Contextual Equivalence for the Pi-Calculus that can Stop

David Sabel and Manfred Schmidt-Schauß

Goethe-University, Frankfurt, Germany

Technical Report Frank-53

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

March 10, 2014

Abstract. The pi-calculus is a well-analyzed model for mobile processes and mobile computations. While a lot of other process and lambda calculi that are core languages of higher-order concurrent and/or functional programming languages use a contextual semantics observing the termination behavior of programs in all program contexts, traditional program equivalences in the pi-calculus are bisimulations and barbed testing equivalences, which observe the communication capabilities of processes under reduction and in contexts.
There is a distance between these two approaches to program equivalence which makes it hard to compare the pi-calculus with other languages. In this paper we contribute to bridging this gap by investigating a contextual semantics of the synchronous pi-calculus with replication and without sums.
To transfer contextual equivalence to the pi-calculus we add a process Stop as constant which indicates success and is used as the base to define and analyze the contextual equivalence which observes may- and should-convergence of processes.
We show as a main result that contextual equivalence in the pi-calculus with Stop conservatively extends barbed testing equivalence in the (Stop-free) pi-calculus. This implies that results on contextual equivalence can be directly transferred to the (Stop-free) pi-calculus with barbed testing equivalence.
We analyze the contextual ordering, prove some nontrivial process equivalences, and provide proof tools for showing contextual equivalences. Among them are a context lemma, and new notions of sound applicative similarities for may- and should-convergence.

1 Introduction

Motivation

The π-calculus [MPW92,Mil99,SW01] is a well-known model for concurrent processes with shared-message passing. It has a minimalistic syntax including parallel process-composition, named channels, and input/output-capabilities on the channels. In contrast to function application (as in the lambda-calculus) the data flow in the π-calculus is programmed by communication between processes.

The π-calculus is a very basic model for concurrent processes, and it has become important not least because it is used and extended for a lot of applications. Examples are the Spi-calculus [AG97] for reasoning about cryptographic protocols the stochastic π-calculus [Pri95] with applications in molecular biology, and the join calculus [Lan96,FG02] for distributed computing.

Our long standing research goal includes questions of expressivity of program calculi. Hence we are investigating the problem whether the π-calculus can be encoded into other concurrent program calculi (which are mainly extended concurrent lambda-calculi, for instance the call-by-value lambda calculus with concurrent futures [NSS06,NSSSS07], or the CHF-calculus modelling

Concurrent Haskell [SSS11,SSS12]), and also vice versa whether other program calculi can be encoded into the π-calculus. Given such an encoding, the important properties are whether it reflects and/or preserves program equivalences from source to the target calculus, i.e. whether the encoding is adequate, full-abstract, etc.

However, before inventing such encodings, a notion of program equivalence for the source and the target calculus is required.

Contextual Equivalence

For program calculi based on the lambda-calculus the usual approach to program equivalence is Morris style-contextual equivalence [Mor68,Plo75] which can be used in a uniform way for a lot of those calculi. For deterministic languages, contextual equivalence is based on the notion of a terminated (or successful) program, and it equates programs, if the ability to terminate (i.e. so-called *may-convergence*) is indistinguishable when exchanging one program by the other in any surrounding program context. For non-deterministic languages this equivalence is too coarse, but it can be strengthened by additionally observing whether the program successfully terminates on all execution paths, i.e. whether the program *must-converges*, or as a slightly different approach, whether the program *should-converges*, which holds, if the the property of being *may-convergent* holds on all execution paths. In difference to must-convergence, should-convergence (see e.g. [NC95,CHS05,RV07,SSS10]) has some kind of fairness built-in: the predicate does not change even if instead of all reduction sequences only *fair* ones are taken into account, where fair reduction sequences ensure that every reducible expression is reduced after finitely many reduction steps.

The advantages of contextual equivalence are many-fold. Contextual equivalence is very natural, since it identifies programs as equal if exchanging programs is not observable; it is a congruence on programs which is an inevitable requirement for any program equivalence; and contextual equivalence is a uniform concept, since it can be defined for any program calculus which is equipped with a small-step reduction relation and a notion of successfulness for programs.

Goal of this Paper

For the π-calculus, a lot of process equivalences exists (which will be discussed later), but none of them is a contextual equivalence in the above sense, since it does not use a notion of successful termination of processes. It is hard to find such a (at least commonly accepted) notion which stems from the fact that the π-calculus models reactive systems which naturally may run forever.

However, our goal is to define and analyze a Morris' style contextual equivalence for π-processes, which will then support research on the expressivity of program calculi w.r.t. the π-calculus. Since no notion of success exists in the π-calculus we add such a notion by introducing a constant Stop as a *syntactic construct*. The constant indicates termination of a process, i.e. whenever Stop is a parallel component of the overall program, then the program is successful – independent of other components. This extended calculus is called Π_{Stop}.

Of course, inventing a new program equivalence in the well-analyzed π-calculus has to be justified. Besides the already mentioned advantages of a contextual equivalence the following arguments support our investigation:

- In concurrent programming languages, like e.g. Haskell, the termination of the main thread leads to termination of the whole program, i.e. all concurrent threads are also terminated. The π-calculus has no process hierarchy, but the primitive Stop models a similar behavior of enforcing global termination, since the occurrence of Stop as process component in some process enforces success of the whole program.

- Concerning encodability results one may also argue that Π_{Stop} is no longer the π-calculus and thus those results will be exotic. However, as we show, this is not the case, since there is a very strong connection between the π-calculus without Stop and the Π_{Stop}: for Stop-free processes contextual equivalence in Π_{Stop} coincides with barbed may- and should testing equivalence (see below) in the π-calculus without Stop. Thus correctness results of encodings between Π_{Stop} and other program calculi, can easily be transferred to the π-calculus with Stop w.r.t. barbed testing equivalence.
- Since barbed testing equivalence and contextual equivalence coincide for Stop-free processes, contextual equivalence and our developed proof techniques can be used to reason about barbed testing equivalence of processes.

Process Equivalences in the π-Calculus

For the π-calculus a lot of (different) process equivalences exist. We discuss and compare them to our approach.

Bisimulations Several different bisimulations were suggested as process equivalence (see e.g. [SW01,MS92]). They equate processes if testing the processes (using reduction) exhibits that they have the same input and output capabilities and that they reach equivalent states. Bisimulations occur in *strong* variants, where bisimilar processes must have an identical reduction behavior for every single reduction step, and there are *weak* bisimulations, where the number of internal reduction need not coincide, but equivalent states w.r.t. the reflexive-transitive closure of reduction must be reached. From our point of view, even weak bisimulations are too fine grained to be used as the notion of process equivalence. However, since proving processes bisimilar is often easy, bisimulations can be used as proof tools to establish process equivalences.

Testing Equivalences An approach which is close to contextual equivalence are testing equivalences (see e.g. [DH84] for CCS, [BD95,FG05] for the π-calculus, and [Lan96] for the join-calculus). They are defined analogously to contextual equivalence, but instead of observing successful termination, other observations are relevant.

For *barbed testing equivalences* the capability of receiving (or emitting) a name on an (open) input (or output) channel is observed (i.e. the process has an input or output *barb*). There are several variants of this definition, i.e. whether only the possible output, only the possible input, or both are observed, and also whether the quantification is over all channel names, or one single channel name, and sometimes also ignoring the concrete name of the channel.

Barbed *may-testing* only tests whether a process may have a barb after reducing it. However, barbed may-testing is not sufficiently discriminating, since obviously different processes are equated, e.g. a process P that can receive input on channel x and a process Q that nondeterministically either diverges without communication capabilities or behaves like P are barbed may-testing equivalent. Consequently, there is some work where may-testing is combined by using a must-testing predicate or a should-testing predicate (which is sometimes called fair must-testing). Roughly speaking, these predicates require an input or output capability on every execution path. A detailed analysis of barbed may and should-testing for the asynchronous π-calculus is in [FG05]. Established results are that barbed may-testing can be omitted, since barbed should-testing already includes may-testing and several soundness and completeness results for several bisimulations and barbed congruence are presented.

Compared to contextual equivalence, the property of having a barb is different from a predicate which indicates success, since it is not stable under reduction, i.e. a process may have a barb and reduce to a process which no longer has a barb. This phenomenon was also observed by [FG05], and as a consequence they also considered a variant of the asynchronous π-calculus (the so-called local π-calculus) where barbs are stable under reduction, by forbidding reductions on open channel names. However, this approach changes the semantics of the calculus.

Another difference between the results in [FG05] and our approach is that we consider the more expressive synchronous π-calculus instead of the asynchronous variant.

Another observation was chosen in [BD95] for a testing equivalence: they added a special action ω to the π-calculus, and their testing equivalence observe whether the action ω may or must occur. This approach looks very similar to our approach of contextual equivalence, since instead of adding a constant Stop we could also add Stop as a new action prefix and adapt the successfulness notion accordingly. This would not change our results, since both extensions of the π-calculus are isomorphic. However, the approach in [BD95] is different. One difference is that the small-step reduction in [BD95] removes the ω-action by executing it, and thus again successfulness is not stable under reduction. Another more substantial difference is that the ω-action only occurs in the contexts for testing, but not in the language itself, and thus the tests are more discriminating than the language itself, which violates the condition of contextual equivalence that programs are identified as equal if their behavior cannot be distinguished if they are used as subprograms in any other program. In [BD95] the surrounding context is no longer a program. A further substantial difference is that the testing preorders in [BD95] are not precongruences, which is problematic, since replacing subprograms by testing equivalent subprograms is not ensured to be semantically correct. Finally, the considered variant of the π-calculus does not include replication or recursion, but a constant for nontermination, and thus it is a restricted variant of the π-calculus.

Results

Adding a new syntactic construct to the π-calculus may change the semantics of the calculus and thus requires an analysis of the contextual equivalence. We first show the coincidence of barbed testing equivalence with contextual equivalence for Stop-free process. Thus we first introduce the π-calculus and results on barbed testing and then extend the calculus by the constant Stop. Thereafter we develop proof tools for showing program equivalences. We show that a context lemma holds, which restricts the required class of contexts to show contextual equivalence. And we introduce notions of applicative similarities for the may- and the should-convergence. Even though there are soundness results on bisimilarities and may- and should-testing for the asynchronous π-calculus in [FG05], to the best of our knowledge our notion of an applicative similarity for should-convergence is new. We prove soundness of these similarities w.r.t. the contextual preorders and thus they can be used for coinductive proofs to show contextual equivalence. Even though the test for may-convergence is subsumed by testing should-convergence, our reasoning tools require also reasoning about may-convergence, and thus we will consider both predicates. Equipped with these tools we show that process interaction is correct, if it is deterministic, prove some further process equations, and investigate the contextual ordering. We show that a contextually least element does not exist in Π_{Stop}, but a largest element exists – the constant Stop.

Outline

In Section 2 we briefly recall (a variant of) the synchronous π-calculus (called Π) and transfer the results about barbed testing equivalence to this calculus. In Section 3 we extend Π by the constant Stop resulting in the calculus Π_{Stop}. We introduce contextual equivalence of Π_{Stop} and prove its connection with barbed testing equivalence in Π. The context lemma and soundness of an applicative similarity is proven in Section 4. We analyze the contextual ordering and prove correctness of deterministic process interaction in Section 5. Finally, we conclude in Section 6. For readability some proofs are given in the appendix.

2 The π-Calculus Π

In this section we briefly recall the synchronous π-calculus, called Π, with replication but without sums, recursion, and matching operators. We will then give an overview of several variants of may and should testing preorders and equivalences for π processes and recall the connections between them. Thereafter we will introduce and discuss our approach of adding the constant Stop to indicate successfulness and the corresponding contextual equivalence.

2.1 Syntax and Operational Semantics

We consider the variant of the synchronous π-calculus with replication. For simplicity we exclude sums, however internal choices can be encoded. We also do not include operators for name matching, since they are not available in usual higher-order programming languages.

Definition 2.1 (Syntax of Processes). *Let $\mathcal{N}$ be a countably infinite set of* names*. Processes Proc and* action prefixes *π are defined by the following grammar, where $x, y \in \mathcal{N}$:*

$$P, Q, R \in Proc ::= \pi.P \mid P_1 \mid P_2 \mid !\,P \mid \mathbf{0} \mid \nu x.P$$
$$\pi ::= x(y) \mid \overline{x}y$$

The prefix $x(y)$ is called an input-prefix*, and $\overline{x}y$ is called an* output-prefix*,*

The two syntactic constructs which introduce a *scope* for a name are the ν-binder in $\nu x.P$ which restricts the scope of name x to P, and in $x(y).P$ the name y is bound with scope P. This induces a notion of α-renaming and α-equivalence $=_\alpha$ as usual. We will use $\mathsf{fn}(P)$ for the set of *free names* of process P and $\mathsf{bn}(P)$ for the set of *bound names* of process P. We assume the distinct name convention, i.e. free names are distinct from bound names and bound names are pairwise distinct, which can always be achieved by α-renaming bound variables.

We give a brief overview of the syntactic constructs. A process $x(y).P$ has the capability to *receive* some name z along the channel named x and then behaves like $P[z/y]$ where $[z/y]$ is the capture free substitution of name y by name z. A process $\overline{x}y.P$ has the capability to *send* a name y along the channel named x. After sending the name the resulting process behaves like process P. $\mathbf{0}$ is the *silent process* which has no capabilities to communicate.

$P_1 \mid P_2$ is the *parallel composition* of processes P_1 and P_2. $\nu z.P$ *restricts* the scope of the name z to process P. $!\,P$ is *replication* of process P, i.e. it can be interpreted as an "abbreviation" for the (infinitely large) process $P \mid P \mid \ldots \mid P$.

To ease reading, we will use the following abbreviations: The processes $x(y).\mathbf{0}$ and $\overline{x}y.\mathbf{0}$ are abbreviated as $x(y)$, and $\overline{x}y$, resp. Instead of $\nu x_1.\nu x_2.\ldots.\nu x_n.P$ we will also write $\nu x_1, \ldots, x_n.P$. We write $\nu\mathcal{X}.P$ as an abbreviation for $\nu x_1, \ldots, x_n.P$ if the concrete names $x_1, \ldots, x_n$ and the number n are not of interest. This also includes the case that $\nu\mathcal{X}.P$ abbreviates P, i.e. if $\mathcal{X}$ is empty. We also use set-notation for $\mathcal{X}$ and e.g. write $x_i \in \mathcal{X}$ with its obvious meaning.

We will also use name substitutions $\sigma : \mathcal{N} \to \mathcal{N}$. With Σ we denote the set of all name substitutions.

We will use the following abbreviation for internal choice of two processes: $\mathtt{choice}(P, Q)$ abbreviates the process $\nu x, y.(x(y).P \mid x(y).Q \mid \overline{x}y)$.

