Checking Equivalence for Higher Order Process

Yuxi Fu*

BASICS† Department of Computer Science
Shanghai Jiaotong University, Shanghai 200030, China

Abstract

In a higher order process calculus, recursion can be achieved by higher order communications. The Turing complete property rules out any algorithm to decide the equivalence between two higher order processes. The paper takes a look at a variant of the calculus of higher order processes that embodies the idea of resource sensitiveness. Resource sensitiveness means that a process received through a communication can only be used once. Two complete systems are given for the finite higher order π-processes and the finite higher order CCS processes. These systems immediately suggest how to derive algorithms to check the equivalences on the finite higher order processes.

Key Words: Process Calculus, Higher Order Mobile Process, Bisimulation

1 Motivation

Higher order features in process calculi are playing a more important role in the theory of mobile computing ([8, 17]). Part of the reason for the growing importance is the demand for theoretical frameworks to explain the ever broadening view of distributed and mobile computing. One recent model of mobile computing, the Ambient Calculus ([1]), studies the movements of particular processes, the ambients, from source processes to target processes. The interactions defined in the Ambient Calculus are of higher order. Previous theoretical investigations into the higher order processes include the work of Thomsen ([18, 19, 20, 21]) and Sangiorgi ([11, 12, 13]), to name a few. The models $CHOCS$ and Plain $CHOCS$ studied by Thomsen, and the model studied by Sangiorgi in [13] are essentially the Higher Order CCS. By Higher Order CCS we mean the pure CCS of synchronization ([7]) extended with the higher order communications of processes. In Higher Order CCS, processes can be passed around through communications, whereas names can not be exchanged over interactions. So there are two kinds of interactions: One is the higher order communication whose interactional behaviour is described by

$$a(X).E \,|\, \overline{a}H.F \xrightarrow{\tau} E\{H/X\} \,|\, F \tag{1}$$

and the other is the first order synchronization with semantics defined by

$$a.E \,|\, \overline{a}.F \xrightarrow{\tau} E \,|\, F \tag{2}$$

Two kinds of bindings are used by Thomsen. In the dynamic binding an occurrence of a bound name can escape the scope of the localization operator through a higher order communication. For instance

$$a(X).E \,|\, (z)\overline{a}H.F \xrightarrow{\tau} E\{H/X\} \,|\, (z)F$$

is dynamic binding since the name z in H becomes free after the higher order interaction. The dynamic binding, studied in $CHOCS$, takes a syntactical view of the localization operator. In the Plain $CHOCS$ static binding is adopted. Using the same process expression the static binding results in a different process expression:

$$a(X).E \,|\, (z)\overline{a}H.F \xrightarrow{\tau} (z)(E\{H/X\} \,|\, F)$$

*The author is supported by The National Distinguished Young Scientist Fund of NNSFC (60225012), The National 973 Project (2003CB317005) and The National Nature Science Foundation of China (60473006).

†Laboratory for **Ba**sic **S**tudies **i**n **C**omputing **S**cience (`http://basics.sjtu.edu.cn`).

Clearly a semantic approach is called for to define the static binding. Dynamic binding can be checked out at compile time, whereas the static binding has to be processed at runtime. In $CHOCS$ care should also be given to the relabelling operator. At first sight there also appears to be a dichotomy between dynamic binding and static binding for the relabelling. In the dynamic style the effect of the relabelling on the higher order communication can be seen from the following reductions:

$$a(X).E \,|\, (\overline{a}H.F)[b\mapsto c] \quad \stackrel{\tau}{\longrightarrow} \quad E\{H/X\} \,|\, F[b\mapsto c] \tag{3}$$

$$(a(X).E)[b\mapsto c] \,|\, \overline{a}H.F \quad \stackrel{\tau}{\longrightarrow} \quad (E\{H/X\})[b\mapsto c] \,|\, F \tag{4}$$

assuming that $a \neq b$. The process expression H can free itself from the range of the relabelling $[b\mapsto c]$ by higher order output. It can also get bound by $[b\mapsto c]$ through higher order input. How about the static binding? It seems to fit well with the higher order output:

$$a(X).E \,|\, (\overline{a}H.F)[b\mapsto c] \quad \stackrel{\tau}{\longrightarrow} \quad E\{H[b\mapsto c]/X\} \,|\, F[b\mapsto c] \tag{5}$$

However it does not appear to make any sense to talk about static relabelling for higher order input. If one defines the static version by brutal force, one gets the following reduction

$$(a(X).E)[b\mapsto c] \,|\, \overline{a}H.F \stackrel{\tau}{\longrightarrow} (E\{H[c\mapsto b]/X\})[b\mapsto c] \,|\, F$$

which is just not correct. We are thus led to adopt the dynamic relabelling. So was Thomsen.

In [11] Sangiorgi investigated a model of higher order π-processes. This model admits the first order communication studied in [8]:

$$a(x).E \,|\, \overline{a}y.F \stackrel{\tau}{\longrightarrow} E\{y/x\} \,|\, F \tag{6}$$

Throughout this paper we shall refer to the Higher Order CCS the language whose interaction semantics is defined by (2) and (1), and the Higher Order π-Calculus the model defined by (6) and (1). Both calculi are assumed to have static binding for the localization operator. The relationship between the first order π-calculus and the Higher Order π-Calculus has been studied by Sangiorgi in [12]. And that between CCS and the Higher Order CCS has been discussed by Thomsen in [18].

The algebraic theory of the higher order processes have been investigated by several researchers. Thomsen introduced the higher order bisimilarity $\approx^d_{ho}$ on the $CHOCS$ processes. The higher order bisimulations are straightforward generalizations of the bisimulations for CCS processes. It has been pointed out that $\approx^d_{ho}$ has too strong a discriminating power to be reasonable. The debatable bisimulation property of the higher order bisimulations is to do with the higher order output actions. In $CHOCS$ it is defined as follows:

(*) If $F \approx^d_{ho} E \stackrel{(\widetilde{y})\overline{a}G}{\longrightarrow} E'$ then $F \Longrightarrow \stackrel{(\widetilde{z})\overline{a}H}{\longrightarrow} F' \approx^d_{ho} E'$, $G \approx^d_{ho} H$ and $\widetilde{y} = \widetilde{z}$.

In (*) the equality $\widetilde{y} = \widetilde{z}$ means that $\widetilde{y}$ and $\widetilde{z}$ are the same finite set of names. The above clause can be rejected on several accounts. First of all the requirement $\widetilde{y} = \widetilde{z}$ is not really sound. A counter example is the pair of processes $(y)\overline{a}[(x)\overline{x}y]$ and $(y)\overline{a}[(x)\overline{x}x]$. They are obviously observationally equivalent. But they are not higher order bisimilar since $(y)\overline{a}[(x)\overline{x}y] \stackrel{(y)\overline{a}[(x)\overline{x}y]}{\longrightarrow} \mathbf{0}$ and $(y)\overline{a}[(x)\overline{x}x] \stackrel{\overline{a}[(x)\overline{x}x]}{\longrightarrow} (y)\mathbf{0}$ have incomparable action labels. A stronger argument against (*) is that neither $E' \approx^d_{ho} F'$ nor $G \approx^d_{ho} H$ is necessary. Assuming that $b \notin fn(P)$, the processes $(b)(\overline{a}[b] \,|\, \overline{b}.P)$ and $(b)(\overline{a}[\overline{b}] \,|\, b.P)$ are clearly observationally equivalent. But obviously neither $\overline{b}.P \approx^d_{ho} b.P$ nor $b \approx^d_{ho} \overline{b}$ holds. The bisimulations Thomsen studied are the delayed bisimulations. The delayed approach ([22]) does not consider any internal actions after an observable action. It was remarked by Thomsen that allowing internal actions after an observable action causes problem to the proof of the equivalence property. The higher order bisimulations for the $CHOCS$ and the Plain $CHOCS$ are not required to be closed under substitutions of names since there is no first order communication.

Sangiorgi studied in [13] a higher order CCS with abstractions. Technically this language bears similarity to $CHOCS$. Sangiorgi studied the context bisimilarity $\approx^d_{ct}$ in the delayed approach. Using our notation, the clause defining the simulation of higher order output actions is the following:

(**) If $F \approx^d_{ct} E \stackrel{(\widetilde{y})\overline{a}G}{\longrightarrow} E'$ then $F \Longrightarrow \stackrel{(\widetilde{z})\overline{a}H}{\longrightarrow} F' \approx^d_{ct} E'$ for some $\widetilde{z}, F'$ such that $(\widetilde{y})(K\{G/X\} \,|\, E) \approx^d_{ct}$ $(\widetilde{z})(K\{H/X\} \,|\, F)$ for every process expression $K[X]$ satisfying $\widetilde{y}\widetilde{z} \cap fn(K[X]) = \emptyset$.

In (**) the process expression $K[X]$ plays the role of a receiving environment. An important tool employed in [13] is the Factorization Theorem. It states that

$$E\{F/X\} \approx^{d}_{ct} (m)(E\{Tr_m/X\} \,|\, !\overline{m}.F) \tag{7}$$

where m does not appear in either E or F. In the above equivalence Tr_m is a trigger defined as $m.\mathbf{0}$. It is worth remarking that equation (7) fails in the presence of the unguarded summation. For instance $b+F$ is clearly not equivalent to $(m)((b+m) \,|\, !\overline{m}.F)$ in general. Factorization is the ability to achieve the effect of a process movement by sending a reference to the process.

Using the Factorization Theorem Sangiorgi was able to give a simpler characterization of the context bisimilarity in terms of the normal bisimulations ([13]). In the normal approach, to show that $a(X).E$ and $a(X).F$ are equivalent is the same as to showing that $E\{Tr_m/X\}$ and $F\{Tr_m/X\}$ are equivalent for a fresh m. In other words to simulate a higher order input it is enough to confine one's attention to the situations where the input processes are fresh triggers. In line with this, the simulations of the higher order output actions can be done in a way that the environments are totally ignored. One has that $(\widetilde{y})\overline{a}[G].E$ and $(\widetilde{z})\overline{a}[H].F$ are equivalent if and only if $(\widetilde{y})(E \,|\, !\overline{m}.G)$ and $(\widetilde{z})(F \,|\, !\overline{m}.H)$ are equivalent.

In [11] Sangiorgi investigated the Higher Order π-Calculus using the same machinery of [13]. The model admits abstraction and guarded summation. The context bisimilarity is studied in its general form. For example simulations of the higher order output actions are defined as follows:

> (***) If $F \approx_{ct} E \stackrel{(\widetilde{y})\overline{a}G}{\longrightarrow} E'$ then $F \stackrel{(\widetilde{z})\overline{a}H}{\Longrightarrow} F'$ for some F' such that $(\widetilde{y})(K\{G/X\} \,|\, E') \approx_{ct} (\widetilde{z})(K\{H/X\} \,|\, F')$ for every process expression $K[X]$ satisfying $\widetilde{y}\widetilde{z} \cap fn(K[X]) = \emptyset$.

The general definition poses some problems. As is already observed in [18] it is not obvious how to prove that $\approx_{ct}$ is an equivalence relation. To illustrate the problem suppose that $G \approx_{ct} F \approx_{ct} E \stackrel{(\widetilde{x})\overline{a}L}{\longrightarrow} E'$. By definition some $\widetilde{z}, M, F_1, F_2, F'$ exist such that $F \Longrightarrow F_1 \stackrel{(\widetilde{y})\overline{a}M}{\longrightarrow} F_2 \Longrightarrow F'$ and $(\widetilde{x})(K\{L/X\} \,|\, E') \approx_{ct} (\widetilde{y})(K\{M/X\} \,|\, F')$ for every process expression $K[X]$ with the process variable X. Again by definition some $\widetilde{z}, N, G_1, G_2$ exist such that $G \Longrightarrow G_1 \stackrel{(\widetilde{z})\overline{a}N}{\Longrightarrow} G_2$, $G_1 \approx_{ct} F_1$ and $(\widetilde{z})(K\{N/X\} \,|\, G_2) \approx_{ct} (\widetilde{y})(K\{M/X\} \,|\, F_2)$. Now $(\widetilde{y})(K\{M/X\} \,|\, F_2) \Longrightarrow (\widetilde{y})(K\{M/X\} \,|\, F')$ is simulated $(\widetilde{z})(K\{N/X\} \,|\, G_2) \Longrightarrow G_3$, which is not necessarily induced by $G_2 \Longrightarrow G'$ for some G'. So we can not conclude that $G \stackrel{(\widetilde{z})\overline{a}N}{\Longrightarrow} G'$ for some G' such that $(\widetilde{x})(K\{L/X\} \,|\, E') \approx_{ct} (\widetilde{z})(K\{N/X\} \,|\, G')$. In fact this problem can be easily bypassed using Bisimulation Lemma. Bisimulation Lemma was used in the studies of the χ-calculus ([2, 3, 5]) and the π-calculus ([4, 6]) to derive some crucial algebraic properties.

A prominent feature of (***) is that it is of a late style. An early counterpart is the following:

> (****) If $F \approx_{ct} E \stackrel{(\widetilde{y})\overline{a}G}{\longrightarrow} E'$ then for every process expression $K[X]$ satisfying $\widetilde{y} \cap fn(K[X]) = \emptyset$ some F' exists such that $F \stackrel{(\widetilde{z})\overline{a}H}{\Longrightarrow} F'$ and $(\widetilde{y})(K\{G/X\} \,|\, E') \approx_{ct} (\widetilde{z})(K\{H/X\} \,|\, F')$.

Sangiorgi has proved that the early and the late versions generate the same context bisimilarity. His proof exploits the coincidence between the context bisimilarity and the normal bisimilarity. Intuitively one may argue that (****) subsumes (***). Take for instance $K[X]$ to be the process expression $\overline{b}[X]$ for a fresh b. Then (****) implies that some F' exists such that $F \stackrel{(\widetilde{z})\overline{a}H}{\Longrightarrow} F'$ and

$$(\widetilde{y})(\overline{b}[G] \,|\, E') \approx_{ct} (\widetilde{z})(\overline{b}[H] \,|\, F') \tag{8}$$

It is clear that (8) is equivalent to saying that

$$\forall \text{ fresh } c.(\widetilde{y})(\overline{c}[G] \,|\, E') \approx_{ct} (\widetilde{z})(\overline{c}[H] \,|\, F') \tag{9}$$

From (9) and the congruence property the following equivalence follows:

$$(\widetilde{y})(\overline{c}[G] \,|\, E') \,|\, c(X).K[X] \approx_{ct} (\widetilde{z})(\overline{c}[H] \,|\, F') \,|\, c(X).K[X] \tag{10}$$

Now for each $K[X]$ pick up a name c that does not appear in $K[X]$. By appealing to the Bisimulation Lemma one gets from (10) that $(\widetilde{y})(K\{G/X\} \,|\, E') \approx_{ct} (\widetilde{z})(K\{H/X\} \,|\, F')$. The above argument also shows that (***) is equivalent to the simulation in a very early style:

> (*****) If $F \approx_{ct} E \stackrel{(\widetilde{y})\overline{a}G}{\longrightarrow} E'$ then for every process expression $K[X]$ satisfying $\widetilde{y} \cap fn(K[X]) = \emptyset$ some F', F'' exist such that $F \stackrel{(\widetilde{z})\overline{a}H}{\Longrightarrow} F''$ and $(\widetilde{z})(K\{H/X\} \,|\, F'') \Longrightarrow F' \approx_{ct} (\widetilde{y})(K\{G/X\} \,|\, E')$.

It should be pointed out that the coincidence of the late and the early higher order output bisimulations depends crucially on the property that we could forbid any interactions between an exported process, like G, and the environments for as long as we want. For higher order calculi with less controlling power, like the Ambient Calculus, the coincidence between the early semantics and the late semantics is not obvious.

In [11] there is a proof of the coincidence of the context bisimilarity and the barbed equivalence ([9, 16]). The proof makes use of an infinite number of processes recursively defined, each of the processes using an infinite number of free names. The limitation of admitting recursive definitions is that, assuming the set of names is countable, one can always define by recursive definition processes that exhaust all (free) names. It is not clear how to define static contexts to tell apart two such processes that are not context bisimilar. Moreover if one admits only fixpoint operator but not recursive definition, every process expression has only finite number of free names. Using Bisimulation Lemma, infinite recursive definitions can be avoided in the coincidence proof.

The context bisimilarity is not closed under first order input prefix operation. Two processes E and F are congruent if $C[E] \approx_{ct} C[F]$ for every context $C[\_]$. This is the congruence relation studied in [11, 13]. There is another standard approach to define congruence equivalence in the calculi of mobile processes. The open bisimulations ([14]) imposes the condition that bisimilar processes are dynamically tested by the environments. That is to say that after each bisimulation step the environment might have totally changed. The open bisimilarity enjoys many good properties. One such good property is that it has clean equational systems. For the Higher Order π-Calculus the open style bisimulations have not be discussed.

As is in the λ-calculus the higher order mechanism provides a computable power that is Turing complete. The following communication is a process version of the computation of the λ-term $(\lambda x.xx)(\lambda x.xx)$:

$$(a)(a(X).(X \mid \overline{a}X) \mid \overline{a}[a(X).(X \mid \overline{a}X)]) \xrightarrow{\tau} (a)(a(X).(X \mid \overline{a}X) \mid \overline{a}[a(X).(X \mid \overline{a}X)]) \tag{11}$$

By modifying this example one can easily define a higher order process $!P$ that enjoys the following operational property:

$$!P \xrightarrow{\tau} P \mid !P \xrightarrow{\tau} P \mid P \mid !P \xrightarrow{\tau} \ldots$$

In $CHOCS$ a process transported through a higher order communication could get bound by or escape the scope of a relabelling operator. This property is exploited in Thomsen's definition of the fixpoint operator in [20]. The Y-operator is defined in the following way:

$$\mathbf{Y}_E \stackrel{\text{def}}{=} (a)(a(X).E[(b)(X[a{\mapsto}b] \mid \overline{b}X)] \mid \overline{a}[a(X).E[(b)(X[a{\mapsto}b] \mid \overline{b}X)]])$$

In the above definition a, b are fresh. It is not difficult to see that

$$\mathbf{Y}_E \xrightarrow{\tau} E(\mathbf{Y}_E) \tag{12}$$

For the reduction to work it is important that $X[a{\mapsto}b]$ is not the same as X. The fundamental difference between substitution and relabelling is that the former is a syntactic operation and the latter is a semantic one. These examples explain why in the higher order process calculi the replication operator or the **fix**-operator are not really necessary. At the same time however the Turing computability of the model defeats all attempts to find an algorithm to check the equivalence of higher order processes. But is there any hope for a restricted goal on decidability? Consider the linear higher order processes. These are the processes in which two copies of a same process variable can never be in a concurrent situation. For instance $a.X + \overline{b}.X$ is a linear process but $a.X \mid \overline{b}.X$ is not. Using the linear processes one can define neither the reduction in (11) nor the reduction in (12). Now the question is: Is there a complete equational system for the context congruence on the finite linear processes? There doesn't appear to be an easy answer. For one thing, it is not known whether it is tractable to test the bisimilarity between the linear processes $(\widetilde{x})\overline{a}[A].P$ and $(\widetilde{y})\overline{a}[B].Q$. A test in the Higher Order π-Calculus will have to place the processes A and B in the contexts where there could be multiple references to A and B. This would immediately break down the restriction on linearity. The normal bisimilarity does not provide any clue either since the explicit replications in $(\widetilde{x})(P \mid !\overline{m}.A)$ and $(\widetilde{y})(Q \mid !\overline{m}.B)$ make things worse. We are forced to impose a stronger restriction: Not only we focus on the linear processes, we also confine to the linear environments. This takes us to a calculus of linear higher order π-processes. It is interesting to see that $(x)\overline{a}[x.\mathbf{0}].\overline{x}.b$ and $(x)\overline{a}[x.\mathbf{0}].\overline{x}.b.\overline{x}.b$ are bisimilar from the viewpoint of the linear environments, but they are not bisimilar in the Higher Order π-Calculus. In the linear scenario, the higher order communications are not able to induce infinite computations. Recursion mechanism need be explicitly introduced into the model. So two programming

styles are present in the models of linear processes. The higher order communications are typical of the functional programming. In the higher order communications private information is exchanged. The recursion mechanism support the object oriented programming style. Universal information is made public by processes in the replication form.

Pragmatically the linear higher order processes have abundant applications in modern computing environments. In commercial applications, platforms are designed to support resource sensitive usages of softwares and computing powers. This happens in the form of gaming, distance learning, video on demand and so on. The mechanism design of buying and selling computing resources is also a key issue in grid computing. In a network computing scenario, computing resources are subject to bargain. Using a piece of resource a couple of time is definitely not the same as using it once. On the theoretical side, models like Ambient Calculus have been proposed in literature to formalize the situation of a piece of code moving from one site to another. After arriving at the target site, the piece of codes is located alongside of the other programmes. In Ambient Calculus imported programs can never be duplicated. The guideline is that a consumer can not duplicate any resource it has got through exchange, whereas a provider may access to a resource in a nonrestricted manner.

The main objective of the paper is to study the algorithmic aspect of the finite higher order processes. This is achieved by constructing complete equational systems. The framework we are working with is the Linear Higher Order π-Calculus. Along the way to reach a completeness result, we will be proving a number of intermediate results that are of interest on their own right. In order for the results and the proof techniques to be more applicable, we set up the framework in a general fashion. The main technical points and contributions are as follows:

- We will be working with the unguarded summation operator. The purpose is to see how much of the theory developed in [11] survives without the help of the Factorization Theorem. This is important since the Factorization Theorem is not a very stable property. In some model of higher order interactions, say the Ambient Calculus, Factorization Theorem does not really make sense.

- We will focus on the general bisimulations rather than the delayed bisimulations. This makes it natural the extension of the first order theory to the higher order theory. The Bisimulation Lemma plays a crucial role in the algebraic theory of the higher order processes. The proof of say the fact that the general bisimilarity is an equivalence relation makes heavy use of the lemma.

