RAIRO-Theor. Inf. Appl. 53 (2019) 153–206
https://doi.org/10.1051/ita/2019005

RAIRO - Theoretical Informatics and Applications
www.rairo-ita.org

PARAMETERIZING HIGHER-ORDER PROCESSES ON NAMES AND PROCESSES*,**

Xian Xu***

Abstract. Parameterization extends higher-order processes with the capability of abstraction and application (like those in lambda-calculus). As is well-known, this extension is strict, meaning that higher-order processes equipped with parameterization are strictly more expressive than those without parameterization. This paper studies strictly higher-order processes (*i.e.*, no name-passing) with two kinds of parameterization: one on names and the other on processes themselves. We present two main results. One is that in presence of parameterization, higher-order processes can interpret first-order (name-passing) processes in a quite elegant fashion, in contrast to the fact that higher-order processes without parameterization cannot encode first-order processes at all. We present two such encodings and analyze their properties in depth, particularly full abstraction. In the other result, we provide a simpler characterization of the standard context bisimilarity for higher-order processes with parameterization, in terms of the normal bisimilarity that stems from the well-known normal characterization for higher-order calculus. As a spinoff, we show that the bisimulation up-to context technique is sound in the higher-order setting with parameterization.

Mathematics Subject Classification. 68Q05, 68Q10, 68Q85.

Received August 23, 2018. Accepted November 15, 2019.

1. Introduction

In concurrent systems, higher-order means that processes communicate by means of process-passing (*i.e.*, program-passing), whereas first-order means that processes communicate through name-passing (*i.e.*, reference-passing). Parameterization originates from lambda-calculus (that is itself of higher-order nature), and enables processes, in a concurrent setting, to do abstraction and application in a way similar to that of lambda-calculus.

* A preliminary version of this work was presented at EXPRESS/SOS 2016. This paper revises and extends that version with full-fledged proofs and more refined discussions (at least more than half new materials), and moreover, the detailed analysis of a variant encoding of interest. This encoding, mentioned only as a further direction in the preliminary version, is given thorough examination in this work.

** This work has been supported by NSF of China (61872142, 61772336, 61772200, 61572318, 61472239, 61261130589) and project ANR 12IS02001 'PACE', and partially supported by Shanghai Municipal NSF (17ZR1406900).

Keywords and phrases: Parameterization, encoding, context bisimulation, normal bisimulation, higher-order, first-order, processes.

East China University of Science and Technology, Shanghai 200237, PR China.

*** Corresponding author: xuxian2004@gmail.com

Article published by EDP Sciences

Say P is a higher-order process, then an abstraction $\langle U\rangle P$ means abstracting the variable U in P to obtain somewhat a function (like $\lambda U.P$ in terms of lambda-calculus), and correspondingly an application $(\langle U\rangle P)\langle K\rangle$ means passing process K to the abstraction and obtaining an instantiation $P\{K/U\}$ (*i.e.*, replacing each variable U in P with K, like $(\lambda U.P)K$ in terms of lambda-calculus). There are basically two kinds of parameterization: parameterization on names and parameterization on processes. In the former, U is a name variable and K is an instance name. In the latter, U is a process variable and K is an instance process. Parameterization is a natural way to extend the capacity of higher-order processes and this extension is strict, that is, the expressiveness strictly increases with the help of parameterization [14]. In this paper, we study higher-order processes in presence of parameterization.

Comparison between higher-order and first-order processes is a frequent topic in concurrency theory. Such comparison, for example, asks whether higher-order processes can correctly express first-order processes, or vice versa. It is well known that first-order processes can elegantly encode higher-order processes [24, 31]; the converse is however not quite the case. As the first issue, this paper addresses how to encode first-order processes with higher-order processes enhanced with parameterization.

The very early work on using higher-order process to interpret first-order ones is contributed by Thomsen [33], who proposes a prototype encoding of first-order processes with higher-order processes with the relabelling operator (the same as that in CCS [20]). This encoding employs a gadget called wire to mimic the function of a name in the higher-order setting, and makes essential use of the relabelling to guarantee that the wires work properly so as to fulfill the role of names. Thomsen establishes some basic operational correspondence between the reductions of a first-order process and its encoding. However, due to the arbitrary ability of relabelling to change names (*e.g.*, from global to local), the encoding is very hard to analyze for full abstraction. Roughly, full abstraction means that the first-order processes are equivalent if and only if their encodings are, with the 'if only' direction called soundness and the other direction completeness. Unfortunately, without the relabelling operator, the basic higher-order process (which has the elementary operators including input, output, parallel composition and restriction) is not capable of encoding first-order processes [35]. In the literature, several variants of higher-order processes are exploited to encode first-order processes. In [31], an asynchronous higher-order calculus with parameterization on names is used to compile the asynchronous localized π-calculus, a somewhat lightweight variant of the first-order π-calculus [21]. The concept of asynchrony means that the output is non-blocking. This encoding relies heavily on the output capability that is the only way a received name can be used in the localized π-calculus, and is proven to be fully abstract with respect to barbed congruence. In [37], we explore the encoding of the full π-calculus using higher-order processes with parameterization on names. To that end, we construct an encoding that harnesses the idea of Thomsen's encoding and show that it is complete. This inspires the present work. In [1], Bundgaard *et al.* use the HOMER to translate the name-passing π-calculus. This translation is possible because a HOMER process can, in a way quite different from parameterization, modify names in the continuation processes (resources), and this allows flexibility so that names can be communicated in an intermediate fashion. In [12], Kouzapas *et al.* propose fully abstract encodings concerning first-order processes and session typed higher-order processes. Their encodings use session types to govern communications and show that in the context of session types, first-order and higher-order processes are equally expressive. This work is well related to those mentioned above and that in this paper, though the context is quite different (*i.e.*, session typed processes).

Despite the extensive research on encoding first-order processes with (variant) higher-order processes, the following question has remained open: *Is there an encoding of first-order processes by the higher-order processes with the capability of parameterization, in a way at least as elegant as that from higher-order processes to first order ones?*

This question is important in two respects. One is that parameterization somewhat brings about the core of lambda-calculus to higher-order concurrency (actually they can indeed encode lambda-calculus [24, 31]). So it makes sense for such an extended higher-order calculus to be able to express first-order pi-calculus. To this end, albeit we have known that higher-order processes are Turing complete even only with its core (*i.e.*, only the higher-order input and output mechanism) [15], we have not been aware of a compositional encoding of first-order pi-calculus in that setting (by 'compositional', we mean that the encoding strategy somehow preserves the

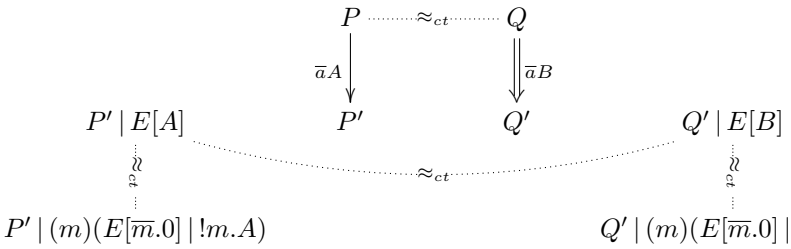
structure of the processes being encoded; see Section 2 for a precise definition); we actually even know that the core of higher-order processes is unable to encode first-order processes [35]. Knowing how this can be achieved with the help of parameterization would be particularly interesting, since parameterization can strictly lift the expressiveness of higher-order processes [14]. Also one can see if the encoding strategy can even vary and yield (possibly drastically) different consequences. The other is that the converse has an almost standard encoding method. First-order processes are so programmable that they can elegantly encode variants of higher-order processes with little difficulty. Yet higher-order processes are still short of a both effective and succinct way to express first-order ones. To this end, the results in [33, 35] are not sufficiently satisfactory because the encoding in those works is far from being elegant and in effect difficult for understanding, analyzing and application. Resolving this can provide (technical) reference for both theoretical study and practical work (*e.g.*, modelling and verification; higher-order programming) beyond the encoding itself.

Closely related with the first issue on expressiveness, the second issue this paper deals with is the characterization of bisimilarity on higher-order processes. Bisimulation studies what it means that two processes have the same behaviour, and is the pivotal part of a process model. Roughly, two processes are bisimilar if whenever one of them does some action, the other must be able to do the same. The bisimulation equality, called *bisimilarity*, is the most widely used form of behavioural equality for various processes. For convenience and to avoid being too technical, sometimes we say bisimulation instead of bisimilarity. In the higher-order models, the almost standard behavioural equality is the context bisimilarity [24]. The central idea of the context bisimilarity is that when comparing output actions, the transmitted process and the residual process (*i.e.*, the process obtained after sending a process) are considered at the same time, rather than separately, as in the (applicative) higher-order bisimulation proposed by Thomsen [32, 33]; the idea of higher-order bisimulation is correct, but it is so fine that it distinguishes processes that are intuitively equivalent. To exemplify the idea of the context bisimilarity (for simplicity we do not consider local names), say P and Q are context bisimilar and $P \xrightarrow{\overline{a}A} P'$ (*i.e.*, P outputs A on a and becomes P'), then $Q \xRightarrow{\overline{a}B} Q'$ (*i.e.*, Q outputs B on a possibly involving some internal actions and becomes Q'), and for every receiving environment $E[\cdot]$, $P' \mid E[A]$ and $Q' \mid E[B]$ are still context bisimilar. Here we write $\mid$ for the parallel composition that models concurrency, and $E[A]$ is a shorthand for E where the hole $[\cdot]$ is substituted by A.

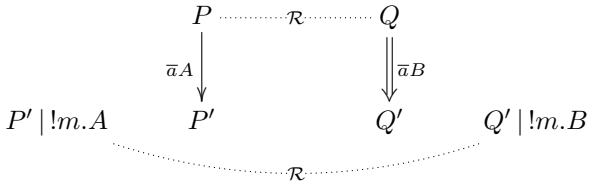
However, in its original form, context bisimilarity is inconvenient to use, because it calls for checking with regard to every possible receiving environment. This leads to works on the simpler characterization of the context bisimilarity, called normal bisimilarity. The core idea of the normal bisimulation, proposed by Sangiorgi [24, 31], is that instead of checking with a general process in input and a general receiving environment in output, one only needs to comply with the matching of some special process or context, on the basis of a class of terms called triggers. To achieve the characterization, a crucial ingredient is a property called factorization. We briefly explain how the normal bisimulation is constructed in the basic higher-order processes without parameterization. In particular as said, we need a property known as the factorization, stating the following property for a fresh name m, where $\approx_{ct}$ denotes the context bisimilarity, $\overline{m}.P$ and $m.P$ are CCS-like prefixes in which the communicated contents are not important, and (m) is the restriction operation that localizes name m.

$$E[A] \approx_{ct} (m)(E[\overline{m}.0] \mid !m.A)$$

One can clearly identify the reposition of the process A of interest, which in fact captures the upshot of the property: move A to a new position as a repository, which in turn can be retrieved as many times as needed in the original environment E, with the help of the pointer undertaken by the fresh channel m ($\overline{m}.0$ is called a trigger). Inspired by the factorization, the normal bisimulation can be developed. We take the output as an example (input is similar), and restriction operation in output is omitted for the sake of simplicity. As stated above, context bisimilarity requires the following chasing diagram, which is now extended with an application of the factorization on the fresh name m.



Since context bisimilarity $\approx_{ct}$ is a congruence, one can somehow cancel the common part of $P' \mid (m)(E[\overline{m}.0] \mid !m.A)$ and $Q' \mid (m)(E[\overline{m}.0] \mid !m.B)$ on the bottom, and simply requires that $P' \mid !m.A$ and $Q' \mid !m.B$ should be related, without any loss in discriminating power. This then leads to the following requirement in the normal bisimulation (assuming $\mathcal{R}$ is a normal bisimulation and m is fresh).



Subsequent works attempt to extend the normal bisimulation to variants of higher-order processes. In Sangiorgi's initial work [24], the normal bisimulation is also obtained for higher-order processes with parameterization. That characterization, however, is made in the presence of first-order name-passing processes and thus not very convincing in a sense. It is not clear if the characterization remains true in a strictly higher-order setting without name-passing. The intrinsic complexity of context bisimulation in presence of parameterization adds to the uncertainty. In [36], we revisit this issue and show that in a purely higher-order setting (viz., no name-passing at all), parameterization on processes does not deprive one of the normal bisimulation. Although the idea is inspired by the original work of Sangiorgi, the design and proof approach is more direct. In [16, 17], Lenglet *et al.* study higher-order processes with passivation (*i.e.*, the process in the output position may evolve), and report a normal bisimulation for a sub-calculus without the restriction operator, but that characterization has somewhat a different flavour, since the higher-order bisimulation [33] rather than the context bisimulation is taken. Though these works carry out insightful research and give meaningful references, it is currently still not clear how to build a simple characterization of context bisimilarity in presence of parameterization over names, and this raises the following fundamental question: *Do higher-order processes with parameterization on names have the normal bisimilarity?* In the second part of this paper, we move further from [24, 36], and offer a normal bisimulation for higher-order processes in the setting of parameterization over both names and processes.

1.1. Contribution

The contribution of this work is as follows. It is worth noting that the all higher-order processes in this work are strictly higher-order, *i.e.*, without name-passing capability.

- We show that the extension with parameterization on both names and processes allows higher-order processes to interpret first-order processes in a surprisingly concise yet elegant manner. The encoding is of a somewhat dissimilar flavour as compared to known encodings between higher-order and first-order models [24, 31, 33, 35], and moreover not possible in absence of parameterization. We give the detailed encoding strategy, and prove that it satisfies a number of desired properties well-known in the field, including full abstraction. In the discussion, we exploit the bisimulation up-to context technique for which we also provide proofs. Additionally, we also demonstrate and analyze a variant encoding of first-order processes using only parameterization on names. Though relevant, the strategy of that encoding is still

more different, and consequently exhibits quite different properties (somewhat beyond expectation). This actually reveals the characteristics of the interwork between the two kinds of parameterization.
The idea of the encodings in this paper is quite different from the abovementioned works in the literature. They make novel and tricky use of parameterization to achieve delivering a name, and this method may potentially be useful for relevant work, *e.g.*, axiomatization or equivalence checking. Moreover, the analysis of the encodings turns out to be not so trivial. It took much effort to pinpoint some counterexamples, which then led to the correct approach of establishing full abstraction. Therefore, besides the ideas of the encoding, which are not reported so far to our knowledge, the exploration of the encodings provides some reference of analytical technique as well.

- We establish the normal bisimilarity, as an effectively simpler characterization of context bisimilarity, for higher-order processes with both kinds of parameterization. This normal bisimilarity extends those for higher-order processes without parameterization, particularly in the manipulation of abstractions on names. As far as we are concerned, similar characterization has not been reported before.

  That the processes are purely higher-order (that is, without name-passing) improves the result in [24], and articulates that the characterization based on normal bisimulation is a property independent of first-order name-passing. Moreover, this does not contradict the argument in [36] that there is little hope for normal bisimilarity to exist in higher-order processes with solely parameterization on names, because here the processes are capable of parameterization on processes as well. The result also reveals that the characterization of normal bisimilarity is not a brittle method and can potentially be of more extensional use.

1.2. Summarizing concerning novelty

We emphasize that the two provided encodings of first-order processes into the higher-order processes with parameterization are newly developed, together with the in-depth analysis (full abstraction or counterexamples). The encodings are completely different from previous ones and provide new ideas of encoding name-passing in the higher-order setting, so is the discussion of the properties which exhibit the intricacy of even such concise encodings (two counterexamples).

In particular, the approach largely improves on the encoding in [37]. The one in [37] adapts from Thomsen's encoding using relabelling [33], and has a quite involved strategy and cumbersome analytical procedure. This is far from satisfaction compared with the encoding in the reverse direction (*i.e.*, from higher-order processes to first-order ones). The encodings in this work have a more succinct strategy and relatively more comprehensive analysis (even with some negative results). Therefore, both the encoding and the analysis approach provide novel strategy and tool for handling name-passing in the higher-order paradigm. To this end, as a recourse to the discussion of the encodings, the technique of bisimulation up-to context is proven to be sound, as we know, for the first time in the setting of higher-order processes with parameterization. This technique can be of independent interest for studies about similar higher-order processes.

We also emphasize that the provided characterization of context bisimilarity with normal bisimilarity in presence of name parameterization is a relatively recent work. It goes beyond previous work in the following two respects. One is that it involves name parameterization besides process parameterization. A critical point here is that one has to find a way to communicate a name in the higher-order setting. In contrast, our previous work in [36, 38] only tackles process parameterization and is technically a bit more tractable to cope with. We also stress that having normal bisimulation in presence of only process parameterization (as indicated in [36, 38]) does not immediately mean that normal bisimulation exists when name parameterization is also included, because how to send a name using certain gadgets is not directly known and needs careful and novel design.

The other respect is that the normal characterization is made in a purely higher-order setting. So one does not have name-passing at all, different from Sangiorgi's work of his thesis [24] in which name-passing is also used as part of the calculus. The normal characterization seems not possible with only name parameterization. Even the combination of name parameterization with process parameterization does not necessarily imply the existence of normal characterization immediately, as the technical manipulation turns out not to be the same as that of the aforementioned works.

In the meanwhile, as somewhat a spinoff, we have given the proofs for the congruence property of normal bisimilarity in presence of name parameterization, as well as for the bisimulation up-to context techniques in the higher-order setting, both in the general case as said above and on the image of the encoding. These proofs also comprise the novel part of this work.

1.3. Organization

The remainder of this paper is organized as follows. In Section 2, we introduce the calculi, relevant proof technique, and a notion of encoding used in this paper. In Section 3, we present the encoding from first-order processes to higher-order processes with parameterization, and discuss its properties. In Section 4, we demonstrate another encoding between the two models, and exploit the different properties. In Section 5, we define the normal bisimulation for higher-order processes with parameterization, and prove that normal bisimilarity truly characterizes the context bisimilarity. Section 6 concludes this work and points to some further directions.

2. Preliminary

In this section, we give the definitions and notations used in this work.

2.1. π-calculus

The first-order (name-passing) pi-calculus, π, is proposed by Milner *et al.* [21]. For the sake of simplicity, throughout the paper, names (ranged over by m, n, u, v, w) are divided into two classes: name constants (ranged over by a, b, c, d, e, f, g, h) and name variables (ranged over by x, y, z) [3, 4, 7]. The grammar is as below with the constructs having their standard meaning. We note that guarded input replication is used instead of general replication, and this does not sacrifice expressiveness [8, 26].

$$P, Q := 0 \;\Big|\; m(x).P \;\Big|\; \overline{m}n.P \;\Big|\; (c)P \;\Big|\; P \mid Q \;\Big|\; !m(x).P$$

A name constant a is bound (or local) in $(a)P$ and free (or global) otherwise. A name variable x is bound in $m(x).P$ and free otherwise. Respectively $\mathrm{fn}(\cdot), \mathrm{bn}(\cdot), \mathrm{n}(\cdot), \mathrm{fnv}(\cdot), \mathrm{bnv}(\cdot), \mathrm{nv}(\cdot)$ denote free name constants, bound name constants, names, free name variables, bound name variables, and name variables in a set of processes or actions (to be defined shortly). A name is fresh if it does not appear in any process under discussion. Processes having no free variables are closed. Usually for convenience, closed processes are considered by default. In the standard way, here are a few derived operators: $\overline{m}(d).P \stackrel{\text{def}}{=} (d)\overline{m}d.P$, $m.P \stackrel{\text{def}}{=} m(x).P$ $(x \notin \mathrm{fnv}(P))$, $\overline{m}.P \stackrel{\text{def}}{=} \overline{m}(d).P$ $(d \notin \mathrm{fn}(P))$; $\tau.P \stackrel{\text{def}}{=} (a)(a.P \mid \overline{a}.0)$ (a fresh). A trailing 0 process is often omitted. We denote tuples by a tilde. For tuple $\widetilde{n}$, $|\widetilde{n}|$ denotes its length and $m\widetilde{n}$ denotes incorporating m. Multiple restriction $(c_1)(c_2)\cdots s(c_k)E$ is abbreviated as $(\widetilde{c})E$. Substitution $\{n/m\}$, ranged over by σ and used in the form $P\{n/m\}$, is a mapping that replaces free m with n in P while keeping the rest unchanged. For tuples, notation $P\{\widetilde{n}/\widetilde{m}\}$ stands for pairwise substitutions. A context C is a process with some subprocess replaced by the hole $[\cdot]$, and $C[A]$ is the process obtained by filling in the hole with A.

The semantics of π is defined by the standard LTS (Labelled Transition System) below.

$$\frac{}{a(x).P \xrightarrow{a(b)} P\{b/x\}} \qquad \frac{}{\overline{a}b.P \xrightarrow{\overline{a}b} P} \qquad \frac{}{!a(x).P \xrightarrow{a(b)} P\{b/x\} \mid !a(x).P}$$

$$\frac{P \xrightarrow{\lambda} P'}{(c)P \xrightarrow{\lambda} (c)P'}\ c \notin \mathrm{n}(\lambda) \qquad \frac{P \xrightarrow{\overline{a}c} P'}{(c)P \xrightarrow{\overline{a}(c)} P'}\ c \neq a \qquad \frac{P \xrightarrow{\lambda} P'}{P \mid Q \xrightarrow{\lambda} P' \mid Q}\ \mathrm{bn}(\lambda) \cap \mathrm{fn}(Q) = \emptyset$$

$$\frac{P \xrightarrow{a(b)} P' \quad Q \xrightarrow{\overline{a}b} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad \frac{P \xrightarrow{a(b)} P' \quad Q \xrightarrow{\overline{a}(b)} Q'}{P \mid Q \xrightarrow{\tau} (b)(P' \mid Q')}\ b \notin \mathrm{fn}(P)$$

Actions, ranged over by λ, α, comprise invisible internal move τ, and visible ones: input $(a(b))$, output $(\overline{a}b)$ and bound output $(\overline{a}(c)$ in which c is bound; every name elsewhere in a label is free). We note that actions occur only on name constants, and a communicated name is also a constant. We denote by $\equiv$ the standard structural congruence [21, 31], which is the smallest relation satisfying α-convertibility, the monoid laws for parallel composition, commutative laws for both composition and restriction, and a distributive law $(c)(P \mid Q) \equiv (c)P \mid Q$ (if $c \notin \mathrm{fn}(Q)$). The structural congruence is not part of the semantics, and we use it only as some structural manipulation method. With the help of α-conversion, we always assume no name capture (which means that a free name falls into the scope of a same bound name). We use $\Longrightarrow$ for the reflexive transitive closure of $\xrightarrow{\tau}$, $\stackrel{\lambda}{\Longrightarrow}$ for $\Longrightarrow\xrightarrow{\lambda}\Longrightarrow$, and $\stackrel{\widehat{\lambda}}{\Longrightarrow}$ for $\stackrel{\lambda}{\Longrightarrow}$ if λ is not τ, and $\Longrightarrow$ otherwise. A process P is divergent, written $P^{\Uparrow}$, if it has an infinite sequence of τ actions.

Throughout the paper, we use the following standard notion of bisimulation [4, 21, 31].

Definition 2.1. A bisimulation is a symmetric relation $\mathcal{R}$ on π processes s.t. whenever $P\,\mathcal{R}\,Q$ the following property holds: If $P \xrightarrow{\lambda} P'$, then $Q \stackrel{\widehat{\lambda}}{\Longrightarrow} Q'$ for some Q' and $P'\,\mathcal{R}\,Q'$. Bisimilarity, $\approx_g$, is the largest bisimulation.

The subscript 'g' in $\approx_g$ is used to differentiate from other notions of bisimulation in this work. We denote by $\sim_g$ the strong bisimilarity (*i.e.*, replacing $\stackrel{\widehat{\alpha}}{\Longrightarrow}$ with $\xrightarrow{\alpha}$ in the definition). Sometimes when it causes no confusion, we will use bisimulation instead of bisimilarity so as to allow more flexibility in some paraphrasing. It is well-known that $\approx_g$ is a congruence [4, 31], and coincides with the so-called local bisimilarity as defined below [5, 35].

Definition 2.2. Local bisimilarity $\approx_l$ is the largest symmetric local bisimulation relation $\mathcal{R}$ on π processes such that: (1) if $P \xrightarrow{\lambda} P'$, λ is not bound output, then $Q \stackrel{\widehat{\lambda}}{\Longrightarrow} Q'$ and $P'\,\mathcal{R}\,Q'$; (2) if $P \xrightarrow{\overline{a}(b)} P'$, then $Q \stackrel{\overline{a}(b)}{\Longrightarrow} Q'$, and for every R, $(b)(P' \mid R)\,\mathcal{R}\,(b)(Q' \mid R)$.

Definitions 2.1 and 2.2 can be extended to open processes. Taking $\approx_g$ as the example, this is defined as follows: suppose $\widetilde{x} \subseteq \mathrm{fnv}(P, Q)$, then $P \approx_g Q$ iff $P\{\widetilde{a}/\widetilde{x}\} \approx_g Q\{\widetilde{a}/\widetilde{x}\}$ for all $\widetilde{a}$.

2.2. Calculus $\Pi^{D,d}$

We first define the basic higher-order calculus and then the extension with parameterization.

2.2.1. Calculus Π

The basic higher-order (process-passing) calculus, Π, is defined by the following grammar in which the operators have their standard meaning. We denote by X, Y, Z process variables.

$$T, T' ::= 0 \;\Big|\; X \;\Big|\; u(X).T \;\Big|\; \overline{u}T'.T \;\Big|\; T \mid T' \;\Big|\; (c)T \;\Big|\; !u(X).T \;\Big|\; !\overline{u}T'.T$$

A process variable X is bound in $u(X).T$ and free otherwise. We reuse the notations for names in π and additionally use $\mathrm{fpv}(\cdot)$, $\mathrm{bpv}(\cdot)$, $\mathrm{pv}(\cdot)$ respectively to denote free process variables, bound process variables and process variables in a set of processes or actions defined shortly. Closed processes are those having no free variables and considered by default. We use $u.0$ for $u(X).0$, $\overline{u}.0$ for $\overline{u}0.0$, and $\tau.P$ for $(c)(c.P \mid \overline{c}.0)$ (c fresh). For convenience later, we may write $\overline{u}(A)$ or $\overline{u}[A]$ for higher-order output. Same as in π, a tilde represents a tuple. A higher-order substitution $T\{A/X\}$ replaces variable X with A and can be extended to tuples in the usual way. Contexts are defined also like in π. In particular, contexts can be extended to multihole ones in the standard way [31]; basically a multihole context has several holes each of which may occur a couple of times. By default, we consider one-hole contexts. To facilitate bisimulation and related discussion, we denote by $E[\widetilde{X}]$ (sometimes called a environment) the expression E (possibly) with free variables $\widetilde{X}$, and $E[\widetilde{A}]$ stands for $E\{\widetilde{A}/\widetilde{X}\}$ respecting name-capture. A context $C[\cdot]$ and an $E[X]$ are different in some sort. Essentially, $E[X]$

is a kind of multihole contexts allowing multiple occurrence of the hole and disallows name capture. So we may also write $E[\cdot]$ in the discussion, bearing in mind its difference from a general context.

The guarded replications used in the grammar can actually be derived [13, 33], and we make them primitive for convenience. The semantics of Π is as below. It is worth noting that we always assume no name capture with resort to α-conversion (*e.g.*, in the last rule the side condition $\widetilde{c} \cap \mathrm{fn}(T_1) = \emptyset$ can be implicit).

$$\frac{}{a(X).T \xrightarrow{a(A)} T\{A/X\}} \qquad \frac{}{\overline{a}A.T \xrightarrow{\overline{a}A} T}$$

$$\frac{}{!\overline{a}A.T \xrightarrow{\overline{a}A} T \mid !\overline{a}A.T} \qquad \frac{}{!a(X).T \xrightarrow{a(A)} T\{A/X\} \mid !a(X).T}$$

$$\frac{T \xrightarrow{\lambda} T'}{(c)T \xrightarrow{\lambda} (c)T'}\, c \notin \mathrm{n}(\lambda) \qquad \frac{T \xrightarrow{(\widetilde{c})\overline{a}A} T'}{(d)T \xrightarrow{(d)(\widetilde{c})\overline{a}A} T'}\, d \in \mathrm{fn}(A) - \{\widetilde{c}, a\}$$

$$\frac{T \xrightarrow{\lambda} T'}{T \mid T_1 \xrightarrow{\lambda} T' \mid T_1}\, \mathrm{bn}(\lambda) \cap \mathrm{fn}(T_1) = \emptyset \qquad \frac{T_1 \xrightarrow{a(A)} T_1' \quad T_2 \xrightarrow{(\widetilde{c})\overline{a}A} T_2'}{T_1 \mid T_2 \xrightarrow{\tau} (\widetilde{c})(T_1' \mid T_2')}\, \widetilde{c} \cap \mathrm{fn}(T_1) = \emptyset$$

We denote by α, λ the actions: internal move (τ), input ($a(A)$), output ($(\widetilde{c})\overline{a}A$) in which $\widetilde{c}$ is some local names carried by A during the output. The notations $\Longrightarrow$, $\overset{\lambda}{\Longrightarrow}$ and $\overset{\widehat{\lambda}}{\Longrightarrow}$ are similar to those in π.

2.2.2. Calculus $\Pi^{D,d}$

Parameterization extends the syntax and semantics of Π as follows. Symbol U_i (respectively, K_i) $(i = 1, \ldots, k)$ is used as a meta-parameter of an abstraction (respectively, meta-instance of an application), and stands for a process variable or name variable (respectively, a process or a name). We reuse the symbol $\equiv$ for the structural congruence in $\Pi^{D,d}$.

Extension of syntax: $\quad \langle U_1, U_2, \ldots, U_k\rangle T \;\big|\; T\langle K_1, K_2, \ldots, K_k\rangle$

Extension of semantics: $\quad \dfrac{Q \equiv P \quad P \xrightarrow{\lambda} P' \quad P' \equiv Q'}{Q \xrightarrow{\lambda} Q'}$

Extension of structural congruence ($\equiv$): $\quad F\langle \widetilde{K}\rangle \equiv T\{\widetilde{K}/\widetilde{U}\}$ where $F \overset{\mathrm{def}}{=} \langle \widetilde{U}\rangle T$ and $|\widetilde{U}| = |\widetilde{K}|$

We denote by $\langle U_1, U_2, \ldots, U_k\rangle T$ a k-ary abstraction in which $U_1, U_2, \ldots, U_k$ are the parameters to be instantiated, and by $T\langle K_1, K_2, \ldots, K_k\rangle$ the application in which the parameters of T are replaced by instances $K_1, K_2, \ldots, K_k$. This application is modelled by an additional rule for structural congruence as above, in combination with the usual LTS rule for structural congruence as well, so as to make the process engaged in application evolve effectively. The condition $|\widetilde{U}| = |\widetilde{K}|$ requires that the parameters and the instantiating objects should be equal in length.

Now parameterization on process is obtained by taking $\widetilde{U}, \widetilde{K}$ as $\widetilde{X}, \widetilde{T}$ respectively, and parameterization on names is obtained by taking $\widetilde{U}, \widetilde{K}$ as $\widetilde{x}, \widetilde{m}$ respectively. The corresponding abstractions are sometimes called process abstraction and name abstraction respectively, and we may use abstraction and parameterization interchangeably. We note that in $\langle U_1, U_2, \ldots, U_k\rangle T$, variables $U_1, U_2, \ldots, U_k$ are bound. For convenience, names are sometimes handled in a similar way as that in π, so are the related notations. That is, we may distinguish between name variables and name constants. For example, we use name variable x in the abstraction $\langle x\rangle T$ while name constant d in the application $T\langle d\rangle$. We denote by $\Pi^{D,d}$ the calculus Π extended with both kinds of parameterizations.

