\lesssim_b$-ordering.

Definition 7.2. *Given a SHOCS H and a standardizing evaluation $\Downarrow_{std}$. Then $\leq_{b,std}$ is defined analogously to $\lesssim_b$ as the greatest fixpoint of $[\cdot^o]_{std}$. Given a relation υ on terms, this is defined as follows:*

$$s\ [\upsilon]_{std}\ t \overset{def}{\iff} (\forall \theta(\overline{s_i}) \in \mathcal{ANS} : s \Downarrow_{std} \theta(\overline{s_i}) \implies (\exists \theta(\overline{t_i}) \in \mathcal{ANS} : t \Downarrow_{std} \theta(\overline{t_i}) \wedge \overline{s_i}\ \upsilon\ \overline{t_i})) \quad (7.1)$$

Theorem 7.3. *Given a SHOCS H and a standardizing evaluation $\Downarrow_{std}$ for H. Let $\leq_{b,std}$ be the similarity w.r.t. $\Downarrow_{std}$ and H as defined in Definition 7.2 above. Then for all closed terms s, t: $s \lesssim_b t$ iff $s \leq_{b,std} t$.*

Proof. Let $\leq_{i,b,std} := \left[(\mathcal{T}_0{}^2)^o\right]^i_{std}$ and $\lesssim_{i,b} := \left[(\mathcal{T}_0{}^2)^o\right]^i$. Now we use the characterization from Lemma 4.6, which holds for $\leq_{i,b,std}$ analogously. I.e.,

$$\lesssim_b \;=\; \bigcap_i \lesssim_{i,b} \quad \text{and} \quad \leq_{b,std} \;=\; \bigcap_i \leq_{i,b,std}$$

The relations $\lesssim_{i,b}$ and $\leq_{i,b,std}$ are reflexive and transitive, which is shown for the first one in Lemma 4.9, and can be proved for the latter in the same way.

(1) By induction on i we show that $\lesssim_{i,b} \subseteq \leq_{i,b,std}$ is valid for all i: For $i = 0$, this is trivial. Let $i > 0$ and let s, t be closed terms such that $s \lesssim_{i,b} t$ holds. We show that $s \leq_{i,b,std} t$ is true. Assume $s \Downarrow_{std} v'_s \equiv \theta(\overline{s'_j}) \in \mathcal{ANS}$. By condition (1) of Definition 7.1 there is some $v_s \equiv \theta(\overline{s_j})$ with $s \Downarrow v_s$ and $v_s \simeq_b v'_s$. This implies $\forall i : v'_s \lesssim_{i,b} v_s$ and thus $\overline{s'_j} \lesssim_{i-1,b}{}^o \overline{s_j}$ since v_s and v'_s both are answers. Furthermore, from $s \lesssim_{i,b} t$ we obtain an answer $v_t \equiv \theta(\overline{t_j})$, such that $t \Downarrow v_t$ as well as $\overline{s_j} \lesssim_{i-1,b}{}^o \overline{t_j}$ holds. Since $\Downarrow_{std}$ is standardizing, there is a $v'_t \in \mathcal{ANS}$, such that $t \Downarrow_{std} v'_t$ and $v_t \lesssim_b v'_t$. Since for all answers w, only $w \Downarrow w$ is possible, we also have $v'_t \equiv \theta(\overline{t'_j})$ and $\overline{t_j} \lesssim_{i-1,b}{}^o \overline{t'_j}$. This implies $\overline{s'_j} \lesssim_{i-1,b}{}^o \overline{t'_j}$, since $\lesssim_{i-1,b}{}^o$ is transitive (see the proof of Lemma 4.9). The induction hypothesis implies $\overline{s'_j} \leq_{i-1,b,std}{}^o \overline{t'_j}$. This proves $s \leq_{i,b,std} t$. Using the characterizations of $\lesssim_b$ and $\leq_{b,std}$, we have shown $\lesssim_b \subseteq \leq_{b,std}$.