- For the Linear Higher Order π-Calculus we will study the open style bisimilarity. The open bisimulations, introduced by Sangiorgi ([14, 16]), refer to the property of closure under the substitution of names. This open style bisimilarity relates to the dynamic barbed bisimialrity ([16]) and thus plays the role of being the authentic bisimilarity of the Linear Higher Order π-Calculus.

- As a major result we construct a complete equational system for the finite $LHO\pi$ processes. The systems are distinguished in that they have rules dealing with higher order prefix operators. The power of these rules is to transform the higher order complexity to the first order one. Based upon the completeness results, algorithms are designed to check up the equivalences between finite linear higher order π-processes as well as finite linear higher order CCS processes

Although the proofs in this paper are for the Linear Higher Order π-Calculus, most of the algebraic results actually hold for the Higher Order π-Calculus.

For the uniformity of terminology, we will in the rest of the paper write $HOCCS$ for the Higher Order CCS, $LHOCCS$ for the Linear Higher Order CCS, $HO\pi$ for the Higher Order π-Calculus, and $LHO\pi$ for the Linear Higher Order π-Calculus.

The paper is structured as follows: Section 2 defines the Linear Higher Order π-Calculus. Section 3 introduces the local bisimilarity for the linear higher order π-processes and establishes the equivalence and congruence properties. Section 4 studies the relationship between the equivalence of the prefixed processes and the equivalence of the continuations. The results of the section lays down the basics for the equivalence checking. Section 5 discusses some local congruence properties. Section 6 establishes the standard properties for the recursive processes. Section 7 takes a look at the open bisimulations for $LHO\pi$. Section 8 discusses head normal forms for the finite linear higher order π-processes. Section 9 proposes two rules to guarantee the saturation property. Section 10 proves the completeness theorem. Section 11 describes the checking algorithm. Section 12 transplants the completeness result of $LHO\pi$ to $HOCCS$. Section 13 summarizes.

2 Linear Higher Order π-Calculus

This section defines the semantics of $LHO\pi$. Let $\mathcal{N}$ be the set of names ranged over by the small letters, and $\mathcal{V}$ be the set of process variables, or variables for short, ranged over by $X, Y, Z, \ldots$. The notation $\overline{\mathcal{N}}$ denotes the set $\{\overline{a} \mid a \in \mathcal{N}\}$ of co-names. The union $\mathcal{N} \cup \overline{\mathcal{N}}$ will be ranged over by α. The set $\mathcal{LHPE}$ of linear higher order process expressions, or just process expressions, ranged over by $E, F, G, H, \ldots$, is defined by the following abstract grammar:

E	:=	X	
		$a(x).E$	a is the subject name; x is not a free name
		$\overline{a}x.E$	a is the subject name; x is a free name
		$a(X).E$	a is the subject name; X is not a free variable
		$\overline{a}[E].E'$	a is the subject name; $fv(E) \cap fv(E') = \emptyset$
		$E \mid E'$	$fv(E) \cap fv(E') = \emptyset$
		$(x)E$	x is a local name; x is not a free name
		$[x{=}y]E$	
		$E{+}E'$	
		$\mathbf{fix}X.E$	X is isolated in E; X is not a free variable

where the property of "X being isolated in E" is inductively defined as follows:

- X is isolated in X;
- If X is isolated in E, then X is isolated in $a(x).E$, $\overline{a}x.E$, $a(Y).E$, $(x)E$, $[x{=}y]E$ and $\mathbf{fix}Y.E$;
- If X is isolated in E and E', then X is isolated in $E + E'$;
- If X is isolated in E and P does not contain any free process variables, then X is isolated in $\overline{a}[E].P$, $\overline{a}[P].E$, $E \mid P$ and $P \mid E$.

The process expressions are built up from the process variables and the standard process operators. The expressions have their usual interpretations. For instance $\overline{a}[E].F$ is a higher order output process expression that intends to export E through channel a. The linearity means that whenever $E_1 \mid E_2$ is a sub-term of E then $fv(E_1) \cap fv(E_2) = \emptyset$.

Throughout the paper we will use the convention that all non-free names (variables) are pairwise distinct and are different from any free name (variable). We write $fn(E)$, respectively $fv(E)$, for the set of free names, respectively the set of free variables, of E. A finite sequence $x_1, \ldots, x_n$ of names is often abbreviated to $\widetilde{x}$. For simplicity we shall also write $\widetilde{x}$ for the set $\{x_1, \ldots, x_n\}$. In this paper we shall say that $a(x)$ binds x in $a(x).E$, (x) localizes x in $(x)E$, and $a(X)$ binds X in $a(X).E$.

For higher order processes two kinds of substitutions are necessary.

- A first order substitution $\{m_1/x_1, \ldots, m_n/x_n\}$ is a function from $\mathcal{N}$ to $\mathcal{N}$ that maps a name $x_i \in \{x_1, \ldots, x_n\}$ onto m_i and maps any other name onto itself. The notation $n(\sigma)$ denotes $\{x_1, \ldots, x_n, m_1, \ldots, m_n\}$ if $\sigma = \{m_1/x_1, \ldots, m_n/x_n\}$. The set of the first order substitutions will be ranged over by $\sigma, \sigma', \sigma'', \ldots$.
- A higher order substitution $\{E_1/X_1, \ldots, E_n/X_n\}$ is a function from $\mathcal{V}$ to $\mathcal{LHPE}$. It maps X_i, for $i \in \{1, \ldots, n\}$, onto E_i and maps any other variable to itself. The set of the higher order substitutions will be ranged over by $\Sigma, \Sigma', \Sigma'', \ldots$.

Applications of a substitution to a term is denoted in a postfix fashion. We postulate that $X\sigma = X$. This is to say that the meta operation has an instantaneous effect. If substitutions are explicit, meaning that it is a built-in operator like the relabelling functions in CCS, then $X\sigma = X$ is definitely false. The slogan is: *substitutions are syntactical*. It is also in line with the fact that in the first order substitutions where the names to be replaced were bound names. A process variable should be instantiated by a process expression that avoids name capture since a foreign process does not know the local information represented by the bound names. Sometimes we write $E[X_1, \ldots, X_n]$ to make it explicit that E might contain the free variables $X_1, \ldots, X_n$. To go along with this notation we write $E[E_1, \ldots, E_n]$ for $E[X_1, \ldots, X_n]\{E_1/X_1, \ldots, E_n/X_n\}$. We say that a substitution $\{E_1/X_1, \ldots, E_n/X_n\}$ in E is well-defined if the following two conditions are met: (i) the free names (variables) in $E_1 \mid \ldots, \mid E_n$

are not captured by any bound name (variable) in E; and (ii) the substitution does not create any non-linear process expressions. In the rest of the paper all references to the higher order substitutions are assumed to be well-formed.

A binary relation $\mathcal{R}$ is closed under substitution of names if for each first order substitution σ, $(E\sigma, F\sigma) \in \mathcal{R}$ whenever $(E, F) \in \mathcal{R}$. A binary relation $\mathcal{R}$ is closed under substitution of variables if, for each higher order substitution Σ, $(E\Sigma, F\Sigma) \in \mathcal{R}$ whenever $(E, F) \in \mathcal{R}$ and $E\Sigma, F\Sigma$ are well-formed.

In the definition of the operational semantics we need to be able to say that two variables are in, or not in, a concurrent position. For that purpose we introduce the following definition:

$$
\begin{aligned}
cp(E, X) &\stackrel{\text{def}}{=} \emptyset, \quad \text{if } X \notin fv(E) \text{ or } E \equiv X \\
cp(a(x).E, X) &\stackrel{\text{def}}{=} cp(E, X) \\
cp(\overline{a}x.E, X) &\stackrel{\text{def}}{=} cp(E, X) \\
cp(a(Y).E, X) &\stackrel{\text{def}}{=} cp(E, X) \\
cp(\overline{a}E'.E, X) &\stackrel{\text{def}}{=} \begin{cases} fv(E', X) \cup cp(E, X), & \text{if } X \in fv(E) \\ fv(E, X) \cup cp(E', X), & \text{if } X \in fv(E') \end{cases} \\
cp(E \mid E', X) &\stackrel{\text{def}}{=} \begin{cases} fv(E', X) \cup cp(E, X), & \text{if } X \in fv(E) \\ fv(E, X) \cup cp(E', X), & \text{if } X \in fv(E') \end{cases} \\
cp((x)E, X) &\stackrel{\text{def}}{=} cp(E, X) \\
cp([x{=}y]E, X) &\stackrel{\text{def}}{=} cp(E, X) \\
cp(E{+}E', X) &\stackrel{\text{def}}{=} cp(E, X) \cup cp(E', X) \\
cp(\mathbf{fix}Z.E, X) &\stackrel{\text{def}}{=} cp(E, X)
\end{aligned}
$$

Intuitively $cp(E, X)$ is the set of variables in E that could be in concurrent position with X.

The operational semantics of $LHO\pi$ is defined by the following labeled transition system on $\mathcal{LHPE}$:

Prefix

$$
\frac{}{a(x).E \xrightarrow{ay} E\{y/x\}} \qquad \frac{}{\overline{a}x.E \xrightarrow{\overline{a}x} E} \qquad \frac{fv(H) \cap cp(E, X) = \emptyset}{a(X).E \xrightarrow{aH} E\{H/X\}} \qquad \frac{}{\overline{a}[H].E \xrightarrow{\overline{a}H} E}
$$

Composition

$$
\frac{E \xrightarrow{\lambda} E'}{E \mid F \xrightarrow{\lambda} E' \mid F} \qquad \frac{E \xrightarrow{ax} E' \quad F \xrightarrow{\overline{a}x} F'}{E \mid F \xrightarrow{\tau} E' \mid F'} \qquad \frac{E \xrightarrow{ax} E' \quad F \xrightarrow{\overline{a}(x)} F'}{E \mid F \xrightarrow{\tau} (x)(E' \mid F')} \qquad \frac{E \xrightarrow{aH} E' \quad F \xrightarrow{(\widetilde{x})\overline{a}H} F'}{E \mid F \xrightarrow{\tau} (\widetilde{x})(E' \mid F')}
$$

Localization

$$
\frac{E \xrightarrow{\lambda} E' \quad x \notin fn(\lambda)}{(x)E \xrightarrow{\lambda} (x)E'} \qquad \frac{E \xrightarrow{\overline{a}x} E' \quad x \neq a}{(x)E \xrightarrow{\overline{a}(x)} E'} \qquad \frac{E \xrightarrow{(\widetilde{x})\overline{a}H} E' \quad y \in fv(H) \setminus \widetilde{x}a}{(y)E \xrightarrow{(y)(\widetilde{x})\overline{a}H} E'}
$$

Condition

$$
\frac{E \xrightarrow{\lambda} E'}{[x{=}x]E \xrightarrow{\lambda} E'}
$$

Choice

$$
\frac{E \xrightarrow{\lambda} E'}{E{+}F \xrightarrow{\lambda} E'}
$$

Recursion

$$
\frac{E\{\mathbf{fix}X.E/X\} \xrightarrow{\lambda} E'}{\mathbf{fix}X.E \xrightarrow{\lambda} E'}
$$

In the composition rules we should make sure that $fv(E) \cap fv(E') = \emptyset$. All the symmetric rules are omitted. For clarity we shall often write $E \xrightarrow{\overline{a}[H]} E'$ for $E \xrightarrow{\overline{a}H} E'$.

For the operational semantics to be well-defined, one needs to show that no actions create nonlinear process expressions. In particular the third prefix rule preserves linearity and the recursion rule does not lead to nonlinearity. These simple properties are guaranteed by the following two lemmas.

Lemma 1. *The following properties hold:*
(i) If $E[X]$ and $F[Y]$ are linear process expressions and $fv(F) \cap cp(E, X) = \emptyset$, then $E[F]$ is a linear process expression.
(ii) If X is isolated in $E[X]$ and Y is isolated in $F[Y]$ then Y is isolated in $E[F[Y]]$.

The proof of the above lemma is an easy structural induction. Apart from using structural inductions, proofs in process calculi often make use of inductions on the height of derivation trees. Since the operational semantics is defined to assign meanings to the operator, inductions on derivations are reincarnated in the form of structural inductions. But bear in mind that structural induction does not work in the presence of the **fix**-operator. The following lemma is a simple induction on derivations.

Lemma 2. *Suppose E and H are linear higher order process expressions. The following properties hold:*
(i) If $E \xrightarrow{\lambda} E'$ and λ is τ or a first order action then E' is a linear higher order process expression;
(ii) If $E \xrightarrow{aH} E'$ then E' is a linear higher order process expression;
(iii) If $E \xrightarrow{\overline{a}G} E'$ then both G and E' are linear higher order process expressions.

Proof. The proof is by induction on the height of the derivations. Use Lemma 1 in the proof of (ii); and use (ii) and (iii) in the proof of (i). □

Throughout the paper the nil process $\mathbf{0}$ abbreviates $(x)x(X).X$. We shall often omit the process $\mathbf{0}$ whenever possible. For instance we shall abbreviate $\lambda.\mathbf{0}$ to λ, $E \,|\, \mathbf{0}$ to E, $(x)\mathbf{0}$ to $\mathbf{0}$ and $E+\mathbf{0}$ to E. There are three kinds of derived prefix operators. One is the higher order bound output prefix as in $(\widetilde{x})\overline{a}H.E$. The other two are the tau prefix and the first order bound output prefix defined below:

$$\begin{aligned} \tau.E &\stackrel{\text{def}}{=} (b)(b(x) \,|\, \overline{b}b.E), \text{ where } b \text{ is fresh} \\ \overline{a}(x).E &\stackrel{\text{def}}{=} (x)\overline{a}x.E, \text{ where } a \neq x \end{aligned}$$

We will frequently use the synchronization notation of the pure CCS defined as follows:

$$\alpha.E \stackrel{\text{def}}{=} \alpha(x).E, \text{ where } x \notin fn(E)$$

According to these abbreviations, $E\{a/X\}$ stands for the process expression obtained by replacing X by $a(x).\mathbf{0}$ in E. We will also write λ to range over the set of all prefix operators, including the derived prefix operators. We will use $\Longrightarrow$ and $\stackrel{\hat{\lambda}}{\Longrightarrow}$ in their standard meanings. A finite sequence of match operations concatenated one after another, called a condition, is often denoted by ϕ, φ, ψ. For a condition φ we denote by σ_φ an arbitrarily chosen substitution generated by φ. The capital letters $A, B, C, O, P, Q, R, S, T$ will denote processes, which are process expressions that do not contain any free process variables. The process expression $a(X).X+X$ for instance is a process. The set of processes is denoted by $\mathcal{LHP}$.

Let $!\lambda.E$ be defined by $\mathbf{fix}X.\lambda.(E \,|\, X)$. It is clear that $!\lambda.E$ is the bounded replication of the first order π-calculus. Even with the condition of E being isolated, the recursion $\mathbf{fix}X.E$ is still very expressive. Let A stand for the process $\mathbf{fix}X.(\tau+b \,|\, X)$. It is clear that $A \xrightarrow{b} A$. That is A may perform a sequence of b actions before terminating itself with a tau action. But this is not all the operational behaviours of A. The following is a legal derivation of the operational semantics:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\tau+b \,|\, A \xrightarrow{\tau} \mathbf{0}}}{A \xrightarrow{\tau} \mathbf{0}}}{b \,|\, A \xrightarrow{\tau} b}}{\tau+b \,|\, A \xrightarrow{\tau} b}}{A \xrightarrow{\tau} b}}{\vdots}}{A \xrightarrow{\tau} b \,|\, \ldots \,|\, b}$$

So A can make an internal choice on the number of b-actions it is going to perform in sequel. In summary A may do a finite number of b-actions, followed by a nondeterministic internal choice, which decides how many b-actions it will do before termination. A more interesting example of using the **fix**-operator is the process $(a)\mathbf{fix}X.(a.b\,|\,(\tau.X+\mathbf{fix}Y.\overline{a}.Y))$. This process can perform n consecutive τ-actions, reaching a state from which the b-actions can be performed precisely n-times.

In the study of $LHO\pi$ we need to consider substitution to guarantee the congruence. The next three lemmas record the properties we shall be using in later investigations. The proofs of these properties are simple inductions on derivation.

Lemma 3. *If $E \xrightarrow{\lambda} E'$ then $E\sigma \xrightarrow{\lambda\sigma} E'\sigma$.*

Lemma 4. *Suppose that E contains the free variables $X_1,\ldots,X_n$ and that $E_1,\ldots,E_n$ are process expressions. If $E \xrightarrow{\lambda} E'$ then $E\{E_1/X_1,\ldots,E_n/X_n\} \xrightarrow{\lambda\{E_1/X_1,\ldots,E_n/X_n\}} E'\{E_1/X_1,\ldots,E_n/X_n\}$.*

Lemma 5. *Suppose that E contains the free variables $X_1,\ldots,X_n$ and that $b_1,\ldots,b_n$ do not appear in any of E,λ,E'. If $E\{b_1/X_1,\ldots,b_n/X_n\} \xrightarrow{\lambda\{b_1/X_1,\ldots,b_n/X_n\}} E'\{b_1/X_1,\ldots,b_n/X_n\}$ then $E \xrightarrow{\lambda} E'$.*

3 Bisimulation

For technical reason we need to introduce an equivalence that identifies process expressions with only structural difference. There are several ways to define such a relation. The following bisimulation approach is the most convenient one when it comes to formal proofs.

Definition 6 (Structural Equivalence)**.** A symmetric relation $\mathcal{R}$ on process expressions is a structural bisimulation if it is closed under substitution of names and variables and whenever $E\mathcal{R}F$ then the following properties hold:
(i) If $E \xrightarrow{\lambda} E'$ and λ is not a higher order output action then F' exists such that $F \xrightarrow{\lambda} F'\mathcal{R}E'$.
(ii) If $E \xrightarrow{(\widetilde{x})\overline{a}G} E'$ then some $\widetilde{x'}, H, F'$ exist such that $F \xrightarrow{(\widetilde{y})\overline{a}H} F'\mathcal{R}E'$, $G\mathcal{R}H$ and $\widetilde{x}=\widetilde{y}$.
The structural equivalence $\sim$ is the largest structural bisimulation.

The structural bisimilarity is precisely Thomsen's strong higher order bisimilarity ([20]). It is our opinion that $\sim$ is best seen as a structural equivalence than an observational equivalence. We will make full use of the fact that the structural equivalence is both an equivalence relation and a congruence relation. The structural equivalence satisfies the well known structural equalities between processes.

From now on we focus on the observational equivalences on the process expressions. In the observational approach processes are compared by their observable actions. The next definition formalizes the notion of observability.

Definition 7. A process expression E is observable at a non-tau action λ, notation $E\Downarrow\lambda$, if $E \Longrightarrow \xrightarrow{\lambda} E'$ for some E'. A process expression E is observable, notation $E\Downarrow$, if it is observable at a non-tau action λ. A binary relation on the process expressions is observed if $E\Downarrow \Leftrightarrow F\Downarrow$ whenever $E\mathcal{R}F$.

The idea of the observational equivalences is to place processes in the environments and then make observations on the consequences. The environments can be formalized as contexts.

Definition 8. Contexts are defined inductively as follows: (i) $[\_]$ is a context; (ii) If $C[\_]$ is a context then $\lambda.C[\_]$, $E\,|\,C[\_]$, $C[\_]\,|\,E$, $(x)C[\_]$, $[x{=}y]C[\_]$ are contexts. Full contexts are defined inductively as follows: (i) a context is a full context; (ii) if $C[\_]$ is a full context then $\overline{a}[C[\_]].E$, $C[\_]{+}E$, $E{+}C[\_]$ and $\mathbf{fix}X.C[\_]$ are contexts. A local context is a context of the form $(\widetilde{x})(\_\,|\,O)$.

A context is an environment that can make finite observations. From a practical point of view it makes sense to assume that each observation is finite, but the number of the potential observations is infinite. Notice that a context $C[\_]$ is different from a process expression $E[X]$. In $C[F]$ free names and variables in F may get bound in $C[F]$. In $E[F]$ however one must make sure that no free names and variables are captured by E.

The local contexts play a crucial role in the theory of π-calculus. Placing E in a local context $(x_1)\ldots(x_n)(O\,|\,\_)$ gives rise to the process expression

$$(x_1)\ldots(x_n)(O\,|\,E)$$

It describes a situation where E shares the local names $x_1, \ldots, x_n$ with the environment. The environment may export the local names to foreigners. Whomever are told of the local information will keep them secret. In the π-calculus a local name remains a local name forever. A local context never disappears. Once a process expression is placed in a local context, it never escapes. The importance of the local contexts is precise because they can not be removed. We shall see that in the algebraic theory we need to reason about the equivalence of process expressions qua local contexts. More often a local context is represented by the following standard form:

$$(x_1)\ldots(x_n)(\overline{c_1}x_1 \mid \ldots \mid \overline{c_n}x_n \mid \_)$$

where $c_1, \ldots, c_n$ are distinct names. The above context is general enough since we can choose $c_1, \ldots, c_n$ to be distinct from the free names of the process expressions in consideration. Such a context incorporates both the persistent and the dynamic aspects of the local names. In sequel we often abbreviate $\overline{c_1}x_1 \mid \overline{c_2}x_2 \mid \ldots \mid \overline{c_n}x_n$ to $\widetilde{\overline{c}x}$ for pairwise distinct $c_1, \ldots, c_n, x_1, \ldots, x_n$.

To investigate the observational theory we begin with the definition of an observational equivalence that imposes the least requirements. The first requirement is that the relation should be closed under context. This is natural because equivalent processes must have non-distinguishable interactive behaviours with environments. The second one is that the equivalence should be an observed relation. This is the minimal condition for an observational equivalence. The third one characterizes the equivalence as a bisimulation relation. Bisimulation is a robust property against hostile environments.

Definition 9. A symmetric binary relation $\mathcal{R}$ on processes is an observed bisimulation if
(i) it is closed under context;
(ii) it is observed;
(iii) and $Q \Longrightarrow Q'\mathcal{R}P'$ for some Q' whenever $Q\mathcal{R}P \xrightarrow{\tau} P'$.
The observed bisimilarity $\approx_o$ is the largest observed bisimulation.