In the remainder of the paper, we use several binary relations on processes. Given a relation $\mathcal{R} \subseteq (Proc \times Proc)$, we write $\mathcal{R}^{-1}$ for the relation $\{(Q, P) \mid (P, Q) \in \mathcal{R}\}$ and $\mathcal{R}^\sigma$ is defined as: $(P, Q) \in \mathcal{R}^\sigma$ iff $(\sigma(P), \sigma(Q)) \in \mathcal{R}$ for all $\sigma \in \Sigma$.

Definition 2.2 (Process Contexts). *A* process context *$C \in \mathcal{C}$ is a process with a* hole *(denoted with $[\cdot]$) at process position, i.e. they are generated by the grammar:*

$$C \in \mathcal{C} ::= [\cdot] \mid \pi.C \mid C \mid P \mid P \mid C \mid !\,C \mid \nu x.C \text{ where } x \in \mathcal{N}.$$

The construct $C[P]$ denotes the process where the hole of C is replaced by process P.

We define structural congruence of processes:

Definition 2.3 (Structural Congruence). Structural congruence $\equiv$ *is the smallest congruence on processes satisfying the axioms:*

$$\begin{aligned} P &\equiv Q, \text{ if } P =_\alpha Q \\ P_1 \mid (P_2 \mid P_3) &\equiv (P_1 \mid P_2) \mid P_3 \\ \nu z.(P_1 \mid P_2) &\equiv P_1 \mid \nu z.P_2, \text{ if } z \notin \mathsf{fn}(P_1) \\ P \mid \mathbf{0} &\equiv P \\ P \mid Q &\equiv Q \mid P \\ \nu z.\mathbf{0} &\equiv \mathbf{0} \\ \nu z.\nu w.P &\equiv \nu w.\nu z.P \\ !\, P &\equiv P \mid !\, P \end{aligned}$$

Definition 2.4 (Standard Reduction). *The reduction rule* $\xrightarrow{ia}$ *performing* interaction *between processes is defined as*

$$x(y).P \mid \overline{x}v.Q \xrightarrow{ia} P[v/y] \mid Q.$$

Reduction contexts $\mathcal{D}$ *are defined by the grammar:*

$$D \in \mathcal{D} ::= [\cdot] \mid D \mid P \mid P \mid D \mid \nu x.D$$

A standard reduction $\xrightarrow{sr}$ *consists of applying an* $\xrightarrow{ia}$*-reduction in a reduction context (modulo structural congruence):*

$$\frac{P \equiv D[P'], P' \xrightarrow{ia} Q', D[Q'] \equiv Q, \text{ and } D \in \mathcal{D}}{P \xrightarrow{sr} Q}$$

We define $\xrightarrow{sr,*} := \bigcup_{i\geq 0} \xrightarrow{sr,i}$ *and* $\xrightarrow{sr,+} := \bigcup_{i>0} \xrightarrow{sr,i}$ *where for* $P, Q \in Proc$*:* $P \xrightarrow{sr,0} P$ *and* $P \xrightarrow{sr,i} Q$ *if there exists* $P' \in Proc$ *s.t.* $P \xrightarrow{sr} P'$ *and* $P' \xrightarrow{sr,i-1} Q$*.*

2.2 Barbed May- and Should-Testing

We now define the notion that a process has a barb on a channel.

Definition 2.5 (Barb). *A process* P has a barb on input x *(written as* $P \Big\Downarrow^{x}$*) iff* P *can receive a name on channel* x*, i.e.* $P \equiv \nu\mathcal{X}.(x(y).P' \mid P'')$ *where* $x \notin \mathcal{X}$*, and* P has a barb on output x *(written as* $P \Big\Downarrow^{\overline{x}}$*) iff* P *can emit a name on channel* x*, i.e.* $P \equiv \nu\mathcal{X}.(\overline{x}y.P' \mid P'')$ *where* $x \notin \mathcal{X}$*.*

As observations in program equivalences we will use two behaviors w.r.t. barbs. On the one hand whether a process may reduce to a process that has a barb, and on the other hand whether the evaluation of processes has the ability to have a barb on every reduction path:

Definition 2.6 (May-barb and Should-barb). *For* $\mu \in \{x, \overline{x}\}$*,* P may have a barb on x *(written as* $P \downarrow_\mu$*) iff* $P \xrightarrow{sr,*} Q \wedge Q \Big\Downarrow^{\mu}$*, and* P should have a barb on x *(written as* $P \Downarrow_\mu$*) iff* $P \xrightarrow{sr,*} P' \implies P' \downarrow_\mu$*. We also write* $P \uparrow_\mu$ *iff* $P \Downarrow_\mu$ *does not hold, and* $P \Uparrow_\mu$ *if* $P \downarrow_\mu$ *does not hold.*

As further notations we write $P \downarrow$ *(*$P \Downarrow$*, resp.) if there exists a name* x *such that* $P \downarrow_x$ *(*$P \Downarrow_x$*, resp.). With* $P \Uparrow$ *and* $P \uparrow$ *we denote the logical negations of* $P \downarrow$ *and* $P \Downarrow$*.*

Note that in the existential predicates ($\downarrow$ and $\Downarrow$) only barbs on input channels are taken into account. Note also that $P \uparrow$ ($P \uparrow_\mu$, resp.) means that P can reduce to a process that has no input capabilities (no input or output capabilities on the channel μ, resp.), i.e. $P \uparrow$ holds iff $P \xrightarrow{sr,*} P'$ and $P' \Uparrow$.

Definition 2.7 (Barbed May- and Should-Preorders). *For $\mu \in \{x, \overline{x}\}$, and $\xi \in \{\downarrow_\mu, \Downarrow_\mu, \uparrow_\mu, \Uparrow_\mu, \downarrow, \Downarrow, \uparrow, \Uparrow\}$*

- *$\sqsubseteq_\xi$ denotes the preservation of observation ξ, i.e.*

$$P \sqsubseteq_\xi Q \text{ iff } P\xi \implies Q\xi, \text{ and}$$

- *$\sqsubseteq_{c,\xi}$ denotes the closure of $\sqsubseteq_\xi$ w.r.t. contexts, i.e.*

$$P \sqsubseteq_{c,\xi} Q \text{ iff } \forall C \in \mathcal{C} : C[P] \sqsubseteq_\xi C[Q], \text{ and}$$

- *let $\sqsupseteq_\xi := \sqsubseteq_\xi \cap (\sqsubseteq_\xi)^{-1}$ and $\sqsupseteq_{c,\xi} := \sqsubseteq_{c,\xi} \cap (\sqsubseteq_{c,\xi})^{-1}$.*

Definition 2.8 (Barbed May- and Should-Testing). *Processes P and Q are* barbed testing equivalent *(written $P \sqsupseteq_c Q$) iff $P \sqsubseteq_c Q \wedge Q \sqsubseteq_c P$, where*

$$\begin{aligned}
&\sqsubseteq_c := \sqsubseteq_{c,\text{may}} \cap \sqsubseteq_{c,\text{should}} \\
&P \sqsubseteq_{c,\text{may}} Q \quad \text{iff } \forall x \in \mathcal{N},\ \mu \in \{x, \overline{x}\}\text{: } P \sqsubseteq_{c,\downarrow_\mu} Q \\
&P \sqsubseteq_{c,\text{should}} Q \text{ iff } \forall x \in \mathcal{N},\ \mu \in \{x, \overline{x}\}\text{: } P \sqsubseteq_{c,\Downarrow_\mu} Q.
\end{aligned}$$

We also write $\sqsupseteq_{c,\text{should}}$ for $\sqsubseteq_{c,\text{should}} \cap (\sqsubseteq_{c,\text{should}})^{-1}$ and $\sqsupseteq_{c,\text{may}}$ for $\sqsubseteq_{c,\text{may}} \cap (\sqsubseteq_{c,\text{may}})^{-1}$.

Some easy consequences of the definitions are:

Lemma 2.9.
- *For $\mu \in \{x, \overline{x}\}$, and $\xi \in \{\downarrow_\mu, \Downarrow_\mu, \uparrow_\mu, \Uparrow_\mu, \downarrow, \Downarrow, \uparrow, \Uparrow\}$ the relation $\sqsubseteq_\xi$ is a preorder and the relation $\sqsubseteq_{c,\xi}$ is a precongruence.*
- *The relations $\sqsubseteq_c$, $\sqsubseteq_{c,\text{may}}$, and $\sqsubseteq_{c,\text{should}}$ are precongruences and $\sqsupseteq_c, \sqsupseteq_{c,\text{may}}, \sqsupseteq_{c,\text{should}}$ are congruences.*
- *For $(\xi_1, \xi_2) \in \{(\downarrow_\mu, \Uparrow_\mu), (\Downarrow_\mu, \uparrow_\mu), (\downarrow, \Uparrow), (\Downarrow, \uparrow)\}$ and $\mu \in \{x, \overline{x}\}$, both $\sqsubseteq_{\xi_1} = (\sqsubseteq_{\xi_2})^{-1}$ and $\sqsubseteq_{c,\xi_1} = (\sqsubseteq_{c,\xi_2})^{-1}$ holds.*

The definition of barbed testing equivalence is given in a general form (or say most discriminating), since the may- and should-behavior, all channel names, the input and output barbs, and also all channels are separately considered. However, in the π-calculus the definition is equivalent to simpler definitions, which also simplify corresponding proofs a little bit. I.e. it is sufficient to observe the should-behavior only, and to observe input (or output) channels exclusively, and to either observe a single channel name, or existentially observing the barb capabilities. The corresponding soundness and completeness proofs are mostly standard, e.g. for the asynchronous π-calculus most of the proofs can be found in [FG05]. So we will only state the results, but for the sake of completeness, the proofs can be found in the appendix.

Theorem 2.10 (Characterizations of Barbed Testing).

- $\sqsubseteq_{c,\text{may}} = \sqsubseteq_{c,\downarrow_x} = \sqsubseteq_{c,\downarrow}$
- $\sqsubseteq_{c,\text{should}} = \sqsubseteq_{c,\Downarrow_x} = \sqsubseteq_{c,\Downarrow}$
- $\sqsubseteq_{c,\text{should}} \subset (\sqsubseteq_{c,\text{may}})^{-1}$ *and thus*
  $\sqsupseteq_c = \sqsupseteq_{c,\text{should}} = \sqsupseteq_{c,\Downarrow_x} = \sqsupseteq_{c,\Downarrow}$.

3 The Pi-calculus with Stop

Choosing the barb behavior as an observation has the drawback that it not really relates to a notion of successful termination, since a process that barbs on a channel x may reduce to a process that does no longer barb on x, e.g. the process $x(z) \mid \overline{x}y$.

Hence we introduce the calculus Π_{Stop} which extends Π by a constant Stop which indicates successful termination.

3.1 The Calculus Π_{Stop} and Contextual Equivalence

We define Π_{Stop} by describing the extensions w.r.t. Π defined in the previous section:

Definition 3.1 (Calculus Π_{Stop}). *Compared to Π the syntax of the calculus Π_{Stop} additionally includes the constant* Stop *which may occur on any process position. We denote the set of all Π_{Stop}-processes by $Proc_{\mathsf{Stop}}$.*

Process contexts $\mathcal{C}_{\mathsf{Stop}}$ extend the contexts $\mathcal{C}$ such that they may also include the constant Stop*. The structural congruence $\equiv$ is extended by the axiom $\nu x.\mathsf{Stop} \equiv \mathsf{Stop}$. The reduction contexts and the standard reduction $\xrightarrow{sr}$ are unchanged except that* Stop *may occur as process component.*

Successfulness of a Π_{Stop}-process means that the constant Stop occurs on the top-level of the process:

Definition 3.2 (Successful Process). *A process P is* successful *if $P \equiv \mathsf{Stop} \mid P'$ for some process P'.*

Clearly, for any successful process P any contractum P' is also successful. Thus successfulness can be used as the usual termination test in contextual equivalence, defined in a straight-forward way (analogous to barbed testing equivalence, but by observing success instead of observing barbs):

Definition 3.3 (Convergencies and Contextual Equivalence). *A process P is* may-convergent *(denoted by $P{\downarrow}$) iff there exists a successful process P' such that $P \xrightarrow{sr,*} P'$. A process P is* should-convergent *(denoted by $P{\Downarrow}$) iff for all processes P': $P \xrightarrow{sr,*} P' \implies P'{\downarrow}$. If a process P is not may-convergent, then we say P is* must-divergent *and denote it as $P{\Uparrow}$. We say a process P is* may-divergent *iff $P{\Downarrow}$ does not hold and denote it as $P{\uparrow}$.*

For $\xi \in \{\downarrow, \Downarrow, \uparrow, \Uparrow\}$ the relations $\leq_\xi$ are defined as

$$P \leq_\xi Q \text{ iff } P\xi \implies Q\xi$$

and $\leq_{c,\xi}$ denote their contextual closures, i.e.

$$P \leq_{c,\xi} Q \text{ iff } \forall C \in \mathcal{C}_{\mathsf{Stop}} : C[P] \leq_\xi C[Q].$$

With $\sim_{c,\xi}$ we denote the intersection of $\leq_{c,\xi}$ and $(\leq_{c,\xi})^{-1}$.

Contextual preorder $\leq_c$ is the intersection $\leq_{c,\downarrow} \cap \leq_{c,\Downarrow}$ and contextual equivalence $\sim_c$ is defined as

$$\sim_c := \leq_c \cap (\leq_c)^{-1}.$$

Since $P{\uparrow}$ iff there exists a process P' such that $P \xrightarrow{sr,*} P'$ and $P'{\Uparrow}$, the following properties are straight-forward:

Lemma 3.4.
- $P \leq_\uparrow Q \iff Q \leq_\Downarrow P$ *and* $P \leq_\Uparrow Q \iff Q \leq_\downarrow P$.
- $P \leq_{c,\uparrow} Q \iff Q \leq_{c,\Downarrow} P$ *and* $P \leq_{c,\Uparrow} Q \iff Q \leq_{c,\downarrow} P$.
- $\leq_\downarrow, \leq_\Downarrow, \leq_\uparrow, \leq_\Uparrow$ *are preorders,* $\leq_c, \leq_{c,\downarrow}, \leq_{c,\Downarrow}, \leq_{c,\uparrow}, \leq_{c,\Uparrow}$ *are precongruences, and* $\sim_c, \sim_{c,\downarrow}, \sim_{c,\Downarrow}, \sim_{c,\uparrow}, \sim_{c,\Uparrow}$ *are congruences.*

Since reduction includes transforming processes using structural congruence, structural congruent processes are contextually equivalent:

Proposition 3.5. *If $P \equiv Q$, then for $\xi \in \{\downarrow, \Downarrow, \uparrow, \Uparrow\}$: $P \leq_\xi Q$, $P \leq_{c,\xi} Q$, $Q \leq_\xi P$, $Q \leq_{c,\xi} P$ and thus in particular $P \sim_c Q$.*

Some more easy properties are:

Lemma 3.6. *For all processes P, Q it holds:*

1. *If $P \xrightarrow{sr} Q$ then $\nu x.P \xrightarrow{sr} \nu x.Q$*
2. *If $\nu x.P \xrightarrow{sr} Q$ then $P \xrightarrow{sr} Q'$ such that either $Q \equiv \nu x.Q'$ or $Q \equiv Q'$*
3. *For $\xi \in \{\downarrow, \Downarrow, \uparrow, \Uparrow\}$: $P \leq_\xi \nu x.P$ and $\nu x.P \leq_\xi P$.*

For completeness the proof of these properties can be found in the appendix (Lemma B.1).