(2) We show by induction on i that $\leq_{i,b,std} \subseteq \lesssim_{i,b}$ holds for all i: For $i = 0$, this is trivial. Let $i > 0$ and s, t be closed terms with $s \leq_{i,b,std} t$. If $s \Downarrow v_s$, then there exists $v'_s \in \mathcal{ANS}$ with $s \Downarrow_{std} v'_s$ and $v_s \lesssim_b v'_s$. There is a closed value $v'_t \in \mathcal{ANS}$ such that $t \Downarrow_{std} v'_t$, $v'_s = \theta(\overline{s'_j})$, $v'_t = \theta(\overline{t'_j})$, and $\overline{s'_j} \leq_{i-1,b,std}{}^o \overline{t'_j}$. The relation $v_s \lesssim_b v'_s$ implies that $v_s = \theta(\overline{s_j})$ and $\overline{s_j} \lesssim_b{}^o \overline{s'_j}$, and also $\overline{s_j} \lesssim_{i-1,b}{}^o \overline{s'_j}$. The first part of the proof implies $\overline{s'_j} \leq_{i-1,b,std}{}^o \overline{t'_j}$, and thus by transitivity of $\leq_{i-1,b,std}{}^o$ also $\overline{s_j} \leq_{i-1,b,std}{}^o \overline{t'_j}$. By induction hypothesis, this implies $\overline{s_j} \lesssim_{i,b}{}^o \overline{t'_j}$. Since there is an answer v_t with $t \Downarrow v_t$ and $v_t \simeq_b v'_t$, we also have $\overline{t'_j} \lesssim_{i,b}{}^o \overline{t_j}$, hence $s \lesssim_{i,b} t$. By the characterization above, this implies $\leq_{b,std} \subseteq \lesssim_b$. □

The similarity test in Algorithm 4.7 can be optimized, since the sets $\texttt{ans}(\cdot)$ can be defined w.r.t. $\Downarrow_{std}$ now. Also, the evaluation to answers can be made less non-deterministic.

Corollary 7.4. *Given a SHOCS. Let s, t be closed terms, and let $s \xrightarrow{bot,*} t$ iff t can be constructed from s by replacing subterms by $\circledcirc$ (or Ω for strict SHOCS). Then $s \lesssim_b t$ and $s \lesssim_c t$.*
Thus $v' \xrightarrow{bot,*} v$ is a sufficient criterion for $v \lesssim_b v'$ in Definition 7.1 (2).

Proof. This follows since ${\lesssim_b}^o$ is a precongruence and $\circledcirc$ (Ω, respectively) is the smallest element w.r.t. ${\lesssim_b}^o$ by Theorem 6.13.

If, for some term, there are $\lesssim_b$-related answers, e.g. inherited from equivalences like $\circledcirc \simeq_b @(\circledcirc, r)$ or $\Omega \simeq_b @(\Omega, r)$, then the standardizing evaluation permits to select the maximal one w.r.t. the $\lesssim_b$-ordering. In this case, $v \Downarrow_{std} v'$ is possible for syntactically different, but $\simeq_b$-equivalent answers v, v'.

8 Applications

We give a series of calculi in which Theorem 6.13 is applicable, and $\lesssim_b$ is thus well-behaved. We also show how Theorem 7.3 can be advantageously applied.

8.1 A Tiny Lazy Approximation Calculus

The reduction rules of an approximation calculus for a lazy call-by-need non-deterministic calculus as defined in [Man05a,Man05b] is slightly reduced, insofar as all the let-shuffling rules are omitted. This is possible, as the arguments in [Man05b] show. The language has application, `let`, `pick`, `lambda`, `seq` as operators, where λ is the only canonical operator. The pseudo-values are the abstractions and $\circledcirc$. The rules are:

`let` $x = v$ `in` s	$\to s[v/x]$	if v is a pseudo-value
`pick` $s\ t$	$\to s$	
`pick` $s\ t$	$\to t$	
`seq` $v\ t$	$\to t$	if v is an abstraction
$((\lambda x.s)\ v)$	$\to s[v/x]$	if v is a pseudo-value
s	$\to \circledcirc$	if s is not a pseudo-value

If λ is the only canonical operator, and the reduction rules are permitted in all surface contexts, then we can apply Theorem 6.13, and obtain that $\lesssim_b$ is well-behaved.

