Applicative May- and Should-Simulation in the Call-by-Value Lambda Calculus with AMB

Manfred Schmidt-Schauß and David Sabel

Goethe University, Frankfurt, Germany
{schauss,sabel}@ki.informatik.uni-frankfurt.de

Technical Report Frank-54

Research group for Artificial Intelligence and Software Technology
Institut für Informatik,
Fachbereich Informatik und Mathematik,
Johann Wolfgang Goethe-Universität,
Postfach 11 19 32, D-60054 Frankfurt, Germany

April 25, 2014

Abstract. Motivated by the question whether sound and expressive applicative similarities for program calculi with should-convergence exist, this paper investigates expressive applicative similarities for the untyped call-by-value lambda-calculus extended with McCarthy's ambiguous choice operator amb. Soundness of the applicative similarities w.r.t. contextual equivalence based on may- and should-convergence is proved by adapting Howe's method to should-convergence. As usual for nondeterministic calculi, similarity is not complete w.r.t. contextual equivalence which requires a rather complex counter example as a witness. Also the call-by-value lambda-calculus with the weaker nondeterministic construct erratic choice is analyzed and sound applicative similarities are provided. This justifies the expectation that also for more expressive and call-by-need higher-order calculi there are sound and powerful similarities for should-convergence.

1 Introduction

Our motivation for investigating program equivalences is to show correctness of program optimizations, more generally of program transformations, and also to get more knowledge of program semantics, since the induced equivalence classes can be viewed as the semantics of the program.

A foundational notion of equality of higher-order programs is contextual equivalence, which holds for two expressions s, t, if the evaluation of program $P[s]$ (may-)terminates successfully if and only if the evaluation of program $P[t]$ (may-)terminates successfully, for all programs $P[\cdot]$. Here we denote by $P[t]$ the program P, where the expression s is replaced by t. For concurrent and/or nondeterministic languages, the situation is a bit more complex, since contextual equivalence based only on successful may-termination is too weak, since it ignores paths that lead to errors, nontermination or deadlocks. There are proposals to remedy this weakness by adding another test: either a must-convergence test, where the test is that every possible evaluation is finite; another proposal is should-convergence, where the test only requests that for every (finite) reduction sequence there is always a possible may-termination. Contextual equivalence based on the combination of may- and should-convergence has been used for several extended, nondeterministic lambda calculi e.g. [3, 24], for process calculi and algebras [9, 5, 23], and also for concurrent lambda calculi that model real concurrent programming languages e.g. Concurrent Haskell, STM Haskell and Alice ML (see [20, 25–27]).

Although contextual equivalence provides a natural notion of program equivalence, proving expressions to be contextually equivalent is usually hard, since all program contexts need to be taken into account. Establishing equivalence proofs is often easier using an applicative (bi)-similarity. For may-convergence, applicative (bi)similarity is the coinductive test consisting of evaluating the expressions to abstractions, applying them to arguments, and showing that the resulting expressions are again applicative (bi)similar.

It is known that applicative (bi)similarities in many (usually deterministic) cases are sound and complete for contextual equivalence (see *e.g.* [1, 7]). On the other hand, there are also some negative results when more expressive and complex languages are considered, e.g. applicative similarity (for may-convergence) is unsound in impure lambda calculi with direct storage modifications [17, 29] and also in nondeterministic languages with recursive bindings [28].

While there are several approaches for an applicative similarity for must-convergence (*e.g.* [21, 13, 12, 10]), to the best of our knowledge, no notion of applicative similarity for should-convergence has been studied. So in this paper we will make a first step to close this gap and investigate a notion of applicative similarity for should-convergence.

We choose a rather small calculus for our foundational investigation to not get sidetracked by the syntactic complexity of the calculus. Hence, we investigate the untyped call-by-value lambda calculus extended by the nondeterministic primitive `amb`. We choose McCarthy's `amb`-operator[18], since its implementation requires concurrency: `amb` s t can be implemented by executing two concurrent threads – one evaluates s and the other one evaluates t, and the first result obtained from one of the two threads is used as the result for `amb` s t. Clearly, if both threads return a result, then the program is free to choose one of them. In a concrete implementation this will depend on the scheduling of the threads. Semantically, any (fair) scheduling must be allowed to ensure the correct implementation of `amb`. The operator `amb` is (locally) bottom-avoiding, *i.e.* speaking denotationally where $\bot$ represents diverging programs, `amb` $\bot$ s and `amb` s $\bot$ are equal to s, and for the case $s \neq \bot \neq t$ the `amb`-operator may freely choose between s and t, *i.e.* then $(\texttt{amb}\ s\ t) \in \{s, t\}$.

The `amb`-operator is also very expressive compared to other nondeterministic operators, *e.g.* using `amb` one can encode an erratic choice which chooses arbitrarily between its arguments, a demonic choice which is the strict variant of erratic choice and requires termination of both of its arguments before choosing between the arguments, and a parallel or. Also semantically, `amb` is challenging, since usual semantic properties do not hold for calculi with `amb`, *e.g.* nonterminating programs are not least elements w.r.t. the ordering of contextual semantics. A further reason for analyzing the calculus with `amb` is that it is being studied for several decades (e.g. [18, 2, 19, 13, 11, 10, 14]) and for the contextual equivalence with may- and must-convergence it is a long standing open question whether a sound applicative similarity exists (see e.g.[10]). A negative result is provided by [14], however it requires a typed calculus and the given counterexample is no longer valid if should-convergence is used instead of must-convergence.

Results. Our main theorem (Main Theorem 3.6) states that an expressive applicative similarity is sound for a contextual equivalence defined as a conjunction of may- and should-contextual equivalence, in the untyped call-by-value calculus with `amb`. The proof is an adaption of Howe's method [7, 8, 22] to should-convergence. We also show that the applicative similarity is not complete *w.r.t.* contextual equivalence by providing a counter-example. We also explore and discuss other possible definitions of applicative similarity and compare them to our definition. Finally, we consider the call-by-value lambda calculus with erratic choice (which is weaker than `amb`) and show that the coarser applicative similarity for may- and should-convergence (called convex similarity) is sound in the calculus with choice, but unsound in the calculus with `amb`.

Outline. In Sect. 2 we introduce the call-by-value lambda-calculus with `amb`, and in Sect. 3 we define the applicative similarities for may- and should-convergence, state our main theorem, and discuss other definition of applicative similarity. The proof of the main theorem is accomplished in Sect. 4. In Sect. 5 we consider the call-by-value calculus with erratic choice and show soundness of applicative similarity for this calculus. We conclude in Sect. 6.

Variables: $x, x_i \in \mathcal{V}$
Expressions: $s, t \in \mathit{Expr}_{LCA} ::= x \mid \lambda x.s \mid (s\ t) \mid (\texttt{amb}\ s\ t)$
Values: $v, v_i \in \mathit{Val} ::= \lambda x.s$
Contexts: $C, C_i \in \mathbb{C}_{LCA} ::= [\cdot] \mid \lambda x.C \mid (C\ s) \mid (s\ C) \mid (\texttt{amb}\ C\ s) \mid (\texttt{amb}\ s\ C)$
Evaluation contexts: $E \in \mathbb{E} ::= [\cdot] \mid (E\ s) \mid (v\ E) \mid (\texttt{amb}\ E\ s) \mid (\texttt{amb}\ s\ E)$
Reduction rules:
(cbvbeta) $((\lambda x.s)\ (\lambda y.t)) \to s[(\lambda y.t)/x]$ where $FV(\lambda y.t) \cap BV(\lambda x.s) = \emptyset$
(ambl) $(\texttt{amb}\ (\lambda x.s)\ t) \to (\lambda x.s)$
(ambr) $(\texttt{amb}\ t\ (\lambda x.s)) \to (\lambda x.s)$
Call-by-value reduction: $\dfrac{s \to t,\ \text{by (cbvbeta), (ambl) or (ambr)} \qquad E \in \mathbb{E}}{E[s] \xrightarrow{LCA} E[t]}$

Fig. 1. Syntax and Operational Semantics of *LCA*

2 Call-by-Value AMB Lambda-Calculus

We introduce the call-by-value lambda-calculus with the amb-operator, and define the contextual semantics based on may- and should-convergence.

Let $\mathcal{V}$ be an infinite set of variables. The syntax of expressions and values of the calculus *LCA* is shown in Fig. 1. In $\lambda x.s$ variable x becomes bound in s. With $FV(s)$ ($BV(s)$, resp.) we denote the set of free (bound resp.) variables of expression s, which are defined as usual. If $FV(s) = \emptyset$ then s is called *closed*, otherwise s is an *open* expression. Note that values $v \in \mathit{Val}$ include all abstractions (also open ones). We assume the distinct variable convention to hold, *i.e.* bound names are pairwise distinct and $BV(s) \cap FV(s) = \emptyset$. This convention can always be fulfilled by applying α-renamings. Contexts $C, C_i \in \mathbb{C}_{LCA}$ (see Fig. 1) are expressions where one subexpression is replaced by a hole, denoted with $[\cdot]$. With $C[s]$ we denote the expression where in C the hole is replaced by expression s.

The reduction rules (cbvbeta), (ambl) and (ambr) and the call-by-value small-step reduction $\xrightarrow{LCA}$ are defined in Fig. 1. Call-by-value reduction applies the reduction rules inside call-by-value evaluation contexts $E \in \mathbb{E}$. With $\xrightarrow{LCA,*}$ we denote the reflexive-transitive closure of $\xrightarrow{LCA}$. The reduction is non-deterministic, i.e. the arguments of amb can be reduced non-deterministically in any sequence, and if one argument is already evaluated to an abstraction, then it is also permitted to project the amb-expression to this argument.

Definition 2.1 (May- and Should-Convergence). *If $s \xrightarrow{LCA,*} \lambda x.s'$ for some abstraction $\lambda x.s'$, then we say s* may-converges *and write $s{\downarrow}$, otherwise s is* must-divergent, *denoted as $s{\Uparrow}$. If $s \xrightarrow{LCA,*} \lambda x.s'$ then we also write $s{\downarrow}\lambda x.s'$.*

If for all s' with $s \xrightarrow{LCA,*} s'$, also $s'{\downarrow}$ holds, then we say s should-converges *and write $s{\Downarrow}$, and otherwise s* may-diverges *(denoted by $s{\uparrow}$). Note that $s{\uparrow}$ iff there is an expression s', such that $s'{\Uparrow}$ and $s \xrightarrow{LCA,*} s'$.*

Definition 2.2 (Contextual Preorder & Equivalence). *For $\xi \in \{\downarrow, \Downarrow, \uparrow, \Uparrow\}$ the* contextual ξ-preorder $\leq_\xi$ *and* contextual ξ-equivalence *are defined as*

- *$s \leq_\xi t$ iff for all $C \in \mathbb{C}_{LCA}$* s.t. *$C[s]$ and $C[t]$ are closed: $C[s]\xi \implies C[t]\xi$.*
- *$s \sim_\xi t$ iff $s \leq_\xi t$ and $t \leq_\xi s$.*

Contextual preorder *$\leq_{LCA}$ is defined by $s \leq_{LCA} t$, iff $s \leq_\downarrow t$ and $s \leq_\Downarrow t$; and* contextual equivalence *$\sim_{LCA}$ is defined by $s \sim_{LCA} t$, iff $s \sim_\downarrow t$ and $s \sim_\Downarrow t$.*

Some abbreviations for expressions that we will use in later examples are $\Omega = (\lambda x.(x\ \ x))\ (\lambda x.(x\ \ x))$, $\mathit{Id} = \lambda x.x$, $\mathit{True} = \lambda x.\lambda y.x$, $\mathit{False} = \lambda x.\lambda y.y$, $Y = \lambda f.(\lambda x.f\ \lambda z.(x\ x\ z))\ (\lambda x.f\ \lambda z.(x\ x\ z))$, $\mathit{Top} = (Y\ \ \mathit{True})$. We will also write $\lambda x_1, x_2, \ldots, x_n.s$ abbreviating nested abstractions $\lambda x_1.\lambda x_2.\ldots.\lambda x_n.s$.

The given operational semantics does not take fairness into account, *e.g.* call-by-value reduction may reduce the left argument in $\texttt{amb}\ \Omega\ Id \xrightarrow{LCA} \texttt{amb}\ \Omega\ Id$ infinitely often ignoring the right argument Id. So the bottom-avoidance of the amb-operator is not fully captured by our operational semantics. However, the convergence predicates may- and should-convergence and thus also the contextual semantics capture this behavior, *i.e.* if we restrict the allowed reduction sequences to fair ones (*i.e.* no redex is ignored infinitely often in an infinite reduction sequence), then the corresponding predicates for may- and should-convergence are identical to our predicates, *i.e.* should-convergence already has this kind of fairness built-in (see *e.g.* [24]). So our operational semantics is a simplification (which greatly simplifies reasoning), but all of our results also hold for an operational semantics which includes the fairness requirement.

The amb-operator is more expressive than a lot of other nondeterministic operators. E.g., amb can encode *erratic choice* which freely chooses between its two arguments and thus we will use *choice* $s\ t$ as an abbreviation for $(\texttt{amb}\ (\lambda x.s)\ (\lambda x.t))\ Id$, where x is a fresh variable. Also a *demonic choice* operator *dchoice* is expressible, which requires termination of both of its arguments before choosing between them: $dchoice\ s\ t := (\texttt{amb}\ (\lambda x, y.x)\ (\lambda x, y.y))\ s\ t$.

