# On Generic Context Lemmas for Lambda Calculi with Sharing

Manfred Schmidt-Schauß and David Sabel

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

## Technical Report Frank-27

6. Januar 2007

**Abstract.** This paper proves several generic variants of context lemmas and thus contributes to improving the tools to develop observational semantics that is based on a reduction semantics for a language. The context lemmas are provided for may- as well as two variants of must-convergence and a wide class of extended lambda calculi, which satisfy certain abstract conditions. The calculi must have a form of node sharing, e.g. plain beta reduction is not permitted. There are two variants, weakly sharing calculi, where the beta-reduction is only permitted for arguments that are variables, and strongly sharing calculi, which roughly correspond to call-by-need calculi, where beta-reduction is completely replaced by a sharing variant. The calculi must obey three abstract assumptions, which are in general easily recognizable given the syntax and the reduction rules. The generic context lemmas have as instances several context lemmas already proved in the literature for specific lambda calculi with sharing. The scope of the generic context lemmas comprises not only call-by-need calculi, but also call-by-value calculi with a form of built-in sharing. Investigations in other, new variants of extended lambda-calculi with sharing, where the language or the reduction rules and/or strategy varies, will be simplified by our result, since specific context lemmas are immediately derivable from the generic context lemma, provided our abstract conditions are met.

**Keywords:** lambda calculus, observational semantics, context lemma, functional programming languages

## 1 Introduction

A workable semantics is indispensable for every formal modelling language, in particular for all kinds of programming languages. This paper will make a contribution to the tools, in particular so-called context-lemmas, which support

reasoning about semantical properties of higher-order functional programming languages and lambda-calculi. Generally, an application of semantics is to obtain knowledge about the evaluation of programs and the correctness of program transformations. Usually, the essence of higher-order formalisms is defined in form of an extended lambda calculus.
For various lambda calculi a widely used observational semantics is contextual equivalence based on a reduction semantics in the style of Morris [Mor68], i.e. two expressions are equal if their termination behavior is always the same when they are plugged into an arbitrary program context. Given a calculus consisting of a language of terms, a small step reduction relation $\rightarrow$ on terms, and a set of answer terms, a term is called may-convergent if there exists a finite sequence of $\rightarrow$-reductions starting with the term and ending in an answer. Usually answers are weak head normal forms for call-by-need and call-by-name calculi, and weak normal forms for call-by-value calculi. For non-deterministic calculi, contextual equivalence must be based on the conjunction of two termination behaviors (see e.g. [Ong93]): May-convergence and must-convergence, where the latter takes all reduction possibilities into account. There are two different definitions of must-convergence in the literature:

– iff every $\rightarrow$-successor of $t$ is may-convergent.
– iff every maximal sequence of reductions starting with $t$ ends in answer. We will call this form of must-convergence also total must-convergence.

The first definition of must-convergence also includes terms that may evaluate infinitely but the chance of finding an answer is never lost. These terms are called weakly divergent in [CHS05]. Note that a similar combination of may- and must-convergence is also known from the use of convex powerdomains in domain-theoretic models [Plo76].
In this paper we will consider several variants of contextual approximations and equivalences based on may-, must- and/or total must-convergence. Usually, a first step and strong tool for further proof techniques is to prove a context lemma that reduces the test for convergence (may- and/or must-) to a subclass of contexts, the reduction contexts (also called evaluation contexts), instead of all contexts. This technique dates back to [Mil77] for showing full abstractness of denotational models of lambda-calculi.
In this paper, we formulate abstract conditions on an extended lambda-calculus and its reduction semantics, and then prove generic context lemmas for the three types of convergences for calculi satisfying these conditions. Our result can be applied to two forms of lambda calculi: strongly sharing and weakly sharing calculi. In these sharing calculi, the normal-order reduction must follow a strict discipline in only modifying reduction positions, with the exception of perhaps replacing variables by variables. They could also be seen as top-down evaluating calculi. In strongly sharing calculi, the (normal-order) reduction may only modify non-reduction positions through renamings. E.g. the full beta-rule $(\lambda x.s)\ t \rightarrow s[t/x]$ violates our assumptions, since there may be non-reduction occurrences of $x$ in $s$ that are replaced by the beta-rule. The restricted beta-rule $(\lambda x.s)\ y \rightarrow s[y/x]$ may be allowed in weakly sharing calculi, whereas the rules

$(\lambda x.s)\ t \to (\texttt{let}\ x = t\ \texttt{in}\ s)$ and $(\texttt{let}\ x = v\ \texttt{in}\ R[x]) \to (\texttt{let}\ x = v\ \texttt{in}\ R[v])$ are fine for strongly sharing calculi. The corresponding rule in explicit substitution calculi is compatible, though not all the rules given in the explicit substitution calculi [ACCL91]. Examples for calculi, where the context-lemma for may-convergence is immediately applicable, are the deterministic calculi [AFM$^+$95,AF97,MOW98,AS98,SS06]. Non-deterministic calculi where also the must-context lemmas are applicable, are [KSS98,Man05,SSS07,NSSSS06], the latter is the calculus from [NSS06] with some adaptations.
The context lemma is an important tool for further investigations into correctness of program transformations, for example, the diagram methods in [SSSS05,SSS07,NSSSS06] demonstrate their strength only if the context lemma is proved. There is no context lemma used in [KSS98], which severely complicates the diagram-proofs.
There is also related work on context lemmas for calculi not satisfying our conditions. For call-by-value languages with beta-reduction, there is a weaker form of a context lemma, the so-called CIU-theorem, which was first proved in [FH92], which also holds for a class of languages, and was even formally checked by an automated reasoner (see [FM01,FM03]). For PCF-like languages, also with full beta-reduction, there is also a context lemma proved for a class of languages extending PCF [JM97]. Another related generic tool is bisimilarity for extended lambda-calculi [How89,How96], and for typed languages [Gor99]. An extension for calculi with sharing w.r.t. may-convergence is done for a non-deterministic calculus in [Man05] and for a class of calculi in [MSS06].
The structure of this paper is a follows. After presenting the abstract syntax for higher-order calculi, the assumptions on the calculi are presented and discussed. Section 4 presents the different convergence relations and contextual approximations, Section 5 contains the proofs of the various generic context lemmas, and the final section 6 contains a discussion on the range of calculi where the instances of the generic context lemmas hold.

## 2 Abstract Syntax and Language

For the generic formulation we use higher-order abstract syntax, see e.g. [How89,How96]. The construction of terms of the language requires variables, operators (i.e. symbols with arity), and variable-binding primitives. We allow the extension by a recursive `letrec` which is used as an extra operator with its own binding rules.

**Definition 2.1.** *Let $\mathcal{L} = (O, \alpha, \beta)$ be a signature, where $O$ is a set of* operators, *which may include* `letrec`*. For every operator $f \in O \setminus \{\texttt{letrec}\}$, $\alpha(f) \in \mathbb{N}_0$ is the arity, and $\beta(f)$ is an $\alpha(f)$-tuple with components in $\mathbb{N} \cup \{\text{“}V\text{”}, \text{“}T\text{”}\}$, indicating the number of possible variables that may be bound at the corresponding argument and if there are no binders, then whether variables terms are permitted (“V”), or terms (“T”). Terms $\mathcal{T}(\mathcal{L})$ are inductively defined as follows:*