Calculus $\Pi^{D,d}$ can be made more precise with the help of a type system [24], to ensure correct use of abstraction and application. For example, A should be an abstraction in $A\langle P\rangle$, a term like $\langle X\rangle P \mid Q$ is ill-formed, and etc.. This however is not important for this work, and such a type system can be referred to Sangiorgi's work [24] (Sect. 2.3). We always assume type consistency and typing error shall not occur whatsoever. Subject to type

consistency, terms of the form $\langle\widetilde{X}\rangle T$ and $\langle\widetilde{x}\rangle T$ are referred to as process abstractions and name abstractions, and processes as those without the (outermost) domination of abstraction operations (up-to structural congruence).

We further remark that the choice of modelling applications as part of the structural congruence evolves from Sangiorgi's early work. We notice that here the application over an abstraction is not modeled as a reduction (or internal action). From certain viewpoint, a term may have an infinite application sequence (substitution), such as $\Omega \stackrel{\text{def}}{=} (A)\langle A\rangle$ in which $A \stackrel{\text{def}}{=} \langle X\rangle(X\langle X\rangle)$ (whose type should be a recursive one, but the specific form is not very interesting for this work). Yet it appears a bit different from that in lambda-calculus, where an application indeed leads to a reduction. So this shall not be a great concern for the work here, though in general it needs some regulation, that is, the substitution should be well-defined and terminating. Interested readers can refer to Sangiorgi's seminal work that explains the typing and substitution issues [24] (Def. 2.3.3 and Lem. 2.3.4).

2.2.3. Context bisimulation

Throughout the paper, we rely on the following notion of context bisimulation [24, 25].

Definition 2.3 (Context bisimulation). A symmetric relation $\mathcal{R}$ on (closed) $\Pi^{D,d}$ processes is a context bisimulation, if $P\,\mathcal{R}\,Q$ implies the following properties:

- if $P \xrightarrow{\alpha} P'$ and α is $a(A)$ or τ, then $Q \stackrel{\widehat{\alpha}}{\Longrightarrow} Q'$ for some Q' and $P'\,\mathcal{R}\,Q'$;
- if $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$ in which A is a process abstraction, a name abstraction or not an abstraction, then $Q \stackrel{(\widetilde{d})\overline{a}B}{\Longrightarrow} Q'$ for some B that is respectively a process abstraction, a name abstraction or not an abstraction, and moreover for every $E[X]$ such that $\{\widetilde{c},\widetilde{d}\} \cap \mathrm{fn}(E) = \emptyset$ it holds that

$$(\widetilde{c})(E[A] \mid P')\ \mathcal{R}\ (\widetilde{d})(E[B] \mid Q'). \qquad (*)$$

Context bisimilarity, written $\approx_{ct}$, is the largest context bisimulation.

Relation $\sim_{ct}$ denotes the strong context bisimilarity. As is well-known, $\approx_{ct}$ is a congruence; see [24, 25] for a proof. We stress that the bisimulations are defined over closed processes. So the expressions in (*) above in Definition 2.3 should be closed in order to be relared by $\mathcal{R}$. This situation also applies to other definitions concerning bisimulations, *e.g.*, bisimulations up-to context. The extension of $\approx_{ct}$ to open processes and abstractions is defined in the expected way: suppose $\widetilde{X} \subseteq \mathrm{fpv}(T, T')$, then $T \approx_{ct} T'$ iff $T\{\widetilde{A}/\widetilde{X}\} \approx_{ct} T'\{\widetilde{A}/\widetilde{X}\}$ for all closed $\widetilde{A}$; $\langle\widetilde{X}\rangle T \approx_{ct} \langle\widetilde{X}\rangle T'$ iff $T\{\widetilde{A}/\widetilde{X}\} \approx_{ct} T'\{\widetilde{A}/\widetilde{X}\}$ for all closed $\widetilde{A}$.

We note that the matching for output in context bisimulation is required to bear the same kind of communicated process as compared to the simulated action, and the resulting two terms in (*) should be processes so as to be in $\mathcal{R}$ again. This is standard in such definitions; see [24]. Since we always assume type consistency, it means that, for example, for the outputted terms A and B, if A is a name abstraction, so is B; similar for other cases. However, we do not have further requirement on the contents on these abstractions, as long as the requirement of the bisimulation is satisfied (actually this is part of the flexibility context bisimulation permits). Also this design somewhat simplifies discussion, *e.g.*, the characterization by normal bisimulation. Perhaps this can be somehow relaxed, but we do not know how to do it now.

2.2.4. Bisimulation up-to

Before going ahead, we make some discussion about the bisimulation proof method, since it is needed during building bisimulation in this work.

The proofs concerning establishing bisimulation in this paper take advantage of a proof technique called up-to technique [23, 31]. The up-to technique is somewhat a standard and universal tool for process models (even sequential models like lambda-calculus). It has been widely validated and applied. Interested readers can

refer to [23, 28, 31] for good introduction to this traditional and developing technique. We will use the up-to technique in, for instance, the discussion of the properties of the encodings.

The up-to techniques used in this work mainly include up-to strong bisimulation ($\sim_{ct}$) or structural congruence ($\equiv$), and up-to context. Technically, a relation $\mathcal{R}$ is a context bisimulation up-to $\sim_{ct}$ (respectively, $\equiv$) if it satisfies all the clauses of the context bisimulation, except that the resulting two processes in the conclusion of each clause are related by $\sim_{ct} \mathcal{R} \sim_{ct}$ (respectively, $\equiv \mathcal{R} \equiv$) instead of $\mathcal{R}$.

Similarly, a relation $\mathcal{R}$ is a context bisimulation up-to context if it satisfies all the clauses of the context bisimulation, except that the resulting two processes originally related by $\mathcal{R}$, say $P' \mathcal{R} Q'$, in the conclusion of each clause can now be written as $P' \equiv C[P''], Q' \equiv C[Q'']$ for some context C, and P'' and Q'' such that $P'' \mathcal{R} Q''$. That is, they can be rewritten into forms that extract the common parts of P', Q' into a context, and separate the possibly different sub-processes P'', Q'' to be indeed related by $\mathcal{R}$ (the common context may also have multiple holes, in which case the processes filling in the holes are bound to be related component-wise).

A relation $\mathcal{R}$ as such above is not necessarily a bisimulation already, however, it is certain that it is subsumed by the largest bisimulation, *i.e.*, bisimilarity. Below we formally define these up-to techniques for context bisimulation that are used in this work. More systematic introduction, comprehensive discussion and development of this technique can be found in [2, 18, 20, 28–31]. We recall that by default we deal with closed processes in bisimulations.

Definition 2.4 (**up-to $\sim_{ct}$**)**.** A symmetric relation $\mathcal{R}$ on $\Pi^{D,d}$ processes is a context bisimulation up-to $\sim_{ct}$, if whenever $P \mathcal{R} Q$ the following properties hold.

- if $P \xrightarrow{\alpha} P'$ and α is $a(A)$ or τ, then $Q \stackrel{\widehat{\alpha}}{\Longrightarrow} Q'$ for some Q' and $P' \sim_{ct} \mathcal{R} \sim_{ct} Q'$;
- if $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$, then $Q \xRightarrow{(\widetilde{d})\overline{a}B} Q'$ where B and A are both process abstractions, name abstractions or not abstractions, and moreover for every $E[X]$ such that $\{\widetilde{c}, \widetilde{d}\} \cap \mathrm{fn}(E) = \emptyset$ it holds that $(\widetilde{c})(E[A] \mid P') \sim_{ct} \mathcal{R} \sim_{ct} (\widetilde{d})(E[B] \mid Q')$.

If one replaces $\sim_{ct}$ with $\equiv$ in the above definition, one gets the context bisimulation up-to $\equiv$.

Definition 2.5 (**up-to context**)**.** A symmetric relation $\mathcal{R}$ on $\Pi^{D,d}$ processes is a context bisimulation up-to context, if whenever $P \mathcal{R} Q$ the following properties hold.

- if $P \xrightarrow{\alpha} P'$ and α is $a(A)$ or τ, then $Q \stackrel{\widehat{\alpha}}{\Longrightarrow} Q'$ for some Q', $P' \equiv C[P'']$ and $Q' \equiv C[Q'']$ for some (closed) P'', Q'' and context C, and $P'' \mathcal{R} Q''$;
- if $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$, then $Q \xRightarrow{(\widetilde{d})\overline{a}B} Q'$ where B and A are both process abstractions, name abstractions or not abstractions, and moreover for every $E[X]$ such that $\{\widetilde{c}, \widetilde{d}\} \cap \mathrm{fn}(E) = \emptyset$ it holds that $(\widetilde{c})(E[A] \mid P') \equiv C[P'']$ and $(\widetilde{d})(E[B] \mid Q') \equiv C[Q'']$ for some (closed) P'', Q'' and context C, and $P'' \mathcal{R} Q''$.

The two up-to techniques above can be combined, *i.e.*, replacing $P'' \mathcal{R} Q''$ in Definition 2.5 with $P'' \sim_{ct} \mathcal{R} \sim_{ct} Q''$ gives a context bisimulation up-to context and $\sim_{ct}$

To be useful, an up-to technique must be sound.

Theorem 2.6 (**Soundness of the up-to technique**)**.** *If $\mathcal{R}$ is a context bisimulation up-to context or/and up-to $\sim_{ct}$ (or $\equiv$), then $\mathcal{R} \subseteq \approx_{ct}$.*

One may not be so confident of the soundness of up-to context as that of up-to $\equiv$ or $\sim_{ct}$, which after all can be proven in a way similar to that for first-order processes (see [31]). We prove this theorem in Appendix A. Intuitively, the up-to context technique for higher-order processes, as well as other up-to techniques, are sound because higher-order processes can be translated into first-order ones in a fully abstract way, and these up-to techniques are sound for first-order processes [18, 24, 31]. We also refer the reader to a recent work [2] for more detailed discussion of the up-to context technique in the higher-order setting, whereby this technique is used to solve the unique solution issue. Further to our need, in Section 3 we will also prove the soundness of the

up-to context technique on top of the encodings presented in that section. As one shall see, that proof indeed is closely related to the proof of the soundness of the general up-to context technique defined in Definition 2.5.

2.3. A notion of encoding

We give a notion of encoding in this section. We assume that a process model $\mathcal{L}$ is a triplet $(\mathcal{P}, \rightarrow, \approx)$, where $\mathcal{P}$ is the set of processes, $\rightarrow$ is the LTS with a set $\mathcal{A}$ of actions, and $\approx$ is a behavioral equivalence (with structural congruence implicit). Given $\mathcal{L}_i \stackrel{\text{def}}{=} (\mathcal{P}_i, \rightarrow_i, \approx_i)$ $(i{=}1,2)$, an encoding from $\mathcal{L}_1$ to $\mathcal{L}_2$ is a function $[\![\cdot]\!] : \mathcal{P}_1 \longrightarrow \mathcal{P}_2$ that satisfies some set of criteria. Notation $[\![\mathcal{P}_1]\!]$ stands for the image of the $\mathcal{L}_1$-processes inside $\mathcal{L}_2$ under the encoding, that is, $[\![\mathcal{P}_1]\!] \stackrel{\text{def}}{=} \{[\![P]\!] \mid P \text{ is an } \mathcal{L}_1\text{-process}\}$. It should be clear that $[\![\mathcal{P}_1]\!] \subseteq \mathcal{P}_2$ [9, 10].

We use $\dot{\approx}_2$ to denote the behavioural equivalence $\approx_2$ restricted to image $[\![\mathcal{P}_1]\!]$ (up-to the standard structural congruence), with the related terms restricted to those on that image. Since the encoding is compositional, it makes sense to assume that a context is also mapped to a context. However, contexts in the target model can be used in a somewhat more relaxed way, which we now explain. A legal context here for $\dot{\approx}_2$ means a context in the target (higher-order) model that evaluates to a term in the image of the encoding, and thus is good for the bisimulation equality. That is, when a process is put in such a context, one gets in return a term that lies in the image of the encoding. Usually a (receiving) context can be the context corresponding to a context in the source model run through the encoding. Yet from the viewpoint of (context) bisimulation, a context in the (target) model can be a bit more versatile so as to allow more flexibility and discrimination. Consequently, it may be a context $C[\cdot]$ that is not an encoding but evaluates, when fed with a process, to a term falling within the image of encoding (such that suitable for the bisimulation).

We also notice that generally the relation $\dot{\approx}_2$ obtained by restricting the (say, context) bisimilarity to the image of the encoding may be up-to some strong congruence (*e.g.*, structural congruence or strong bisimilarity), *i.e.*, the relation itself is defined on the image of the encoding, yet since generally the encoding may have some supporting structure that becomes garbage during evolution, certain strong congruence is used to collect these garbage and stick to the image of the encoding. Here in this work, as will be seen, things are much simpler. This is because the encodings to be presented in the coming sections are direct enough to yield a somewhat seamless operational correspondence (*i.e.*, an encoding properly evolves into another with little garbage).

The following criteria set (Def. 2.7) used in this paper, as a benchmark for encodings, stems from [14] (the version provided in [14] is based on [9]). As is known, encodability enjoys transitivity [14].

Definition 2.7 (**Criteria for encodings**)**. Static criteria**: (1) Compositionality. *For any k-ary operator op of $\mathcal{L}_1$, and all $P_1, \ldots, P_k \in \mathcal{P}_1$, $[\![op(P_1, \ldots, P_k)]\!] = C_{op}[[\![P_1]\!], \ldots, [\![P_k]\!]]$ for some (multihole) context $C_{op}[\cdots] \in \mathcal{P}_2$;* **Dynamic criteria**: (1a) Adequacy. $P \approx_1 P'$ *implies* $[\![P]\!] \approx_2 [\![P']\!]$. Adequacy is also known as *soundness*. (1b) Completeness. The converse of soundness, *i.e.*, $[\![P]\!] \approx_2 [\![P']\!]$ *implies* $P \approx_1 P'$, is the *completeness*. *Full abstraction* refers to both soundness and completeness. (1c) Weak adequacy (or weak soundness). $P \approx_1 P'$ *implies* $[\![P]\!] \dot{\approx}_2 [\![P']\!]$; (2) Divergence-reflecting. *If* $[\![P]\!]$ *diverges, so does* P.

Adequacy (1a) obviously entails weak adequacy (1b), since $\approx_2$ allows more processes in the target model $\mathcal{L}_2$ (thus more variety of contexts). Yet weak adequacy is still useful because it may be too strong if one requires the encoding process to be compatible with all kinds of contexts in the target model. For instance, in order to achieve first-order interactions in a higher-order target model, it appears quite demanding to require equivalence under all kinds of input because the target higher-order model may have more powerful computational capability (so it can feed a much involved input). More often than not, using limited contexts in the target model, *e.g.*, the encoding processes themselves, may be sufficient to meet the goal of the encoding.

It is worthwhile to note that the criteria are short of those for operational correspondence. Although generally soundness and completeness may appear not very informative in absence of operational correspondence [11, 22], we make this choice in this work out of the following consideration. The criteria for operational correspondence used in [9], though proven useful in many models, appear not quite convenient when discussing encodings into higher-order models [14], since (for example) the case of input can be hard to comply with the criteria due to the increased complexity in the (target) environment. Namely, in the context of the target

higher-order model, toward the establishment of context bisimulation, an input process is arbitrary and can be rather intricate (not necessarily an encoding process at all), so to break the desired correspondence with the original action in the source model. After all, we are more interested in full abstraction, and it is built on the standard bisimulation equalities rather than arbitrary ones as discussed in [22] (so the full abstraction is not trivial). That said, the soundness and completeness are likely based on a different manner of operational correspondence. Notwithstanding, we will discuss the operational correspondence of the encoding in this work, and moreover, as will be seen, the concrete operational correspondence therein somehow strengthens the criteria of operational correspondence used in [9, 14] (in [9] the criteria are not action-labelled and thus the notion of success sensitiveness is contrived; in [14] a labelled variant criteria is posited to its purpose). Beyond the scope of this paper, it would be intriguing to examine the possibility of formally pinning down some variant criteria of operational correspondence having vantage for higher-order (process) models.

3. Encoding π into $\Pi^{D,d}$

We show that π can be encoded in $\Pi^{D,d}$, by presenting an encoding that satisfies all the criteria in Definition 2.7 except adequacy.

3.1. The encoding

The encoding has its core defined as follows, and is homomorphic on the other operators.

$$\begin{array}{rcl} [\![m(x).P]\!] & \stackrel{\text{def}}{=} & m(Y).Y\langle\langle x\rangle[\![P]\!]\rangle \\ [\![\overline{m}n.Q]\!] & \stackrel{\text{def}}{=} & \overline{m}[\langle Z\rangle(Z\langle n\rangle)].[\![Q]\!] \end{array}$$

The encoding appears laconic, and uses both name parameterization and process parameterization. Typically one can assume that Y and Z are fresh for simplicity, but this is not essential, because these variables are bound and can be α-converted whenever necessary, and moreover the encoded first-order process does not have higher-order variables. Specifically, the encoding of an output 'transmits' the name to be sent (*i.e.*, n) in terms of a process parameterization (*i.e.*, $\langle Z\rangle(Z\langle n\rangle)$, sometimes called an encapsulation of n in effect) that, once being received by the encoding of an input, is instantiated by a name-parameterized term (*i.e.*, $\langle x\rangle[\![P]\!]$), which then can apply n on x in the encoding of P, thus fulfilling 'name-passing'. Below we give an example.

Example 3.1. Suppose $P \stackrel{\text{def}}{=} (c)(a(x).\overline{x}c.P_1)$ and $Q \stackrel{\text{def}}{=} (d)(\overline{a}d.d(y).Q_1)$. So

$$\begin{array}{rcl} P \,|\, Q & \stackrel{\tau}{\to} & (d)((c)(\overline{d}c.P_1\{d/x\}) \,|\, d(y).Q_1) \\ & \stackrel{\tau}{\to} & (dc)(P_1\{d/x\} \,|\, Q_1\{c/y\}) \end{array}$$

The encoding and interactions of $[\![P \,|\, Q]\!]$ are as below. The last equivalence is due to the name invariance property to be given shortly in Lemma 3.2.

$$\begin{array}{rcl} [\![P \,|\, Q]\!] & \equiv & (c)(a(Y).Y\langle\langle x\rangle[\![\overline{x}c.P_1]\!]\rangle) \mid (d)(\overline{a}[\langle Z\rangle(Z\langle d\rangle)].[\![d(y).Q_1]\!]) \\ & \stackrel{\tau}{\to} & (d)\big((c)(((\langle Z\rangle(Z\langle d\rangle))\langle\langle x\rangle[\![\overline{x}c.P_1]\!]\rangle) \mid [\![d(y).Q_1]\!]\big) \\ & \equiv & (d)\big((c)([\![\overline{x}c.P_1]\!]\{d/x\}) \mid [\![d(y).Q_1]\!]\big) \\ & \equiv & (d)\big((c)((\overline{x}[\langle Z\rangle(Z\langle c\rangle)].[\![P_1]\!])\{d/x\}) \mid d(Y).Y\langle\langle y\rangle[\![Q_1]\!]\rangle\big) \\ & \equiv & (d)\big((c)((\overline{d}[\langle Z\rangle(Z\langle c\rangle)].[\![P_1]\!]\{d/x\})) \mid d(Y).Y\langle\langle y\rangle[\![Q_1]\!]\rangle\big) \\ & \stackrel{\tau}{\to} & (dc)\big([\![P_1]\!]\{d/x\} \mid (\langle Z\rangle(Z\langle c\rangle))(\langle\langle y\rangle[\![Q_1]\!]\rangle)\big) \\ & \equiv & (dc)\big([\![P_1]\!]\{d/x\} \mid [\![Q_1]\!]\{c/y\}\big) \\ & \equiv & (dc)\big([\![P_1\{d/x\}]\!] \mid [\![Q_1\{c/y\}]\!]\big) \end{array}$$

3.2. Static properties and operational correspondence

The encoding is compositional and divergence-reflecting (since the encoding does not introduce any extra internal action), preserves the (free) names and the structural congruence, as stated in the follow-up lemma whose proof is standard induction.

Lemma 3.2. *Assume P, Q are π processes. The encoding above from π to $\Pi^{D,d}$ is compositional; moreover $[\![P]\!]\{n/m\} \equiv [\![P\{n/m\}]\!]$ (called name invariance), and $[\![P]\!] \equiv [\![Q]\!]$ iff $P \equiv Q$.*

Proof of Lemma 3.2. It is straightforward to check that the encoding is compositional since the designated contexts are easy to capture. For the core of the encoding, the contexts for input and output are respectively $m(Y).Y\langle\langle x\rangle[\cdot]\rangle$ and $\overline{m}[\langle Z\rangle(Z\langle n\rangle)].[\cdot]$. As such, it is a simple induction, on the rules deriving $\equiv$, to show that the encoding preserves structural congruence. Below we prove by induction on the structure of P that $[\![P]\!]\sigma \equiv [\![P\sigma]\!]$ in which σ is a substitution (recall that σ is a mapping on names).

- P is 0. This is trivial.
- P is $m(x).Q$. Then

$$\begin{array}{rcll} [\![P]\!]\sigma & \equiv & (m(Y).Y\langle\langle x\rangle[\![Q]\!]\rangle)\sigma & \\ & \equiv & m'(Y).Y\langle\langle x\rangle([\![Q]\!]\sigma)\rangle & m' \text{ is } \sigma(m) \\ & \equiv & m'(Y).Y\langle\langle x\rangle([\![Q\sigma]\!])\rangle & \text{ind. hyp. (short for induction hypothesis)} \\ & \equiv & [\![m'(x).Q\sigma]\!] & \\ & \equiv & [\![(m(x).Q)\sigma]\!] & \\ & \equiv & [\![P\sigma]\!] & \end{array}$$

- P is $\overline{m}n.Q$. Then

$$\begin{array}{rcll} [\![P]\!]\sigma & \equiv & (\overline{m}[\langle Z\rangle(Z\langle n\rangle)].[\![Q]\!])\sigma & \\ & \equiv & \overline{m'}[\langle Z\rangle(Z\langle n'\rangle)].([\![Q]\!]\sigma) & m', n' \text{ are respectively } \sigma(m), \sigma(n) \\ & \equiv & \overline{m'}[\langle Z\rangle(Z\langle n'\rangle)].([\![Q\sigma]\!]) & \text{ind. hyp.} \\ & \equiv & [\![\overline{m'}n'.(Q\sigma)]\!] & \\ & \equiv & [\![(\overline{m}n.Q)\sigma]\!] & \\ & \equiv & [\![P\sigma]\!] & \end{array}$$

- P is $(c)Q$. Then

$$\begin{array}{rcll} [\![P]\!]\sigma & \equiv & ((c)[\![Q]\!])\sigma & \\ & \equiv & (c)[\![Q]\!]\sigma & \\ & \equiv & (c)[\![Q\sigma]\!] & \text{ind. hyp.} \\ & \equiv & [\![(c)(Q\sigma)]\!] & \\ & \equiv & [\![((c)Q)\sigma]\!] & \\ & \equiv & [\![P\sigma]\!] & \end{array}$$

- P is $Q \mid R$. Then

$$\begin{array}{rcll} [\![P]\!]\sigma & \equiv & ([\![Q]\!] \mid [\![R]\!])\sigma & \\ & \equiv & [\![Q]\!]\sigma \mid [\![R]\!]\sigma & \\ & \equiv & [\![Q\sigma]\!] \mid [\![R\sigma]\!] & \text{ind. hyp.} \\ & \equiv & [\![Q\sigma \mid R\sigma]\!] & \\ & \equiv & [\![(Q \mid R)\sigma]\!] & \\ & \equiv & [\![P\sigma]\!] & \end{array}$$

- P is $!m(x).Q$. Then

$$\begin{array}{rcll} [\![P]\!]\sigma & \equiv & (![\![m(x).Q]\!])\sigma & \\ & \equiv & !([\![m(x).Q]\!])\sigma & \\ & \equiv & !([\![(m(x).Q)\sigma]\!]) & \text{similar to the input case} \\ & \equiv & [\![!((m(x).Q)\sigma)]\!] & \\ & \equiv & [\![(!m(x).Q)\sigma]\!] & \\ & \equiv & [\![P\sigma]\!] & \end{array}$$

□

We now give the correspondence of actions before and after the encoding. To delineate some case of the operational correspondence in terms of certain special input, *i.e.*, a trigger, we define $Tr_m^D \stackrel{\text{def}}{=} \langle Z\rangle\overline{m}Z$ in which m is assumed to be fresh (it will also be used in Section 5, but here simply allows for more flexible characterization of the operational correspondence). We note that sometimes existential quantification is omitted when it is clear from context. The operational correspondence actually says that the encoding of a process always evolves into the encoding of another, thus forming a somewhat closed image domain.

Lemma 3.3 gives the forward operational correspondence while Lemma 3.4 gives the backward operational correspondence. Their proofs are similar. We give the proof of Lemma 3.3 in Appendix B.1. In these lemmas, the input cases are not stated for any process inputted on $[\![P]\!]$, because the action correspondence would become unclear if arbitrary input other than those used in the lemmas is permitted. This is attributed to the richer environment of $\Pi^{D,d}$, and actually is one reason why we argue for relinquishing uniform requirement on operational semantics in the notion of encoding (see Sect. 2.3). We note that, as mentioned above, clause (2) of the two lemmas is based on an input that does not come from an encoding; it is provided for the sake of enabling observation of input from a different angle. These two lemmas can be proven in a similar fashion and we only give the proof of the forward operation correspondence. Moreover, they can be lifted to the weak situation. That is, if one replaces strong transitions (single arrows) with weak transitions (double arrows), the results still hold ($\equiv$ retains because the encoding does not bring any extra internal action); see [24, 31] for a reference. We will however simply refer to these two lemmas in related discussions.

Lemma 3.3. *Suppose P is a π process.*

(1) If $P \xrightarrow{a(b)} P'$, then $[\![P]\!] \xrightarrow{a(\langle Z\rangle(Z\langle b\rangle))} T$ and $T \equiv [\![P']\!]$;
(2) If $P \xrightarrow{a(b)} P'$, then $[\![P]\!] \xrightarrow{a(Tr_m^D)} T$ and $(m)(T \mid !m(Y).Y\langle b\rangle) \approx_{ct} [\![P']\!]$;
(3) If $P \xrightarrow{\overline{a}b} P'$, then $[\![P]\!] \xrightarrow{\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T$ and $T \equiv [\![P']\!]$;
(4) If $P \xrightarrow{\overline{a}(b)} P'$, then $[\![P]\!] \xrightarrow{(b)\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T$ and $T \equiv [\![P']\!]$;
(5) If $P \xrightarrow{\tau} P'$, then $[\![P]\!] \xrightarrow{\tau} T$ and $T \equiv [\![P']\!]$.

The converse is as below.

Lemma 3.4. *Suppose P is a π process.*

(1) If $[\![P]\!] \xrightarrow{a(\langle Z\rangle(Z\langle b\rangle))} T$, then $P \xrightarrow{a(b)} P'$ and $T \equiv [\![P']\!]$;
(2) If $[\![P]\!] \xrightarrow{a(Tr_m^D)} T$, then $P \xrightarrow{a(b)} P'$ and $(m)(T \mid !m(Y).Y\langle b\rangle) \approx_{ct} [\![P']\!]$;
(3) If $[\![P]\!] \xrightarrow{\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T$, then $P \xrightarrow{\overline{a}b} P'$ and $T \equiv [\![P']\!]$;
(4) If $[\![P]\!] \xrightarrow{(b)\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T$, then $P \xrightarrow{\overline{a}(b)} P'$ and $T \equiv [\![P']\!]$;
(5) If $[\![P]\!] \xrightarrow{\tau} T$, then $P \xrightarrow{\tau} P'$ and $T \equiv [\![P']\!]$.

An immediate corollary of the operational correspondence is that the encoding is divergence-reflecting, for the reason that it does not bring about any divergence (Lems. 3.3 and 3.4).

Corollary 3.5. *The encoding from π to $\Pi^{D,d}$ is divergence-reflecting.*

3.3. Soundness

In this section, we discuss the soundness of the encoding. First of all, it is unfortunate that the encoding is not sound in general. To see this, take the processes R_1 and R_2 below. We recall that the CCS-like prefixes are defined as usual, *i.e.*, $a.P \stackrel{\text{def}}{=} a(x).P$ $(x \notin \mathrm{n}(P))$, $\overline{a}.P \stackrel{\text{def}}{=} (c)\overline{a}c.P$ $(c \notin \mathrm{n}(P))$; sometimes we trim the trailing 0, *e.g.*, a stands for $a.0$ and $\overline{a}$ for $\overline{a}.0$.

$$R_1 \stackrel{\text{def}}{=} (b)(a.\overline{b} \mid b.\overline{c}) \qquad\qquad R_2 \stackrel{\text{def}}{=} (b)(a.\overline{b} \mid b.\overline{c} \mid b.\overline{c})$$

Obviously, R_1 and R_2 are bisimilar. Now we examine their encodings.

$$\begin{array}{rcl} [\![R_1]\!] & \equiv & (b)(a(Y).Y\langle\langle x\rangle[\![\overline{b}]\!]\rangle \mid b(Y).Y\langle\langle x\rangle[\![\overline{c}]\!]\rangle) \\ [\![R_2]\!] & \equiv & (b)(a(Y).Y\langle\langle x\rangle[\![\overline{b}]\!]\rangle \mid b(Y).Y\langle\langle x\rangle[\![\overline{c}]\!]\rangle \mid b(Y).Y\langle\langle x\rangle[\![\overline{c}]\!]\rangle) \end{array}$$

We show that $[\![R_1]\!]$ and $[\![R_2]\!]$ are not context bisimilar. Define

$$T \stackrel{\text{def}}{=} (m)(\overline{a}[\langle Z\rangle\overline{m}Z] \mid m(X).(X\langle d\rangle \mid X\langle d\rangle)$$

Then $(a)([\![R_1]\!] \mid T)$ and $(a)([\![R_2]\!] \mid T)$ can be distinguished. The latter can fire two outputs on c, whereas the former cannot, as shown below.

$$\begin{array}{rcl} (a)([\![R_1]\!] \mid T) & \xrightarrow{\tau} \sim_{ct} & (m)((b)(\overline{m}[\langle x\rangle[\![\overline{b}]\!]] \mid b(Y).Y\langle\langle x\rangle[\![\overline{c}]\!]\rangle \\ & & \qquad \mid m(X).(X\langle d\rangle \mid X\langle d\rangle))) \\ & \xrightarrow{\tau} \sim_{ct} & (b)(b(Y).Y\langle\langle x\rangle[\![\overline{c}]\!]\rangle \mid [\![\overline{b}]\!] \mid [\![\overline{b}]\!]) \\ & \equiv & (b)(b(Y).Y\langle\langle x\rangle[\![\overline{c}]\!]\rangle \mid (e)\overline{b}[\langle Z\rangle(Z\langle e\rangle)] \mid [\![\overline{b}]\!]) \\ & \xrightarrow{\tau} \sim_{ct} & (b)([\![\overline{c}]\!] \mid [\![\overline{b}]\!]) \\ & \equiv & (b)((f)\overline{c}[\langle Z\rangle(Z\langle f\rangle)] \mid [\![\overline{b}]\!]) \\ & \xrightarrow{(f)\overline{c}[\langle Z\rangle(Z\langle f\rangle)]} \sim_{ct} & 0 \end{array}$$

$$\begin{array}{rcl} (a)([\![R_2]\!] \mid T) & \xrightarrow{\tau} \sim_{ct} & (m)((b)(\overline{m}[\langle x\rangle[\![\overline{b}]\!]] \mid b(Y).Y\langle\langle x\rangle[\![\overline{c}]\!]\rangle \mid b(Y).Y\langle\langle x\rangle[\![\overline{c}]\!]\rangle) \\ & & \qquad \mid m(X).(X\langle d\rangle \mid X\langle d\rangle))) \\ & \xrightarrow{\tau} \sim_{ct} & (b)(b(Y).Y\langle\langle x\rangle[\![\overline{c}]\!]\rangle \mid b(Y).Y\langle\langle x\rangle[\![\overline{c}]\!]\rangle \mid [\![\overline{b}]\!] \mid [\![\overline{b}]\!]) \\ & \equiv & (b)(b(Y).Y\langle\langle x\rangle[\![\overline{c}]\!]\rangle \mid b(Y).Y\langle\langle x\rangle[\![\overline{c}]\!]\rangle \mid (e)\overline{b}[\langle Z\rangle(Z\langle e\rangle)] \\ & & \qquad \mid (e)\overline{b}[\langle Z\rangle(Z\langle e\rangle)]) \\ & \xrightarrow{\tau}\xrightarrow{\tau} \sim_{ct} & [\![\overline{c}]\!] \mid [\![\overline{c}]\!] \\ & \equiv & (f)\overline{c}[\langle Z\rangle(Z\langle f\rangle)] \mid (f)\overline{c}[\langle Z\rangle(Z\langle f\rangle)] \\ & \xrightarrow{(f)\overline{c}[\langle Z\rangle(Z\langle f\rangle)]} \sim_{ct} & (f)\overline{c}[\langle Z\rangle(Z\langle f\rangle)] \\ & \xrightarrow{(f)\overline{c}[\langle Z\rangle(Z\langle f\rangle)]} \sim_{ct} & 0 \end{array}$$

Intuitively, the reason general soundness does not hold is that context bisimilarity is somewhat more discriminating in the target higher-order calculus, which can have more flexibility when dealing with blocks of processes in presence of parameterization (*e.g.*, some subprocess of interest can be sent as a whole as needed). This is however beyond the capability of a first-order process. In particular, the example above fails the general soundness because it is possible to capture the process corresponding to $\overline{b}$ then to duplicate it, by harnessing

the capability of parameterization. This is similar to some behaviour that can be seen in presence of passivation (see [17], Sect. 2.3).