Unlike calculi with erratic or demonic choice, in LCA the inequation $s \leq_{\Downarrow} t$ implies $t \leq_{\downarrow} s$, since there is the so-called "bottom-avoiding context" which can be used to test for must-divergence using the should-convergence test. This also implies that contextual equivalence and $\sim_{\Downarrow}$ coincide.

Proposition 2.3. *$\leq_{\Downarrow} \subseteq \leq_{\Uparrow}$ and thus $\leq_{LCA} \subseteq \sim_{\downarrow}$ as well as $\sim_{LCA} = \sim_{\Downarrow}$.*

Proof. For the context $BA := (\texttt{amb}\ ((\lambda x.\lambda y.\Omega)\ [\cdot])\ Id)\ Id$ and any LCA-expression s the equivalence $BA[s]{\Downarrow} \iff s{\Uparrow}$ holds: if $s{\Uparrow}$, then the amb-expression can only evaluate to its right argument Id, and thus $BA[s]$ is should-convergent in this case. If $s{\downarrow}$, then the reduction sequence $BA[s] \xrightarrow{LCA,*} (\texttt{amb}\ (\lambda y.\Omega)\ Id)\ Id \xrightarrow{LCA,*} \Omega$ shows $BA[s]{\uparrow}$. Now let $s \leq_{\Downarrow} t$ and assume $s \not\leq_{\Uparrow} t$. Then there exists a context C *s.t.* $C[s], C[t]$ are closed and $C[s]{\Uparrow}$ but $C[t]{\downarrow}$. Then $BA[C[s]], BA[C[t]]$ are closed and $BA[C[s]]{\Downarrow}$ and $BA[C[t]]{\uparrow}$, which contradicts $s \leq_{\Downarrow} t$. Thus our assumption was wrong and $s \leq_{\Uparrow} t$ must hold. □

3 Applicative Similarities for LCA

In this section we define applicative similarities for may- and should-convergence in LCA. Then we present our main theorem: the applicative similarities are sound for contextual preorder. We also discuss our definitions and also consider and analyze alternative definitions of similarity. Due to its complexity, the proof of the main theorem is not included in this section, but given in the subsequent section. We use several binary relations on expressions. Sometimes the relations are defined on closed expressions only, and thus we deal with their extensions to open expressions and vice versa with the restrictions to closed expressions:

Definition 3.1. *For a binary relation η on closed LCA-expressions, η^o is the* open value-extension *on LCA: For (open) LCA-expressions s_1, s_2, the relation $s_1\ \eta^o\ s_2$ holds, if for all value-substitutions σ,* i.e. *that replace the free variables in s_1, s_2 by closed abstractions, and where $\sigma(s_1), \sigma(s_2)$ are closed, the relation $\sigma(s_1)\ \eta\ \sigma(s_2)$ holds. Conversely, for a binary relation μ on open expressions, $(\mu)^c$ is its restriction to closed expressions.*

Lemma 3.2. *Let η be a binary relation on closed expressions, and μ be a binary relation on open expressions. Then 1. $((\eta)^o)^c = \eta$, and 2. $s\ \eta^o\ t$ implies $\sigma(s)\ \eta^o\ \sigma(t)$ for any value-substitution σ, and 3. $\mu \subseteq ((\mu)^c)^o$ is equivalent to: $\forall s, t$ and all closing value-substitutions σ: $s\ \mu\ t \implies \sigma(s)\ \mu\ \sigma(t)$*

3.1 Applicative Similarities for May- and Should-Convergence

We define applicative similarity $\preccurlyeq_\downarrow$ for may-convergence and applicative similarity $\preccurlyeq_\uparrow$ for should-convergence (where in fact its negation may-divergence is used). Also mutual similarities and applicative bisimilarities are defined.

Definition 3.3. *We will define operators F_α on binary relations of closed expressions, where α is a name or a mark. The corresponding* similarity, *denoted as $\preccurlyeq_\alpha$ is the greatest fixpoint* $\mathrm{gfp}(\mathrm{F}_\alpha)$ *of F_α, and the* mutual similarity *is $\approx_\alpha := \preccurlyeq_\alpha \cap \succcurlyeq_\alpha$. If F_α is symmetric, then it is a* bisimilarity, *denoted as $\simeq_\alpha$.*

We always define monotone operators F_α, hence the greatest fixpoints exist. For closed s, t and a binary relation η on closed expressions let $LR(s, t, \eta)$ be the condition: $s{\downarrow}\lambda x.s' \implies \big(\exists \lambda x.t'$ with $t{\downarrow}\lambda x.t'$ and $s'\ \eta^o\ t'\big)$.

Definition 3.4 (Similarities for LCA). *On closed expressions we define:*

May-Similarity in LCA, $\preccurlyeq_\downarrow := \mathrm{gfp}(\mathrm{F}_\downarrow)$: *Let $s\ F_\downarrow(\eta)\ t$ hold iff $LR(s, t, \eta)$.*

Should-Similarity in LCA, $\preccurlyeq_\uparrow := \mathrm{gfp}(\mathrm{F}_\uparrow)$:
Let $s\ F_\uparrow(\eta)\ t$ hold iff $s\uparrow \implies t\uparrow$, $t \preccurlyeq_\downarrow s$ and $LR(s, t, \eta)$.

Should-Bisimilarity in LCA, $\simeq_\Downarrow := \mathrm{gfp}(\mathrm{F}_\Downarrow)$:
Let $s\ F_\Downarrow(\eta)\ t$ hold iff $s\uparrow \iff t\uparrow$, $LR(s, t, \eta)$, and $LR(t, s, \eta)$.

Since $\mathrm{gfp}(\mathrm{F}_\alpha) := \bigcup\{\eta \mid \eta \subseteq \mathrm{F}_\alpha(\eta)\}$ by the Knaster-Tarski-Theorem on fixpoints, the following principle of coinduction holds (see e.g.[4, 6]):

Proposition 3.5 (Coinduction). *If a relation η on closed expressions is F_α*-dense, i.e. $\eta \subseteq F_\alpha(\eta)$*, then $\eta \subseteq \preccurlyeq_\alpha$, and also $(\eta)^o \subseteq (\preccurlyeq_\alpha)^o$ holds.*

We now present our main theorem, i.e. soundness of may- and should-similarity and also should-bisimilarity. Here we state it for the open extensions of the relations, however it also holds for the relations on closed expressions and the restriction of contextual preorders and equivalence on closed expressions.

Main Theorem 3.6 *The similarities $\preccurlyeq_\downarrow^o$ and $\preccurlyeq_\uparrow^o$ are precongruences, the mutual similarities $\approx_\downarrow^o$, $\approx_\uparrow^o$, and the bisimilarity $\simeq_\Downarrow^o$ are congruences. Moreover, the following soundness results hold:*

1. $\preccurlyeq_\downarrow^o \ \subset\ \leq_\downarrow$ *and* $\approx_\downarrow^o \ \subset\ \sim_\downarrow$.
2. $\preccurlyeq_\uparrow^o \ \subset\ \geq_{LCA}$ *and* $\approx_\uparrow^o \ \subset\ \sim_{LCA}$.
3. $\simeq_\Downarrow^o \ \subseteq\ \approx_\uparrow^o \ \subset\ \sim_{LCA}$.

We prove Main Theorem 3.6 in Sect. 4: the results for may-similarity $\preccurlyeq_\downarrow$ are standard and a sketch is given in Theorem 4.6, the full proof is given in Appendix B.The results for should-similarity $\preccurlyeq_\uparrow$ are proved in Theorems 4.14 and 4.15. For should-bisimilarity the inclusion $\simeq_\Downarrow^o \subseteq \approx_\uparrow^o$ holds, since $\simeq_\Downarrow$ is $F_\uparrow$-dense. The congruence property for $\simeq_\Downarrow$ requires a separate proof which is in Appendix C. Strictness of the inclusions will be proved by counter-examples.

3.2 Discussion on Similarities for Should-Convergence

In this section we discuss other variants of should-similarity for LCA. As we show, the first and second are unsound, the third may be a slight generalization, and the status of the fourth is unknown.

Definition 3.7. Naive Should-Similarity in LCA, $\preccurlyeq_{\uparrow_N} := \mathrm{gfp}(\mathrm{F}_{\uparrow_N})$:
Let $s\ F_{\uparrow_N}(\eta)\ t$ hold iff $s\uparrow \implies t\uparrow$ and $LR(s, t, \eta)$.

Convex Should-Similarity in LCA**,** $\preccurlyeq_{\uparrow_X} := \mathrm{gfp}(\mathrm{F}_{\uparrow_X})$**:**
Let $s\ F_{\uparrow_X}(\eta)\ t$ *hold iff* $s\uparrow \implies t\uparrow$, $t \preccurlyeq_{\downarrow} s$, *and* $t\Downarrow \implies LR(s,t,\eta)$.
Should-Similarity in LCA**, variant** $\preccurlyeq_{\uparrow_C} := \mathrm{gfp}(\mathrm{F}_{\uparrow_C})$**:**
Let $s\ F_{\uparrow_C}(\eta)\ t$ *hold iff* $s\uparrow \implies t\uparrow$, $t \leq_{\downarrow} s$, *and* $LR(s,t,\eta)$.
Should-Similarity in LCA**, variant** $\preccurlyeq_{\uparrow'} := \mathrm{gfp}(\mathrm{F}_{\uparrow'})$**:**
Let $s\ F_{\uparrow'}(\eta)\ t$ *hold iff* $s\uparrow \implies t\uparrow$, $LR(s,t,\eta)$, *and* $LR(t,s,\eta^{-1})$.

Obviously, (*choice False True*) $\not\leq_{\uparrow}$ *True* using the context ($[\cdot]$ *Id* Ω). This suggests the naive should-similarity $\preccurlyeq_{\uparrow_N}$ which, however, is insufficient:

Lemma 3.8. $\preccurlyeq_{\uparrow_N}$ *is unsound* w.r.t. $\leq_{\uparrow}$.

Proof. While $Id \preccurlyeq_{\uparrow_N} \lambda x.choice\ x\ Id$ holds, we have $(Y\ (\lambda x.choice\ x\ Id)\ Id)\Downarrow$, but $(Y\ Id\ Id)\Uparrow$. Thus $\preccurlyeq_{\uparrow_N}$ is not a precongruence and not sound *w.r.t.* $\leq_{\uparrow}$. □

In the definition of $\preccurlyeq_{\uparrow}$ this is the reason for the additional condition $t \preccurlyeq_{\downarrow} s$ inside $F_{\uparrow}$ (which in fact implies $s \approx_{\downarrow} t$, since $\preccurlyeq_{\uparrow} \subset \preccurlyeq_{\downarrow}$). Further generalizing the definition of $\preccurlyeq_{\uparrow}$ by requiring the recursive test to hold only if the right expression is should-convergent leads to the convex should-similarity, $\preccurlyeq_{\uparrow_X}$, which is analogous to the definition of so-called (unsound) "convex similarity" in [19] for a call-by-name lambda-calculus with `amb`, but using must-convergence instead of should-convergence. However, also for LCA the similarity $\preccurlyeq_{\uparrow_X}$ is unsound:

Lemma 3.9. $\preccurlyeq_{\uparrow_X}$ *is unsound w.r.t.* $\leq_{\uparrow}$.

Proof. Let $s_1 := \texttt{amb}\ (\lambda x.\Omega)\ (\lambda x,y,z.\Omega)$ and $s_2 := \texttt{amb}\ s_1\ (\lambda x,y.\Omega)$. Then $s_2 \preccurlyeq_{\uparrow_X} s_1$, but $s_2 \not\leq_{\uparrow} s_1$, since for the context $C := (\texttt{amb}\ ([\cdot]\ \texttt{Id})\ \texttt{Id})\ \texttt{Id}$ we have $C[s_2] \xrightarrow{LCA,*} \Omega$ and thus $C[s_2]\uparrow$, but $C[s_1]\Downarrow$.

For calculi with only erratic or demonic choice, $\preccurlyeq_{\uparrow_X}$ is sound (see Sect. 5).

A further generalization of the successful similarity $\preccurlyeq_{\uparrow}$ by replacing the $t \preccurlyeq_{\downarrow} s$ condition by $t \leq_{\downarrow} s$ leads to $\preccurlyeq_{\uparrow_C}$, for which it is easy to see that $\preccurlyeq_{\uparrow} \subseteq \preccurlyeq_{\uparrow_C}$, and we conjecture that it is sound, but a soundness proof would require at least a ciu-Lemma for LCA. As another strengthening of the conditions inside $F_{\uparrow_N}$ we added the condition $LR(t,s,\eta^{-1})$ resulting in the should-similarity $\preccurlyeq_{\uparrow'}$ We did neither find a soundness proof for $\preccurlyeq_{\uparrow'}$, since the condition $\forall t \downarrow \lambda x.t' \exists s \downarrow \lambda x.s'$ is inappropriate for Howe's method, nor did we find a counter-example showing unsoundness, so we leave soundness of $\preccurlyeq_{\uparrow'}$ as an open question.
Our results imply that the following properties hold for $\preccurlyeq_{\uparrow'}$:

Lemma 3.10. $\preccurlyeq_{\uparrow'}\ \subseteq\ \preccurlyeq_{\downarrow}\ \subseteq\ \leq_{\downarrow}$ *and* $\simeq_{\Downarrow}\ \subseteq\ \approx_{\uparrow'}\ \subseteq\ \approx_{\downarrow}\ \subseteq\ \sim_{\downarrow}$.