– *Every $x \in V$ is a term. (There are infinitely many variables)*

– *If $f \in O \setminus \{\texttt{letrec}\}$ with $\alpha(f) = 0$, then $f$ is a term.*
– *If $f \in O \setminus \{\texttt{letrec}\}$, then $f(a_1, \ldots, a_n)$ is a term if $n = \alpha(f) \geq 1$ and for every $i = 1, \ldots, n$: if $\beta(f)_i =$ "$V$", then $a_i$ is a variable, if $\beta(f)_i =$ "$T$", then $a_i$ is a term, and if $\beta(f)_i = m \in \mathbb{N}$, then $a_i$ is of the form $x_1, \ldots, x_m \,.\, t_i$, where $x_1, \ldots, x_m$ are different variables, and $t_i$ is a term* [1]*.*
– *If $\texttt{letrec} \in O$, $n \geq 0$, $x_1, \ldots, x_n$ are different variables, and if $t_1, \ldots, t_n, s$ are terms, then ($\texttt{letrec}$ $x_1 = t_1, \ldots, x_n = t_n$ $\texttt{in}$ $s$) is a term.*

*As usual, the scope of every variable $x_i, i = 1, \ldots, n$ in $x_1, \ldots, x_m \,.\, t$ is the term $t$, and the scope of every variable $x_i$ in a* `letrec` *as above is the set of terms $t_1, \ldots, t_n, s$. Variable occurrences that are in the scope of a binder that binds them, are called* bound *occurrences of variables, others are* free occurrences *of variables.*

The set of *free variables* of a term $t$ is denoted as $\mathcal{FV}(t)$. As usual, a term $t$ is *closed* if all of its variables are bound, i.e. $\mathcal{FV}(t) = \emptyset$, otherwise it is called *open*. Since we have to deal in depth with different kinds of renamings, we do not assume anything about implicit renamings, though it is known how to correctly rename terms (cf. [Bar84]).

*Example 2.2.* With $O = \{\lambda, \texttt{letrec}, @, \texttt{Cons}, \texttt{Nil}, \texttt{case}\}$ a language with `let(rec)`, application, abstractions, lists and a `case` is defined. The description is $\alpha(\lambda) = 1$, $\beta(\lambda) = (1)$, $\alpha(@) = 2$, $\beta(@) =$ ("$T$", "$T$"), `Cons` is specified like @, $\alpha(\texttt{Nil}) = 0$, and $\alpha(\texttt{case}) = 3, \beta(\texttt{case}) =$ ("$T$", 2, "$T$"). The usual lambda-term $\lambda x.x$ is represented as $\lambda(x.x)$). A term like `let x = (Cons x Nil) in case x of (Cons z1 z2) -> y; Nil -> Nil` would be expressed as ($\texttt{letrec}$ $x = \texttt{Cons}(x, \texttt{Nil})$ $\texttt{in}$ $\texttt{case}(x, z_1 z_2.y, \texttt{Nil})$).

Examples of languages with the variable restriction are: [MSC99] where arguments of applications are only variables, and the language in [NSSSS06], where the first argument in cell-expressions ($x$ `c` $t$) must be a variable.
We will use *positions* to address subterms, variables and variables in binders using a slightly extended Dewey decimal notation. The addressing is such that prefixes of addresses of term positions are term positions. We write $s_{|p}$ for the subterm of $s$ at position $p$, and $s(p)$ for the head-symbol of the subterm $s_{|p}$. For example, the term $t = \texttt{Cons}(\lambda x.@(x, @(y, x)), z)$ has e.g. the following term positions: $y$ is at position 1.1.2.1, $x$ occurs at positions 1.1.1 and 1.1.2.2, and $t(1) = \lambda$ and $t(1.1.1) = x$. We also assume that there is a virtual binder for free variables at the top.
A context $C$ is like a term, where the hole $[\cdot]$ is allowed at a single term-position. The expression $C[s]$ denotes the result of plugging in a term $s$ into $C$, where capture of variables is permitted. We will also use *multi-contexts $M$* that may have more holes.

**Definition 2.3 (Distinct Variable Convention).** *A term $t$ satisfies the* distinct variable convention (DVC)*, iff all bound variables in $t$ are distinct, and moreover, all bound variables in $t$ are distinct from all the free variables in $t$.*

[1] The expressions $x_1, \ldots, x_m \,.\, t_i$ are called operands in [How96]

## 2.1 Renamings and Substitutions

A renaming may rename bound variables in a term $t$. It is called *capture-free*, iff the relation between occurrence of a variable and its binding position remains unchanged. A renaming of variables of the term $s$ is described by a finite set $S$ of pairs $(p, x \mapsto y)$, where $p$ is a binding position of $x$, and renaming means to apply all the replacements $x \mapsto y$ to all variables $x$ in the scope of the binder at $p$, and also to the binder. A *bv-renaming* is defined as a capture-free renaming of bound variables (the virtual binder is not allowed in this case) of the term $s$. If $t$ can be reached from $s$ by a sequence of bv-renamings, then we define this as $s =_\alpha t$. An *fvbv-renaming* is defined as a capture-free renaming of free and bound variables of a term $s$ that is injective on $\mathcal{FV}(s)$. With $\mathrm{fvp}(\sigma)$ we denote that mapping on $\mathcal{FV}(s)$ for a fvbv-renaming $\sigma$ of $s$. If $\{x_1 \mapsto y_1, \ldots, x_n \mapsto y_n\}$ is the representation of the mapping on variables, then $\{x_1, \ldots, x_n\}$ is the *domain*, and $\{y_1, \ldots, y_n\}$ the *codomain*. A *vvbv-substitution* $\gamma$ of a term $s$ is like an fvbv-renaming, but it is permitted to have name clashes for the free variables of $s$. We also use $\mathrm{fvp}(\gamma)$, which we call in this case a *vv-substitution*, denoted as $\nu$. A vv-substitution may cause a capture of variables, however, vvbv-substitutions have the built-in property that no variable capture is permitted. Given a term $s$, a vv-substitution $\nu$, a set of variables $\mathcal{FV}(s) \subseteq W$, and a vvbv-substitution $\gamma$. Then we say $\gamma$ is *derived* from $\nu$ on $W$, iff $\mathrm{fvp}(\gamma)(x) = \nu(x)$ for all $x \in W$. This notion is also used for fvbv-renamings.

*Example 2.4.* The term $\lambda y.x$ can be modified into $\lambda x.y$ by an fvbv-renaming (or a vvbv-substitution). However, this cannot be represented as $\sigma\mu_1$ for a bv-renaming $\mu_1$ and a substitution $x \mapsto y$, nor as $\mu_1\sigma$. It can only be represented as $\sigma_1\nu_1\sigma_2$ with $\sigma_2 = \{y \mapsto z\}$, and $\sigma_1 = \{z \mapsto x\}$.

**Lemma 2.5.**

- *If* $t_1 \xrightarrow{\sigma_1} t_2 \xrightarrow{\sigma_2} t_3$ *by vvbv-substitutions* $\sigma_1, \sigma_2$*, then the composition* $\sigma_3$ *is an vvbv-substitution with* $\mathrm{fvp}(\sigma_3)(x) = \mathrm{fvp}(\sigma_2)\mathrm{fvp}(\sigma_1)(x)$ *for all* $x \in \mathcal{FV}(t_1)$*. This can be specialized to fvbv-renamings and bv-renamings.*
- $s =_\alpha t$ *iff* $s \xrightarrow{\sigma} t$ *by a single bv-renaming* $\sigma$*.*
- *If* $s \xrightarrow{\sigma} t$ *by an fvbv-renaming, then the reverse renaming is also capture-free, i.e. a fvbv-renaming. Again this holds also for bv-renamings.*
- *If* $s$ *is a term not satisfying the DVC, then there is a term* $s'$ *satisfying the DVC, and a bv-renaming* $\sigma$ *with* $\sigma(s) = s'$*. This can be accomplished by renaming bound variables with fresh variables.*
- *If* $s \xrightarrow{\sigma} t$ *is an vvbv-substitution, and* $\rho = \mathrm{fvp}(\sigma)$*, then there are bv-renamings* $\sigma_1, \sigma_2$*, such that* $s \xrightarrow{\sigma_1}\xrightarrow{\rho}\xrightarrow{\sigma_2} t$*.*
- *Given a fvbv-renaming* $\sigma$ *of a term* $s$*. Then* $\mathrm{fvp}(\sigma)$ *is injective on* $\mathcal{FV}(s)$*.*

It is interesting to note that terms and vvbv-substitutions form a category with terms as objects; the same holds for fvbv-renamings and bv-renamings. We will also use fvbv-renamings and bv-renamings for contexts and multicontexts.