An example of terms for which $\lesssim_c$ can be derived via $\lesssim_b$, is given. First we define some abbreviations, and use the usual conventions for the notations of lambda-expressions. We will make use of Theorem 7.3 to avoid redundant answers.

$$
\begin{aligned}
K &:= \lambda xy.x \\
Y &:= \lambda f.(\lambda x.f(x\ x))\ (\lambda x.f(x\ x)) \\
Y_2 &:= \lambda f.(\lambda xy.f(x\ y\ y))\ (\lambda xy.f(x\ y\ y))\ (\lambda xy.f(x\ y\ y))
\end{aligned}
$$

We show that $Y\ K \lesssim_b Y_2\ K$ using the optimization method of Theorem 7.3. First we ignore the stop-reductions. $Y\ K$ reduces to $(\lambda x.K(x\ x))\ (\lambda x.K(x\ x))$ and this to $K\ ((\lambda x.K(x\ x))\ (\lambda x.K(x\ x)))$. Continuing this, the terms

$$K\ (K\ \ldots(K((\lambda x.K(x\ x))\ (\lambda x.K(x\ x))))\ldots)$$

may be reached. It is easy to see that, now permitting also stop-reductions, the answers that can be reached by evaluation are of the form $\lambda x_1.\lambda x_2 \ldots \lambda x_n.r$, where $r = \circledcirc$, or $r = (\circledcirc\ r')$. It is easy o verify in this calculus that $\circledcirc \simeq_b (\circledcirc\ r')$, hence we can simply use a standardizing reduction that only reaches the answers of the form $\lambda x_1.\lambda x_2 \ldots \lambda x_n.\circledcirc$.

The second term $Y_2\ K$ reduces to $((\lambda xy.K(x\ y\ y))\ (\lambda xy.K(x\ y\ y))\ (\lambda xy.K(x\ y\ y)))$ This term to $K\ ((\lambda xy.K(x\ y\ y))\ (\lambda xy.K(x\ y\ y))\ (\lambda xy.K(x\ y\ y)))$

Again, the possible answers are $\lambda x_1.\lambda x_2 \ldots \lambda x_n.r$, where $r = \circledcirc$, or $r = (\circledcirc\ r')$. We use the same standardizing reduction that only reaches the answers of the form $\lambda x_1.\lambda x_2 \ldots \lambda x_n.\circledcirc$. Now, the definition of $\lesssim_b$ together with Theorem 7.3 shows that the terms are mutually similar.

8.2 A Tiny Lazy Approximation Calculus with Constructors

We extend the approximation calculus for a lazy call-by-need non-deterministic calculus as defined in [Man05b] to constructors and case, following the same guidelines as in subsection 8.1 as follows. The language has application, `let`, `pick`, λ, `seq`, `case` and all the constructors as operators. The canonical operators are λ and the constructors. In variations of the language, several `case`-operators may be used, one for each data type.

$\texttt{let}\ x = v\ \texttt{in}\ s$	$\to s[v/x]$	if v is a pseudo-value
s	$\to \circledcirc$	if s is not a value
$\texttt{pick}\ s\ t$	$\to s$	
$\texttt{pick}\ s\ t$	$\to t$	
$\texttt{seq}\ v\ t$	$\to t$	if v is an answer
s	$\to \circledcirc$	if s is not a pseudo-value
$(\lambda x.s\ v)$	$\to s[v/x]$	if v is a pseudo-value
$(\texttt{case}\ (c\ s_1 \ldots\ s_n)$		
$\quad \text{of}\ \ldots (c\ x_1 \ldots\ x_n \to t) \ldots)$	$\to t[\overline{s_i}/\overline{x_i}]$	if s_i are pseudo-values

Values are are constructed from constructors and abstractions. There may be different kinds of case-rules. The slightly modified encoding as in Example 6.3 is used, i.e., for 0-ary constructors, a higher-order operand is used for the continuation, and the reduction replaces the dummy-variable by a pseudo-value. Again, we can apply Theorem 6.13 and obtain the $\lesssim_b$ is well-behaved.