In spite of the falsity of soundness in general, we can have a somewhat weaker yet still sensible soundness. Remember that our main goal is to achieve first-order concurrency in the higher-order model, so maybe we do not need to be so demanding when coping with the encodings of first-order processes, that is, when testing an encoding process with an input, one can focus on those representing a name (say, its encapsulation) instead of a general one. Then it is expected that soundness will hold under this assumption. Fortunately, this is indeed true.

We have Lemma 3.8 stating the weak soundness of the encoding. We recall that $\dot{\approx}_{ct}$ is the $\approx_{ct}$ restricted to the image of the encoding as defined in Section 2; or in other words, context bisimilarity defined over the image of the encoding. Formally we have the following definition.

Definition 3.6. We define $[\![\pi]\!] \stackrel{\text{def}}{=} \{[\![P]\!] \mid P \text{ is a } \pi \text{ process}\}$. to be the image of the encoding. Since the encoding is compositional, this can be extended to contexts, with a hole mapped to a hole.

We denote by $\dot{\approx}_{ct}$ the context bisimilarity over $[\![\pi]\!]$. That is, it is defined as in Definition 2.3, except that the domain is now the image of the encoding. To be complete, we write down the precise definition below.

Realtion $\dot{\approx}_{ct}$ is the largest context bisimulation $\mathcal{R}$ on $[\![\pi]\!]$ that satisfies the following properties whenever $P \,\mathcal{R}\, Q$.

- if $P \xrightarrow{\alpha} P'$ and α is $a(A)$ (in which A is in $[\![\pi]\!]$) or τ, then $Q \stackrel{\widehat{\alpha}}{\Longrightarrow} Q'$ for some Q' and $P' \,\mathcal{R}\, Q'$;
- if $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$ in which A is a process abstraction, a name abstraction or not an abstraction, then $Q \stackrel{(\widetilde{d})\overline{a}B}{\Longrightarrow} Q'$ for some B that is respectively a process abstraction, a name abstraction or not an abstraction, and moreover for every $E[X]$ such that $\{\widetilde{c}, \widetilde{d}\} \cap \mathrm{fn}(E) = \emptyset$ it holds that

$$(\widetilde{c})(E[A] \mid P') \;\mathcal{R}\; (\widetilde{d})(E[B] \mid Q'). \tag{**}$$

Notice that in the definition the context $E[X]$ in the output case might not itself be an encoding, but indeed is required to become a process in range after evaluation with A and B respectively, *i.e.*, $E[A]$ and $E[B]$ and thus the two processes in (**) are in $[\![\pi]\!]$. See also the explanation in Section 2.3.

It should be clear that $\dot{\approx}_{ct}$ is well defined, due to the operational correspondence and static properties of the encoding. A crucial point here, owing to the tight operational correspondence (Lems. 3.3 and 3.4), is that the contexts to be considered in the bisimulation are those processes in the target model that have reverse-image w.r.t. the encoding. By 'tight', we mean that an encoding $\Pi^{D,d}$ process always evolves into the encoding of another π process, and thus rendering the restriction to the encodings sensible.

We note that the proof of Lemma 3.8 uses the up-to context technique to establish the bisimulation [20, 23, 25, 27, 31]. The up-to technique for $\dot{\approx}_{ct}$ is defined in the same way as that in Definitions 2.4 and 2.5, except that the communicated processes and contexts are all from images of the encoding. The following proposition shows the correctness of the up-to context technique, and its proof is in Appendix B.2. This up-to context technique is sufficient for our purpose, though the general up-to context technique is also true in the broader situation (see Thm. 2.6).

Proposition 3.7. *If a relation $\mathcal{R}$ on the images of the encoding is a context bisimulation up-to context, then $\mathcal{R} \subseteq \dot{\approx}_{ct}$.*

We now prove the weak soundness of the encoding.

Lemma 3.8. *Suppose P is a π process. Then $P \approx_g Q$ implies $[\![P]\!] \dot{\approx}_{ct} [\![Q]\!]$.*

Proof. We show that $\mathcal{R} \stackrel{\text{def}}{=} \{([\![P]\!], [\![Q]\!]) \mid P \approx_g Q\} \cup \dot{\approx}_{ct}$ is a context bisimulation up-to context and $\equiv$. Suppose $[\![P]\!] \,\mathcal{R}\, [\![Q]\!]$ and notice that we consier closed processes as default. We make a case analysis. We note that in

the input and out, we do not consider general processes, but instead those sendable by an encoding, because we confine to the image of the encoding. This is where, particularly the input, this proof cannot be extended to the general case, otherwise the counterexample above would become in vain.

- $[\![P]\!] \xRightarrow{a(\langle Z\rangle(Z\langle b\rangle))} T$. By Lemma 3.4, $P \xRightarrow{a(b)} P'$ and $T \equiv [\![P']\!]$. Because $P \approx_g Q$, we know that $Q \xRightarrow{a(b)} Q' \approx_g P'$ and thus $[\![P']\!] \mathcal{R} [\![Q']\!]$. Then by Lemma 3.3, $[\![Q]\!] \xRightarrow{a(\langle Z\rangle(Z\langle b\rangle))} T'$ and $T' \equiv [\![Q']\!]$. So we have $T \equiv [\![P']\!] \mathcal{R} [\![Q']\!] \equiv T'$.
- $[\![P]\!] \xRightarrow{(b)\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T$. By Lemma 3.4, $P \xRightarrow{\overline{a}(b)} P'$ and $T \equiv [\![P']\!]$. Because $P \approx_g Q$, we know that $Q \xRightarrow{\overline{a}(b)} Q' \approx_g P'$ and thus $[\![P']\!] \mathcal{R} [\![Q']\!]$. Then by Lemma 3.3, $[\![Q]\!] \xRightarrow{(b)\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T'$ and $T' \equiv [\![Q']\!]$. Consider the following pair

  $$(b)(T \mid E[A]) \quad , \quad (b)(T' \mid E[A])$$

  in which $b \notin \mathrm{fn}(E[X])$ and $A \stackrel{\mathrm{def}}{=} \langle Z\rangle(Z\langle b\rangle)$. So

  $$(b)(T \mid E[A]) \equiv (b)([\![P']\!] \mid E[A]) \quad , \quad (b)([\![Q']\!] \mid E[A]) \equiv (b)(T' \mid E[A])$$

  By setting a context $C \stackrel{\mathrm{def}}{=} (b)([\cdot] \mid E[A])$, we have the following pair in which $[\![P']\!] \mathcal{R} [\![Q']\!]$.

  $$C[[\![P']\!]] \quad , \quad C[[\![Q']\!]]$$

  This suffices to close this case in terms of the up-to context requirement.
- $[\![P]\!] \xRightarrow{\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T$. This case is similar to the last case.
- $[\![P]\!] \xRightarrow{\tau} T$. By Lemma 3.4, $P \xRightarrow{\tau} P'$ and $T \equiv [\![P']\!]$. From $P \approx_g Q$, we know $Q \Longrightarrow Q' \approx_g P'$ and thus $[\![P']\!] \mathcal{R} [\![Q']\!]$. Then by Lemma 3.3, $[\![Q]\!] \Longrightarrow T'$ and $T' \equiv [\![Q']\!]$. So we have $T \equiv [\![P']\!] \mathcal{R} [\![Q']\!] \equiv T'$.

□

Remark. We caution again that Lemma 3.8 cannot be extend to general soundness mainly because the input case would become incorrect. Notice that the current clause only allows the input of the 'encapsulation of a name' from encodings. Moving to general process would fail the lemma, because two processes bisimilar in the image of the encodings may not be so in the general case, when we allow more contexts that can be non-encodings. This is actually what the counterexample above suggests.

3.4. Completeness

The completeness of the encoding is given in Lemma 3.10. We note that completeness is true even if we do not constrain the domain to be the image of the encoded π processes.

We first give an intermediate lemma to be used in the proof of the completeness.

Lemma 3.9. *Suppose P and Q are π processes, and $[\![P]\!] \approx_{ct} [\![Q]\!]$. If we have an action $[\![P]\!] \xRightarrow{\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T$ (respectively $[\![P]\!] \xRightarrow{(b)\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T$), then $[\![P]\!]$ must be matched by $[\![Q]\!] \xRightarrow{\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T'$ (respectively $[\![Q]\!] \xRightarrow{(b)\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T'$).*

Proof. We only prove the output in which b is not bound, and the other case is similar. First of all, $[\![Q]\!]$ can only output such shape of processes (by Lem. 3.3). We claim that if the matching is, say $[\![Q]\!] \xRightarrow{\overline{a}[\langle Z\rangle(Z\langle c\rangle)]} T''$ (similar for the case $[\![Q]\!] \xRightarrow{(b)\overline{a}[\langle Z\rangle(Z\langle b\rangle)]}$), then $[\![P]\!]$ and $[\![Q]\!]$ would be immediately distinguishable. In particular, a context can be designed to distinguish between $[\![P]\!]$ and $[\![Q]\!]$ by harnessing c to exhibit different behaviour with some singular process; say, $H \stackrel{\mathrm{def}}{=} a(X).(X\langle A\rangle)$ in which $A \stackrel{\mathrm{def}}{=} \langle x\rangle\overline{x}[\overline{d}]$ where d is fresh. When putting H in

parallel and counteracting some interfering factors up-to structural equivalence (if necessary), we can see that $[\![P]\!]$ and $[\![Q]\!]$ would trigger output of fresh d on different names (*i.e.*, b, c), causing neither of them to be able to match the other. □

Now we prove the completeness of the encoding.

Lemma 3.10. *Suppose P and Q are π processes. Then $[\![P]\!] \approx_{ct} [\![Q]\!]$ implies $P \approx_g Q$.*

Proof. We show that $\mathcal{R} \stackrel{\text{def}}{=} \{(P, Q) \mid [\![P]\!] \approx_{ct} [\![Q]\!]\} \cup \approx_g$ is a local bisimulation. Suppose $P \mathcal{R} Q$. We make a case analysis below.

- $P \xrightarrow{a(b)} P'$. By Lemma 3.3, $[\![P]\!] \xrightarrow{a(\langle Z\rangle(Z\langle b\rangle))} T$ and $T \equiv [\![P']\!]$. Because $[\![P]\!] \approx_{ct} [\![Q]\!]$, we know that $[\![Q]\!] \xrightarrow{a(\langle Z\rangle(Z\langle b\rangle))} T' \approx_{ct} T$. By Lemma 3.4, $Q \xrightarrow{a(b)} Q'$ and $T' \equiv [\![Q']\!]$. Thus we have $[\![P']\!] \equiv T \approx_{ct} T' \equiv [\![Q']\!]$, so $P' \mathcal{R} Q'$, which fulfills this case.
- $P \xrightarrow{\overline{a}b} P'$. By Lemma 3.3, $[\![P]\!] \xrightarrow{\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T$ and $T \equiv [\![P']\!]$. Since $[\![P]\!] \approx_{ct} [\![Q]\!]$, we know that $[\![P]\!]$ must be matched by $[\![Q]\!] \xrightarrow{\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T'$, by Lemma 3.9.
  Therefore for every $E[X]$, we have $T \mid E[\langle Z\rangle(Z\langle b\rangle)] \approx_{ct} T' \mid E[\langle Z\rangle(Z\langle b\rangle)]$. By Lemma 3.4, $Q \xrightarrow{\overline{a}b} Q'$ and $T' \equiv [\![Q']\!]$. So we know

$$[\![P']\!] \mid E[\langle Z\rangle(Z\langle b\rangle)) \approx_{ct} [\![Q']\!] \mid E[\langle Z\rangle(Z\langle b\rangle)] \tag{3.1}$$

  We want to show

$$P' \mathcal{R} Q' \quad \text{that is,} \quad [\![P']\!] \approx_{ct} [\![Q']\!] \tag{3.2}$$

  By setting E to be 0 in (3.1), we obtain (3.2), and thus close this case.
- $P \xrightarrow{\overline{a}(b)} P'$. By Lemma 3.3, $[\![P]\!] \xrightarrow{(b)\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T$ and $T \equiv [\![P']\!]$. Since $[\![P]\!] \approx_{ct} [\![Q]\!]$, we know that $[\![P]\!]$ must be able to be matched by $[\![Q]\!] \xrightarrow{(b)\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T'$ (apply α-conversion if needed), by Lemma 3.9. So for all $E[X]$ s.t. $b \notin \text{fn}(E)$, we have $(b)(T \mid E[\langle Z\rangle(Z\langle b\rangle)]) \approx_{ct} (b)(T' \mid E[\langle Z\rangle(Z\langle b\rangle)])$. By Lemma 3.4, $Q \xrightarrow{\overline{a}(b)} Q'$ and $T' \equiv [\![Q']\!]$. So we know

$$(b)([\![P']\!] \mid E[\langle Z\rangle(Z\langle b\rangle)]) \approx_{ct} (b)([\![Q']\!] \mid E[\langle Z\rangle(Z\langle b\rangle)]) \tag{3.3}$$

  In terms of local bisimulation [5, 35], for every π process R, we need to show

$$(b)(P' \mid R)\, \mathcal{R}\, (b)(Q' \mid R) \quad \textit{i.e.,} \quad (b)([\![P']\!] \mid [\![R]\!]) \approx_{ct} (b)([\![Q']\!] \mid [\![R]\!]) \tag{3.4}$$

  Comparing equations (3.3) and (3.4), one can see that the different part is $E[\langle Z\rangle(Z\langle b\rangle)]$ and $[\![R]\!]$. Since the inverse of the encoding is a surjection, if all possible forms of E is itinerated (notice that E may have the knowledge of b from $\langle Z\rangle(Z\langle b\rangle)$), $[\![R]\!]$ must be hit somewhere (*i.e.*, some choice of E makes $E[\langle Z\rangle(Z\langle b\rangle)]$ and $[\![R]\!]$ equal). Therefore we infer that (3.4) is true and thus complete this case. (As a matter of fact, this could be made somewhat more precise by setting, for instance, E to be $X\langle\langle x\rangle R'\rangle$, where R' does not contain b, so that the implication from (3.3) to (3.4) follows by taking $R'\{b/x\}$ as $[\![R]\!]$.).
- $P \xrightarrow{\tau} P'$. By Lemma 3.3, $[\![P]\!] \xrightarrow{\tau} T$ and $T \equiv [\![P']\!]$. Because $[\![P]\!] \approx_{ct} [\![Q]\!]$, we know $[\![Q]\!] \Longrightarrow T' \approx_{ct} T$. Then by Lemma 3.4, $Q \Longrightarrow Q'$ and $T' \equiv [\![Q']\!]$. So we have $P' \mathcal{R} Q'$ because $[\![P']\!] \equiv T \approx_{ct} T' \equiv [\![Q']\!]$.

□

4. Another approach of encoding π into $\Pi^{D,d}$

In this section, we present another way of encoding π with $\Pi^{D,d}$, which was suggested by A. Schmitt. This encoding, from which the encoding in the previous section borrows inspiration, appears more natural in some sense, though its properties were not clear before this work. We note that the discussion of this section, among others, extends the preliminary work [34] and offers more insight into the 'first-order' programming capacity of $\Pi^{D,d}$.

The encoding, as given below skipping the homomorphic parts, somewhat swaps the roles of input and output and treats $a(x).P$ somehow as $a.\langle x\rangle P$ (like those calculi admitting abstractions and concretions [24]). For convenience, we reuse the encoding notation $[\![\ ]\!]$ and there should be no confusion.

$$\begin{array}{rcl} [\![m(x).P]\!] & \stackrel{\text{def}}{=} & \overline{m}[\langle x\rangle [\![P]\!]] \\ [\![\overline{m}n.Q]\!] & \stackrel{\text{def}}{=} & m(Y).(Y\langle n\rangle \mid [\![Q]\!]) \end{array}$$

The encoding strategy above only employs parameterization on name. One might wonder why not use $\Pi^{D,d}$ with process parameterization eliminated. We will comment on this at the end of this section. From the angle of achieving first-order interaction, the encoding makes full use of the name-abstraction mechanism and is truly appealing. Based on the results in Section 3, it is tempting to expect that this encoding have (nearly) the same properties. However at first sight, it appears not to satisfy some operational correspondence as required in [9, 14], and full abstraction is thus not quite clear. Indeed, this is worthwhile for more investigation. Below first give an example of the encoding by reusing the instances in the last section, and then move on with the discussion of the encoding.

Example 4.1. We recall the two processes $P \stackrel{\text{def}}{=} (c)(a(x).\overline{x}c.P_1)$ and $Q \stackrel{\text{def}}{=} (d)(\overline{a}d.d(y).Q_1)$. Their composition has the following transitions.

$$\begin{array}{rcl} P \mid Q & \stackrel{\tau}{\to} & (d)((c)(\overline{d}c.P_1\{d/x\}) \mid d(y).Q_1) \\ & \stackrel{\tau}{\to} & (dc)(P_1\{d/x\} \mid Q_1\{c/y\}) \end{array}$$

Now the encoding $[\![P \mid Q]\!]$ and its corresponding transitions are given below.

$$\begin{array}{rcl} [\![P \mid Q]\!] & \equiv & (c)(\overline{a}[\langle x\rangle [\![\overline{x}c.P_1]\!]]) \mid (d)(a(Y).(Y\langle d\rangle \mid [\![d(y).Q_1]\!])) \\ & \stackrel{\tau}{\to} \equiv & (c)(d)((\langle x\rangle [\![\overline{x}c.P_1]\!])\langle d\rangle \mid [\![d(y).Q_1]\!]) \\ & \equiv & (c)(d)([\![\overline{x}c.P_1]\!])\{d/x\} \mid [\![d(y).Q_1]\!]) \\ & \equiv & (c)(d)((x(Y).(Y\langle c\rangle \mid [\![P_1]\!]))\{d/x\} \mid \overline{d}[\langle y\rangle [\![Q_1]\!]]) \\ & \equiv & (c)(d)(d(Y).(Y\langle c\rangle \mid [\![P_1]\!]\{d/x\}) \mid \overline{d}[\langle y\rangle [\![Q_1]\!]]) \\ & \stackrel{\tau}{\to} \equiv & (c)(d)((\langle y\rangle [\![Q_1]\!])\langle c\rangle \mid [\![P_1]\!]\{d/x\}) \\ & \equiv & (dc)([\![P_1]\!]\{d/x\} \mid [\![Q_1]\!])\{c/y\}) \\ & \equiv & (dc)([\![P_1\{d/x\}]\!] \mid [\![Q_1]\!]\{c/y\})) \end{array}$$

The remainder of this section is devoted to a full analysis of the encoding above. We first note that, similar to the encoding in Section 3, the encoding comes with compositionality and divergence-reflection, and preserves name substitution (*i.e.*, name invariant) and the structural congruence. These properties will be used implicitly in the arguments of this section. We also point out again that since the encoding is compositional, it makes sense to extend the definition of the encoding to be over contexts, with a hole translated to a hole.

4.1. Operational correspondence

It is pronounced that the operational correspondence does not hold in a direct fashion, because the encoding now has two swaps: one is between input and output; the other between synchrony and asynchrony. From a certain perspective, this encoding is more of a conceptual one, and its analysis is more tricky than the encoding in Section 3.1. Nevertheless, as for the encoding in Section 3.1, from the operational correspondence (Lems. 4.2 and 4.3), we can still see that an encoding always evolves into another one (think of A in these two lemmata as an encoding abstracted on a name, *i.e.*, $\langle x\rangle[\![P]\!]$). So the image of the translation yields a closed (sub)range in the target model.

Lemma 4.2. *Suppose P is a π process and A is a name abstraction.*

1. *If $P \xrightarrow{a(b)} P'$, then $P \sim_g (\widetilde{e})C[a(x).P_1]$ and $P' \equiv (\widetilde{e})C[P_1\{b/x\}] \equiv (\widetilde{e})(C[0] \mid P_1\{b/x\})$ for some evaluation context C, P_1 and $\widetilde{e}$ being the bound names shared between P_1 and its context, and $[\![P]\!] \xrightarrow{(\widetilde{e})\overline{a}[\langle x\rangle([\![P_1]\!])]} T \equiv [\![C[0]]\!]$.*
2. *If $P \xrightarrow{\overline{a}b} P'$, then $[\![P]\!] \xrightarrow{a(A)} T \equiv A\langle b\rangle \mid [\![P']\!]$.*
3. *If $P \xrightarrow{\overline{a}(b)} P'$, then $[\![P]\!] \xrightarrow{a(A)} T \equiv (b)(A\langle b\rangle \mid [\![P']\!])$.*
4. *If $P \xrightarrow{\tau} P'$, then $[\![P]\!] \xrightarrow{\tau} T \equiv [\![P']\!]$.*

The converse is as below.

Lemma 4.3. *Suppose P is a π process and A is a name abstraction.*

1. *If $[\![P]\!] \xrightarrow{(\widetilde{e})\overline{a}[\langle x\rangle([\![P_1]\!])]} T$, then $P \xrightarrow{a(b)} P'$, with $P \sim_g (\widetilde{e})C[a(x).P_1]$ and $P' \equiv (\widetilde{e})C[P_1\{b/x\}] \equiv (\widetilde{e})(C[0] \mid P_1\{b/x\})$ for some evaluation context C with $\widetilde{e}$ being the bound names shared between P_1 and its context, and $T \equiv [\![C[0]]\!]$.*
2. *If $[\![P]\!] \xrightarrow{a(A)} T$, then*
   (i) $P \xrightarrow{\overline{a}b} P'$, and $T \equiv A\langle b\rangle \mid [\![P']\!]$, or
   (ii) $P \xrightarrow{\overline{a}(b)} P'$, and $T \equiv (b)(A\langle b\rangle \mid [\![P']\!])$.
3. *If $[\![P]\!] \xrightarrow{\tau} T$, then $P \xrightarrow{\tau} P'$ and $T \equiv [\![P']\!]$.*

Lemmas 4.2 and 4.3 can be similarly proven, and we prove the former in Appendix C.1. We note that these lemmas can be lifted to the weak case, as for the encoding in Section 3.1. We also notice that the clause (1) of these lemmas uses $\sim_g$, instead of $\equiv$, to deal with P because it is needed in discussing the case for replication.

4.2. Soundness

We reuse the bisimilar processes R_1 and R_2 in Section 3.3, and see what happens to their new encodings. The CCS-like prefixes are defines as there.

$$R_1 \stackrel{\text{def}}{=} (b)(a.\overline{b} \mid b.\overline{c}) \qquad R_2 \stackrel{\text{def}}{=} (b)(a.\overline{b} \mid b.\overline{c} \mid b.\overline{c})$$

Now we examine their encodings.

$$\begin{array}{lcl} [\![R_1]\!] & \equiv & (b)(\overline{a}[\langle x\rangle[\![\overline{b}]\!]] \mid \overline{b}[\langle x\rangle[\![\overline{c}]\!]]) \\ [\![R_2]\!] & \equiv & (b)(\overline{a}[\langle x\rangle[\![\overline{b}]\!]] \mid \overline{b}[\langle x\rangle[\![\overline{c}]\!]] \mid \overline{b}[\langle x\rangle[\![\overline{c}]\!]]) \\ \text{in which} & & \\ [\![\overline{b}]\!] & \equiv & (f)(b(Y).(Y\langle f\rangle \mid 0)) \\ [\![\overline{c}]\!] & \equiv & (g)(c(Z).(Z\langle g\rangle \mid 0)) \end{array}$$

Take $T \stackrel{\text{def}}{=} a(X).(X\langle d\rangle \mid X\langle d\rangle)$. Then $(a)([\![R_1]\!] \mid T)$ and $(a)([\![R_2]\!] \mid T)$ can be distinguished, as shown below. This demonstrates that $[\![R_1]\!]$ and $[\![R_2]\!]$ are not context bisimilar. Particularly, the latter can do two inputs on c, while the former cannot.

$$\begin{array}{rcll}
(a)([\![R_1]\!] \mid T) & \xrightarrow{\tau} \sim_{ct} & (ab)([\![\overline{b}]\!] \mid [\![\overline{b}]\!] \mid \overline{b}[\langle x\rangle [\![\overline{c}]\!]]) \\
& \equiv & (ab)((f)(b(Y).(Y\langle f\rangle \mid 0)) \mid [\![\overline{b}]\!] \mid \overline{b}[\langle x\rangle [\![\overline{c}]\!]]) \\
& \xrightarrow{\tau} \sim_{ct} & (b)([\![\overline{c}]\!] \mid [\![\overline{b}]\!]) \\
& \equiv & (b)((g)(c(Z).(Z\langle g\rangle \mid 0)) \mid [\![\overline{b}]\!]) \\
& \xrightarrow{c(\langle z\rangle 0)} \sim_{ct} & 0
\end{array}$$

$$\begin{array}{rcll}
(a)([\![R_2]\!] \mid T) & \xrightarrow{\tau} \sim_{ct} & (ab)([\![\overline{b}]\!] \mid [\![\overline{b}]\!] \mid \overline{b}[\langle x\rangle [\![\overline{c}]\!]] \mid \mid \overline{b}[\langle x\rangle [\![\overline{c}]\!]]) \\
& \equiv & (ab)((f)(b(Y).(Y\langle f\rangle \mid 0)) \mid (f)(b(Y).(Y\langle f\rangle \mid 0)) \\
& & \qquad \mid \overline{b}[\langle x\rangle [\![\overline{c}]\!]] \mid \overline{b}[\langle x\rangle [\![\overline{c}]\!]]) \\
& \xrightarrow{\tau}\xrightarrow{\tau} \sim_{ct} & [\![\overline{c}]\!] \mid [\![\overline{c}]\!] \\
& \equiv & (g)(c(Z).(Z\langle g\rangle \mid 0)) \mid (g)(c(Z).(Z\langle g\rangle \mid 0)) \\
& \xrightarrow{c(\langle z\rangle 0)} \sim_{ct} & (g)(c(Z).(Z\langle g\rangle \mid 0)) \\
& \xrightarrow{c(\langle z\rangle 0)} \sim_{ct} & 0
\end{array}$$

4.2.1. Weak soundness

Short of soundness, one might expect that the encoding at least has weak soundness, *i.e.*, sound w.r.t. $\dot{\approx}_{ct}$, as defined in Section 2 and used in Section 3.3. We demonstrate that it is not the case. We carry out the analysis with a new counterexample below.

Fix two π processes R_3 and R_4. That they are bisimilar is in the clear. We now show that their encodings are not related by $\dot{\approx}_{ct}$.

$$R_3 \stackrel{\text{def}}{=} (b)(a.b \mid \overline{b}.\overline{c}) \qquad\qquad R_4 \stackrel{\text{def}}{=} (b)(a.b \mid \overline{b}.\overline{c} \mid \overline{b}.\overline{c})$$

Their encodings should have the follow-up simulation.

$$\begin{array}{ccc}
[\![R_3]\!] \equiv (b)(\overline{a}[\langle x\rangle([\![b]\!])] \mid [\![\overline{b}.\overline{c}]\!]) & \qquad & (b)(\overline{a}[\langle x\rangle([\![b]\!])] \mid [\![\overline{b}.\overline{c}]\!] \mid [\![\overline{b}.\overline{c}]\!]) \equiv [\![R_4]\!] \\
\Big\downarrow{\scriptstyle (b)\overline{a}[\langle x\rangle([\![b]\!])]} & & \Big\downarrow{\scriptstyle (b)\overline{a}[\langle x\rangle([\![b]\!])]} \\
0 \mid [\![\overline{b}.\overline{c}]\!] & & 0 \mid [\![\overline{b}.\overline{c}]\!] \mid [\![\overline{b}.\overline{c}]\!]
\end{array}$$

In terms of context bisimulation, we choose $E[X] \stackrel{\text{def}}{=} X\langle e\rangle \mid X\langle e\rangle$ and have

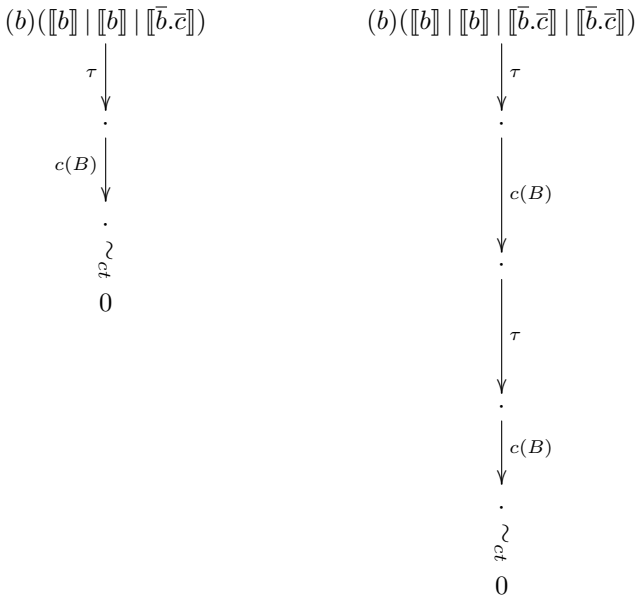
$$\begin{array}{rcl}
E[\langle x\rangle([\![b]\!])] & \equiv & [\![b]\!] \mid [\![b]\!] \\
[\![b]\!] & \equiv & \overline{b}[\langle x\rangle 0] \\
[\![\overline{b}.\overline{c}]\!] & \equiv & (f)(b(Y).(Y\langle f\rangle \mid (g)(c(Z).(Z\langle g\rangle \mid 0))))
\end{array}$$

We note that $E[\langle x\rangle([\![b]\!])]$ corresponds to the π process $b \mid b$ so it is legal for $\dot{\approx}_{ct}$. We now argue that $(b)(E[\langle x\rangle([\![b]\!])] \mid [\![\overline{b}.\overline{c}]\!])$ and $(b)(E[\langle x\rangle([\![b]\!])] \mid [\![\overline{b}.\overline{c}]\!] \mid [\![\overline{b}.\overline{c}]\!])$ are not bisimilar as regard to $\dot{\approx}_{ct}$, *i.e.*,

$$(b)([\![b]\!] \mid [\![b]\!] \mid [\![\overline{b}.\overline{c}]\!]) \quad \not\dot{\approx}_{ct} \quad (b)([\![b]\!] \mid [\![b]\!] \mid [\![\overline{b}.\overline{c}]\!] \mid [\![\overline{b}.\overline{c}]\!])$$

thus violating the output requirement of the context bisimulation. To see this, fix a $\Pi^{D,d}$ abstraction B, *e.g.*, $\langle x\rangle 0$. We then have the following chasing diagram showing the inequality above. Here, . is some unspecified

process.