**Definition 2.6.** *Let $C$ be a (one-hole) context. Then $BP_{\text{hole}}(C)$ is defined as the set of binder-positions in $C$ that have the hole of $C$ in their scope, $BP_{\overline{\text{hole}}}(C)$ is the complement, i.e. the set of binder-positions that do not have the hole in their scope, and $V_{\text{hole}}(C)$ is the set of variables that are bound by the binders in $BP_{\text{hole}}(C)$.*

It is obvious that all variables bound by binders in $BP_{\text{hole}}(C)$ are different.

**Lemma 2.7.** *Let $C$ be a context, $s$ be a term, and $\sigma$ be an fvbv-renaming of $C[s]$. Then $\sigma$ can be splitted into an fvbv-renaming $\sigma_C$ of $C$, and an fvbv-renaming $\sigma_s$ of $s$, where the mapping $\text{fvp}(\sigma_s)$ is injective on $V_{\text{hole}}(C)$ as well as on $\mathcal{FV}(s)$. Also the mapping induced by $\sigma$ on $V_{\text{hole}}(C)$ is injective.*

## 3 Generic Lambda Calculi with Sharing

In this section we describe the required notions and the required abstract properties. We assume that for the calculus CALC the following is given:

- An algorithm UNWIND, detecting all the potential reduction positions (which must be term positions). We assume that the algorithm has a term or a multicontext as input and non-deterministically produces a sequence of positions, all of which are reduction positions.
- A small-step reduction relation $\rightarrow_0$ on terms, where $s \rightarrow_0 t_0$ is defined only for terms $s$ satisfying the DVC, but the term $t_0$ is not further restricted. The small-step reduction $s \rightarrow t$ is then defined as $s \rightarrow_0 t_0 \rightarrow_1 t$, where $\rightarrow_1$ is a bv-renaming (see Assumption 3.4).
- A set of answers ANS, which are accepted as successful results of reductions.

We distinguish *weakly* and *strongly sharing calculi* in the following, where strongly sharing calculi do not make use of vv-substitutions. To ease notation we use a set $VV$ of vv-substitutions, with $VV = \{Id\}$ for *strongly sharing calculus* and $VV$ the set of all vv-substitutions for weakly sharing calculi.
In the considered calculi, the reduction $\rightarrow$ is usually the (call-by-need) normal-order reduction, and the answers are the WHNFs.
The algorithm UNWIND has a term or a multicontext $t$ as input and (perhaps non-deterministically) produces a sequence of term-positions, starting with $p_1 = \varepsilon$. Given $t$, the possible sequences $p_1, p_2, \ldots$ produced by UNWIND are called the *valid runs* of $t$. We do not enforce the sequences to be maximal. The following conditions must hold:

*Assumption 3.1 (*UNWIND*-Assumptions).*

1. If $p_1, \ldots, p_n$ is a valid run, then for $1 \leq i \leq n$, $p_1, \ldots, p_i$ is also a valid run, the set $\{p_1, \ldots, p_n\}$ is a prefix-closed set of positions, and $p_n \notin \{p_1, \ldots, p_{n-1}\}$.
2. If $t'$ is a term or a multicontext, $p_1, \ldots, p_n$ is a valid run for $t$, and for all $i < n$, we have $t(p_i) = t'(p_i)$, then $p_1, \ldots, p_n$ is also a valid run for $t'$.

3. If $t'$ is a term or multicontext, and $\nu \in VV$, $\gamma$ a vvbv-substitution derived from $\nu$ with $t' = \gamma(t)$, and $p_1, \ldots, p_n$ is a valid run for $t$, then $p_1, \ldots, p_n$ is also a valid run for $t'$.

Given a term or multicontext $t$. Then the position $p$ of $t$ is a *reduction position* in $t$, iff $p$ is contained in some valid run of $t$. The set of all reduction positions is defined as $\mathrm{RP}(t) = \{p \mid p \text{ is contained in some valid run of } t\}$. Note that every reduction position is a term-position by definition. Since we also apply the formalism to multicontexts, we can speak of reduction positions of multicontexts as well as of terms. A single-hole context $C[]$ is defined as a *a reduction context*, if the hole $[]$ of $C$ is a reduction position in $C[]$: We denote reduction contexts as $R[]$.

**Lemma 3.2.** *Let $M$ be a multicontext with $n$ holes, and $s_j, j = 1, \ldots, n$ be terms, such that for some $i$: $M[s_1, \ldots, s_{i-1}, [], s_{i+1}, \ldots s_n]$ is a reduction context. Then there is some $j \in \{1, \ldots, n\}$, such that for all terms $t_k, k = 1, \ldots, n$, $M[t_1, \ldots, t_{j-1}, [], t_{j+1}, \ldots t_n]$ is a reduction context.*

*Proof.* Let $p$ be the position of the hole in $t := M[s_1, \ldots, s_{i-1}, [], s_{i+1}, \ldots s_n]$. By the assumption on UNWIND, there is a valid run $p_1, \ldots, p_m$ of $t$, such that $p_m = p$. Let $Q$ be the set of the positions of the $n$ holes of $M$. Then there is a least $k$ such that $p_1, \ldots, p_k$ is a valid run of $t$, $p_k \in Q$ and $p_k$ is the position of some hole, say the $j^{th}$ hole. Minimality of $k$ implies that $p_1, \ldots, p_{k-1}$ are positions within $M$, but not the position of any hole of $M$. Now we can apply the conditions on UNWIND, in particular condition (2): UNWIND produces the valid run $p_1, \ldots, p_k$, irrespective of the terms in the holes of $M$. Hence the claim of the lemma holds. □

*Assumption 3.3 (Answer-Assumption).* There is a set ANS of *answer terms.* We assume that the following conditions are satisfied:

1. If $t \xrightarrow{\sigma} t'$ for terms $t, t'$ by a fvbv-renaming $\sigma$, and $t \in \mathrm{ANS}$, then $t' \in \mathrm{ANS}$.
2. If $t = M[t_1, \ldots, t_n]$ is an answer for some multicontext $M$, no hole of $M$ is a reduction position, then $M[t'_1, \ldots, t'_n]$ is also an answer.

The essence of the following assumption is that reduction commutes with renaming, and that reduction in strongly sharing calculi does not modify non-reduction positions, and in weakly sharing calculi, it may replace variables by variables.