3.2 Relating Contextual Equivalence to Barbed Testing Equivalence

We show that the new constant Stop and the related test of successfulness coincides with barbed testing equivalence on Stop-free processes, i.e. contextual equivalence of Π_{Stop} conservatively extends barbed testing equivalence of the π-calculus: $P \sqsupseteq_c Q \implies P \sim_c Q$. Moreover, on stop-free processes contextual equivalence is not only sound for barbed testing, it is also complete, i.e. $P \sim_c Q \implies P \sqsupseteq_c Q$. Or equivalently expressed: the identity translation from Π into Π_{Stop} is fully-abstract w.r.t. barbed testing equivalence in Π and contextual equivalence in Π_{Stop}.

Theorem 3.7. *For all processes $P, Q \in Proc$: $P \sqsubseteq_c Q \iff P \leq_c Q$, and hence also $P \sqsupseteq_c Q \iff P \sim_c Q$.*

Proof. Let P, Q be Stop-free processes. It is sufficient to show that $P \sqsubseteq_{c,\text{may}} Q$ iff $P \leq_{c,\downarrow} Q$ and $P \sqsubseteq_{c,\text{should}} Q$ iff $P \leq_{c,\Downarrow} Q$.

- $P \sqsubseteq_{c,\text{may}} Q \implies P \leq_{c,\downarrow} Q$: Let $P \sqsubseteq_{c,\text{may}} Q$ and $C \in \mathcal{C}$ with $C[P]\downarrow$, i.e. $C[P] = P_0 \xrightarrow{sr} P_1 \ldots \xrightarrow{sr} P_n$ where P_n is successful. Let C' be the context C where every occurrence of Stop in C is replaced by $x(y)$ where x, y are fresh names, and let P'_i be P_i where every occurrence of Stop is replaced by $x(y)$. Then $C'[P] = P'_0 \xrightarrow{sr} P'_1 \ldots \xrightarrow{sr} P'_n$, since ia-reductions do not use Stop, and the axiom $\nu z.\mathsf{Stop} \equiv \mathsf{Stop}$ of structural congruence can equivalently be replaced by $\nu z.x(y) \equiv x(y)$ which holds, since x is chosen fresh.
  Since $P_n \equiv \mathsf{Stop} \mid Q$ also P'_n must be of the right form, i.e. $P'_n \equiv (x(y) \mid P')$ and thus $P'_n \Uparrow^x$. This shows $C'[P] \downarrow_x$. Now $P \sqsubseteq_{c,\text{may}} Q$ shows $C'[Q] \downarrow_x$. Convergence $C[Q]\downarrow$ can be shown by the same argument: the reduction $C'[Q] \xrightarrow{sr} Q'_1 \ldots \xrightarrow{sr} Q'_m$ where $Q'_m \Uparrow^x$, i.e. $Q'_m \equiv \nu\mathcal{X}.(x(y).R_1 \mid R_2)$ can never perform an ia-reduction using the input-prefix $x(y)$, since x was chosen fresh, and thus there is no corresponding output prefix. Inspecting the reduction possibilities this also shows $R_1 = \mathbf{0}$. Thus the reduction $C[Q] \xrightarrow{sr} Q_1 \ldots \xrightarrow{sr} Q_m$ exists where Q_i is Q'_i but the subprocesses $x(y)$ are replaced by Stop. Thus $Q_m \equiv \nu x_1, \ldots x_n.(\mathsf{Stop} \mid R'_2)$ and $C[Q]\downarrow$.
- $P \leq_{c,\downarrow} Q \implies P \sqsubseteq_{c,\text{may}} Q$: Since $\sqsubseteq_{c,\text{may}} = \sqsubseteq_{c,\downarrow_x}$ (Theorem 2.10) it is sufficient to consider the following case: Let $P \leq_{c,\downarrow} Q$, C be a Stop-free context, and $C[P] \downarrow_x$. Then $C[P] \xrightarrow{sr} P_1 \ldots \xrightarrow{sr} P_n \equiv \nu\mathcal{X}.(x(y).P' \mid P'')$, and for $C_1 := ([\cdot] \mid \overline{x}y.\mathsf{Stop})$ we have $C_1[C[P]]\downarrow$, since the reduction for $C[P]$ can be used resulting in $C_1[P_n]$ which reduces in one step to a successful process. $P \leq_c Q$ also implies $C_1[C[Q]]\downarrow$ and the corresponding reduction sequence $C_1[C[Q]] \xrightarrow{sr,*} Q_m$ where Q_m is successful must include an ia-reduction with a redex of the form $x(z).R \mid \overline{x}y.\mathsf{Stop}$. Let $Q_i \xrightarrow{sr,*} Q_{i+1}$ be this step in $C_1[C[Q]] \xrightarrow{sr,*} Q_m$. Then the prefix $C_1[C[Q]] \xrightarrow{sr,i} Q_i$ can be used to construct a reduction sequence $C[Q] \xrightarrow{sr,i} Q'_i$ where $Q'_i \Uparrow^x$ and thus $C[Q] \downarrow_x$.
- $P \leq_{c,\Downarrow} Q \implies P \sqsubseteq_{c,\text{should}} Q$: Let $P \leq_{c,\Downarrow} Q$. We show $C[Q]\Downarrow_x \implies C[P]\Downarrow_x$ for any Stop-free context $C \in \mathcal{C}$. which is sufficient since $\sqsubseteq_{c,\text{should}} = \sqsubseteq_{c,\Downarrow_x}$ (Theorem 2.10). So let C be a Stop-free context s.t. $C[Q]\Downarrow_x$ holds, i.e. $C[Q] \xrightarrow{sr,*} Q'$ and $\neg(Q'\downarrow_x)$. Then also $C_1[C[Q]]\uparrow$ with $C_1 = [\cdot] \mid \overline{x}y.\mathsf{Stop}$, since $C_1[C[Q]] \xrightarrow{sr,*} C_1[Q']$ and $C_1[Q']$ cannot become successful, since otherwise $Q'\downarrow_x$ would hold. Now $P \leq_{c,\Downarrow} Q$ implies $C_1[C[P]]\uparrow$, i.e. $C_1[C[P]] \xrightarrow{sr,*} P'$ and $P'\Uparrow$. Moreover, the reduction sequence $C_1[C[P]] \xrightarrow{sr,*} P'$ can never reduce $\overline{x}y.\mathsf{Stop}$, since otherwise P' cannot be must-divergent, and thus we can assume that $P' \equiv C_1[P'']$ and $C[P] \xrightarrow{sr,*} P''$. Now again $P''\downarrow_x$ cannot hold, since otherwise $C_1[P'']\Uparrow$ would not hold. This shows $C[P]\Downarrow_x$.
- $P \sqsubseteq_{c,\text{should}} Q \implies P \leq_{c,\Downarrow} Q$. Let $P \sqsubseteq_{c,\text{should}} Q$ and C be a context with $C[Q]\uparrow$. We will show $C[P]\uparrow$. Let $C[Q] = Q_0 \xrightarrow{sr} \ldots \xrightarrow{sr} Q_n$ where $Q_n\Uparrow$. Let C' (Q'_i, resp.) be the context C (Q_i, resp.) such that all occurrences of Stop are replaced by $x(y)$ where x, y are fresh for C, P, and Q_i. Then $\neg(Q'_n\downarrow_x)$ since otherwise Q_n would be may-convergent, and also $Q'_i \xrightarrow{sr} Q'_{i+1}$ and thus $C'[Q]\Downarrow_x$. From $P \sqsubseteq_{c,\text{should}} Q$ we also have $C'[P]\Downarrow_x$, i.e. $C'[P] \xrightarrow{sr,*} P'_n$ and $\neg(P'_n \downarrow_x)$.

Let P_i be P'_i where all occurrences of $x(y)$ are replaced by Stop. Then $P_n\Uparrow$, since otherwise $P'_n\downarrow_x$ would hold. Also $P_i \xrightarrow{sr} P_{i+1}$, since $P'_i \xrightarrow{sr} P'_{i+1}$ cannot reduce any occurrence of $x(y).0$. This shows $C[P]\uparrow$. □

4 Proof Methods for Contextual Equivalence

For disproving an equation $P \sim_c Q$ it is sufficient to find a distinguishing context. Proving an equation $P \sim_c Q$ is in general harder, since all contexts must be considered. Hence, in this section we develop proof tools for those proofs.

In Section 4.1 we prove a context lemma, which restricts the class of contexts that need to be taken into account for proving $P \sim_c Q$. In contrast to the coinductive proofs given in [PS96,SW01] for a context lemma for barbed congruence in the π-calculus, we will use an inductive argument and also extend the results to should-convergence. In Section 4.2 we show soundness of applicative similarities which allow coinductive proofs by evaluating the processes P and Q and then analyzing input or output possibilities on open channels. The applicative similarities are related to the "weak early bisimilarities" in the π-calculus, but there are some differences which will be discussed after the definitions.

4.1 A Context Lemma for May- and Should-Convergence

As a preparation for the context lemma we show two basic extension lemmas for $\leq_\downarrow$ and $\leq_\uparrow$. We use contexts of hole depth 1, which are exactly the contexts of the form $[\cdot] \mid R$, $R \mid [\cdot]$, $x(y).[\cdot]$, $\overline{x}y.[\cdot]$, $\nu x.[\cdot]$, and $!\,[\cdot]$.

Lemma 4.1. *Let $P, Q \in Proc_{\mathsf{Stop}}$. If $\sigma(P) \mid R \leq_\downarrow \sigma(Q) \mid R$ for all $\sigma \in \Sigma$ and $R \in Proc_{\mathsf{Stop}}$, then also $\sigma(C[P]) \mid R \leq_\downarrow \sigma(C[Q]) \mid R$ for all $C \in \mathcal{C}_{\mathsf{Stop}}$ of hole depth 1, all $\sigma \in \Sigma$ and $R \in Proc_{\mathsf{Stop}}$.*

Proof. Here we only consider the case $C = !\,[\cdot]$, the other cases are proved in the appendix (Lemma B.2). Let $\sigma(!\,P) \mid R \xrightarrow{sr,n} P_n$ where P_n is successful. We show $\sigma(!\,Q) \mid R\downarrow$ by induction on n. If $n = 0$, i.e. $\sigma(!\,P) \mid R$ is successful, then R or P is successful. Thus also $\sigma(P) \mid R$ is successful, and the precondition of the lemma shows $(\sigma(Q) \mid R)\downarrow$, which in turn also implies $\sigma(!\,Q) \mid R\downarrow$. For $n > 0$ let $\sigma(!\,P) \mid R \xrightarrow{sr} P_1$ be the first reduction of $\sigma(!\,P) \mid R \xrightarrow{sr,n} P_n$:

- If the redex is inside R, then the same reduction can be performed for $\sigma(!\,Q \mid R)$ and thereafter the induction hypothesis shows the claim.
- The redex uses one instance of $\sigma(P)$ and perhaps parts of R, i.e. $P_1 \equiv \sigma(!\,P) \mid R_P$, where $R \mid \sigma(P) \xrightarrow{sr} R_P$. The induction hypothesis applied to P_1 shows that $(\sigma(!\,Q) \mid R_P)\downarrow$. This implies $\sigma(!\,Q) \mid \sigma(P) \mid R)\downarrow$, since $\sigma(!\,Q) \mid \sigma(P) \mid R) \xrightarrow{sr} \sigma(!\,Q) \mid R_P)$. Finally, the precondition of the lemma shows that $\sigma(!\,Q) \mid \sigma(P) \mid R \equiv \sigma(P) \mid (\sigma(!\,Q) \mid R) \leq_\downarrow \sigma(Q) \mid (\sigma(!\,Q) \mid R) \equiv \sigma(!\,Q) \mid R$. Thus we also have $\sigma(!\,Q) \mid R\downarrow$.
- The redex uses two instances of $\sigma(P)$, i.e. $P_1 \equiv \sigma(!\,P) \mid R \mid P'$, s.t. $\sigma(P) \mid \sigma(P) \xrightarrow{sr} P'$. The induction hypothesis applied to P_1 shows $\sigma(!\,Q) \mid R \mid P'\downarrow$. We also have $\sigma(!\,Q) \mid R \mid \sigma(P) \mid \sigma(P)\downarrow$, since $\sigma(P) \mid \sigma(P) \xrightarrow{sr} P'$. Applying the precondition of the lemma twice shows:

$$\begin{aligned} & \sigma(!\,Q) \mid R \mid \sigma(P) \mid \sigma(P) \\ \equiv\ & \sigma(P) \mid (\sigma(P) \mid (\sigma(!\,Q) \mid R)) \\ \leq_\downarrow\ & \sigma(Q) \mid (\sigma(P) \mid (\sigma(!\,Q) \mid R)) \equiv \sigma(P) \mid (\sigma(!\,Q) \mid R) \\ \leq_\downarrow\ & \sigma(Q) \mid (\sigma(!\,Q) \mid R) \equiv \sigma(!\,Q) \mid R \end{aligned}$$

and thus $\sigma(!\,Q) \mid R\downarrow$. □

Lemma 4.2. *Let $P, Q \in Proc_{\mathsf{Stop}}$. Assume that $\sigma(C[Q]) \mid R \leq_\downarrow \sigma(C[P]) \mid R$ for all $\sigma \in \Sigma$, all $C \in \mathcal{C}_{\mathsf{Stop}}$, and all $R \in Proc_{\mathsf{Stop}}$. If $\sigma(P) \mid R \leq_\uparrow \sigma(Q) \mid R$ for all $\sigma \in \Sigma$ and $R \in Proc_{\mathsf{Stop}}$, then also $\sigma(C[P]) \mid R \leq_\uparrow \sigma(C[Q]) \mid R$ for all $C \in \mathcal{C}_{\mathsf{Stop}}$ of hole depth 1, all $\sigma \in \Sigma$ and $R \in Proc_{\mathsf{Stop}}$.*

Proof. This can be proved analogously to Lemma 4.1 where $\leq_\downarrow$ is replaced by $\leq_\uparrow$, and the base cases of the form "if $\sigma(C[P]) \mid R$ is successful, then $\sigma(C[Q]) \mid R\downarrow$" are replaced by "if $(\sigma(C[P]) \mid R)\Uparrow$, then $(\sigma(C[Q]) \mid R)\uparrow$" . which always holds, since $\sigma(C[Q]) \mid R \leq_\downarrow \sigma(C[P]) \mid R$.