We thus conclude that $[\![R_3]\!] \not\dot{\approx}_{ct} [\![R_4]\!]$.

To summarize, we have the following lemma.

Lemma 4.4. *Assume P and Q are π processes. Then $P \approx_g Q$ implies neither $[\![P]\!] \approx_{ct} [\![Q]\!]$ nor $[\![P]\!] \dot{\approx}_{ct} [\![Q]\!]$.*

4.3. Completeness

In spite of the failure of the soundness in either general or weak cases, fortuitously the encoding indeed has completeness. It is worth noting that we need Theorem 5.1 (Factorization) in the proof. This not only exhibits somewhat the use of that theorem, but also confers upon in a sense the synergy of the two kinds of abstractions. For the sake of modular organization (viz., putting the two encodings in neighbouring sections), the statement and proof of Theorem 5.1 is deferred until Section 5. Basically, as explained in Section 1, that theorem expresses that a sub-process of a process can be replaced with the so-called trigger, and moved to a new parallel position, which is accessible using the trigger acting as a messenger that faithfully transmits the information at the original position to the new positition, thus retaining the equivalence with the original process. Interested readers are encouraged to have a quick look at the statement of the theorem so as to understand the proof of completeness. More details of the theorem and its proof can be consulted later.

Lemma 4.5. *Assume P and Q are π processes. Then $[\![P]\!] \approx_{ct} [\![Q]\!]$ implies $P \approx_g Q$.*

The proof of Lemma 4.5 is placed in Appendix C.2.

Remark. Basically, the discussion of this section shows that the variant encoding is not weakly sound, let alone sound, although fortunately it is still complete. This warns us that a seemingly small difference in encoding strategy, *e.g.*, swapping the input and output, can lead to rather unexpected results. Notice also that due to this swapping, the counterexample for weak soundness in this section does not work for the encoding in Section 3. A closely relevant issue is whether we can use Π^d instead of $\Pi^{D,d}$ (Π^d denotes the expectable calculus, *i.e.*, Π with parameterization only on names). The point is that this would not bring back the (weak) soundness (the counterexamples still work), rather causes us to lose the factorization property – as far as we can see, thus

making the completeness unclear. Therefore it leaves open the problem of designing a (weakly) sound encoding of name-passing using parameterization on names alone.

5. Normal bisimulation for $\Pi^{D,d}$

In this section, we show that the context bisimilarity in $\Pi^{D,d}$ can be characterized by the much simpler normal bisimilarity. We focus on unary parameterization, *i.e.*, abstractions and applications that admit only one variable or instance respectively, for the sake of simplicity. Moreover, unary parameterization appears more elementary; for example, a k-ary abstraction can be treated as a sequence of unary abstractions. Also recall that the encodings in the previous sections actually utilize only unary parameterization. Intuitively, by means of polyadic communication [31], the characterization by normal bisimilarity can be extended to arbitrary dimensions of parameterization.

5.1. The factorization theorem

We first give the factorization theorem. As explained in Section 1, the factorization theorem is the cornerstone of characterizing context bisimilarity with normal bisimilarity, and the upshot of establishing the factorization theorem is to find the right small processes so-called triggers. Here we have three kinds of triggers, to tackle different kinds of parameterization. In particular, we stipulate that the triggers are as follows: $Tr^d_m \stackrel{\text{def}}{=} \langle z\rangle\overline{m}[\langle Y\rangle(Y\langle z\rangle)]$, $Tr^D_m \stackrel{\text{def}}{=} \langle Z\rangle\overline{m}Z$, and $Tr_m \stackrel{\text{def}}{=} \overline{m}$. These triggers are of somewhat a similar flavour but quite different in shape, with the aim at factorizing out respectively a name abstraction, a process abstraction and a non-abstraction process in some context. The first trigger, *i.e.*, Tr^d_m, is the main contribution of this work, whereas the other two are inherited from [36] and [24] respectively. We note that $\widetilde{\langle U\rangle}E'$ denotes a series of (unary) abstraction on E'.

Theorem 5.1 (**Factorization**). *Given $E[X]$ of $\Pi^{D,d}$, it holds for every A, fresh m (i.e., $m \notin \mathrm{fn}(E,A)$) that*

(1) if A is not an abstraction, then
  (i) if $E[Tr_m]$ is $\widetilde{\langle U\rangle}E'$ for some non-abstraction E', then
    $E[A] \approx_{ct} \widetilde{\langle U\rangle}((m)(E' \mid !m.A))$;
  (ii) if $E[Tr_m]$ is not an abstraction, then
    $E[A] \approx_{ct} (m)(E[Tr_m] \mid !m.A)$.
(2) if A is a process abstraction, then
  (i) if $E[Tr^D_m]$ is $\widetilde{\langle U\rangle}E'$ for some non-abstraction E', then
    $E[A] \approx_{ct} \widetilde{\langle U\rangle}((m)(E' \mid !m(Z).A\langle Z\rangle))$;
  (ii) if $E[Tr^D_m]$ is not an abstraction, then
    $E[A] \approx_{ct} (m)(E[Tr^D_m] \mid !m(Z).A\langle Z\rangle)$.
(3) if A is a name abstraction, then
  (i) if $E[Tr^d_m]$ is $\widetilde{\langle U\rangle}E'$ for some non-abstraction E', then
    $E[A] \approx_{ct} \widetilde{\langle U\rangle}((m)(E' \mid !m(Z).Z\langle A\rangle))$;
  (ii) if $E[Tr^d_m]$ is not an abstraction, then
    $E[A] \approx_{ct} (m)(E[Tr^d_m] \mid !m(Z).Z\langle A\rangle)$.

In Theorem 5.1, we explicitly distinguish between the parameterized case and non-parameterized case, and state them in separate clauses so as to emphasize their difference in forms and allow convenience in related discussion, though certainly it could be stated in a more uniform way [24, 31], as follows.

> Given $\widetilde{\langle U\rangle}E[X]$ in which E is not an abstraction, it holds for every A, fresh m (*i.e.*, $m \notin \mathrm{fn}(E,A)$) that: (1) if A is not an abstraction, then $\widetilde{\langle U\rangle}E[A] \approx_{ct} \widetilde{\langle U\rangle}((m)(E[Tr_m] \mid !m.A))$. (2) if A is a process

abstraction, then $\widetilde{\langle U\rangle}E[A] \approx_{ct} \widetilde{\langle U\rangle}((m)(E[Tr^D_m] \mid !m(Z).A\langle Z\rangle))$. (3) if A is a name abstraction, then $\widetilde{\langle U\rangle}E[A] \approx_{ct} \widetilde{\langle U\rangle}((m)(E[Tr^d_m] \mid !m(Z).Z\langle A\rangle))$.

Clause (1) of Theorem 5.1 is actually Sangiorgi's seminal work [24, 31]. Clause (2) is shown in [36]. Clause (3) depicts the factorization for the abstraction on names, and is the contribution of this work. Intuitively, it allows one to replace, in a context E, a specific component A (being a name abstraction) with the uniform trigger Tr^d_m, move A into a parallel new place, and in the meanwhile build up a connection between Tr^d_m and the new place, so as to maintain the invariance of observational behaviour. Moreover to this end, when the context E reveals some outmost abstractions, then such connection must be upheld by positioning these abstractions outmost accordingly. As opposed to the original process, the process after factorization has a lighter structure to discuss, and one can concentrate on the new place where the factorized process might exhibit potentially distinct behaviour.

With regard to the method of *trigger* (including the technical approach), the fundamental framework is well-developed in the field, due to Sangiorgi [31]. The key to establishing the factorization property for processes allowing abstraction on names is the *trigger*, which had been unknown for a long time in contrast to the case of abstraction on processes and that without abstractions. Once a right trigger is found, the remainder of discussion is relatively more tractable, though still complicated at some points.

Theorem 5.1 depends on some distribution property concerning the process on the move. We put the proof of that property and the proof of Theorem 5.1 in Appendix D. Below we give an example of the factorization concerning abstraction on names.

Example 5.2. The basic idea of factorization concerning abstraction on names can be illustrated in the following example in which m is fresh (*i.e.*, not in $A\langle d\rangle$).

$$\begin{array}{rl} A\langle d\rangle \approx_{ct} & (m)(\ (\langle z\rangle\overline{m}[\langle Y\rangle(Y\langle z\rangle)])\langle d\rangle \mid m(Z).Z\langle A\rangle\ ) \\ \equiv & (m)(\ \overline{m}[\langle Y\rangle(Y\langle d\rangle)] \mid m(Z).Z\langle A\rangle\ ) \end{array}$$

For example, if A is $\langle x\rangle\overline{x}b$, then $A\langle d\rangle \equiv \overline{d}b$, and

$$\begin{array}{rll} A\langle d\rangle & \approx_{ct} & (m)(\ \overline{m}[\langle Y\rangle(Y\langle d\rangle)] \mid m(Z).Z\langle A\rangle\ ) \\ & \approx_{ct} & (\langle Y\rangle(Y\langle d\rangle))\langle A\rangle \\ & \equiv & A\langle d\rangle \ \equiv\ \overline{d}b \end{array}$$

5.2. Normal bisimulation for $\Pi^{D,d}$

Now we give the definition of normal bisimulation whose clauses are designed on top of the factorization theorem. We recall that $Tr_m \stackrel{\text{def}}{=} \overline{m}$, $Tr^D_m \stackrel{\text{def}}{=} \langle Z\rangle\overline{m}Z$, and $Tr^d_m \stackrel{\text{def}}{=} \langle z\rangle\overline{m}[\langle Y\rangle(Y\langle z\rangle)]$.

Definition 5.3. A symmetric binary relation $\mathcal{R}$ on closed processes of $\Pi^{D,d}$ is a normal bisimulation, if whenever $P\,\mathcal{R}\,Q$ the following properties hold:

1. If $P \xrightarrow{a(Tr_m)} P'$ (m is fresh), then $Q \xRightarrow{a(Tr_m)} Q'$ for some Q' s.t. $P'\,\mathcal{R}\,Q'$.
2. If $P \xrightarrow{a(Tr^D_m)} P'$ (m is fresh), then $Q \xRightarrow{a(Tr^D_m)} Q'$ for some Q' s.t. $P'\,\mathcal{R}\,Q'$.
3. If $P \xrightarrow{a(Tr^d_m)} P'$ (m is fresh), then $Q \xRightarrow{a(Tr^d_m)} Q'$ for some Q' s.t. $P'\,\mathcal{R}\,Q'$.
4. If $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$ and A is not an abstraction, then $Q \xRightarrow{(\widetilde{d})\overline{a}B} Q'$ for some $\widetilde{d}, Q'$ and B that is not an abstraction, and it holds that (m is fresh)

$$(\widetilde{c})(P' \mid !m.A)\ \mathcal{R}\ (\widetilde{d})(Q' \mid !m.B).$$

5. If $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$ and A is an abstraction on process, then $Q \xRightarrow{(\widetilde{d})\overline{a}B} Q'$ for some $\widetilde{d}, Q'$ and B that is a process abstraction, and it holds that (m is fresh) $(\widetilde{c})(P' \mid !m(Z).A\langle Z\rangle)\ \mathcal{R}\ (\widetilde{d})(Q' \mid !m(Z).B\langle Z\rangle)$.
6. If $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$ and A is a name abstraction, then $Q \xRightarrow{(\widetilde{d})\overline{a}B} Q'$ for some $\widetilde{d}, Q'$ and B that is a name abstraction, and it holds that (m is fresh)

$$(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle)\ \mathcal{R}\ (\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle).$$

7. If $P \xrightarrow{\tau} P'$, then $Q \Longrightarrow Q'$ for some Q' s.t. $P'\,\mathcal{R}\,Q'$.

Process P is normal bisimilar to Q, written $P \approx_{nr} Q$, if $P\,\mathcal{R}\,Q$ for some normal bisimulation $\mathcal{R}$. Relation $\approx_{nr}$ is called normal bisimilarity. The strong version of $\approx_{nr}$ is denoted by $\sim_{nr}$, which is obtained by replacing above every double arrow with a single arrow and $\Longrightarrow$ with $\xrightarrow{\tau}$ in particular.

Normal bisimilarity is a congruence, as stated in the following theorem.

Theorem 5.4. *Relation $\approx_{nr}$ is a congruence w.r.t. parallel composition and restriction.*

The proof is similar to the congruence proofs in [2, 24, 31, 36]. The framework of proving congruence in the references can be reused. Interestingly, this framework somewhat reminds one of the up-to technique [2]. To be complete, we give the proof of Theorem 5.4 in Appendix D. We also note that in much the same way as Definition 2.5 and Theorem 2.6, we can define normal bisimulation up-to (context) and show its validity.

5.3. Coincidence between normal bisimilarity and context bisimilarity in $\Pi^{D,d}$

We are now in a good position to prove the following characterization theorem.

Theorem 5.5. *In $\Pi^{D,d}$, normal bisimilarity coincides with context bisimilarity; that is, $\approx_{nr} = \approx_{ct}$.*

Proof. Obviously $\approx_{ct}$ is finer than $\approx_{nr}$ because it requires no less than $\approx_{nr}$. In particular, $\approx_{nr}$ can be considered as a special case of $\approx_{ct}$ when the input and the receiving environment of output are chosen to be as in $\approx_{nr}$. So we only need to prove that $\approx_{nr}$ implies $\approx_{ct}$. To do this, we define the relation $\mathcal{R}$ as follows and show that it is a context bisimulation using Theorem 5.1.

$$\mathcal{R} \stackrel{\text{def}}{=} \{(P, Q) \mid P \approx_{nr} Q\}$$

Suppose $P\,\mathcal{R}\,Q$. There are several cases to analyze. We focus on the major ones, that is, the cases when P communicates a name abstraction and when Theorem 5.1 applies for some non-abstraction E, and leave out the other cases.

- $P \xrightarrow{a(A)} P'$. There are three subcases.
  - A is a name abstraction. Then P' must be of the form $E[A]$ for some $E[Z]$ (one can choose such an E that Z is instantiated by the incoming A; similar for the other two subcases). So $P \xrightarrow{a(Tr^d_m)} E[Tr^d_m]$. Since $P \approx_{nr} Q$, we know $Q \xRightarrow{a(Tr^d_m)} Q'' \equiv F[Tr^d_m]$ for some $F[Z]$ and $E[Tr^d_m] \approx_{nr} F[Tr^d_m]$. Then $Q \xRightarrow{a(A)} Q' \equiv F[A]$. Due to the factorization theorem (Thm. 5.1), and the fact that $\approx_{ct}$ implies $\approx_{nr}$, we have

$$\begin{array}{rll} P' \equiv E[A] & \approx_{nr} & (m)(E[Tr^d_m] \mid !m(Z).Z\langle A\rangle) \\ & \approx_{nr} & (m)(F[Tr^d_m] \mid !m(Z).Z\langle A\rangle) \\ & \approx_{nr} & F[A] \equiv Q' \end{array}$$

    So $P'\mathcal{R}Q'$.
  - A is an abstraction on process. Similar to the last subcase, except that Tr^D_m is used instead.

- A is not an abstraction. Similar to the first subcase, except that Tr_m is used.

- $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$. There are again three subcases.
  - A is a name abstraction. Because $P \approx_{nr} Q$, $Q \xRightarrow{(\widetilde{d})\overline{a}B} Q'$ and it holds that (m is fresh)

$$(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle) \approx_{nr} (\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)$$

    Using some structural manipulation, for every $E[X]$ ($\mathrm{fn}(E[X]) \cap \widetilde{c}\widetilde{d} = \emptyset$), we have

$$(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle \mid E[Tr_m^d]) \approx_{nr} (\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle \mid E[Tr_m^d])$$

    and then

$$(\widetilde{c})(P' \mid (m)(!m(Z).Z\langle A\rangle \mid E[Tr_m^d])) \approx_{nr} (\widetilde{d})(Q' \mid (m)(!m(Z).Z\langle B\rangle \mid E[Tr_m^d]))$$

    Now by the factorization theorem (Thm. 5.1) and the fact that $\approx_{ct}$ implies $\approx_{nr}$, we know

$$(\widetilde{c})(P' \mid E[A]) \approx_{nr} (\widetilde{d})(Q' \mid E[B])$$

    and thus

$$(\widetilde{c})(P' \mid E[A])\, \mathcal{R}\, (\widetilde{d})(Q' \mid E[B])$$

    as required by context bisimulation.
  - A is an abstraction on process. Similar to the last subcase, except that Tr_m^D is used.
  - A is not an abstraction. Similar to the first subcase, except that Tr_m is used.

- $P \xrightarrow{\tau} P'$. Then we immediately have $Q \Longrightarrow Q'$ and $P' \approx_{nr} Q'$, so $P' \mathcal{R} Q'$.

□

6. Conclusion

In this paper, we have exhibited two new encodings of name-passing in the higher-order paradigm that allows parameterization, and a normal bisimulation in that setting as well. In the former, we demonstrate the conformance or inconsistency of the encoding with respect to some well-established criteria in the literature. In the latter, we prove the coincidence between normal bisimilarity and context bisimilarity by pinpointing how to factorize an abstraction on some name. As a supporting byproduct, we show that the bisimulation up-to technique, particularly up-to context, works in the higher-order process model with parameterization.

The results of this paper can be dedicated to facilitate further study on the expressiveness of higher-order processes. The following questions, among others, are still open: whether π can be encoded in a higher-order setting only allowing parameterization on processes; whether there is a better encoding of π than the one in Section 4 or in [37], using higher-order processes only capable of parameterization on names (the corresponding calculus is written as Π^d); whether Π^d affords a normal-like characterization of context bisimilarity. For the last one, the major difficulty is to find a proper trigger-like apparatus, and then of course, the factorization property. To have a taste, let us revisit the example in Section 5, *i.e.*, the Π^d process $W \stackrel{\text{def}}{=} A\langle d\rangle$, in which $A \stackrel{\text{def}}{=} \langle x\rangle\overline{x}$, and clearly $W \equiv \overline{d} \xrightarrow{\overline{d}} 0$. We note that the concrete name d can be provided by the environment dynamically, *i.e.*, during run-time. We expect to have a factorization property like the one as follows:

$$(\widetilde{c})(T\langle d\rangle \mid F[A]) \approx_{ct} A\langle d\rangle \equiv \overline{d} \tag{6.1}$$

where T is supposed to be some 'trigger' subject to the capability of the calculus, and F to be the new place holding a repository of A, with $\widetilde{c}$ being the local names shared by T and F. Now the left hand side of (6.1) must have a transition $\stackrel{\overline{d}}{\Longrightarrow}$, which should result from the interaction between $T\langle d\rangle$ and $F[A]$, and during these interactions d must be received by F so as to be fed to A in its own setting such that A eventually gets the right instantiation in some uniform way. However, strikingly different from the case of $\Pi^{D,d}$, there appears no hope that T can finish this job of activating A in $F[A]$ in need, with only name parameterization in a purely higher-order realm (*i.e.*, no name-passing). To this point, resolving this conundrum may amount to exploiting further the expressiveness of the parameterization on name; or otherwise, showing the impossibility of characterizing context bisimilarity should rest on some precisely formulated criteria. This might take us one step further to scrutinize the discrepancy in the role of names in higher-order models.

Appendix A. Proofs for Section 2

Remark. For the sake of conciseness, in the arguments we sometimes omit the existential statement such as "for some ..." if clear from context.

We give the proof of Theorem 2.6. We make two remarks before going into the proof.

- The proof can be readily adapted to the proof of the correctness of the up-to context technique for other bisimulations such as normal bisimulation, as well as the context bisimulation on the encodings. The main difference is that the receiving contexts are restricted and the input processes are limited. Nevertheless, we provide somewhat a more direct proof for this technique on the encodings (Prop. 3.7). In the case of normal bisimulation, for instance, the input objects are triggers and the receiving context are the corresponding trigger-receiving ones.
- The main thrust of the proof reminds one of proving the congruence properties of a bisimilarity in a higher-order setting [31]. Actually this indeed can lead to the congruence of the context bisimulation [24, 31].

Proof of Theorem 2.6. Assume $\mathcal{R}$ is as defined in Definition 2.5. We define the relations $\mathcal{R}_n$ and $\mathcal{R}'$ on (closed) processes as follows. Recall that $\mathbb{N}$ is the set of natural numbers.

$$\begin{array}{lcl}
\mathcal{R}_0 & \stackrel{\text{def}}{=} & \mathcal{R} \\
\mathcal{R}_{n+1} & \stackrel{\text{def}}{=} & \{(C[M], C[N]) \mid M\,\mathcal{R}_n\,N \text{ and } C \in \mathcal{D}\} \\
\mathcal{R}' & \stackrel{\text{def}}{=} & \bigcup_{i\in\mathbb{N}} \mathcal{R}_i
\end{array}$$

$$\mathcal{D} \stackrel{\text{def}}{=} \left\{ \begin{array}{cccccc} a(X).[\cdot], & \overline{a}([\cdot]).R, & \overline{a}A_1.[\cdot], & \langle X\rangle[\cdot], & \langle x\rangle[\cdot], \\ [\cdot]\langle A_1\rangle, & [\cdot]\langle d\rangle, & R\,|\,[\cdot], & (d)[\cdot] & \end{array} \right\}$$

We note that here $\mathcal{R}'$ is somehow degined to be closed under the operations. This vantage will be used essentially in the passing of the proof. We shall prove by induction on n that $\mathcal{R}'$ is a context bisimulation up-to $\equiv$, that is, each $\mathcal{R}_n$ meets the requirement of the context bisimulation. Since $\mathcal{R} \subseteq \mathcal{R}'$, $\mathcal{R} \subseteq \approx_{ct}$ follows. To be clear, for each $\mathcal{R}_n$ and every pair (P, Q) in it, we show that it satisfies the simulation requirement, and the resulting pair is in $\mathcal{R}'$. In particular, if $P\,\mathcal{R}_n\,Q$ and P does an action step, then Q does a simulating step and the results are related by $\mathcal{R}'$, in the way of a context bisimulation. So each $\mathcal{R}_n$ is not necessarily a context bisimulation by itself, but when taking them together as $\mathcal{R}'$, one eventually comes by a context bisimulation indeed.

I. The induction basis: n is 0, and $\mathcal{R}_0$ is $\mathcal{R}$. This case is immediate in terms of $\mathcal{R}$'s definition.

II. Suppose the result holds for $\mathcal{R}_k$, we prove the result for $\mathcal{R}_{k+1}$. Now suppose $C[P]\,\mathcal{R}_{k+1}\,C[Q]$ for $P\,\mathcal{R}_k\,Q$. For convenience, we give the result for $\mathcal{R}_k$ as follows.

- if $P \xrightarrow{\alpha} P'$ and α is $a(A)$ or τ, then $Q \stackrel{\widehat{\alpha}}{\Longrightarrow} Q'$ for some Q', and $P'\,\mathcal{R}'\,Q'$;
- if $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$, then $Q \stackrel{(\widetilde{d})\overline{a}B}{\Longrightarrow} Q'$ for some same-type B, and moreover for every $E[X]$ as stipulated, it holds that $(\widetilde{c})(E[A] \mid P')\,\mathcal{R}'\,(\widetilde{d})(E[B] \mid Q')$.

Now in order to achieve the induction step, we make a case analysis of the actions from $C[P]$ w.r.t. C, to show that $C[P]$ and $C[Q]$ satisfies the bisimulation requirement. In the argument, $\cdot$ stands for some unspecified process.

1. $C[P] \xrightarrow{a(A)} \cdot$.
   - The cases when C is $\overline{a}([\cdot]).R$, $\overline{a}A_1.[\cdot]$, $\langle X\rangle[\cdot]$, $\langle x\rangle[\cdot]$, $[\cdot]\langle A_1\rangle$, $[\cdot]\langle d\rangle$ are not possible.
   - C is $a(X).[\cdot]$. In this case, since P and Q are closed, $C[P] \xrightarrow{a(A)} P$ can be simulated by $C[Q] \xrightarrow{a(A)} Q$. So we have $P\,\mathcal{R}'\,Q$ because $P\,\mathcal{R}_k\,Q$.
   - C is $R \mid [\cdot]$. There are two possibilities.
     - The action is from P. That is, $P \xrightarrow{a(A)} P'$, and $C[P] \xrightarrow{a(A)} R \mid P'$. Since $P\,\mathcal{R}_k\,Q$, we know that $Q \stackrel{a(A)}{\Longrightarrow} Q'$ and $P'\,\mathcal{R}'\,Q'$. So $C[Q]$ simulates by $C[Q] \stackrel{a(A)}{\Longrightarrow} R \mid Q'$. From $P'\,\mathcal{R}'\,Q'$, we have
     $$R \mid P' \;\; \mathcal{R}' \;\; R \mid Q'$$
     - The action is from R. That is, $R \xrightarrow{a(A)} R'$, and $C[P] \xrightarrow{a(A)} R' \mid P$. So $C[Q]$ simulates by $C[Q] \xrightarrow{a(A)} R' \mid Q$. Because $P\,\mathcal{R}_k\,Q$, we have
     $$R' \mid P \;\; \mathcal{R}' \;\; R' \mid Q$$
   - C is $(d)[\cdot]$. The action is from P. That is, $P \xrightarrow{a(A)} P'$, and $C[P] \xrightarrow{a(A)} (d)P'$. Since $P\,\mathcal{R}_k\,Q$, we know that $Q \stackrel{a(A)}{\Longrightarrow} Q'$ for some Q' and $P'\,\mathcal{R}'\,Q'$. So $C[Q]$ simulates by $C[Q] \stackrel{a(A)}{\Longrightarrow} (d)Q'$. Because $P'\,\mathcal{R}'\,Q'$, we have
   $$(d)P' \;\; \mathcal{R}' \;\; (d)Q'$$

2. $C[P] \xrightarrow{(\widetilde{c})\overline{a}A} \cdot$.
   - The cases when C is $a(X).[\cdot]$, $\langle X\rangle[\cdot]$, $\langle x\rangle[\cdot]$, $[\cdot]\langle A_1\rangle$, $[\cdot]\langle d\rangle$ are not possible.
   - C is $\overline{a}([\cdot]).R$. In this case, $C[P] \xrightarrow{\overline{a}P} R$. Then $C[Q]$ simulates by $C[Q] \xrightarrow{\overline{a}Q} R$, and we have for every E as stipulated in the definition of context bisimulation (for conciseness, we will skip this stipulation henceforth)
   $$E[P] \mid R \;\equiv\; C'[P]\,\mathcal{R}'\,C'[Q] \;\equiv\; E[Q] \mid R$$
   in which $C' \stackrel{\text{def}}{=} E[\cdot] \mid R$, because $P\,\mathcal{R}_k\,Q$.
   - C is $\overline{a}A_1.[\cdot]$. In this case, $C[P] \xrightarrow{\overline{a}A_1} P$. Then $C[Q]$ simulates by $C[Q] \xrightarrow{\overline{a}A_1} Q$, and we have for every E as stipulated
   $$E[A_1] \mid P \;\equiv\; C'[P]\,\mathcal{R}'\,C'[Q] \;\equiv\; E[A_1] \mid Q$$
   in which $C' \stackrel{\text{def}}{=} E[A_1] \mid [\cdot]$, because $P\,\mathcal{R}_k\,Q$.

- C is $R \mid [\cdot]$. There are several possibilities.
  - The action is from R. That is, $R \xrightarrow{(\widetilde{c})\overline{a}A} R'$ and $C[P] \equiv R \mid P \xrightarrow{(\widetilde{c})\overline{a}A} R' \mid P$. In this case, $C[Q]$ simulates by $C[Q] \xrightarrow{(\widetilde{c})\overline{a}A} R' \mid Q$. Now we have for every E as demanded, and $C' \stackrel{\text{def}}{=} (\widetilde{c})(E[A] \mid R' \mid [\cdot])$.

$$\begin{array}{rcl}(\widetilde{c})(E[A] \mid R' \mid P) & \equiv & C'[P] \\ (\widetilde{c})(E[A] \mid R' \mid Q) & \equiv & C'[Q]\end{array}$$

    Hence we obtain $C'[P] \,\mathcal{R}'\, C'[Q]$ with $P \,\mathcal{R}_k\, Q$.
  - The action is from P. That is, $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$ and $C[P] \equiv R \mid P \xrightarrow{(\widetilde{c})\overline{a}A} R \mid P'$. In this case, since $P \,\mathcal{R}_k\, Q$, we know that $Q \xRightarrow{(\widetilde{d})\overline{a}B} Q'$, and for every E as stipulated, it holds that $(\widetilde{c})(E[A] \mid P') \,\mathcal{R}'\, (\widetilde{d})(E[B] \mid Q')$. So $C[Q]$ simulates by $C[Q] \xRightarrow{(\widetilde{d})\overline{a}B} R \mid Q'$. Now, we have for every E:

$$\begin{array}{rcl}(\widetilde{c})(E[A] \mid R \mid P') & \equiv & R \mid (\widetilde{c})(E[A] \mid P') \equiv C[(\widetilde{c})(E[A] \mid P')] \\ (\widetilde{d})(E[B] \mid R \mid Q') & \equiv & R \mid (\widetilde{d})(E[B] \mid Q') \equiv C[(\widetilde{d})(E[B] \mid Q')]\end{array}$$

    Hence we obtain $C[(\widetilde{c})(E[A] \mid P')] \,\mathcal{R}'\, C[(\widetilde{d})(E[B] \mid Q')]$ because $(\widetilde{c})(E[A] \mid P') \,\mathcal{R}'\, (\widetilde{d})(E[B] \mid Q')$.
- C is $(e)[\cdot]$ (we use e instead of d here for clarity). There are a number of possibilities.
  - $e \notin \mathrm{fn}(A)$. The action is $C[P] \xrightarrow{(\widetilde{c})\overline{a}A} (e)P'$ from $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$ (we assume $e \notin \widetilde{c}$). Then since $P \,\mathcal{R}_k\, Q$, we know that $Q \xRightarrow{(\widetilde{d})\overline{a}B} Q'$ (we assume $e \notin \widetilde{d}$, and for every E as stipulated, it holds that $(\widetilde{c})(E[A] \mid P') \,\mathcal{R}'\, (\widetilde{d})(E[B] \mid Q')$. There are two possibilities of the simulation by $C[Q]$.
    - $e \notin \mathrm{fn}(B)$. Then $C[Q]$ simulates by $C[Q] \xRightarrow{(\widetilde{d})\overline{a}B} (e)Q'$. Now for every E as stipulated,

$$\begin{array}{rcl}(\widetilde{c})(E[A] \mid (e)P') & \equiv & (e)(\widetilde{c})(E[A] \mid P') \equiv C[(\widetilde{c})(E[A] \mid P')] \\ (\widetilde{d})(E[B] \mid (e)Q') & \equiv & (e)(\widetilde{d})(E[B] \mid Q') \equiv C[(\widetilde{d})(E[B] \mid Q')]\end{array}$$

      Hence we obtain $C[(\widetilde{c})(E[A] \mid P')] \,\mathcal{R}'\, C[(\widetilde{d})(E[B] \mid Q')]$ because $(\widetilde{c})(E[A] \mid P') \,\mathcal{R}'\, (\widetilde{d})(E[B] \mid Q')$.
    - $e \in \mathrm{fn}(B)$. Then $C[Q]$ simulates by $C[Q] \xRightarrow{(\widetilde{d}e)\overline{a}B} Q'$. Now for every E as stipulated,

$$\begin{array}{rcl}(\widetilde{c})(E[A] \mid (e)P') & \equiv & (e)(\widetilde{c})(E[A] \mid P') \equiv C[(\widetilde{c})(E[A] \mid P')] \\ (\widetilde{d}e)(E[B] \mid Q') & \equiv & (e)(\widetilde{d})(E[B] \mid Q') \equiv C[(\widetilde{d})(E[B] \mid Q')]\end{array}$$

      Hence we obtain $C[(\widetilde{c})(E[A] \mid P')] \,\mathcal{R}'\, C[(\widetilde{d})(E[B] \mid Q')]$ because $(\widetilde{c})(E[A] \mid P') \,\mathcal{R}'\, (\widetilde{d})(E[B] \mid Q')$.
  - $e \in \mathrm{fn}(A)$. The action is $C[P] \xrightarrow{(\widetilde{c}e)\overline{a}A} P'$ from $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$ (we assume $e \notin \widetilde{c}$). Then since $P \,\mathcal{R}_k\, Q$, we know that $Q \xRightarrow{(\widetilde{d})\overline{a}B} Q'$ (we assume $e \notin \widetilde{d}$, and for every E as stipulated, it holds that $(\widetilde{c})(E[A] \mid P') \,\mathcal{R}'\, (\widetilde{d})(E[B] \mid Q')$. There are two possibilities of the simulation by $C[Q]$.
    - $e \notin \mathrm{fn}(B)$. Then $C[Q]$ simulates by $C[Q] \xRightarrow{(\widetilde{d})\overline{a}B} (e)Q'$. Now for every E as stipulated,

$$\begin{array}{rcl}(\widetilde{c}e)(E[A] \mid P') & \equiv & (e)(\widetilde{c})(E[A] \mid P') \equiv C[(\widetilde{c})(E[A] \mid P')] \\ (\widetilde{d})(E[B] \mid (e)Q') & \equiv & (e)(\widetilde{d})(E[B] \mid Q') \equiv C[(\widetilde{d})(E[B] \mid Q')]\end{array}$$

      Hence we obtain $C[(\widetilde{c})(E[A] \mid P')] \,\mathcal{R}'\, C[(\widetilde{d})(E[B] \mid Q')]$ because $(\widetilde{c})(E[A] \mid P') \,\mathcal{R}'\, (\widetilde{d})(E[B] \mid Q')$.