*Assumption 3.4 (Reduction-Assumption).* It is assumed that CALC only defines a (small-step) relation $\to_0$ that is applicable to terms satisfying the DVC, and that the full small-step relation $\to$ is derived from $\to_0$ and a subsequent renaming of variables. The relation $\to$ is defined such that $s \to t$ holds whenever $s, t$ satisfy the DVC, and $s \to_0 t_0 \xrightarrow{\sigma} t$ for some $t_0$ and some bv-renaming $\sigma$ of $t_0$. We assume that the following conditions are satisfied for $\to_0$:

1. Let $t = M[t_1, \ldots, t_n]$ be a term that satisfies the DVC, where $M$ is a multicontext with $n$ holes that are at non-reduction positions, and let

$t'$ be a term with $t \to_0 t'$. Then there is a multicontext $M'$ with $n'$ holes, a mapping $\pi : \{1,\ldots,n'\} \to \{1,\ldots,n\}$, vv-substitutions $\nu_i \in VV, i \in \{1,\ldots,n'\}$, where the domain and codomain-variables already occur in $M$, such that for all terms $s_1,\ldots,s_n$: If $M[s_1,\ldots,s_n]$ satisfies the DVC, then $M[s_1,\ldots,s_n] \to_0 M'[\nu_1(s_{\pi(1)}),\ldots,\nu_{n'}(s_{\pi(n')})]$. In particular, $t' = M'[\nu_1(t_{\pi(1)}),\ldots,\nu_{n'}(t_{\pi(n')})]$.
Note that $\nu$ is capture-free since DVC holds before reduction.
2. If $s$ is a term satisfying the DVC, $s \to_0 t$, $s \xrightarrow{\sigma} s'$ an fvbv-renaming of $s$, such that $s' := \sigma(s)$ satisfies the DVC. Then there is a term $t'$ and an fvbv-renaming $\sigma'$ of $t$, where $\mathrm{fvp}(\sigma')$ is a restriction of $\mathrm{fvp}(\sigma)$, such that $s' \to_0 t'$ and $t' = \sigma'(t)$.

$$\begin{tikzcd} s \arrow[r, "\sigma"] \arrow[d, "0"'] & s' \arrow[d, dashed, "0"] \\ t \arrow[r, dashed, "\sigma'"'] & t' \end{tikzcd}$$

There is no easy generalization of the first assumption to calculi with beta-reduction: Let $M[.,.] := (\lambda x.[])(\lambda y.[])$, and $s_1 = (f\ x)$, then $M[s_1, s_2] \xrightarrow{\beta} f(\lambda y.s_2)$.

## 4 Contextual Preorder and Equivalence for May- and Must-Convergence

In this section we define different kinds of convergence properties of terms, and the corresponding notions of contextual preorder and equivalence. There are three main notions of convergence of a term $t$: may-convergence, which means that $t$ may reduce to an answer, total must-convergence, which means that $t$ has no reduction to a must divergent term (failure term) and no infinite reduction, and (strong) must-convergence, which means that every term reachable by reduction from $t$ is may-convergent.

**Definition 4.1.** *A term $t$ is called*

- may-convergent *iff there is a some answer $t'$ with $t \xrightarrow{*} t'$, denoted as $t\downarrow$.*
- must-divergent *iff $t$ is not may-convergent, denoted as $t\Uparrow$.*
- may-divergent, *iff there is some term $t' \Uparrow$ with $t \xrightarrow{*} t'$, denoted as $t\uparrow$.*
- must-convergent *iff it $t \xrightarrow{*} t'$ implies $t'\downarrow$, denoted as $t\Downarrow$.*
- totally must-convergent *iff $t \Downarrow$ and there is no infinite reduction starting with $t$, denoted as $t \Downarrow\!\!\Downarrow$.*
- totally may-divergent *iff $t\uparrow$, or there is an infinite reduction starting with $t$, denoted as $t \uparrow\uparrow$.*

Note that $t$ is not may-divergent iff it is must-convergent. Note also that total must-divergence is the same as must-divergence, and total may-convergence is the same as may-convergence.
In calculi with a deterministic $\to_0$-reduction the may- and must-predicates are identical. We could call a calculus *deterministic* iff the may- and must-convergence predicates are identical for all terms. Terms $t$ with $t\Downarrow$, but not $t \Downarrow\!\!\Downarrow$, are called *weakly divergent* in [CHS05]. Must-convergence is interesting because it is linked to fairness (see e.g. [CHS05,SSS07,NSSSS06]); further justification for non-total may-divergence is in [SS03].

**Definition 4.2.** *Let $s, t$ be two terms (of the same type), and $\mathcal{M} \in \{\downarrow, \Downarrow, \Downarrow\!\!\Downarrow\}$. Then $s \leq_{\mathcal{M}} t$ iff for all $C[] : C[s]\mathcal{M} \implies C[t]\mathcal{M}$, and*
*$s \leq_{\mathcal{M}\nu} t$, iff for all $C[]$, for all vv-substitutions $\nu$ and for all vvbv-substitutions $\gamma_s, \gamma_t$ derived from $\nu$ on $\mathcal{FV}(s) \cup \mathcal{FV}(t)$: $C[\gamma_s(s)]\mathcal{M} \implies C[\gamma_t(t)]\mathcal{M}$.*

Easy consequences are that for all terms $s$: $s\Downarrow\!\!\Downarrow \implies s\Downarrow \implies s\downarrow$, that for $\mathcal{M} \in \{\downarrow, \Downarrow, \Downarrow\!\!\Downarrow\}$, the relations $\leq_{\mathcal{M}}$ are compatible with contexts, and reflexive and transitive, and that $\leq_{\mathcal{M}\nu} \subseteq \leq_{\mathcal{M}}$.
Note that for a general proof of transitivity of $\leq_{\mathcal{M}}$ it is unavoidable that $C[s], C[t]$ may contain free variables. The corresponding proof w.r.t. a definition of $\leq_{\mathcal{M}}$ that restricts $C[s], C[t]$ to be closed is in general not applicable, since the middle term $C[s_2]$ is not necessarily closed, if $C[s_1], C[s_3]$ are closed.
*Contextual equivalence* is defined as $\sim_{\downarrow} := \leq_{\downarrow} \cap \geq_{\downarrow}$ for deterministic calculi and for nondeterministic calculi as $\sim_{\downarrow\Downarrow} := \sim_{\downarrow} \cap \leq_{\Downarrow} \cap \geq_{\Downarrow}$ or $\sim_{\downarrow\Downarrow\!\!\Downarrow} := \sim_{\downarrow} \cap \leq_{\Downarrow\!\!\Downarrow} \cap \geq_{\Downarrow\!\!\Downarrow}$ depending on the used must-convergence predicate. The relations $\sim_{\downarrow\nu}$ are defined analogously using the respective $\leq$-relations.

*Example 4.3.* The relation $\leq_{\downarrow\nu}$ may be different from $\leq_{\downarrow}$ (in exotic calculi): Consider the calculus with one binary constructor $c$ and a constant $d$, and the reduction rule: $c\ x\ x \to d$, let $d$ be the only answer, and let all positions be reduction positions. Then $c\ x\ y \leq_{\downarrow} c\ x\ z$, but $c\ x\ y \not\leq_{\downarrow\nu} c\ x\ z$.

For simplifying several proofs in the following sections, we introduce a *0-1-labelled variant* of $\to$-reduction sequences, which is nothing else but a reduction of the form $s_{1,0} \to_0 s_{1,1} \to_1 s_{2,0} \to_0 s_{2,1} \to_1 \ldots$. I.e., a reduction, where $\to_0$-reductions and bv-renamings $\to_1$ are alternating, and the terms $s_{i,0}$ satisfy the DVC.

$$\begin{tikzcd}
s_{1,0} \arrow[r, "\sigma_{1,0}"] \arrow[d, "0"'] & t_{1,0} \arrow[d, dashed, "0"] \\
s_{1,1} \arrow[r, dashed, "\sigma_{1,1}"'] \arrow[d, "1"'] & t_{1,1} \arrow[d, dashed, "1"] \\
s_{2,0} \arrow[r, dashed, "\sigma_{2,0}"] \arrow[d, "0"'] & t_{2,0} \arrow[d, dashed, "0"] \\
\cdots & \cdots
\end{tikzcd}
\qquad
\begin{tikzcd}
s \arrow[r, "\sigma_1"] \arrow[d, "\sigma_2"'] & t \arrow[d, dashed, "\sigma_3"] \\
s' \arrow[r, dashed, "\sigma_4"'] & t'
\end{tikzcd}$$