Now we prove the context lemma:

Theorem 4.3 (Context Lemma). *For all processes P, Q:*

- *If for all σ, R: $\sigma(P) \mid R \leq_\downarrow \sigma(Q) \mid R$, then $P \leq_{c,\downarrow} Q$.*
- *If for all σ, R: $\sigma(P) \mid R \leq_\downarrow \sigma(Q) \mid R \wedge \sigma(P) \mid R \leq_\Downarrow \sigma(Q) \mid R$, then $P \leq_c Q$.*

Proof. For the first part it is sufficient to show that $\sigma(C[P]) \mid R \leq_\downarrow \sigma(C[Q]) \mid R$ for all $C \in \mathcal{C}_{\mathsf{Stop}}$, all $\sigma \in \Sigma$ and $R \in Proc_{\mathsf{Stop}}$, which follows from Lemma 4.1 by using induction on the depth of the hole of the context C.

For the second part we use the fact that $\sigma(P) \mid R \leq_\downarrow \sigma(Q) \mid R$ for all σ, R implies $\sigma(C[P]) \mid R \leq_\downarrow \sigma(C[Q]) \mid R$ for all σ, R, C. By induction on the depth of the hole of the context C, the fact that $\sigma(P) \mid R \leq_\Downarrow \sigma(Q) \mid R$ is equivalent to $\sigma(Q) \mid R \leq_\uparrow \sigma(P) \mid R$, and Lemma 4.2 it follows $\sigma(C[Q]) \mid R \leq_\uparrow \sigma(C[P]) \mid R$ for all $C \in \mathcal{C}_{\mathsf{Stop}}, \sigma \in \Sigma$ which shows $P \leq_{c,\Downarrow} Q$.

Remark 4.4. Note that the condition for all σ, R: $\sigma(P) \mid R \leq_\Downarrow \sigma(Q) \mid R$ is in general not sufficient for $P \leq_{c,\Downarrow} Q$. Let $P := \nu x.\overline{x}y \mid x(y).\mathsf{Stop} \mid x(y)$ and $Q := \mathbf{0}$. Then $\sigma(P) \mid R \leq_\Downarrow \sigma(Q) \mid R$ for all σ and R, but $P \not\leq_{c,\Downarrow} Q$, since the context $!\,[\cdot]$ distinguishes the processes P and Q: $!P\Downarrow$ while $!Q\Uparrow$.

4.2 Applicative Similarities

As a first step we provide coinductive definitions of $\leq_\downarrow$ and $\leq_\uparrow$, which will ease some of our proofs:

Definition 4.5 ($\downarrow$- and $\uparrow$-similarity). $\downarrow$-Similarity $\precsim_\downarrow$ *is the greatest fixpoint of the operator $F_\downarrow$ on binary relations on processes: for $\eta \subseteq (Proc_{\mathsf{Stop}} \times Proc_{\mathsf{Stop}})$, $(P, Q) \in F_\downarrow(\eta)$ iff*

- *If P is successful, then $Q\downarrow$.*
- *If $P \xrightarrow{sr} P'$, then there exists Q' such that $Q \xrightarrow{sr,*} Q'$ and $(P', Q') \in \eta$.*

$\uparrow$-Similarity $\precsim_\uparrow$ *is the greatest fixpoint of the operator $F_\uparrow$ on binary relations on processes: for $\eta \subseteq (Proc_{\mathsf{Stop}} \times Proc_{\mathsf{Stop}})$, $(P, Q) \in F_\uparrow(\eta)$ iff*

- *If $P\Uparrow$, then $Q\uparrow$.*
- *If $P \xrightarrow{sr} P'$, then there exists Q' such that $Q \xrightarrow{sr,*} Q'$ and $(P', Q') \in \eta$.*

For an operator F on relations $\subseteq (Proc_{\mathsf{Stop}} \times Proc_{\mathsf{Stop}})$, the relation η is *F-dense*, iff $\eta \subseteq F(\eta)$. The coinduction principle is that an F-dense relation η is contained in the greatest fixpoint of F.

Lemma 4.6. $\leq_\downarrow = \precsim_\downarrow$ *and* $\leq_\uparrow = \precsim_\uparrow$.

Definition of Applicative Similarity

Before defining "applicative similarities" for $\downarrow$ and $\uparrow$, we define the property of a relation to preserve the input and output capabilities of one process w.r.t. another process. This definition is analogous to preserving actions in labelled bisimilarities. We prefer this definition here, since we can omit the definition of a labelled transition system.

Definition 4.7. *Given processes $P, Q \in Proc_{\mathsf{Stop}}$ and a binary relation $\eta \subseteq (Proc_{\mathsf{Stop}} \times Proc_{\mathsf{Stop}})$, we say η* preserves the input/output capabilities of P w.r.t. Q *iff:*

- Open input: *If* $P \equiv \nu\mathcal{X}.(x(y).P_1 \mid P_2)$ *(with* $x \notin \mathcal{X}$*), then for every name* z *there exists a process* Q' *s.t.* $Q \xrightarrow{sr,*} Q'$, $Q' \equiv \nu\mathcal{Y}.(x(y).Q_1 \mid Q_2)$ *with* $x \notin \mathcal{Y}$ *and* $(\nu\mathcal{X}.(P_1[z/y] \mid P_2), \nu\mathcal{Y}.(Q_1[z/y] \mid Q_2)) \in \eta$.
- Open output: *If* $P \equiv \nu\mathcal{X}.(\overline{x}y.P_1 \mid P_2)$ *with* $x, y \notin \mathcal{X}$*, then there exists a process* Q' *s.t.* $Q \xrightarrow{sr,*} Q'$, $Q' \equiv \nu\mathcal{Y}.(\overline{x}y.Q_1 \mid Q_2)$ *with* $x, y \notin \mathcal{Y}$ *and* $(\nu\mathcal{X}.(P_1 \mid P_2), \nu\mathcal{Y}.(Q_1 \mid Q_2)) \in \eta$.
- Bound output: *If* $P \equiv \nu\mathcal{X}, \nu y.(\overline{x}y.P_1 \mid P_2)$ *with* $x \notin \mathcal{X}$*, then there exists a process* Q' *s.t.* $Q \xrightarrow{sr,*} Q'$, $Q' \equiv \nu\mathcal{Y}, \nu y.(\overline{x}y.Q_1 \mid Q_2)$ *with* $x \notin \mathcal{Y}$ *and* $(\nu\mathcal{X}.(P_1 \mid P_2), \nu\mathcal{Y}.(Q_1 \mid Q_2)) \in \eta$.

Definition 4.8 ((Full) Applicative Similarities). *Let* $F_{b,\downarrow}$ *be the following operator: for* $\eta \subseteq (Proc_{\mathsf{Stop}} \times Proc_{\mathsf{Stop}})$, $(P, Q) \in F_{b,\downarrow}(\eta)$ *iff*

- *If* P *is successful, then* $Q\downarrow$.
- *If* $P \xrightarrow{sr} P'$*, then* $\exists Q'$ *with* $Q \xrightarrow{sr,*} Q'$ *and* $(P', Q') \in \eta$.
- *If* P *is not successful, then* η *preserves the input/output capabilities of* P *w.r.t.* Q.

Let $F_{b,\uparrow}$ *be the following operator: for* $\eta \subseteq (Proc_{\mathsf{Stop}} \times Proc_{\mathsf{Stop}})$, $(P, Q) \in F_{b,\uparrow}(\eta)$ *iff*

- *If* $P\Uparrow$*, then* $Q\uparrow$.
- *If* $P \xrightarrow{sr} P'$*, then* $\exists Q'$ *with* $Q \xrightarrow{sr,*} Q'$ *and* $(P', Q') \in \eta$.
- *If* P *is not must-divergent, then* η *preserves the input/output capabilities of* P *w.r.t.* Q.
- $Q \precsim_{b,\downarrow} P$

Applicative $\downarrow$-similarity $\precsim_{b,\downarrow}$ *is the greatest fixpoint of* $F_{b,\downarrow}$*, and* applicative $\uparrow$-similarity $\precsim_{b,\uparrow}$ *is the greatest fixpoint of* $F_{b,\uparrow}$. Full applicative $\downarrow$-similarity $\precsim^{\sigma}_{b,\downarrow}$ *and* full applicative $\uparrow$-similarity $\precsim^{\sigma}_{b,\uparrow}$ *are defined as* $P \precsim^{\sigma}_{b,\downarrow} Q$ *(*$P \precsim^{\sigma}_{b,\uparrow} Q$*, resp.) iff* $\sigma(P) \precsim_{b,\downarrow} \sigma(Q)$ *(*$\sigma(P) \precsim_{b,\uparrow} \sigma(Q)$*, resp.) for all* $\sigma \in \Sigma$.

Full applicative similarity $\precsim_b$ *is defined as the intersection* $\precsim_b := \precsim^{\sigma}_{b,\downarrow} \cap (\precsim^{\sigma}_{b,\uparrow})^{-1}$. Mutual full $\Downarrow$-applicative similarity $\simeq_{b,\Downarrow}$ *is the intersection* $\precsim^{\sigma}_{b,\uparrow} \cap (\precsim^{\sigma}_{b,\uparrow})^{-1}$ *and* mutual full applicative similarity $\simeq_b$ *is the intersection of full applicative similarity with its inverse relation, i.e.* $\simeq_b := \precsim_b \cap (\precsim_b)^{-1}$.

Discussion and Properties

We discuss our definitions of applicative similarity. We first consider the applicative $\downarrow$-similarity. Its definition is related to early labelled bisimilarity for the π-calculus [SW01], but adapted to the successfulness-test. However, there is a difference whether a similarity or a bisimilarity is used. Applicative $\downarrow$-bisimilarity would be defined as the largest relation $\mathcal{R}$ such that $\mathcal{R}$ and $\mathcal{R}^{-1}$ are $F_{b,\downarrow}$-dense. The intersection of $\precsim_{b,\downarrow}$ with its inverse, i.e. the relation $\precsim_{b,\downarrow} \cap (\precsim_{b,\downarrow})^{-1}$, is much coarser than applicative $\downarrow$-bisimilarity. For instance, the processes $P_{a,bc} := \mathtt{choice}(a(y), \mathtt{choice}(b(y), c(y)))$ and $P_{ab,c} := \mathtt{choice}(\mathtt{choice}(a(y), b(y)), c(y))$ are not applicative $\downarrow$-bisimilar, since e.g. after reducing $P_{a,bc} \xrightarrow{sr} \mathtt{choice}(b(y), c(y))$ there is no process P' with $P_{ab,c} \xrightarrow{sr,*} P'$ s.t. $\mathtt{choice}(b(y), c(y))$ and P' are applicative $\downarrow$-bisimilar. However, $P_{a,bc} \precsim_{b,\downarrow} P_{ab,c}$ and $P_{ab,c} \precsim_{b,\downarrow} P_{a,bc}$.

However, the following example adapted from an example given by [SW01] shows that even $\precsim_{b,\downarrow}$ is more discriminating than contextual may preorder:

Proposition 4.9. *Let* $S_{xy} := x(z).\overline{y}z$ *and* $S_{yx} := y(z).\overline{x}z$*. Then for the processes* $P := \overline{a}x \mid !\, S_{xy} \mid !\, S_{yx}$ *and* $Q := \overline{a}y \mid !\, S_{xy} \mid !\, S_{yx}$ *it holds:* $\neg(P \precsim_{b,\downarrow} Q)$ *(and thus also* $\neg(P \precsim^{\sigma}_{b,\downarrow} Q)$*), but* $P \leq_{c,\downarrow} Q$.

Proof. $P \precsim_{b,\downarrow} Q$ does not hold, since the output on channel a is different. $P \leq_{c,\downarrow} Q$ is proved in the appendix in Lemma B.3.

The definition of applicative $\uparrow$-similarity includes the property $Q \precsim_{b,\downarrow} P$, i.e.:

Lemma 4.10. $P \precsim_{b,\uparrow} Q \implies Q \precsim_{b,\downarrow} P$.

Thus – like the discussion before on bisimilarity – this requirement makes the relation $\precsim_{b,\uparrow}$ very fine-grained: the processes $P_{a,bc}$ and $P_{ab,c}$ are not applicative $\uparrow$-similar, although the processes are contextually equivalent. The reason for our chosing this definition is that we did not find a coarser $\uparrow$-similarity which is sound for contextual should-preorder. Properties that must hold for such a definition are that it preserves may-divergence, but also (due to Theorem 3.7) that it preserves the predicate $\restriction_x$ for Stop-free processes. The second condition holds for $\precsim_{b,\uparrow}$, since we added $Q \precsim_{b,\downarrow} P$. Obviously, $\precsim_{b,\downarrow}$ preserves may-convergence and $\precsim_{b,\uparrow}$ preserves may-divergence:

Lemma 4.11. $\precsim_{b,\downarrow} \subseteq \precsim_{\downarrow}$ *and* $\precsim_{b,\uparrow} \subseteq \precsim_{\uparrow}$.

We show that for Stop-free processes $\precsim_{b,\downarrow}$ preserves the $\downarrow$-property, and $\precsim_{b,\uparrow}$ preserves the $\restriction$-property:

Lemma 4.12. *For* Stop*-free processes* $P, Q \in Proc$*:*

- *If* $P \precsim_{b,\downarrow} Q$*, then* $P \sqsubseteq_{\downarrow_x} Q$.
- *If* $P \precsim_{b,\uparrow} Q$*, then* $P \sqsubseteq_{\restriction_x} Q$.

Proof. The first part can be shown by induction on the length of a reduction $P \xrightarrow{sr,n} P'$ s.t. $P' \not\downarrow_x$. For the second part let $P \xrightarrow{sr,n} P'$ such that $P' \restriction_x$. By induction on n and $P \precsim_{b,\uparrow} Q$ one can show that there is a process Q' with $Q \xrightarrow{sr,*} Q'$ and $P' \precsim_{b,\uparrow} Q'$. Since $P' \precsim_{b,\uparrow} Q'$ implies $Q' \precsim_{b,\downarrow} P'$ the first part of this lemma shows that $Q' \restriction_x$ must hold.

Soundness of Applicative Similarity

We now show soundness of our applicative similarities.

Proposition 4.13. *For all* $P, Q, R \in Proc_{\mathsf{Stop}}$ *and* ν*-prefixes* $\mathcal{X}$ *the following implications hold:*

1. $(P \precsim_{b,\downarrow} Q) \implies \nu\mathcal{X}.(P \mid R) \precsim_{\downarrow} \nu\mathcal{X}.(Q \mid R)$.
2. $(P \precsim_{b,\uparrow} Q) \implies \nu\mathcal{X}.(P \mid R) \precsim_{\uparrow} \nu\mathcal{X}.(Q \mid R)$.

Proof. The first claim holds, since the relation $\precsim_{\downarrow} \cup \{(\nu\mathcal{X}.(P \mid R), \nu\mathcal{X}.(Q \mid R)) \mid P \precsim_{b,\downarrow} Q, \text{ for any } \mathcal{X}, R\}$ is $F_{\downarrow}$-dense, which is proved in the appendix, Lemma B.4.