Definition 9 imposes three *minimal* conditions on an observational bisimulation equivalence: (i) is the smallest requirement for a well behaved *equivalence*; (ii) is the least condition for an *observational* equivalence; and (iii) is the most basic property of a *bisimulation* equivalence.

Here are some examples of observed bisimilarity. The process $(x)\overline{a}[\overline{x}].x.b$ is observed bisimilar to the process $(x)\overline{a}[x].\overline{x}.b$. This example serves to show that $(\widetilde{y})\overline{a}[A].P \approx_o (\widetilde{y})\overline{a}[B].Q$ does not imply either $A \approx_o B$ or $P \approx_o Q$. However $\overline{a}[A].P \approx_o \overline{a}[B].Q$ does imply $P \approx_o Q$ and, in the event that P is finite, $A \approx_o B$ as well. The second example is that the process $\overline{a}[x.b].!x.b$ is observed bisimilar to the process $\overline{a}[\mathbf{0}].!x.b$. The higher order output produces a redundant copy of $x.b$, which amounts to outputting nothing. The third example is due to the linearity of the calculus: The process $(x)\overline{a}[\overline{x}].x.b$ is observed bisimilar to the process $(x)\overline{a}[\overline{x}].(x.b \mid x.b)$. It is interesting to notice that had we defined the operational semantics of the higher order output prefix in the following call-by-value style

$$\frac{H \xrightarrow{\tau} H'}{\overline{a}[H].E \xrightarrow{\tau} \overline{a}[H'].E}$$

we would have had the property that $\overline{a}[A].P \approx_o \overline{a}[B].Q$ implies both $P \approx_o Q$ and $A \approx_o B$.

The observed bisimilarity is defined on $\mathcal{LHP}$. We can extend the above definition to the process expressions in the following manner: Suppose $E \mid F$ contains process variables $X_1, \ldots, X_n$. Then $E \approx_o F$ if and only if

$$E\{P_1/X_1, \ldots, P_n/X_n\} \approx_o F\{P_1/X_1, \ldots, P_n/X_n\}$$

for all processes $P_1, \ldots, P_n$.

The nice thing about Definition 9 is that it does not distinguish between any two observable actions, which makes it a good candidate for a standard when comparing different calculi. The definition of the observed bisimilarity draws similarity to that of the barbed bisimilarity ([9, 16]). The former appears weaker and more general than the latter. The arguments against Definition 9 could be that the equivalence it introduces might be too weak and that the definition is too general to work with. In the rest of the section an alternative characterization of the observed bisimilarity is given to support Definition 9.

Definition 10. A symmetric relation $\mathcal{R}$ on $\mathcal{LHPE}$ is a local bisimulation if it is closed under substitution of names and whenever $P\mathcal{R}Q$ then the following properties hold:
(i) If $P \xrightarrow{\tau} P'$ then Q' exists such that $Q \Longrightarrow Q'\mathcal{R}P'$.

(ii) If $P \xrightarrow{\alpha x} P'$ then Q' exists such that $Q \stackrel{\alpha x}{\Longrightarrow} Q'\mathcal{R}P'$.
(iii) If $P \xrightarrow{\overline{a}(x)} P'$ then Q' exists such that $Q \stackrel{\overline{a}(x)}{\Longrightarrow} Q'$ and $(x)(P' \mid O)\ \mathcal{R}\ (x)(Q' \mid O)$ for all O.
(iv) If $P \xrightarrow{aA} P'$ then Q' exists such that $Q \stackrel{aA}{\Longrightarrow} Q'\mathcal{R}P'$.
(v) If $P \xrightarrow{(\widetilde{x})\overline{a}A} P'$ then some $\widetilde{x'}, B, Q'$ exist such that $Q \stackrel{(\widetilde{x'})\overline{a}B}{\Longrightarrow} Q'$ and $(\widetilde{x})(E[A] \mid P')\ \mathcal{R}\ (\widetilde{x'})(E[B] \mid Q')$ for every process expression $E[X]$ with free variable X such that $\widetilde{x}\widetilde{x'} \cap fn(E[X]) = \emptyset$.
P is local bisimilar to Q, notation $P \approx_l Q$, if there exists a local bisimulation $\mathcal{R}$ such that $(P, Q) \in \mathcal{R}$.

Clause (iii) takes into account that the local name x might be sent to any foreign process whatsoever. The residuals, P', Q' must be considered together with the receiving party. Notice that (iii) appears in late style. An early version would be stated like this:

> (iii') If $P \xrightarrow{\overline{a}(x)} P'$ then for every O some Q', Q'' exist such that $Q \Longrightarrow \xrightarrow{\overline{a}(x)} Q''$ and $(x)(Q'' \mid O) \Longrightarrow Q'\ \mathcal{R}\ (x)(P' \mid O)$.

Clause (iv) is in an early fashion. The late version of (iv) is as follows:

> (iv') If $P \xrightarrow{aX} E'$ and $X \notin fv(E|F)$ then F' exists such that $Q \stackrel{aX}{\Longrightarrow} F'$ and $E'\{A/X\}\mathcal{R}F'\{A/X\}$ for all A.

Clause (v) is of a late style. Its early counterpart can be stated as follows:

> (v') If $P \xrightarrow{(\widetilde{x})\overline{a}A} P'$ then for every process expression $E[X]$ with free variable X such that $\widetilde{x}\widetilde{x'} \cap fn(E[X]) = \emptyset$ there exist some $\widetilde{x'}, B, Q', Q''$ such that $Q \Longrightarrow \xrightarrow{(\widetilde{x'})\overline{a}B} Q''$ and $(\widetilde{x'})(E[B] \mid Q'') \Longrightarrow Q'\ \mathcal{R}\ (\widetilde{x})(E[A] \mid P')$.

Late simulation is desirable since it reduces the complexity of equivalence checking. It is one of the key properties underpinning our algorithmic approach to higher order equivalence checking. It will become clear that (iii), (iv) and (v) are equivalent to (iii'), (iv') and (v') respectively. The proofs of these facts make extensive use of the following lemma.

Lemma 11 (Bisimulation Lemma)**.** *If $E \Longrightarrow\approx_l F$ and $F \Longrightarrow\approx_l E$ then $E \approx_l F$.*

Proof. Suppose that E, F contain the free variables $X_1, \ldots, X_n$. It follows from Lemma 4 that

$$E\{A_1/X_1, \ldots, A_n/X_n\} \Longrightarrow\approx_l F\{A_1/X_1, \ldots, A_n/X_n\}$$

and

$$F\{A_1/X_1, \ldots, A_n/X_n\} \Longrightarrow\approx_l E\{A_1/X_1, \ldots, A_n/X_n\}$$

for all processes $A_1, \ldots, A_n$. It is clear that an action of $F\{A_1/X_1, \ldots, A_n/X_n\}$ can be simulated by $E\{A_1/X_1, \ldots, A_n/X_n\}$ by first performing a sequence of internal interactions to reach a state locally bisimilar to $F\{A_1/X_1, \ldots, A_n/X_n\}$, and then simulating the action according to the definition of $\approx_l$. Therefore

$$E\{A_1/X_1, \ldots, A_n/X_n\} \approx_l F\{A_1/X_1, \ldots, A_n/X_n\}$$

for all processes $A_1, \ldots, A_n$. Hence $E \approx_l F$ by definition. □

Notice that since the calculus does not have the mismatch operator, it follows from $E \Longrightarrow\approx_l F \wedge F \Longrightarrow\approx_l E$ and Lemma 3 that $E\sigma \Longrightarrow\approx_l F\sigma$ and $F\sigma \Longrightarrow\approx_l E\sigma$ for every substitution σ. Bisimulation Lemma is a universal property for the observational bisimulation equivalences. It holds of all the observational equivalences of this paper.

It is obvious from the above proof that we may assume that E and F are processes without loss of generality. In this and next sections we often make that assumption.

The proofs of the next two lemmas are typical examples of how to use the Bisimulation Lemma.

Lemma 12. *If $(x)(E \mid a.G) \approx_l (x)(F \mid a.G)$ for a fresh a then $(x)(E \mid G) \approx_l (x)(F \mid G)$.*

Proof. $(x)(P \mid a.R) \xrightarrow{a} (x)(P \mid R)$ must be matched up by

$$(x)(Q \mid a.R) \Longrightarrow (x)(Q_1 \mid a.R) \xrightarrow{a} (x)(Q_1 \mid R) \Longrightarrow Q' \approx_l (x)(P \mid R)$$

which can be rewritten as $(x)(Q \mid a.R) \xrightarrow{a} (x)(Q \mid R) \Longrightarrow Q' \approx_l (x)(P \mid R)$. Similarly one could show that $(x)(P \mid R) \Longrightarrow P' \approx_l (x)(Q \mid R)$ for some P'. So $(x)(P \mid R) \approx_l (x)(Q \mid R)$ by Bisimulation Lemma. □

Lemma 13. *$(\widetilde{y})(\overline{a}[H] \mid E) \approx_l (\widetilde{z})(\overline{a}[K] \mid F)$ for a fresh a if and only if $(\widetilde{y})(G[H] \mid E) \approx_l (\widetilde{z})(G[K] \mid F)$ for every process expression $G[Z]$.*

Proof. Implication in one direction is easy. Now suppose $(\widetilde{y})(\overline{a}[A] \mid P) \approx_l (\widetilde{z})(\overline{a}[B] \mid Q)$ for a fresh a. Then

$$(\widetilde{y})(\overline{a}[A] \mid P) \xrightarrow{(\widetilde{y_1})\overline{a}[A]} (\widetilde{y_2})P$$

where $\widetilde{y_1}\widetilde{y_2} = \widetilde{y}$, must be simulated by

$$\begin{array}{rcl} (\widetilde{z})(\overline{a}[B] \mid Q) & \Longrightarrow & (\widetilde{z})(\overline{a}[B] \mid Q_1) \\ & \xrightarrow{(\widetilde{z_1})\overline{a}[B]} & (\widetilde{z_2})Q_1 \\ & \Longrightarrow & (\widetilde{z_2})Q' \end{array}$$

where $\widetilde{z_1}\widetilde{z_2} = \widetilde{z}$, such that $(\widetilde{z_1})(G[B] \mid (\widetilde{z_2})Q') \approx_l (\widetilde{y_1})(G[A] \mid (\widetilde{y_2})P)$ for every process expression $G[Z]$. It follows that

$$(\widetilde{z})(G[B] \mid Q) \Longrightarrow \approx_l (\widetilde{y})(G[A] \mid P)$$

Similarly

$$(\widetilde{y})(G[A] \mid P) \Longrightarrow \approx_l (\widetilde{z})(G[B] \mid Q)$$

We may then conclude by Bisimulation Lemma that $(\widetilde{y})(G[A] \mid P) \approx_l (\widetilde{z})(G[B] \mid Q)$ for every process expression $G[Z]$. □

Using the above lemma, one could argue that (v') on page 11 is equivalent to the (v) of Definition 10 in the sense that they give rise to the same local bisimilarity. Let $E[X]$ be $\overline{b}[X]$ for a fresh b. The early simulation for the higher order output actions take the following form:

> If $P \xrightarrow{(\widetilde{x})\overline{a}A} P'$ then there exist some $\widetilde{x'}, B, Q', Q''$ such that $Q \Longrightarrow \xrightarrow{(\widetilde{x'})\overline{a}B} Q''$ and $(\widetilde{x'})(\overline{b}[B] \mid Q'') \Longrightarrow Q' \approx_l (\widetilde{x})(\overline{b}[A] \mid P')$.

Now Q' must be of the form $(\widetilde{x'})(\overline{b}[B] \mid Q_1)$. Therefore $Q \xRightarrow{(\widetilde{x'})\overline{a}B} Q_1$ and $(\widetilde{x'})(G[B] \mid Q_1) \approx_l (\widetilde{x})(G[A] \mid P')$ for every $G[X]$ by Lemma 13.

Bisimulation Lemma also underpins the proof that $\approx_l$ is an equivalence relation.

Proposition 14. *The local bisimilarity is an equivalence relation.*

Proof. Suppose that $P \approx_l Q \approx_l R$. We need to check the bismulation property for the first order bound output and the higher order output actions.

- $P \xrightarrow{\overline{b}(x)} P'$. By definition some Q_1, Q_2, Q' exist such that $Q \Longrightarrow Q_1 \xrightarrow{\overline{b}(x)} Q_2 \Longrightarrow Q'$ and the equivalence $(x)(P' \mid a.O) \approx_l (x)(Q' \mid a.O)$ for every O and every fresh a not in O. Now $R \Longrightarrow R_1 \approx_l Q_1$ for some R_1 and $R_1 \xRightarrow{\overline{b}(x)} R_2$ for some R_2 such that $(x)(Q_2 \mid a.O) \approx_l (x)(R_2 \mid a.O)$. The reduction $(x)(Q_2 \mid a.O) \Longrightarrow (x)(Q' \mid a.O)$ must be simulated by $(x)(R_2 \mid a.O) \Longrightarrow (x)(R' \mid a.O) \approx_l (x)(Q' \mid a.O)$ for some R'. Clearly $R_2 \Longrightarrow R'$. In summary $R \xRightarrow{\overline{b}(x)} R'$ and $(x)(P' \mid a.O) \approx_l (x)(R' \mid a.O)$. It follows from Lemma 12 that $(x)(P' \mid O) \approx_l (x)(R' \mid O)$.

- $P \xrightarrow{(\widetilde{x})\overline{b}A} P'$. By definition some $\widetilde{y}, B, Q_1, Q_2, Q'$ exist such that $Q \Longrightarrow Q_1 \xrightarrow{(\widetilde{y})\overline{b}B} Q_2 \Longrightarrow Q'$ and $(\widetilde{x})(\overline{a}[A] \mid P') \approx_l (\widetilde{y})(\overline{a}[B] \mid Q')$ for a fresh name a. Using similar argument one could show that some $\widetilde{z}, C, R'$ exist such that $R \xRightarrow{(\widetilde{z})\overline{b}C} R'$ and $(\widetilde{x})(\overline{a}[A] \mid P') \approx_l (\widetilde{y})(\overline{a}[C] \mid R')$. It follows from Lemma 13 that $(\widetilde{x})(E[A] \mid P') \approx_l (\widetilde{y})(E[C] \mid R')$ for every $E[Z]$.

we are done. □

The well-definedness of the local bisimilarity should also be judged by the closure properties it satisfies. The next two lemmas show that $\approx_l$ has the expected closure properties.

Lemma 15. *Suppose $E \approx_l F$. Then the following equalities hold:*
(i) $a(x).E \approx_l a(x).F$, $\overline{a}x.E \approx_l \overline{a}x.F$;
(ii) $E \mid G \approx_l F \mid G$, $G \mid E \approx_l G \mid F$, $(x)E \approx_l (x)F$ and $[x{=}y]E \approx_l [x{=}y]F$;
(iii) $a(X).E \approx_l a(X).F$;
(iv) $\overline{a}[H].E \approx_l \overline{a}[H].F$.

Proof. Construct a series of relations as follows:

$$\begin{array}{rcl} \mathcal{S}_0 & \stackrel{\text{def}}{=} & \approx_l \\ & \vdots & \\ \mathcal{S}_{i+1} & \stackrel{\text{def}}{=} & \left\{ \begin{array}{r|l} (a(z).E, a(z).F) & \\ (\overline{a}z.E, \overline{a}z.F) & \\ (\overline{a}[H].E, \overline{a}[H].F) & \\ (a(X).E, a(X).F) & E \, \mathcal{S}_i \, F \\ (E|G, F|G) & \\ (G|E, G|F) & \\ ((z)E, (z)F) & \\ ([y{=}z]E, [x{=}y]F) & \end{array} \right\} \\ & \vdots & \end{array}$$

It is clear that all the $\mathcal{S}_i$'s are closed under substitutions of names and variables. Let $\mathcal{S}$ be $\bigcup_{i\in\omega} \mathcal{S}_i$. We prove by structural induction that $\mathcal{S}$ is a local bisimulation up to $\sim$. We only have to consider process expressions without free variables. Now suppose that all pairs in $\mathcal{S}_0 \cup \ldots \cup \mathcal{S}_i$ satisfy the bisimulation property of Definition 9, meaning that whenever $(P, Q) \in \mathcal{S}_0 \cup \ldots \cup \mathcal{S}_i$ then the following properties hold:

1. If $P \xrightarrow{\tau} P'$ then Q' exists such that $Q \Longrightarrow Q' \sim \mathcal{S} \sim P'$.
2. If $P \xrightarrow{\alpha x} P'$ then Q' exists such that $Q \stackrel{\alpha x}{\Longrightarrow} Q' \sim \mathcal{S} \sim P'$.
3. If $P \xrightarrow{\overline{a}(x)} P'$ then Q' exists such that $Q \stackrel{\overline{a}(x)}{\Longrightarrow} Q'$ and $(x)(P' \mid O) \sim \mathcal{S} \sim (x)(Q' \mid O)$ for all O.
4. If $P \xrightarrow{aA} P'$ then Q' exists such that $Q \stackrel{aA}{\Longrightarrow} Q' \sim \mathcal{S} \sim P'$.
5. If $P \xrightarrow{(\widetilde{x})\overline{a}A} P'$ then some $\widetilde{x'}, B, Q'$ exist such that $Q \stackrel{(\widetilde{x'})\overline{a}B}{\Longrightarrow} Q'$ and $(\widetilde{x})(G[A] \mid P') \sim \mathcal{S} \sim (\widetilde{x'})(G[B] \mid Q')$ for every context $G[Y]$ such that $\widetilde{x}\widetilde{x'} \cap fn(G[Y]) = \emptyset$.

We now check that all pairs in $\mathcal{S}_{i+1}$ satisfy the bisimulation properties as well:

- $a(z).P \; \mathcal{S}_{i+1} \; a(z).Q$. Clearly $a(z).P \xrightarrow{ax} P\{x/z\}$ is matched up by $a(z).Q \xrightarrow{ax} Q\{x/z\}\mathcal{S}_i P\{x/z\}$.
- $\overline{a}z.P \; \mathcal{S}_{i+1} \; \overline{a}z.Q$. Obviously $\overline{a}z.P \xrightarrow{\overline{a}z} P$ is matched up by $\overline{a}z.Q \xrightarrow{\overline{a}z} Q\mathcal{S}_i P$.
- $a(X).E \; \mathcal{S}_{i+1} \; a(X).F$. Apparently $a(X).E \xrightarrow{aA} E[A]$ is matched up by $a(X).F \xrightarrow{aA} F[A]$ since $E[A]\mathcal{S}_i F[A]$ for each A.
- $\overline{a}A.P \; \mathcal{S}_{i+1} \; \overline{a}A.Q$. Now $\overline{a}A.P \xrightarrow{\overline{a}A} P$ is matched up by $\overline{a}A.Q \xrightarrow{\overline{a}A} Q$ since $(E[A] \mid P)\mathcal{S}_{i+1}(E[A] \mid Q)$ for each E.
- $P|R \; \mathcal{S}_{i+1} \; Q|R$.
  1. If $P \xrightarrow{\tau} P'$ then $Q \mid R \Longrightarrow Q' \mid R \sim \mathcal{S} \sim P' \mid R$.
  2. $P \xrightarrow{\alpha x} P'$. There are two cases:
     - $P \mid R \xrightarrow{\alpha x} P' \mid R$ is matched up by $Q \mid R \stackrel{\alpha x}{\Longrightarrow} Q' \mid R \sim \mathcal{S} \sim P' \mid R$.
     - $R \xrightarrow{\overline{\alpha} x} R'$. $P \mid R \xrightarrow{\tau} P' \mid R'$ is matched up by $Q \mid R \stackrel{\tau}{\Longrightarrow} Q' \mid R' \sim \mathcal{S} \sim P' \mid R'$.
  3. If $P \xrightarrow{\overline{a}(x)} P'$. There are two cases:

– $P \mid R \xrightarrow{\overline{a}(x)} P' \mid R$ is simulated by $Q \mid R \stackrel{\overline{a}(x)}{\Longrightarrow} Q' \mid R$ and $(x)(P' \mid R \mid O) \sim \mathcal{S} \sim (x)(Q' \mid R \mid O)$ for all O.
– $R \xrightarrow{ax} R'$. $P \mid R \xrightarrow{\tau} (x)(P' \mid R')$ is simulated by $Q \mid R \stackrel{\tau}{\Longrightarrow} (x)(Q' \mid R)$ and $(x)(P' \mid R') \sim \mathcal{S} \sim (x)(Q' \mid R')$.

4. $P \xrightarrow{aA} P'$. There are two cases:
– $P \mid R \xrightarrow{aA} P' \mid R$ is simulated by $Q \mid R \stackrel{aA}{\Longrightarrow} Q' \mid R \sim \mathcal{S} \sim P' \mid R$.
– $R \xrightarrow{(\widetilde{y})\overline{a}A} R'$. $P \mid R \xrightarrow{\tau} (\widetilde{y})(P' \mid R')$ is simulated by $Q \mid R \stackrel{\tau}{\Longrightarrow} (\widetilde{y})(Q' \mid R') \sim \mathcal{S} \sim (\widetilde{y})(P' \mid R')$.

5. $P \xrightarrow{(\widetilde{x})\overline{a}A} P'$. There are two cases:
– $P \mid R \xrightarrow{(\widetilde{x})\overline{a}A} P' \mid R$ is simulated by $Q \mid R \stackrel{(\widetilde{x'})\overline{a}B}{\Longrightarrow} Q' \mid R$ and
$$(\widetilde{x})(G[A] \mid R \mid P') \sim \mathcal{S} \sim (\widetilde{x'})(G[B] \mid R \mid Q')$$
for every context $G[Y]$ such that $\widetilde{x}\widetilde{x'} \cap fn(G[Y]) = \emptyset$.
– $R \xrightarrow{aA} R'$. $P \mid R \xrightarrow{\tau} (\widetilde{x})(P' \mid R')$ is simulated by $Q \mid R \stackrel{\tau}{\Longrightarrow} (\widetilde{x})(Q' \mid R')$ and
$$(\widetilde{x})(G[A] \mid R') \sim \mathcal{S} \sim (\widetilde{x'})(G[B] \mid R')$$
for every context $G[Y]$ such that $\widetilde{x}\widetilde{x'} \cap fn(G[Y]) = \emptyset$.