Proof. The first chain of inclusions is valid, since $\preccurlyeq_{\uparrow'}$ is $F_{\downarrow}$-dense, *i.e.* $\preccurlyeq_{\uparrow'} \subseteq F_{\downarrow}(\preccurlyeq_{\uparrow'})$, and since $\preccurlyeq_{\downarrow}$ is sound for $\leq_{\downarrow}$ (Main Theorem 3.6). In the second chain, the inclusion $\simeq_{\Downarrow} \subseteq \approx_{\uparrow'}$ holds, since $\simeq_{\Downarrow} \subseteq F_{\uparrow'}(\simeq_{\Downarrow})$ and since $\simeq_{\Downarrow}$ is symmetric. The remaining inclusions follow from the first chain. □

4 Soundness Proofs for Similarity in LCA

4.1 Preliminaries on Howe's Method

In this section we will introduce the necessary notions to apply Howe' method for the soundness proofs of similarities *w.r.t.* contextual preorder and contextual equivalence in LCA. Here we employ higher order abstract syntax as *e.g.* in [7] for the proof and write $\tau(..)$ for an expression with top operator τ, which may be λ, application, or `amb`. For consistency of terminology and treatment with that in other papers such as [7], we assume that removing the top constructor λx in relations is done after a renaming. For example, $\lambda x.s\ \mu\ \lambda y.t$ is renamed to the same bound variable before further reasoning about s, t, to $\lambda z.s[z/x]\ \mu\ \lambda z.t[z/y]$ for a fresh variable z. A relation μ is *operator-respecting*, iff $s_i\ \mu\ t_i$ for $i = 1, \ldots, n$ implies $\tau(s_1, \ldots, s_n)\ \mu\ \tau(t_1, \ldots, t_n)$.In these preliminaries for Howe's method we assume that there is a preorder $\preccurlyeq$, which is a reflexive and transitive relation on closed expressions. The goal is to show that $\preccurlyeq$ is a precongruence. We then define the *Howe candidate relation* $\preccurlyeq_H$ and show its properties. Later $\preccurlyeq$ is instantiated by the may- or should-similarity or by the should-bisimilarity.

Definition 4.1. *Given a reflexive and transitive relation $\preccurlyeq$ on closed expressions, the* Howe (precongruence candidate) relation $\preccurlyeq_H$ *is a binary relation on open expressions defined inductively on the structure of the left hand expression:*

1. *If $x \preccurlyeq^o s$ then $x \preccurlyeq_H s$.*
2. *If there are expressions s, s_i, s'_i* s.t. *$\tau(s'_1, \ldots, s'_n) \preccurlyeq^o s$ with $s_i \preccurlyeq_H s'_i$ for $i = 1, \ldots, n$, then $\tau(s_1, \ldots, s_n) \preccurlyeq_H s$.*

Lemma 4.2. *We have $x \preccurlyeq_H s$ iff $x \preccurlyeq^o s$; and $\tau(s_1, \ldots, s_n) \preccurlyeq_H s$ iff there is some expression $\tau(s'_1, \ldots, s'_n) \preccurlyeq^o s$ such that $s_i \preccurlyeq_H s'_i$ for $i = 1, \ldots, n$.*

Helpful properties of $\preccurlyeq_H$ (proved in Appendix A) are:

Lemma 4.3. *The following properties hold:*

1. *$\preccurlyeq_H$ is reflexive.*
2. *$\preccurlyeq_H$ and $(\preccurlyeq_H)^c$ are operator-respecting.*
3. *$\preccurlyeq^o \subseteq \preccurlyeq_H$ and $\preccurlyeq \subseteq (\preccurlyeq_H)^c$.*
4. *$\preccurlyeq_H \circ \preccurlyeq^o \subseteq \preccurlyeq_H$.*
5. *$(v \preccurlyeq_H v' \wedge t \preccurlyeq_H t') \implies t[v/x] \preccurlyeq_H t'[v'/x]$ for values v, v'.*
6. *$s \preccurlyeq_H t$ implies that $\sigma(s) \preccurlyeq_H \sigma(t)$ for every value-substitution σ.*
7. *$\preccurlyeq_H \subseteq ((\preccurlyeq_H)^c)^o$.*
8. *If $(\preccurlyeq_H)^c = \preccurlyeq$, then $\preccurlyeq_H = \preccurlyeq^o$.*
9. *If s, t are closed, $s = \tau(s_1, \ldots, s_{\mathrm{ar}(\tau)})$ and $s \preccurlyeq_H t$ holds, then there are s'_i, such that $\tau(s'_1, \ldots, s'_{\mathrm{ar}(\tau)})$ is closed, $\forall i : s_i \preccurlyeq_H s'_i$ and $\tau(s'_1, \ldots, s'_{\mathrm{ar}(\tau)}) \preccurlyeq t$.*

As a general outline, the goal of Howe's method is to show that $\preccurlyeq_H = \preccurlyeq^o$, which implies that $\preccurlyeq^o$ is operator-respecting and hence it is a precongruence.

Lemma 4.4. *The relations $\preccurlyeq_\alpha$, $\preccurlyeq^o_\alpha$ from Definition 3.4 are reflexive and transitive. The relations $\simeq_\Downarrow$, and $\simeq^o_\Downarrow$ are equivalence relations.*

Proof. Reflexivity holds since $\eta := \{(s, s) \mid s \in \mathit{Expr}_{LCA}, s \text{ closed}\} \cup \preccurlyeq_\alpha$ satisfies $\eta \subseteq F_\downarrow(\eta)$. Transitivity holds since $\eta := \preccurlyeq_\downarrow \cup (\preccurlyeq_\downarrow \circ \preccurlyeq_\downarrow)$ satisfies $\eta \subseteq F_\downarrow(\eta)$. Similar coinduction arguments show the other claims. □

Lemma 4.5. $s \preccurlyeq^o_\alpha t \iff \lambda x.s \preccurlyeq^o_\alpha \lambda x.t$.

4.2 Soundness of May-Similarity

Theorem 4.6. *May-similarity behaves as expected: The similarity $\preccurlyeq_\downarrow$ for may-convergence is a precongruence on closed expressions and sound for $\leq^c_\downarrow$. Extending this on all expressions: $\preccurlyeq^o_\downarrow$ is a precongruence and sound for $\leq_\downarrow$.*

Proof (Sketch(see Appendix B)). Use Howe's method. Define $\preccurlyeq_{\downarrow H}$ as an extension of $\preccurlyeq_\downarrow$ using Definition 4.1. Then show that $\preccurlyeq^c_{\downarrow H}$ satisfies the fixpoint conditions for $\preccurlyeq_\downarrow$, which implies $\preccurlyeq^c_{\downarrow H} \subseteq \preccurlyeq_\downarrow$, and so $\preccurlyeq^c_{\downarrow H} = \preccurlyeq_\downarrow$, which implies the precongruence property, and $\preccurlyeq_{\downarrow H} = \preccurlyeq^o$. □

Corollary 4.7. *The mutual similarity* $\approx_{\downarrow}$ *is a congruence and sound for* $\sim^c_{\downarrow}$. *Also* $\approx^o_{\downarrow}$ *is a congruence and sound for* $\sim_{\downarrow}$.

But note that $\approx_{\downarrow}$ is not complete using a similar example as in [15]:

Proposition 4.8. $\approx^o_{\downarrow} \neq \sim_{\downarrow}$

Proof. With $F = \lambda f.\lambda z.choice\ (\lambda x.\Omega)\ ((\lambda x_1, x_2.x_1)\ (f\ z))$ one can verify that $Y\ F\ Id$ reduces to $\lambda x_1, \ldots, x_n.\Omega$ for any $n \geq 1$. The reduction sequence is: $Y\ F\ Id \to F'\ F'\ Id$ with $F' = (\lambda x.F\ (\lambda z.x\ x\ z))$.
$\to F\ (\lambda z.(F'\ F'\ z))\ Id$
$\to choice\ (\lambda x.\Omega)\ ((\lambda x_1, x_2.x_1)\ ((\lambda z, F'\ F'\ z)\ Id))$
$\to (\lambda x_1, x_2.x_1)\ (F'\ F'\ Id)$. Using a context lemma for LCA, one can show that $Y\ F\ Id \sim_{\downarrow} Top$. However, $Top \not\preccurlyeq_{\downarrow} Y\ F\ Id$, since after evaluating Top to $\lambda z.(True\ Top\ z) = v_1$, we have to choose a value $\lambda x_1, \ldots, x_n.\Omega = v_2$ of $(Y\ F\ Id)$ for a fixed number n, and applying v_1 to n arguments converges, but the application of v_2 to n arguments diverges. □

4.3 Soundness of Should-Similarity

In this section we present a proof for soundness of should-similarity, *i.e.* $\preccurlyeq^o_{\uparrow}\ \subseteq\ \leq_{LCA}$. We first show some properties of $\preccurlyeq_{\uparrow}$:

Lemma 4.9. $\preccurlyeq_{\uparrow}\ \subseteq\ \approx_{\downarrow}\ \subseteq\ \sim_{\downarrow}$ *and* $\simeq_{\Downarrow}\ \subseteq\ \approx_{\uparrow}\ \subseteq\ \sim_{\downarrow}$.

Proof. The first inclusion holds, since $\preccurlyeq_{\uparrow} \subseteq \succcurlyeq_{\downarrow}$ by definition, $\preccurlyeq_{\uparrow} \subseteq \preccurlyeq_{\downarrow}$ (since $\preccurlyeq_{\uparrow}$ is $F_{\downarrow}$-dense), and $\preccurlyeq_{\downarrow} \subseteq \leq_{\downarrow}$ by Theorem 4.6. In the second chain, the inclusion $\simeq_{\Downarrow}\ \subseteq\ \approx_{\uparrow}$ holds, since $\simeq_{\Downarrow}$ satisfies all the conditions of $F_{\uparrow}$, and since $\simeq_{\Downarrow}$ is symmetric. The remaining inclusion follows from the first chain.

The goal in the following is to show that the candidate relation $\preccurlyeq_{\uparrow H}$ derived from $\preccurlyeq_{\uparrow}$ can be treated using Howe's method to prove its soundness. Our proof relies on the precongruence property of $\preccurlyeq^o_{\downarrow}$ (which is already proved in Theorem 4.6) for the transfer of may-divergence over the candidate relation.

Definition 4.10. *The candidate relation* $\preccurlyeq_{\uparrow H}$ *is defined* w.r.t. *the relation* $\preccurlyeq_{\uparrow}$.

Lemma 4.11. $\preccurlyeq_{\uparrow H} \subseteq \approx^o_{\downarrow}$.

Proof. To show that $s \preccurlyeq_{\uparrow H} t \implies s \approx^o_{\downarrow} t$, we use induction on the structure of s. In the case $s = x$ the definition of the candidate implies $x \preccurlyeq^o_{\uparrow} t$, which implies $x \approx^o_{\downarrow} t$ by Lemma 4.9. If $s = \tau(s_1, \ldots, s_n)$, there is some $\tau(t_1, \ldots, t_n)\ \preccurlyeq^o_{\uparrow}\ t$ with $s_i \preccurlyeq_{\uparrow H} t_i$ for all i. The induction hypothesis implies $s_i \approx^o_{\downarrow} t_i$ for all i, and the congruence property of $\approx^o_{\downarrow}$ shows $\tau(s_1, \ldots, s_n) \approx^o_{\downarrow} \tau(t_1, \ldots, t_n)$. Transitivity of $\approx^o_{\downarrow}$ and $\preccurlyeq^o_{\uparrow}\ \subseteq\ \approx^o_{\downarrow}$ now shows $s = \tau(s_1, \ldots, s_n) \approx^o_{\downarrow} t$. □

Proposition 4.12. *Let* s, t *be closed expressions,* $s \preccurlyeq_{\uparrow H} t$ *and* $s{\downarrow}\lambda x.s'$. *Then there is some* $\lambda x.t'$ *such that* $t{\downarrow}\lambda x.t'$ *and* $s' \preccurlyeq_{\uparrow H} t'$.

Proof. The proof is by induction on the length of the reduction of $s{\downarrow}\lambda x.s'$.

- If $s = \lambda x.s'$, then there is some closed $\lambda x.t'$ with $s' \preccurlyeq_{\uparrow H} t'$ and $\lambda x.t' \preccurlyeq_{\uparrow}\ t$. The latter implies that there is some closed $\lambda x.t''$ with $t{\downarrow}\lambda x.t''$ and $t' \preccurlyeq^o_{\uparrow}\ t''$, and so $s' \preccurlyeq_{\uparrow H} t''$ by Lemma 4.3 (4).
- Case $s =$ `amb` $s_1\ s_2$, and $s{\downarrow}\lambda x.s'$. Then there is some closed expression `amb` $t_1\ t_2 \preccurlyeq_{\uparrow}\ t$ with $s_i\ \preccurlyeq_{\uparrow H}\ t_i$ for $i = 1, 2$. W.l.o.g. let $s_1{\downarrow}\lambda x.s'$. Then by induction, there is some $\lambda x.t'$ with $t_1{\downarrow}\lambda x.t'$ and $s' \preccurlyeq_{\uparrow H} t'$. Obviously, also `amb` $t_1\ t_2{\downarrow}\lambda x.t'$. From `amb` $t_1\ t_2\ \preccurlyeq_{\uparrow}\ t$, we obtain that there is some $\lambda x.t''$ with $t{\downarrow}\lambda x.t''$ and $t' \preccurlyeq^o_{\uparrow} t''$, which implies $s' \preccurlyeq_{\uparrow H} t''$ by Lemma 4.3 (4).