The second claim holds, since the relation $\precsim_{\uparrow} \cup \{(\nu\mathcal{X}.(P \mid R), \nu\mathcal{X}.(Q \mid R)) \mid P \precsim_{b,\uparrow} Q, \text{ for any } \mathcal{X}, R\}$ is $F_{\uparrow}$-dense, which is proved in the appendix, Lemma B.5.

Theorem 4.14 (Soundness of Full Applicative Similarities). *The following inclusions hold:*

- $\precsim^{\sigma}_{b,\downarrow} \subseteq \leq_{c,\downarrow}$,
- $\precsim_b \subseteq \leq_c$*, and*
- $\simeq_{b,\Downarrow} = \simeq_b \subseteq \sim_c$.

Proof. For the first part Proposition 4.13 part (1) shows that $\sigma(P) \precsim_{b,\downarrow} \sigma(Q)$ implies $\sigma(P) \mid R \precsim_{\downarrow} \sigma(Q) \mid R$ for all σ, R. Hence also $P \precsim^{\sigma}_{b,\downarrow} Q$ implies $\sigma'(P) \mid R \precsim_{\downarrow} \sigma'(Q) \mid R$ for all σ', R. Replacing $\precsim_{\downarrow}$ by $\leq_{\downarrow}$ (Lemma 4.6) makes the context lemma (Theorem 4.3) applicable and shows $P \leq_{c,\downarrow} Q$.

For the second part we apply both parts of Proposition 4.13 which shows that $P \precsim^{\sigma}_{b,\downarrow} Q$ and $Q \precsim^{\sigma}_{b,\uparrow} P$ imply that $\sigma'(P) \mid R \precsim_{\downarrow} \sigma'(Q) \mid R$ and $\sigma'(Q) \mid R \precsim_{\uparrow} \sigma'(P) \mid R$ for all σ', R. Replacing $\precsim_{\downarrow}$ by $\leq_{\downarrow}$ and $\precsim_{\uparrow}$ by $\geq_{\Downarrow}$ (Lemma 4.6) makes Theorem 4.3 applicable and shows $P \leq_c Q$.

The equation of the last part follows from Lemma 4.10. The inclusion of the last part follows from the second part and the definitions of $\simeq_b$ and $\sim_c$.

5 Equivalences and the Contextual Ordering

In this section we analyze the contextual ordering and also show some contextual equivalences.

5.1 Correctness of Deterministic Interaction

As a first result we apply Theorem 4.14 to show correctness of a restricted variant of the *ia*-reduction that ensures determinism. This correctness result is expected, but it lets us demonstrate our developed proof techniques for an exemplary program optimization, and moreover the result can be used to show a completeness result w.r.t. the tests in the context lemma (Corollary 5.3).

Theorem 5.1 (Correctness of Deterministic Interaction). *For all processes P, Q the equation $\nu x.(x(y).P \mid \overline{x}z.Q)) \sim_c \nu x.(P[z/y] \mid Q)$ holds.*

Proof. We use Theorem 4.14 and show that $\nu x.(x(y).P \mid \overline{x}z.Q)) \simeq_{b,\Downarrow} \nu x.(P[z/y] \mid Q)$. Let

$$\mathcal{S} = \{(\sigma(\nu x.(x(y).P \mid \overline{x}z.Q)), \sigma(\nu x.(P[z/y] \mid Q))) \mid \text{for all } x, y, z, P, Q, \sigma\} \cup \equiv$$

We will show that $\mathcal{S}$ and $\mathcal{S}^{-1}$ are $F_{b,\uparrow}$-dense.

We first show that $\mathcal{S}$ and $\mathcal{S}^{-1}$ are $F_{b,\downarrow}$-dense: Let $(R_1, R_2) = (\sigma(\nu x.(x(y).P \mid \overline{x}z.Q)), \sigma(\nu x.(P[z/y] \mid Q)))$

- R_1 is not successful, so there is nothing to show.
- If $R_1 \xrightarrow{sr} R_1'$, then $R_1' \equiv R_2$ and $(R_2, R_2) \in \mathcal{S}$
- R_1 does not haven an open input or output, thus there is nothing to show.
- If R_2 is success, then $R_1\downarrow$, since $R_1 \xrightarrow{sr} R_2$.
- If $R_2 \xrightarrow{sr} R_2'$, then $R_1 \xrightarrow{sr,2} R_2'$ and $(R_2', R_2') \in \mathcal{S}^{-1}$
- If R_2 has an open input or output, then $R_1 \xrightarrow{sr} R_2$ and the condition of $F_{b,\downarrow}$ can be fullfilled.

Thus $R_1 \precsim_{b,\downarrow} R_2$ and $R_2 \precsim_{b,\downarrow} R_1$ for any $(R_1, R_2) \in \mathcal{S}$.

We now prove that $\mathcal{S}$ and $\mathcal{S}^{-1}$ are $F_{b,\uparrow}$-dense.

- If $R_1\Uparrow$ then $R_2\Uparrow$, since $R_1 \xrightarrow{sr} R_2$.
- If $R_1 \xrightarrow{sr} R_1'$, then $R_1' \equiv R_2$ and $(R_1', R_1') \in \mathcal{S}$.
- R_1 does not have an open input or output, thus there is nothing to show.
- $R_2 \precsim_{b,\downarrow} R_1$ is already proved.
- If $R_2\Uparrow$ then clearly $R_1\uparrow$.
- If $R_2 \xrightarrow{sr} R_2'$, then $R_2' \equiv R_1$ and $(R_2', R_2') \in \mathcal{S}^{-1}$.
- If R_2 has an open input or output, then $R_1 \xrightarrow{sr} R_2$ and the condition of $F_{b,\uparrow}$ can be fullfilled.
- $R_1 \precsim_{b,\downarrow} R_2$ is already proved. □

Now we show that contextual preorder and equivalence do not change, if we additionally consider all name substitutions:

Lemma 5.2. *For $\xi \in \{\downarrow, \Downarrow\}$: $P \leq_{c,\xi} Q$ iff $\forall C \in \mathcal{C}_{\mathsf{Stop}}$, $\sigma \in \Sigma$: $C[\sigma(P)] \leq_\xi C[\sigma(Q)]$.*

Proof. "⇐" is trivial. For "⇒" we define for $\sigma = \{x_1 \mapsto y_1, \ldots, x_n \mapsto y_n\}$ the context $C_\sigma := \nu\mathcal{W}.(w_1(x_1).w_2(x_2).\ldots.w_n(x_n).[\cdot] \mid \overline{w_1}y_1 \mid \ldots \mid \overline{w_n}y_n)$ where $\mathcal{W} = \{w_1, \ldots, w_n\}$ and $\mathcal{W} \cap (\mathsf{fn}(P) \cup \mathsf{fn}(Q)) = \emptyset$. The reductions $C_\sigma[P] \xrightarrow{ia,*} \sigma(P)$ and $C_\sigma[Q] \xrightarrow{ia,*} \sigma(Q)$ are valid, where all *ia*-steps are deterministic and thus by Theorem 5.1 $C_\sigma[P] \sim_c \sigma(P)$ and $C_\sigma[Q] \sim_c \sigma(Q)$. Now let $P \leq_{c,\xi} Q$ and let C, σ s.t. $C[\sigma(P)]\xi$. Since $\sigma(P) \sim_c C_\sigma[P]$ also $C[C_\sigma(P)]\xi$ which in turn implies $C[C_\sigma(Q)]\xi$. Since $C_\sigma[Q] \sim_c \sigma(Q)$, this shows $C[\sigma(Q)]\xi$. Since C, σ were chosen arbitrarily, $C[\sigma(P)] \leq_\xi C[\sigma(Q)]$ holds for all $C \in \mathcal{C}_{\mathsf{Stop}}$ and $\sigma \in \Sigma$.

The theorem implies that the tests of the context lemma (Theorem 4.3) are complete w.r.t. contextual preorder:

Corollary 5.3. *For all $P, Q \in Proc_{\mathsf{Stop}}$:*

- $P \leq_{c,\downarrow} Q$ *iff for all* $\sigma \in \Sigma, R \in Proc_{\mathsf{Stop}}$: $\sigma(P) \mid R \leq_{\downarrow} \sigma(Q) \mid R$.
- $P \leq_c Q$ *iff for all* $\sigma \in \Sigma, R \in Proc_{\mathsf{Stop}}, \xi \in \{\downarrow, \Downarrow\}$: $\sigma(P) \mid R \leq_{\xi} \sigma(Q) \mid R$.

5.2 Results on the Contextual Ordering

We show several properties on the contextual ordering and equivalence. All successful processes are in the same equivalence class. More surprisingly, all may-convergent processes are equal w.r.t. contextual may-convergence, which is a strong motivation to also consider should-convergence. Further results are that Stop is the largest element in the contextual ordering, and there is no least element:

Theorem 5.4. *1. If P, Q are two successful processes, then $P \sim_c Q$.*
2. If P, Q are two processes with $P\downarrow, Q\downarrow$, then $P \sim_{c,\downarrow} Q$.
3. There are may-convergent processes P, Q with $P \not\sim_c Q$.
4. Stop is the greatest process w.r.t. $\leq_c$.
5. $\mathbf{0}$ is the smallest process w.r.t. $\leq_{c,\downarrow}$.
6. There is no smallest process w.r.t. $\leq_c$.

Proof. For (1) let P and Q be successful. Then for any $\sigma \in \Sigma$ and any $R \in Proc_{\mathsf{Stop}}$ also $\sigma(P) \mid R$ and $\sigma(Q) \mid R$ are successful. This implies $\sigma(P) \mid R\downarrow$, $\sigma(Q) \mid R\downarrow$, $\sigma(P) \mid R\Downarrow$, and $\sigma(Q) \mid R\Downarrow$ for all $R \in Proc_{\mathsf{Stop}}$ and $\sigma \in \Sigma$ and thus the context lemma (Theorem 4.3) shows $P \sim_c Q$.

Since $P\downarrow \implies \sigma(P) \mid R\downarrow$ for any process P, R and $\sigma \in \Sigma$, the context lemma (Theorem 4.3) shows part (2).

For (3) the empty context distinguishes $\mathtt{choice}(\mathsf{Stop}, \mathbf{0})$ and Stop: $\mathtt{choice}(\mathsf{Stop}, \mathbf{0})\downarrow$, and $\mathtt{choice}(\mathsf{Stop}, \mathbf{0})\uparrow$, while $\mathsf{Stop}\Downarrow$, hence $\mathtt{choice}(\mathsf{Stop}, \mathbf{0}) \not\sim_c \mathsf{Stop}$.

For part (4) clearly $\mathsf{Stop} \mid R\Downarrow$ for all R. Since $\mathsf{Stop} \mid R\Downarrow \implies \mathsf{Stop} \mid R\downarrow$, we have $\sigma(P) \mid R \leq_{\downarrow} \mathsf{Stop} \mid R$ and $\sigma(P) \mid R \leq_{\Downarrow} \mathsf{Stop} \mid R$ for any P, σ, and R. Thus the Theorem 4.3 shows $P \leq_c \mathsf{Stop}$ for any process P.

Part (5) follows from Theorem 4.14, since $\{(\mathbf{0}, P) \mid P \in Proc_{\mathsf{Stop}}\}$ is $F_{b,\downarrow}$-dense.

For (6) assume that there is a process P_0 that is the smallest one, i.e. $P_0 \leq_c P$ for all processes P. Then $P_0\Uparrow$, since $P_0 \leq_c \mathbf{0}$. Let $P_0 \xrightarrow{*} P_1$, such that $P_1 = D[x(y).P_3]$, and where x is free. With $D_1 = \overline{x}y.\mathsf{Stop}$ we obtain $D_1[P_1]\downarrow$, but $D_1[\mathbf{0}] \equiv \overline{x}y.\mathsf{Stop}\Uparrow$. We argue similarly for outputs. Thus the reducts of P_0 do not have open outputs. Now let $P = x(y).\mathbf{0}$, where by our assumption $P_0 \leq_{c,\Downarrow} P$ holds. Let $D = [\cdot] \mid \overline{x}y.\mathbf{0} \mid x(y).\mathsf{Stop}$. Then $D[P_0]\Downarrow$, since there is no communication between the reducts of P_0 and D, but D can always be reduced to a successful process. Now consider $D[P]$. It is $D[P] \to \mathbf{0} \mid x(y).\mathsf{Stop}$, which is must-divergent, hence we have reached the contradiction $P_0 \not\leq_{c,\Downarrow} P$.

As an important result, we show that it is sufficient to test the should-convergence behavior in all contexts, since all tests for may-convergence can be encoded:

Theorem 5.5. $\leq_{c,\Downarrow} \;=\; \leq_c \;\neq\; \leq_{c,\downarrow}$.

Proof. For the equation $\leq_{c,\Downarrow} \;=\; \leq_c$ it sufficient to show that $\leq_{c,\Downarrow} \subseteq \leq_{c,\downarrow}$: let $C_{x,y,\mathcal{X}} := \mathbf{!}\,\nu x, y, \nu\mathcal{X}.[\cdot]$. For any process P with $x, y \notin \mathsf{fn}(P)$ and $\mathcal{X} \supseteq \mathsf{fn}(P)$ one can verify that $P\downarrow$ iff $C_{x,y,\mathcal{X}}[P]\Downarrow$: If $P\downarrow$ then $P' := \nu x, y.\nu\mathcal{X}.P\downarrow$ by Lemma 3.6 and for $\mathbf{!}\,P'$ we can always generate a parallel copy of P', and thus $C_{x,y,\mathcal{X}}[P]\Downarrow$. If $C_{x,y,\mathcal{X}}[P]\Downarrow$, then $\nu x, y, \mathcal{X}.P\downarrow$, since parallel copies of $\nu x, y, \mathcal{X}.P$ cannot communicate due to the name restriction. Now Lemma 3.6 shows $P\downarrow$. Now let $P \leq_{\Downarrow} Q$, $C[P]\downarrow$, but $C[Q]\Uparrow$. With fresh names x, y, $\mathcal{X} = \mathsf{fn}(P) \cup \mathsf{fn}(Q)$: $C_{x,y}[C[P]]\Downarrow$ but $C_{x,y}[C[Q]]\uparrow$ which contradicts $P \leq_{\Downarrow} Q$. The inequality follows from Theorem 5.4 items (5), (6).

In Π_{Stop} contextual should-preorder does not imply contextual may-equivalence.

Proposition 5.6. $\leq_{c,\Downarrow} \not\subseteq \sim_{c,\downarrow}$

Proof. Clearly, $\mathbf{0} \leq_c \mathsf{Stop}$, since Stop is a largest element of $\leq_c$, but $\mathbf{0} \Uparrow$ while $\mathsf{Stop}\downarrow$.

We conclude this subsection, by analyzing several equations, including the ones from [EG04].