* $e \in \mathrm{fn}(B)$. Then $C[Q]$ simulates by $C[Q] \overset{(\widetilde{de})\overline{a}B}{\Longrightarrow\!\!\!=\!\!\!\Longrightarrow} Q'$. Now for every E as stipulated,

$$\begin{array}{rcl} (\widetilde{ce})(E[A] \mid P') & \equiv & (e)(\widetilde{c})(E[A] \mid P') \equiv C[(\widetilde{c})(E[A] \mid P')] \\ (\widetilde{de})(E[B] \mid Q') & \equiv & (e)(\widetilde{d})(E[B] \mid Q') \equiv C[(\widetilde{d})(E[B] \mid Q')] \end{array}$$

Hence we obtain $C[(\widetilde{c})(E[A] \mid P')] \,\mathcal{R}'\, C[(\widetilde{d})(E[B] \mid Q')]$ because $(\widetilde{c})(E[A] \mid P') \,\mathcal{R}'\, (\widetilde{d})(E[B] \mid Q')$.

3. $C[P] \xrightarrow{\tau} \cdot$.
   - The cases when C is $a(X).[\cdot]$, $\overline{a}([\cdot]).R$, $\overline{a}A_1.[\cdot]$, $\langle X\rangle[\cdot]$, $\langle x\rangle[\cdot]$, $[\cdot]\langle A_1\rangle$, $[\cdot]\langle d\rangle$ are not possible.
   - C is $R \mid [\cdot]$. Three possibilities.
     - τ is from R. That is, $R \xrightarrow{\tau} R'$, and $C[P] \xrightarrow{\tau} R' \mid P$. Then $C[Q]$ simulates by $C[Q] \xrightarrow{\tau} R' \mid Q$. So we have $R' \mid P \,\mathcal{R}'\, R' \mid Q$ because $P \,\mathcal{R}_k\, Q$.
     - τ is from P. That is, $P \xrightarrow{\tau} P'$, and $C[P] \xrightarrow{\tau} R \mid P'$. Since $P \,\mathcal{R}_k\, Q$, we know $Q \Longrightarrow Q'$ and $P' \,\mathcal{R}'\, Q'$. So $C[Q]$ simulates by $C[Q] \Longrightarrow R \mid Q'$. Because $P' \,\mathcal{R}'\, Q'$, we have the following pair in $\mathcal{R}'$ as required.

$$R \mid P' \quad \mathcal{R}' \quad R \mid Q'$$

     - τ is from interaction between R and P. Two more subcases.
     * P makes an output and R makes an input. That is, $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$, $R \xrightarrow{a(A)} R' \equiv E'[A]$ for some $E'[Y]$ (such that Y is only replaced by the inputted A), and $C[P] \xrightarrow{\tau} (\widetilde{c})(R' \mid P')$. Since $P \,\mathcal{R}_k Q$, we know that $Q \overset{(\widetilde{d})\overline{a}B}{\Longrightarrow\!\!\!=\!\!\!\Longrightarrow} Q'$, and for every E as required, it holds that $(\widetilde{c})(E[A] \mid P') \,\mathcal{R}'\, (\widetilde{d})(E[B] \mid Q')$. So $C[Q]$ simulates by $C[Q] \overset{\tau}{\Longrightarrow} (\widetilde{d})(R'' \mid Q')$ in which $R \xrightarrow{a(B)} R'' \equiv E'[B]$. We can take E to be E' and obtain the following pair in $\mathcal{R}'$:

$$(\widetilde{c})(R' \mid P') \;\equiv\; (\widetilde{c})(E'[A] \mid P') \quad \mathcal{R}' \quad (\widetilde{d})(E'[B] \mid Q') \;\equiv\; (\widetilde{d})(R'' \mid Q')$$

     * P makes an input and R makes an output. That is, $P \xrightarrow{a(A)} P'$, $R \xrightarrow{(\widetilde{c})\overline{a}A} R'$, and $C[P] \xrightarrow{\tau} (\widetilde{c})(R' \mid P')$. Since $P \,\mathcal{R}_k Q$, we know that $Q \overset{a(A)}{\Longrightarrow\!\!\!=\!\!\!\Longrightarrow} Q'$ and $P' \,\mathcal{R}'\, Q'$. So $C[Q]$ simulates by $C[Q] \overset{\tau}{\Longrightarrow} (\widetilde{c})(R' \mid Q')$. Because $P' \,\mathcal{R}'\, Q'$, we have

$$(\widetilde{c})(R' \mid P') \quad \mathcal{R}' \quad (\widetilde{c})(R' \mid Q')$$

   - C is $(d)[\cdot]$. In this case, the action $C[P] \equiv (d)P \xrightarrow{\tau} (d)P'$ comes from $P \xrightarrow{\tau} P'$. Since $P \,\mathcal{R}_k\, Q$, we know $Q \Longrightarrow Q'$ and $P' \,\mathcal{R}'\, Q'$. So $C[Q]$ simulates by $C[Q] \Longrightarrow (d)Q'$. Because $P' \,\mathcal{R}'\, Q'$, we have the following pair in $\mathcal{R}'$.

$$(d)P' \quad \mathcal{R}' \quad (d)Q'$$

□

Appendix B. Proofs for Section 3

B.1 Proof for the operational correspondence of the encoding in Section 3

We here give the proof of Lemma 3.3.

Proof of Lemma 3.3. We prove the clauses by induction on the structure of P. The case P is 0 is trivial.

- P is $a(x).P_1$.
  - Input. $P \xrightarrow{a(b)} P_1\{b/x\} \equiv P'$. So $[\![P]\!] \equiv a(Y).Y\langle\langle x\rangle[\![P_1]\!]\rangle \xrightarrow{a(\langle Z\rangle(Z\langle b\rangle))} \equiv [\![P_1]\!]\{b/x\} \equiv T$. We then have $T \equiv [\![P_1\{b/x\}]\!] \equiv [\![P']\!]$. Moreover, $[\![P]\!] \xrightarrow{a(Tr_m^D)} \overline{m}[\langle x\rangle[\![P_1]\!]] \equiv T'$. We have

$$\begin{array}{ll} & (m)(T' \mid !m(Y).Y\langle b\rangle) \\ \equiv & (m)(\overline{m}[\langle x\rangle[\![P_1]\!]] \mid !m(Y).Y\langle b\rangle) \\ \approx_{ct} & (m)(!m(Y).Y\langle b\rangle \mid [\![P_1]\!]\{b/x\}) \\ \approx_{ct} & [\![P_1]\!]\{b/x\}) \equiv [\![P']\!] \end{array}$$

    Notice that the first $\approx_{ct}$ follows from a communication over the restricted name m.
- P is $\overline{a}b.P_1$.
  - Output. $P \xrightarrow{\overline{a}b} P_1 \equiv P'$. Then $[\![P]\!] \equiv \overline{a}[\langle Z\rangle(Z\langle b\rangle)].[\![P_1]\!] \xrightarrow{\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} [\![P_1]\!] \equiv T$, so $T \equiv [\![P']\!]$.
- P is $P_1 \mid P_2$.
  - Input. Assume $P_1 \xrightarrow{a(b)} P_1'$, and $P \xrightarrow{a(b)} P_1' \mid P_2 \equiv P'$. By ind. hyp., $[\![P_1]\!] \xrightarrow{a(\langle Z\rangle(Z\langle b\rangle))} T_1$ and $T_1 \equiv [\![P_1']\!]$. Then $[\![P]\!] \equiv [\![P_1]\!] \mid [\![P_2]\!] \xrightarrow{a(\langle Z\rangle(Z\langle b\rangle))} T_1 \mid [\![P_2]\!] \stackrel{\text{def}}{=} T$. So $T \equiv [\![P_1']\!] \mid [\![P_2]\!] \equiv [\![P']\!]$. Moreover, $[\![P_1]\!] \xrightarrow{a(Tr_m^D)} T_2$ and $(m)(T_2 \mid !m(Y).Y\langle b\rangle) \approx_{ct} [\![P_1']\!]$. Thus $[\![P]\!] \equiv [\![P_1]\!] \mid [\![P_2]\!] \xrightarrow{a(Tr_m^D)} T_2 \mid [\![P_2]\!] \stackrel{\text{def}}{=} T'$. Hence $(m)(T' \mid !m(Y).Y\langle b\rangle) \equiv (m)(T_2 \mid [\![P_2]\!] \mid !m(Y).Y\langle b\rangle) \approx_{ct} [\![P_1']\!] \mid [\![P_2]\!] \equiv [\![P']\!]$.
  - Output. Assume $P_1 \xrightarrow{\overline{a}b} P_1'$, and $P \xrightarrow{\overline{a}b} P_1' \mid P_2 \equiv P'$. By ind. hyp., $[\![P_1]\!] \xrightarrow{\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T_1$ and $T_1 \equiv [\![P_1']\!]$. Then $[\![P]\!] \equiv [\![P_1]\!] \mid [\![P_2]\!] \xrightarrow{\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T_1 \mid [\![P_2]\!] \stackrel{\text{def}}{=} T$. So $T \equiv [\![P_1']\!] \mid [\![P_2]\!] \equiv [\![P']\!]$.
  - Bound output. Assume $P_1 \xrightarrow{\overline{a}(b)} P_1'$, and $P \xrightarrow{\overline{a}(b)} P_1' \mid P_2 \equiv P'$. By ind. hyp., $[\![P_1]\!] \xrightarrow{(b)\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T_1$ and $T_1 \equiv [\![P_1']\!]$. Then $[\![P]\!] \equiv [\![P_1]\!] \mid [\![P_2]\!] \xrightarrow{(b)\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T_1 \mid [\![P_2]\!] \stackrel{\text{def}}{=} T$. So $T \equiv [\![P_1']\!] \mid [\![P_2]\!] \equiv [\![P']\!]$.
  - Internal action (τ). The interesting case is when there is a communication between P_1 and P_2. Take, for example, the case $P_1 \xrightarrow{a(b)} P_1'$ and $P_2 \xrightarrow{\overline{a}(b)} P_2'$, and $P' \equiv (b)(P_1' \mid P_2')$. By ind. hyp., $[\![P_1]\!] \xrightarrow{a(\langle Z\rangle(Z\langle b\rangle))} T_1$ and $T_1 \equiv [\![P_1']\!]$, and $[\![P_2]\!] \xrightarrow{(b)\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T_2$ and $T_2 \equiv [\![P_2']\!]$. So $[\![P]\!] \equiv [\![P_1]\!] \mid [\![P_2]\!] \xrightarrow{\tau} (b)(T_1 \mid T_2) \stackrel{\text{def}}{=} T$. Thus $[\![P']\!] \equiv (b)([\![P_1']\!] \mid [\![P_2']\!]) \equiv (b)(T_1 \mid T_2) \equiv T$.
- P is $(c)P_1$.
  - Input. Assume $P_1 \xrightarrow{a(b)} P_1'$, and $P \xrightarrow{a(b)} (c)P_1' \equiv P'$. By ind. hyp., $[\![P_1]\!] \xrightarrow{a(\langle Z\rangle(Z\langle b\rangle))} T_1$ and $T_1 \equiv [\![P_1']\!]$. Then $[\![P]\!] \equiv (c)[\![P_1]\!] \xrightarrow{a(\langle Z\rangle(Z\langle b\rangle))} (c)T_1 \stackrel{\text{def}}{=} T$. So $T \equiv (c)[\![P_1']\!] \equiv [\![P']\!]$. Moreover, $[\![P_1]\!] \xrightarrow{a(Tr_m^D)} T_2$ and $(m)(T_2 \mid !m(Y).Y\langle b\rangle) \approx_{ct} [\![P_1']\!]$. Thus $[\![P]\!] \equiv (c)[\![P_1]\!] \xrightarrow{a(Tr_m^D)} (c)T_2 \stackrel{\text{def}}{=} T'$. Hence $(m)(T' \mid !m(Y).Y\langle b\rangle) \equiv (m)((c)T_2 \mid !m(Y).Y\langle b\rangle) \approx_{ct} (c)[\![P_1']\!] \equiv [\![P']\!]$.
  - Output. Assume $P_1 \xrightarrow{\overline{a}b} P_1'$, and $P \xrightarrow{\overline{a}b} (c)P_1' \equiv P'$. By ind. hyp., $[\![P_1]\!] \xrightarrow{\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} T_1$ and $T_1 \equiv [\![P_1']\!]$. Then $[\![P]\!] \equiv (c)[\![P_1]\!] \xrightarrow{\overline{a}[\langle Z\rangle(Z\langle b\rangle)]} (c)T_1 \stackrel{\text{def}}{=} T$. So $T \equiv (c)[\![P_1']\!] \equiv [\![P']\!]$.
  - Bound output. The interesting case is $P_1 \xrightarrow{\overline{a}c} P_1'$, and $P \xrightarrow{\overline{a}(c)} P' \equiv P_1'$. By ind. hyp., $[\![P_1]\!] \xrightarrow{\overline{a}[\langle Z\rangle(Z\langle c\rangle)]} T_1$ and $T_1 \equiv [\![P_1']\!]$. So $[\![P]\!] \equiv (c)[\![P_1]\!] \xrightarrow{(c)\overline{a}[\langle Z\rangle(Z\langle c\rangle)]} T \equiv T_1 \equiv [\![P']\!]$.
  - Internal action (τ). Assume $P_1 \xrightarrow{\tau} P_1'$, and $P \xrightarrow{\tau} (c)P_1' \equiv P'$. By ind. hyp., $[\![P_1]\!] \xrightarrow{\tau} T_1$ and $T_1 \equiv [\![P_1']\!]$. Then $[\![P]\!] \equiv (c)[\![P_1]\!] \xrightarrow{\tau} (c)T_1 \stackrel{\text{def}}{=} T$. So $T \equiv (c)[\![P_1']\!] \equiv [\![P']\!]$.
- P is $!a(x).P_1$.

- Input. $P \xrightarrow{a(b)} P_1\{b/x\} \mid P \equiv P'$. So $[\![P]\!] \equiv !a(Y).Y\langle\langle x\rangle[\![P_1]\!]\rangle \xrightarrow{a(\langle Z\rangle(Z\langle b\rangle))} \equiv [\![P_1]\!]\{b/x\} \mid [\![P]\!] \equiv T$. We then have $T \equiv [\![P_1\{b/x\}]\!] \mid [\![P]\!] \equiv [\![P']\!]$. Moreover, $[\![P]\!] \xrightarrow{a(Tr^D_m)} \overline{m}[\langle x\rangle[\![P_1]\!]] \mid [\![P]\!] \equiv T'$. We now have

$$\begin{array}{ll} & (m)(T' \mid !m(Y).Y\langle b\rangle) \\ \equiv & (m)(\overline{m}[\langle x\rangle[\![P_1]\!]] \mid [\![P]\!] \mid !m(Y).Y\langle b\rangle) \\ \approx_{ct} & (m)(!m(Y).Y\langle b\rangle \mid [\![P]\!] \mid [\![P_1]\!]\{b/x\}) \\ \approx_{ct} & [\![P_1]\!]\{b/x\}) \mid [\![P]\!] \equiv [\![P']\!] \end{array}$$

  Notice that the first $\approx_{ct}$ results from a communication over the restricted name m.

□

B.2 Proof of up-to context technique in Section 3

We give the proof of Proposition 3.7. For convenience, we reproduce the definition of the context bisimulation up-to context. By saying some term is an encoding, we mean that it is the encoding of certain (π) process. Recall that the encoding of a context is a context put through the encoding with the hole translated into a hole. In the output clause of Definition B.1, Q is required to make the same output as P. This corresponds to the requirement of bound output in the bisimulation of π-calculus, and suffices for our purpose.

Definition B.1 (**up-to context**). A symmetric relation $\mathcal{R}$ on the image of the encoding is a context bisimulation up-to context, if whenever $P\,\mathcal{R}\,Q$ the following properties hold.

- if $P \xrightarrow{\alpha} P'$ and α is τ or $a(A)$ in which A is $\langle Z\rangle(Z\langle b\rangle)$, then $Q \xRightarrow{\widehat{\alpha}} Q'$ for some Q', $P' \equiv C[P'']$ and $Q' \equiv C[Q'']$ for some P'', Q'' and context C that is an encoding, and $P''\,\mathcal{R}\,Q''$;
- if $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$ in which A is $\langle Z\rangle(Z\langle b\rangle)$ and $\widetilde{c}$ is either empty or $\{b\}$, then $Q \xRightarrow{(\widetilde{c})\overline{a}A} Q'$ and for every $E[X]$ s.t. $\{\widetilde{c}\} \cap \mathrm{fn}(E) = \emptyset$, it holds that $(\widetilde{c})(E[A] \mid P') \equiv C[P'']$ and $(\widetilde{c})(E[A] \mid Q') \equiv C[Q'']$ for some P'', Q'' and context C that are encodings, and $P''\,\mathcal{R}\,Q''$.

The framework of the proof is akin to proving the congruence of a bisimilarity in a higher-order setting [2, 6, 19, 24, 31] (think of $\mathcal{R}$ as the context bisimilarity in the definition of $\mathcal{R}'$ below and the argument is similar). Indeed this observation ensures that the context bisimilarity on top of the encodings is a congruence.

Proof of Proposition 3.7. Assume $\mathcal{R}$ is as defined in Definition B.1. We show that the relation $\mathcal{R}'$ defined below is a context bisimulation up-to $\equiv$, thus proving the proposition.

$$\mathcal{R}' \stackrel{\text{def}}{=} \{(C[P], C[Q]) \mid P\,\mathcal{R}\,Q, \text{ and } C \text{ is a context as by Definition B.1}\}$$

We make a case analysis. Suppose $P\,\mathcal{R}'\,Q$ where P and Q are processes. We notice that we may sometimes omit the $\cdot$ in $C[\cdot]$ for the sake of conciseness.

1. $C[P] \xrightarrow{a(A)} T_1$ in which $A \equiv \langle X\rangle X\langle b\rangle$.
   - The case when C is $[\cdot]$ is immediate because $C[P]$ and $C[Q]$ are P and Q respectively, and $P\,\mathcal{R}\,Q$ from premise. The cases when C is $\overline{a}(C_1[\cdot]).R$, $\overline{a}A_1.C_1[\cdot]$, $\langle X\rangle C_1[\cdot]$, $\langle x\rangle C_1[\cdot]$ are not possible. The cases when C is $C_1[\cdot]\langle A_1\rangle$, $C_1[\cdot]\langle d\rangle$ fall onto the other cases after possible application.
   - C is $a(X).C_1[\cdot]$. In this case, $C[P] \xrightarrow{a(A)} T_1 \equiv C_1[P]\{A/X\} \equiv C_2[P]$ for $C_2 \stackrel{\text{def}}{=} C_1\{A/X\}$ since P is closed. Then $C[Q]$ simulates by $C[Q] \xrightarrow{a(A)} C_1[Q]\{A/X\} \equiv C_2[Q]$. So we have $C_2[P]\,\mathcal{R}'\,C_2[Q]$ with $P\,\mathcal{R}\,Q$.
   - C is $R \mid C_1[\cdot]$. There are three possibilities.
     - The action is from P. That is, $P \xrightarrow{a(A)} P'$, and $C[P] \xrightarrow{a(A)} R \mid C_1[P']$. Since $P\,\mathcal{R}\,Q$, we know that $Q \xRightarrow{a(A)} Q'$ for some Q', $P' \equiv C'[P'']$ and $Q' \equiv C'[Q'']$ for some P'', Q'' and context C as stipulated, and

$P'' \,\mathcal{R}\, Q''$. So $C[Q]$ simulates by $C[Q] \stackrel{a(A)}{\Longrightarrow} R \,|\, C_1[Q']$. Now we have for context $C_2 \stackrel{\text{def}}{=} R \,|\, C_1[C'[]]$

$$\begin{array}{rcl} R \,|\, C_1[P'] & \equiv & R \,|\, C_1[C'[P'']] \;\equiv\; C_2[P''] \\ R \,|\, C_1[Q'] & \equiv & R \,|\, C_1[C'[Q'']] \;\equiv\; C_2[Q''] \end{array}$$

Hence we obtain $C_2[P''] \,\mathcal{R}'\, C_2[Q'']$ with $P'' \,\mathcal{R}\, Q''$.

- The action is from C_1. That is, $C_1[P] \xrightarrow{a(A)} C_1'[P]$, and $C[P] \xrightarrow{a(A)} R \,|\, C_1'[P]$. So $C[Q]$ simulates by $C[Q] \xrightarrow{a(A)} R \,|\, C_1'[Q]$. Now we have for context $C_2 \stackrel{\text{def}}{=} R \,|\, C_1'[]$

$$\begin{array}{rcl} R \,|\, C_1'[P] & \equiv & C_2[P] \\ R \,|\, C_1'[Q] & \equiv & C_2[Q] \end{array}$$

Hence we obtain $C_2[P] \,\mathcal{R}'\, C_2[Q]$ with $P \,\mathcal{R}\, Q$.

- The action is from R. That is, $R \xrightarrow{a(A)} R'$, and $C[P] \xrightarrow{a(A)} R' \,|\, C_1[P]$. So $C[Q]$ simulates by $C[Q] \xrightarrow{a(A)} R' \,|\, C_1[Q]$. Now we have for context $C_2 \stackrel{\text{def}}{=} R' \,|\, C_1[]$

$$\begin{array}{rcl} R' \,|\, C_1[P] & \equiv & C_2[P] \\ R' \,|\, C_1[Q] & \equiv & C_2[Q] \end{array}$$

Hence we obtain $C_2[P] \,\mathcal{R}'\, C_2[Q]$ with $P \,\mathcal{R}\, Q$.

- C is $(d)C_1[\cdot]$. There are two possibilities.
  - The action is from P. That is, $P \xrightarrow{a(A)} P'$, and $C[P] \xrightarrow{a(A)} (d)C_1[P']$. Since $P \,\mathcal{R}\, Q$, we know that $Q \stackrel{a(A)}{\Longrightarrow} Q'$ for some Q', $P' \equiv C'[P'']$ and $Q' \equiv C'[Q'']$ for some P'', Q'' and context C as stipulated, and $P'' \,\mathcal{R}\, Q''$. So $C[Q]$ simulates by $C[Q] \stackrel{a(A)}{\Longrightarrow} (d)C_1[Q']$. Now for some context $C_2 \stackrel{\text{def}}{=} (d)C_1[C'[]]$ we have

$$\begin{array}{rcl} (d)C_1[P'] & \equiv & (d)C_1[C'[P'']] \;\equiv\; C_2[P''] \\ (d)C_1[Q'] & \equiv & (d)C_1[C'[Q'']] \;\equiv\; C_2[Q''] \end{array}$$

  Hence we obtain $C_2[P''] \,\mathcal{R}'\, C_2[Q'']$ with $P'' \,\mathcal{R}\, Q''$.

  - The action is from C_1. That is, $C_1[P] \xrightarrow{a(A)} C_1'[P]$, and $C[P] \xrightarrow{a(A)} (d)C_1'[P]$. So $C[Q]$ simulates by $C[Q] \xrightarrow{a(A)} (d)C_1'[Q]$. Now we have for context $C_2 \stackrel{\text{def}}{=} (d)C_1'[]$

$$\begin{array}{rcl} (d)C_1'[P] & \equiv & C_2[P] \\ (d)C_1'[Q] & \equiv & C_2[Q] \end{array}$$

  Hence we obtain $C_2[P] \,\mathcal{R}'\, C_2[Q]$ with $P \,\mathcal{R}\, Q$.

2. $C[P] \xrightarrow{(\widetilde{c})\overline{a}A} T_1$ in which $A \equiv \langle X \rangle X\langle b \rangle$ and $\widetilde{c}$ is $\{b\}$ or empty.
   - The case when C is $[\cdot]$ is immediate. The cases when C is $a(X).C_1[\cdot]$, $\langle X \rangle C_1[\cdot]$, $\langle x \rangle C_1[\cdot]$ are not possible. The cases when C is $C_1[\cdot]\langle A_1 \rangle$, $C_1[\cdot]\langle d \rangle$ fall onto the other cases after possible application.
   - C is $\overline{a}(C_1[\cdot]).R$. This case does not make sense because in the encodings the term allowed to be on the object position of an output can only be of the form $\langle X \rangle X\langle b \rangle$.
   - C is $\overline{a}A_1.C_1[\cdot]$. Then here $\widetilde{c}$ is empty, A_1 is A, and T_1 is $C_1[P]$. So $C[Q]$ simulates by $C[Q] \xrightarrow{\overline{a}A} C_1[Q]$. Now for every $E[\cdot]$ as specified in the definition, consider $E[A] \,|\, C_1[P] \equiv C'[P]$ and $E[A] \,|\, C_1[Q] \equiv C'[Q]$ in which $C' \stackrel{\text{def}}{=} E[A] \,|\, C_1[]$. Thus we have $C'[P] \,\mathcal{R}'\, C'[Q]$ with $P \,\mathcal{R}\, Q$.
   - C is $R \,|\, C_1[\cdot]$. There are four possibilities.

◦ The action is from R. That is, $R \xrightarrow{(\widetilde{c})\overline{a}A} R'$ and $C[P] \equiv R \,|\, C_1[P] \xrightarrow{(\widetilde{c})\overline{a}A} T_1 \equiv R' \,|\, C_1[P]$. In this case, $C[Q]$ simulates by $C[Q] \xrightarrow{(\widetilde{c})\overline{a}A} R' \,|\, C_1[Q]$. Now we have for every E as demanded, and $C_2 \stackrel{\text{def}}{=} (\widetilde{c})(E[A] \,|\, R' \,|\, C_1[])$.

$$\begin{array}{rcl}(\widetilde{c})(E[A] \,|\, R' \,|\, C_1[P]) & \equiv & C_2[P] \\ (\widetilde{c})(E[A] \,|\, R' \,|\, C_1[Q]) & \equiv & C_2[Q]\end{array}$$

Hence we obtain $C_2[P] \,\mathcal{R}'\, C_2[Q]$ with $P \,\mathcal{R}\, Q$.

◦ The action is from C_1. That is, $C_1[P] \xrightarrow{(\widetilde{c})\overline{a}A} C_1'[P]$ and $C[P] \equiv R \,|\, C_1[P] \xrightarrow{(\widetilde{c})\overline{a}A} T_1 \equiv R \,|\, C_1'[P]$. In this case, $C[Q]$ simulates by $C[Q] \xrightarrow{(\widetilde{c})\overline{a}A} R \,|\, C_1'[Q]$. Now we have for every E as demanded, and $C_2 \stackrel{\text{def}}{=} (\widetilde{c})(E[A] \,|\, R \,|\, C_1'[])$.

$$\begin{array}{rcl}(\widetilde{c})(E[A] \,|\, R \,|\, C_1'[P]) & \equiv & C_2[P] \\ (\widetilde{c})(E[A] \,|\, R \,|\, C_1'[Q]) & \equiv & C_2[Q]\end{array}$$

Hence we obtain $C_2[P] \,\mathcal{R}'\, C_2[Q]$ with $P \,\mathcal{R}\, Q$.

◦ The action is from P. That is, $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$ and $C[P] \equiv R \,|\, C_1[P] \xrightarrow{(\widetilde{c})\overline{a}A} T_1 \equiv R \,|\, C_1[P']$. In this case, since $P \,\mathcal{R}\, Q$, we know that $Q \xRightarrow{(\widetilde{c})\overline{a}A} Q'$, and for every E as described in the definition, it holds that $(\widetilde{c})(E[A] \,|\, P') \equiv C'[P'']$, $(\widetilde{c})(E[A] \,|\, Q') \equiv C'[Q'']$, and $P'' \,\mathcal{R}\, Q''$. So $C[Q]$ simulates by $C[Q] \xRightarrow{(\widetilde{c})\overline{a}A} R \,|\, C_1[Q']$. Now, we have for every E, and some C_2 (which exists because P is in a firable position) and $C_3 \stackrel{\text{def}}{=} R \,|\, C_2[C'[]]$.

$$\begin{array}{rcl}(\widetilde{c})(E[A] \,|\, R \,|\, C_1[P']) & \equiv & R \,|\, (\widetilde{c})(E[A] \,|\, C_1[P']) \\ & \equiv & R \,|\, C_2[(\widetilde{c})(E[A] \,|\, P')] \\ & \equiv & R \,|\, C_2[C'[P'']] \\ & \equiv & C_3[P''] \\ \\ (\widetilde{c})(E[A] \,|\, R \,|\, C_1[Q']) & \equiv & R \,|\, (\widetilde{c})(E[A] \,|\, C_1[Q']) \\ & \equiv & R \,|\, C_2[(\widetilde{c})(E[A] \,|\, Q')] \\ & \equiv & R \,|\, C_2[C'[Q'']] \\ & \equiv & C_3[Q'']\end{array}$$

Hence we obtain $C_3[P''] \,\mathcal{R}'\, C_3[Q'']$ with $P'' \,\mathcal{R}\, Q''$.

◦ The action is from P and C_1. That is, $P \xrightarrow{\overline{a}A} P'$ and $C[P] \equiv R \,|\, C_1[P] \xrightarrow{(b)\overline{a}A} T_1 \equiv R \,|\, C_1'[P']$. Since $P \,\mathcal{R}\, Q$, we know that $Q \xRightarrow{\overline{a}A} Q'$, and for every E as described in the definition, it holds that $E[A] \,|\, P' \equiv C'[P'']$, $E[A] \,|\, Q' \equiv C'[Q'']$, and $P'' \,\mathcal{R}\, Q''$. So $C[Q]$ simulates by $C[Q] \xRightarrow{(b)\overline{a}A} R \,|\, C_1'[Q']$. Now, we have for

every E, and some C_2 (which exists because P is in a firable position) and $C_3 \stackrel{\text{def}}{=} R \mid (b)C_2[C'[]]$.

$$\begin{array}{rcl}(b)(E[A] \mid R \mid C_1'[P']) & \equiv & R \mid (b)(E[A] \mid C_1'[P']) \\ & \equiv & R \mid (b)C_2[E[A] \mid P'] \\ & \equiv & R \mid (b)C_2[C'[P'']] \\ & \equiv & C_3[P'']\end{array}$$

$$\begin{array}{rcl}(b)(E[A] \mid R \mid C_1'[Q']) & \equiv & R \mid (b)(E[A] \mid C_1'[Q']) \\ & \equiv & R \mid (b)C_2[E[A] \mid Q'] \\ & \equiv & R \mid (b)C_2[C'[Q'']] \\ & \equiv & C_3[Q'']\end{array}$$

Hence we obtain $C_3[P''] \,\mathcal{R}'\, C_3[Q'']$ with $P'' \,\mathcal{R}\, Q''$.