- $P|R\ \mathcal{S}_{i+1}\ Q|R$. Symmetric to the previous case.
- $(y)P\ \mathcal{S}_{i+1}\ (y)Q$.

1. $P \xrightarrow{\tau} P'$. $(y)P \xrightarrow{\tau} (y)P'$ is simulated by $(y)Q \Longrightarrow (y)Q' \sim \mathcal{S} \sim (y)P'$.
2. $P \xrightarrow{\alpha x} P'$.
– $(y)P \xrightarrow{\alpha x} (y)P'$ is simulated by $(y)Q \stackrel{\alpha x}{\Longrightarrow} (y)Q' \sim \mathcal{S} \sim (y)P'$.
– $(x)P \xrightarrow{\overline{a}(x)} P'$ is simulated by $(x)Q \stackrel{\overline{a}(x)}{\Longrightarrow} Q'$ and $(x)(Q' \mid O) \sim \mathcal{S} \sim (x)(P' \mid O)$ for all O.
3. $P \xrightarrow{\overline{a}(x)} P'$. $(y)P \xrightarrow{\overline{a}(x)} (y)P'$ is simulated by $(y)Q \stackrel{\overline{a}(x)}{\Longrightarrow} (y)Q'$ and $(y)(x)(P' \mid O) \sim \mathcal{S} \sim (y)(x)(Q' \mid O)$ for all O.
4. $P \xrightarrow{aA} P'$. $(x)P \xrightarrow{aA} (x)P'$ is simulated by $(x)Q \stackrel{aA}{\Longrightarrow} (x)Q' \sim \mathcal{S} \sim (x)P'$.
5. $P \xrightarrow{(\widetilde{x})\overline{a}A} P'$. $(y)P \xrightarrow{(y)(\widetilde{x})\overline{a}A} P'$ or $(y)P \xrightarrow{(\widetilde{x})\overline{a}A} (y)P'$ is simulated by $(y)Q \stackrel{(y)(\widetilde{x'})\overline{a}B}{\Longrightarrow} Q'$ or $(y)Q \stackrel{(\widetilde{x'})\overline{a}B}{\Longrightarrow} (y)Q'$ such that $(y)(\widetilde{x})(G[A] \mid P') \sim \mathcal{S} \sim (y)(\widetilde{x'})(G[B] \mid Q')$ for every context $G[Y]$ such that $\widetilde{x}\widetilde{x'} \cap fn(G[Y]) = \emptyset$.

- $[x{=}y]P\ \mathcal{S}_{i+1}\ [x{=}y]Q$. If $x = y$ then the induction is obvious. Otherwise there is nothing to prove.

So $\mathcal{S}$ is a local bisimulation up to $\sim$. It follows that $\mathcal{S} \subseteq \approx_l$. □

The above proof is basically a structural induction. This is possible only if the **fix**-operator is excluded.

The local bisimilarity is not closed under summation for well-known reason. For $LHO\pi$, the higher order output prefix also causes problems. Notice that $P \approx_l Q$ does not imply $\overline{a}[P].A \approx_l \overline{a}[Q].A$. As a matter of fact $\overline{a}[P] \not\approx_l \overline{a}[\tau.P]$ in general.

Theorem 16. *$E \approx_o F$ if and only if $E \approx_l F$.*

Proof. By Lemma 15, $\approx_l$ is an observed bisimulation. Hence $\approx_l \subseteq \approx_o$. The inclusion $\approx_o \subseteq \approx_l$ amounts to showing that $\approx_o$ is a local bisimulation, which is a routine exercise. Again the Bisimulation Lemma is necessary to tidy up the argument. □

Before ending this section we state a technical lemma to be used later on.

Lemma 17. *Suppose a, b, c are distinct names not free in $E \mid F$. Then the following properties hold:*
(i) $(x)(y)(\overline{a}x \mid \overline{b}y \mid E) \not\approx_l (y)(\overline{a}y \mid \overline{b}y \mid F)$;
(ii) $(x)(y)(\overline{a}x \mid \overline{b}y \mid \overline{c}y \mid E) \not\approx_l (x)(y)(\overline{a}x \mid \overline{b}x \mid \overline{c}y \mid F)$;
(iii) If $(x)(\overline{a}x \mid \overline{b}x \mid E) \approx_l (x)(\overline{a}x \mid \overline{b}x \mid F)$ then $(x)(\overline{a}x \mid E) \approx_l (x)(\overline{a}x \mid F)$.

It follows from (i) of Lemma 17 that $(y)(\overline{a}x \mid \overline{b}y \mid E) \not\approx_l (y)(\overline{a}y \mid \overline{b}y \mid F)$. Lemma 17 will be used implicitly when we assume that the a derivative of an action must take a certain shape.

4 Prefix and Continuation

This section studies the relationship between the equivalence of the prefixed processes and the equivalence of the continuations. For instance two such relationships are the followings:

Input Prefix: $a(x).E \approx_l a(x).F$ if and only if $E \approx_l F$.
Output Prefix: $\overline{a}x.E \approx_l \overline{a}x.F$ if and only if $E \approx_l F$.

The general question is this: What can be said about the equivalence between E and F if $\lambda.E$ and $\lambda.F$ are equivalent? This is about the reverse of the congruence property. For the other forms of the prefix operators, the characterizations are not as simple as the above ones. More complexity would be added if things are considered in local contexts.

Proposition 18. *Suppose $\widetilde{c} = c_1, \ldots, c_n$ are pairwise distinct fresh names, $\widetilde{z} = z_1, \ldots, z_n$ are pairwise distinct, $a \notin \widetilde{c}$, and $x \notin \widetilde{z}$. Then the following properties hold:*
(i) $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, a(x).E) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, a(x).F)$ if and only if $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, E) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, F)$;
(ii) $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, \overline{a}y.E) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, \overline{a}y.F)$ if and only if $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, E) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, F)$.

Proof. (i) There are two cases:

- $a \notin \widetilde{z}$. One implication is obvious. Now suppose $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, a(x).P) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, a(x).Q)$. It is clear that $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, a(x).P) \xrightarrow{ax} (\widetilde{z})(\widetilde{\overline{c}z} \,|\, P)$ is simulated by $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, a(x).Q) \xrightarrow{ax} (\widetilde{z})(\widetilde{\overline{c}z} \,|\, Q) \Longrightarrow Q' \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, P)$. Similarly $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, P) \Longrightarrow\approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, Q)$. Hence $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, P) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, Q)$ by Bisimulation Lemma.
- $a = z_i \in \widetilde{z}$. Now assume that $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, a(x).P) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, a(x).Q)$. Let d, e be fresh names. Then

$$\begin{array}{ll} & (e{+}c_i(z).(d{+}\overline{z}x.\overline{c_i}z)) \,|\, (\widetilde{z})(\widetilde{\overline{c}z} \,|\, z_i(x).P) \\ \xrightarrow{\tau}\sim & (\widetilde{z})((d{+}\overline{z_i}x.\overline{c_i}z_i) \,|\, (\overline{c_1}z_1 \,|\, \ldots \,|\, \overline{c_{i-1}}z_{i-1} \,|\, \overline{c_{i+1}}z_{i+1} \,|\, \ldots \,|\, \overline{c_n}z_n \,|\, z_i(x).P)) \\ \xrightarrow{\tau}\sim & (\widetilde{z})(\widetilde{\overline{c}z} \,|\, P) \end{array}$$

must be simulated by

$$\begin{array}{ll} & (e{+}c_i(z).(d{+}\overline{z}x.\overline{c_i}z)) \,|\, (\widetilde{z})(\widetilde{\overline{c}z} \,|\, z_i(x).Q) \\ \xrightarrow{\tau}\sim & (\widetilde{z})((d{+}\overline{z_i}x.\overline{c_i}z_i) \,|\, (\overline{c_1}z_1 \,|\, \ldots \,|\, \overline{c_{i-1}}z_{i-1} \,|\, \overline{c_{i+1}}z_{i+1} \,|\, \ldots \,|\, \overline{c_n}z_n \,|\, z_i(x).Q)) \\ \xrightarrow{\tau}\sim & (\widetilde{z})(\widetilde{\overline{c}z} \,|\, Q) \\ \Longrightarrow & (\widetilde{z})(\widetilde{\overline{c}z} \,|\, Q') \\ \approx_l & (\widetilde{z})(\widetilde{\overline{c}z} \,|\, P) \end{array}$$

We are done by resorting to Bisimulation Lemma.

(ii) We show that $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, \overline{a}y.P) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, \overline{a}y.Q)$ implies $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, P) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, Q)$. The other direction is simple. We confine ourselves to the situation where $a = z_i$. There are two major cases:

- $a = y = z_i \in \widetilde{x}$. Let d, e be fresh. Then

$$\begin{array}{ll} & (e{+}c_i(z).(d{+}z(x).\overline{c_i}x)) \,|\, (\widetilde{z})(\widetilde{\overline{c}z} \,|\, \overline{z_i}z_i.P) \\ \xrightarrow{\tau}\sim & (\widetilde{z})((d{+}\overline{z_i}(x).\overline{c_i}x) \,|\, (\overline{c_1}z_1 \,|\, \ldots \,|\, \overline{c_{i-1}}z_{i-1} \,|\, \overline{c_{i+1}}z_{i+1} \,|\, \ldots \,|\, \overline{c_n}z_n \,|\, \overline{z_i}z_i.P)) \\ \xrightarrow{\tau}\sim & (\widetilde{z})(\widetilde{\overline{c}z} \,|\, P) \end{array}$$

must be simulated by

$$\begin{array}{ll} & (e{+}c_i(z).(d{+}z(x).\overline{c_i}x)) \,|\, (\widetilde{z})(\widetilde{\overline{c}z} \,|\, \overline{z_i}z_i.Q) \\ \xrightarrow{\tau}\sim & (\widetilde{z})((d{+}\overline{z_i}(x).\overline{c_i}x) \,|\, (\overline{c_1}z_1 \,|\, \ldots \,|\, \overline{c_{i-1}}z_{i-1} \,|\, \overline{c_{i+1}}z_{i+1} \,|\, \ldots \,|\, \overline{c_n}z_n \,|\, \overline{z_i}z_i.Q)) \\ \xrightarrow{\tau}\sim & (\widetilde{z})(\widetilde{\overline{c}z} \,|\, Q) \\ \Longrightarrow & (\widetilde{z})(\widetilde{\overline{c}z} \,|\, Q') \\ \approx & (\widetilde{z})(\widetilde{\overline{c}z} \,|\, P) \end{array}$$

Now apply the Bisimulation Lemma.

- $\widetilde{z} \ni z_i = a \neq y = z_j \in \widetilde{z}$. Let $d, e, c_{j'}$ be fresh. Then

$$\begin{array}{rl} & (e{+}c_i(z).(d{+}z(x).(\overline{c_i}z \,|\, c'_j x))) \,|\, (\widetilde{z})(\widetilde{\overline{c}z} \,|\, \overline{z_i} z_j.P) \\ \xrightarrow{\tau}\sim & (\widetilde{z})((d{+}z_i(x).(\overline{c_i}z_i \,|\, c'_j x)) \,|\, (\overline{c_1}z_1 \,|\, \ldots \,|\, \overline{c_{i-1}}z_{i-1} \,|\, \overline{c_{i+1}}z_{i+1} \,|\, \ldots \,|\, \overline{c_n}z_n \,|\, \overline{z_i}z_j.P)) \\ \xrightarrow{\tau}\sim & (\widetilde{z})(\overline{c'_j}z_j \,|\, \widetilde{\overline{c}z} \,|\, P) \end{array}$$

must be simulated in the following manner

$$\begin{array}{rl} & (e{+}c_i(z).(d{+}z(x).(\overline{c_i}z \,|\, c'_j x))) \,|\, (\widetilde{z})(\widetilde{\overline{c}z} \,|\, \overline{z_i} z_j.Q) \\ \xrightarrow{\tau}\sim & (\widetilde{z})((d{+}z_i(x).(\overline{c_i}z_i \,|\, c'_j x)) \,|\, (\overline{c_1}z_1 \,|\, \ldots \,|\, \overline{c_{i-1}}z_{i-1} \,|\, \overline{c_{i+1}}z_{i+1} \,|\, \ldots \,|\, \overline{c_n}z_n \,|\, \overline{z_i}z_j.Q)) \\ \xrightarrow{\tau}\sim & (\widetilde{z})(\overline{c'_j}z_j \,|\, \widetilde{\overline{c}z} \,|\, Q) \\ \Longrightarrow & (\widetilde{z})(\overline{c'_j}z_j \,|\, \widetilde{\overline{c}z} \,|\, Q') \\ \approx & (\widetilde{z})(\overline{c'_j}z_j \,|\, \widetilde{\overline{c}z} \,|\, P) \end{array}$$

It follows from Lemma 17 (iii) that $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, Q') \approx (\widetilde{z})(\widetilde{\overline{c}z} \,|\, P)$. The rest of the proof is routine.

We are done. □

In the rest of the section we investigate similar properties for the first order bound output prefix and the higher order prefixes. These characterizations play a key role in the algebraic theory. The proof of Proposition 18 shows how to deal with the local contexts. The proofs in the rest of this section will ignore the local contexts.

4.1 Localization Theorem

It is important to bear in mind that for the first order bound output prefixes, it is not valid that $\overline{a}(x).P \approx_l \overline{a}(x).Q$ if and only if $P \approx_l Q$. The equality $P \approx_l Q$ is far more strong than $\overline{a}(x).P \approx_l \overline{a}(x).Q$. However the relationship between the bound output and the continuations is still simple.

Theorem 19 (Localization)**.** *Suppose $\widetilde{c} = c_1, \ldots, c_n$ are pairwise distinct fresh names and $\widetilde{z} = z_1, \ldots, z_n$ are pairwise distinct. The following statements are equivalent:*
(i) $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, \overline{b}(x).E) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, \overline{b}(x).F)$;
(ii) $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, (x)(\overline{a}x \,|\, E)) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, (x)(\overline{a}x \,|\, F))$ for some fresh a;
(iii) $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, (x)(G \,|\, E)) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, (x)(G \,|\, F))$ for every process expression G.

Proof. (i⇒ii): The action $\overline{b}(x).P \xrightarrow{\overline{b}(x)} P'$ is simulated by $\overline{b}(x).Q \xrightarrow{\overline{b}(x)}\!\!\Longrightarrow Q'$ such that $(x)(\overline{a}x \,|\, P) \approx_l (x)(\overline{a}x \,|\, Q')$ for some fresh a. Hence $(x)(\overline{a}x \,|\, Q) \Longrightarrow\!\!\approx_l (x)(\overline{a}x \,|\, P)$. Similarly $(x)(\overline{a}x \,|\, P) \Longrightarrow\!\!\approx_l (x)(\overline{a}x \,|\, Q)$. Conclude by Bisimulation Lemma that $(x)(\overline{a}x \,|\, P) \approx_l (x)(\overline{a}x \,|\, Q)$.

(ii⇒iii): Suppose $(x)(\overline{a}x \,|\, P) \approx_l (x)(\overline{a}x \,|\, Q)$ for some fresh a. Then $a(x).G \,|\, (x)(\overline{a}x \,|\, P) \xrightarrow{\tau} (x)(G \,|\, P)$. The simulation must be $a(x).G \,|\, (x)(\overline{a}x \,|\, Q) \Longrightarrow a(x).G \,|\, (x)(\overline{a}x \,|\, Q') \xrightarrow{\tau} (x)(G \,|\, Q') \Longrightarrow Q''$. It follows that $(x)(G \,|\, Q) \Longrightarrow\!\!\approx_l (x)(G \,|\, P)$. Symmetrically $(x)(G \,|\, P) \Longrightarrow\!\!\approx_l (x)(G \,|\, Q)$. Hence $(x)(G \,|\, P) \approx_l (x)(G \,|\, Q)$ by Bisimulation Lemma.

(iii⇒i): This follows immediately from definition. □

The Localization Theorem implies that (iii') stated on page 11 is equivalent to the (iii) of Definition 10.

4.2 Abstraction Theorem

Intuitively $a(X).E \approx a(X).F$ if and only if $E\{A/X\} \approx F\{A/X\}$ for all process A. It would be nice if we could remove the universal quantification. We are looking for a particular process U such that $a(X).E \approx a(X).F$ if and only if $E\{U/X\} \approx F\{U/X\}$. There are many choices for U. We take a look at two of them. Let I_a abbreviate $a(X).X$ and a abbreviate $a(x).\mathbf{0}$. Sangiorgi has studied the issue for $HOCCS$. He proved that $a(X).E$ is delayed bisimilar to $a(X).F$ if and only if $E\{a/X\}$ is delayed bisimilar to $F\{a/X\}$. For our purpose his proof has to be modified in two aspects. First he worked in a framework without the choice operator, so his result follows immediately from the Factorization Theorem, which in our case does not hold. Secondly Sangiorgi's results were established for the delayed bisimulation equivalence. The proofs need be extended to the more general bisimulations.

Lemma 20. *If A participates in $E[A] \xrightarrow{\lambda} E'[A']$ then $E[I_a] \xrightarrow{aA} E''[A] \xrightarrow{\lambda} E'[A']$ for a fresh a and for some E''.*

Proof. This is a simple exercise using induction on the height of derivation. Notice that a communication between E and A must be through a global name since E should neither bind nor localize any free names in A. □

Theorem 21 (Abstraction)**.** *Suppose $\widetilde{c} = c_1, \ldots, c_n$ are pairwise distinct fresh names and $\widetilde{z} = z_1, \ldots, z_n$ are pairwise distinct. The following statements are equivalent:*
(i) $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, b(X).E) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, b(X).F)$ for some name b;
(ii) $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, E\{I_a/X\}) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, F\{I_a/X\})$ for a fresh name a;
(ii') $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, E\{a/X\}) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, F\{a/X\})$ for a fresh name a;
(iii) $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, E\{G/X\}) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, F\{G/X\})$ for every process expression G.

Proof. Without loss of generality, we may assume that E, F contain at most the process variable X.

(i) ⇔ (ii): Only one implication is nontrivial. Let $\mathcal{R}$ be the following symmetric relation:

$$\left\{ (E\{A/X\}, F\{A/X\}) \;\middle|\; \begin{array}{l} E\{I_a/X\} \approx_l F\{I_a/X\} \\ a \text{ is fresh, } A \text{ a process} \end{array} \right\} \cup \approx_l$$

Suppose that $E\{A/X\} \mathcal{R} F\{A/X\}$ and that σ is a substitution. By definition

$$E\{I_a/X\} \approx_l F\{I_a/X\}$$

Let b be a fresh name that does not appear in σ. Since $\approx_l$ is closed under substitution of names, one must have $E\{I_b/X\} \approx_l F\{I_b/X\}$. For the same reason one has that

$$E\sigma\{I_b/X\} \approx_l F\sigma\{I_b/X\}$$

Therefore $E\sigma\{A\sigma/X\} \;\mathcal{R}\; F\sigma\{A\sigma/X\}$. We conclude that $\mathcal{R}$ is closed under substitution of names.

Now we show that $\mathcal{R}$ is a local bisimulation up to $\sim$. Suppose that $E\{A/X\} \xrightarrow{\lambda} P$. There are four major cases for the action. We discuss them in turn:

- The action is caused by a copy of A, say $A \xrightarrow{\lambda} A'$. By the fact that substitutions avoid name capture, one has that
$$E\{A/X\} \xrightarrow{\lambda} E'\{A'/X\} \equiv P$$
for some E'. Then clearly
$$E\{I_a/X\} \,|\, \overline{g} \xrightarrow{a[g.I_b]} E'\{g.I_b/X\} \,|\, \overline{g}$$
for fresh b, g. It follows from $E\{I_a/X\} \approx_l F\{I_a/X\}$ that some F' exists such that
$$\begin{array}{rcl} F\{I_a/X\} \,|\, \overline{g} & \stackrel{a[g.I_b]}{\Longrightarrow} & F'\{g.I_b/X\} \,|\, \overline{g} \\ & \approx_l & E'\{g.I_b/X\} \,|\, \overline{g} \end{array}$$
Using the Bisimulation Lemma, one could easily show that $F'\{I_b/X\} \approx_l E'\{I_b/X\}$. It follows that $F\{A/X\} \stackrel{\lambda}{\Longrightarrow} F'\{A'/X\} \mathcal{R} E'\{A'/X\}$.

- The action is caused by a prefix in E, that is
$$E\{A/X\} \xrightarrow{\lambda} E'\{A/X\} \equiv P$$
for some E'. Thus $E\{I_a/X\} \xrightarrow{\lambda} E'\{I_a/X\}$. By assumption there must exist some F' such that
$$F\{I_a/X\} \stackrel{\widehat{\lambda}}{\Longrightarrow} F'\{I_a/X\}$$
Therefore $E\{A/X\} \xrightarrow{\lambda} E'\{A/X\}$ is simulated by $F\{A/X\} \stackrel{\widehat{\lambda}}{\Longrightarrow} F'\{A/X\}$.