- If $s = (s_1\ s_2)$, then there is some closed $t' = (t'_1\ t'_2) \preccurlyeq_\uparrow\ t$ with $s_i \preccurlyeq_{\uparrow H}\ t'_i$ for $i = 1, 2$. Since $(s_1\ s_2){\downarrow}\lambda x.s'$ there is a reduction sequence $(s_1\ s_2) \xrightarrow{LCA,*} (\lambda x.s'_1)\ s_2 \xrightarrow{LCA,*} (\lambda x.s'_1)\ (\lambda x.s'_2) \xrightarrow{LCA} s'_1[\lambda x.s'_2/x] \xrightarrow{LCA,*} \lambda x.s'$ such that $s_i{\downarrow}\lambda x.s'_i$ for $i = 1, 2$. By induction, there are expressions $\lambda x.t''_i$ with $t'_i{\downarrow}\lambda x.t''_i$ and $s'_i \preccurlyeq_{\uparrow H}\ t''_i$. Lemma 4.3 (5) now shows $s'_1[\lambda x.s'_2/x] \preccurlyeq_{\uparrow H}\ t''_1[\lambda x.t''_2/x]$. Now we can again use the induction hypothesis which shows that there is some $\lambda x.t''$ with $t''_1[\lambda x.t''_2/x]{\downarrow}\lambda x.t''$ and $s' \preccurlyeq_{\uparrow H}\ t''$. The relation $(t'_1\ t'_2) \preccurlyeq_\uparrow\ t$ implies that $t{\downarrow}\lambda x.t_0$ with $t'' \preccurlyeq^o_\uparrow t_0$, and hence $s' \preccurlyeq_{\uparrow H} t_0$ by Lemma 4.3 (4). □

Proposition 4.13. *Let s, t be closed expressions, $s \preccurlyeq_{\uparrow H} t$ and $s\uparrow$. Then $t\uparrow$.*

Proof. The proof is by induction on the number of reductions of s to a must-divergent expression, and on the size of expressions as a second measure.

- The base case is that $s\Uparrow$. Then Lemma 4.11 shows $t\Uparrow$.
- Let $s = \texttt{amb}\ s_1\ s_2$ with $s\uparrow$. Then there is some closed expression $t' = \texttt{amb}\ t_1\ t_2$ with $s_i \preccurlyeq_{\uparrow H}\ t_i$ for $i = 1, 2$ and $\texttt{amb}\ t_1\ t_2\ \preccurlyeq_\uparrow\ t$. It follows that $s_1\uparrow$ as well as $s_2\uparrow$. Applying the induction hypothesis shows that $t_1\uparrow$ as well as $t_2\uparrow$, and hence $(\texttt{amb}\ t_1\ t_2)\uparrow$. From $\texttt{amb}\ t_1\ t_2\ \preccurlyeq_\uparrow\ t$ we obtain $t\uparrow$.
- Let $s = (s_1\ s_2)$ with $s\uparrow$. Then there is some closed expression $t' = (t_1\ t_2) \preccurlyeq_\uparrow t$ and $s_i \preccurlyeq_{\uparrow H} t_i$ for $i = 1, 2$. There are several cases:
  1. If $(s_1\ s_2) \xrightarrow{LCA,*} (s'_1\ s_2)$ and $s'_1\Uparrow$, then $s_1\uparrow$ and by the induction hypothesis also $t_1\uparrow$, and hence $t'\uparrow$, which implies $t\uparrow$.
  2. If $(s_1\ s_2) \xrightarrow{LCA,*} (\lambda x.s'_1)\ s_2 \xrightarrow{LCA,*} (\lambda x.s'_1)\ s'_2$ and $s'_2\Uparrow$, then $s_2\uparrow$ and by induction hypothesis also $t_2\uparrow$, and hence $t'\uparrow$, which implies $t\uparrow$.
  3. If $(s_1\ s_2) \xrightarrow{LCA,*} (\lambda x.s'_1)\ s_2 \xrightarrow{LCA,*} (\lambda x.s'_1)\ (\lambda x.s'_2) \xrightarrow{LCA} s'_1[\lambda x.s'_2/x] \xrightarrow{LCA,*} s_0$ where $s_0\Uparrow$. Then $s_i{\downarrow}\lambda x.s'_i$ for $i = 1, 2$ and by Proposition 4.12 there are reductions $t_i{\downarrow}\lambda x.t'_i$ for $i = 1, 2$ with $s'_i \preccurlyeq_{\uparrow H}\ t'_i$. Thus $s'_1[\lambda x.s'_2/x] \preccurlyeq_{\uparrow H}\ t'_1[\lambda x.t'_2/x]$, and hence by the induction hypothesis $t'_1[\lambda x.t'_2/x]\uparrow$. Thus $(t_1\ t_2)\uparrow$, and now $(t_1\ t_2) \preccurlyeq_\uparrow t$ implies $t\uparrow$. □

Theorem 4.14. *The relation $\preccurlyeq_\uparrow$ is a precongruence on closed expressions and $\preccurlyeq^o_\uparrow$ is a precongruence on all expressions.*

Proof. We have $\preccurlyeq_\uparrow\ \subseteq\ \preccurlyeq^c_{\uparrow H}$ by Lemma 4.3 (3). Since $\preccurlyeq^c_{\uparrow H}$ satisfies the fixpoint conditions of $\preccurlyeq_\uparrow$ (using Propositions 4.12 and 4.13), coinduction shows that $\preccurlyeq^c_{\uparrow H}\ \subseteq\ \preccurlyeq_\uparrow$. Hence, $\preccurlyeq^c_{\uparrow H}\ =\ \preccurlyeq_\uparrow$ and also $\preccurlyeq_{\uparrow H}\ =\ \preccurlyeq^o_\uparrow$.

Theorem 4.15. *$\preccurlyeq^o_\uparrow$ is sound for $\geq_{LCA}$.*

Proof. Let $s \preccurlyeq^o_\uparrow t$, and let C be a context such that $C[s], C[t]$ are closed. First assume that $C[s]\uparrow$. Theorem 4.14 shows that $C[s] \preccurlyeq^o_\uparrow C[t]$, and so $C[t]\uparrow$. Lemma 4.9 and Theorem 4.6. imply $C[s]{\downarrow} \iff C[t]{\downarrow}$. Hence $s \geq_{LCA} t$. □

Theorem 4.16. *The similarity $\preccurlyeq_\uparrow$ is incomplete for $\geq_\Downarrow$.*

Proof. We give a counterexample (details are in Appendix D): Let $A = \texttt{choice}\ \Omega\ (\lambda x.A)$, $B_0 = Top$, $B_{i+1} = \lambda x.\texttt{choice}\ \Omega\ B_i$; and $B = \texttt{choice}\ \Omega\ (\texttt{choice}\ B_0\ (\texttt{choice}\ B_1\ \ldots))$. Then $Top \simeq_\downarrow A \simeq_\downarrow B_i$ for all i and $Top \simeq_\downarrow B$. Also $B_i <_\uparrow A$ for all i. Using a context lemma for closed expressions it can be shown that $A \sim_{LCA} B$. It is easy to see that $B \preccurlyeq_\uparrow A$, but $A \not\preccurlyeq_\uparrow B$. □

Comparing s, t for $\leq_\uparrow$, the incompleteness of $\preccurlyeq_\uparrow$ cannot appear if t reduces to only finitely many abstractions.

Proposition 4.17. *Assume that s is a closed abstraction and t is a closed expression such that $s \leq_{\uparrow} t$ and there is a nonempty set $T := \{t_1, \dots, t_n\}$ of closed abstractions, such that $t{\downarrow}\lambda x.t'$ implies $\lambda x.t' \in T$. Then there is some i with $s \leq_{\uparrow} t_i$.*

Proof. Suppose this is false. Then there are contexts $C_1, \dots, C_n$, such that $C_i[s], C_i[t_i]$ are closed for all i, and for all $i = 1, \dots, n$: $C_i[s]{\uparrow}$ and $C_i[t_i]{\Downarrow}$. The context $C = (\lambda x.\mathtt{amb}\ C_1[x]\ (\mathtt{amb}\ \dots (\mathtt{amb}\ C_{n-1}[x]\ C_n[x])))\ [\cdot]$ has the property: $C[s]{\uparrow}$, but $C[t]{\Downarrow}$, which is a contradiction.

Soundness of the applicative similarities implies:

Proposition 4.18. *Let s, t be closed expressions, such that for all $\lambda x.s'$: $s{\downarrow}\lambda x.s' \iff t{\downarrow}\lambda x.s'$ (the same results modulo alpha-equivalence), and $s{\uparrow} \iff t{\uparrow}$, then $s \approx_{\uparrow} t$, and hence also $s \sim_{LCA} t$.*
If s, t are open expressions, such that for all value substitutions σ, such that $\sigma(s), \sigma(t)$ are closed: $\sigma(s){\downarrow}\lambda x.s' \iff \sigma(t){\downarrow}\lambda x.s'$ (modulo alpha-equivalence), and $\sigma(s){\uparrow} \iff \sigma(t){\uparrow}$, then $s \approx_{\uparrow}^{o} t$, and hence also $s \sim_{LCA} t$.

Corollary 4.19. *Several identities obviously hold in LCA:*

$$(\lambda x.s)\ (\lambda x.t) \sim_{LCA} s[\lambda x.t/x] \qquad (\mathtt{amb}\ \Omega\ s) \sim_{LCA} s \qquad (\mathtt{amb}\ s\ s) \sim_{LCA} s$$
$$(\mathtt{amb}\ s\ t) \sim_{LCA} (\mathtt{amb}\ t\ s) \qquad \mathtt{amb}\ s_1\ (\mathtt{amb}\ s_2\ s_3) \sim_{LCA} \mathtt{amb}\ (\mathtt{amb}\ s_1\ s_2)\ s_3$$

An example that is a bit more complex is:

Example 4.20. Let $F = \lambda f.\lambda x.\mathtt{amb}\ x\ (f\ x)$. We show that $Y\ F \sim Id$ using similarities. It is easy to see that for all closed abstrations r: $Id\ r{\downarrow}r$ and also $(Y\ F\ r){\downarrow}r' \implies r = r'$. The reduction sequences for $(Y\ F\ r)$ are as follows:
$(Y\ F\ r) \to F'\ F'\ r$ where $F' = (\lambda x.F\ (\lambda z.x\ x\ z)$. The next expression in the sequence is $F\ (\lambda z.F'\ F'\ z)\ r \to \dots \to \mathtt{amb}\ r\ (F'\ F'\ r)$. Hence r is one possible outcome. It is also the only possible abstraction as ed of the reduction sequence. Note that $(Y\ F\ r)$ has arbitrary long successful reduction sequences to r. We also have $(Id\ r){\Downarrow}$ as well as $(Y\ F\ r){\Downarrow}$. The simulation definitions imply $Id \simeq (Y\ F)$, and hence $Id \sim (Y\ F)$.

5 Simulations for the Call-By-Value Choice Calculus

Even though amb can simulate choice in different variants, if only (erratic or demonic) choice is permitted instead of amb, then the expressivity is different, which is reflected in different contextual equivalences. For example Ω is the smallest element if only choice is permitted, which is false in LCA. In this section we consider erratic choice only, since demonic and erratic choice can encode each other in a call-by-value calculus.

Definition 5.1 (The calculus LCC). *The calculus LCC is defined analogous to LCA with the following differences:*

- *Instead of* amb *the syntax has a binary operator* choice.
- *The hole of evaluation contexts is not inside arguments of* choice.
- *The reduction rules are (cbvbeta) and choice-reductions:*
  (choicel) : $(\mathtt{choice}\ s\ t) \to s$; *and* *(choicer)* : $(\mathtt{choice}\ s\ t) \to t$.
- *Reduction* $\xrightarrow{LCC}$ *applies the reduction rules in evaluation contexts.*
- *The definitions of contextual equivalences are as for LCA.*

The general properties on similarities and the candidate relation presented in Sect. 4.1 also hold for LCC. We immediately start with the similarity definitions and use the convex variant. In abuse of notation, we use the same symbols for the relations as for LCA.