Consider $s = \texttt{Cons}\ ((\texttt{pick}\ 0\ 1), \texttt{Nil})$ and $t = \texttt{pick}\ ((\texttt{Cons}\ 0\ \texttt{Nil}), (\texttt{Cons}\ 1\ \texttt{Nil}))$ as an example for two terms, for which $s \simeq_b t$ is easy to verify, and hence $s \simeq_c t$ holds.

Note that the call-by-need non-deterministic calculus that we are targeting at is different from the calculus above insofar as the reduction rules are different and a normal-order reduction is defined (see e.g. [SSSS04]).

We will give a sketch, how an optimized reduction may be obtained from a normal order reduction in the call-by-need calculus: Given a closed term s, the reductions will be roughly as follows (of course the reduction is in general non-deterministic):

1. Compute a weak head normal form of s. Usually, a weak head normal form is of the form $\texttt{let}\ x_1 = t_1\ \texttt{in}\ (\texttt{let}\ x_2 = t_2\ \texttt{in} \ldots\ldots \texttt{in}\ v)$, where v is an abstraction or a term of the form $c(\overline{r_i})$, where r is a constructor (i.e. canonical).
2. There are two different directions, where the reduction has to be continued: i. Reducing every t_i to WHNF, and ii. reducing r_i in the term $c(\overline{r_i})$ to WHNF.
3. Since this may go on indefinitely, there may be a bound or some other decision to stop further reduction, and to replace the unevaluated subterms by $\circledcirc$, and after this, perform replacements of the let-bound variables, i.e. the first rule of the rules above, until all top-lets are eliminated. The same has to be performed within the constructor term $c(\overline{r_i})$.
4. The result must be a pseudo-value.

8.3 A Tiny Lazy Approximation Calculus with Constructors and amb

It is no problem to replace in the previous subsection the `pick`-rules by the following two `amb`-rules.

$$\texttt{amb}\ v\ t \to v \text{ if } v \text{ is an answer}$$
$$\texttt{amb}\ t\ v \to v \text{ if } v \text{ is an asnwer}$$

Also in this case, theorem 6.13 is applicable.

Using an example from [How89], the terms $(\texttt{amb}\ \lambda x.0\ \lambda x.1)$ and $(\lambda x.\texttt{amb}\ 0\ 1)$ are in our formulation neither $\simeq_c$ nor $\lesssim_b$. Here the difference between the call-by-name non-determinism in [How89] and our call-by-need non-determinism becomes visible. This is strongly connected to our restriction of the open extension to pseudo-values, i.e. to allow only copying pseudo-values in reductions.

As a positive example, it is not hard to see that $(\texttt{amb}\ s\ t) \lesssim_b (\texttt{amb}\ t\ s)$ for closed terms s, t, which implies $(\texttt{amb}\ s\ t) \simeq_c (\texttt{amb}\ t\ s)$ by theorem 6.13, but to prove this using only the tools of contextual equivalence is rather hard.

8.4 Application to Abramsky's Lazy Lambda Calculus

The language has application and λ as operators, where λ is canonical. In general, one rule is sufficient:

$$((\lambda x.s)\ t) \to s[t/x]$$

Our approach yields ${\lesssim_b}^o \subseteq \lesssim_c$ (see [Abr90]). The set of pseudo-values is defined using the lazy variant of SHOCS. There is also a non-terminating term Ω available. The only difference between the call-by-value and the lazy variant is in the definition of the pseudo-values which may contain Ω as surface subterms in the lazy variant, but not in the call-by-value variant.