- Suppose the action $E\{A/X\} \xrightarrow{\tau} E'\{A'/X\}$ is caused by a communication between E and A. There are six cases:

– $E \xrightarrow{cz} E'$ and $A \xrightarrow{\overline{c}(z)} A'$. Let C be $\overline{a}[\overline{c}(z).h.h'.A'+g].\overline{h}+f$ where f,g,h,h' are fresh. Then by Lemma 20 some $E''[X]$ exists such that

$$\begin{array}{rcl} E\{I_a/X\}\,|\,C & \xrightarrow{\tau} & E''\{\overline{c}(z).h.h'.A'+g/X\}\,|\,\overline{h} \\ & \xrightarrow{\tau}\sim & (z)E'\{(h.h'.A'\,|\,h)/X\}\,|\,\overline{h} \\ & \xrightarrow{\tau} & (z)E'\{h'.A'/X\} \end{array}$$

It follows from $E\{I_a/X\} \approx_l F\{I_a/X\}$ that F_1, F_2, F' exist such that

$$\begin{array}{rcl} F\{I_a/X\}\,|\,C & \stackrel{\tau}{\Longrightarrow} & F_2\{\overline{c}(z).h.h'.A'+g/X\}\,|\,\overline{h} \\ & \stackrel{\tau}{\Longrightarrow} & F_1\{h.h'.A'/X\}\,|\,\overline{h} \\ & \stackrel{\tau}{\Longrightarrow} & F'\{h'.A'/X\} \\ & \approx_l & (z)E'\{h'.A'/X\} \end{array}$$

It follows easily from $F'\{h'.A'/X\} \approx_l (z)E'\{h'.A'/X\}$ and Bisimulation Lemma that $F'\{A'/X\} \approx_l (z)E'\{A'/X\}$. Consequently $F\{A/X\} \stackrel{\tau}{\Longrightarrow} F'\{A'/X\}\ \mathcal{R}\ (z)E'\{A'/X\}$.

– $E \xrightarrow{cz} E'$ and $A \xrightarrow{\overline{c}z} A'$. This case is simpler than the previous one.

– $E \xrightarrow{\overline{c}(z)} E'$ and $A \xrightarrow{cz} A'$. Let C be $\overline{a}[c(z).f.(\sum_{v\in V}[z{=}v]b_v{+}g.A')].\overline{f}$ where V is the set of free names in $E\,|\,F$ and b_v, f, g, for every $v \in V$, are fresh names. Again by Lemma 20 some $E''[X]$ exists such that

$$\begin{array}{rcl} E\{I_a/X\}\,|\,C & \xrightarrow{\tau} & E''\{c(z).f.(\sum\limits_{v\in V}[z{=}v]b_v{+}g.A')/X\}\,|\,\overline{f} \\ & \xrightarrow{\tau}\sim & (z)E'\{f.(\sum\limits_{v\in V}[z{=}v]b_v{+}g.A')/X\}\,|\,\overline{f} \\ & \xrightarrow{\tau} & (z)E'\{\sum\limits_{v\in V}[z{=}v]b_v{+}g.A'/X\} \end{array}$$

It follows from $E\{I_a/X\} \approx_l F\{I_a/X\}$ that F_1, F_2, F' exist such that

$$\begin{array}{rcl} F\{I_a/X\}\,|\,C & \stackrel{\tau}{\Longrightarrow} & F_2\{c(z).f.(\sum\limits_{v\in V}[z{=}v]b_v{+}g.A')/X\}\,|\,\overline{f} \\ & \stackrel{\tau}{\Longrightarrow} & F_1\{f.(\sum\limits_{v\in V}[z{=}v]b_v{+}g.A')/X\}\,|\,\overline{f} \\ & \Longrightarrow & F'\{\sum\limits_{v\in V}[z{=}v]b_v{+}g.A'/X\} \\ & \approx_l & (z)E'\{\sum\limits_{v\in V}[z{=}v]b_v{+}g.A'/X\} \end{array}$$

Now $F'\{A'/X\} \approx_l (z)E'\{A'/X\}$ follows from

$$F'\{\sum_{v\in V}[z{=}v]b_v{+}g.A'/X\} \approx_l (z)E'\{\sum_{v\in V}[z{=}v]b_v{+}g.A'/X\}$$

by Bisimulation Lemma. Consequently $F\{A/X\} \stackrel{\tau}{\Longrightarrow} F'\ \mathcal{R}\ (z)E'\{A'/X\}$.

– $E \xrightarrow{\overline{c}z} E'$ and $A \xrightarrow{cz} A'$. This case is simpler than the previous one.

– $E \xrightarrow{(\widetilde{z})\overline{c}R} E'$ and $A \xrightarrow{cR} G[R]$. Let C be $\overline{a}[c(X).g.h.G[X]].\overline{g}+f$ for fresh f,g,h. By Lemma 20 some $E''[X]$ exists such that

$$\begin{array}{rcl} E\{I_a/X\}\,|\,C & \xrightarrow{\tau} & E''\{c(X).g.h.G[X]/X\}\,|\,\overline{g} \\ & \xrightarrow{\tau}\sim & (\widetilde{z})E'\{g.h.G[R]/X\}\,|\,\overline{g} \\ & \xrightarrow{\tau} & (\widetilde{z})E'\{h.G[R]/X\} \end{array}$$

and some F_1, F_2, F' exist such that the above actions must be matched up by

$$\begin{array}{rcl} F\{I_a/X\} \,|\, C & \stackrel{\tau}{\Longrightarrow} & F_2\{c(X).g.h.G[X]/X\} \,|\, \overline{g} \\ & \stackrel{\tau}{\Longrightarrow} & (\widetilde{z})F_1\{g.h.G[R]/X\} \,|\, \overline{g} \\ & \stackrel{\tau}{\longrightarrow} & (\widetilde{z})F'\{h.G[R]/X\} \\ & \approx_l & (\widetilde{z})E'\{h.G[R]/X\} \end{array}$$

Again $(\widetilde{z})F'\{G[R]/X\} \approx_l (\widetilde{z})E'\{G[R]/X\}$ follows from $(\widetilde{z})F'\{h.G[R]/X\} \approx_l (\widetilde{z})E'\{h.G[R]/X\}$ by Bisimulation Lemma. It is then clear that $F\{A/X\} \stackrel{\tau}{\Longrightarrow} (\widetilde{z})F'\{G[R]/X\}\ \mathcal{R}\ (\widetilde{z})E'\{G[R]/X\}$.

– $E \stackrel{cR}{\longrightarrow} E'$ and $A \stackrel{(\widetilde{z})\overline{c}R}{\longrightarrow} A'$. Let C be $\overline{a}[(\widetilde{z})\overline{c}R.g.h.A'].\overline{g} + f$ for fresh f, g, h. The proof is similar to the previous case.

We are done.

(i) $\Leftrightarrow$ (ii'): The implication (i) $\Rightarrow$ (ii') is obvious. The proof of the converse implication is similar to the proof of (ii) $\Rightarrow$ (i). Notice that $(a)(E\{a/X\} \,|\, \overline{a}.A)) \sim (a)(E\{I_a/X\} \,|\, \overline{a}[A])$ for a fresh name a.

(i) $\Leftrightarrow$ (iii): This is straightforward by Bisimulation Lemma. □

4.3 Concretion Theorem

Higher order output prefix in $LHO\pi$ is more subtle than the higher order input prefix. When $(\widetilde{y})\overline{a}[A].P$ emits the process A through a, A must be evaluated in all possible environments while keeping in touch with P through $\widetilde{y}$. In the receiving environment A may or may not be used. Half way through computation, it could be exported to another alien environment to be further evaluated. The situation could be very complicated. A key observation that would lead to a useful characterization is that an exported process can not be fired unless it is in a position able to perform an external action. So instead of sending out a process A one could send off a trigger that can kick off A when the trigger is in a fire-able position. This idea is due to Thomsen and Sangiorgi. Again we need to bypass the Factorization Theorem in our framework.

First we prove two technical lemmas.

Lemma 22. *Suppose a, b are fresh names. The following properties hold:*
(i) If A contributes in the action $(\widetilde{y})E[A] \stackrel{\lambda}{\longrightarrow} P$, then $(\widetilde{y})(E[a] \,|\, \overline{a}.(A{+}b)) \stackrel{\lambda}{\Longrightarrow} P' \sim P$ for some P'.
(ii) If $(\widetilde{y})(E[a] \,|\, \overline{a}.(A{+}b)) \stackrel{\lambda}{\Longrightarrow} P'$ such that P' contains neither a nor b, then $(\widetilde{y})E[A] \stackrel{\lambda}{\Longrightarrow} P \sim P'$ for some P.

Proof. (i) First we prove that

If A contributes in the action $E[A] \stackrel{\lambda}{\longrightarrow} P$, then $E[a] \,|\, \overline{a}.(A{+}b) \stackrel{\lambda}{\Longrightarrow} P' \sim P$ for some P'

This is established by induction on the height of the derivation of $E[A] \stackrel{\lambda}{\longrightarrow} P$. Since the operational semantics is defined according to the structure of the processes, we should therefore carry out a structural analysis. As A contributes in the action $E[A] \stackrel{\lambda}{\longrightarrow} P$ the process $E[X]$ can not be in prefix form.

- $E \equiv X$. Then $A \stackrel{\lambda}{\longrightarrow} P$. Clearly $a \,|\, \overline{a}.(A{+}b) \stackrel{\tau}{\longrightarrow} A{+}b \stackrel{\lambda}{\longrightarrow} P$.
- $E[X] \equiv R \,|\, E_1[X]$. It is clear that P should be of the form $(\widetilde{w})(R_1 \,|\, P_1)$. There are several cases:
  - $E_1[A] \stackrel{\lambda}{\longrightarrow} P_1$. This is simple using induction.
  - $\lambda = \tau$ and it is caused by a communication between R and $E_1[A]$. There are ten subcases. All can be routinely verified.
- $E[X] \equiv (z)E_1[X]$. In this case we continue the structural analysis with $E_1[X]$.
- $E[X] \equiv [v{=}v]E_1[X]$. Now $E_1[A] \stackrel{\lambda}{\longrightarrow} P$ has a derivation tree of less height. So we could apply the induction hypothesis.

- $E[X] \equiv E_1[X] + E_2[X]$. Suppose that $E_1[A]+E_2[A] \xrightarrow{\lambda} P$ is caused by $E_1[A] \xrightarrow{\lambda} P$. Using the induction hypothesis one gets that $E_1[a] \,|\, \overline{a}.(A+b) \stackrel{\lambda}{\Longrightarrow} P' \sim P$ for some P'. Thus $(E_1[a]+E_2[a]) \,|\, \overline{a}.(A+b) \stackrel{\lambda}{\Longrightarrow} P' \sim P$.
- $E[X] \equiv \mathbf{fix}Y.E_1[X,Y]$. By the labeled semantics one must have $E_1[A, \mathbf{fix}Y.E_1[A,Y]] \xrightarrow{\lambda} P$. Thus $E_1[a, \mathbf{fix}Y.E_1[a,Y]] \,|\, \overline{a}.(A+b) \stackrel{\lambda}{\Longrightarrow} P' \sim P$ by induction hypothesis. So $E[a] \,|\, \overline{a}.(A+b) \stackrel{\lambda}{\Longrightarrow} P' \sim P$.

(i) then follows easily.

(ii) The proof is by induction on derivation. It is easy to show that there must exist E_1, E_2, E_3, E_4 such that $(\widetilde{y})(E[a] \,|\, \overline{a}.(A+b)) \stackrel{\lambda}{\Longrightarrow} P'$ takes the following shape:

$$\begin{array}{rcl} (\widetilde{y})(E[a] \,|\, \overline{a}.(A+b)) & \Longrightarrow & (\widetilde{y})(E_1[a] \,|\, \overline{a}.(A+b)) \\ & \xrightarrow{\tau} & (\widetilde{y})(E_2[\mathbf{0}] \,|\, (A+b)) \\ & \Longrightarrow & (\widetilde{y})(E_3[\mathbf{0}] \,|\, (A+b)) \\ & \xrightarrow{\lambda} & (\widetilde{y'})(E_4[\mathbf{0}] \,|\, A') \\ & \Longrightarrow & P' \end{array}$$

Using structural induction it is routine to show that $(\widetilde{y})E[A] \Longrightarrow (\widetilde{y})E_1[A] \xrightarrow{\lambda} (\widetilde{y'})E_4[A'] \Longrightarrow P \sim P'$ for some P since $(\widetilde{y'})E_4[A'] \sim (\widetilde{y'})(E_4[\mathbf{0}] \,|\, A')$. □

Lemma 23. *Suppose $\widetilde{b}, c, d$ are fresh names. If $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, (\widetilde{y})(\overline{c}.(H+d) \,|\, E[c])) \approx (\widetilde{x})(\widetilde{\overline{b}x} \,|\, (\widetilde{z})(\overline{c}.(G+d) \,|\, F[c]))$ then $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, (\widetilde{y})E[H]) \approx (\widetilde{x})(\widetilde{\overline{b}x} \,|\, (\widetilde{z})F[G])$.*

Proof. Let $\mathcal{R}$ be the following relation:

$$\{((\widetilde{y})E[A], (\widetilde{z})F[B]) \mid (\widetilde{y})(E[a] \,|\, \overline{a}.(A+b)) \approx_l (\widetilde{z})(F[a] \,|\, \overline{a}.(B+b)), \text{ where } a, b \text{ are fresh}\} \cup \approx_l$$

It is routine to show that $\mathcal{R}$ is closed under substitution of names. Suppose that $(\widetilde{z})F[B]\mathcal{R}(\widetilde{y})E[A] \xrightarrow{\lambda} P$.

- If A does not participate in the action $(\widetilde{y})E[A] \xrightarrow{\lambda} P$, then $P \equiv (\widetilde{y})E'[A]$ for some $E'[X]$. Clearly $(\widetilde{y})(E[a] \,|\, \overline{a}.(A+b)) \xrightarrow{\lambda} (\widetilde{y})(E'[a] \,|\, \overline{a}.(A+b))$. To match that there must have some $F'[X]$ such that $(\widetilde{z})(F[a] \,|\, \overline{a}.(A+b)) \stackrel{\lambda}{\Longrightarrow} (\widetilde{z})(F'[a] \,|\, \overline{a}.(A+b)) \approx_l (\widetilde{y})(E'[a] \,|\, \overline{a}.(A+b))$. Hence $(\widetilde{z})F[B] \stackrel{\lambda}{\Longrightarrow} (\widetilde{z})F'[B]\mathcal{R}(\widetilde{y})E'[A]$.
- If A contributes in the action $(\widetilde{y})E[A] \xrightarrow{\lambda} P$, then by Lemma 22, $(\widetilde{y})(E[a] \,|\, \overline{a}.(A+b)) \stackrel{\lambda}{\Longrightarrow} P' \sim P$ for some P'. Therefore $(\widetilde{z})(F[a] \,|\, \overline{a}.(B+b)) \stackrel{\lambda}{\Longrightarrow} Q' \approx_l P'$. By Lemma 22 again, $(\widetilde{z})F[B] \stackrel{\lambda}{\Longrightarrow} Q \sim Q'$, which implies that $(\widetilde{z})F[B] \stackrel{\lambda}{\Longrightarrow} Q\mathcal{R}P$.

We conclude that $\mathcal{R}$ is a local bisimulation. □

Sangirogi's characterization of the higher order output actions depends crucially on the replication or recursion. This would not be helpful if one is interested in the algorithmic aspect of the calculus. For our calculus this problem can be removed thanks to the linearity.

Theorem 24 (Concretion)**.** *Suppose $\widetilde{c} = c_1, \ldots, c_n$ are pairwise distinct fresh names and $\widetilde{z} = z_1, \ldots, z_n$ are pairwise distinct. The following statements are equivalent in LHOπ:*
(i) $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, (\widetilde{x})\overline{b}[H].E) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, (\widetilde{y})\overline{b}[K].F)$ for some b;
(ii) $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, (\widetilde{x})(\overline{a}[H] \,|\, E)) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, (\widetilde{y})(\overline{a}[K] \,|\, F))$ for a fresh a;
(ii') $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, (\widetilde{x})(\overline{a}.(H+b) \,|\, E)) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, (\widetilde{y})(\overline{a}.(K+b) \,|\, F))$ for fresh a, b;
(iii) $(\widetilde{z})(\widetilde{\overline{c}z} \,|\, (\widetilde{x})(G[H] \,|\, E)) \approx_l (\widetilde{z})(\widetilde{\overline{c}z} \,|\, (\widetilde{y})(G[K] \,|\, F))$ for every process expression $G[X]$.

Proof. (i⇒ii): Suppose $(\widetilde{x})\overline{b}[A].P \approx_l (\widetilde{y})\overline{b}[B].Q$. Now $(\widetilde{x})\overline{b}[A].P \stackrel{(\widetilde{x_1})\overline{b}[A]}{\longrightarrow} (\widetilde{x_2})P$, where $\widetilde{x} = \widetilde{x_1}\widetilde{x_2}$, must be matched up by $(\widetilde{y})\overline{b}[B].Q \stackrel{(\widetilde{y_1})\overline{b}[B]}{\longrightarrow} (\widetilde{y_2})Q \Longrightarrow (\widetilde{y_2})Q'$ for some Q', where $\widetilde{y} = \widetilde{y_1}\widetilde{y_2}$, such that for each fresh name a it holds that $(\widetilde{y})(\overline{a}[B] \,|\, Q) \Longrightarrow (\widetilde{y})(\overline{a}[B] \,|\, Q') \approx_l (\widetilde{x})(\overline{a}[A] \,|\, P)$. Similarly

$$(\widetilde{x})(\overline{a}[A] \,|\, P) \Longrightarrow (\widetilde{x})(\overline{a}[A] \,|\, P') \approx_l (\widetilde{y})(\overline{a}[B] \,|\, Q)$$

Hence $(\widetilde{x})(\overline{a}[A] \,|\, P) \approx_l (\widetilde{y})(\overline{a}[B] \,|\, Q)$ by Bisimulation Lemma.

(iii⇔ii) is just Lemma 13. (iii⇒i) is valid by definition. (iii⇒ii') is obvious. And (ii'⇒iii) is an easy consequence of Lemma 23. □

5 Local Congruence

The local bisimulations are defined on the set of processes and are extended to the set of process expressions. The Abstraction Theorem seems to suggest that one might as well define the bisimulations on the process expressions in the first place. It would be interesting to see if the bisimilarity so defined coincide with the local bisimilarity.

Definition 25. A symmetric relation $\mathcal{R}$ on $\mathcal{LHPE}$ is a bisimulation if it is closed under substitution of names and of variables, and whenever $E\mathcal{R}F$ then the following properties hold:
(i) If $E \xrightarrow{\tau} E'$ then F' exists such that $F \Longrightarrow F'\mathcal{R}E'$.
(ii) If $E \xrightarrow{\alpha x} E'$ then F' exists such that $F \stackrel{\alpha x}{\Longrightarrow} F'\mathcal{R}E'$.
(iii) If $E \xrightarrow{\overline{a}(x)} E'$ then F' exists such that $F \stackrel{\overline{a}(x)}{\Longrightarrow} F'$ and $(x)(E' \,|\, G)\ \mathcal{R}\ (x)(F' \,|\, G)$ for all G.
(iv) If $E \xrightarrow{aX} E'$ and $X \notin fv(E \,|\, F)$ then F' exists such that $F \stackrel{aX}{\Longrightarrow} F'\mathcal{R}E'$.
(v) If $E \xrightarrow{(\widetilde{x})\overline{a}G} E'$ then some $\widetilde{x'}, H, F'$ exist such that $F \stackrel{(\widetilde{x'})\overline{a}H}{\Longrightarrow} F'$ and $(\widetilde{x})(K[G] \,|\, E')\ \mathcal{R}\ (\widetilde{x'})(K[H] \,|\, F')$ for every process expression $K[X]$ such that $\widetilde{x}\widetilde{x'} \cap fn(K[X]) = \emptyset$.
E is bisimilar to F, notation $E \approx F$, if there exists a bisimulation $\mathcal{R}$ such that $(E, F) \in \mathcal{R}$.

Several points need be made. First bisimulations are closed under substitution of variables. That means that if $E \approx F$ then $E\{H/X\} \approx F\{H/X\}$ for every H. Second the simulation of the higher order input actions is defined in the late style. By the definition $E \xrightarrow{aX} E'$ is simulated by $F \stackrel{aX}{\Longrightarrow} F'$ such that $E\{H/X\} \approx F\{H/X\}$ for every process expression H. Working with the process expressions makes it easy to define the late semantics.

The following result is reassuring.

Proposition 26. *$E \approx_l F$ if and only if $E \approx F$.*

Proof. It is easy to see that $\approx$ and $\approx_l$ coincide on $\mathcal{LHP}$. Suppose E, F contain the free variables $X_1, \ldots, X_n$. If $E \approx F$ then it follows from the closure property of $\approx$ that $E\{A_1/X_1, \ldots, A_n/X_n\} \approx F\{A_1/X_1, \ldots, A_n/X_n\}$. Thus $E\{A_1/X_1, \ldots, A_n/X_n\} \approx_l F\{A_1/X_1, \ldots, A_n/X_n\}$ by definition. Hence $E \approx_l F$.

On the other hand $\approx_l$ is closed under substitution of variables. Let $a_1, \ldots, a_n$ be distinct fresh names. If $E \approx_l F$ then by definition $E\{a_1/X_1, \ldots, a_n/X_n\} \approx_l F\{a_1/X_1, \ldots, a_n/X_n\}$. Now suppose that $E \xrightarrow{\lambda} E'$ where λ is not a higher order output. Then

$$E\{a_1/X_1, \ldots, a_n/X_n\} \stackrel{\lambda\{a_1/X_1,\ldots,a_n/X_n\}}{\longrightarrow} E'\{a_1/X_1, \ldots, a_n/X_n\}$$

by Lemma 4. This action must be matched up by

$$F\{a_1/X_1, \ldots, a_n/X_n\} \stackrel{\lambda\{a_1/X_1,\ldots,a_n/X_n\}}{\Longrightarrow} F'\{a_1/X_1, \ldots, a_n/X_n\}$$

By Lemma 5, one has that $F \stackrel{\lambda}{\Longrightarrow} F'$. Now $E'\{a_1/X_1, \ldots, a_n/X_n\}$ and $F'\{a_1/X_1, \ldots, a_n/X_n\}$ are related by local bisimilarity, which must imply that E' and F' are related by the local bisimilarity using Theorem 21. For higher order output actions the argument is similar. □

The above proposition implies that the late simulation of higher order output actions stated as clause (iv') on page 11 is equivalent to early simulation defined in (iv) of Definition 10.