- C is $(d)C_1[\cdot]$. There are seven possibilities.
  - The action is from P, $\widetilde{c}$ is empty and d is b. That is, $P \xrightarrow{\overline{a}A} P'$ and $C[P] \xrightarrow{(b)\overline{a}A} C_1[P']$. Then since $P \,\mathcal{R}\, Q$, we know that $Q \stackrel{\overline{a}A}{\Longrightarrow} Q'$, and for every E as described in the definition, it holds that $E[A] \mid P' \equiv C'[P'']$, $E[A] \mid Q' \equiv C'[Q'']$, and $P'' \,\mathcal{R}\, Q''$. So $C[Q]$ simulates by $C[Q] \stackrel{(b)\overline{a}A}{\Longrightarrow} C_1[Q']$. Now, we have for every E, and some C_2 (which exists because P is in a firable position) and $C_3 \stackrel{\text{def}}{=} (b)C_2[C'[]]$.

$$\begin{array}{rcl}(b)(E[A] \mid C_1[P']) & \equiv & (b)C_2[E[A] \mid P'] \equiv (b)C_2[C'[P'']] \equiv C_3[P''] \\ (b)(E[A] \mid C_1[Q']) & \equiv & (b)C_2[E[A] \mid Q'] \equiv (b)C_2[C'[Q'']] \equiv C_3[Q'']\end{array}$$

Hence we obtain $C_3[P''] \,\mathcal{R}'\, C_3[Q'']$ with $P'' \,\mathcal{R}\, Q''$.

  - The action is from P, $\widetilde{c}$ is $\{b\}$ and d is not b. That is, $P \xrightarrow{(b)\overline{a}A} P'$ and $C[P] \xrightarrow{(b)\overline{a}A} (d)C_1[P']$. Then since $P \,\mathcal{R}\, Q$, we know that $Q \stackrel{(b)\overline{a}A}{\Longrightarrow} Q'$, and for every E as described in the definition, it holds that $(b)(E[A] \mid P') \equiv C'[P'']$, $(b)(E[A] \mid Q') \equiv C'[Q'']$, and $P'' \,\mathcal{R}\, Q''$. So $C[Q]$ simulates by $C[Q] \stackrel{(b)\overline{a}A}{\Longrightarrow} (d)C_1[Q']$. Now, we have for every E, and some C_2 (which exists because P is in a firable position) and $C_3 \stackrel{\text{def}}{=} (d)C_2[C'[]]$.

$$\begin{array}{rcl}(b)(E[A] \mid (d)C_1[P']) & \equiv & (d)C_2[(b)(E[A] \mid P')] \\ & \equiv & (d)C_2[C'[P'']] \\ & \equiv & C_3[P'']\end{array}$$

$$\begin{array}{rcl}(b)(E[A] \mid (d)C_1[Q']) & \equiv & (d)C_2[(b)(E[A] \mid Q')] \\ & \equiv & (d)C_2[C'[Q'']] \\ & \equiv & C_3[Q'']\end{array}$$

Hence we obtain $C_3[P''] \,\mathcal{R}'\, C_3[Q'']$ with $P'' \,\mathcal{R}\, Q''$.

  - The action is from P and C_1, $\widetilde{c}$ is $\{b\}$ and d is not b. That is, $P \xrightarrow{\overline{a}A} P'$ and $C[P] \xrightarrow{(b)\overline{a}A} (d)C_1'[P']$. Then since $P \,\mathcal{R}\, Q$, we know that $Q \stackrel{\overline{a}A}{\Longrightarrow} Q'$, and for every E as described in the definition, it holds that $E[A] \mid P' \equiv C'[P'']$, $E[A] \mid Q' \equiv C'[Q'']$, and $P'' \,\mathcal{R}\, Q''$. So $C[Q]$ simulates by $C[Q] \stackrel{(b)\overline{a}A}{\Longrightarrow} (d)C_1'[Q']$. Now,

we have for every E, and some C_2 (which exists because P is in a firable position) and $C_3 \stackrel{\text{def}}{=} (bd)C_2[C'[]]$.

$$\begin{array}{rcl}(b)(E[A] \mid (d)C_1'[P']) & \equiv & (bd)C_2[E[A] \mid P'] \\ & \equiv & (bd)C_2[C'[P'']] \\ & \equiv & C_3[P''] \end{array}$$

$$\begin{array}{rcl}(b)(E[A] \mid (d)C_1'[Q']) & \equiv & (bd)C_2[E[A] \mid Q'] \\ & \equiv & (bd)C_2[C'[Q'']] \\ & \equiv & C_3[Q''] \end{array}$$

Hence we obtain $C_3[P'']\,\mathcal{R}'\,C_3[Q'']$ with $P''\,\mathcal{R}\,Q''$.

- The action is from P, $\widetilde{c}$ is empty and d is not b. That is, $P \xrightarrow{\overline{a}A} P'$ and $C[P] \xrightarrow{\overline{a}A} (d)C_1[P']$. Then since $P\,\mathcal{R}\,Q$, we know that $Q \stackrel{\overline{a}A}{\Longrightarrow} Q'$, and for every E as stipulated in the definition, it holds that $E[A] \mid P' \equiv C'[P'']$, $E[A] \mid Q' \equiv C'[Q'']$, and $P''\,\mathcal{R}\,Q''$. So $C[Q]$ simulates by $C[Q] \stackrel{\overline{a}A}{\Longrightarrow} (d)C_1[Q']$. Now, we have for every E, and some C_2 (which exists because P is in a firable position) and $C_3 \stackrel{\text{def}}{=} (d)C_2[C'[]]$.

$$\begin{array}{rcl}E[A] \mid (d)C_1[P'] & \equiv & (d)C_2[E[A] \mid P'] \equiv (d)C_2[C'[P'']] \equiv C_3[P''] \\ E[A] \mid (d)C_1[Q'] & \equiv & (d)C_2[E[A] \mid Q'] \equiv (d)C_2[C'[Q'']] \equiv C_3[Q''] \end{array}$$

Hence we obtain $C_3[P'']\,\mathcal{R}'\,C_3[Q'']$ with $P''\,\mathcal{R}\,Q''$.

- The action is from C_1, $\widetilde{c}$ is empty and d is b. That is, we have $C_1[P] \xrightarrow{\overline{a}A} C_1'[P]$ and $C[P] \xrightarrow{(b)\overline{a}A} C_1'[P]$. Then $C[Q]$ simulates by $C[Q] \xrightarrow{(b)\overline{a}A} C_1'[Q]$. Now for every E and $C_2 \stackrel{\text{def}}{=} (b)(E[A] \mid C_1'[])$ we have

$$\begin{array}{rcl}(b)(E[A] \mid C_1'[P]) & \equiv & C_2[P] \\ (b)(E[A] \mid C_1'[Q]) & \equiv & C_2[Q] \end{array}$$

Hence we obtain $C_2[P]\,\mathcal{R}'\,C_2[Q]$ with $P\,\mathcal{R}\,Q$.

- The action is from C_1, $\widetilde{c}$ is $\{b\}$ and d is not b. That is, we have $C_1[P] \xrightarrow{(b)\overline{a}A} C_1'[P]$ and $C[P] \xrightarrow{(b)\overline{a}A} (d)C_1'[P]$. Then $C[Q]$ simulates by $C[Q] \xrightarrow{(b)\overline{a}A} (d)C_1'[Q]$. Now, we have for every E, and $C_2 \stackrel{\text{def}}{=} (b)(E[A] \mid (d)C_1'[])$.

$$\begin{array}{rcl}(b)(E[A] \mid (d)C_1'[P]) & \equiv & C_2[P] \\ (b)(E[A] \mid (d)C_1'[Q]) & \equiv & C_2[Q] \end{array}$$

Hence we obtain $C_2[P]\,\mathcal{R}'\,C_2[Q]$ with $P\,\mathcal{R}\,Q$.

- The action is from C_1, $\widetilde{c}$ is empty and d is not b. That is, we have $C_1[P] \xrightarrow{\overline{a}A} C_1'[P]$ and $C[P] \xrightarrow{\overline{a}A} (d)C_1'[P]$. Then $C[Q]$ simulates by $C[Q] \xrightarrow{\overline{a}A} (d)C_1'[Q]$. Now for every E and $C_2 \stackrel{\text{def}}{=} E[A] \mid (d)C_1'[]$, we have

$$\begin{array}{rcl}E[A] \mid (d)C_1'[P] & \equiv & C_2[P] \\ E[A] \mid (d)C_1'[Q] & \equiv & C_2[Q] \end{array}$$

Hence we obtain $C_2[P]\,\mathcal{R}'\,C_2[Q]$ with $P\,\mathcal{R}\,Q$.

3. $C[P] \xrightarrow{\tau} T_1$.

- The case when C is $[\cdot]$ is immediate. The cases when C is $a(X).C_1[\cdot]$, $\overline{a}(C_1[\cdot]).R$, $\overline{a}A_1.C_1[\cdot]$, $\langle X\rangle C_1[\cdot]$, $\langle x\rangle C_1[\cdot]$ are not possible. The cases when C is $C_1[\cdot]\langle A_1\rangle$, $C_1[\cdot]\langle d\rangle$ fall onto the other cases after possible application.
- C is $R \mid C_1[\cdot]$. Several possibilities. For two-part communication, we only consider one subcase in which one part makes input and the other makes output, and omit the other similar (and actually simpler) symmetric subcase in which their roles are switched.
  - τ is from R. Suppose $R \xrightarrow{\tau} R'$, and $C[P] \xrightarrow{\tau} R' \mid C_1[P]$. Then $C[Q]$ simulates by $C[Q] \xrightarrow{\tau} R' \mid C_1[Q]$. So we set $C' \stackrel{\text{def}}{=} R' \mid C_1[]$, and have $C'[P] \,\mathcal{R}'\, C'[Q]$ in which $P \,\mathcal{R}\, Q$.
  - τ is from C_1. Suppose $C_1[P] \xrightarrow{\tau} C_1'[P]$, and $C[P] \xrightarrow{\tau} R \mid C_1'[P]$. Then $C[Q]$ simulates by $C[Q] \xrightarrow{\tau} R \mid C_1'[Q]$. So we set $C' \stackrel{\text{def}}{=} R \mid C_1'[]$, and have $C'[P] \,\mathcal{R}'\, C'[Q]$ in which $P \,\mathcal{R}\, Q$.
  - τ is from P. Suppose $P \xrightarrow{\tau} P'$, and $C[P] \xrightarrow{\tau} R \mid C_1[P']$. Since $P \,\mathcal{R}\, Q$, we know $Q \Longrightarrow Q'$ and $P' \equiv C_2[P'']$ and $Q' \equiv C_2[Q'']$ for some P'', Q'' and context C_2 that is an encoding, and $P'' \,\mathcal{R}\, Q''$. So $C[Q]$ simulates by $C[Q] \Longrightarrow R \mid C_1[Q']$. Now consider the following pair:

$$\begin{array}{rcl} R \mid C_1[P'] & \equiv & R \mid C_1[C_2[P'']] \\ R \mid C_1[Q'] & \equiv & R \mid C_1[C_2[Q'']] \end{array}$$

    Now setting $C_3 \stackrel{\text{def}}{=} R \mid C_1[C_2[]]$ to be the context closes this case as required, because $C_3[P''] \,\mathcal{R}'\, C_3[Q'']$ with $P'' \,\mathcal{R}\, Q''$.
  - τ is from interaction between R and C_1. Suppose $C_1[P] \xrightarrow{(\widetilde{c})\overline{a}A} C_1'[P]$ for some C_1, in which A is $\langle Z\rangle(Z\langle b\rangle)$ and $\widetilde{c}$ is either empty or $\{b\}$, $R \xrightarrow{a(A)} R'$, and $C[P] \xrightarrow{\tau} (\widetilde{c})(R' \mid C_1'[P])$. Then $C[Q] \xrightarrow{\tau} (\widetilde{c})(R' \mid C_1'[Q])$. We close this case by setting the context to be $C_2 \stackrel{\text{def}}{=} (\widetilde{c})(R' \mid C_1'[])$, and having $C_2[P] \,\mathcal{R}'\, C_2[Q]$ with $P \,\mathcal{R}\, Q$.
  - τ is from interaction between R and P. Suppose $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$ in which A is $\langle Z\rangle(Z\langle b\rangle)$ and $\widetilde{c}$ is either empty or $\{b\}$, $R \xrightarrow{a(A)} R'$, and $C[P] \xrightarrow{\tau} (\widetilde{c})(R' \mid C_1[P']) \equiv C_2[(\widetilde{c})(R' \mid P')]$ for some C_2 since P is in a firable position. Then since $P \mathcal{R} Q$, we know that $Q \xRightarrow{(\widetilde{c})\overline{a}A} Q'$, and for every E as described in the definition, it holds that $(\widetilde{c})(E[A] \mid P') \equiv C'[P'']$, $(\widetilde{c})(E[A] \mid Q') \equiv C'[Q'']$, and $P'' \,\mathcal{R}\, Q''$. So $C[Q]$ simulates by $C[Q] \xRightarrow{\tau} (\widetilde{c})(R' \mid C_1[Q']) \equiv C_2[(\widetilde{c})(R' \mid Q')]$. It meets the bisimulation requirement because one can take $E[A]$ to be R' and obtain the following pair for some C_3, P''', Q''' with $P''' \,\mathcal{R}\, Q'''$.

$$\begin{array}{rcl} (\widetilde{c})(R' \mid C_1[P']) & \equiv & C_2[(\widetilde{c})(R' \mid P')] \equiv C_2[C_3[P''']] \\ (\widetilde{c})(R' \mid C_1[Q']) & \equiv & C_2[(\widetilde{c})(R' \mid Q')] \equiv C_2[C_3[Q''']] \end{array}$$

    Now we set $C_4 \stackrel{\text{def}}{=} C_2[C_3[]]$ to be the context, and have as required $C_4[P'''] \,\mathcal{R}'\, C_4[Q''']$ with $P''' \,\mathcal{R}\, Q'''$.
  - τ is from interaction between R and P together with C_1. Suppose $P \xrightarrow{\overline{a}A} P'$ in which A is $\langle Z\rangle(Z\langle b\rangle)$ and $\widetilde{c}$ is $\{b\}$, $R \xrightarrow{a(A)} R'$, and $C[P] \xrightarrow{\tau} (b)(R' \mid C_1'[P']) \equiv (b)C_2[R' \mid P']$ for some C_2 since P is in a firable position. Then since $P \mathcal{R} Q$, we know that $Q \xRightarrow{\overline{a}A} Q'$, and for every E as described in the definition, it holds that $E[A] \mid P' \equiv C'[P'']$, $E[A] \mid Q' \equiv C'[Q'']$, and $P'' \,\mathcal{R}\, Q''$. So $C[Q]$ simulates by $C[Q] \xRightarrow{\tau} (b)(R' \mid C_1'[Q']) \equiv (b)C_2[R' \mid Q']$. It meets the bisimulation requirement because one can take $E[A]$ to be R' and obtain the following pair for some C_3, P''', Q''' with $P''' \,\mathcal{R}\, Q'''$.

$$\begin{array}{rcl} (b)(R' \mid C_1'[P']) & \equiv & (b)C_2[R' \mid P'] \equiv (b)C_2[C_3[P''']] \\ (b)(R' \mid C_1'[Q']) & \equiv & (b)C_2[R' \mid Q'] \equiv (b)C_2[C_3[Q''']] \end{array}$$

    Now we set $C_4 \stackrel{\text{def}}{=} (b)C_2[C_3[]]$ to be the context, and have as required $C_4[P'''] \,\mathcal{R}'\, C_4[Q''']$ with $P''' \,\mathcal{R}\, Q'''$.

- τ is from interaction between P and C_1. This case is similar to the last one. Notice that C_1 here has a role similar to C in the last case. Suppose $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$ in which A is $\langle Z\rangle(Z\langle b\rangle)$ and $\widetilde{c}$ is either empty or $\{b\}$, $C_1[P] \xrightarrow{a(A)} C_1'[P]$, and $C[P] \xrightarrow{\tau} R \,|\, (\widetilde{c})(C_1'[P']) \equiv R \,|\, C_2[(\widetilde{c})(T \,|\, P')]$ for some C_2 and T (which possibly has A occurrence) since P is in a firable position. Then since $P\mathcal{R}Q$, we know that $Q \xRightarrow{(\widetilde{c})\overline{a}A} Q'$, and for every E as described in the definition, it holds that $(\widetilde{c})(E[A] \,|\, P') \equiv C'[P'']$, $(\widetilde{c})(E[A] \,|\, Q') \equiv C'[Q'']$, and $P'' \,\mathcal{R}\, Q''$. So $C[Q]$ simulates by $C[Q] \xRightarrow{\tau} R \,|\, (\widetilde{c})(C_1'[Q']) \equiv R \,|\, C_2[(\widetilde{c})(T \,|\, Q')]$. It meets the bisimulation requirement because one can take $E[A]$ to be T and obtain the following pair for some C_3, P''', Q''' with $P''' \,\mathcal{R}\, Q'''$.

$$\begin{array}{rcl} R \,|\, (\widetilde{c})(C_1'[P']) & \equiv & R \,|\, C_2[(\widetilde{c})(T \,|\, P')] \equiv R \,|\, C_2[C_3[P''']] \\ R \,|\, (\widetilde{c})(C_1'[Q']) & \equiv & R \,|\, C_2[(\widetilde{c})(T \,|\, Q')] \equiv R \,|\, C_2[C_3[Q''']] \end{array}$$

  Now we set $C_4 \stackrel{\text{def}}{=} R \,|\, C_2[C_3[]]$ to be the context, and have as required $C_4[P'''] \,\mathcal{R}'\, C_4[Q''']$ with $P''' \,\mathcal{R}\, Q'''$.

- C is $(d)C_1[\cdot]$. The action $C[P] \equiv (d)C_1[P] \xrightarrow{\tau} T_1$ must come from $C_1[P] \xrightarrow{\tau} T_2$ such that $(d)T_2$. There are several subcases.
  - τ is from C_1. Suppose $C_1[P] \xrightarrow{\tau} C_1'[P]$, and $C[P] \xrightarrow{\tau} (d)C_1'[P]$. Then $C[Q]$ simulates by $C[Q] \xrightarrow{\tau} (d)C_1'[Q]$. So we set $C' \stackrel{\text{def}}{=} (d)C_1'[]$, and have $C'[P] \,\mathcal{R}'\, C'[Q]$ in which $P \,\mathcal{R}\, Q$.
  - τ is from P. Suppose $P \xrightarrow{\tau} P'$, and $C[P] \xrightarrow{\tau} (d)C_1[P']$. Since $P \,\mathcal{R}\, Q$, we know $Q \Longrightarrow Q'$ and $P' \equiv C_2[P'']$ and $Q' \equiv C_2[Q'']$ for some P'', Q'' and context C_2 that is an encoding, and $P'' \,\mathcal{R}\, Q''$. So $C[Q]$ simulates by $C[Q] \Longrightarrow (d)C_1[Q']$. Now consider the following pair:

$$\begin{array}{rcl} (d)C_1[P'] & \equiv & (d)C_1[C_2[P'']] \\ (d)C_1[Q'] & \equiv & (d)C_1[C_2[Q'']] \end{array}$$

    Now setting $C_3 \stackrel{\text{def}}{=} (d)C_1[C_2[]]$ to be the context closes this case, because $C_3[P''] \,\mathcal{R}'\, C_3[Q'']$ with $P'' \,\mathcal{R}\, Q''$.
  - τ is from interaction between P andC_1. Suppose $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$ in which A is $\langle Z\rangle(Z\langle b\rangle)$ and $\widetilde{c}$ is either empty or $\{b\}$, $C_1[P] \xrightarrow{a(A)} C_1'[P]$, and $C[P] \xrightarrow{\tau} (d)((\widetilde{c})(C_1'[P'])) \equiv (d)C_2[(\widetilde{c})(T \,|\, P')]$ for some C_2 and T (which possibly has A occurrence) since P is in a firable position. Then since $P\mathcal{R}Q$, we know that $Q \xRightarrow{(\widetilde{c})\overline{a}A} Q'$, and for every E as described in the definition, it holds that $(\widetilde{c})(E[A] \,|\, P') \equiv C'[P'']$, $(\widetilde{c})(E[A] \,|\, Q') \equiv C'[Q'']$, and $P'' \,\mathcal{R}\, Q''$. So $C[Q]$ simulates by $C[Q] \xRightarrow{\tau} (d)((\widetilde{c})(C_1'[Q'])) \equiv (d)C_2[(\widetilde{c})(T \,|\, Q')]$. It meets the bisimulation requirement because one can take $E[A]$ to be T and obtain the following pair for some C_3, P''', Q''' with $P''' \,\mathcal{R}\, Q'''$.

$$\begin{array}{rcl} (d)((\widetilde{c})(C_1'[P'])) & \equiv & (d)C_2[(\widetilde{c})(T \,|\, P')] \equiv (d)C_2[C_3[P''']] \\ (d)((\widetilde{c})(C_1'[Q'])) & \equiv & (d)C_2[(\widetilde{c})(T \,|\, Q')] \equiv (d)C_2[C_3[Q''']] \end{array}$$

    Now we set $C_4 \stackrel{\text{def}}{=} (d)C_2[C_3[]]$ to be the context as required, and have $C_4[P'''] \,\mathcal{R}'\, C_4[Q''']$ with $P''' \,\mathcal{R}\, Q'''$.

□

Appendix C. Proofs for Section 4

C.1 Proof of the operational correspondence of the encoding in Section 4

We give the proof of Lemma 4.2 in Section 4.1.

Proof of Lemma 4.2. We prove the clauses by induction on P. We caution that for the sake of simplicity, we always assume no name capture up-to α-conversion.

- P is 0. This is trivial.
- P is $a(x).P_1 \equiv C[a(x).P_1]$ in which $C \stackrel{\text{def}}{=} [\cdot]$.
  - Input. We have $P \xrightarrow{a(b)} P_1\{b/x\} \stackrel{\text{def}}{=} P' \equiv C[P_1\{b/x\}]$ and $C[0] \mid P_1\{b/x\} \equiv P'$. Then $[\![P]\!] \equiv \overline{a}[\langle x\rangle [\![P_1]\!]].0 \xrightarrow{\overline{a}[\langle x\rangle([\![P_1]\!])]} 0 \stackrel{\text{def}}{=} T$, and $T \equiv [\![C[0]]\!] \equiv 0$.
- P is $\overline{a}b.P_1$.
  - Output. We have $[\![P]\!] \equiv a(Y).(Y\langle b\rangle \mid [\![P_1]\!]) \xrightarrow{a(A)} A\langle b\rangle \mid [\![P_1]\!] \stackrel{\text{def}}{=} T$, as required.
- P is $P_1 \mid P_2$.
  - Input. Suppose $P_1 \xrightarrow{a(b)} P_1'$ (the case P_2 does is similar) and $P \xrightarrow{a(b)} P_1' \mid P_2 \equiv P'$. By ind. hyp., $P_1 \sim_g (\widetilde{e_1})C'[a(x).P_3]$ and $P_1' \equiv (\widetilde{e_1})C'[P_3\{b/x\}] \equiv (\widetilde{e_1})(C'[0] \mid P_3\{b/x\})$ for some context C', P_3 and $\widetilde{e_1}$ being the bound names shared between P_3 and its context within P_1, and $[\![P_1]\!] \xrightarrow{(\widetilde{e_1})\overline{a}[\langle x\rangle([\![P_3]\!])]} T_1 \equiv [\![C'[0]]\!]$. Defining $C \stackrel{\text{def}}{=} C' \mid P_2$, we have $P \sim_g (\widetilde{e_1})C[a(x).P_3]$ and $P' \equiv (\widetilde{e_1})C[P_3\{b/x\}] \equiv (\widetilde{e_1})(C[0] \mid P_3\{b/x\})$. Moreover

$$\begin{array}{rl} [\![P]\!] \equiv & [\![P_1]\!] \mid [\![P_2]\!] \\ \xrightarrow{(\widetilde{e_1})\overline{a}[\langle x\rangle([\![P_3]\!])]} & T_1 \mid [\![P_2]\!] \stackrel{\text{def}}{=} T \\ \equiv & [\![C'[0]]\!] \mid [\![P_2]\!] \\ \equiv & [\![C'[0] \mid P_2]\!] \equiv [\![C[0]]\!] \end{array}$$

  - Output. Suppose $P_1 \xrightarrow{\overline{a}b} P_1'$ (the case P_2 does is similar) and $P \xrightarrow{\overline{a}b} P_1' \mid P_2 \equiv P'$. By ind. hyp., $[\![P_1]\!] \xrightarrow{a(A)} T_1$ and $T_1 \equiv A\langle b\rangle \mid [\![P_1']\!]$. Then we have

$$\begin{array}{rl} [\![P]\!] \equiv & [\![P_1]\!] \mid [\![P_2]\!] \\ \xrightarrow{a(A)} & T_1 \mid [\![P_2]\!] \stackrel{\text{def}}{=} T \\ \equiv & A\langle b\rangle \mid [\![P_1']\!] \mid [\![P_2]\!] \\ \equiv & A\langle b\rangle \mid [\![P_1' \mid P_2]\!] \equiv A\langle b\rangle \mid [\![P']\!] \end{array}$$

  - Bound output. Suppose $P_1 \xrightarrow{\overline{a}(b)} P_1'$ and $b \notin \text{fn}(P_2)$, and $P \xrightarrow{\overline{a}b} P_1' \mid P_2 \equiv P'$ (the case P_2 does is similar). By ind. hyp., $[\![P_1]\!] \xrightarrow{a(A)} T_1$ and $T_1 \equiv (b)(A\langle b\rangle \mid [\![P_1']\!])$. Then we have

$$\begin{array}{rl} [\![P]\!] \equiv & [\![P_1]\!] \mid [\![P_2]\!] \\ \xrightarrow{a(A)} & T_1 \mid [\![P_2]\!] \stackrel{\text{def}}{=} T \\ \equiv & (b)(A\langle b\rangle \mid [\![P_1']\!]) \mid [\![P_2]\!] \\ \equiv & (b)(A\langle b\rangle \mid [\![P_1']\!] \mid [\![P_2]\!]) \\ \equiv & (b)(A\langle b\rangle \mid [\![P_1' \mid P_2]\!]) \\ \equiv & (b)(A\langle b\rangle \mid [\![P']\!]) \end{array}$$

  - Interaction (τ). The most interesting and hard case is when P_1 and P_2 engages a communication (of a bound name). Consider, for example, the case $P_1 \xrightarrow{a(b)} P_1'$ and $P_2 \xrightarrow{\overline{a}(b)} P_2'$ ($b \notin \text{fn}(P_2)$ wolg), and $P \xrightarrow{\tau} P' \equiv (b)(P_1' \mid P_2')$. By ind. hyp., $P_1 \sim_g (\widetilde{e_1})C'[a(x).P_3]$ and $P_1' \equiv (\widetilde{e_1})C'[P_3\{b/x\}] \equiv (\widetilde{e_1})(C'[0] \mid P_3\{b/x\})$ for some context C', P_3 and $\widetilde{e_1}$ being the bound names shared between P_3 and its context within P_1, and $[\![P_1]\!] \xrightarrow{(\widetilde{e_1})\overline{a}[\langle x\rangle([\![P_3]\!])]} T_1 \equiv [\![C'[0]]\!]$; $[\![P_2]\!] \xrightarrow{a(A)} T_2$ and $T_2 \equiv (b)(A\langle b\rangle \mid [\![P_2']\!])$. Take A

as $\langle x\rangle([\![P_3]\!])$, and we have

$$\begin{array}{rl}
[\![P]\!] \equiv & [\![P_1]\!] \mid [\![P_2]\!] \\
\xrightarrow{\tau} & (\widetilde{e_1})(T_1 \mid T_2) \stackrel{\text{def}}{=} T \\
\equiv & (\widetilde{e_1})([\![C'[0]]\!] \mid (b)((\langle x\rangle([\![P_3]\!]))\langle b\rangle \mid [\![P_2']\!])) \\
\equiv & (\widetilde{e_1})([\![C'[0]]\!] \mid (b)([\![P_3]\!]\{b/x\} \mid [\![P_2']\!])) \\
\equiv & (\widetilde{e_1})([\![C'[0]]\!] \mid (b)([\![P_3\{b/x\}]\!] \mid [\![P_2']\!])) \\
\equiv & (b\widetilde{e_1})([\![C'[0]]\!] \mid [\![P_3\{b/x\}]\!] \mid [\![P_2']\!]) \\
\equiv & (b\widetilde{e_1})([\![C'[0] \mid P_3\{b/x\} \mid P_2']\!]) \\
\equiv & (b\widetilde{e_1})([\![C'[0] \mid P_3\{b/x\} \mid P_2']\!]) \\
\equiv & (b)([\![(\widetilde{e_1})(C'[0] \mid P_3\{b/x\}) \mid P_2']\!]) \\
\equiv & (b)([\![P_1' \mid P_2']\!])
\end{array}$$

- P is $(c)P_1$.
  - Input. Suppose $P_1 \xrightarrow{a(b)} P_1'$ and $P \xrightarrow{a(b)} (c)P_1' \equiv P'$. By ind. hyp., $P_1 \sim_g (\widetilde{e_1})C'[a(x).P_3]$ and $P_1' \equiv (\widetilde{e_1})C'[P_3\{b/x\}] \equiv (\widetilde{e_1})(C'[0] \mid P_3\{b/x\})$ for some context C', P_3 and $\widetilde{e_1}$ being the bound names shared between P_3 and its context within P_1, and $[\![P_1]\!] \xrightarrow{(\widetilde{e_1})\overline{a}[\langle x\rangle([\![P_3]\!])]} T_1 \equiv [\![C'[0]]\!]$. There are two subcases.

  (i) If $c \in \text{fn}(P_3)$, we set $C \stackrel{\text{def}}{=} C'$ and have $P \sim_g (\widetilde{e_1}c)C[a(x).P_3]$ and $P' \equiv (\widetilde{e_1}c)C[P_3\{b/x\}] \equiv (\widetilde{e_1}c)(C[0] \mid P_3\{b/x\})$, and moreover

$$\begin{array}{rl}
[\![P]\!] \equiv & (c)[\![P_1]\!] \\
\xrightarrow{(\widetilde{e_1}c)\overline{a}[\langle x\rangle([\![P_3]\!])]} & T_1 \stackrel{\text{def}}{=} T \\
\equiv & [\![C'[0]]\!] \equiv [\![C[0]]\!]
\end{array}$$

  (ii) If $c \notin \text{fn}(P_3)$, we set $C \stackrel{\text{def}}{=} (c)C'$. Then we have $P \sim_g (\widetilde{e_1})C[a(x).P_3]$ and $P' \equiv (\widetilde{e_1})C[P_3\{b/x\}] \equiv (\widetilde{e_1})((c)C'[0] \mid P_3\{b/x\}) \equiv (\widetilde{e_1})(C[0] \mid P_3\{b/x\})$, and moreover

$$\begin{array}{rl}
[\![P]\!] \equiv & (c)[\![P_1]\!] \\
\xrightarrow{(\widetilde{e_1})\overline{a}[\langle x\rangle([\![P_3]\!])]} & (c)T_1 \stackrel{\text{def}}{=} T \\
\equiv & (c)[\![C'[0]]\!] \\
\equiv & [\![(c)C'[0]]\!] \equiv [\![C[0]]\!]
\end{array}$$

  - Output. There are two subcases.