Theorem 5.7. *For all processes P, Q the following equivalences hold:*

1. $!\,P \sim_c !!\,P$.
2. $!\,P \mid !\,P \sim_c !\,P$.
3. $!\,(P \mid Q) \sim_c !\,P \mid !\,Q$.
4. $!\,\mathbf{0} \sim_c \mathbf{0}$.
5. $!\,\mathsf{Stop} \sim_c \mathsf{Stop}$.
6. $!\,(P \mid Q) \sim_c !\,(P \mid Q) \mid P$.
7. $x(y).\nu z.P \sim_c \nu z.x(y).P$ *if* $z \notin \{x, y\}$.
8. $\overline{x}y.\nu z.P \sim_c \nu z.\overline{x}y.P$ *if* $z \notin \{x, y\}$.

Proof. This holds, since $\mathcal{S}_i \cup \precsim_{b,\uparrow}$ and $\mathcal{S}_i^{-1} \cup \precsim_{b,\uparrow}$ are $F_{b,\uparrow}$-dense, where $\mathcal{S}_i := \{(R \mid l_i, R \mid r_i) \mid \text{for all } R, P, Q\}$, and l_i, r_i are the left and right hand side of the i^{th} equation.

5.3 Results for the Stop-free Calculus

In this section we go back to the π-calculus Π, and transfer some of the results for Π_{Stop} into Π. The key property is Theorem 3.7 which enables this transfer.

Corollary 5.8. *All equations in Theorem 5.7 (except for equation 5) also hold in Π for* Stop*-free processes, and for barbed testing equivalence $\sqsupseteq_c$. Deterministic interaction (see Theorem 5.1) is correct in Π for $\sqsupseteq_c$.*

Also Theorem 5.5 can be transferred to Π by applying Theorem 3.7, which shows that barbed should-testing preorder implies barbed may-testing equivalence (which does not hold for Π_{Stop} and contextual preorders, see Proposition 5.6):

Corollary 5.9. *For all* Stop*-free processes $P, Q \in Proc$: $P \sqsubseteq_{c,\text{should}} Q$ implies $P \sqsupseteq_{c,\text{may}} Q$.*

Proof. The inclusion $\sqsubseteq_{c,\text{should}} \subseteq (\sqsubseteq_{c,\text{may}})^{-1}$ was proved in Theorem 2.10. The inclusion $\sqsubseteq_{c,\text{should}} \subseteq (\sqsubseteq_{c,\text{may}})$ follows from Theorems 5.5 and 3.7.

Finally, we show that there is no surjective encoding from Π_{Stop} into Π which preserves the ordering of processes w.r.t. contextual preorder in Π_{Stop} and barbed testing preorder in Π. This result supports the conjecture that Stop cannot be encoded in the Stop-free calculus.

Theorem 5.10. *There is no surjective translation $\psi : \Pi_{\mathsf{Stop}} \to \Pi$ s.t. for all $P, Q \in Proc_{\mathsf{Stop}}$: $P \leq_c Q \implies \psi(P) \sqsubseteq_c \psi(Q)$.*

Proof. This holds since Stop is a largest element of Π_{Stop} w.r.t. $\leq_c$, but in Π there is no largest element w.r.t. $\sqsubseteq_c$: Assume the claim is false, and P is a largest element w.r.t. $\sqsubseteq_c$. Let $\mathcal{X} = \mathsf{fn}(P)$ and $x \notin \mathcal{X}$. Then $x(z) \not\sqsubseteq_c P$, since $\nu\mathcal{X}.x(z)\downarrow_x$ but $\nu\mathcal{X}.P \not\downarrow_x$.

6 Conclusion

We introduced and analyzed the calculus Π_{Stop} – a π-calculus with Stop which indicates successful termination. Using contextual equivalence with may- and should-convergence we have shown a context lemma and proved soundness of an applicative similarity. By proving that Π_{Stop} with contextual equivalence conservatively extends the π-calculus without Stop and barbed testing equivalence we provided an alternative method to prove processes to be barbed testing equivalent.

Future work may investigate extensions or variants of the calculus Π_{Stop}, e.g. with (guarded) sums, or with recursion. But more importantly the results of this paper now opens easier possibilites to define and analyze embeddings of Π_{Stop} into other concurrent program calculi (for instance, the CHF-calculus [SSS12]) which also use a contextual semantics.

References

[AG97] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. In Richard Graveman, Philippe A. Janson, Clifford Neumann, and Li Gong, editors, *CCS '97*, pages 36–47. ACM, 1997.

[BD95] Michele Boreale and Rocco De Nicola. Testing equivalence for mobile processes. *Inform. and Comput.*, 120(2):279–303, 1995.

[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.

[DH84] R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoret. Comput. Sci.*, 34:83–133, 1984.

[EG04] Joost Engelfriet and Tjalling Gelsema. A new natural structural congruence in the pi-calculus with replication. *Acta Inf.*, 40(6-7):385–430, 2004.

[FG02] Cédric Fournet and Georges Gonthier. The join calculus: A language for distributed mobile programming. In *APPSEM 2000*, volume 2395 of *Lecture Notes in Comput. Sci.*, pages 268–332. Springer, 2002.

[FG05] Cédric Fournet and Georges Gonthier. A hierarchy of equivalences for asynchronous calculi. *J. Log. Algebr. Program.*, 63(1):131–173, 2005.

[Lan96] C. Laneve. On testing equivalence: May and must testing in the join-calculus. Technical Report Technical Report UBLCS 96-04, University of Bologna, 1996.

[Mil99] Robin Milner. *Communicating and Mobile Systems: the π-calculus.* Cambridge university press, 1999.

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

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, i & ii. *Inform. and Comput.*, 100(1):1–77, 1992.

[MS92] Robin Milner and Davide Sangiorgi. Barbed bisimulation. In W. Kuich, editor, *Automata, Languages and Programming*, volume 623 of *Lecture Notes in Comput. Sci.*, pages 685–695. Springer, 1992.

[NC95] V. Natarajan and Rance Cleaveland. Divergence and fair testing. In *ICALP 1995*, volume 944 of *Lecture Notes in Comput. Sci.*, pages 648–659. Springer, 1995.

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

[NSSSS07] Joachim Niehren, David Sabel, Manfred Schmidt-Schauß, and Jan Schwinghammer. Observational semantics for a concurrent lambda calculus with reference cells and futures. *Electron. Notes Theor. Comput. Sci.*, 173:313–337, 2007.

[Plo75] Gordon D. Plotkin. Call-by-name, call-by-value, and the lambda-calculus. *Theoret. Comput. Sci.*, 1:125–159, 1975.

[Pri95] Corrado Priami. Stochastic pi-calculus. *Comput. J.*, 38(7):578–589, 1995.

[PS96] Benjamin C. Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. *Math. Structures Comput. Sci.*, 6(5):409–453, 1996.

[RV07] Arend Rensink and Walter Vogler. Fair testing. *Inform. and Comput.*, 205(2):125–198, 2007.

[SSS10] Manfred Schmidt-Schauß and David Sabel. Closures of may-, should- and must-convergences for contextual equivalence. *Inform. Process. Lett.*, 110(6):232 – 235, 2010.

[SSS11] David Sabel and Manfred Schmidt-Schauß. A contextual semantics for Concurrent Haskell with futures. In *Proc. 13th international ACM SIGPLAN symposium on principles and practices of declarative programming*, PPDP '11, pages 101–112, New York, NY, USA, July 2011. ACM.

[SSS12] David Sabel and Manfred Schmidt-Schauß. Conservative concurrency in Haskell. In Nachum Dershowitz, editor, *LICS*, pages 561–570. IEEE, 2012.

[SW01] D. Sangiorgi and D. Walker. *The π-calculus: a theory of mobile processes.* Cambridge university press, 2001.

A Proofs for Barbed Testing Equivalence in Π

In this section we prove the following theorem in a series of lemmas:

Theorem A.1 (Characterizations of Barbed Testing).

- $\sqsubseteq_{c,\text{may}} = \sqsubseteq_{c,\downarrow x} = \sqsubseteq_{c,\downarrow}$
- $\sqsubseteq_{c,\text{should}} = \sqsubseteq_{c,\Downarrow x} = \sqsubseteq_{c,\Downarrow}$
- $\sqsubseteq_{c,\text{should}} \subset (\sqsubseteq_{c,\text{may}})^{-1}$ *and thus* $\square_c = \square_{c,\text{should}} = \square_{c,\Downarrow x} = \square_{c,\Downarrow}$.

Proof. The first two items will be proved in Lemmas A.4 and A.5. The inclusion in the third item will be proved in Lemma A.3, and the inclusion is strict, since e.g. $\nu u, z.(z(w).x(y) \mid z(w) \mid \overline{z}u) \sqsubseteq_{c,\text{may}} x(y)$ but $x(y) \not\sqsubseteq_{c,\text{should}} \nu u, z.(z(w).x(y) \mid z(w) \mid \overline{z}u)$ Finally, the last item follows from the three previous parts.

Lemma A.2. *It is sufficient to observe the input behavior for the testing preorders, i.e.*

- $P \sqsubseteq_{c,\text{may}} Q$ *iff* $\forall x \in \mathcal{N} : P \sqsubseteq_{c,\downarrow x} Q$
- $P \sqsubseteq_{c,\text{should}} Q$ *iff* $\forall x \in \mathcal{N} : P \sqsubseteq_{c,\Downarrow x} Q$

Proof. For the first item and the nontrivial direction, assume that $P \sqsubseteq_{c,\downarrow x} Q$ for all $x \in \mathcal{N}$ and there is a context C and a name x such that $C[P]\downarrow_{\overline{x}}$ but $C[Q] \not\downarrow_{\overline{x}}$. Let $C_1 = x(a).y(b) \mid [\cdot]$ and w.l.o.g. $a, b, y \notin (\mathsf{fn}(C[P]) \cup \mathsf{fn}(C[Q]))$ then $C_1[C[P]]\downarrow_y$ but $C_1[C[Q]] \not\downarrow_y$, but this contradicts the assumption $P \sqsubseteq_{c,\downarrow_y} Q$.

For the second item and the nontrivial direction, assume that $P \sqsubseteq_{c,\Downarrow x} Q$ for all $x \in \mathcal{N}$ and there is a context C and a name x such that $C[P] \Downarrow_{\overline{x}}$ but $C[Q]\uparrow_{\overline{x}}$. Let $C_2 = !\, x(a).y(b) \mid [\cdot]$ where $a, b, y \notin (\mathsf{fn}(C[P]) \cup \mathsf{fn}(C[Q]))$. Then $C_2[C[P]] \Downarrow_y$. Since $C[Q]\uparrow_{\overline{x}}$ there is a reduction $C[Q] \xrightarrow{sr,*} Q'$ with $Q' \not\downarrow_{\overline{x}}$. The reduction is also possible for $C_2[C[Q]]$ resulting in $Q'' = !\, x(-).y(-) \mid Q'$. Obviously $Q'' \not\downarrow_y$, since Q' cannot emit x, and thus $C_2[C[Q]]\uparrow_y$. This contradicts the assumption $P \sqsubseteq_{c,\Downarrow_y} Q$.

Lemma A.3. $\sqsubseteq_{c,\text{should}} \subseteq (\sqsubseteq_{c,\text{may}})^{-1}$ *and thus* $\sqsubseteq_{c,\text{should}} = \sqsubseteq_c$.

Proof. Due to the previous lemma it is sufficient to observe the input behavior. Let P be a process and let $C_1 = \nu z.(\overline{z}y \mid z(w).w(w') \mid \overline{x}x'.z(z')) \mid [\cdot]$ where w.l.o.g $\mathsf{fn}(P) \cap \{w, w', z, z', x'\} = \emptyset$. Then $P\downarrow_x$ iff $C_1[P]\uparrow_y$. If $P\downarrow_x$ then $C_1[P]$ can be reduced to $P'' := \nu z.(z(w).w(w')) \mid P'$ where P' is the contractum of P after receiving x' along x. Clearly, P'' cannot barb on y (i.e. $P'' \not\downarrow_x$) and thus $C_1[P]\uparrow_x$.

Now we prove the lemma: Let $P \sqsubseteq_{c,\text{should}} Q$, $C[Q]\downarrow_x$, but $C[P] \not\downarrow_x$. From $P \sqsubseteq_{c,\text{should}} Q$ it follows $Q \leq_{\uparrow_y} P$. But $C_1[C[Q]]\uparrow_y$ while $C_1[C[P]] \Downarrow_y$ which is a contradiction.

Lemma A.4. *Let $x \in \mathcal{N}$. Then*

- $P \sqsubseteq_{c,\text{may}} Q$ *iff* $P \sqsubseteq_{c,\downarrow x} Q$
- $P \sqsubseteq_{c,\text{should}} Q$ *iff* $P \sqsubseteq_{c,\Downarrow x} Q$.

Proof. For the nontrivial part of the first item let $P \sqsubseteq_{c,\downarrow x} Q$, $C[P]\downarrow_y$, but $C[Q] \not\downarrow_y$ Then for $C_1 = \nu a, b.(\overline{y}a.x(b)) \mid \nu\mathcal{X}.[\cdot]$ where $\mathcal{X} = (\mathsf{fn}(C[P]) \cup \mathsf{fn}(C[Q])) \setminus \{y\}$ we have $C_1[C[P]]\downarrow_x$, but $C_1[C[Q]] \not\downarrow_x$ which contradicts $P \sqsubseteq_{c,\downarrow x} Q$.

For the nontrivial part of the second item let $P \sqsubseteq_{c,\Downarrow x} Q$, $C[P] \Downarrow_y$, but $C[Q]\uparrow_y$. Then for $C_2 = \nu a, b.(\overline{y}a.x(b)) \mid \nu\mathcal{X}.[\cdot]$ where $\mathcal{X} = (\mathsf{fn}(C[P]) \cup \mathsf{fn}(C[Q])) \setminus \{y\}$ we have $C_2[C[P]] \Downarrow_x$, but $C_2[C[Q]]\uparrow_x$ which contradicts $P \sqsubseteq_{c,\Downarrow x} Q$.

Lemma A.5. $\sqsubseteq_{c,\text{may}} = \sqsubseteq_{c,\downarrow}$ *and* $\sqsubseteq_{c,\text{should}} = \sqsubseteq_{c,\Downarrow}$

Proof. For the nontrivial direction of the first part, we show that $\sqsubseteq_{c,\downarrow_x} \supseteq \sqsubseteq_{c,\downarrow}$ which is sufficient due to the previous lemma.

Let $P \sqsubseteq_{c,\downarrow} Q$, and $C[P]\downarrow_x$. Let $C_1 = \nu\mathcal{X}.[\cdot]$ where $\mathcal{X} = (\mathsf{fn}(C[P]) \cup \mathsf{fn}(C[Q])) \setminus \{x\}$. Then $C_1[C[P]]\downarrow_x$ and since $P \sqsubseteq_{c,\downarrow} Q$ we have $C_1[C[Q]]\downarrow_{x'}$. However, since x is the only free name in $C_1[C[Q]]$ the equation $x = x'$ must hold and thus also $C_1[C[Q]]\downarrow_x$ which implies $C[Q]\downarrow_x$, since C_1 only hides names.