In the rest of the paper we shall write $\approx$ for any of the bisimilarities defined in Definition 9, Definition 10 and Definition 25.

Having worked out the definition and the properties of the bisimilarity for the linear higher order processes, we are in the position to study the congruence induced by the bisimilarity. The definition of congruence is standard.

Definition 27. E and F are congruent, notation $E \simeq F$, if $E \approx F$ and the following properties hold for every first order substitution σ:
(i) If $E\sigma \xrightarrow{\tau} E'$ then $F\sigma \stackrel{\tau}{\Longrightarrow} F' \approx E'$ for some F';
(ii) If $F\sigma \xrightarrow{\tau} F'$ then $E\sigma \stackrel{\tau}{\Longrightarrow} E' \approx F'$ for some E'.

An interesting point about the above definition is that there is no explicit requirement that $\simeq$ is closed under substitution of variables. This is not necessary.

Lemma 28. *The relation $\simeq$ is closed under substitution of names and variables.*

Proof. By definition $\simeq$ is closed under substitution of names. Now suppose $E \simeq F$ and

$$E\{H_1/X_1, \ldots, H_n/X_n\} \stackrel{\tau}{\longrightarrow} E'$$

is caused by $H_i \stackrel{\tau}{\longrightarrow} H_i'$. Then $E\{H_1/X_1, \ldots, a.H_i'/X_i, \ldots, H_n/X_n\} \,|\, \overline{a} \stackrel{\tau}{\longrightarrow} E'$ for a fresh a. Therefore

$$F\{H_1/X_1, \ldots, a.H_i'/X_i, \ldots, H_n/X_n\} \,|\, \overline{a} \stackrel{\tau}{\Longrightarrow} F' \approx E'$$

for some F'. Consequently $F\{H_1/X_1, \ldots, H_i/X_i, \ldots, H_n/X_n\} \stackrel{\tau}{\Longrightarrow} F' \approx E'$. If

$$E\{H_1/X_1, \ldots, H_n/X_n\} \stackrel{\tau}{\longrightarrow} E'$$

is caused by E then $E' \equiv E_1\{H_1/X_1, \ldots, H_n/X_n\}$ and $E \stackrel{\tau}{\longrightarrow} E_1$. Consequently $F \stackrel{\tau}{\Longrightarrow} F_1 \approx E_1$ for some F_1. But then $F\{H_1/X_1, \ldots, H_n/X_n\} \stackrel{\tau}{\Longrightarrow} F_1\{H_1/X_1, \ldots, H_n/X_n\} \approx E_1\{H_1/X_1, \ldots, H_n/X_n\}$ by Lemma 4. □

The relation $\simeq$ is indeed a congruence relation.

Proposition 29. *Suppose $E \simeq F$. Then the following equalities hold:*
(i) $a(x).E \simeq a(x).F$, $\overline{a}x.E \simeq \overline{a}x.F$;
(ii) $E \,|\, G \simeq F \,|\, G$, $G \,|\, E \simeq G \,|\, F$, $(x)E \simeq (x)F$ and $[x{=}y]E \simeq [x{=}y]F$;
(iii) $a(X).E \simeq a(X).F$;
(iv) $\overline{a}[H].E \simeq \overline{a}[H].F$, $\overline{a}[E].H \simeq \overline{a}[F].H$;
(v) $E{+}G \simeq F{+}G$, $G{+}E \simeq G{+}F$;
(vi) $\mathbf{fix}X.E \simeq \mathbf{fix}X.F$.

Proof. (i) through (v) are proved by extending the proof of Lemma 15 with the help of Lemma 28. (vi) is subsumed by Proposition 42. □

It is easy to axiomatize the congruence property stated in Proposition 29. However these axioms are not strong enough to derive equalities involving the localization operator in the most general form. Let's take a look at one example. Suppose $(x)(\overline{a}x \,|\, E) \approx (x)(\overline{a}x \,|\, F)$ where $a \notin fn(E \,|\, F)$. It is not difficult to see that $(x)(\overline{a}x \,|\, c(z).E) \approx (x)(\overline{a}x \,|\, c(z).F)$. This equivalence holds even if $c = x$. In some sense this is a local congruence property: If E and F are equivalent in a local context $(x)(\overline{a}x \,|\, \_)$ then $c(z).E$ and $c(z).F$ are equivalent in the same local context. For the congruences defined in [8] the local congruence is a derived property. Since from the equivalence between $(x)(\overline{a}x \,|\, E)$ and $(x)(\overline{a}x \,|\, F)$ one could derive the equivalence between $(x)(\overline{a}x \,|\, c(z).E)$ and $(x)(\overline{a}x \,|\, c(z).F)$. For the congruence defined in this paper as well as the (quasi) open congruence studied in [14, 16], the local congruence property does not seem to be equivalent to the global congruence property. For algebraic studies we need a theory of local congruence. The following lemmas are all about the local congruence property.

Lemma 30. *If $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F)$ for pairwise distinct fresh $\widetilde{b}$, then $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, E \,|\, G) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F \,|\, G)$.*

Proof. Suppose $\widetilde{x} \cap fn(G) = \{x_{i_1}, \ldots, x_{i_n}\}$. Let $a_{i_1}, \ldots, a_{i_n}$ be fresh and let H be

$$b_{i_1}(x_{i_1}).(\ldots.(b_{i_n}(x_{i_n}).(G \,|\, \overline{b_{i_1}}x_{i_1} \,|\, \ldots \,|\, \overline{b_{i_n}}x_{i_n}) + a_{i_n}) \ldots) + a_{i_1}$$

It is clear that $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, E) \,|\, H \Longrightarrow\sim (\widetilde{x})(\widetilde{\overline{b}x} \,|\, E \,|\, G)$ is simulated by

$$(\widetilde{x})(\widetilde{\overline{b}x} \,|\, F) \,|\, H \Longrightarrow\sim (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F \,|\, G) \Longrightarrow\approx (\widetilde{x})(\widetilde{\overline{b}x} \,|\, E \,|\, G)$$

Similarly $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, E \,|\, G) \Longrightarrow\approx (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F \,|\, G)$. The result follows from Bisimulation Lemma. □

Corollary 31. *If $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F)$ for pairwise distinct fresh $\widetilde{b}$, then $(\widetilde{x})(E \,|\, G) \simeq (\widetilde{x})(F \,|\, G)$ for every process expression G.*

Proof. By Lemma 30 one has $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, E \,|\, G) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F \,|\, G)$. Use the context $b_1(x_1).\ldots.b_n(x_n)$ to consume $\widetilde{\overline{b}x}$ and use the Bisimulation Lemma to complete the argument. □

The most interesting local congruence property is to do with prefix operations.

Lemma 32. *Suppose $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F)$ for pairwise distinct fresh $\widetilde{b}$. The followings hold:*
(i) $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, \tau.E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \tau.F)$;
(ii) $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, a(z).E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, a(z).F)$ whenever $z \notin \widetilde{x}$;
(iii) $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, \overline{a}z.E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \overline{a}z.F)$;
(iv) $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, a(X).E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, a(X).F)$;
(v) $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, \overline{a}H.E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \overline{a}H.F)$.

Proof. Let $\mathcal{R}$ be the following relation:

$$\left\{ ((\widetilde{x})(\widetilde{y})(O \,|\, \lambda.E), (\widetilde{x})(\widetilde{y})(O \,|\, \lambda.F)) \;\middle|\; \begin{array}{l} (\widetilde{x})(\widetilde{\overline{b}x} \,|\, E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F) \\ \text{where } \widetilde{b} \text{ are fresh names} \\ \lambda \text{ binds and localizes none of } \widetilde{x} \\ \widetilde{y} = y_1, \ldots, y_m \text{ are pairwise distinct} \end{array} \right\} \cup \simeq$$

The relation $\mathcal{R}$ is closed under substitution of names and of variables. It is a bisimulation up to $\sim$ by Corollary 31. □

In Lemma 32, λ should not be an input prefix that binds one of $\widetilde{x}$. For instance $(x)(\overline{b}x \,|\, [x{=}y]\mathbf{0}) \simeq (x)(\overline{b}x \,|\, [x{=}y]y)$. But $(x)(\overline{b}x \,|\, y(x).[x{=}y]\mathbf{0}) \not\simeq (x)(\overline{b}x \,|\, y(x).[x{=}y]y)$. On the other hand, from $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F)$ one could derive $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, \overline{a}(x_i).E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \overline{a}(x_i).F)$ for every $x_i \in \widetilde{x}$. This is because $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F)$ implies $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, \overline{a}x_i.E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \overline{a}x_i.F)$, which in turn implies that

$$(\widetilde{x})(\overline{b_1}x_1 \,|\, \ldots \,|\, \overline{b_{i-1}}x_{i-1} \,|\, \overline{b_{i+1}}x_{i+1} \,|\, \ldots \,|\, \overline{b_n}x_n \,|\, \overline{a}x_i.E)$$

and

$$(\widetilde{x})(\overline{b_1}x_1 \,|\, \ldots \,|\, \overline{b_{i-1}}x_{i-1} \,|\, \overline{b_{i+1}}x_{i+1} \,|\, \ldots \,|\, \overline{b_n}x_n \,|\, \overline{a}x_i.F)$$

are congruent by Corollary 31. Therefore

$$(x_1 \ldots x_{i-1}x_{i+1} \ldots x_n)(\overline{b_1}x_1 \,|\, \ldots \,|\, \overline{b_{i-1}}x_{i-1} \,|\, \overline{b_{i+1}}x_{i+1} \,|\, \ldots \,|\, \overline{b_n}x_n \,|\, \overline{a}(x_i).E)$$

and

$$(x_1 \ldots x_{i-1}x_{i+1} \ldots x_n)(\overline{b_1}x_1 \,|\, \ldots \,|\, \overline{b_{i-1}}x_{i-1} \,|\, \overline{b_{i+1}}x_{i+1} \,|\, \ldots \,|\, \overline{b_n}x_n \,|\, \overline{a}(x_i).F)$$

are congruent. Hence $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, \overline{a}(x_i).E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \overline{a}(x_i).F)$.

Using the above arguments one could prove the following two lemmas about localization.

Lemma 33. *If $(\widetilde{x})(z)(\widetilde{\overline{b}x} \,|\, \overline{b'}z \,|\, E) \simeq (\widetilde{x})(z)(\widetilde{\overline{b}x} \,|\, \overline{b'}z \,|\, F)$ for pairwise distinct fresh $\widetilde{b}, b'$, then it follows that $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, (z)E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, (z)F)$.*

Lemma 34. *If $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F)$ for pairwise distinct fresh $\widetilde{b}$, then $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, (z)E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, (z)F)$.*

The next two lemmas are about the local properties for choice and match operations.

Lemma 35. *If $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F)$ and $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, G) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, H)$ for pairwise distinct fresh $\widetilde{b}$, then $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, (E{+}G)) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, (F{+}H))$.*

Proof. Let $\mathcal{R}$ be the following relation:

$$\left\{ ((\widetilde{x})(O \,|\, (E{+}G)), (\widetilde{x})(O \,|\, (F{+}H))) \;\middle|\; \begin{array}{l} (\widetilde{x})(\widetilde{\overline{b}x} \,|\, E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F) \\ (\widetilde{x})(\widetilde{\overline{b}x} \,|\, G) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \,|\, H) \\ \text{where } \widetilde{b} \text{ are fresh names} \end{array} \right\} \cup \simeq$$

It is clearly closed under substitution of names. It is a simple exercise to show that $\mathcal{R}$ is a bisimulation up to $\sim$. □

Lemma 36. *If* $(\widetilde{x})(\widetilde{\overline{b}x} \mid E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \mid F)$ *for pairwise distinct fresh* $\widetilde{b}$*, then* $(\widetilde{x})(\widetilde{\overline{b}x} \mid \varphi E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \mid \varphi F)$ *for every condition* φ *that does not contain any of* $\overline{b}$*.*

Proof. By the L-rules and their derived rules, we may assume that none of $\widetilde{x}$ appears in φ. The equality $(\widetilde{x})(\widetilde{\overline{b}x} \mid \varphi E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \mid \varphi F)$ follows essentially from the fact that $(\widetilde{x})(\widetilde{\overline{b}x} \mid E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \mid F)$ is closed under substitution of names. □

The next three lemmas discuss the local congruence issue for the higher order features of the calculus.

Lemma 37. *Suppose* $\widetilde{b}$*,* c *and* d *are pairwise distinct fresh names. If*

$$(\widetilde{x})(\widetilde{\overline{b}x} \mid (\widetilde{y})(\overline{c}.(G{+}d) \mid E)) \approx (\widetilde{x})(\widetilde{\overline{b}x} \mid (\widetilde{z})(\overline{c}.(H{+}d) \mid F))$$

then $(\widetilde{x})(\widetilde{\overline{b}x} \mid (\widetilde{y})\overline{a}[G].E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \mid (\widetilde{z})\overline{a}[H].F)$*.*

Proof. Observe that for each process expression $O[X]$ such that $\widetilde{y} \cap fn(O[X]) = \emptyset$, it follows from

$$(\widetilde{x})(\widetilde{\overline{b}x} \mid (\widetilde{y})(\overline{c}.(G{+}d) \mid E)) \approx (\widetilde{x})(\widetilde{\overline{b}x} \mid (\widetilde{z})(\overline{c}.(H{+}d) \mid F))$$

and Lemma 30 that

$$(\widetilde{x})(\widetilde{\overline{b}x} \mid (\widetilde{y})(\overline{c}.(G{+}d) \mid O[c] \mid E)) \approx (\widetilde{x})(\widetilde{\overline{b}x} \mid (\widetilde{z})(\overline{c}.(H{+}d) \mid O[c] \mid F)) \tag{13}$$

By applying Lemma 23 to (13) one obtains that

$$(\widetilde{x})(\widetilde{\overline{b}x} \mid (\widetilde{y})(O[G] \mid E)) \approx (\widetilde{x})(\widetilde{\overline{b}x} \mid (\widetilde{z})(O[H] \mid F)) \tag{14}$$

The equivalence (14) makes it clear that

$$\left\{ ((\widetilde{x})(O \mid (\widetilde{y})\overline{a}[G].E), (\widetilde{x})(O \mid (\widetilde{z})\overline{a}[H].F)) \;\middle|\; \begin{array}{l} (\widetilde{x})(\widetilde{\overline{b}x} \mid (\widetilde{y})(\overline{c}.(G{+}d) \mid E)) \approx (\widetilde{x})(\widetilde{\overline{b}x} \mid (\widetilde{z})(\overline{c}.(H{+}d) \mid F)) \\ \text{where } b, c, d \text{ are fresh names} \end{array} \right\} \cup \simeq$$

is a bisimulation up to $\sim$. □

Lemma 38. *If* $(\widetilde{x})(\widetilde{\overline{b}x} \mid E) \approx (\widetilde{x})(\widetilde{\overline{b}x} \mid F)$ *for pairwise distinct fresh names* $\widetilde{b}$*, then* $(\widetilde{x})(\widetilde{\overline{b}x} \mid \overline{a}[E].H) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \mid \overline{a}[F].H)$*.*

Proof. By Lemma 30, Lemma 32 and Lemma 35, one has that

$$(\widetilde{x})(\widetilde{\overline{b}x} \mid (\overline{c}.(E{+}d) \mid H)) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \mid (\overline{c}.(F{+}d) \mid H))$$

We are done by applying Lemma 37. □

Lemma 39. *If* $(\widetilde{x})(\widetilde{\overline{b}x} \mid E[X]) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \mid F[X])$ *and* $(\widetilde{x})(\widetilde{\overline{b}x} \mid G) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \mid H)$ *for pairwise distinct fresh names* $\widetilde{b}$*, then* $(\widetilde{x})(\widetilde{\overline{b}x} \mid E[G]) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \mid F[H])$*.*

Proof. By Lemma 32 and Lemma 38,

$$(\widetilde{x})(\widetilde{\overline{b}x} \mid a(X).E[X]) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \mid a(X).F[X])$$

and

$$(\widetilde{x})(\widetilde{\overline{b}x} \mid \overline{a}[G]) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \mid \overline{a}[H])$$

for a fresh a. It follows from Lemma 30 that

$$(\widetilde{x})(\widetilde{\overline{b}x} \mid a(X).E[X] \mid \overline{a}[G]) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \mid a(X).F[X] \mid \overline{a}[H])$$

It is then routine to derive $(\widetilde{x})(\widetilde{\overline{b}x} \mid E[G]) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \mid F[H])$ using Bisimulation Lemma. □

The results stated in the above lemmas and in Proposition 42 to be proved in the next section can be summarized by the following theorem.

Theorem 40 (Local Congruence)**.** *Suppose* $(\widetilde{x})(\widetilde{\overline{b}x} \mid E) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \mid F)$ *for pairwise distinct fresh names* $\widetilde{b}$*. Then* $(\widetilde{x})(\widetilde{\overline{b}x} \mid C[E]) \simeq (\widetilde{x})(\widetilde{\overline{b}x} \mid C[F])$ *for every full context* $C[\_]$ *that neither contains any of* $\widetilde{b}$ *nor binds any of* $\widetilde{x}$*.*

6 Recursion

Whatever the equivalence relation one introduces for the calculus, one needs to make sure that it is well-behaved with regards to the fundamental operator of the calculus. The **fix**-operator is a fundamental operator. To establish the closure property for the **fix**-operator, we need to carry out inductions on the height of derivations, making full use of the fact that, no matter what, the height of a derivation tree is finite. One consequence of this finitary property is that if $\mathbf{fix}X.E \xrightarrow{\lambda} A$ then there is a natural number i no greater than the height of the derivation of $\mathbf{fix}X.E \xrightarrow{\lambda} A$ such that

$$\underbrace{E[E[\ldots[E}_{i \text{ times}}[\mathbf{fix}X.E]]\ldots]] \xrightarrow{\lambda} A$$

Moreover some λ' and $E'[X]$ exist such that $\lambda = \lambda'\{\mathbf{fix}X.E/X\}$, $A \equiv A'\{\mathbf{fix}X.E/X\}$ and

$$\underbrace{E[E[\ldots[E}_{i \text{ times}}[F]]\ldots]] \xrightarrow{\lambda'\{F/X\}} E'[F] \tag{15}$$

for all F. Intuitively F does not participate in the action of (15). The action in (15) is caused solely by the process expression $\underbrace{E[E[\ldots[E}_{i \text{ times}}[X]]\ldots]]$. By Lemma 4 the reduction in (15) is subsumed by

$$\underbrace{E[E[\ldots[E}_{i \text{ times}}[X]]\ldots]] \xrightarrow{\lambda'} E'[X] \tag{16}$$

This fact is a crucial observation about the operational semantics of the recursive process expressions. It underlines almost all the properties of the recursively defined processes.

Lemma 41. *Suppose $G[\mathbf{fix}X.E] \xrightarrow{\lambda} K$. Then the following two properties hold:*
(i) $K \equiv G'[\mathbf{fix}X.E]$ for some process expression $G'[X]$ and $\lambda \equiv \lambda'\{\mathbf{fix}X.E/X\}$ for some λ';
(ii) There exists some natural number i such that $G[\underbrace{E[E[\ldots[E}_{i \text{ times}}[X]]\ldots]]] \xrightarrow{\lambda'} G'[X]$ and i is no greater than the height of the derivation $G[\mathbf{fix}X.E] \xrightarrow{\lambda} K$.

Proof. We prove the lemma by induction on the height of the derivation $G[\mathbf{fix}X.E] \xrightarrow{\lambda} K$. For that purpose we need to take a look at the structure of G:

- $G \equiv X$. By the operational semantics one must have $E[\mathbf{fix}X.E] \xrightarrow{\lambda} K$ with a shorter derivation. In this case we can simply apply the induction hypothesis.
- $G \equiv G_1 \,|\, G_2$. Either $X \notin fv(G_1)$ or $X \notin fv(G_2)$. Suppose $X \notin fv(G_2)$. If for example the transition $G[\mathbf{fix}X.E] \xrightarrow{\lambda} K$ is caused by $G_1[\mathbf{fix}X.E] \xrightarrow{\lambda_1} K_1$ and $G_2 \xrightarrow{\lambda_2} K_2$. By induction hypothesis some i_1, λ_1' and $G_1'[X]$ exist such that $\lambda_1 = \lambda_1'\{\mathbf{fix}X.E/X\}$, $K_1 \equiv G_1'[\mathbf{fix}X.E]$ and

  $$G_1[\underbrace{E[\ldots[E}_{i_1 \text{ times}}[X]]\ldots]] \xrightarrow{\lambda_1'} G_1'[X]$$

  It follows that

  $$G[\underbrace{E[\ldots[E}_{i_1 \text{ times}}[X]]\ldots]] \xrightarrow{\lambda'} (\widetilde{v})(G_1'[X] \,|\, K_2)$$

  for some $\lambda', \widetilde{v}$. Now let $G'[X]$ be $(\widetilde{v})(G_1'[X] \,|\, K_2)$. Then $G[\underbrace{E[\ldots[E}_{i_1 \text{ times}}[X]]\ldots]] \xrightarrow{\lambda'} G'[X]$.
- $G \equiv \mathbf{fix}Y.G_1[X, Y]$. It follows from $\mathbf{fix}Y.G_1[\mathbf{fix}X.E, Y] \xrightarrow{\lambda} K$ that

  $$G_1[\mathbf{fix}X.E, \mathbf{fix}Y.G_1[\mathbf{fix}X.E, Y]] \xrightarrow{\lambda} K$$

is derivable with a shorter derivation. Let $G_2[X]$ be $G_1[X, \mathbf{fix}Y.G_1[X,Y]]$. Then by induction hypothesis one sees that some $i, \lambda', G'[X]$ exist such that $\lambda = \lambda'\{\mathbf{fix}X.E/X\}$, $G'[\mathbf{fix}X.E] \equiv K$ and

$$G_2[\underbrace{E[\ldots[E}_{i \text{ times}}[X]]\ldots]] \xrightarrow{\lambda'} G'[X]$$

It follows that $G[\underbrace{E[\ldots[E}_{i \text{ times}}[X]]\ldots]] \xrightarrow{\lambda'} G'[X]$.