**Fig. 1.** Reduction diagrams for Lemma 4.4

**Lemma 4.4.** *Let $s, t$ be terms satisfying the DVC with $s \xrightarrow{\sigma} t$ by an fvbv-renaming $\sigma$, and Red be a 0-1-labelled reduction for $s$ as follows: $s = s_{1,0} \to_0 s_{1,1} \to_1 s_{2,0} \to_0 s_{2,1} \ldots \to_0 s_{n,1} \to_1 s_{n+1,0}$. Then there is also a reduction for $t$ of the form $t = t_{1,0} \to_0 t_{1,1} \to_1 t_{2,0} \to_0 t_{2,1} \ldots \to_0 t_{n,1} \to_1 t_{n+1,0}$ with terms $t_{i,k}$, such that for all $i, k$: $s_{i,k} \xrightarrow{\sigma_{i,k}} t_{i.k}$ by fvbv-renamings $\sigma_{i,k}$, $\mathrm{fvp}(\sigma_{i,k})$ is a*

*restriction of* $\mathrm{fvp}(\sigma)$*, the terms* $t_{i,0}$ *satisfy the DVC, and* $s_{n+1,0}$ *is an answer iff* $t_{n+1,0}$ *is an answer (see left diagram in figure 1).*
*The lemma holds also for bv-renamings instead of fvbv-renamings. The lemma also holds, if the 0-1-labelled reduction of* $s$ *starts with a* $\rightarrow_1$*-reduction, in which case the reduction of* $t$ *also starts with a* $\rightarrow_1$*-reduction.*

*Proof.* Follows by induction on the length of reductions; from answer assumption 3.3, reduction assumption 3.4, and Lemma 2.5. □

**Proposition 4.5.** *Let* $s, s'$ *be terms with* $s' = \sigma(s)$*, where* $\sigma$ *is a fvbv-renaming. Then* $s\mathcal{M} \Leftrightarrow s'\mathcal{M}$ *for all* $\mathcal{M} \in \{\downarrow, \Downarrow, \uparrow, \Uparrow, ⟱, ⤊\}$*.*

*Proof.* Follows by easy arguments using Lemma 4.4, Lemma 2.5 and the assumptions on answer-terms 3.3. □

## 5 Context Lemmas

**Definition 5.1.** *For all terms* $s, t$ *and* $\mathcal{M} \in \{\downarrow, \Downarrow, ⟱\}$*:*
$s \leq_{\mathcal{M},R\nu} t$ *iff for all reduction contexts* $R$*, all vv-substitutions* $\nu$*, all vvbv-substitutions* $\gamma_s, \gamma_t$ *of* $s, t$ *derived from* $\nu$ *on* $\mathcal{FV}(s) \cup \mathcal{FV}(t)$*, we have* $R[\gamma_s(s)]\mathcal{M} \implies R[\gamma_t(t)]\mathcal{M}$*.*
*For strongly sharing calculi, if only the identical vv-substitution in the above relations is permitted, and also only the identical vvbv-substitution, we indicate this by omitting the* $\nu$ *in the notation, and denote the relations as* $\leq_{\mathcal{M},R}$*.*

**Lemma 5.2.** *Let* CALC *be strongly sharing and* $\mathcal{M} \in \{\downarrow, \Downarrow, ⟱\}$*. Then* $\leq_{\mathcal{M},R\nu} \;=\; \leq_{\mathcal{M},R}$*.*

*Proof.* This follows from Proposition 4.5, since for a reduction context $R$, $\sigma_s, \sigma_t$ (with adapted positions), are also bv-renamings of $R[s], R[t]$, respectively. □

For a finite set of variables $W$, a context $C$ is called *fresh for* $W$, iff for all variables $x \in W$, $x$ is not bound by a binder in $BP_{\overline{\text{hole}}}(C)$. We can slightly restrict the necessary reduction contexts for $\leq_{\mathcal{M},R}$, by using renamings and Proposition 4.5.
We separate the proofs for weakly and strongly sharing calculi, since there are differences in the renamings, and the weakly sharing part requires a lot of treatments of vvbv-substitutions.

**Lemma 5.3 (May-Convergence and Weakly Sharing).** *Let* CALC *be weakly sharing. Then* $\leq_{\downarrow,R\nu} \;=\; \leq_{\downarrow\nu}$*.*

*Proof.* We show the following generalized claim:
For all $n$, all multicontexts $M$ with $n$ holes, all vv-substitutions $\nu_i$ and derived vvbv-substitutions $\gamma_{s,i}, \gamma_{t,i}$ on $\mathcal{FV}(s_i) \cup \mathcal{FV}(t_i)$: If for terms $s_i, t_i, i = 1, \ldots, n$, and for all $i = 1, \ldots, n$: $s_i \leq_{\downarrow,R\nu} t_i$, then $M[\gamma_{s,1}(s_1), \ldots, \gamma_{s,n}(s_n)] \downarrow \implies M[\gamma_{t,1}(t_1), \ldots, \gamma_{t,n}(t_n)] \downarrow$. For convenience let $s'_i := \gamma_{s,i}(s_i), t'_i := \gamma_{t,i}(t_i)$. Note

that in the inductive proof below we will only use the weakened precondition $\gamma_{s,i}(s_i) \leq_{\downarrow,R\nu} \gamma_{t,i}(t_i)$. Proposition 4.5 permits us to assume, by applying bv-renamings, that the bound variables in $s'_i$ and $t'_i$ are distinct.
The claim is shown by induction on the length $l$ of 0-1-labelled-reductions of $M[s'_1,\ldots,s'_n]$ to an answer, and second on the number of holes of $M$.
As a base case, the claim is obviously true, if the number $n$ of holes is equal to 0, since then $M[s'_1,\ldots,s'_n] = M[t'_1,\ldots,t'_n]$.
There are two cases:

1. In $M[s'_1,\ldots,s'_n]$ some $s'_i$ is in a reduction position.
This means that at least one of the contexts $M_i = M[s'_1,\ldots,s'_{i-1},[],s'_{i+1},\ldots,s'_n]$ is a reduction context. Then Lemma 3.2 shows that there is some $j$, such that $M[s'_1,\ldots,s'_{j-1},[],s'_{j+1},\ldots,s'_n]$ as well as $M[t'_1,\ldots,t'_{j-1},[],t'_{j+1},\ldots,t'_n]$ is a reduction context. Using the induction hypothesis for the context $M' := M[[],\ldots,[],s'_j,[],\ldots,[]]$, which has $n-1$ holes, it follows that $M[s'_1,\ldots,s'_n]\downarrow \implies M[t'_1,\ldots,t'_{j-1},s'_j,t'_{j+1},\ldots,t'_n]\downarrow$.
Since $M[t'_1,\ldots,t'_{j-1},[],t'_{j+1},\ldots,t'_n]$ is a reduction context, the assumption and $M[t'_1,\ldots,t'_{j-1},s'_j,t'_{j+1},\ldots,t'_n]\downarrow$ imply that $M[t'_1,\ldots,t'_{j-1},t'_j,t'_{j+1},\ldots,t'_n]\downarrow$.