For the second item the reasoning is completely analogous.

B Proofs for the Calculus Π_{Stop}

Lemma B.1. *For all processes P, Q it holds:*

1. *If $P \xrightarrow{sr} Q$ then $\nu x.P \xrightarrow{sr} \nu x.Q$*
2. *If $\nu x.P \xrightarrow{sr} Q$ then $P \xrightarrow{sr} Q'$ such that either $Q \equiv \nu x.Q'$ or $Q \equiv Q'$*
3. *For $\xi \in \{\downarrow, \Downarrow, \uparrow, \Uparrow\}$: $P \leq_\xi \nu x.P$ and $\nu x.P \leq_\xi P$.*

Proof. (1) holds, since $\nu x.D \in \mathcal{D}$ whenever $D \in \mathcal{D}$. (2) follows by induction on the number of congruence axiom applications using the facts that the *ia*-reduction does not depend on ν-binders and that ν-binders can be reintroduced by structural congruence if they were deleted by reduction. For (3) clearly P is successful iff $\nu x.P$ is successful. Hence, by induction on the length of a given reduction sequence ending in a successful process and the first two items we can show $P \leq_\downarrow \nu x.P$ and $\nu x.P \leq_\downarrow P$. By Lemma 3.4 this also shows $P \Uparrow \implies \nu x.P \Uparrow$ and $\nu x.P \Uparrow \implies P \Uparrow$. Now by induction on the length of a reduction sequence ending in a must-divergent process using the first two items and using the last shown implications as a base case we can verify that $P \leq_\uparrow \nu x.P$ and $\nu x.P \leq_\uparrow P$ which is equivalent to $\nu x.P \leq_\Downarrow P$ and $P \leq_\Downarrow \nu x.P$ (Lemma 3.4).

Lemma B.2. *Let $P, Q \in Proc_{\mathsf{Stop}}$. If $\sigma(P) \mid R \leq_\downarrow \sigma(Q) \mid R$ for all $\sigma \in \Sigma$ and $R \in Proc_{\mathsf{Stop}}$, then also $\sigma(C[P]) \mid R \leq_\downarrow \sigma(C[Q]) \mid R$ for $C \in \{[\cdot] \mid S, S \mid [\cdot], C = \nu x.[\cdot], x(y).[\cdot], C = \overline{x}y.[\cdot] \mid x, y \in \mathcal{N}, S \in Proc_{\mathsf{Stop}}\}$, all $\sigma \in \Sigma$ and $R \in Proc_{\mathsf{Stop}}$.*

Proof. 1. $C = [\cdot] \mid S$: Then $\sigma(C[P]) \mid R \equiv (\sigma(P) \mid R')$ and $\sigma(C[Q]) \mid R \equiv \sigma(Q) \mid R'$ with $R' = \sigma(S) \mid R$. The precondition of the claim implies that $\sigma(P) \mid R' \leq_\downarrow \sigma(Q) \mid R'$ and thus Proposition 3.5 shows $\sigma(C[P]) \mid R \leq_\downarrow \sigma(C[Q]) \mid R$.

2. $C = S \mid [\cdot]$: The claim follows from the previous item and Proposition 3.5.
3. $C = \nu x.[\cdot]$: Since $\sigma(P) \mid R \leq_\downarrow \sigma(Q) \mid R$ holds by the precondition of the claim, Lemma 3.6 shows the claim.
4. $C = x(y).[\cdot]$: Let $\sigma(C[P]) \mid R \xrightarrow{sr,n} P_n$ where P_n is successful. We use induction on n. The base case $n = 0$ holds, since in this case R must be successful, and thus $\sigma(C[Q]) \mid R$ is successful, too. For the induction step assume $\sigma(x) = x_1$ and w.l.o.g. $\sigma(y) = y$. Let $x_1(y).\sigma(P) \mid R \xrightarrow{sr} \nu\mathcal{X}.\sigma(P)[z/y] \mid R'$ be the first reduction step of the reduction sequence, where $\mathcal{X} \subseteq \{z\}$. The same reduction step for $\sigma(x(y).Q) \mid R$ results in $\nu\mathcal{X}.\sigma'(Q)[z/y] \mid R'$. By induction assumption, the lemma holds for the pair $\sigma(P)[z/y]$ and $\sigma(Q)[z/y]$, and by item (3) also for extending it with ν.
5. $C = \overline{x}y.[\cdot]$: This case is similar to the previous item. □

Lemma B.3. *For P, Q, S_{xy}, S_{yx} as defined in Proposition 4.9: $P \leq_{c,\downarrow} Q$.*

Proof. Let $\mathcal{S} := \mathcal{S}_1 \cup \mathcal{S}_2 \cup \precsim_\downarrow$ where

$$\begin{aligned}\mathcal{S}_1 := \{&(!\, S_{xy} \mid !\, S_{yx} \mid R[x/w] \mid \overline{y}u_1 \mid \ldots \mid \overline{y}u_n,\\ &\quad !\, S_{xy} \mid !\, S_{yx} \mid R[y/w] \mid \overline{x}u_1 \mid \ldots \mid \overline{x}u_n)\\ &\quad \mid \text{ for any } R, \text{ any } x, y, w, u_i, \text{ and any } n \geq 0\}\\ \mathcal{S}_2 := \{&(\sigma(P) \mid R, \sigma(Q) \mid R) \mid \text{ for any } R \text{ and } \sigma\}\end{aligned}$$

For proving $P \leq_{c,\downarrow} Q$ it suffices to show that $\mathcal{S}$ is $F_{\downarrow}$-dense: This implies $\sigma(P) \mid R \leq_{\downarrow} \sigma(Q) \mid R$ for all R, σ and thus the context lemma (Theorem 4.3) shows $P \leq_{c,\downarrow} Q$.

First let $(P_1, P_2) \in \mathcal{S}_1$. If P_1 is successful, then clearly also P_2 is successful and thus $P_2 \downarrow$. If $P_1 \xrightarrow{sr} P_1'$, then there are following cases:

- If the redex is inside $R[x/w]$, then either the same reduction can also be performed for P_2, then $P_2 \xrightarrow{sr} P_2'$ and $(P_1', P_2') \in \mathcal{S}$, or the name x occurs in R. We consider two cases, where we use the abbreviations $L_x := \overline{x}u_1 \mid \ldots \mid \overline{x}u_n$ and $L_y := \overline{y}u_1 \mid \ldots \mid \overline{y}u_n$:
  - If $R = \nu\mathcal{W}.(w(z').R_1 \mid \overline{x}v.R_2 \mid R_3)$ and $P_1' = !\,S_{xy} \mid !\,S_{yx} \mid \nu\mathcal{W}.(R_1[v/z'] \mid R_2 \mid R_3)[x/w] \mid L_y$, then $P_2 \xrightarrow{sr} P_2' \xrightarrow{sr} P_2''$ with

$$\begin{aligned} P_2 = \;& !\,S_{xy} \mid !\,S_{yx} \mid L_x \\ & \mid \nu\mathcal{W}.(w(z').R_1 \mid \overline{x}v.R_2 \mid R_3)[y/w] \\ P_2'' := \;& !\,S_{xy} \mid !\,S_{yx} \mid L_x \\ & \mid \nu\mathcal{W}.(R_1[v/z'] \mid R_2 \mid R_3))[y/w], \end{aligned}$$

    since $\overline{x}v.R_2 \mid S_{xy} \xrightarrow{sr} R_2 \mid \overline{y}v$. Now $(P_1', P_2'') \in \mathcal{S}$ and thus we are finished.
  - If $R = \nu\mathcal{W}.(x(z').R_1 \mid \overline{w}v.R_2 \mid R_3)$ and $P_1' = !\,S_{xy} \mid !\,S_{yx} \mid \nu\mathcal{W}.(R_1[v/z'] \mid R_2 \mid R_3)[x/w] \mid L_y$, then $P_2 \xrightarrow{sr} P_2' \xrightarrow{P_2''}$ with

$$\begin{aligned} P_2 = \;& !\,S_{xy} \mid !\,S_{yx} \mid L_x \\ & \nu\mathcal{W}.(x(z').R_1 \mid \overline{w}v.R_2 \mid R_3)[y/w] \\ P_2'' := \;& !\,S_{xy} \mid !\,S_{yx} \mid L_x \\ & \mid \nu\mathcal{W}.(R_1[v/z'] \mid R_2 \mid R_3))[y/w], \end{aligned}$$

    since $\overline{y}v.R_2 \mid S_{yx} \xrightarrow{sr} R_2 \mid \overline{x}v$ and thus $(P_1', P_2'') \in \mathcal{S}$.
- The redex is $S_{yx} \mid \overline{y}u_i$, i.e. for $L_y = \overline{y}u_1 \mid \ldots \overline{y}u_{i-1} \mid \overline{y}u_{i+1} \mid \ldots \overline{y}u_n$:
  $P_1 = !\,S_{xy} \mid !\,S_{yx} \mid R[x/w] \mid \overline{y}u_i \mid L_y \xrightarrow{sr} !\,S_{xy} \mid !\,S_{yx} \mid R[x/w] \mid \overline{x}u_i \mid L_y \equiv !\,S_{xy} \mid !\,S_{yx} \mid (R \mid \overline{w}u_i)[x/w] \mid L_y = P_1'$.
  Then for $L_x := \overline{x}u_1 \mid \ldots \overline{x}u_{i-1} \mid \overline{x}u_{i+1} \mid \ldots \overline{x}u_n$:
  $P_2 = !\,S_{xy} \mid !\,S_{yx} \mid R[y/w] \mid \overline{x}u_i \mid L_x \xrightarrow{sr} !\,S_{xy} \mid !\,S_{yx} \mid R[y/w] \mid \overline{y}u_i \mid L_x \equiv !\,S_{xy} \mid !\,S_{yx} \mid (R \mid \overline{w}u_i)[y/w] \mid L_x = P_2'$ and thus $(P_1', P_2') \in \mathcal{S}$.
- The redex is $S_{xy} \mid R[x/w]$, i.e. $R = \overline{w}v.R'$ and for $L_y := \overline{y}u_1 \mid \ldots \mid \overline{y}u_n$:
  $P_1 = !\,S_{xy} \mid !\,S_{yx} \mid \overline{x}v.R'[x/w] \mid L_y \xrightarrow{sr} !\,S_{xy} \mid !\,S_{yx} \mid R'[x/w] \mid \overline{y}v \mid L_y = P_1'$. Then for $L_x := \overline{x}u_1 \mid \ldots \mid \overline{x}u_n$: $P_2 = !\,S_{xy} \mid !\,S_{yx} \mid \overline{y}v.R'[y/w] \mid L_x \xrightarrow{sr} !\,S_{xy} \mid !\,S_{yx} \mid R'[y/w] \mid \overline{x}v \mid L_x = P_2'$ and thus $(P_1', P_2') \in \mathcal{S}$.

Now let $(P_1, P_2) \in \mathcal{S}_2$ and let $a' = \sigma(a), x' = \sigma(x), y' = \sigma(y), z' = \sigma(z)$. If P_1 is successful, then P_2 is successful. If $P_1 \xrightarrow{sr} P_1'$, then there are the cases:

- The redex is inside R, then $P_2 \xrightarrow{sr} P_2'$ and $(P_1', P_2') \in \mathcal{S}$.
- $R = \nu\mathcal{W}.(a'(w).R' \mid R'')$ and $P_1 \xrightarrow{sr} !\,\sigma(S_{xy}) \mid !\,\sigma(S_{yx}) \mid \nu\mathcal{W}.(R'[x/w] \mid R'') \equiv !\,\sigma(S_{xy}) \mid !\,\sigma(S_{yx}) \mid \nu\mathcal{W}.(R' \mid R'')[x/w] = P_1'$ where the last congruence step is possible, since we may assume that w was renamed fresh for R''.
  Then $P_2 \xrightarrow{sr} !\,\sigma(S_{xy}) \mid !\,\sigma(S_{yx}) \mid \nu\mathcal{W}.(R'[y/w] \mid R'') \equiv !\,\sigma(S_{xy}) \mid !\,\sigma(S_{yx}) \mid \nu\mathcal{W}.(R' \mid R'')[y/w] = P_2'$ and $(P_1', P_2') \in \mathcal{S}$.
- $R = \nu\mathcal{W}.(\overline{x'}u.R' \mid R'')$ and $P_1 \xrightarrow{sr} \overline{a'}x' \mid !\,\sigma(S_{xy}) \mid !\,\sigma(S_{yx}) \mid (R' \mid R) \mid \overline{y'}u = P_1'$. Then $P_2 \xrightarrow{sr} \overline{a'}y' \mid !\,\sigma(S_{xy}) \mid !\,\sigma(S_{yx}) \mid (R' \mid R) \mid \overline{y'}u = P_2'$. and $(P_1', P_2') \in \mathcal{S}$.
- $R = \nu\mathcal{W}.(\overline{y'}u.R' \mid R'')$ and $P_1 \xrightarrow{sr} \overline{a'}x' \mid !\,\sigma(S_{xy}) \mid !\,\sigma(S_{yx}) \mid (R' \mid R) \mid \overline{x'}u = P_1'$. Then $P_2 \xrightarrow{sr} \overline{a'}y' \mid !\,\sigma(S_{xy}) \mid !\,\sigma(S_{yx}) \mid (R' \mid R) \mid \overline{x'}u = P_2'$ and $(P_1', P_2') \in \mathcal{S}$. □

Lemma B.4. *The relation*

$$\mathcal{S} := \left\{ (\nu\mathcal{X}.(P \mid R), \nu\mathcal{X}.(Q \mid R)) \;\middle|\; \begin{array}{l} P \precsim_{b,\downarrow} Q, \\ \textit{for any } \mathcal{X},\ R \end{array} \right\} \cup \precsim_{\downarrow}.$$

is $F_{\downarrow}$-dense.

Proof. Let $(\nu\mathcal{X}.(P \mid R), \nu\mathcal{X}.(Q \mid R)) \in \mathcal{S}$. We have to show $(\nu\mathcal{X}.(P \mid R), \nu\mathcal{X}.(Q \mid R)) \in F_{\downarrow}(\mathcal{S})$.

If $\nu\mathcal{X}.(P \mid R)$ is successful, then P or R is successful too, and thus either $Q \downarrow$ and so does $\nu\mathcal{X}.Q \mid R$ or $\nu\mathcal{X}.(Q \mid R)$ is already successful.