Definition 5.2. *We define simulations for LCC on closed expressions:*

May-Similarity in LCC**,** $\preccurlyeq_{\downarrow} := \mathrm{gfp}(\mathrm{F}_{\downarrow})$**:** *Let* $s\ F_{\downarrow}(\eta)\ t$ *hold iff* $LR(s,t,\eta)$*.*

Should-Similarity in LCC**,** $\preccurlyeq_{\uparrow_X} := \mathrm{gfp}(\mathrm{F}_{\uparrow_X})$**:**
Let $s\ F_{\uparrow_X}(\eta)\ t$ *hold iff* $s\uparrow \implies t\uparrow$, $t \preccurlyeq_{\downarrow} s$, *and* $t\Downarrow \implies LR(s,t,\eta)$*.*

Doing the same using Howe's method for $\preccurlyeq_{\uparrow_X}$ as for LCA shows:

Theorem 5.3. *May-similarity* $\preccurlyeq_{\downarrow}$ *in LCC is a precongruence and sound for the contextual may-preorder, and the mutual may-similarity* $\approx_{\downarrow}$ *is a congruence and sound for may-equivalence.*

Definition 5.4. *The candidate relation* $\preccurlyeq_{\uparrow_X H}$ *is defined w.r.t. the relation* $\preccurlyeq_{\uparrow_X}$*.*

Lemma 5.5. $\preccurlyeq_{\uparrow_X H} \subseteq \succcurlyeq^o_{\downarrow}$.

Mostly, the proofs are the same as for LCA. So we only exhibit the differences.

Proposition 5.6. *Let* s,t *be closed LCC-expressions,* $s \preccurlyeq_{\uparrow_X H} t$, $t\Downarrow$, $s\downarrow\lambda x.s'$*. Then there is some* $\lambda x.t'$ *such that* $t\downarrow\lambda x.t'$ *and* $s' \preccurlyeq_{\uparrow_X H} t'$*.*

Proof. We work in the calculus LCC. The proof is by induction on the length of the reduction of $s\downarrow\lambda x.s'$. There are three cases: $s = \lambda x.s'$, $s = (\texttt{choice}\ s_1\ s_2)$ and $s = (s_1\ s_2)$, where the first and third cases are the same as for LCA. So we only show the case for the choice-expression:
Case $s = \texttt{choice}\ s_1\ s_2$, and $s\downarrow\lambda x.s'$. Then there is some closed expression $\texttt{choice}\ t_1\ t_2 \preccurlyeq_{\uparrow_X} t$ with $s_i \preccurlyeq_{\uparrow_X H} t_i$ for $i = 1,2$. Note that $t\Downarrow$ implies $t_1\Downarrow$ and $t_2\Downarrow$. W.l.o.g. let $s_1\downarrow\lambda x.s'$. Then by induction, there is some $\lambda x.t'$ with $t_1\downarrow\lambda x.t'$ and $s' \preccurlyeq_{\uparrow_X H} t'$. Obviously, also $\texttt{choice}\ t_1\ t_2\downarrow\lambda x.t'$. From $\texttt{choice}\ t_1\ t_2 \preccurlyeq_{\uparrow_X} t$ and $t\Downarrow$, we obtain that there is some $\lambda x.t''$ with $t\downarrow\lambda x.t''$ and $t' \preccurlyeq^o_{\uparrow_X} t''$, which implies $s' \preccurlyeq_{\uparrow_X H} t''$ by Lemma 4.3 (4). □

Note that in the calculus LCA this proof fails, since the induction hypothesis cannot be proved for s_i, t_i.

Proposition 5.7. *Let* s,t *be closed expressions,* $s \preccurlyeq_{\uparrow_X H} t$*, and* $s\uparrow$*. Then* $t\uparrow$*.*

Proof. The proof is by induction on the number of reductions of s to a must-divergent expression, and on the size of expressions as a second measure.
The base case is that $s\Uparrow$. Then Lemma 5.5 shows $t\Uparrow$, since $t \preccurlyeq_{\downarrow} s$ must hold, which implies $s \geq_{\downarrow} t$ and thus $s \leq_{\Uparrow} t$.
Let $s = \texttt{choice}\ s_1\ s_2$ with $s\uparrow$, and assume that $t\Downarrow$. Then there is some closed expression $t' = \texttt{choice}\ t_1\ t_2$ with $s_i \preccurlyeq_{\uparrow_X H} t_i$ for $i = 1,2$ and $\texttt{choice}\ t_1\ t_2 \preccurlyeq_{\uparrow_X} t$. This implies $t_1\Downarrow$ and $t_2\Downarrow$. It follows that $s_1\uparrow$ or $s_2\uparrow$. Applying the induction hypothesis shows that $t_1\uparrow$ or $t_2\uparrow$, which contradicts the assumption $t\Downarrow$.

Theorem 5.8. *The relation* $\preccurlyeq_{\uparrow_X}$ *in LCC is a precongruence on closed expressions and* $\preccurlyeq^o_{\uparrow_X}$ *is a precongruence on all expressions.*

Proof. We already have $\preccurlyeq_{\uparrow_X} \subseteq \preccurlyeq^c_{\uparrow_X H}$ by Lemma 4.3 (3). Propositions 5.6 and 5.7 show that $(\preccurlyeq_{\uparrow_X H})^c$ satisfies the fixpoint conditions of $\preccurlyeq_{\uparrow_X}$ and thus coinduction shows $(\preccurlyeq_{\uparrow_X H})^c \subseteq \preccurlyeq_{\uparrow_X}$. Hence we have $(\preccurlyeq_{\uparrow_X H})^c = \preccurlyeq_{\uparrow_X}$. Lemma 4.3.(8) then shows the equation $\preccurlyeq_{\uparrow_X H} = \preccurlyeq^o_{\uparrow_X}$. □

Theorem 5.9. $\preccurlyeq^o_{\uparrow_X}$ *is sound for* $\geq_{LCC}$*, and* $\approx^o_{\uparrow_X}$ *is sound for* $\sim_{LCC}$*.*

Proof. We first show that $\preccurlyeq^o_{\uparrow_X}$ is sound for $\leq_{\uparrow,LCC}$ (and thus also for $\geq_{\Downarrow,LCC}$): Let $s \preccurlyeq^o_{\uparrow_X} t$, and let C be a context such that $C[s], C[t]$ are closed. First assume that $C[s]\uparrow$. Theorem 5.8 shows that $C[s] \preccurlyeq_{\uparrow_X} C[t]$, and so $C[t]\uparrow$. Since $s \preccurlyeq^o_{\uparrow_X} t$ also implies $t \preccurlyeq^o_{\downarrow} s$ and thus $t \leq_{\downarrow,LCC} s$, we have $\preccurlyeq^o_{\uparrow_X} \subseteq \geq_{LCC}$. The second part of the theorem follows by symmetry. □

Proposition 5.10. *Let s, t be closed with $s\uparrow, t\uparrow$. Then $s \approx_{\downarrow} t \implies s \sim_{LCC} t$.*

Proof. First note that $\Omega \leq_{LCC} r$ for all r, which follows from Theorems 5.3 and 5.9. Theorem 5.9 shows that $s \approx_{\downarrow} t$, $s\uparrow, t\uparrow$ implies that $s \sim_{LCC} t$.

Note that this proposition is not valid in LCA.

Proposition 5.11. *Convex should-simulation $\preccurlyeq_{\uparrow X}$ is not complete for $\leq_{\uparrow,LCC}$.*

Proof. Let $s = \texttt{choice}\ \Omega\ (\lambda x.\Omega)$ and $t = \texttt{choice}\ \Omega\ \mathit{Top}$. Then $s \leq_{\uparrow,LCC} t$, as well as $t \leq_{\uparrow,LCC} s$ holds, since for every context C, if $C[s]\uparrow$, then also $C[t]\uparrow$ by selecting always the Ω in a choice-reduction, and also vice versa. However, $t \not\preccurlyeq_{\downarrow} s$ (since $\mathit{Top} \not\preccurlyeq_{\downarrow} \lambda x.\Omega$), and thus $s \preccurlyeq_{\uparrow X} t$ does not hold. □

6 Conclusion

We have shown that in the call-by-value lambda calculus with amb there exists a very expressive (an argument for this is Proposition 4.17) mutual similarity for should-convergence, which is a congruence and sound for contextual equivalence. We also showed that the used method can be transferred to the call-by-value lambda calculus with choice. This novel and encouraging result may enable further research for more expressive non-deterministic and/or concurrent calculi and languages and for call-by-need lambda calculi using the approximation techniques from *e.g.* [15, 16].

References

1. Abramsky, S.: The lazy lambda calculus. In Turner, D.A., ed.: Research Topics in Functional Programming, Addison-Wesley (1990) 65–116
2. Broy, M.: A theory for nondeterminism, parallelism, communication, and concurrency. Theoret. Comput. Sci. **45** (1986) 1–61
3. Carayol, A., Hirschkoff, D., Sangiorgi, D.: On the representation of McCarthy's amb in the pi-calculus. Theoret. Comput. Sci. **330**(3) (2005) 439–473
4. Davey, B., Priestley, H.: Introduction to Lattices and Order. Cambridge University Press, Cambridge (1992)
5. Fournet, C., Gonthier, G.: A hierarchy of equivalences for asynchronous calculi. J. Log. Algebr. Program. **63**(1) (2005) 131–173
6. Gordon, A.D.: Bisimilarity as a theory of functional programming. Theoret. Comput. Sci. **228**(1-2) (1999) 5–47
7. Howe, D.: Equality in lazy computation systems. In: 4th IEEE Symp. on Logic in Computer Science. (1989) 198–203
8. Howe, D.: Proving congruence of bisimulation in functional programming languages. Inform. and Comput. **124**(2) (1996) 103–112
9. Laneve, C.: On testing equivalence: May and must testing in the join-calculus. Technical Report Technical Report UBLCS 96-04, University of Bologna (1996)
10. Lassen, S.B.: Normal form simulation for McCarthy's amb. Electr. Notes Theor. Comput. Sci. **155** (2006) 445–465
11. Lassen, S.B., Moran, A.: Unique fixed point induction for McCarthy's amb. In Kutylowski, M., Pacholski, L., Wierzbicki, T., eds.: Proc. 24th International Symposium on Mathematical Foundations of Computer Science. Number 1672 in LNCS, Springer (1999) 198–208
12. Lassen, S.B., Pitcher, C.S.: Similarity and bisimilarity for countable non-determinism and higher-order functions. Electron. Notes Theor. Comput. Sci. **10** (2000)
13. Lassen, S.B.: Relational Reasoning about Functions and Nondeterminism. PhD thesis, University of Aarhus (1998)
14. Levy, P.B.: Amb breaks well-pointedness, ground amb doesn't. Electron. Notes Theor. Comput. Sci. **173**(1) (2007) 221–239
15. Mann, M.: Congruence of bisimulation in a non-deterministic call-by-need lambda calculus. Electron. Notes Theor. Comput. Sci. **128**(1) (2005) 81–101
16. Mann, M., Schmidt-Schauß, M.: Similarity implies equivalence in a class of non-deterministic call-by-need lambda calculi. Inform. and Comput. **208**(3) (2010) 276 – 291

17. Mason, I., Talcott, C.L.: Equivalence in functional languages with effects. J. Funct. Programming **1**(3) (1991) 287–327
18. McCarthy, J.: A Basis for a Mathematical Theory of Computation. In Braffort, P., Hirschberg, D., eds.: Computer Programming and Formal Systems, North-Holland, Amsterdam (1963) 33–70
19. Moran, A.K.D.: Call-by-name, call-by-need, and McCarthy's Amb. PhD thesis, Chalmers University, Sweden (1998)
20. Niehren, J., Sabel, D., Schmidt-Schauß, M., Schwinghammer, J.: Observational semantics for a concurrent lambda calculus with reference cells and futures. Electron. Notes Theor. Comput. Sci. **173** (2007) 313–337
21. Ong, C.H.L.: Non-determinism in a functional setting. In Vardi, M.Y., ed.: Proc. 8th IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press (1993) 275–286
22. Pitts, A.M.: Howe's method for higher-order languages. In Sangiorgi, D., Rutten, J., eds.: Advanced Topics in Bisimulation and Coinduction. Volume 52 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press (2011) 197–232 (chapter 5).
23. Rensink, A., Vogler, W.: Fair testing. Inform. and Comput. **205**(2) (2007) 125–198
24. Sabel, D., Schmidt-Schauß, M.: A call-by-need lambda-calculus with locally bottom-avoiding choice: Context lemma and correctness of transformations. Math. Structures Comput. Sci. **18**(03) (2008) 501–553
25. Sabel, D., Schmidt-Schauß, M.: A contextual semantics for Concurrent Haskell with futures. In Schneider-Kamp, P., Hanus, M., eds.: Proc. 13th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, ACM (2011) 101–112
26. Sabel, D., Schmidt-Schauß, M.: Conservative concurrency in Haskell. In Dershowitz, N., ed.: Proc. 27th ACM/IEEE Symposium on Logic in Computer Science (LICS '12), IEEE Computer Society (2012) 561–570
27. Schmidt-Schauß, M., Sabel, D.: Correctness of an STM Haskell implementation. In Morrisett, G., Uustalu, T., eds.: Proc. 18th ACM SIGPLAN International Conference on Functional programming, ACM (2013) 161–172
28. Schmidt-Schauß, M., Sabel, D., Machkasova, E.: Counterexamples to applicative simulation and extensionality in non-deterministic call-by-need lambda-calculi with letrec. Inf. Process. Lett. **111**(14) (2011) 711–716
29. V.Koutavas, P.B.Levy, E.Sumii: Limitations of applicative bisimulation. In: Modelling, Controlling and Reasoning about State. Number 10351 in Dagstuhl Seminar Proceedings (2010)