2. For all $i$: None of the contexts $M_i = M[s'_1,\ldots,s'_{i-1},[],s'_{i+1},\ldots,s'_n]$ is a reduction context. Lemma 3.2 implies that none of the holes of $M$ is at a reduction position. If $l = 0$, then $M[s'_1,\ldots,s'_n]$ is an answer-term, and by Assumption 3.3, the expression $M[t'_1,\ldots,t'_n]$ is also an answer term. Now assume that $l > 0$:

2a. First we consider the case that $M[s'_1,\ldots,s'_n]$ satisfies the DVC and that the reduction on $M[s'_1,\ldots,s'_n]$ is a $\rightarrow_0$-reduction. Let $M[s'_1,\ldots,s'_n] \rightarrow_0 s'$ be the start of the reduction of length $l$ to an answer. By the assumption 3.4 on reductions, there is a multicontext $M'$ with $n'$ holes, $\nu_i \in VV$ for $i \in \{1,\ldots,n'\}$, and a mapping $\pi : \{1,\ldots,n'\} \rightarrow \{1,\ldots,n\}$, such that $s' = M'[\nu_1(s'_{\pi(1)}),\ldots,\nu_{n'}(s'_{\pi(n')})]$.
The same holds by assumption 3.4 for $M[t'_1,\ldots,t'_n]$: There is a reduction $M[t'_1,\ldots,t'_n] \rightarrow_0 M'[\nu_1(t'_{\pi(1)}),\ldots,\nu_{n'}(t'_{\pi(n')})]$. Now we can apply the induction hypothesis, since the number of reductions to an answer of $s'$ is $l-1$, and the required preconditions hold: For all $R, \nu_R$ and if $\gamma_{Rs,i}, \gamma_{Rt,i}$ are derived from $\nu_R$ on $\mathcal{FV}(s'_i) \cup \mathcal{FV}(t'_i)$, then for all $i = 1,\ldots,n'$: $R[\gamma_{Rs,i}\nu_i(s'_{\pi(i)}])\downarrow \implies R[\gamma_{Rt,i}\nu_i(t'_{\pi(i)})]\downarrow$ holds, since $\gamma_{Rs,i}\nu_i$ and $\gamma_{Rt,i}\nu_i$ are also vvbv-substitutions derived from a common vv-substitution (see Lemma 2.5).

2b. The other case is that $M[s'_1,\ldots,s'_n]$ is the result of a $\rightarrow_0$ and the next reduction step in the 0-1-labelled reduction is a renaming. Then the reduction consists of applying some bv-renaming $M[s'_1,\ldots,s'_n] \xrightarrow{\sigma} M'[s''_1,\ldots,s''_n]$, such that $M'[s''_1,\ldots,s''_n]$ satisfies the DVC.
Let $\sigma_M$ be the part of the renaming for the binder positions that are in $M$. Let $W_i, i = 1,\ldots,n$ be the set of variables that may be potentially bound in hole $i$, and let $\rho_i, i = 1,\ldots,n$ be the mappings on $W_i$ induced by $\sigma$. Note that $\rho_i$ is injective on $W_i$. The effect of the bv-renaming $\sigma$ can be modelled as follows:
It is a bv-renaming $M \xrightarrow{\sigma_M} M'$, and fvbv-renamings $\mu_i$ with $s'_i \xrightarrow{\mu_i} s''_i$, where $\mu_i$

is derived from $\rho_i$. We construct an appropriate bv-renaming $\sigma'$ for $M[t'_1,\dots,t'_n]$ by using $\sigma_M$ again for $M$, and for every $i$ fvbv-renamings $\mu'_i$ for $t'_i$, where for all $i$: $\mu'_i$ is derived from $\rho_i$. For the bv-part of $\mu'_i$ fresh variables must be used, which ensures that $\sigma'(M[t'_1,\dots,t'_n])$ satisfies the DVC. By construction, we have $\sigma'(M[t'_1,\dots,t'_n]) = M'[\mu'_1(t'_1),\dots,\mu'_n(t'_n)]$.
It remains to show that the precondition $\mu_i(s'_i) \leq_{\downarrow,R\nu} \mu'_i(t'_i))$ holds for the all pairs $(\mu_i(s'_i),\mu'_i(t'_i))$: Let $i$ be fixed in the following, and let $R$ be a reduction context, let $\nu'$ be a vv-substitution, and $\gamma'_{s,i}$ be vvbv-substitutions of $\mu_i(s'_i)$ derived from $\nu'$ and $R[\gamma'_{s,i}\mu_i(s'_i)]\downarrow$. Then let $\gamma'_{t,i}$ be a vvbv-substitution of $\mu'_i(s'_i)$ derived from $\nu'$. Since $\gamma'_{s,i}\mu_i$ and $\gamma'_{t,i}\mu'_i$ are derived from the same vv-substitution $\nu\rho_{i|V(\rho_i(W_i))}$, we obtain by the assumption that $R[\gamma_{t,i}\mu_i(s'_i)]\downarrow$.
Since the preconditions are satisfied, the multicontext is the same $M'$ for $s_i$ and $t_i$, and the reduction length has been reduced, we can apply the induction hypothesis. □

**Lemma 5.4.** *Let $s,t$ be terms, $\mathcal{M} \in \{\downarrow,\Downarrow,\Downarrow\!\!\Downarrow\}$ and $W$ be a finite set of variables that contains all variables in $s,t$. Then $s \leq_{\mathcal{M},R} t$ holds, iff for all reduction contexts $R$ that are fresh for $W$, we have $R[s]\mathcal{M} \implies R[t]\mathcal{M}$.*

**Lemma 5.5 (May-Convergence and Strongly Sharing).** *Let* CALC *be strongly sharing. Then $\leq_{\downarrow,R} \;=\; \leq_{\downarrow}$.*

*Proof.* We show the following generalized claim:
For all $n$ and all multicontexts $M$ with $n$ holes: If for terms $s_i,t_i,i=1,\dots,n$, and for all $i=1,\dots,n$: $s_i \leq_{\downarrow,R} t_i$, then $M[s_1,\dots,s_n]\downarrow \implies M[t_1,\dots,t_n]\downarrow$.
Note that the induction and the cases are instances of the cases in the proof of lemma 5.3, where vv-substitutions can be omitted; only the final part on the preconditions is different: We present the proof that the precondition $\mu_i(s_i) \leq_{\downarrow,R} \mu'_i(t_i))$ holds for the all pairs $(\mu_i(s_i),\mu'_i(t_i))$:
Let $i$ be fixed in the following, and let $R$ be a reduction context with $R[\mu_i(s_i)]\downarrow$. We assume using Lemma 5.4 that the binders $BP_{\overline{\text{hole}}}(R)$ use fresh variables.
Let $\sigma_2$ be a fvbv-renaming of $R[\mu_i(s_i)]$, such that the mapping on $V_{\text{hole}}(R)$ is a restriction of $\rho_i^{-1}$, and $\sigma_2$ acts as the inverse of $\mu_i$ on $s_i$. Note that $\sigma_2$ may also act on free variables in $R[\mu_i(s_i)]$, since $R$ may have too few binders. The construction of $\sigma_2$ is possible due to the assumption on $R$. Then $\sigma_2(R[\mu_i(s_i)]) = \sigma_2(R)[s_i]$, and $\sigma_2(R)$ is a reduction context by the assumption 3.1 on UNWIND. Proposition 4.5 shows that $\sigma_2(R)[s_i]\downarrow$. The assumptions of the main claim of this proof now implies that $\sigma_2(R)[t_i]\downarrow$. Now starting with $R[\mu'_i(t_i)]$, let $\sigma_3$ be an fvbv-renaming of $R[\mu'_i(t_i)]$ such that the mapping on $V_{\text{hole}}(R)$ is a restriction of $\rho_i^{-1}$, and $\sigma_3$ acts as a reverse of $\mu'_i$ on $t_i$. There is no conflict within $t_i$, and also no conflict between binders of $R$ and variables in $\mu'_i(t_i)$, hence $\sigma_3$ is indeed an fvbv-renaming. Then $\sigma_3(R[\mu'_i(t_i)] = \sigma_3(R)[t_i] = \sigma_2(R)[t_i]$, and $\sigma_2(R)[t_i]\downarrow$ and Proposition 4.5 imply $R[\mu'_i(t_i)]\downarrow$. □

**Lemma 5.6 (Must-Convergence and Strongly Sharing).** *Let* CALC *be strongly sharing. Then $\leq_{\downarrow,R} \cap \leq_{\Downarrow,R} \;\subseteq\; \leq_{\Downarrow}$.*