For $\nu\mathcal{X}.(P \mid R) \xrightarrow{sr} P_1$ we show that $\nu\mathcal{X}.(Q \mid R) \xrightarrow{sr,*} Q_1$, s.t. $(P_1, Q_1) \in \mathcal{S}$:

If the redex of $\nu\mathcal{X}.(P \mid R) \xrightarrow{sr} P_1$ is inside P, i.e. $P_1 = \nu\mathcal{X}.(P' \mid R)$, then by $P \precsim_{b,\downarrow} Q$ there exists Q' with $Q \xrightarrow{sr,*} Q'$, $P' \precsim_{b,\downarrow} Q'$. Since also $\nu\mathcal{X}.(Q \mid R) \xrightarrow{sr,*} \nu\mathcal{X}.(Q' \mid R)$ and thus $(\nu\mathcal{X}.(P' \mid R), \nu\mathcal{X}.(Q' \mid R)) \in \mathcal{S}$, this case is finished.

If the redex of $\nu\mathcal{X}.(P \mid R) \xrightarrow{sr} P_1$ is inside R, i.e. $P_1 = \nu\mathcal{X}.(P \mid R')$ then also $\nu\mathcal{X}.(Q \mid R) \xrightarrow{sr} \nu\mathcal{X}.(Q \mid R')$ and $(\nu\mathcal{X}.(P \mid R'), \nu\mathcal{X}.(Q \mid R')) \in \mathcal{S}$.

The remaining cases are that the redex uses parts of P and parts of R.

- If $P \equiv \nu\mathcal{X}_1.(x(y).P' \mid P'')$, $R \equiv \nu\mathcal{X}_2.(\overline{x}z.R' \mid R'')$ with $z \notin \mathcal{X}_2$ and $\nu\mathcal{X}.(P \mid R) \xrightarrow{sr} \nu\mathcal{X}.(\nu\mathcal{X}_1.(P'[z/y] \mid P'') \mid \nu\mathcal{X}_2.R' \mid R'') =: P_1$, then by $P \precsim_{b,\downarrow} Q$ there exists Q_0 with $Q \xrightarrow{sr,*} Q_0$, $Q_0 = \nu\mathcal{Y}_1.(x(y).Q' \mid Q'')$ s.t. $\mathcal{X}_1.(P'[z/y] \mid P'') \precsim_{b,\downarrow} \nu\mathcal{Y}_1.(Q'[z/y] \mid Q'')$. Since $\nu\mathcal{X}.(Q \mid R) \xrightarrow{sr,*} \nu\mathcal{X}.(Q_0 \mid R) \xrightarrow{sr} \nu\mathcal{Y}_1.(Q'[z/y] \mid Q'') \mid \nu\mathcal{X}_2.R' \mid R'') = Q_1$, also $(P_1, Q_1) \in \mathcal{S}$.
- If $P \equiv \nu\mathcal{X}_1.(x(y).P' \mid P'')$, $R \equiv \nu z, \mathcal{X}_2.(\overline{x}z.R' \mid R'')$ and $\nu\mathcal{X}.(P \mid R) \xrightarrow{sr} \nu\mathcal{X}.(\nu z.(\nu\mathcal{X}_1.P'[z/y] \mid P'' \mid \nu\mathcal{X}_2.(R' \mid R''))) = P_1$, then by $P \precsim_{b,\downarrow} Q$ there exists Q_0 with $Q \xrightarrow{sr,*} Q_0$, $Q_0 = \nu\mathcal{Y}_1.(x(y).Q' \mid Q'')$ s.t. $\nu\mathcal{X}_1.(P'[z/y] \mid P'') \precsim_{b,\downarrow} \nu\mathcal{Y}_1.(Q'[z/y] \mid Q'')$. Since also $\nu\mathcal{X}.(Q \mid R) \xrightarrow{sr,*} \nu\mathcal{X}.(Q_0 \mid R) \xrightarrow{sr} \nu\mathcal{X}.(\nu z.(\nu\mathcal{Y}_1.(Q'[z/y] \mid Q'') \mid \nu\mathcal{X}_2.(R' \mid R''))) =: Q_1$ we have $(P_1, Q_1) \in \mathcal{S}$.
- If $P \equiv \nu\mathcal{X}_1.(\overline{x}y.P' \mid P'')$ and $R \equiv \nu\mathcal{X}_2.(x(z).R' \mid R'')$ with $y \notin \mathcal{X}_1$ and $\nu\mathcal{X}.(P \mid R) \xrightarrow{sr} \nu\mathcal{X}.(\nu\mathcal{X}_1.P' \mid P'') \mid \nu\mathcal{X}_2.(R'[y/z] \mid R'') =: P_1$, then by $P \precsim_{b,\downarrow} Q$ there exists Q_0 with $Q \xrightarrow{sr,*} Q_0$, $Q_0 = \nu\mathcal{Y}_1.(\overline{x}y.Q' \mid Q'')$ where $y \notin \mathcal{Y}_1$ s.t. $\nu\mathcal{X}_1.(P' \mid P'') \precsim_{b,\downarrow} \mathcal{Y}_1.(Q' \mid Q'')$. Since also $\nu\mathcal{X}.(Q \mid R) \xrightarrow{sr,*} \nu\mathcal{X}.(Q_0 \mid R) \xrightarrow{sr} \nu\mathcal{X}.(\nu\mathcal{Y}_1.(Q' \mid Q'') \mid \nu\mathcal{X}_2.(R'[y/z] \mid R'')) =: Q_1$, we have $(P_1, Q_1) \in \mathcal{S}$.
- If $P \equiv \nu y.\nu\mathcal{X}_1.(\overline{x}y.P' \mid P'')$, $R \equiv \nu\mathcal{X}_2.(x(z).R' \mid R'')$, and $\nu\mathcal{X}.(P \mid R) \xrightarrow{sr} \nu\mathcal{X}.\nu y.(\nu\mathcal{X}_1.(P' \mid P'') \mid \nu\mathcal{X}_2.(R'[y/z] \mid R'')) =: P_1$, then by $P \precsim_{b,\downarrow} Q$ there exists Q_0 with $Q \xrightarrow{sr,*} Q_0$, $Q_0 = \nu y.\nu\mathcal{Y}_1.(\overline{x}y.Q' \mid Q'')$ s.t. $\nu\mathcal{X}_1.(P' \mid P'') \precsim_{b,\downarrow} \nu\mathcal{Y}_1.(Q' \mid Q'')$. Since also $\nu\mathcal{X}.\nu y.(Q \mid R) \xrightarrow{sr,*} \nu\mathcal{X}.\nu y.(Q_0 \mid R) \xrightarrow{sr} \nu\mathcal{X}.\nu y.(\nu\mathcal{Y}_1.(Q' \mid Q'') \mid \nu\mathcal{X}_2.(R'[y/z] \mid R'')) =: Q_1$, we have $(P_1, Q_1) \in \mathcal{S}$. □

Lemma B.5. *The relation*

$$\mathcal{S} := \left\{ (\nu\mathcal{X}.(P \mid R), \nu\mathcal{X}.(Q \mid R)) \;\middle|\; \begin{array}{l} P \precsim_{b,\uparrow} Q, \\ \textit{for any } \mathcal{X},\ R \end{array} \right\} \cup \precsim_{\uparrow}$$

is $F_{\uparrow}$-dense.

Proof. Note that if $P \precsim_{b,\uparrow} Q$, then $Q \precsim_{b,\downarrow} P$.

Now let $(\nu\mathcal{X}.(P \mid R), \nu\mathcal{X}.(Q \mid R)) \in \mathcal{S}$. We have to show that $(\nu\mathcal{X}.(P \mid R), \nu\mathcal{X}.(Q \mid R)) \in F_{\uparrow}(\mathcal{S})$.

If $\nu\mathcal{X}.(P \mid R) \Uparrow$, then $Q \precsim_{b,\downarrow} P$ and Proposition 4.13 show that $\nu\mathcal{X}.(Q \mid R) \precsim_{\downarrow} \nu\mathcal{X}.(P \mid R)$ which implies that $\nu\mathcal{X}.(P \mid R) \leq_{\Uparrow} \nu\mathcal{X}.(Q \mid R)$ and thus $\nu\mathcal{X}.(Q \mid R) \downarrow$.

If $\nu\mathcal{X}.(P \mid R) \xrightarrow{sr} P_1$, then we have to show that $\nu\mathcal{X}.(Q \mid R) \xrightarrow{sr,*} Q_1$, s.t. $(P_1, Q_1) \in \mathcal{S}$.

If the redex of $\nu\mathcal{X}.(P \mid R) \xrightarrow{sr} P_1$ is inside P, i.e. $\nu\mathcal{X}.(P \mid R) \xrightarrow{sr} \nu\mathcal{X}.(P' \mid R)$ then $P \precsim_{b,\uparrow} Q$ shows that there exists Q' with $Q \xrightarrow{sr,*} Q'$ and $P' \precsim_{b,\uparrow} Q'$. Since also $\nu\mathcal{X}.(Q \mid R) \xrightarrow{sr,*} \nu\mathcal{X}.(Q' \mid R)$ and thus $(\nu\mathcal{X}.(P' \mid R), \nu\mathcal{X}.(Q' \mid R)) \in \mathcal{S}$ this case is finished.

If the redex of $\nu\mathcal{X}.(P \mid R) \xrightarrow{sr} P_1$ is inside R, i.e. $\nu\mathcal{X}.(P \mid R) \xrightarrow{sr} \nu\mathcal{X}.(P \mid R')$ then also $\nu\mathcal{X}.(Q \mid R) \xrightarrow{sr} \nu\mathcal{X}.(Q \mid R')$ and thus $(\nu\mathcal{X}.(P \mid R'), \nu\mathcal{X}.(Q \mid R')) \in \mathcal{S}$ and this case is finished.

It remains to consider the cases where the redex uses parts of P and parts of R.

- If $P \equiv \nu\mathcal{X}_1.(x(y).P' \mid P'')$, $R \equiv \nu\mathcal{X}_2.(\overline{x}z.R' \mid R'')$ with $z \notin \mathcal{X}_2$ and $\nu\mathcal{X}.(P \mid R) \xrightarrow{sr} \nu\mathcal{X}.(\nu\mathcal{X}_1.(P'[z/y] \mid P'') \mid \nu\mathcal{X}_2.R' \mid R'') = P_1$ then $P \precsim_{b,\uparrow} Q$ shows that there exists Q_0 with $Q \xrightarrow{sr,*} Q_0$ and $Q_0 = \nu\mathcal{Y}_1.(x(y).Q' \mid Q'')$ s.t. $\mathcal{X}_1.(P'[z/y] \mid P'') \precsim_{b,\uparrow} \nu\mathcal{Y}_1.(Q'[z/y] \mid Q'')$ Since $\nu\mathcal{X}.(Q \mid R) \xrightarrow{sr,*} \nu\mathcal{X}.(Q_0 \mid R) \xrightarrow{sr} \nu\mathcal{Y}_1.(Q'[z/y] \mid Q'') \mid \nu\mathcal{X}_2.R' \mid R'') = Q_1$ this shows $(P_1, Q_1) \in \mathcal{S}$.
- If $P \equiv \nu\mathcal{X}_1.(x(y).P' \mid P'')$, $R \equiv \nu z, \mathcal{X}_2.(\overline{x}z.R' \mid R'')$ and $\nu\mathcal{X}.(P \mid R) \xrightarrow{sr} \nu\mathcal{X}.(\nu z.(\nu\mathcal{X}_1.P'[z/y] \mid P'' \mid \nu\mathcal{X}_2.(R' \mid R''))) = P_1$ then $P \precsim_{b,\uparrow} Q$ shows that there exists Q_0 with $Q \xrightarrow{sr,*} Q_0$ and $Q_0 = \nu\mathcal{Y}_1.(x(y).Q' \mid Q'')$ s.t. $\nu\mathcal{X}_1.(P'[z/y] \mid P'') \precsim_{b,\uparrow} \nu\mathcal{Y}_1.(Q'[y/z] \mid Q'')$. Since also $\nu\mathcal{X}.(Q \mid R) \xrightarrow{sr,*} \nu\mathcal{X}.(Q_0 \mid R) \xrightarrow{sr} \nu\mathcal{X}.(\nu z.(\nu\mathcal{Y}_1.(Q'[z/y] \mid Q'') \mid \nu\mathcal{X}_2.(R' \mid R''))) = Q_1$ we have $(P_1, Q_1) \in \mathcal{S}$.
- If $P \equiv \nu\mathcal{X}_1.(\overline{x}y.P' \mid P'')$ and $R \equiv \nu\mathcal{X}_2.(x(z).R' \mid R'')$ with $y \notin \mathcal{X}_1$ and $\nu\mathcal{X}.(P \mid R) \xrightarrow{sr} \nu\mathcal{X}.(\nu\mathcal{X}_1.P' \mid P'') \mid \nu\mathcal{X}_2.(R'[y/z] \mid R'') = P_1$, then $P \precsim_{b,\uparrow} Q$ shows that there exists Q_0 with $Q \xrightarrow{sr,*} Q_0$ and $Q_0 = \nu\mathcal{Y}_1.(y.Q' \mid Q'')$ where $y \notin \mathcal{Y}_1$ s.t. $\nu\mathcal{X}_1.(P' \mid P'') \precsim_{b,\uparrow} \mathcal{Y}_1.(Q' \mid Q'')$. Since also $\nu\mathcal{X}.(Q \mid R) \xrightarrow{sr,*} \nu\mathcal{X}.(Q_0 \mid R) \xrightarrow{sr} \nu\mathcal{X}.(\nu\mathcal{Y}_1.(Q' \mid Q'') \mid \nu\mathcal{X}_2.(R'[y/z] \mid R'')) = Q_1$, we have $(P_1, Q_1) \in \mathcal{S}$.
- If $P \equiv \nu y, \mathcal{X}_1.(\overline{x}y.P' \mid P'')$, $R \equiv \nu\mathcal{X}_2.(x(z).R' \mid R'')$, and $\nu\mathcal{X}.(P \mid R) \xrightarrow{sr} \nu\mathcal{X}.\nu y.(\nu\mathcal{X}_1.(P' \mid P'') \mid \nu\mathcal{X}_2.(R'[y/z] \mid R'')) = P_1$, then $P \precsim_{b,\uparrow} Q$ shows that there exists Q_0 with $Q \xrightarrow{sr,*} Q_0$ and $Q_0 = \nu y.\nu\mathcal{Y}_1.(y.Q' \mid Q'')$ s.t. $\nu\mathcal{X}_1.(P' \mid P'') \precsim_{b,\uparrow} \nu\mathcal{Y}_1.(Q' \mid Q'')$. Since also $\nu\mathcal{X}.\nu y.(Q \mid R) \xrightarrow{sr,*} \nu\mathcal{X}.\nu y.(Q_0 \mid R) \xrightarrow{sr} \nu\mathcal{X}.\nu y.(\nu\mathcal{Y}_1.(Q' \mid Q'') \mid \nu\mathcal{X}_2.(R'[y/z] \mid R'')) = Q_1$, we have $(P_1, Q_1) \in \mathcal{S}$. □