A Proofs for the Howe-Candidate Relation

Proof of Lemma 4.3

Proof. Parts (1), (2), and (3) can be shown by structural induction and using reflexivity of $\preccurlyeq^o$. Part (4) follows from the definition, Lemma 4.2, and transitivity of $\preccurlyeq^o$. Part (5) is shown by structural induction on the expression t: In the case $x \preccurlyeq_H t'$, we obtain $x \preccurlyeq^o t'$ from the definition, and so $v' \preccurlyeq^o t'[v'/x]$ by Lemma 3.2 (2), since v, v' are values, and hence $x[v/x] \preccurlyeq_H t'[v'/x]$ using (4). In the case $y \preccurlyeq_H t'$ with $x \neq y$, we obtain $y \preccurlyeq^o t'$ from the definition, and $y[v/x] = y \preccurlyeq^o t'[v'/x]$ and thus $y = y[v/x] \preccurlyeq_H t'[v'/x]$. If $t = \tau(r_1, \ldots, r_n)$, $t \preccurlyeq_H t'$, then there is some $\tau(t'_1, \ldots, t'_n) \preccurlyeq^o t'$ with $t_i \preccurlyeq_H t'_i$. W.l.o.g. bound variables have fresh names. We have $t_i[v/x] \preccurlyeq_H t'_i[v'/x]$ and $\tau(t'_1, \ldots, t'_n)[v'/x] \preccurlyeq^o t'[v'/x]$. Thus $t[v/x] \preccurlyeq_H t'[v'/x]$. Part (6) follows from item (5). Part (7) follows from item (6) and Lemma 3.2. For part (8) assume that $\preccurlyeq_H^c = \preccurlyeq$. Then $(\preccurlyeq_H^c)^o = \preccurlyeq^o$. Part (7) shows $\preccurlyeq_H \subseteq (\preccurlyeq_H^c)^o$ and part (3) shows $\preccurlyeq^o \subseteq \preccurlyeq_H$, which together implies equality $\preccurlyeq_H = \preccurlyeq^o$. □

Lemma A.1. *The middle expression in the definition of $\preccurlyeq_H$ can be chosen to be closed if s, t are closed: Let $s = \tau(s_1, \ldots, s_{\mathrm{ar}(\tau)})$, such that $s \preccurlyeq_H t$ holds. Then there are operands s'_i, such that $\tau(s'_1, \ldots, s'_{\mathrm{ar}(\tau)})$ is closed, $\forall i : s_i \preccurlyeq_H s'_i$ and $\tau(s'_1, \ldots, s'_{\mathrm{ar}(\tau)}) \preccurlyeq^o t$.*

Proof. The definition of $\preccurlyeq_H$ implies that there is an expression $\tau(s''_1, \ldots, s''_{\mathrm{ar}(\tau)})$ such that $s_i \preccurlyeq_H s''_i$ for all i and $\tau(s''_1, \ldots, s''_{\mathrm{ar}(\tau)}) \preccurlyeq^o t$. Let σ be the substitution with $\sigma(x) := v_x$ for all $x \in FV(\tau(s''_1, \ldots, s''_{\mathrm{ar}(\tau)}))$, where v_x is any closed abstraction. Lemma 4.3 now shows that $s_i = \sigma(s_i) \preccurlyeq_H \sigma(s''_i)$ holds for all i. The relation $\sigma(\tau(s''_1, \ldots, s''_{\mathrm{ar}(\tau)})) \preccurlyeq t$ holds, since t is closed and due to the definition of an open extension. The requested expression is $\tau(\sigma(s''_1), \ldots, \sigma(s''_{\mathrm{ar}(\tau)}))$. □

B Precongruence of May-Similarity

The goal in the following is to show that $\preccurlyeq_\downarrow$ is a precongruence and sound for $\leq_\downarrow$. Here we use the definitions and results in Sections 4.1 and 3.1, where the definitions in Sections 4.1 are applied to $\preccurlyeq_\downarrow$. This proof proceeds by defining a congruence candidate $\preccurlyeq_{\downarrow H}$ by the closure of $\preccurlyeq_\downarrow$ within contexts using Howe's technique, which obviously is operator-respecting, but transitivity needs to be shown. By proving that $\preccurlyeq_\downarrow$ and $\preccurlyeq_{\downarrow H}$ coincide, on the one hand transitivity of $\preccurlyeq_{\downarrow H}$ follows (since $\preccurlyeq_\downarrow$ is transitive) and on the other hand (and more importantly) it follows that $\preccurlyeq_\downarrow$ is operator-respecting (since $\preccurlyeq_{\downarrow H}$ is operator-respecting) and thus a precongruence.

Definition B.1. *The relation $\preccurlyeq_{\downarrow H}$ is defined using Definition 4.1 for $\preccurlyeq_\downarrow$.*

Lemma B.2. *If $s \xrightarrow{LCA} s'$, then $s' \preccurlyeq_\downarrow^o s$.*

Lemma B.3. *If $s = \lambda x.s'$ and t are closed, and $\lambda x.s' \preccurlyeq_{\downarrow H} t$, then there is some closed $\lambda x.t'$ with $t \xrightarrow{LCA,*} \lambda x.t'$ and $s' \preccurlyeq_{\downarrow H} t'$, and thus also $\lambda x.s' \preccurlyeq_{\downarrow H} \lambda x.t'$.*

Proof. The relation $\lambda x.s' \preccurlyeq_{\downarrow H} t$ implies that there is a closed $\lambda x.t''$, such that $s' \preccurlyeq_{\downarrow H} t''$ (and hence $\lambda x.s' \preccurlyeq_{\downarrow H} \lambda x.t''$), and $\lambda x.t'' \preccurlyeq_\downarrow t$. This in turn implies that for some $\lambda x.t'$, we have $t \xrightarrow{LCA,*} \lambda x.t'$ with $t'' \preccurlyeq_\downarrow^o t'$. Lemma 4.3 (4) thus implies $s' \preccurlyeq_{\downarrow H} t'$ and Lemma 4.5 shows $\lambda x.s' \preccurlyeq_{\downarrow H} \lambda x.t'$.

Proposition B.4. *Let s, t be closed expressions, $s \preccurlyeq_{\downarrow H} t$ and $s \xrightarrow{LCA} s'$ where s is the redex. Then $s' \preccurlyeq_{\downarrow H} t$.*

Proof. The relation $s \preccurlyeq_{\downarrow H} t$ implies that $s = \tau(s_1, \ldots, s_n)$ and by Lemma 4.3 part 9 there is some closed $t' = \tau(t'_1, \ldots, t'_n)$ with $s_i \preccurlyeq_{\downarrow H} t'_i$ for all i and $t' \preccurlyeq_\downarrow t$.

- For the (cbvbeta)-reduction, $s = (s_1\ s_2)$, where $s_1 = \lambda x.s'_1$, $s_2 = \lambda x.s'_2$ are closed, and $t' = (t'_1\ t'_2)$ is also closed. The relation $(\lambda x.s'_1) = s_1 \preccurlyeq_{\downarrow H} t'_1$ and Lemma B.3 imply that there exists a closed expression $\lambda x.t''_1 \preccurlyeq_{\downarrow} t'_1$ with $t'_1 \xrightarrow{LCA,*} \lambda x.t''_1$, $s'_1 \preccurlyeq_{\downarrow H} t''_1$, and $\lambda x.s'_1 \preccurlyeq_{\downarrow H} \lambda x.t''_1$. Also for t'_2 and since $s_2 = \lambda x.s'_2 \preccurlyeq_{\downarrow H} t'_2$, there is some closed $\lambda x.t''_2 \preccurlyeq_{\downarrow} t'_2$ with $t'_2 \xrightarrow{LCA,*} \lambda x.t''_2$, $s'_2 \preccurlyeq_{\downarrow H} t''_2$, and $\lambda x.s'_2 \preccurlyeq_{\downarrow H} \lambda x.t''_2$. Since $\preccurlyeq_{\downarrow H}$ is operator-respecting, we have $(s_1\ s_2) \preccurlyeq_{\downarrow H} ((\lambda x.t''_1)\ (\lambda x.t''_2))$ and also $t' = (t'_1\ t'_2) \xrightarrow{LCA,*} ((\lambda x.t''_1)\ (\lambda x.t''_2))$. Now on both sides a call-by-value beta-reduction is possible and results in $s'_1[s_2/x]$ and $t''_1[(\lambda x.t''_2)/x]$, respectively. Since $s'_1 \preccurlyeq_{\downarrow H} t''_1$ and $s_2 \preccurlyeq_{\downarrow H} \lambda x.t''_2$, we have $s'_1[s_2/x] \preccurlyeq_{\downarrow H} t''_1[(\lambda x.t''_2)/x]$. From $t' \xrightarrow{LCA,*} t''_1[(\lambda x.t''_2)/x]$, we obtain $t''_1[(\lambda x.t''_2)/x] \preccurlyeq_{\downarrow} t' \preccurlyeq_{\downarrow} t$, and so $s'_1[s_2/x] \preccurlyeq_{\downarrow H} t$.
- Suppose, the reduction is a (ambl)-reduction, where $s = (\texttt{amb}\ s_1\ s_2)$ and $s \xrightarrow{LCA} s_1$, which is only possible if $s_1 = \lambda x.s'_1$. Then there is $t' = (\texttt{amb}\ t'_1\ t'_2)$ with $s_i \preccurlyeq_{\downarrow H} t'_i$ for $i = 1, 2$ and $t' \preccurlyeq_{\downarrow} t$. From $s_1 \preccurlyeq_{\downarrow H} t'_1$ we derive that $t'_1 \xrightarrow{LCA,*} \lambda x.t''_1$ with $s_1 \preccurlyeq_{\downarrow H} \lambda x.t''_1$. Now we have the reduction sequence $t' = (\texttt{amb}\ t'_1\ t'_2) \xrightarrow{LCA,*} (\texttt{amb}\ (\lambda x.t''_1)\ t'_2) \xrightarrow{LCA} (\lambda x.t''_1)$, and by $(\lambda x.t''_1) \preccurlyeq_{\downarrow} (\texttt{amb}\ (\lambda x.t''_1)\ t'_2) \preccurlyeq_{\downarrow} t' \preccurlyeq_{\downarrow} t$, we derive $(\lambda x.t''_1) \preccurlyeq_{\downarrow} t$. Together with $s_1 \preccurlyeq_{\downarrow H} \lambda x.t''_1$ we obtain $s_1 \preccurlyeq_{\downarrow H} t$.
- The reasoning is completely analogous for an (ambr)-reduction. □

Proposition B.5. *Let s, t be closed, $s \preccurlyeq_{\downarrow H} t$ and $s \xrightarrow{LCA} s_0$. Then $s_0 \preccurlyeq_{\downarrow H} t$.*

Proof. Let $s = E[s']$, where s' is the redex and $E \in \mathbb{E}$. We use induction on the length of the path to the redex within $s = E[s']$, i.e. the path of the hole of E. The base case where $E = [\cdot]$ is proven in Proposition B.4. Let $E[s'], t$ be closed, $E[s'] \preccurlyeq_{\downarrow H} t$ and $E[s'] \xrightarrow{LCA} E[s'']$, where we assume that the redex is not at the top level. The relation $E[s'] \preccurlyeq_{\downarrow H} t$ implies that $E[s'] = \tau(s_1, \ldots, s_n)$ and that there is some closed $t' = \tau(t'_1, \ldots, t'_n) \preccurlyeq_{\downarrow}^{o} t$ with $s_i \preccurlyeq_{\downarrow H} t'_i$ for all i. Let j be the first index in the path to the redex. There are two cases:

1. j is also a reduction position in t'. If $s_j \xrightarrow{LCA} s'_j$, then by induction hypothesis $s'_j \preccurlyeq_{\downarrow H} t'_j$. Since $\preccurlyeq_{\downarrow H}$ is operator-respecting, we also obtain $E[s''] = \tau(s_1, \ldots, s_{j-1}, s'_j, s_{j+1}, \ldots, s_n) \preccurlyeq_{\downarrow H} \tau(t'_1, \ldots, t'_{j-1}, t'_j, t'_{j+1}, \ldots, t'_n)$, and from $\tau(t'_1, \ldots, t'_n) \preccurlyeq_{\downarrow}^{o} t$ we have $E[s''] = \tau(s_1, \ldots, s_{j-1}, s'_j, s_{j+1}, \ldots, s_n) \preccurlyeq_{\downarrow H} t$.
2. j is not a reduction position in t'. Then the only possibility is that $s = (s_1\ s_2)$, $j = 2$, $t' = (t'_1\ t'_2)$, s_1 is an abstraction, but t'_1 is not an abstraction. Lemma B.3 shows that there is an expression $\lambda x.t''_1$ with $s_1 \preccurlyeq_{\downarrow H} \lambda x.t''_1$ and $\lambda x.t''_1 \xleftarrow{LCA,*} t'_1$. Hence also $(\lambda x.t''_1)\ t'_2 \xleftarrow{LCA,*} t'$, and so $(\lambda x.t''_1)\ t'_2 \preccurlyeq_{\downarrow} t'$. The first index of the redex position in $((\lambda x.t''_1)\ t'_2)$ is also $j = 2$. Since $s_2 \xrightarrow{LCA} s'_2$, by the induction hypothesis $s'_2 \preccurlyeq_{\downarrow H} t'_2$. We have $(s_1\ s'_2) \preccurlyeq_{\downarrow H} ((\lambda x.t''_1)\ t'_2) \preccurlyeq_{\downarrow} t' \preccurlyeq_{\downarrow} t$, hence also $(s_1\ s'_2) \preccurlyeq_{\downarrow H} t$. □

Now we are ready to prove that the (closed restriction of the) precongruence candidate and similarity coincide.