*Proof.* We show the following generalized claim, using may-divergence.
For all $n$ and all multicontexts $M$ with $n$ holes: If for all for terms $s_i, t_i, i = 1, \ldots, n$, and for all $i = 1, \ldots, n$: $s_i \leq_{\downarrow,R} t_i \wedge s_i \leq_{\Downarrow,R} t_i$, then $M[t_1, \ldots, t_n] \uparrow \implies M[s_1, \ldots, s_n] \uparrow$.
The claim is shown by induction on the number $l$ of $\rightarrow_0$ and $\rightarrow_1$-reductions of 0-1-labeled reductions of $M[t_1, \ldots, t_n]$ to a must-divergent term, and second on the number of holes of $M$. The proof is almost a copy of the proof of the context lemma 5.5 for may-divergence; we give a sketch and emphasize the differences:
If some term $t_i$ is in a reduction position in $M[t_1, \ldots, t_n]$, then the arguments are the same as in proof of Lemma 5.5.
If no hole of $M[t_1, \ldots, t_n]$ is a reduction position, $l > 0$, and the reduction is a $\rightarrow_0$-reduction, then the same arguments as in the in proof of Lemma 5.5 show that we can use induction on $l$. The base case $l = 0$ is that $M[t_1, \ldots, t_n]$ is must-divergent. Now suppose that $M[s_1, \ldots, s_n]$ is not must-divergent. Then it is may-convergent, which by the assumption $\forall i : R[s_i] \downarrow \implies R[t_i] \downarrow$ and the context lemma 5.5 implies that $M[t_1, \ldots, t_n] \downarrow$, which is a contradiction. Hence $M[s_1, \ldots, s_n] \uparrow$, and the base case is proved. If no hole of $M[t_1, \ldots, t_n]$ is a reduction position, $l > 0$, and the reduction is a renaming $\rightarrow_1$, then the same arguments as in the proof of the may-context lemma 5.5 apply. □

An immediate consequence is:

**Corollary 5.7.** *Let* CALC *be strongly sharing. Then* $\leq_{\downarrow} \cap \leq_{\Downarrow,R} \;\subseteq\; \leq_{\Downarrow}$.

**Lemma 5.8 (Must-Convergence and Weakly Sharing).** *Let* CALC *be weakly sharing. Then* $\leq_{\downarrow,R\nu} \cap \leq_{\Downarrow,R\nu} \;\subseteq\; \leq_{\Downarrow\nu}$.

*Proof.* The proof can be done along the argumentation of the proof of Lemma 5.6, with analogous extensions as done in the proof of Lemma 5.3. □

**Lemma 5.9 (Total Must-Convergence and Strongly Sharing).** *Let* CALC *be a strongly sharing calculus. Then* $\leq_{\Ddownarrow,R} t \;=\; \leq_{\Ddownarrow}$.

*Proof.* We show the following generalized claim:
For all $n$ and all multicontexts $M$ with $n$ holes: If for all for terms $s_i, t_i, i = 1, \ldots, n$, and for all $i = 1, \ldots, n$: $s_i \leq_{\Ddownarrow,R} t_i$, then $M[s_1, \ldots, s_n] \Ddownarrow \implies M[t_1, \ldots, t_n] \Ddownarrow$.
Thus, let us assume that $M, s_i, t_i$ are given, that $M[s_1, \ldots, s_n] \Ddownarrow$, and that the claim holds for all terms that can be reached from $M[s_1, \ldots, s_n]$ by a $\rightarrow$-reduction. Proposition 4.5 permits us to assume, by applying bv-renamings, that the bound variables in $s_i$ and $t_i$ are distinct. The claim is shown by well-founded induction on the order $\xrightarrow{+}$ defined by the reduction $\rightarrow$ for all the descendents of $M[s_1, \ldots, s_n]$, and second on the number of holes of $M$. As a base case, the claim is obviously true, if $n = 0$.
There are several cases: We give a sketch for every case:

1. Some $s_i$ or $t_i$ is in a reduction position in $M[s_1, \ldots, s_n]$ or $M[t_1, \ldots, t_n]$, respectively. Then some hole of $M[., \ldots, .]$ is in a reduction context, and the arguments in case (1) of the proof of Lemma 5.5 apply.

2. None of the contexts $M_i = M[s_1, \ldots, s_{i-1}, [], s_{i+1}, \ldots, s_n]$ is a reduction context. Then no hole of $M$ is a reduction position. We have to show that all reduction sequences of $M[t_1, \ldots, t_n]$ terminate. If $M[t_1, \ldots, t_n]$ is an answer-term, then we are finished. If $M[t_1, \ldots, t_n]$ is irreducible, then by the same arguments as in in the proof of Lemma 5.5, $M[s_1, \ldots, s_n]$ is also irreducible, which implies that $M[s_1, \ldots, s_n]$ is an answer, and hence $M[t_1, \ldots, t_n]$ is an answer, too. If $M[t_1, \ldots, t_n]$ has a reduction, then using the reduction assumptions 3.4 for $\rightarrow_0$ and the same arguments as in Lemma 5.5, we can apply the induction hypothesis. □

The already demonstrated techniques suffice to prove:

**Lemma 5.10 (Total Must-Convergence for Weakly Sharing).** *Let* CALC *be weakly sharing. Then* $\leq_{\Downarrow\!\Downarrow,R\nu} \; = \; \leq_{\Downarrow\!\Downarrow\nu}$.

**Theorem 5.11 (Generic Context Lemma).** *If our assumptions hold for* CALC*, then:*

- *For strongly sharing calculi:* $\leq_{\downarrow,R} = \leq_{\downarrow}$, $\leq_{\Downarrow\!\Downarrow,R} = \leq_{\Downarrow\!\Downarrow}$, *and* $\leq_{\downarrow,R} \cap \leq_{\Downarrow,R} \subseteq \leq_{\Downarrow}$.
- *For weakly sharing calculi:* $\leq_{\downarrow,R\nu} = \leq_{\downarrow\nu} \subseteq \leq_{\downarrow}$, $\leq_{\Downarrow\!\Downarrow,R\nu} = \leq_{\Downarrow\!\Downarrow,\nu} \subseteq \leq_{\Downarrow\!\Downarrow}$, *and* $\leq_{\downarrow,R\nu} \cap \leq_{\Downarrow,R\nu} \subseteq \leq_{\Downarrow\nu} \subseteq \leq_{\Downarrow}$.

### 5.1 Strengthening the Context Lemma for Weakly Sharing Calculi

The context lemma for weakly sharing calculi has the slight disadvantage that in addition to all reduction contexts, also all vv-substitutions have to be checked. This can be avoided in most calculi by simulating $s[y_1/x_1, \ldots, y_n/x_n]$ by (`letrec` $x_1 = y_1, \ldots, x_n = y_n$ `in` $s$), which is usually of the form $R[s]$. Note, however, that the relation $\forall \ldots : s[y_1/x_1, \ldots, y_n/x_n] \sim$ (`letrec` $x_1 = y_1, \ldots, x_n = y_n$ `in` $s$), which is sufficient to drop all the $\nu$'s, may require an extra proof, depending on the calculus.