- The other cases are simpler.

This completes the proof. □

One could prove a similar result for the replicator: If $!E \xrightarrow{\lambda} K$ then $K \equiv E' \,|\, !E$ and there is a natural number i, no greater than the height of the derivation of $!E \xrightarrow{\lambda} K$, such that

$$\underbrace{E \,|\, \ldots \,|\, E}_{i \text{ times}} \,|\, X \xrightarrow{\lambda} E' \,|\, X$$

The above lemma provides a powerful tool to reason about the recursive processes. One of its implications is the congruence property of the **fix**-operator. This property has been established in the framework of CCS by Milner ([7]). Milner's proof has been simplified by Ying ([23]). We prove below a localized version. We believe that our proof is more instructive.

Proposition 42. *Suppose $\widetilde{b}$ are pairwise distinct fresh names. The congruence $(\widetilde{x})(\widetilde{\bar{b}x} \,|\, E) \simeq (\widetilde{x})(\widetilde{\bar{b}x} \,|\, F)$ implies the congruence $(\widetilde{x})(\widetilde{\bar{b}x} \,|\, \mathbf{fix}X.E) \simeq (\widetilde{x})(\widetilde{\bar{b}x} \,|\, \mathbf{fix}X.F)$.*

Proof. Let $\mathcal{R}$ be the following relation:

$$\left\{ ((\widetilde{x})(\widetilde{\bar{b}x} \,|\, G_1[\mathbf{fix}X.E]), (\widetilde{x})(\widetilde{\bar{b}x} \,|\, G_2[\mathbf{fix}X.F])) \;\middle|\; \begin{array}{l} (\widetilde{x})(\widetilde{\bar{b}x} \,|\, E) \simeq (\widetilde{x})(\widetilde{\bar{b}x} \,|\, F) \\ (\widetilde{x})(\widetilde{\bar{b}x} \,|\, G_1) \simeq (\widetilde{x})(\widetilde{\bar{b}x} \,|\, G_2) \\ \text{where } \widetilde{b} \text{ are distinct fresh names} \end{array} \right\}$$

We prove that $\mathcal{R}$ is a bisimulation up to $\sim$. Consider for instance the higher order output actions. By Lemma 41 we may assume for example that

$$(\widetilde{x})(\widetilde{\bar{b}x} \,|\, G_1[\mathbf{fix}X.E]) \overset{(\widetilde{x_1'})(\widetilde{y})\overline{c}R[\mathbf{fix}X.E]}{\longrightarrow} (\widetilde{x_1''})(\widetilde{\bar{b}x} \,|\, G'[\mathbf{fix}X.E])$$

for some $\widetilde{x_1'}$ and $\widetilde{x_1''}$ such that $\widetilde{x_1'}\widetilde{x_1''} = \widetilde{x}$. Again by Lemma 41 it holds for some natural number i that

$$(\widetilde{x})(\widetilde{\bar{b}x} \,|\, G_1[\underbrace{E[\ldots[E}_{i \text{ times}}[X]]]\ldots]) \overset{(\widetilde{x_1'})(\widetilde{y})\overline{c}R[X]}{\longrightarrow} (\widetilde{x_1''})(\widetilde{\bar{b}x} \,|\, G'[X])$$

By Lemma 39 one has that

$$(\widetilde{x})(\widetilde{\bar{b}x} \,|\, G_1[E[\ldots[E[X]]\ldots]]) \approx (\widetilde{x})(\widetilde{\bar{b}x} \,|\, G_2[F[\ldots[F[X]]\ldots]])$$

Therefore some $\widetilde{z}, S, H', \widetilde{x_2'}, \widetilde{x_2''}$ exist such that $\widetilde{x_2'}\widetilde{x_2''} = \widetilde{x}$ and

$$(\widetilde{x})(\widetilde{\bar{b}x} \,|\, G_2[\underbrace{F[\ldots[F}_{i \text{ times}}[X]]\ldots]]) \overset{(\widetilde{x_2'})(\widetilde{z})\overline{c}S[X]}{\Longrightarrow} (\widetilde{x_2''})(\widetilde{\bar{b}x} \,|\, H'[X])$$

and

$$(\widetilde{x_1'})(\widetilde{y})(K[R[X]] \,|\, (\widetilde{x_1''})(\widetilde{\bar{b}x} \,|\, G'[X])) \approx (\widetilde{x_2'})(\widetilde{z})(K[S[X]] \,|\, (\widetilde{x_2''})(\widetilde{\bar{b}x} \,|\, H'[X]))$$

for every process expressions $K[X]$. Therefore

$$(\widetilde{x})(\widetilde{\bar{b}x} \,|\, (\widetilde{y})(K[R[X]] \,|\, G'[X])) \approx (\widetilde{x})(\widetilde{\bar{b}x} \,|\, (\widetilde{z})(K[S[X]] \,|\, H'[X]))$$

Define

$$G'_1[X] \stackrel{\text{def}}{=} (\widetilde{y})(K[R[X]] \,|\, G'[X])$$
$$G'_2[X] \stackrel{\text{def}}{=} (\widetilde{z})(K[S[X]] \,|\, H'[X])$$

Clearly $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, G'_1[\mathbf{fix}X.E]) \;\mathcal{R}\; (\widetilde{x})(\widetilde{\overline{b}x} \,|\, G'_2[\mathbf{fix}X.F])$. Other cases can be proved similarly. □

Another interesting issue about recursion is the uniqueness of the solution to the equation $X \simeq E[X]$. The uniqueness does not come for free. An obvious counter example is that every process expression is the solution of $X \simeq X$. On the other hand the equation $X \simeq a.X$ has a unique solution. But $X \simeq X+a.X$ has many solutions of the form $A+\mathbf{fix}X.(A+a.X)$. Similarly $X \simeq X \,|\, a.X$ has for example solutions of the form $!B \,|\, \mathbf{fix}X.a.X$. In the framework of higher order processes the uniqueness of solution is more intriguing. Take for instance the equation

$$X \simeq \overline{a}[X] \tag{17}$$

The solution $\mathbf{fix}X.\overline{a}[X]$ is capable of repeatedly exporting itself through channel a. Does (17) have a unique solution? The answer is not obvious. There is however a standard result giving a sufficient condition to guarantee the uniqueness, the condition being that X must be both sequential and guarded in $E[X]$.

Definition 43. X is sequential in $E[X]$ if either $E[X] \equiv \mathbf{0}$, or $E[X] \equiv \lambda.E'[X]$ such that $X \notin fv(\lambda)$ and X is sequential in $E'[X]$, or $E[X] \equiv E_1[X]+E_2[X]$ and X is sequential in $E_1[X]$ and $E_2[X]$. X is guarded in $E[X]$ if every occurrence of X is in a sub-term $\lambda.E'$ of $E[X]$ such that $X \notin fv(\lambda)$ and $\lambda \neq \tau$.

The reason to introduce sequentiality and guardedness is the following lemma. The proof of this lemma depends crucially on the sequentiality.

Lemma 44. *Suppose X is guarded and sequential in $G[X]$. If $G[E] \stackrel{\lambda}{\longrightarrow} H$ then the followings hold:*
(i) If $\lambda = \tau$ then $H \equiv G'[E]$ for some $G'[X]$ in which X is guarded and $G[X] \stackrel{\lambda}{\longrightarrow} G'[X]$.
(ii) If $\lambda \neq \tau$ then $H \equiv G'[E]$ for some $G'[X]$ and $G[X] \stackrel{\lambda}{\longrightarrow} G'[X]$.

In the following proof we need to employ technique of the bisimulations up to $\approx$. See [15] for a discussion on the technique.

Definition 45. A symmetric relation $\mathcal{R}$ on processes is a bisimulation up to $\approx$ if it is closed under substitution of names and of variables and whenever $E\mathcal{R}F$ then the following properties hold:
(i) If $E \stackrel{\tau}{\Longrightarrow} E'$ then F' exists such that $F \Longrightarrow F' \approx \mathcal{R} \approx E'$.
(ii) If $E \stackrel{\alpha x}{\Longrightarrow} E'$ then F' exists such that $F \stackrel{\alpha x}{\Longrightarrow} F' \approx \mathcal{R} \approx E'$.
(iii) If $E \stackrel{\overline{a}(x)}{\Longrightarrow} E'$ then F' exists such that $F \stackrel{\overline{a}(x)}{\Longrightarrow} F'$ and $(x)(E' \,|\, O) \;\approx \mathcal{R} \approx\; (x)(F' \,|\, O)$ for all O.
(iv) If $E \stackrel{aX}{\Longrightarrow} E'$ and $X \notin fv(E \,|\, F)$, then F' exists such that $F \stackrel{aX}{\Longrightarrow} F' \approx \mathcal{R} \approx E'$.
(v) If $E \stackrel{(\widetilde{x})\overline{a}G}{\Longrightarrow} E'$ then some $\widetilde{x'}, H, F'$ exist such that $F \stackrel{(\widetilde{x'})\overline{a}H}{\Longrightarrow} F'$ and $(\widetilde{x})(K[G] \,|\, E') \;\approx \mathcal{R} \approx\; (\widetilde{x'})(K[H] \,|\, F')$ for every process expression $K[X]$ such that $\widetilde{x}\widetilde{x'} \cap fn(K[X]) = \emptyset$.

The usefulness of the definition is witnessed by the following lemma. The proof also makes reference to the Bisimulation Lemma.

Lemma 46. *If $\mathcal{R}$ is a bisimulation up to $\approx$ then $\mathcal{R} \subseteq \approx$.*

Proof. The proof makes use of the results from the previous section. The idea is to prove that $\approx \mathcal{R} \approx$ is a bisimulation. Now suppose that $E \approx \mathcal{R} \approx F$ and $E \stackrel{(\widetilde{y})\overline{a}G}{\longrightarrow} E'$. By definition some E_1, F_1 exist such that $E \approx E_1 \;\mathcal{R}\; F_1 \approx F$. According to Definition 10 some $\widetilde{y'}, G', E'_1$ exist such that $E_1 \stackrel{(\widetilde{y'})\overline{a}G'}{\Longrightarrow} E'_1$ and $(\widetilde{y})(K[G] \,|\, E') \;\approx \mathcal{R} \approx\; (\widetilde{y'})(K[G'] \,|\, E'_1)$ for every process expression $K[X]$. By Definition 45 some $\widetilde{z'}, H', F'_1$ exist such that $F_1 \stackrel{(\widetilde{z'})\overline{a}H'}{\Longrightarrow} F'_1$ and $(\widetilde{y'})(K[G'] \,|\, E'_1) \;\approx \mathcal{R} \approx\; (\widetilde{z'})(K[H'] \,|\, F'_1)$. Now assume that

$$F_1 \Longrightarrow F'_2 \stackrel{(\widetilde{z'})\overline{a}H'}{\longrightarrow} F'_3 \Longrightarrow F'_1$$

Then the followings hold:

- $F \Longrightarrow F_2 \approx F_2'$ for some F_2;
- $F_2 \stackrel{(\widetilde{z})\overline{a}H}{\Longrightarrow} F_3$ for some $\widetilde{z}, H, F_3$ such that $(\widetilde{z'})(\overline{b}[H'] \mid F_3') \approx (\widetilde{z})(\overline{b}[H] \mid F_3)$ for a fresh b;
- and consequently $(\widetilde{z})(\overline{b}[H] \mid F_3) \Longrightarrow (\widetilde{z})(\overline{b}[H] \mid F') \approx (\widetilde{z'})(\overline{b}[H'] \mid F_1')$ for some F'.

By Theorem 24, $(\widetilde{z})(K[H] \mid F') \approx (\widetilde{z'})(K[H'] \mid F_1')$ for every $K[X]$ such that $\widetilde{z}\widetilde{z'} \cap fn(K[X]) = \emptyset$. In summary $F \stackrel{(\widetilde{z})\overline{a}H}{\Longrightarrow} F'$ and $(\widetilde{z})(K[H] \mid F') \approx (\widetilde{y})(K[G] \mid E')$ for all $K[X]$ such that $\widetilde{z}\widetilde{z'} \cap fn(K[X]) = \emptyset$. □

We are now ready to prove the Fixpoint Theorem for $LHO\pi$. The proof of the theorem follows the general methodology. Our contribution is in showing that the general methodology applies, thanks to the Bisimulation Lemma.

Theorem 47 (Fixpoint)**.** *Suppose X is guarded and sequential in $G[X]$. If $E \simeq G[E]$ and $F \simeq G[F]$ then $E \simeq F$.*

Proof. Define $\mathcal{R}$ to be the following relation:

$$\left\{ (H[E], H[F]) \;\middle|\; \begin{array}{l} E \simeq G[E] \text{ and } F \simeq G[F] \\ X \text{ is guarded and sequential in } G[X] \text{ and } H[X] \end{array} \right\}$$

Suppose $H[E] \stackrel{(\widetilde{y})\overline{a}J}{\Longrightarrow} K$. By Lemma 44, some $H_1[X], H_2[X]$ exist such that the following properties hold:

- $H[E] \Longrightarrow H_1[E] \stackrel{(\widetilde{y})\overline{a}J}{\longrightarrow} H_2[E] \Longrightarrow K$;
- X is guarded in $H_1[X]$;
- $H[F] \Longrightarrow H_1[F] \stackrel{(\widetilde{y})\overline{a}J}{\longrightarrow} H_2[F]$;

Now $E \simeq G[E]$ implies $H_2[E] \simeq H_2[G[E]]$. Since X is guarded in $H_2[G[X]]$, Lemma 44 implies that $H_2[G[E]] \Longrightarrow H'[E] \approx K$ for some $H'[X]$ in which X is guarded and that $H_2[G[F]] \Longrightarrow H'[F]$. It follows from $H_2[F] \simeq H_2[G[F]]$ that $H_2[E] \Longrightarrow L \approx H'[E]$ for some L. Conclude that $H[F] \stackrel{(\widetilde{y})\overline{a}J}{\Longrightarrow} L$ and that $(\widetilde{y})(N[J] \mid K) \approx \mathcal{R} \approx (\widetilde{y})(N[J] \mid L)$ for every $N[X]$ such that $\widetilde{y} \cap fn(N[X]) = \emptyset$. This should be enough to convince the reader that $\mathcal{R}$ is a bisimulation up to $\approx$. By Lemma 46 one has that $G[E] \approx G[F]$. But clearly $G[E] \simeq G[F]$. Hence $E \simeq G[E] \simeq G[F] \simeq F$. □

The Fixpoint Theorem was first studied by Milner for CCS ([7]). Ying and Wirsing studied in [24] the fixpoint property for the strong congruence on $HOCCS$ processes. Using the Bisimulation Lemma we are able to prove a more general result in this paper.

It should be remarked that sequentiality and guardedness are sufficient conditions, but neither is necessary. A counter example is given by the equation in (17). Now X is neither sequential nor guarded in $\overline{a}[X]$. We will argue informally that it has a unique solution. Suppose A is a solution. That is

$$A \simeq \overline{a}[A] \tag{18}$$

Consider the equivalence

$$B \approx \overline{a}[A] \tag{19}$$

What can be said about the behaviour of B? The following observations provide part of the answer:

- If $B \stackrel{\tau}{\longrightarrow} B'$ then $B' \approx \overline{a}[A]$ according to (19).
- If $B \stackrel{(\widetilde{x})\overline{a}B'}{\longrightarrow} B''$ then (i) $(\widetilde{x})(B' \mid B'') \approx A \approx \overline{a}[A]$ according to (18,19) and (ii) the B'' in $(\widetilde{x})(E[B'] \mid B'')$ can not perform any observable actions.
- B can only perform these two forms of actions.

Now suppose B participates in the action $E[B] \xrightarrow{\tau} F$ by performing the action $B \xrightarrow{(\widetilde{x})\overline{a}B'} B''$. Then F must be of the form $E'[(\widetilde{x})(E''[B'] \mid B'')]$. If B' is not in a fire-able position then neither B' nor B'' can have any observable actions. If B' is in a fire-able position then $E'[(\widetilde{x})(E''[B'] \mid B'')] \sim E'[E''[(\widetilde{x})(B' \mid B'')]]$. The implication of these facts is that $E[B]$ and $E[\mathbf{fix}X.\overline{a}[X]]$ can simulate each other by doing essentially the same actions. Hence $A \approx \mathbf{fix}X.\overline{a}[X]$ and $A \simeq \overline{a}[A] \simeq \mathbf{fix}X.\overline{a}[X]$.

It is worth remarking that the Fixpoint Theorem fails for $\approx$. Here is a counter example: Suppose $G[X]$ is $a.(b{+}X) + \tau.a.(b{+}X)$. Let A be $\mathbf{fix}X.a.(b{+}X)$ and B be $\mathbf{fix}X.\tau.a.(b{+}X)$. Clearly

$$\begin{aligned} A &\approx G[A] \equiv a.(b{+}A) + \tau.a.(b{+}A) \\ B &\approx G[B] \equiv a.(b{+}B) + \tau.a.(b{+}B) \end{aligned}$$

However $A \not\approx B$. This is because the only way to simulate the action sequence

$$B \xrightarrow{\tau} a.(b{+}B) \xrightarrow{a} b{+}B \xrightarrow{\tau} a.(b{+}B)$$

by A had to be

$$A \xrightarrow{a} b{+}A$$

But $b{+}A \not\approx a.(b{+}B)$ since the next action of $a.(b{+}B)$ can not be b.

7 Open Bisimulation

Sangiorgi proposed the open bisimulations in [14]. One advantage of the open bisimilarity is that it is more amenable to inductive analysis than the other bisimulation equivalences. In [16] the open bisimulations are criticized for being a little too strong. The counter example Sangiorgi and Walker presented in [16] is this: Intuitively $\overline{b}(z).(a(x){+}a(x).P{+}a(x).[x{=}z]P)$ should be bisimilar to $\overline{b}(z).(a(x){+}a(x).P)$. However they are not open bisimilar. This example has led Sangiorgi and Walker to introduce the quasi open bisimulations. This section takes a sketchy look at the (quasi) open bisimulations for $LHO\pi$.

In what follows, $\widetilde{z} \subseteq_f \mathcal{N}$ means that $\widetilde{z}$ is a finite subset of $\mathcal{N}$. The next definition introduces a crucial definition.

Definition 48. A substitution σ respects $\widetilde{z}$ if $(\forall x \in \widetilde{z}.\sigma(x) = x) \wedge (\forall x \notin \widetilde{z}.\sigma(x) \notin \widetilde{z})$.

A family $\{\mathcal{R}^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f\mathcal{N}}$ of relations on $\mathcal{LHPE}$ is closed under respectful substitution if, for each finite set $\widetilde{z}$ of names, the validity of $E\mathcal{R}^{\widetilde{z}}F$ implies the validity of $E\sigma\mathcal{R}^{\widetilde{z}}F\sigma$ for all substitutions σ that respect $\widetilde{z}$.

It will soon become clear that it is better to introduce the quasi open bisimulations before the open bisimulations.

Definition 49. A family $\{\mathcal{R}^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f\mathcal{N}}$ of symmetric relations on $\mathcal{LHPE}$ is a quasi open bisimulation if it is closed under the respectful substitution of names and the substitution of variables, and the following properties hold whenever $E\mathcal{R}^{\widetilde{z}}F$:
(i) If $E \xrightarrow{\tau} E'$ then F' exists such that $F \Longrightarrow F'\mathcal{R}^{\widetilde{z}}E'$.
(ii) If $E \xrightarrow{\alpha x} E'$ then F' exists such that $F \xRightarrow{\alpha x} F'\mathcal{R}^{\widetilde{z}}E'$.
(iii) If $E \xrightarrow{\overline{a}(x)} E'$ then F' exists such that $F \xRightarrow{\overline{a}(x)} F'\mathcal{R}^{\widetilde{z}x}E'$.
(iv) If $E \xrightarrow{aX} E'$ and $X \notin fv(E \mid F)$ then F' exists such that $F \xRightarrow{aX} F'\mathcal{R}^{\widetilde{z}}E'$.
(v) If $E \xrightarrow{(\widetilde{x})\overline{a}G} E'$ then some $\widetilde{x'}, H, F'$ exist such that $F \xRightarrow{(\widetilde{x'})\overline{a}H} F'$ and $(\widetilde{x})(K[G] \mid E')\ \mathcal{R}^{\widetilde{z}}\ (\widetilde{x'})(K[H] \mid F')$ for every process expression $K[X]$ such that $\widetilde{x}\widetilde{x'} \cap fn(K[X]) = \emptyset$.
The quasi open bisimilarity $\{\approx^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f\mathcal{N}}$ is the largest quasi open bisimulation. We write $\approx_{qo}$ for $\approx^{\emptyset}$.

Sangirogi and Walker have showed in [16] that $\approx_{qo}$ is equivalent to the open barbed bisimilarity for the first order π-calculus. In [4], Fu has proved that $\approx_{qo}$ is the same as the local bisimilarity for the first order π-calculus. It is routine to extend the result of [4] to the higher order scenario of this paper. Without further ado we state the coincidence result.