Theorem B.6. $\preccurlyeq_{\downarrow H}^{c} = \preccurlyeq_{\downarrow}$ *and* $\preccurlyeq_{\downarrow H} = \preccurlyeq_{\downarrow}^{o}$.

Proof. Since $\preccurlyeq_{\downarrow} \subseteq \preccurlyeq_{\downarrow H}^{c}$ by Lemma 4.3, we have to show that $\preccurlyeq_{\downarrow H}^{c} \subseteq \preccurlyeq_{\downarrow}$. It is sufficient to show that $\preccurlyeq_{\downarrow H}^{c}$ satisfies the fixpoint equation for $\preccurlyeq_{\downarrow}$. We show that $\preccurlyeq_{\downarrow H}^{c} \subseteq F_{\downarrow}(\preccurlyeq_{\downarrow H}^{c})$. Let $s \preccurlyeq_{\downarrow H}^{c} t$ for closed terms s, t. We show that $s\ F_{\downarrow}(\preccurlyeq_{\downarrow H}^{c})\ t$: If $s \Uparrow_{LCA}$, then $s\ F_{\downarrow}(\preccurlyeq_{\downarrow H}^{c})\ t$ holds by Definition 3.4. If $s \downarrow \lambda x.s_1$, then $\lambda x.s_1 \preccurlyeq_{\downarrow H}^{c} t$ by Proposition B.5. Lemma B.3 shows that $t \xrightarrow{LCA,*} \lambda x.t_1$ for some $\lambda x.t_1$ and $s_1 \preccurlyeq_{\downarrow H} t_1$. Hence also $s_1\ ((\preccurlyeq_{\downarrow H})^{c})^{o}\ t_1$ by Lemma 4.3. This implies $s\ F_{\downarrow}(\preccurlyeq_{\downarrow H}^{c})\ t$. Thus the fixpoint property of $\preccurlyeq_{\downarrow H}^{c}$ *w.r.t.* $F_{\downarrow}$ holds, and hence $\preccurlyeq_{\downarrow H}^{c} = \preccurlyeq_{\downarrow}$.

Now we prove the second part. The first part, $\preccurlyeq_{\downarrow H}^{c} = \preccurlyeq_{\downarrow}$, implies $(\preccurlyeq_{\downarrow H}^{c})^{o} = \preccurlyeq_{\downarrow}^{o}$. Lemma 4.3 (7) implies $\preccurlyeq_{\downarrow H} \subseteq (\preccurlyeq_{\downarrow H}^{c})^{o} = \preccurlyeq_{\downarrow}^{o}$. The other direction is proven in Lemma 4.3 (3). □

Since $\preccurlyeq_{\downarrow}^{o}$ is reflexive and transitive (Lemma 4.4) and $\preccurlyeq_{\downarrow H}^{c}$ is operator-respecting (Lemma 4.3 (2)), this immediately implies:

Corollary B.7. *$\preccurlyeq_{\downarrow}^{o}$ is a precongruence on expressions $Expr_{LCA}$. If σ is a value-substitution, then $s \preccurlyeq_{\downarrow}^{o} t$ implies $\sigma(s) \preccurlyeq_{\downarrow}^{o} \sigma(t)$.*

We have soundness of may-simulation:

Theorem B.8. $\preccurlyeq_{\downarrow}^{o} \ \subseteq \ \leq_{\downarrow}$.

Proof. Let s, t be expressions with $s \preccurlyeq_{\downarrow}^{o} t$ and C be a context such that $C[s], C[t]$ are closed, and $C[s]\downarrow$. Since $\preccurlyeq_{\downarrow}^{o}$ is a congruence, the relation $C[s] \preccurlyeq_{\downarrow}^{o} C[t]$ holds, and in fact $C[s] \preccurlyeq_{\downarrow} C[t]$. From the definition of $\preccurlyeq_{\downarrow}$ we see that $C[t]\downarrow$ also holds. Since this is valid for all contexts C, we have proved $s \leq_{\downarrow} t$. Hence $\preccurlyeq_{\downarrow}^{o} \ \subseteq \ \leq_{\downarrow}$.

Proposition B.9. $\leq_{\downarrow} \ \not\subseteq \ \preccurlyeq_{\downarrow}^{o}$.

Proof. An example similar to the one in [15] shows that there is an expression s that is like an infinite `amb` of expressions $\lambda x_1, \dots, x_n.\bot$, where it can be shown that $s \sim_{\downarrow} (Y\ K)$, however, the simulation cannot detect this relation.

C Precongruence and Soundness of a Bisimilarity

C.1 Preliminaries for the Candidate Relation for Bisimilarities

The standard Howe-technique for similarities can be extended for bisimilarities. We will present some preparations for this extension.

Definition C.1. *The transitive closure $\preccurlyeq_H^*$ of $\preccurlyeq_H$ is defined as the least transitive relation such that $\preccurlyeq_H \subseteq \preccurlyeq_H^*$. Equivalently, $\preccurlyeq_H^*$ is the union of all relations $(\preccurlyeq_H)_i$, where $(\preccurlyeq_H)_i$ is the i-fold relational composition of $\preccurlyeq_H$.*

The following lemma represents the core of the transitive closure trick explained in [22]. It helps to circumvent the asymmetry of $\preccurlyeq_H$.

Lemma C.2. *If $\preccurlyeq$ is an equivalence relation (i.e. we could also write $\simeq$), then the transitive closure $\preccurlyeq_H^*$ is also an equivalence relation.*

Proof. Reflexivity of $\preccurlyeq_H^*$ follows from reflexivity of $\preccurlyeq_H$, transitivity from its definition. Symmetry of $\preccurlyeq_H^*$ requires an inductive argument on the size of expressions, where it is sufficient to show that $s \preccurlyeq_H t$ implies $t \preccurlyeq_H^* s$ using induction on the construction of the transitive closure. $x \preccurlyeq_H t$ implies $t \preccurlyeq_H x$, since $x \preccurlyeq^o t$, hence $t \preccurlyeq^o x$ by symmetry, and since $\preccurlyeq^o \ \subseteq \ \preccurlyeq_H$. If $\tau(s_1, \dots, s_n) \preccurlyeq_H t$, then there is some $\tau(t_1, \dots, t_n) \preccurlyeq^o t$ with $s_i \preccurlyeq_H t_i$. By induction hypothesis $t_i \preccurlyeq_H s_i$ for $i = 1, 2$. Hence $\tau(t_1, \dots, t_n) \preccurlyeq_H \tau(s_1, \dots, s_n)$. Symmetry of $\preccurlyeq^o$ implies $t \preccurlyeq^o \tau(t_1, \dots, t_n)$, and hence from $t \preccurlyeq_H \tau(t_1, \dots, t_n)$, we derive $t \preccurlyeq_H^* \tau(s_1, \dots, s_n)$.

Lemma C.3. *If $\preccurlyeq$ is symmetric (i.e. it is an equivalence relation), then the claims of Lemma 4.3 also hold for $\preccurlyeq_H^*$ instead of $\preccurlyeq_H$.*

Proof. That $\preccurlyeq_H^*$ is operator-respecting follows by induction on the formation of the transitive closure, since $\preccurlyeq_H$ is operator respecting. That $\preccurlyeq_H^*$ is stable under value-substitutions also follows by an induction on the formation of the transitive closure. The other claims can now be easily transferred to the transitive closure.

C.2 Precongruence of Bisimulation for Should-Convergence

In this section we present a proof for soundness of should-bisimulation, i.e. $\simeq^o_{\Downarrow} \subseteq \sim_{LCA}$. The goal in the following is to show that the candidate relation $\preccurlyeq_{\Downarrow H}$ derived from $\simeq_{\Downarrow}$ can be treated using the method of Howe to prove soundness of the applicative simulations. In particular we exploit the transitive-closure extension as mentioned in [8] and presented in [22].

The following lemma is straight-forward.

Lemma C.4. $\simeq_{\Downarrow} \subseteq \approx_{\downarrow}$ *and* $\simeq_{\Downarrow} \subseteq \approx_{\uparrow} \subseteq \sim_{LCA}$

Proof. The inclusion $\simeq_{\Downarrow} \subseteq \approx_{\downarrow}$ holds, since $\simeq_{\Downarrow}$ is $F_{\downarrow}$-dense and since $\simeq_{\Downarrow}$ is symmetric. The inclusion $\simeq_{\Downarrow} \subseteq \approx_{\uparrow}$ holds, since $\simeq_{\Downarrow}$ is $F_{\uparrow}$-dense and since $\simeq_{\Downarrow}$ is symmetric. The inclusion $\approx_{\uparrow} \subseteq \sim_{LCA}$ follows from Theorem 4.15

We have already proved that $\preccurlyeq^o_{\downarrow}$ is a precongruence (Corollary B.7). This will be required for the transfer of may-divergence over the candidate relation.

Definition C.5. *The candidate relation $\preccurlyeq_{\Downarrow H}$ is defined w.r.t. the relation $\simeq_{\Downarrow}$.*

Lemma C.6. $\preccurlyeq_{\Downarrow H} \subseteq \preccurlyeq^o_{\downarrow} \cap \succcurlyeq^o_{\downarrow}$.

Proof. We know that $\preccurlyeq^o_{\downarrow}$ is a precongruence from Corollary B.7, hence this also holds for $\succcurlyeq^o_{\downarrow}$. To show that $s \preccurlyeq_{\Downarrow H} t \implies s \preccurlyeq^o_{\downarrow} t$, we use induction on the structure of s. In the case $s = x$ the implication follows from the definition of the candidate and from Lemma C.4. If $s = \tau(s_1, \ldots, s_n)$, there is some $\tau(t_1, \ldots, t_n) \simeq^o_{\Downarrow} t$ with $s_i \preccurlyeq_{\Downarrow H} t_i$ for all i. The induction hypothesis implies $s_i \preccurlyeq^o_{\downarrow} t_i$ for all i, and the precongruence property of $\preccurlyeq^o_{\downarrow}$ shows $\tau(s_1, \ldots, s_n) \preccurlyeq^o_{\downarrow} \tau(t_1, \ldots, t_n)$. Transitivity of $\preccurlyeq^o_{\downarrow}$ and $\simeq^o_{\Downarrow} \subseteq \preccurlyeq^o_{\downarrow}$ now shows $s = \tau(s_1, \ldots, s_n) \preccurlyeq^o_{\downarrow} t$. The proof for $\succcurlyeq^o_{\downarrow}$ is similar.

Proposition C.7. *Let s, t be closed expressions, $s \preccurlyeq_{\Downarrow H} t$ and $s{\downarrow}\lambda x.s'$. Then there is some $\lambda x.t'$ such that $t{\downarrow}\lambda x.t'$ and $s' \preccurlyeq_{\Downarrow H} t'$.*

Proof. The proof is by induction on the length of the reduction of $s{\downarrow}\lambda x.s'$.

- If $s = \lambda x.s'$, then there is some closed $\lambda x.t'$ with $s' \preccurlyeq_{\Downarrow H} t'$ and $\lambda x.t' \simeq_{\Downarrow} t$. The latter implies that there is some closed $\lambda x.t''$ with $t{\downarrow}\lambda x.t''$ and $t' \simeq^o_{\Downarrow} t''$, and so $s' \preccurlyeq_{\Downarrow H} t''$ by Lemma 4.3 (4).
- Case $s = \texttt{amb}\ s_1\ s_2$, and $s{\downarrow}\lambda x.s'$. Then there is some closed expression $\texttt{amb}\ t_1\ t_2 \simeq_{\Downarrow} t$ with $s_i \preccurlyeq_{\Downarrow H} t_i$ for $i = 1, 2$. W.l.o.g. let $s_1{\downarrow}\lambda x.s'$. Then by induction, there is some $\lambda x.t'$ with $t_1{\downarrow}\lambda x.t'$ and $s' \preccurlyeq_{\Downarrow H} t'$. Obviously, also $\texttt{amb}\ t_1\ t_2{\downarrow}\lambda x.t'$. From $\texttt{amb}\ t_1\ t_2 \simeq_{\Downarrow} t$, we obtain that there is some $\lambda x.t''$ with $t{\downarrow}\lambda x.t''$ and $t' \simeq^o_{\Downarrow} t''$, which implies $s' \preccurlyeq_{\Downarrow H} t''$ by Lemma 4.3 (4).
- If $s = (s_1\ s_2)$, then there is some closed $t' = (t'_1\ t'_2) \simeq_{\Downarrow} t$ with $s_i \preccurlyeq_{\Downarrow H} t'_i$ for $i = 1, 2$. Since $(s_1\ s_2){\downarrow}\lambda x.s'$ there is a reduction sequence $(s_1\ s_2) \xrightarrow{LCA,*} (\lambda x.s'_1)\ s_2 \xrightarrow{LCA,*} (\lambda x.s'_1)\ (\lambda x.s'_2) \xrightarrow{LCA} s'_1[\lambda x.s'_2/x] \xrightarrow{LCA,*} \lambda x.s'$ such that $s_i{\downarrow}\lambda x.s'_i$ for $i = 1, 2$. By induction, there are expressions $\lambda x.t''_i$ with $t'_i{\downarrow}\lambda x.t''_i$ and $s'_i \preccurlyeq_{\Downarrow H} t''_i$. Lemma 4.3 (5) now shows $s'_1[\lambda x.s'_2/x] \preccurlyeq_{\Downarrow H} t''_1[\lambda x.t''_2/x]$. Now we can again use the induction hypothesis which shows that there is some $\lambda x.t''$ with $t''_1[\lambda x.t''_2/x]{\downarrow}\lambda x.t''$ and $s' \preccurlyeq_{\Downarrow H} t''$. The relation $(t'_1\ t'_2) \simeq_{\Downarrow} t$ implies that $t{\downarrow}\lambda x.t_0$ with $t'' \simeq^o_{\Downarrow} t_0$, and hence $s' \preccurlyeq_{\Downarrow H} t_0$ by Lemma 4.3 (4).