  (i) $P_1 \xrightarrow{\overline{a}b} P_1'$ and $P \xrightarrow{\overline{a}b} (c)P_1' \equiv P'$. By ind. hyp., $[\![P_1]\!] \xrightarrow{a(A)} T_1 \equiv A\langle b\rangle \mid [\![P_1']\!]$. Then we have

$$\begin{array}{rl}
[\![P]\!] \equiv & (c)[\![P_1]\!] \\
\xrightarrow{a(A)} & (c)T_1 \stackrel{\text{def}}{=} T \\
\equiv & (c)(A\langle b\rangle \mid [\![P_1']\!]) \\
\equiv & A\langle b\rangle \mid (c)[\![P_1']\!] \\
\equiv & A\langle b\rangle \mid [\![(c)P_1']\!] \equiv A\langle b\rangle \mid [\![P']\!]
\end{array}$$

(ii) $P_1 \xrightarrow{\overline{a}c} P_1'$ and $P \xrightarrow{\overline{a}(c)} P_1' \equiv P'$. By ind. hyp., we know $[\![P_1]\!] \xrightarrow{a(A)} T_1 \equiv A\langle c\rangle \mid [\![P_1']\!]$. Then we have

$$\begin{array}{rl} [\![P]\!] \equiv & (c)[\![P_1]\!] \\ \xrightarrow{a(A)} & (c)T_1 \stackrel{\text{def}}{=} T \\ \equiv & (c)(A\langle c\rangle \mid [\![P_1']\!]) \\ \equiv & (c)(A\langle c\rangle \mid [\![P']\!]) \end{array}$$

- Bound output. Suppose $P_1 \xrightarrow{\overline{a}(b)} P_1'$ and $P \xrightarrow{\overline{a}(b)} (c)P_1' \equiv P'$. By ind. hyp., $[\![P_1]\!] \xrightarrow{a(A)} T_1 \equiv (b)(A\langle b\rangle \mid [\![P_1']\!])$. Then we have

$$\begin{array}{rl} [\![P]\!] \equiv & (c)[\![P_1]\!] \\ \xrightarrow{a(A)} & (c)T_1 \stackrel{\text{def}}{=} T \\ \equiv & (c)(b)(A\langle b\rangle \mid [\![P_1']\!]) \\ \equiv & (b)(A\langle b\rangle \mid (c)[\![P_1']\!]) \\ \equiv & (b)(A\langle b\rangle \mid [\![(c)P_1']\!]) \\ \equiv & (b)(A\langle b\rangle \mid [\![P']\!]) \end{array}$$

- Interaction (τ). Suppose $P_1 \xrightarrow{\tau} P_1'$ and $P \xrightarrow{\tau} (c)P_1' \equiv P'$. By ind. hyp., $[\![P_1]\!] \xrightarrow{\tau} T_1 \equiv [\![P_1']\!]$. Then we have

$$\begin{array}{rl} [\![P]\!] \equiv & (c)[\![P_1]\!] \\ \xrightarrow{\tau} & (c)T_1 \stackrel{\text{def}}{=} T \\ \equiv & (c)[\![P_1']\!] \\ \equiv & [\![(c)P_1']\!] \equiv [\![P']\!] \end{array}$$

- P is $!a(x).P_1 \sim_g C[a(x).P_1]$ in which $C \stackrel{\text{def}}{=} [\cdot] \mid !a(x).P_1$.
  - Input. We have $P \xrightarrow{a(b)} P_1\{b/x\} \mid !a(x).P_1 \stackrel{\text{def}}{=} P' \equiv C[P_1\{b/x\}]$ and $P' \equiv C[0] \mid P_1\{b/x\}$. Then $[\![P]\!] \equiv !\overline{a}[\langle x\rangle [\![P_1]\!]] \xrightarrow{\overline{a}[\langle x\rangle([\![P_1]\!])]} 0 \mid !\overline{a}[\langle x\rangle [\![P_1]\!]] \stackrel{\text{def}}{=} T$, and $T \equiv [\![C[0]]\!]$.

□

C.2 Proof of the completeness of the encoding in Section 4

Proof of Lemma 4.5. We show that relation $\mathcal{R}$ below is a local bisimulation up-to $\equiv$.

$$\mathcal{R} \stackrel{\text{def}}{=} \{(P, Q) \mid [\![P]\!] \approx_{ct} [\![Q]\!]\} \cup \approx_g$$

Assume $P \,\mathcal{R}\, Q$ and $P \xrightarrow{\lambda} P'$. We proceed by a case analysis.

- Input: $P \xrightarrow{a(b)} P'$. By Lemma 4.2, we know that $P \sim_g (\widetilde{e})C[a(x).P_1]$ and $P' \equiv (\widetilde{e})C[P_1\{b/x\}] \equiv (\widetilde{e})(C[0] \mid P_1\{b/x\})$ for some context C, P_1 and $\widetilde{e}$ being the bound names shared between P_1 and its context, and $[\![P]\!] \xrightarrow{(\widetilde{e})\overline{a}[\langle x\rangle([\![P_1]\!])]} T \equiv [\![C[0]]\!]$. Since $[\![P]\!] \approx_{ct} [\![Q]\!]$, we know that $[\![Q]\!] \xRightarrow{(\widetilde{e'})\overline{a}[\langle x\rangle([\![Q_1]\!])]} T'$ and for every $E[X]$ with $\mathrm{fn}(E) \cap \widetilde{e}\widetilde{e'} = \emptyset$, it holds that

$$(\widetilde{e})(E[\langle x\rangle([\![P_1]\!])] \mid T) \;\approx_{ct}\; (\widetilde{e'})(E[\langle x\rangle([\![Q_1]\!])] \mid T') \tag{C.1}$$

  By Lemma 4.3, we know that $Q \xRightarrow{a(b)} Q'$ with $Q \sim_g (\widetilde{e'})C'[a(x).Q_1]$ and $Q' \equiv (\widetilde{e'})C'[Q_1\{b/x\}] \equiv (\widetilde{e'})(C'[0] \mid Q_1\{b/x\})$ for some context C' with $\widetilde{e}$ being the bound names shared between Q_1 and its context,

and $[\![C'[0]]\!] \Longrightarrow T'' \approx_{ct} T'$, where we notice that there might be a few τ's distance between $[\![C'[0]]\!]$ and T' due to the weak transition by $[\![Q]\!]$. Still by Lemma 4.3, we know that there exists some context C'' s.t. $C'[0] \Longrightarrow C''[0]$ with $[\![C''[0]]\!] \approx_{ct} T'' \approx_{ct} T'$, and as such we also know that $Q' \Longrightarrow (\widetilde{e'})(C''[0] \mid Q_1\{b/x\}) \stackrel{\text{def}}{=} Q''$, so $Q \stackrel{a(b)}{\Longrightarrow} Q''$. We need to prove $P' \mathcal{R} Q''$, *i.e.*, $[\![P']\!] \approx_{ct} [\![Q'']\!]$. Through setting $E[X]$ as $X\langle b\rangle$, equation (C.1) turns into

$$(\widetilde{e})((\langle x\rangle([\![P_1]\!]))\langle b\rangle \mid T) \;\approx_{ct}\; (\widetilde{e'})((\langle x\rangle([\![Q_1]\!]))\langle b\rangle \mid T')$$

which is exactly

$$(\widetilde{e})([\![P_1\{b/x\}]\!] \mid T) \;\approx_{ct}\; (\widetilde{e'})([\![Q_1\{b/x\}]\!] \mid T')$$

Since we already know that $T \equiv [\![C[0]]\!]$ and $T' \approx_{ct} [\![C''[0]]\!]$, we have as needed

$$[\![P']\!] \;\equiv\; (\widetilde{e})([\![P_1\{b/x\}]\!] \mid [\![C[0]]\!]) \;\approx_{ct}\; (\widetilde{e'})([\![Q_1\{b/x\}]\!] \mid [\![C''[0]]\!]) \;\equiv\; [\![Q'']\!]$$

- Output: $P \xrightarrow{\overline{a}b} P'$. By Lemma 4.2, $[\![P]\!] \xrightarrow{a(A)} T \equiv A\langle b\rangle \mid [\![P']\!]$. Since $[\![P]\!] \approx_{ct} [\![Q]\!]$, we know that $[\![Q]\!] \stackrel{a(A)}{\Longrightarrow} T' \approx_{ct} T$. By Lemma 4.3, it must be that $Q \stackrel{\overline{a}b}{\Longrightarrow} Q'$ (otherwise $[\![P]\!]$ and $[\![Q]\!]$ would be distinguishable, similar to the case in Lemma 3.10), and $A\langle b\rangle \mid [\![Q']\!] \Longrightarrow T'' \approx_{ct} T'$. Setting A to be $\langle x\rangle 0$, we have

$$\begin{aligned} &[\![P]\!] \xrightarrow{a(\langle x\rangle 0)} T \equiv 0 \mid [\![P']\!] \equiv [\![P']\!] \\ &[\![Q']\!] \equiv 0 \mid [\![Q']\!] \Longrightarrow T'' \approx_{ct} T' \end{aligned} \tag{C.2}$$

From (C.2) and Lemma 4.3, we know that $Q' \Longrightarrow Q''$ and $[\![Q'']\!] \approx_{ct} T'' \approx_{ct} T'$. So we have $Q \stackrel{\overline{a}b}{\Longrightarrow} Q''$ and need to prove $P' \mathcal{R} Q''$, *i.e.*, $[\![P']\!] \approx_{ct} [\![Q'']\!]$. This is immediate from the following equations.

$$[\![P']\!] \;\approx_{ct}\; T \;\approx_{ct}\; T' \;\approx_{ct}\; T'' \;\approx_{ct}\; [\![Q'']\!]$$

- Bound output: $P \xrightarrow{\overline{a}(b)} P'$. By Lemma 4.2, $[\![P]\!]$ can make an input over a; we set the input as the trigger $Tr_m^d \equiv \langle z\rangle \overline{m}[\langle Y\rangle(Y\langle z\rangle)]$ (m fresh). Then we have $[\![P]\!] \xrightarrow{a(Tr_m^d)} T \equiv (b)(Tr_m^d\langle b\rangle \mid [\![P']\!])$. Since $[\![P]\!] \approx_{ct} [\![Q]\!]$, we know that $[\![Q]\!] \stackrel{a(Tr_m^d)}{\Longrightarrow} T' \approx_{ct} T$. By Lemma 4.3, it must be that $Q \stackrel{\overline{a}(b)}{\Longrightarrow} Q'$ (otherwise $[\![P]\!]$ and $[\![Q]\!]$ would be distinguishable, as in the last case), and $(b)(Tr_m^d\langle b\rangle \mid [\![Q']\!]) \Longrightarrow T'' \approx_{ct} T'$. Thus we have $[\![Q']\!] \Longrightarrow T'''$ and $(b)(Tr_m^d\langle b\rangle \mid [\![Q']\!]) \Longrightarrow (b)(Tr_m^d\langle b\rangle \mid T''') \approx_{ct} T'' \approx_{ct} T'$. By Lemma 4.3, we know that $Q' \Longrightarrow Q''$ and $[\![Q'']\!] \approx_{ct} T'''$. So we have $Q \stackrel{\overline{a}(b)}{\Longrightarrow} Q''$ and $(b)(Tr_m^d\langle b\rangle \mid [\![Q'']\!]) \approx_{ct} T'' \approx_{ct} T'$. The above boils down to the following equation.

$$(b)(Tr_m^d\langle b\rangle \mid [\![P']\!]) \;\approx_{ct}\; T \;\approx_{ct}\; T' \;\approx_{ct}\; T'' \;\approx_{ct}\; (b)(Tr_m^d\langle b\rangle \mid [\![Q'']\!])$$

Since $\approx_{ct}$ is a congruence, we can derive that

$$\begin{aligned} (m)(\; (b)(Tr_m^d\langle b\rangle \mid [\![P']\!]) \;\mid !m(Z).Z\langle A\rangle) \\ \approx_{ct}\; (m)(\; (b)(Tr_m^d\langle b\rangle \mid [\![Q'']\!]) \;\mid !m(Z).Z\langle A\rangle) \end{aligned}$$

Using Theorem 5.1, we have

$$(b)([\![P']\!] \mid A\langle b\rangle) \approx_{ct} (b)([\![Q'']\!] \mid A\langle b\rangle) \tag{C.3}$$

We now recall that since $P \xrightarrow{\overline{a}(b)} P'$ and $Q \overset{\overline{a}(b)}{\Longrightarrow} Q''$, our goal is to prove that for every R it holds

$$(b)(P' \mid R)\ \mathcal{R}\ (b)(Q'' \mid R)$$

That is,

$$(b)([\![P']\!] \mid [\![R]\!]) \approx_{ct} (b)([\![Q'']\!] \mid [\![R]\!]) \tag{C.4}$$

Comparing (C.3) and (C.4), we can observe that since the image of the encoding is contained by the target model $\Pi^{D,d}$ and A is arbitrary (it has the knowledge of b as well), if we traverse all possibility of A then $[\![R]\!]$ must be hit somewhere. We thus conclude that (C.4) is true and the simulation is closed.

- τ action: $P \xrightarrow{\tau} P'$. By Lemma 4.2, $[\![P]\!] \xrightarrow{\tau} T \equiv [\![P']\!]$. Since $[\![P]\!] \approx_{ct} [\![Q]\!]$, we know that $[\![Q]\!] \Longrightarrow T' \approx_{ct} T$. By Lemma 4.3, $Q \Longrightarrow Q'$ and $T' \approx_{ct} [\![Q']\!]$. We need to prove $P' \mathcal{R} Q'$, *i.e.*, $[\![P']\!] \approx_{ct} [\![Q']\!]$. This is straightforward because

$$[\![P']\!] \approx_{ct} T \approx_{ct} T' \approx_{ct} [\![Q']\!]$$

The proof is now completed. □

Appendix D. Proofs for Section 5

D.1 Proof of the factorization theorem

We prove the factorization theorem (Thm. 5.1) in this section. In order to do this, we need some laws as to the behaviour of the trigger Tr^d_m. The counterparts concerning the other two kinds of triggers, *i.e.*, Tr_m and Tr^D_m, are referred to [24, 36]. We note that these triggers are of different types, so there would be no conflict. Intuitively, Lemma D.1 states some distributive laws about the replication $!m(Z).Z\langle A\rangle$ and the various process operators, *i.e.*, prefixes, parallel composition, and application. We note that the proofs may use some up-to technique.

Lemma D.1. *Suppose $E[X], E_1[X], E_2[X]$ belong to $\Pi^{D,d}$ and X stands for a name abstraction, and assume $m \notin \mathrm{fn}(E, E_1, E_2, A, B)$.*

(1) If $m \notin \mathrm{fn}(\alpha)$ and $\mathrm{bpv}(\alpha) \notin \mathrm{fpv}(A)$, then
(i) we have

$$(m)(\alpha.E[Tr^d_m] \mid !m(Z).Z\langle A\rangle) \approx_{ct} \alpha.(m)(E[Tr^d_m] \mid !m(Z).Z\langle A\rangle)$$

(ii) moreover, if $E[Tr^d_m] \equiv \widetilde{\langle U\rangle} E'$ for some non-abstraction E', then

$$(m)(\alpha.E[Tr^d_m] \mid !m(Z).Z\langle A\rangle) \approx_{ct} \alpha.\widetilde{\langle U\rangle}((m)(E' \mid !m(Z).Z\langle A\rangle))$$

(2) For output prefix
(i) we have

$$(m)(\overline{a}B_1.E_1[Tr^d_m] \mid !m(Z).Z\langle A\rangle) \approx_{ct} (m)(\overline{a}B_2.E_1[Tr^d_m] \mid !m(Z).Z\langle A\rangle)$$

where $B_1 \equiv E_2[Tr_m^d]$, $B_2 \equiv (m)(E_2[Tr_m^d] \mid !m(Z).Z\langle A\rangle)$.

(ii) moreover, if $E_2[Tr_m^d] \equiv \widetilde{\langle U\rangle}E_2'$ *and* $E_1[Tr_m^d] \equiv \widetilde{\langle U'\rangle}E_1'$ *for some non-abstraction* E_1' *and* E_2'*, then*

$$(m)(\overline{a}B_1.E_1[Tr_m^d] \mid !m(Z).Z\langle A\rangle) \approx_{ct} \overline{a}B_2'.B_2''$$

where $B_1 \equiv E_2[Tr_m^d]$, $B_2' \equiv \widetilde{\langle U\rangle}(m)(E_2' \mid !m(Z).Z\langle A\rangle)$, *and* $B_2'' \equiv \widetilde{\langle U'\rangle}(m)(E_1' \mid !m(Z).Z\langle A\rangle)$.

(3) It holds for parallel composition that

$$\begin{aligned}(m)(E_1[Tr_m^d] \mid E_2[Tr_m^d] \mid !m(Z).Z\langle A\rangle)\\ \approx_{ct} (m)(E_1[Tr_m^d] \mid !m(Z).Z\langle A\rangle) \mid (m)(E_2[Tr_m^d] \mid !m(Z).Z\langle A\rangle)\end{aligned}$$

(4) For application operation

(i) we have

$$B\langle(m)(E_1[Tr_m^d] \mid !m(Z).Z\langle A\rangle)\rangle \approx_{ct} (m)(B\langle E_1[Tr_m^d]\rangle \mid !m(Z).Z\langle A\rangle)$$

(ii) moreover, if $E_1[Tr_m^d] \equiv \widetilde{\langle U\rangle}E_1'$ *for some non-abstraction* E_1'*, then*

$$B\langle\widetilde{\langle U\rangle}((m)(E_1' \mid !m(Z).Z\langle A\rangle))\rangle \approx_{ct} (m)(B\langle E_1[Tr_m^d]\rangle \mid !m(Z).Z\langle A\rangle))$$

Proof. (1) We focus on (i) since (ii) is similar. We define $P_1 \stackrel{\text{def}}{=} (m)(\alpha.E[Tr_m^d] \mid !m(Z).Z\langle A\rangle)$ and $Q_1 \stackrel{\text{def}}{=} \alpha.(m)(E[Tr_m^d] \mid !m(Z).Z\langle A\rangle)$. We define $\mathcal{R}_1 \stackrel{\text{def}}{=} \{(P_1, Q_1)\} \cup \mathrm{ID}$ in which $\mathrm{ID} \stackrel{\text{def}}{=} \{(P, P)\}$ is the identity relation. We show that $\mathcal{R}_1$ is a strong context bisimulation up-to $\equiv$, which forces the result to be true. We iterate the cases on α, which is obviously the only action P_1 incurs.

- α is τ. Then $P_1 \xrightarrow{\tau} (m)(E[Tr_m^d] \mid !m(Z).Z\langle A\rangle)$ is matched by $Q_1 \xrightarrow{\tau} (m)(E[Tr_m^d] \mid !m(Z).Z\langle A\rangle)$.
- α is $c(Y)$ for some c. Then $P_1 \xrightarrow{c(R)} (m)(E[Tr_m^d]\{R/Y\} \mid !m(Z).Z\langle A\rangle)$ is matched by

$$\begin{aligned}Q_1 \xrightarrow{c(R)} &((m)(E[Tr_m^d] \mid !m(Z).Z\langle A\rangle))\{R/Y\}\\ &\equiv (m)(E[Tr_m^d]\{R/Y\} \mid !m(Z).Z\langle A\rangle)\end{aligned}$$

  due to $bv(\alpha) \notin fv(A)$.
- α is $\overline{b}B$ for some b and B. Then $P_1 \xrightarrow{\overline{b}B} (m)(E[Tr_m^d] \mid !m(Z).Z\langle A\rangle) \stackrel{\text{def}}{=} R_1$ is matched by $Q_1 \xrightarrow{\overline{b}B} R_1$, and clearly we have $(E[B] \mid R_1)\,\mathcal{R}\,(E[B] \mid R_1)$ for every $E[X]$.

(2) We only prove (i) since it is technically more involved and (ii) can be proven in a similar manner. For convenience, let $P_2 \stackrel{\text{def}}{=} (m)(\overline{a}B_1.E_1[Tr_m^d] \mid !m(Z).Z\langle A\rangle)$ and $Q_2 \stackrel{\text{def}}{=} (m)(\overline{a}B_2.E_1[Tr_m^d] \mid !m(Z).Z\langle A\rangle)$. We define $\mathcal{R}_2$ as follows.

$$\left\{\Big((m)(G[B_1] \mid !m(Z).Z\langle A\rangle), (m)(G[B_2] \mid !m(Z).Z\langle A\rangle)\Big)\Big|\text{for } G[\cdot], E_2, A\right\} \cup \sim_{ct} \tag{D.1}$$

We note that E_2 appears in B_1, B_2 and m is fresh, *i.e.*, $m \notin \mathrm{fn}(G, E_1, E_2, A)$. One can easily observe that $(P_2, Q_2) \in \mathcal{R}_2$ by taking G as $\overline{a}[\cdot].E_1[Tr_m^d]$. We show that $\mathcal{R}_2$ is a strong context bisimulation up-to $\sim_{ct}$ by induction on G. For the sake of convenience, we denote the element in $\mathcal{R}_2$ as: $P_3 \stackrel{\text{def}}{=} (m)(G[B_1] \mid !m(Z).Z\langle A\rangle)$ and $Q_3 \stackrel{\text{def}}{=} (m)(G[B_2] \mid !m(Z).Z\langle A\rangle)$.

- $G[\cdot]$ is $[\cdot]$. Then we have

$$\begin{array}{rcl} P_3 & \equiv & (m)(B_1 \,|\, !m(Z).Z\langle A\rangle) \\ Q_3 & \equiv & (m)(B_2 \,|\, !m(Z).Z\langle A\rangle) \end{array}$$

  Because $m \notin fn(E_2, A)$, we know $Q_3 \sim_{ct} B_2 \equiv P_3$. This closes the current case.
- $G[\cdot]$ is $b(Y).G_1[\cdot]$. We assume that $Y \notin \mathrm{fpv}(A)$. Now we have

$$\begin{array}{rcl} P_3 & \equiv & (m)(b(Y).G_1[B_1] \,|\, !m(Z).Z\langle A\rangle) \\ Q_3 & \equiv & (m)(b(Y).G_1[B_2] \,|\, !m(Z).Z\langle A\rangle) \end{array}$$

  Assume that P_3 releases an action (the case Q_3 does is similar). Clearly the only action from P_3 is $b(A')$ by $G[B_1]$, *i.e.*,

$$\begin{array}{rcl} P_3 & \xrightarrow{b(A')} & (m)((G_1[B_1])\{A'/Y\} \,|\, !m(Z).Z\langle A\rangle) \\ & & \equiv (m)((G'_1[B'_1]) \,|\, !m(Z).Z\langle A\rangle) \end{array}$$

  where

$$\begin{array}{l} G'_1 \stackrel{\text{def}}{=} G_1\{A'/Y\}, \text{ and} \\ B'_1 \stackrel{\text{def}}{=} B_1\{A'/Y\} \equiv (E_2[Tr^d_m])\{A'/Y\} \equiv E'_2[Tr^d_m] \text{ for some } E'_2[X] \end{array}$$

  So Q_3 simulates with the following action

$$\begin{array}{rcl} Q_3 & \xrightarrow{b(A')} & (m)((G_1[B_2])\{A'/Y\} \,|\, !m(Z).Z\langle A\rangle) \\ & & \equiv (m)((G'_1[B'_2]) \,|\, !m(Z).Z\langle A\rangle) \end{array}$$

  where $B'_2 \stackrel{\text{def}}{=} B_2\{A'/Y\} \equiv (m)(E'_2[Tr^d_m] \,|\, !m(Z).Z\langle A\rangle)$. This case is closed by taking G and E_2 in (D.1) respectively as G'_1 and E'_2.
- $G[\cdot]$ is $\overline{b}\big[G_1[\cdot]\big].T$. Then we have

$$\begin{array}{rcl} P_3 & \equiv & (m)(\overline{b}\big[G_1[B_1]\big].T \,|\, !m(Z).Z\langle A\rangle) \\ Q_3 & \equiv & (m)(\overline{b}\big[G_1[B_2]\big].T \,|\, !m(Z).Z\langle A\rangle) \end{array}$$

  Assume that P_3 releases an action (the case Q_3 does is similar). Clearly the only action from P_3 is $(m)\overline{b}\big[G_1[B_1]\big]$, *i.e.*,

$$P_3 \xrightarrow{(m)\overline{b}\left[G_1[B_1]\right]} T \,|\, !m(Z).Z\langle A\rangle$$

  Then Q_3 simulates by the following action

$$Q_3 \xrightarrow{\overline{b}\left[G_1[B_2]\right]} (m)(T \,|\, !m(Z).Z\langle A\rangle)$$

  Now for every $F[X]$ s.t. $\{m\} \cap fn(F) = \emptyset$, we need to establish

$$\begin{array}{rcl} (m)(F[G_1[B_1]] \,|\, T \,|\, !m(Z).Z\langle A\rangle) & \mathcal{R}_2 & F[G_1[B_2]] \,|\, (m)(T \,|\, !m(Z).Z\langle A\rangle) \\ & & \sim_{ct} (m)(F[G_1[B_2]] \,|\, T \,|\, !m(Z).Z\langle A\rangle) \end{array}$$

where the $\sim_{ct}$ may involve some α-conversion to avoid name capture. Then this case is closed by taking G as $F[G_1[\cdot]] \,|\, T$ in (D.1).

- $G[\cdot]$ is $\overline{b}A.G_1[\cdot]$. This is similar to the previous case.
- $G[\cdot]$ is $G_1[\cdot] \,|\, T$. In this case we have P_3 and Q_3 as below.

$$\begin{array}{rcl} P_3 & \equiv & (m)(G_1[B_1] \,|\, T \,|\, !m(Z).Z\langle A\rangle) \\ Q_3 & \equiv & (m)(G_1[B_2] \,|\, T \,|\, !m(Z).Z\langle A\rangle) \end{array}$$

Assume that P_3 releases an action α (the situation Q_3 does is similar). There are a number of cases concerning where α originates from, specifically: 1) T; 2) G_1; 3) B_1; 4) interaction between G_1, B_1, T and $!m(Z).Z\langle A\rangle$ (5 subcases). Below we look into two cases. The remainder is similar.

- First, the case α comes from an interaction between B_1 and T (not on m since T has no knowledge of m). We look at the subcase B_1 does an output and T does an input. That is

$$\begin{array}{l} B_1 \xrightarrow{(\widetilde{c})\overline{d}A'} B_1' \qquad T \xrightarrow{d(A')} T' \\ P_3 \xrightarrow{\tau} \sim_{ct} (m)((\widetilde{c})(G_1[B_1'] \,|\, T') \,|\, !m(Z).Z\langle A\rangle) \stackrel{\text{def}}{=} P_3' \end{array}$$

Since d is not m, the action by B_1 must stem from E_2, *i.e.*, $E_2[Tr_m] \xrightarrow{(\widetilde{c})\overline{d}A'} E_2'[Tr_m^d] \equiv B_1'$. So

$$B_2 \xrightarrow{(\widetilde{c})\overline{d}A'} B_2' \equiv (m)(E_2'[Tr_m^d] \,|\, !m(Z).Z\langle A\rangle)$$

and

$$Q_3 \xrightarrow{\tau} \sim_{ct} (m)((\widetilde{c})(G_1[B_2'] \,|\, T') \,|\, !m(Z).Z\langle A\rangle) \stackrel{\text{def}}{=} Q_3'$$

Then we have $(P_3', Q_3') \in \mathcal{R}_2$ by regarding G and E_2 in (D.1) respectively as $(\widetilde{c})(G_1[\cdot] \,|\, T')$ and E_2'.

- Second, the case α comes from an interaction between $!m(Z).Z\langle A\rangle$ and B_1. That is

$$\begin{array}{l} B_1 \equiv E_2[Tr_m^d] \xrightarrow{(\widetilde{c})\overline{m}A'} B_1' \qquad A' \equiv \langle Y\rangle(Y\langle h\rangle) \;\; \text{(actually } \widetilde{c} \text{ only has } h) \\ !m(Z).Z\langle A\rangle \xrightarrow{m(A')} \sim_{ct} A'\langle A\rangle \,|\, !m(Z).Z\langle A\rangle \\ P_3 \xrightarrow{\tau} \sim_{ct} (m)((\widetilde{c})(G_1[B_1'] \,|\, A'\langle A\rangle) \,|\, T \,|\, !m(Z).Z\langle A\rangle) \stackrel{\text{def}}{=} P_3' \end{array}$$

Since B_1 is free to go (*i.e.*, not underneath any prefix), we have

$$\begin{array}{l} P_3' \sim_{ct} (m)(G_1[B_1''] \,|\, T \,|\, !m(Z).Z\langle A\rangle) \stackrel{\text{def}}{=} P_3'' \\ \text{where } B_1'' \stackrel{\text{def}}{=} (\widetilde{c})(B_1' \,|\, A'\langle A\rangle) \end{array}$$

Moreover, because Tr_m^d has been fired in B_1, we can write B_1' as $E_2'[Tr_m^d]$ for some $E_2'[X]$ (in which $X \notin \text{fpv}(E_2')$). Then

$$B_1'' \equiv E_2''[Tr_m^d] \qquad \text{where} \qquad E_2''[X] \stackrel{\text{def}}{=} (\widetilde{c})(E_2'[X] \,|\, A'\langle A\rangle)$$

Thus

$$\begin{array}{rl} B_2 \equiv (m)(E_2[Tr_m^d] \mid !m(Z).Z\langle A\rangle) & \\ \xrightarrow{\tau} \sim_{ct} & (m)((\widetilde{c})(B_1' \mid A'\langle A\rangle) \mid !m(Z).Z\langle A\rangle) \\ \sim_{ct} & (m)(B_1'' \mid !m(Z).Z\langle A\rangle) \stackrel{\text{def}}{=} B_2'' \end{array}$$

$$Q_3 \xrightarrow{\tau} (m)(G_1[B_2''] \mid T \mid !m(Z).Z\langle A\rangle) \stackrel{\text{def}}{=} Q_3''$$

In summary,

$$P_3 \xrightarrow{\tau} \sim_{ct} (m)(G_1[B_1''] \mid T \mid !m(Z).Z\langle A\rangle) \equiv P_3'' \quad \text{where } B_1'' \equiv E_2''[Tr_m^d]$$

$$Q_3 \xrightarrow{\tau} \sim_{ct} (m)(G_1[B_2''] \mid T \mid !m(Z).Z\langle A\rangle) \equiv Q_3'' \quad \text{where } B_2'' \equiv (m)(E_2''[Tr_m^d] \mid !m(Z).Z\langle A\rangle)$$

Hence we have $(P_3'', Q_3'') \in \mathcal{R}_2$ by treating G and E_2 in (D.1) respectively as $G_1[\cdot] \mid T$ and E_2''.

- $G[\cdot]$ is $(c)G_1[\cdot]$. Here we have P_3 and Q_3 as below.

$$\begin{array}{rcl} P_3 & \equiv & (m)((c)G_1[B_1] \mid !m(Z).Z\langle A\rangle) \\ Q_3 & \equiv & (m)((c)G_1[B_2] \mid !m(Z).Z\langle A\rangle) \end{array}$$

  Suppose P_3 makes an action α (the situation for Q_3 is similar). Like the previous case, there are several subcases pertaining to where α is originated: 1) G_1; 2) B_1; 3) interaction between G_1, B_1 and $!m(Z).Z\langle A\rangle$ (3 subcases). The discussion can be conducted in a way similar to the previous case (only caution that the design of G in (D.1) may involve the restriction on c).
- $G[\cdot]$ is $\langle y\rangle G'[\cdot]$ or $\langle Y\rangle G'[\cdot]$. This case, somehow correlated with the next case, is trivial because P_3 and Q_3 do not exhibit any action.
- $G[\cdot]$ is $G_1[\cdot]\langle z\rangle$ or $G_1[\cdot]\langle B'\rangle$. Take the former (the latter is similar), which means G_1 takes the shape $\langle y\rangle G_2[\cdot]$, *i.e.*, $G[\cdot] \equiv (\langle y\rangle G_2[\cdot])\langle z\rangle$. So $G[\cdot]$ is essentially $G_2'[\cdot]$ for some G_2'. Then this case eventually falls into one of the other cases.