8.5 Application to a Call-By-Value Lambda Calculus

The language has application, and λ as operators. In general, one rule is sufficient:

$$(\lambda x.s\ v) \to s[v/x] \text{ if } v \text{ is a value}$$

It is clear that also in this case, we have ${\lesssim_b}^o \subseteq \lesssim_c$. There is a nonterminating term Ω available.

9 Conclusion and Further Research

We have shown that in a large class of higher-order calculi, in particular in the approximation variants of call-by-need, non-deterministic lambda-calculi, $\lesssim_b$ is well-behaved, i.e., similarity is a precongruence. Applying the theorem on schematic higher-order computation systems for a given calculus is rather simple and does not need any deep knowledge of Howe's machinery.

Given a calculus with a normal-order reduction, our approach in general requires a preparation, insofar as a termination-equivalent approximation calculus is required, since our approach cannot directly treat normal-order reduction.

Possible directions of further research are to investigate the relationship between L and its approximation variant L_A, for specific calculi and if possible for classes of calculi; to investigate calculi with a recursive let; to apply the method to non-deterministic call-by-need calculi where a must-convergence is part of the definition of the contextual preorder as in [MSC99]; and also to adapt the method to typed languages.

Acknowledgement

We thank David Sabel for discussions and reading earlier versions of the paper.

References

Abr90. Samson Abramsky. The lazy lambda calculus. In D. A. Turner, editor, *Research Topics in Functional Programming*, Univ. of Texas at Austin Year of Programming Series, chapter 4, pages 65–116. Addison-Wesley, 1990.

DP92. B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, 1992.

Gor94. Andrew D. Gordon. A tutorial on co-induction and functional programming. In *Functional Programming, Glasgow 1994*, Workshops in Computing, pages 78–95. Springer-Verlag, 1994.

Gor99. Andrew D. Gordon. Bisimilarity as a theory of functional programming. *TCS*, 228(1-2):5–47, 1999.

How89. Douglas J. Howe. Equality in lazy computation systems. In *Proceedings, Fourth LICS*, pages 198–203. IEEE Computer Society Press, 1989.

How96. Douglas J. Howe. Proving congruence of bisimulation in functional programming languages. *Information and Computation*, 124(2):103–112, 1996.

KvOvR93. J. W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems: introduction and survey. *TCS*, 121(1–2):279–308, 1993.

Las98. Søren Bøgh Lassen. *Relational Reasoning about Functions and Nondeterminism.* PhD thesis, University of Aarhus, Department of Computer Science, Ny Munkegade, building 540, DK-8000 Aarhus C, 1998.

Man05a. Matthias Mann. Congruence of bisimulation in a non-deterministic call-by-need lambda calculus. *Electronic Notes in Theoretical Computer Science*, 128(1):81–101, 2005.

Man05b. Matthias Mann. *A Non-Deterministic Call-By-Need Lambda Calculus: Proving Simlarity a Precongruence by an Extension of Howe's Method to Sharing.* PhD thesis, J.W.Goethe-Universität, Frankfurt, Germany, 2005.

Mil71. Robin Milner. An algebraic definition of simulation between programs. In D. C. Cooper, editor, *Proc. 2nd IJCAI*, pages 481–489, London, 1971. William Kaufmann.

MSC99. Andrew K.D. Moran, David Sands, and Magnus Carlsson. Erratic fudgets: A semantic theory for an embedded coordination language. In *Coordination '99*, volume 1594 of *LNCS*, pages 85–102. Springer-Verlag, 1999.

NNH99. Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Analysis.* Springer-Verlag, 1999.

Par81. David Park. Concurrency and automata on infinite sequences. In Peter Deussen, editor, *Theoretical Computer Science*, volume 104 of *LNCS*, pages 167–183. Springer, 1981.

San97. David Sands. From SOS rules to proof principles. Technical report, Chalmers University of Technology and Göteborg University, 1997.

SSSS04. Manfred Schmidt-Schauß, Marko Schütz, and David Sabel. On the safety of Nöcker's strictness analysis. Technical Report Frank-19, Institut für Informatik. J.W.Goethe-University, 2004.