Proposition C.8. *Let s, t be closed, $s \preccurlyeq_{\Downarrow H} t$ and $s{\uparrow}$. Then also $t{\uparrow}$.*

Proof. The proof is by induction on the number of reductions of s to a must-divergent expression, and on the size of expressions as a second measure.

- The base case is that $s{\Uparrow}$. Then Lemma C.6 shows $t{\Uparrow}$.

- Let $s = \texttt{amb}\ s_1\ s_2$ with $s\uparrow$. Then there is some closed expression $t' = \texttt{amb}\ t_1\ t_2$ with $s_i \preccurlyeq_{\Downarrow H}\ t_i$ for $i = 1, 2$ and $\texttt{amb}\ t_1\ t_2\ \simeq_{\Downarrow}\ t$. It follows that $s_1\uparrow$ as well as $s_2\uparrow$. Applying the induction hypothesis shows that $t_1\uparrow$ as well as $t_2\uparrow$, and hence $(\texttt{amb}\ t_1\ t_2)\uparrow$. From $\texttt{amb}\ t_1\ t_2\ \simeq_{\Downarrow}\ t$ we obtain $t\uparrow$.
- Let $s = (s_1\ s_2)$ with $s\uparrow$. Then there is some closed expression $t' = (t_1\ t_2) \simeq_{\Downarrow} t$ and $s_i \preccurlyeq_{\Downarrow H} t_i$ for $i = 1, 2$. There are several cases:
  1. If $(s_1\ s_2) \xrightarrow{LCA,*} (s'_1\ s_2)$ and $s'_1\Uparrow$, then $s_1\uparrow$ and by the induction hypothesis also $t_1\uparrow$, and hence $t'\uparrow$, which implies $t\uparrow$.
  2. If $(s_1\ s_2) \xrightarrow{LCA,*} (\lambda x.s'_1)\ s_2 \xrightarrow{LCA,*} (\lambda x.s'_1)\ s'_2$ and $s'_2\Uparrow$, then $s_2\uparrow$ and by induction hypothesis also $t_2\uparrow$, and hence $t'\uparrow$, which implies $t\uparrow$.
  3. If $(s_1\ s_2) \xrightarrow{LCA,*} (\lambda x.s'_1)\ s_2 \xrightarrow{LCA,*} (\lambda x.s'_1)\ (\lambda x.s'_2) \xrightarrow{LCA,*} s'_1[\lambda x.s'_2/x] \xrightarrow{LCA,*} s_0$ where $s_0\Uparrow$. Then $s_i\downarrow\lambda x.s'_i$ for $i = 1, 2$ and by Proposition C.7 there are reductions $t_i\downarrow\lambda x.t'_i$ for $i = 1, 2$ with $s_i \preccurlyeq_{\Downarrow H} t_i$. Thus $s'_1[\lambda x.s'_2/x] \preccurlyeq_{\Downarrow H} t'_1[\lambda x.t'_2/x]$, and hence by the induction hypothesis $t'_1[\lambda x.t'_2/x]\uparrow$. Thus $(t_1\ t_2)\uparrow$, and from $(t_1\ t_2) \simeq_{\Downarrow} t$ we obtain $t\uparrow$.

Now we make use of the transitive closure trick explained in [22].
Let $\preccurlyeq^*_{\Downarrow H}$ be the transitive closure of $\preccurlyeq_{\Downarrow H}$.

Proposition C.9. *The claims of Propositions C.7 and 4.13 also hold for* $\preccurlyeq^*_{\Downarrow H}$.

Proof. By induction on the construction of the transitive closure.

Proposition C.10. $(\preccurlyeq^*_{\Downarrow H})^c$ *satisfies the fixpoint conditions of* $\simeq_{\Downarrow}$.

Proof. This follows from Proposition C.9 and from the symmetry of $\preccurlyeq^*_{\Downarrow H}$ proved in Lemma C.2.

Theorem C.11. *The relation* $\simeq_{\Downarrow}$ *is a congruence on closed expressions and* $\simeq^o_{\Downarrow}$ *is a congruence on all expressions.*

Proof. We already have $\simeq_{\Downarrow}\ \subseteq\ \preccurlyeq^c_{\Downarrow H}\ \subseteq\ (\preccurlyeq^*_{\Downarrow H})^c$ by Lemma 4.3 part 3. By coinduction, since $(\preccurlyeq^*_{\Downarrow H})^c$ satisfies the fixpoint conditions of $\simeq_{\Downarrow}$, we obtain $(\preccurlyeq^*_{\Downarrow H})^c\ \subseteq\ \simeq_{\Downarrow}$, which shows $(\preccurlyeq^*_{\Downarrow H})^c\ =\ \simeq_{\Downarrow}$. Also, since $((\preccurlyeq^*_{\Downarrow H})^c)^o = \preccurlyeq^*_{\Downarrow H}$, the equation $(\preccurlyeq^*_{\Downarrow H})\ =\ \simeq^o_{\Downarrow}$ holds.

Theorem C.12. $\simeq^o_{\Downarrow}$ *is sound for* $\sim_{LCA}$.

Proof. By Theorem Let $s \simeq^o_{\Downarrow} t$, and let C be a context such that $C[s], C[t]$ are closed with $C[s]\uparrow$. Theorem C.11 shows that $C[s] \simeq^o_{\Downarrow} C[t]$, and so $C[t]\uparrow$. The other implication follows from symmetry of $\simeq^o_{\Downarrow}$. We also have $C[s]\downarrow \iff C[t]\downarrow$, which follows from Lemma C.4 and Theorem B.8.

D Incompleteness of May-Divergence Simulation in *LCA*

We argue for incompleteness of $\preccurlyeq_{\uparrow}$. The idea is to construct an $\leq_{\uparrow}$-ascending chain of expressions B_i with limit A, such that the infinite choice of B_i is contextually equivalent with A, which cannot be detected by the simulation. First we define expressions and a series of expressions, recursively:

- $A = \mathit{choice}\ \Omega\ (\lambda x.A)$.
- $B_0 = \mathit{Top}$, $B_{i+1} = \lambda x.\mathit{choice}\ \Omega\ B_i$;
- $B = \mathit{choice}\ \Omega\ (\mathit{choice}\ B_0\ (\mathit{choice}\ B_1\ \ldots))$.

Eliminating the recursion, we get the following non-recursive definitions:

- $A = Y\ (\lambda a.\mathit{choice}\ \Omega\ (\lambda x.a))$.
- First define the recursive function $b = \lambda bi.\mathit{choice}\ bi\ (b\ \lambda x.\mathit{choice}\ \Omega\ bi)$ with the intention to define B as $\mathit{choice}\ \Omega\ (b\ \mathit{Top})$. Then:
  $B = \mathit{choice}\ \Omega\ ((Y\ \lambda b.\lambda bi.\mathit{choice}\ bi\ (b\ \lambda x.\mathit{choice}\ \Omega\ bi))\ \mathit{Top})$

Lemma D.1.

1. *$Top \approx_{\downarrow} A \approx_{\downarrow} B_i$ for all i.*
2. *$Top \approx_{\downarrow} B$.*
3. *$B_i <_{\uparrow} A$ for all i.*

Proof. Item (1) is shown using simulation for may-convergence (see Theorem 3.6). The relations $A \approx_{\downarrow} Top$ and $Top \approx_{\downarrow} choice\ \Omega\ Top$ hold. Also, $Top \approx_{\downarrow} \lambda x.choice\ \Omega\ Top$ by applying simulation. Using that $\approx_{\downarrow}$ is a congruence, item (1) follows. Also $B \approx_{\downarrow} Top$ holds using simulation in both directions, which follows from item (1).

Item (3) holds using simulation for may-divergence, using the previous items, and since simulation $\preccurlyeq_{\uparrow}$ is sound for $\leq_{\uparrow}$ (see Theorem 3.6). The other direction does not hold, since $A\uparrow$, but $B_i\Downarrow$.

The following context lemma in LCA holds:

Lemma D.2. *Let s, t be closed LCA-expressions.*

1. *If for all closed evaluation contexts E: $E[s]\downarrow \implies E[t]\downarrow$, then $s \leq_{\downarrow} t$.*
2. *If $s \sim_{\downarrow} t$ and for all closed evaluation contexts E: $E[s]\uparrow \implies E[t]\uparrow$, then $s \leq_{\uparrow} t$.*

Proof. The proof is a simple variant of the context lemma proofs using induction on the length and size of multicontexts, which is unproblematic since s, t are closed.

Lemma D.3. *$A \sim_{LCA} B$.*

Proof. $A \leq_{LCA} B$ follows from Lemma D.1 using simulation for may-divergence. (Note that $\leq_{LCA} = \leq_{\downarrow} \cap (\leq_{\uparrow})^{-1}$.) The other direction $B \leq_{LCA} A$ requires an application of the context lemma D.2. Let $E[A]\uparrow$ for some closed evaluation context E. Then there is a reduction of length n to a must-divergent expression.

We mimic this reduction for $E[B]$. If the first choice of A is Ω, then we do the same for B, and obtain equal expressions. If the first choice is $\lambda x.A$, then we select B_{n+1} for B. Thus we have to compare $E[\lambda x.A]$ and $E[B_{n+1}] \xrightarrow{LCA,*} E[\lambda x.\texttt{choice}\ \Omega\ B_n]$. Generally, we have a reduction $E[A] \xrightarrow{LCA} C_1[A'_1, \ldots, A'_k] \xrightarrow{LCA} \ldots \xrightarrow{LCA} C_n[A''_1, \ldots A''_m]\Uparrow$, where the expressions A'_i and A''_i are either Ω or $\lambda x.A$. For $E[\lambda x.\texttt{choice}\ \Omega\ B_n]$ we mimic the $E[A]$-reduction by either using the same reduction, or, in the case that $((\lambda x.A)\ r)$ has to be beta-reduced, by either also choosing Ω at the according position, or by choosing B_{j_i}, where $j_i \geq 1$. With this construction we derive a reduction $E[\lambda x.B_{n+1}] \xrightarrow{LCA} C_n[D_1, \ldots, D_m]$ where $D_i = \Omega$ if $A''_i = \Omega$; or $D_i = B_{j_i}$ for some $j_i \geq 1$ if $A''_i = \lambda x.A$. Note that $A''_i \sim_{\downarrow} D_i$ by Lemma D.1 (since also $A \sim_{\downarrow} \lambda x.A$), and thus $C_n[A''_1, \ldots, A''_m] \sim_{\downarrow} C_n[D_1, \ldots, D_m]$. Thus $C_n[A''_1, \ldots, A''_m]\Uparrow$ implies $C_n[D_1, \ldots, D_m]\Uparrow$, and hence $E[B]\uparrow$. This holds for all E, hence by the context lemma: $B \leq_{LCA} A$.

Proposition D.4. *$B \preccurlyeq_{\uparrow} A$ holds.*

Proposition D.5. *$A \not\preccurlyeq_{\uparrow} B$. Hence $\preccurlyeq_{\uparrow}$ and $\approx_{\uparrow}$ are incomplete.*

Proof. Should-similarity requires that $A \preccurlyeq_{\uparrow} B_i$ for some i. This in turn requires that $A \preccurlyeq_{\uparrow} B_{i-1}$, and so it is sufficient to refute $A \preccurlyeq_{\uparrow} B_0$, which again requires $A \preccurlyeq_{\uparrow} Top$, which does not hold

Comparing s, t for $\leq_{\uparrow}$, the incompleteness of $\preccurlyeq_{\uparrow}$ cannot appear if t reduces to only finitely many abstractions.

Lemma D.6. *If s is a closed abstraction and t is a closed expression such that $s \leq_{\uparrow} t$, and for some $n \geq 2$ there is a set $T := \{t_1, \ldots, t_n\}$ of closed expression, such that $t\downarrow\lambda x.t'$ implies $\lambda x.t' \in T$. Then there is some i with $s \leq_{\uparrow} t_i$.*

Proof. Suppose this is false. Then there are contexts $C_1, \ldots, C_n$, such that $C_i[s], C_i[t_i]$ are closed for all i, and for all $i = 1, \ldots, n$: $C_i[s]\uparrow$ and $C_i[t_i]\Downarrow$. The context $C = (\lambda x.\texttt{amb}\ C_1[x]\ (\texttt{amb}\ \ldots(\texttt{amb}\ C_{n-1}[x]\ C_n[x])))\ [\cdot]$ has the property: $C[s]\uparrow$, but $C[t]\Downarrow$, which is a contradiction.