(3) Let

$$\begin{array}{l} P_3 \stackrel{\text{def}}{=} (m)(E_1[Tr_m^d] \mid E_2[Tr_m^d] \mid !m(Z).Z\langle A\rangle) \\ Q_3 \stackrel{\text{def}}{=} (m)(E_1[Tr_m^d] \mid !m(Z).Z\langle A\rangle) \mid (m)(E_2[Tr_m^d] \mid !m(Z).Z\langle A\rangle) \end{array}$$

This result follows from (2) in this lemma, since $(P_3, Q_3) \in \mathcal{R}_2$ in (D.1) by taking G as $E_1[Tr_m^d] \mid [\cdot]$ (actually we prove a more general result in there).

(4) We focus on (i) because (ii) is similar. Suppose $B \equiv \langle Y\rangle T$, it amounts to proving

$$\begin{array}{rl} & P_1 \stackrel{\text{def}}{=} T\{T_1/Y\} \approx_{ct} (m)(T\{T_2/Y\} \mid !m(Z).Z\langle A\rangle) \stackrel{\text{def}}{=} Q_1 \\ \text{in which} & T_1 \stackrel{\text{def}}{=} (m)(E_1[Tr_m^d] \mid !m(Z).Z\langle A\rangle) \text{ and } T_2 \stackrel{\text{def}}{=} E_1[Tr_m^d] \end{array}$$

To achieve this, we define $\mathcal{R}_3$ as follows.

$$\left\{ \Big(G[T_1], (m)(G[T_2] \mid !m(Z).Z\langle A\rangle) \Big) \,\middle|\, \text{for } G[\cdot], E_1,\ m \notin fn(G, E_1, A) \right\} \cup \sim_{ct} \qquad \text{(D.2)}$$

We note that E_1 appears in T_1, T_2, and $(P_1, Q_1) \in \mathcal{R}_3$. We can show $\mathcal{R}_3$ is a strong context bisimulation up-to $\sim_{ct}$ by induction on G. We skip the details because they are basically the same to the previous part (2-3) of this lemma. □

Now we are ready to prove the factorization theorem.

Proof of Theorem 5.1.

We focus on the case when A is a name abstraction, *i.e.*, clause (3) of this theorem. We proceed by induction on the structure of E, and focus on the main cases, namely those relevant most to name-abstraction. The proof for the others is similar (caution for using different triggers) and can be referred to [24, 36]. The cases 1,2,3 are the base cases.

1. E is 0 or E is Y and $Y \neq X$. These cases are trivial.
2. E is X. In this case, $E[Tr^d_m]$ is $Tr^d_m \equiv \langle z\rangle\overline{m}[\langle Y\rangle(Y\langle z\rangle)]$. The two terms to compare are

$$A \quad \text{and} \quad \langle z\rangle((m)(\overline{m}[\langle Y\rangle(Y\langle z\rangle)] \,|\, !m(Z).Z\langle A\rangle))$$

   Suppose $A \equiv \langle z'\rangle F$, then the following equality is straightforward for every h, which completes this case.

$$A\langle h\rangle \approx_{ct} (m)(\overline{m}[\langle Y\rangle(Y\langle h\rangle)] \,|\, !m(Z).Z\langle A\rangle)$$

3. E is $X\langle h\rangle$. We show that the following relation $\mathcal{R}$ is a context bisimulation up-to $\sim_{ct}$.

$$\mathcal{R} \stackrel{\text{def}}{=} \{(A\langle h\rangle, (m)(Tr^d_m\langle h\rangle \,|\, !m(Z).Z\langle A\rangle)) \,\big|\, m \text{ is fresh w.r.t. } A \text{ and } h\} \cup \approx_{ct}$$

   Assume $(A\langle h\rangle, (m)(Tr^d_m\langle h\rangle \,|\, !m(Z).Z\langle A\rangle)) \in \mathcal{R}$ and $A \equiv \langle z'\rangle T$. Then the pair of interest is

$$(T\{h/z'\}\ ,\ (m)(\overline{m}[\langle Y\rangle(Y\langle h\rangle)] \,|\, !m(Z).Z\langle A\rangle)$$

   There are mainly two cases to analyze.

   • $(m)(\overline{m}[\langle Y\rangle(Y\langle h\rangle)] \,|\, !m(Z).Z\langle A\rangle) \xrightarrow{\alpha} T'$. Then α must be τ, and thus

$$T' \sim_{ct} (m)(T\{h/z'\} \,|\, !m(Z).Z\langle A\rangle)$$

   By $T\{h/z'\} \Longrightarrow T\{h/z'\}$ (null transition), we have

$$T\{h/z'\} \sim_{ct} T\{h/z'\}\ \mathcal{R}\ T\{h/z'\} \sim_{ct} T'$$

   • $T\{h/z'\} \xrightarrow{\alpha} T_1$. This is simulated by

$$\begin{array}{ll} & (m)(\overline{m}[\langle Y\rangle(Y\langle h\rangle)] \,|\, !m(Z).Z\langle A\rangle) \\ \xrightarrow{\tau} & (m)(T\{h/z'\} \,|\, !m(Z).Z\langle A\rangle) \\ \xrightarrow{\alpha} & (m)(T_1 \,|\, !m(Z).Z\langle A\rangle) \stackrel{\text{def}}{=} T_2 \end{array}$$

   So it holds that

$$T_1 \sim_{ct} T_1\ \mathcal{R}\ T_1 \sim_{ct} T_2$$

The following cases 4,5,6,7,8,9 are those involving the induction hypothesis (ind. hyp. for short). We note that in the induction steps, we will focus on the case when (ii) of (3) in this theorem occurs, because these are the most interesting cases and the other situations can be tackled similarly.

4. E is $\langle y\rangle E_1$. By ind. hyp., we immediately have, as required,

$$\begin{array}{rcl} E[A] & \equiv & \langle y\rangle E_1[A] \\ & \approx_{ct} & \langle y\rangle((m)(E_1[Tr_m^d] \mid !m(Z).Z\langle A\rangle)) \end{array}$$

5. E is $\overline{a}E_2.E_1$. We have

$$\begin{array}{rcll} E[A] & \equiv & \overline{a}E_2[A].E_1[A] & \\ & \approx_{ct} & \overline{a}[(m)(E_2[Tr_m^d] \mid !m(Z).Z\langle A\rangle)].((m)(E_1[Tr_m^d] \mid !m(Z).Z\langle A\rangle)) & \\ & & & \text{(ind. hyp.)} \\ & \approx_{ct} & (m)(\overline{a}[(m)(E_2[Tr_m^d] \mid !m(Z).Z\langle A\rangle)].E_1[Tr_m^d] \mid !m(Z).Z\langle A\rangle) & \\ & & & \text{(Lem. D.1(1))} \\ & \approx_{ct} & (m)(\overline{a}E_2[Tr_m^d].E_1[Tr_m^d] \mid !m(Z).Z\langle A\rangle) & \\ & & & \text{(Lem. D.1(2))} \end{array}$$

6. E is $a(Y).E_1$. This is similar to the previous case.
7. E is $E_1 \mid E_2$. We have

$$\begin{array}{rcll} & & E[A] \equiv E_1[A] \mid E_2[A] & \\ \approx_{ct} & & (m)(E_1[Tr_m^d] \mid !m(Z).Z\langle A\rangle) \mid (m)(E_2[[Tr_m^d] \mid !m(Z).Z\langle A\rangle) & \text{(ind. hyp.)} \\ \approx_{ct} & & (m)(E_1[[Tr_m^d] \mid E_2[[Tr_m^d] \mid !m(Z).Z\langle A\rangle) & \text{(Lem. D.1(3))} \end{array}$$

8. E is $(c)E_1$. This is similar to the previous case.
9. E is $E_2\langle E_1\rangle$. This case can be reduced to the case E is $Y\langle E_1\rangle$, because otherwise it falls into one of the previous cases (up-to structural congruence). So we have

$$\begin{array}{rcll} E[A] & \equiv & Y\langle E_1[A]\rangle & \\ & \approx_{ct} & Y\langle(m)(E[[Tr_m^d] \mid !m(Z).Z\langle A\rangle)\rangle \stackrel{\text{def}}{=} T & \text{(ind. hyp.)} \end{array}$$

For every B, we have the follow-up equation by Lemma D.1(4).

$$B\langle(m)(E_1[Tr_m^d] \mid !m(Z).Z\langle A\rangle)\rangle \approx_{ct} (m)(B\langle E_1[Tr_m^d]\rangle \mid !m(Z).Z\langle A\rangle)$$

Thus we have $T \approx_{ct} (m)(Y\langle E_1[Tr_m^d]\rangle \mid !m(Z).Z\langle A\rangle)$.

The proof is now completed. □

D.2 Proof of the congruence of the normal bisimilarity

In this section, we give the proof of Theorem 5.4, that is, the congruence of the normal bisimilarity defined in Definition 5.3. We take advantage of the up-to restriction technique [21, 24, 31]. Basically, a relation $\mathcal{R}$ that is a (context/normal) bisimulation up-to restriction is obtained as follows: for the pair of processes, say P' and Q', in the conclusion of each clause of the (context/normal) bisimulation, it holds that $P' \equiv (\widetilde{c})P''$, $Q' \equiv (\widetilde{c})Q''$, and $P''\,\mathcal{R}\,Q''$. More details about this technique are referred to [21, 24, 31].

Proof of Theorem 5.4. We define the relation $\mathcal{R}$ as below.

$$\mathcal{R} \stackrel{\text{def}}{=} \{(C[P], C[Q]) \mid P \approx_{nr} Q, \text{ and } C \text{ is } R \mid [\cdot], \text{ or } (d)[\cdot]\} \cup \approx_{nr}$$

We show that $\mathcal{R}$ is a normal bisimulation up-to restriction and $\approx_{nr}$. We note that we concentrate on clauses concerning communicating triggers corresponding to name parameterization, and the other clauses are similar and can be referred to [24]. We make a case analysis. In the argument, $\cdot$ represents some unspecified process.

1. $C[P] \xrightarrow{a(A)} \cdot$ in which A is Tr^d_m (m is fresh).
   - C is $R \mid [\cdot]$. There are two possibilities.
     - The action is from P. That is, $P \xrightarrow{a(A)} P'$, and $C[P] \xrightarrow{a(A)} R \mid P'$. Since $P \approx_{nr} Q$, we know that $Q \stackrel{a(A)}{\Longrightarrow} Q'$ and $P' \approx_{nr} Q'$. So $C[Q]$ simulates by $C[Q] \stackrel{a(A)}{\Longrightarrow} R \mid Q'$. From $P' \approx_{nr} Q'$, we have
     $$R \mid P' \;\; \mathcal{R} \;\; R \mid Q'$$
     - The action is from R. That is, $R \xrightarrow{a(A)} R'$, and $C[P] \xrightarrow{a(A)} R' \mid P$. So $C[Q]$ simulates by $C[Q] \xrightarrow{a(A)} R' \mid Q$. Because $P \approx_{nr} Q$, we have
     $$R' \mid P \;\; \mathcal{R} \;\; R' \mid Q$$
   - C is $(d)[\cdot]$. The action is from P. That is, $P \xrightarrow{a(A)} P'$, and $C[P] \xrightarrow{a(A)} (d)P'$. Since $P \approx_{nr} Q$, we know that $Q \stackrel{a(A)}{\Longrightarrow} Q'$ for some Q' and $P' \approx_{nr} Q'$. So $C[Q]$ simulates by $C[Q] \stackrel{a(A)}{\Longrightarrow} (d)Q'$. Because $P' \approx_{nr} Q'$, we have
   $$(d)P' \;\; \mathcal{R} \;\; (d)Q'$$

2. $C[P] \xrightarrow{(\widetilde{c})\overline{a}A} \cdot$. (As mentioned, we assume by default such communicated term is a name abstraction.)
   - C is $R \mid [\cdot]$. There are several possibilities.
     - The action is from R. That is, $R \xrightarrow{(\widetilde{c})\overline{a}A} R'$ and $C[P] \equiv R \mid P \xrightarrow{(\widetilde{c})\overline{a}A} R' \mid P$. In this case, $C[Q]$ simulates by $C[Q] \xrightarrow{(\widetilde{c})\overline{a}A} R' \mid Q$. Now we have for $C' \stackrel{\text{def}}{=} R' \mid !m(Z).Z\langle A\rangle \mid [\cdot]$.
     $$\begin{array}{rcl}(\widetilde{c})(R' \mid P \mid !m(Z).Z\langle A\rangle) & \equiv & (\widetilde{c})C'[P] \\ (\widetilde{c})(R' \mid Q \mid !m(Z).Z\langle A\rangle) & \equiv & (\widetilde{c})C'[Q]\end{array}$$
     in which $C'[P] \,\mathcal{R}\, C'[Q]$ with $P \approx_{nr} Q$.
     - The action is from P. That is, $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$ and $C[P] \equiv R \mid P \xrightarrow{(\widetilde{c})\overline{a}A} R \mid P'$. In this case, since $P \approx_{nr} Q$, we know that $Q \stackrel{(\widetilde{d})\overline{a}B}{\Longrightarrow} Q'$, and it holds that $(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle) \approx_{nr} (\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)$. So $C[Q]$ simulates by $C[Q] \stackrel{(\widetilde{d})\overline{a}B}{\Longrightarrow} R \mid Q'$. Now, we have
     $$\begin{array}{rcl}(\widetilde{c})(R \mid P' \mid !m(Z).Z\langle A\rangle) & \equiv & R \mid (\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle) \\ & \equiv & C[(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle)] \\ \\ (\widetilde{d})(R \mid Q' \mid !m(Z).Z\langle B\rangle) & \equiv & R \mid (\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle) \\ & \equiv & C[(\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)]\end{array}$$
     Hence we obtain $C[(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle)] \,\mathcal{R}\, C[(\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)]$ because $(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle) \approx_{nr} (\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)$.
   - C is $(e)[\cdot]$ (we use e instead of d here for clarity). There are a number of possibilities.

∘ $e \notin \mathrm{fn}(A)$. The action is $C[P] \xrightarrow{(\widetilde{c})\overline{a}A} (e)P'$ from $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$ (we assume $e \notin \widetilde{c}$). Then since $P \approx_{nr} Q$, we know that $Q \xRightarrow{(\widetilde{d})\overline{a}B} Q'$ (we assume $e \notin \widetilde{d}$, and it holds that $(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle) \approx_{nr} (\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)$. There are two possibilities of the simulation by $C[Q]$.

* $e \notin \mathrm{fn}(B)$. Then $C[Q]$ simulates by $C[Q] \xRightarrow{(\widetilde{d})\overline{a}B} (e)Q'$. Now we have

$$\begin{aligned}(\widetilde{c})((e)P' \mid !m(Z).Z\langle A\rangle) &\equiv (e)(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle)\\ &\equiv C[(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle)]\end{aligned}$$

$$\begin{aligned}(\widetilde{d})((e)Q' \mid !m(Z).Z\langle B\rangle) &\equiv (e)(\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)\\ &\equiv C[(\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)]\end{aligned}$$

Hence we obtain $C[(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle)] \,\mathcal{R}\, C[(\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)]$ because $(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle) \approx_{nr} (\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)$.

* $e \in \mathrm{fn}(B)$. Then $C[Q]$ simulates by $C[Q] \xRightarrow{(\widetilde{d}e)\overline{a}B} Q'$. Now we have

$$\begin{aligned}(\widetilde{c})((e)P' \mid !m(Z).Z\langle A\rangle) &\equiv (e)(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle)\\ &\equiv C[(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle)]\end{aligned}$$

$$\begin{aligned}(\widetilde{d}e)(Q' \mid !m(Z).Z\langle B\rangle) &\equiv (e)(\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)\\ &\equiv C[(\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)]\end{aligned}$$

Hence we obtain $C[(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle)] \,\mathcal{R}\, C[(\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)]$ because $(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle) \approx_{nr} (\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)$.

∘ $e \in \mathrm{fn}(A)$. The action is $C[P] \xrightarrow{(\widetilde{c}e)\overline{a}A} P'$ from $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$ (we assume $e \notin \widetilde{c}$). Then since $P \approx_{nr} Q$, we know that $Q \xRightarrow{(\widetilde{d})\overline{a}B} Q'$ (we assume $e \notin \widetilde{d}$, and it holds that $(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle) \approx_{nr} (\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)$. There are two possibilities of the simulation by $C[Q]$.

* $e \notin \mathrm{fn}(B)$. Then $C[Q]$ simulates by $C[Q] \xRightarrow{(\widetilde{d})\overline{a}B} (e)Q'$. Now we have

$$\begin{aligned}(\widetilde{c}e)(P' \mid !m(Z).Z\langle A\rangle) &\equiv (e)(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle)\\ &\equiv C[(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle)]\end{aligned}$$

$$\begin{aligned}(\widetilde{d})((e)Q' \mid !m(Z).Z\langle B\rangle) &\equiv (e)(\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)\\ &\equiv C[(\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)]\end{aligned}$$

Hence we obtain $C[(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle)] \,\mathcal{R}\, C[(\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)]$ because $(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle) \approx_{nr} (\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)$.

* $e \in \mathrm{fn}(B)$. Then $C[Q]$ simulates by $C[Q] \xRightarrow{(\widetilde{d}e)\overline{a}B} Q'$. Now for every E as stipulated,

$$\begin{aligned}(\widetilde{c}e)(P' \mid !m(Z).Z\langle A\rangle) &\equiv (e)(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle)\\ &\equiv C[(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle)]\end{aligned}$$

$$\begin{aligned}(\widetilde{d}e)(Q' \mid !m(Z).Z\langle B\rangle) &\equiv (e)(\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)\\ &\equiv C[(\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)]\end{aligned}$$

Hence we obtain $C[(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle)]\,\mathcal{R}\,C[(\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)]$ because $(\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle) \approx_{nr} (\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)$.

3. $C[P] \xrightarrow{\tau} \cdot$.
   - C is $R \mid [\cdot]$. Three possibilities.
     - τ is from R. That is, $R \xrightarrow{\tau} R'$, and $C[P] \xrightarrow{\tau} R' \mid P$. Then $C[Q]$ simulates by $C[Q] \xrightarrow{\tau} R' \mid Q$. So we have $R' \mid P\,\mathcal{R}\,R' \mid Q$ because $P \approx_{nr} Q$.
     - τ is from P. That is, $P \xrightarrow{\tau} P'$, and $C[P] \xrightarrow{\tau} R \mid P'$. Since $P \approx_{nr} Q$, we know that $Q \Longrightarrow Q'$ and $P' \approx_{nr} Q'$. So $C[Q]$ simulates by $C[Q] \Longrightarrow R \mid Q'$. Because $P' \approx_{nr} Q'$, we have the following pair in $\mathcal{R}$.

$$R \mid P' \;\; \mathcal{R} \;\; R \mid Q'$$

     - τ is from interaction between R and P. Two more subcases.
       We note that although the factorization properties are proven for context bisimilarity $\approx_{ct}$, they are true for normal bisimilarity $\approx_{nr}$ as well, because $\approx_{ct}$ implies $\approx_{nr}$.
       * P makes an output and R makes an input. That is, $P \xrightarrow{(\widetilde{c})\overline{a}A} P'$, $R \xrightarrow{a(A)} R' \equiv E'[A]$ for some $E'[Y]$ (such that Y is only replaced by the inputted A), and $C[P] \xrightarrow{\tau} (\widetilde{c})(R' \mid P')$. Since $P \approx_{nr} Q$, we know that $Q \xRightarrow{(\widetilde{d})\overline{a}B} Q'$, and it holds for m fresh that

$$P'' \stackrel{\text{def}}{=} (\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle) \;\approx_{nr}\; (\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle) \stackrel{\text{def}}{=} Q''$$

       So $C[Q]$ simulates by $C[Q] \xRightarrow{\tau} (\widetilde{d})(R'' \mid Q')$ in which $R \xrightarrow{a(B)} R'' \equiv E'[B]$. We now apply the Factorization theorem (Thm. 5.1) and obtain the following.

$$\begin{array}{rcl}
(\widetilde{c})(R' \mid P') & \equiv & (\widetilde{c})(E'[A] \mid P') \\
& \approx_{nr} & (\widetilde{c})((m)(E'[Tr^d_m] \mid !m(Z).Z\langle A\rangle) \mid P') \\
& \equiv & (m)(E'[Tr^d_m] \mid (\widetilde{c})(P' \mid !m(Z).Z\langle A\rangle)) \\
& \equiv & (m)(E'[Tr^d_m] \mid P'')
\end{array}$$

$$\begin{array}{rcl}
(\widetilde{d})(R'' \mid Q') & \equiv & (\widetilde{d})(E'[B] \mid Q') \\
& \approx_{nr} & (\widetilde{d})((m)(E'[Tr^d_m] \mid !m(Z).Z\langle B\rangle) \mid Q') \\
& \equiv & (m)(E'[Tr^d_m] \mid (\widetilde{d})(Q' \mid !m(Z).Z\langle B\rangle)) \\
& \equiv & (m)(E'[Tr^d_m] \mid Q'')
\end{array}$$

       Summarizing, we have

$$\begin{array}{rcl}
(\widetilde{c})(R' \mid P') & \approx_{nr} & (m)(E'[Tr^d_m] \mid P'') \\
(\widetilde{d})(R'' \mid Q') & \approx_{nr} & (m)(E'[Tr^d_m] \mid Q'')
\end{array}$$

       in which $E'[Tr^d_m] \mid P''\,\mathcal{R}\,E'[Tr^d_m] \mid Q''$ because $P'' \approx_{nr} Q''$. Thus we are done with this subcase.
       * P makes an input and R makes an output. That is, $P \xrightarrow{a(A)} P' \equiv E_1[A]$ for some $E_1[Y]$ (such that Y is only replaced by the inputted A), $R \xrightarrow{(\widetilde{c})\overline{a}A} R'$, and $C[P] \xrightarrow{\tau} (\widetilde{c})(R' \mid P')$. We thus have $P \xrightarrow{a(Tr^d_m)} P'' \equiv E_1[Tr^d_m]$. Since $P \approx_{nr} Q$, we know that $Q \xRightarrow{a(Tr^d_m)} Q'' \equiv E_2[Tr^d_m]$ for some $E2[Z]$ (such that Z is only replaced by the inputted Tr^d_m), and $P'' \approx_{nr} Q''$. So $Q \xRightarrow{a(A)} Q' \equiv E_2[A]$. Therefore $C[Q]$ simulates by

$C[Q] \stackrel{\tau}{\Longrightarrow} (\widetilde{c})(R' \mid Q')$. Using the Factorization theorem (Thm. 5.1), we have

$$\begin{array}{rcl}(\widetilde{c})(R' \mid P') & \equiv & (\widetilde{c})(R' \mid E_1[A]) \\ & \approx_{nr} & (\widetilde{c})(R' \mid (m)(E_1[Tr_m^d] \mid !m(Z).Z\langle A\rangle)) \\ & \equiv & (m)((\widetilde{c})(R' \mid !m(Z).Z\langle A\rangle) \mid E_1[Tr_m^d]) \\ & \equiv & (m)(R_1 \mid P'')\end{array}$$

$$\begin{array}{rcl}(\widetilde{c})(R' \mid Q') & \equiv & (\widetilde{c})(R' \mid E_2[A]) \\ & \approx_{nr} & (\widetilde{c})(R' \mid (m)(E_2[Tr_m^d] \mid !m(Z).Z\langle A\rangle)) \\ & \equiv & (m)((\widetilde{c})(R' \mid !m(Z).Z\langle A\rangle) \mid E_2[Tr_m^d]) \\ & \equiv & (m)(R_1 \mid Q'')\end{array}$$

in which $R_1 \stackrel{\text{def}}{=} (\widetilde{c})(R' \mid !m(Z).Z\langle A\rangle)$. Summarizing, we have

$$\begin{array}{rcl}(\widetilde{c})(R' \mid P') & \approx_{nr} & (m)(R_1 \mid P'') \\ (\widetilde{c})(R' \mid Q') & \approx_{nr} & (m)(R_1 \mid Q'')\end{array}$$

in which $R_1 \mid P'' \,\mathcal{R}\, R_1 \mid Q''$ because $P'' \approx_{nr} Q''$. Thus we are done with this subcase.

- C is $(d)[\cdot]$. In this case, the action $C[P] \equiv (d)P \stackrel{\tau}{\rightarrow} (d)P'$ comes from $P \stackrel{\tau}{\rightarrow} P'$. Since $P \approx_{nr} Q$, we know $Q \Longrightarrow Q'$ and $P' \approx_{nr} Q'$. So $C[Q]$ simulates by $C[Q] \Longrightarrow (d)Q'$. Because $P' \approx_{nr} Q'$, we have the following pair in $\mathcal{R}$.

$$(d)P' \;\; \mathcal{R} \;\; (d)Q'$$

□

Acknowledgements. We are grateful to the referees of EXPRESS/SOS 2016 for their useful comments on this article and related works. We also thank Alan Schmitt, Qiang Yin, and the anonymous reviewers for the helpful suggestions and communications.

References

[1] M. Bundgaard, T. Hildebrandt and J.C. Godskesen, A cps encoding of name-passing in higher-order mobile embedded resources. *Theor. Comput. Sci.* **356** (2006) 422–439.
[2] A. Durier, D. Hirschkoff and D. Sangiorgi, Towards "up to context" reasoning about higher-order processes. *Theor. Comput. Sci.* DOI: 10.1016/j.tcs.2019.09.036 (2019).
[3] U.H. Engberg and M. Nielsen, A calculus of communicating systems with label passing. Tech. Rep. DAIMI PB-208, Computer Science Department, University of Aarhus (1986).
[4] U.H. Engberg and M. Nielsen, A calculus of communicating systems with label passing - ten years after. In Proof, Language, and Interaction: Essays in Honour of Robin Milner. MIT Press Cambridge (2000) 599–622.
[5] Y. Fu, On quasi open bisimulation. *Theor. Comput. Sci.* **338** (2005) 96–126.
[6] Y. Fu, Fair ambients. *Acta Inform.* **43** (2007) 535–594.
[7] Y. Fu, Theory of interaction. *Theor. Comput. Sci.* **611** (2015) 1–49.
[8] Y. Fu and H. Lu, On the expressiveness of interaction. *Theor. Comput. Sci.* **411** (2010) 1387–1451.
[9] D. Gorla, Towards a unified approach to encodability and separation results for process calculi. In *Proceedings of the 19th International Conference on Concurrency Theory (CONCUR 2008)*, vol. 5201 of *Lecture Notes in Computer Science*. Springer Verlag (2008) 492–507.
[10] D. Gorla, On the relative expressive power of calculi for mobility. *Electr. Notes Theor. Comput. Sci.* **249** (2009) 269–286.
[11] D. Gorla and U. Nestmann, Full abstraction for expressiveness: History, myths and facts. *Math. Struct. Comput. Sci.* **26** (2016) 639–654.
[12] D. Kouzapas, J.A. Pérez and N. Yoshida, On the relative expressiveness of higher-order session processes. In *Proceedings of the 25th European Symposium on Programming (ESOP 2016)*. In Vol. 9632 of *Lecture Notes in Computer Science*. Springer (2016) 446–475.

[13] I. Lanese, J. Pérez, D. Sangiorgi and A. Schmitt, On the expressiveness and decidability of higher-order process calculi. In *Proceedings of the 23rd Annual IEEE Symposium on Logic in Computer Science (LICS 2008)*. IEEE Computer Society (2008) 145–155.

[14] I. Lanese, J.A. Pérez, D. Sangiorgi and A. Schmitt, On the expressiveness of polyadic and synchronous communication in higher-order process calculi. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP 2010)*. In Vol. 6199 of *Lecture Notes in Computer Science*. Springer Verlag (2010) 442–453.

[15] I. Lanese, J.A. Pérez, D. Sangiorgi and A. Schmitt, On the expressiveness and decidability of higher-order process calculi. *Inf. Comput.* **209** (2011) 198–226.

[16] S. Lenglet, A. Schmitt and J.B. Stefani, Normal bisimulations in calculi with passivation. In: *Proceedings of the 12th International Conference on Foundations of Software Science and Computational Structures (FOSSACS 2009)*. Vol. 5504 of *Lecture Notes in Computer Science*. Springer Verlag (2009) 257–271.

[17] S. Lenglet, A. Schmitt and J.B. Stefani, Characterizing contextual equivalence in calculi with passivation. *Inf. Comput.* **209** (2011) 1390–1433.

[18] J.M. Madiot, D. Pous and D. Sangiorgi, Bisimulations up-to: Beyond first-order transition systems. In: *Proceedings of the 25th Conference on Concurrency Theory (CONCUR 2014)*. In Vol. 8704 of *Lecture Notes in Computer Science*. Springer Verlag (2014) 93–108.

[19] M. Merro and M. Hennessy, Bisimulation congruences in safe ambients. In: *Proceedings of the 29th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 02). Portland, Oregon* (2002) 71–80.

[20] R. Milner, Communication and Concurrency. Prentice Hall (1989).

[21] R. Milner, J. Parrow and D. Walker, A calculus of mobile processes (parts i and ii). *Inf. Comput.* **100** (1992) 1–77.

[22] J. Parrow, General conditions for full abstraction. *Math. Struct. Comput. Sci.* **26** (2016) 655–657.

[23] D. Pous and D. Sangiorgi, Enhancements of the bisimulation proof method, chap. Enhancements of the coinductive proof method. Cambridge University Press (2011).

[24] D. Sangiorgi, *Expressing mobility in process algebras: First-order and higher-order paradigms*. Ph.D. thesis, University of Edinburgh (1992).

[25] D. Sangiorgi, Bisimulation for higher-order process calculi. *Inf. Comput.* **131** (1996) 141–178.

[26] D. Sangiorgi, On the bisimulation proof method. *Math. Struct. Comput. Sci.* **8** (1998) 447–479.

[27] D. Sangiorgi, Introduction to Bisimulation and Coinduction. Cambridge University Press (2011).

[28] D. Sangiorgi, Concurrency theory: timed automata, testing, program synthesis. *Distrib. Comput.* **25** (2012) 3–4.

[29] D. Sangiorgi, N. Kobayashi and E. Sumii, Environmental bisimulations for higher-order languages. *ACM Trans. Prog. Lang. Syst.* **33** (2011) 5.

[30] D. Sangiorgi and J. Rutten, Advanced Topics in Bisimulation and Coinduction. Cambridge University Press (2012).

[31] D. Sangiorgi and D. Walker, The Pi-calculus: a Theory of Mobile Processes. Cambridge University Press (2001).

[32] B. Thomsen, *Calculi for higher order communicating systems*. Ph.D. thesis, Department of Computing, Imperial College (1990).

[33] B. Thomsen, Plain CHOCS, a second generation calculus for higher-order processes. *Acta Inf.* **30** (1993) 1–59.

[34] X. Xu, Higher-order processes with parameterization over names and processes. In: *Proceedings of Combined 23rd International Workshop on Expressiveness in Concurrency and 13th Workshop on Structural Operational Semantics (EXPRESS/SOS 2016), EPTCS* 222 (2016) 15–29.

[35] X. Xu, Distinguishing and relating higher-order and first-order processes by expressiveness. *Acta Inf.* **49** (2012) 445–484.

[36] X. Xu, On context bisimulation for parameterized higher-order processes. In: *Proceedings of the 6th Interaction and Concurrency Experience (ICE 2013), EPTCS* 131 (2013) 37–51.

[37] X. Xu, Q. Yin and H. Long, On the computation power of name parameterization in higher-order processes. In: *Proceedings of 8th Interaction and Concurrency Experience (ICE 2015). EPTCS* 189 (2015) 114–127.

[38] Q. Yin, X. Xu and H. Long, On parameterization of higher-order processes. *Int. J. Comput. Math.* **7**(94) (2017) 1451–1478.