## 6 Examples of Calculi and Context Lemmas

Our context lemmas can be applied to higher-order lambda-calculi, even with `letrec`, with strict and non-strict reduction, provided there are no substituting rules, which is usually only possible, if a form of sharing is permitted by e.g. `let`, `letrec` or explicit substitutions. As a general guideline, note that the beta-rule in general violates our assumptions. The restricted beta-rule $(\lambda x.s)\ y \rightarrow s[y/x]$ may be allowed in weakly sharing calculi, provided $y$ is in a reduction or a variable-only position, The rules $(\lambda x.s)\ t \rightarrow$ (`let` $x = t$ `in` $s$), (`let` $x = v$ `in` $R[x]$) $\rightarrow$ (`let` $x = v$ `in` $R[v]$) are usually permitted in strongly sharing calculi, if the replaced position of $x$ is in a reduction position. Our result can be used for may-

as well as must-convergence in its two forms, with or without taking infinite reductions into account.
We mention several calculi, where the result is applicable:
The call-by-need-calculi in [AFM$^+$95,AF97,MOW98] are deterministic, use a `let` to represent sharing, and use a sharing variant of beta-reduction. All the assumptions are satisfied, where the answers according to our definition are of the form `let x1 = t1 in let x2 = t2 in ...  in` $\lambda$`x.s`. The context lemma for may-termination holds for these strongly sharing calculi.
The `letrec`-calculi in [AS98,SS06] are deterministic and provide `letrec` for expressing sharing. The context lemma for may-convergence holds for these strongly sharing calculi. The non-deterministic call-by-need calculi in [KSS98,Man05] provide a `let` and a non-deterministic choice. The assumptions are satisfied, where UNWIND is deterministic. Context lemmas for may-termination as well as must-termination for these strongly sharing calculi hold. Note that [KSS98] uses total must-divergence, and makes no use of a context lemma, whereas the calculus in [Man05] did not treat must-divergence.
The call-by-need calculus in [MSC99] with letrec, choice, case and constructors uses may- and total must-convergence, and satisfies our assumptions. The calculus is weakly sharing since the beta-rule-variant and the case-rule use vv-substitutions. Note that this calculus is an example with positions that may only be occupied by variables, namely the arguments in applications, which is permitted by our higher-order syntax. The context lemmas for may- and total must-convergence hold in this calculus.
The call-by-need calculi in [SSS07,Mor98] provide amb, letrec, case and constructors. They satisfy our criteria, where the first is strongly, and the second is weakly sharing. UNWIND and normal-order reduction are non-deterministic. Our results confirms the respective context lemmas, and also shows a new one for the call-by-need variant in [Mor98], since there is no proof of context lemma for total must-convergence in [Mor98].
The call-by-value concurrent process calculus in [NSSSS06] has a sharing variant of beta-reduction, and is derived from a calculus with beta-reduction [NSS06]. The sharing variant in [NSSSS06] has mutable cells, and a non-deterministic reduction. It satisfies our assumptions for a strongly sharing calculus. Note that UNWIND is nondeterministic. After some preprocessing is done, the context lemmas for may- and must can be derived from our results.
If the calculus permits substituting rules like plain beta-reduction, then Theorem 5.11 is not applicable, since then Assumption 3.4.(1) does not hold.

## 7 Conclusion

It is easy to check that the lemmas also hold for a simple type system, like in simply typed lambda calculus. To extend the results to other type systems is left for future research. A further topic for future research is to investigate bisimilarity for non-deterministic calculi with sharing also w.r.t. must-convergence.

## References

ACCL91. M. Abadi, L. Cardelli, P.-L. Curien, and J.-J Lévy. Explicit substitutions. *J. Funct. Programming*, 1(4):375–416, 1991.

AF97. Z.M. Ariola and M Felleisen. The call-by-need lambda calculus. *J. Funct. Programming*, 7(3):265–301, 1997.

AFM$^+$95. Z.M. Ariola, M. Felleisen, J. Maraist, M. Odersky, and P. Wadler. A call-by-need lambda calculus. In *Principles of Programming Languages*, pages 233–246, San Francisco, California, 1995. ACM Press.

AS98. Zena M. Ariola and Amr Sabry. Correctness of monadic state: An imperative call-by-need calculus. In *POPL 98*, pages 62–74, 1998.

Bar84. H.P. Barendregt. *The Lambda Calculus. Its Syntax and Semantics.* North-Holland, Amsterdam, New York, 1984.

CHS05. Arnaud Carayol, Daniel Hirschkoff, and Davide Sangiorgi. On the representation of McCarthy's amb in the pi-calculus. *Theoret. Comput. Sci.*, 330(3):439–473, 2005.

FH92. Matthias Felleisen and R. Hieb. The revised report on the syntactic theories of sequential control and state. *Theoret. Comput. Sci.*, 103:235–271, 1992.

FM01. Jonathan Ford and Ian A. Mason. Operational techniques in PVS - a preliminary evaluation. *Electron. Notes Theor. Comput. Sci.*, 42, 2001.

FM03. Jonathan Ford and Ian A. Mason. Formal foundations of operational semantics. *Higher-Order and Symbolic Computation*, 16(3):161–202, 2003.

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

How89. D. Howe. Equality in lazy computation systems. In *4th IEEE Symp. on Logic in Computer Science*, pages 198–203, 1989.

How96. D. Howe. Proving congruence of bisimulation in functional programming languages. *Inform. and Comput.*, 124(2):103–112, 1996.

JM97. T. Jim and A.R. Meyer. Full abstraction and the context lemma. *SIAM J. Comput.*, 25(3):663–696, 1997.

KSS98. Arne Kutzner and Manfred Schmidt-Schauß. A nondeterministic call-by-need lambda calculus. In *International Conference on Functional Programming 1998*, pages 324–335. ACM Press, 1998.

Man05. Matthias Mann. Congruence of bisimulation in a non-deterministic call-by-need lambda calculus. *Electron. Notes Theor. Comput. Sci.*, 128(1):81–101, 2005.

Mil77. R. Milner. Fully abstract models of typed $\lambda$-calculi. *Theoret. Comput. Sci.*, 4:1–22, 1977.

Mor68. J.H. Morris. *Lambda-Calculus Models of Programming Languages.* PhD thesis, MIT, 1968.

Mor98. A.K.D. Moran. *Call-by-name, call-by-need, and McCarthys Amb.* PhD thesis, Dept. of Comp. Science, Chalmers university, Sweden, 1998.

MOW98. John Maraist, Martin Odersky, and Philip Wadler. The call-by-need lambda calculus. *J. Funct. Programming*, 8:275–317, 1998.

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 *Lecture Notes in Computer Science*, pages 85–102. Springer-Verlag, 1999.

MSS06. Matthias Mann and Manfred Schmidt-Schauß. How to prove similarity a precongruence in non-deterministic call-by-need lambda calculi. Frank report 22, Inst. f. Informatik, J.W.Goethe-University, Frankfurt, January 2006.

NSS06. Joachim Niehren, Jan Schwinghammer, and Gert Smolka. A concurrent lambda calculus with futures. *Theoret. Comput. Sci.*, 364(3):338–356, November 2006.

NSSSS06. Joachim Niehren, David Sabel, Manfred Schmidt-Schauß, and Jan Schwinghammer. Program equivalence for a concurrent lambda calculus with futures. Frank report 26, Inst. f. Informatik, J.W.Goethe-University, Frankfurt, October 2006.

Ong93. C.-H. L. Ong. Non-determinism in a functional setting. In *Proc. 8th IEEE Symposium on Logic in Computer Science (LICS '93)*, pages 275–286. IEEE Computer Society Press, 1993.

Plo76. G.D. Plotkin. A powerdomain construction. *SIAM Journal of Computing*, 5(3):452–487, 1976.

SS03. Manfred Schmidt-Schauß. FUNDIO: A lambda-calculus with a `letrec`, `case`, constructors, and an IO-interface: Approaching a theory of `unsafePerformIO`. Frank report 16, Inst. f. Informatik, J.W.Goethe-University, Frankfurt, 2003.

SS06. Manfred Schmidt-Schauß. Equivalence of call-by-name and call-by-need for lambda-calculi with letrec. Frank report 25, Inst. f. Informatik, J.W.Goethe-University, Frankfurt, September 2006. submitted for publication.

SSS07. David Sabel and Manfred Schmidt-Schauß. A call-by-need lambda-calculus with locally bottom-avoiding choice: Context lemma and correctness of transformations. *Math. Structures Comput. Sci.*, 2007. accepted for publication.

SSSS05. Manfred Schmidt-Schauß, Marko Schütz, and David Sabel. A complete proof of the safety of Nöcker's strictness analysis. Frank report 20, Inst. f. Informatik, J.W.Goethe-University, Frankfurt, 2005. submitted for publication.