Theorem 50. *$E \approx^{\widetilde{x}} F$ if and only if $(\widetilde{x})(\widetilde{\overline{b}x} \mid E) \approx (\widetilde{x})(\widetilde{\overline{b}x} \mid F)$ for pairwise distinct fresh $\widetilde{b}$.*

Proof. Section 5 essentially proves that $\{\mathcal{R}^{\widetilde{x}}\}_{\widetilde{x}\subseteq_f\mathcal{N}}$, where

$$\mathcal{R}^{\widetilde{x}} \stackrel{\text{def}}{=} \left\{ (E,F) \;\middle|\; \begin{array}{l} (\widetilde{x})(\widetilde{\overline{b}x}\,|\,E) \approx (\widetilde{x})(\widetilde{\overline{b}x}\,|\,F) \\ \widetilde{b} \text{ pairwise distinct fresh} \end{array} \right\}$$

is a quasi open bisimulation. On the other hand

$$\mathcal{R} \stackrel{\text{def}}{=} \left\{ ((\widetilde{x})(\widetilde{\overline{b}x}\,|\,E),(\widetilde{x})(\widetilde{\overline{b}x}\,|\,F)) \;\middle|\; \begin{array}{l} E \approx^{\widetilde{x}} F \\ \widetilde{b} \text{ pairwise distinct fresh} \end{array} \right\}$$

is a bisimulation. See [4] for more details of the proof in the first order case. □

Corollary 51. *The equivalences $\approx_{qo}$ and $\approx$ are the same.*

In [8] the authors studied two equivalences for the first order π-calculus: the early equivalence $\approx_e$ and the late equivalence $\approx_l$. The difference between the two equivalences is to do with the simulations of the input actions. For the late equivalence the simulation is defined by the following clause:

> If $Q \approx_l P \xrightarrow{ax} P'$ for some $x \notin fn(P\,|\,Q)$ then Q' exists such that $Q \stackrel{ax}{\Longrightarrow} Q'$ and $Q'\{y/x\} \approx_l P'\{y/x\}$ for all y.

The distinction between the early and late equivalences persists through the open semantics. In [6] it is pointed out that in the presence of the mismatch operator the inclusion of the late open bisimilarity and the early open bisimilarity is strict. In the rest of this section we shall establish an interesting result that the late version of $\approx_{qo}$ is actually the open bisimilarity. The next definition is new.

Definition 52. A family $\{\mathcal{R}^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f\mathcal{N}}$ of symmetric relations on $\mathcal{LHPE}$ is a late quasi open bisimulation if it is closed under the respectful substitution of names and the substitution of variables, and the following properties hold whenever $E\mathcal{R}^{\widetilde{z}}F$:
(i) If $E \xrightarrow{\tau} E'$ then F' exists such that $F \Longrightarrow F'\mathcal{R}^{\widetilde{z}}E'$.
(ii) If $E \xrightarrow{\overline{a}x} E'$ then F' exists such that $F \stackrel{\overline{a}x}{\Longrightarrow} F'\mathcal{R}^{\widetilde{z}}E'$.
(iii) If $E \xrightarrow{ax} E'$ and $x \notin fn(E\,|\,F)$ then F' exists such that $F \stackrel{ax}{\Longrightarrow} F'$ and $F'\{y/x\}\mathcal{R}^{\widetilde{z}}E'\{y/x\}$ for all y.
(iv) If $E \xrightarrow{\overline{a}(x)} E'$ then F' exists such that $F \stackrel{\overline{a}(x)}{\Longrightarrow} F'\mathcal{R}^{\widetilde{z}x}E'$.
(v) If $E \xrightarrow{aX} E'$ and $X \notin fv(E\,|\,F)$ then F' exists such that $F \stackrel{aX}{\Longrightarrow} F'\mathcal{R}^{\widetilde{z}}E'$.
(vi) If $E \xrightarrow{(\widetilde{x})\overline{a}G} E'$ then some $\widetilde{x'}, H, F'$ exist such that $F \stackrel{(\widetilde{x'})\overline{a}H}{\Longrightarrow} F'$ and $(\widetilde{x})(K[G]\,|\,E')\ \mathcal{R}^{\widetilde{z}}\ (\widetilde{x'})(K[H]\,|\,F')$ for every process expression $K[X]$ such that $\widetilde{x}\widetilde{x'} \cap fn(K[X]) = \emptyset$.
The late quasi open bisimilarity $\{\approx_l^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f\mathcal{N}}$ is the largest late quasi open bisimulation. Write $\approx_{lqo}$ for $\approx_l^{\emptyset}$.

We leave out the algebraic investigations of the equivalence $\approx_{lqo}$. The interested reader is referred to [4]. Notice that $\approx_{lqo} \subseteq \approx_{qo}$ is strict since $\overline{b}(z).(a(x)+a(x).P+a(x).[x{=}z]P) \not\approx_{lqo} \overline{b}(z).(a(x)+a(x).P)$.

To continue we define the notion of distinction.

Definition 53. A finite binary relation D on $\mathcal{N}$ is a distinction if it is symmetric and irreflexive. The set of distinctions is denoted by $\mathcal{D}$.

A distinction postulates the perpetual difference among some names. A substitution σ respects $D \in \mathcal{D}$ if $\sigma(x) \neq \sigma(y)$ whenever $(x,y) \in D$. A distinction indexed family of relations $\{\mathcal{R}^D\}_{D\in\mathcal{D}}$ is closed under the respectful substitution if, for each $D \in \mathcal{D}$, $E\sigma\mathcal{R}^D F\sigma$ whenever $E\mathcal{R}^D F$ and σ respects D.

Definition 54. A family $\{\mathcal{R}^D\}_{D\in\mathcal{D}}$ of symmetric relations on $\mathcal{LHPE}$ is an open bisimulation if it is closed under the respectful substitution of names and the substitution of variables, and the following properties hold whenever $E\mathcal{R}^D F$:
(i) If $E \xrightarrow{\tau} E'$ then F' exists such that $F \Longrightarrow F'\mathcal{R}^D E'$.
(ii) If $E \xrightarrow{\alpha x} E'$ then F' exists such that $F \stackrel{\alpha x}{\Longrightarrow} F'\mathcal{R}^D E'$.
(iii) If $E \xrightarrow{\overline{a}(x)} E'$ then F' exists such that $F \stackrel{\overline{a}(x)}{\Longrightarrow} F'\mathcal{R}^{D'}E'$ where D' is $D \cup \{x\} \times (fn(E\,|\,F) \cup D) \cup (fn(E\,|\,F) \cup D) \times \{x\}$.
(iv) If $E \xrightarrow{aX} E'$ and $X \notin fv(E\,|\,F)$ then F' exists such that $F \stackrel{aX}{\Longrightarrow} F'\mathcal{R}^D E'$.
(v) If $E \xrightarrow{(\widetilde{x})\overline{a}G} E'$ then some $\widetilde{x'}, H, F'$ exist such that $F \stackrel{(\widetilde{x'})\overline{a}H}{\Longrightarrow} F'$ and $(\widetilde{x})(K[G]\,|\,E')\ \mathcal{R}^D\ (\widetilde{x'})(K[H]\,|\,F')$ for every process expression $K[X]$ such that $\widetilde{x}\widetilde{x'} \cap fn(K[X]) = \emptyset$.
The open bisimilarity $\{\approx^D\}_{D\in\mathcal{D}}$ is the largest open bisimulation. We write $\approx_{opn}$ for $\approx^{\emptyset}$.

Due to space limitation, we do not report on the investigation of the open bisimulations. The interested reader is referred to [14, 6].

Now suppose $F \approx^D E \xrightarrow{ax} E'$ for some fresh name x. By definition $F \stackrel{ax}{\Longrightarrow} F' \approx^D E'$ for some F' and $F'\{y/x\} \approx^D E'\{y/x\}$ for every name y whatsoever, since $\{y/x\}$ respects D. This property draws similarity to the clause (iii) of Definition 52. This observation leads immediately to the following theorem.

Theorem 55. *$E \approx_l^{\widetilde{z}} F$ if and only if $E \approx^D F$ where D is $\{(x,y),(y,x) \mid \widetilde{z} \ni x \neq y \in \widetilde{z} \cup fn(E \,|\, F)\}$.*

Proof. For each $D \in \mathcal{D}$, let $\mathcal{R}^D$ be defined by

$$\mathcal{R}^D \stackrel{\text{def}}{=} \left\{ (E,F) \;\middle|\; \begin{array}{l} E \approx_l^{\widetilde{z}} F \text{ and} \\ D = \{(x,y),(y,x) \mid \widetilde{z} \ni x \neq y \in \widetilde{z} \cup fn(E \,|\, F)\} \end{array} \right\} \cup \approx^D$$

The family $\{\mathcal{R}^D\}_{D \in \mathcal{D}}$ is an open bisimulation. For each finite set $\widetilde{z}$ of names, let $\mathcal{S}^{\widetilde{z}}$ be defined by

$$\mathcal{S}^{\widetilde{z}} \stackrel{\text{def}}{=} \left\{ (E,F) \;\middle|\; \begin{array}{l} E \approx^D F \text{ and} \\ D = \{(x,y),(y,x) \mid \widetilde{z} \ni x \neq y \in \widetilde{z} \cup fn(E \,|\, F)\} \end{array} \right\} \cup \approx^{\widetilde{z}}$$

The family $\{\mathcal{S}^{\widetilde{z}}\}_{\widetilde{z} \subseteq_f \mathcal{N}}$ is a late quasi open bisimulation. □

Corollary 56. *The equivalences $\approx_{lqo}$ and $\approx_{opn}$ coincide.*

In other words, the open bisimilarity is the late quasi open bisimilarity. This remark puts the quasi open bisimilarity and the open bisimilarity in a right order. It is our opinion that Definition 52 is an improvement of Definition 54. The former is slightly easier to deal with.

8 Head Normal Form

The finite process expressions, those containing no **fix**-operator, admit only finite computations. It makes sense to discuss algorithms that decide if any two given finite process expressions are equivalent. One standard approach to provide such an algorithm is to design an equational system that is sound and complete with respect to the congruence on the finite process expressions. In this section we propose an equational system for $LHO\pi$. An important law that underlines the whole approach is the well-known Expansion Law:

$$\begin{array}{rcl} E \,|\, F & = & \displaystyle\sum_{i \in I} \phi_i \lambda_i.(E_i \,|\, F) + \sum_{j \in J} \phi_j \lambda_j.(E \,|\, F_j) \\ & & + \displaystyle\sum_{i \in I, j \in J}^{\lambda_i = a_i(x),\ \lambda_j = \overline{b_j} y_j} \phi_i \phi_j [a_i{=}b_j] \tau.(E_i\{y_j/x\} \,|\, F_j) \\ & & + \displaystyle\sum_{i \in I, j \in J}^{\lambda_i = \overline{a_i} y_i,\ \lambda_j = b_j(x)} \phi_i \phi_j [a_i{=}b_j] \tau.(E_i \,|\, F_j\{y_i/x\}) \\ & & + \displaystyle\sum_{i \in I, j \in J}^{\lambda_i = a_i(X),\ \lambda_j = (\widetilde{x})\overline{b_j} H_j} \phi_i \phi_j [a_i{=}b_j] \tau.(\widetilde{x})(E_i\{H_j/X\} \,|\, F_j) \\ & & + \displaystyle\sum_{i \in I, j \in J}^{\lambda_i = (\widetilde{x})\overline{a_i} H_i,\ \lambda_j = b_j(X)} \phi_i \phi_j [a_i{=}b_j] \tau.(\widetilde{x})(E_i \,|\, F_j\{H_i/X\}) \end{array}$$

where E is $\sum_{i \in I} \phi_i \lambda_i.E_i$ and F is $\sum_{j \in J} \phi_j \lambda_j.F_j$.

In the first order π-calculus the axioms concerning the parallel composition is not necessary since every process can be reduced equationally to one without the parallel composition operator. The Expansion Law plays a crucial role in this procedure. In the higher order calculus however the parallel composition can not be removed. A simple instance is for example the process $a(X).(X \,|\, \overline{b}b)$. This fact also suggests that the laws for the higher order processes should be defined on process expressions rather than on

L1	$(x)\mathbf{0}$	$=$	$\mathbf{0}$	
L2	$(x)X$	$=$	X	
L3	$(x)\lambda.E$	$=$	$\lambda.(x)E$	if x does not appear in λ
L4	$(x)\lambda.E$	$=$	$\mathbf{0}$	if x is the subject name of λ
L5	$(x)(E \mid F)$	$=$	$E \mid (x)F$	if x is not free in E
L6	$(x)(y)E$	$=$	$(y)(x)E$	
L7	$(x)[y{=}z]E$	$=$	$[y{=}z](x)E$	if $x \notin \{y, z\}$
L8	$(x)[x{=}y]E$	$=$	$\mathbf{0}$	if $x \neq y$
L9	$(x)(E{+}F)$	$=$	$(x)E{+}(x)F$	
M1	ϕE	$=$	ψE	if $\phi \Leftrightarrow \psi$
M2	$[x{=}y]E$	$=$	$[x{=}y]E\{y/x\}$	
M3	$[x{=}y](E{+}F)$	$=$	$[x{=}y]E{+}[x{=}y]F$	
S1	$E{+}\mathbf{0}$	$=$	E	
S2	$E{+}F$	$=$	$F{+}E$	
S3	$E{+}(F{+}G)$	$=$	$(E{+}F){+}G$	
S4	$[x{=}y]E{+}E$	$=$	E	
T1	$\lambda.\tau.E$	$=$	$\lambda.E$	
T2	$E + \tau.E$	$=$	$\tau.E$	
T3	$\lambda.(E + \tau.F)$	$=$	$\lambda.(E + \tau.F) + \lambda.F$	
T4	$\tau.E$	$=$	$\tau.(E + [x{=}y]\tau.E)$	

Figure 1: Axioms for $LHO\pi$

processes. In Figure 1 the standard laws for the mobile processes are listed. Most of the axioms appear in [7, 8, 10]. The law T4 was proposed by Fu. In [6] it is proved that T4 is equivalent to

$$\tau.E = \tau.(E + \sum_{i \in I} \psi_i \tau.E) \tag{20}$$

More derived laws can also be found in [10, 6]. In this paper we shall use these derived laws without any further comment. Let AS stand for the set of axioms defined in Figure 1. We write $AS \vdash E = F$ if $E = F$ can be inferred from the axioms in AS and the axioms for congruence and equivalence.

One important role of an equational system is to rewrite a process to a normal form. For a higher order calculus there are two choices for the normal forms. One is that the normal forms are defined for process expressions. At the moment it is not clear how to reason about the normal forms in the presence of variables and composition operator. The other is to define head normal forms which ignore the structures of the process expressions underneath a prefix operation. Using the latter approach all processes can be rewritten to a head normal form.

Definition 57. A process expression E is a head normal form if E is of the form $\sum_{i \in I} \psi_i \lambda_i.E_i$.

Clearly a substitution does not change the shape of a head normal form.

Lemma 58. *If E is a head normal form then $E\sigma$ is a head normal form.*

In the following proofs we need a metrics measuring the structural complexity of the process expressions. The basic idea for such a metric function is that it should assign to the higher order prefixes bigger weight than the first order prefixes. The function defined below maps a process expression onto a binary tuple whose first value is the maximum number of the nested higher order prefixes and the second value

records the maximum number of the nested first order prefixes:

$$\begin{aligned}
d(\mathbf{0}) &\stackrel{\text{def}}{=} \langle 0,0\rangle \\
d(X) &\stackrel{\text{def}}{=} \langle 0,0\rangle \\
d(a(x).E) &\stackrel{\text{def}}{=} d(E)+\langle 0,1\rangle \\
d(\overline{a}x.E) &\stackrel{\text{def}}{=} d(E)+\langle 0,1\rangle \\
d(a(X).E) &\stackrel{\text{def}}{=} d(E)+\langle 1,0\rangle \\
d(\overline{a}H.E) &\stackrel{\text{def}}{=} d(H)+d(E)+\langle 1,0\rangle \\
d(E\,|\,F) &\stackrel{\text{def}}{=} d(E)+d(F) \\
d((x)E) &\stackrel{\text{def}}{=} d(E) \\
d([x{=}y]E) &\stackrel{\text{def}}{=} d(E) \\
d(E{+}F) &\stackrel{\text{def}}{=} max\{d(E),d(F)\}
\end{aligned}$$

where $\langle m_0,m_1\rangle+\langle n_0,n_1\rangle$ is defined to be $\langle m_0+n_0,m_1+n_1\rangle$. The partial order $\preceq$ is defined as follows: $\langle m_0,m_1\rangle \preceq \langle n_0,n_1\rangle$ if and only if $(m_0<n_0)\vee(m_0=n_0)\wedge(m_1\leq n_1)$. The following lemma says that the degree of a process is not increased when rewriting a process to a head normal form.

Lemma 59 (Normalization)**.** *For each finite linear higher order π-process P there is a head normal form P' such that the followings hold:*
(i) $AS\vdash P=P'$;
(ii) $d(P')\preceq d(P)$;
(iii) and for every substitution σ, $P\sigma\stackrel{\lambda}{\longrightarrow}P''$ if and only if $P'\sigma\stackrel{\lambda}{\longrightarrow}P''$.

Proof. The proof is carried out by structural induction:

- If P is of prefix form then P is already in head normal form.
- $P\equiv P_1\,|\,P_2$. By induction hypothesis there are head normal forms $\sum_{i\in I}\phi_i\lambda_i.P_1^i$ and $\sum_{j\in J}\psi_j\lambda_j.P_2^j$ such that the following properties hold:
  - $AS\vdash P_1=\sum_{i\in I}\phi_i\lambda_i.P_1^i$ and $d(\sum_{i\in I}\phi_i\lambda_i.P_1^i)\preceq d(P_1)$;
  - $AS\vdash P_2=\sum_{j\in J}\psi_j\lambda_j.P_2^j$ and $d(\sum_{j\in J}\psi_j\lambda_j.P_2^j)\preceq d(P_2)$;
  - $P_1\sigma$ and $(\sum_{i\in I}\phi_i\lambda_i.P_1^i)\sigma$ have the same derivatives;
  - $P_2\sigma$ and $(\sum_{j\in J}\psi_j\lambda_j.P_2^j)\sigma$ have the same derivatives.

  It follows that $d(\sum_{i\in I}\phi_i\lambda_i.P_1^i\,|\,\sum_{j\in J}\psi_j\lambda_j.P_2^j)\preceq d(P_1\,|\,P_2)$. We are done by applying the Expansion Law. Notice that the Expansion Law keeps the degree unchanged.

Other cases are all simple. □

9 Saturation

Normalization is about syntactical conversion of processes. Saturation on the other hand provides an equational characterization of the operational semantics. In order to discuss the saturation property for the higher order actions we need rules that allow us to derive equalities involving the higher order prefixes. In Figure 2, two rules are defined. In Concretion Rule the names $\widetilde{b},c,d$ must be distinct and fresh. The Concretion Rule generates an equality between two process expressions with higher order output prefixes. The Abstraction Rule enables us to reason about the process variables. The two rules are sound by Concretion Theorem (Theorem 24) and Abstraction Theorem (Theorem 21).

Let $AS_{LHO\pi}$ be $AS\cup\{CR,AR\}$. In $AS_{LHO\pi}$ we are able to prove the saturation properties for all forms of actions.

Concretion Rule

$$\frac{(\widetilde{x})(\widetilde{\overline{b}x}\,|\,(\widetilde{y})(\overline{c}.(H+d)\,|E)) = (\widetilde{x})(\widetilde{\overline{b}x}\,|\,(\widetilde{z})(\overline{c}.(G+d)\,|F))}{(\widetilde{x})(\widetilde{\overline{b}x}\,|\,(\widetilde{y})\overline{a}[H].E) = (\widetilde{x})(\widetilde{\overline{b}x}\,|\,(\widetilde{z})\overline{a}[G].F)}\ CR$$

Abstraction Rule

$$\frac{E\{c/X\} = F\{c/X\} \quad c \notin fn(E\,|\,F)}{E = F}\ AR$$

Figure 2: Higher Order Rules for $LHO\pi$

Lemma 60 (Saturation)**.** *Suppose σ is a substitution induced by ψ. The following properties hold:*
(i) If $E\sigma \stackrel{\tau}{\Longrightarrow} E'$ then $AS_{LHO\pi} \vdash E = E + \psi\tau.E'$.
(ii) If $E\sigma \stackrel{\overline{a}x}{\Longrightarrow} E'$ then $AS_{LHO\pi} \vdash E = E + \psi\overline{a}x.E'$.
(iii) If $E\sigma \stackrel{\overline{a}(x)}{\Longrightarrow} E'$ then $AS_{LHO\pi} \vdash E = E + \psi\overline{a}(x).E'$.
(iv) If $E\sigma \stackrel{ax}{\Longrightarrow} E'$, for $x \notin fn(E\sigma)$, then $AS_{LHO\pi} \vdash E = E + \psi a(x).E'$.
(v) If $E\sigma \stackrel{aX}{\Longrightarrow} E'$ for $X \notin fv(E)$ then $AS_{LHO\pi} \vdash E = E + \psi a(X).E'$.
(vi) If $E\sigma \stackrel{(\widetilde{y})\overline{a}H}{\Longrightarrow} E'$ then $AS_{LHO\pi} \vdash E = E + \psi(\widetilde{y})\overline{a}H.E'$.

Proof. By Lemma 4, Lemma 5 and the Abstraction Rule we may assume that E contains no variables.

(i) Suppose $P\sigma \stackrel{\tau}{\longrightarrow} P''\sigma \Longrightarrow P'$. By Lemma 59 some head normal form P_1 exists such that $AS_{LHO\pi} \vdash P = P_1$ and $P_1\sigma \stackrel{\tau}{\longrightarrow} P''\sigma$. Suppose that $P_1\sigma \stackrel{\tau}{\longrightarrow} P''\sigma$ is caused by the summand $\varphi\tau.P''$. Then clearly $\psi \Rightarrow \varphi$ and $AS_{LHO\pi} \vdash P_1 = P_1 + \varphi\tau.P''\sigma = P_1 + \psi\tau.P''\sigma = P_1 + \psi\tau.P''$. By induction hypothesis, $AS_{LHO\pi} \vdash P'' = P'' + \psi\tau.P'$. Therefore

$$\begin{aligned} P &= P_1 \\ &= P_1 + \psi\tau.P'' \\ &= P + \psi\tau.(P'' + \psi\tau.P') \\ &\stackrel{T2}{=} P + \psi\tau.P' \end{aligned}$$

(v) Suppose that $P\sigma \Longrightarrow P_1\sigma \stackrel{a[b]}{\longrightarrow} E_2\{b/X\}\sigma \Longrightarrow E'\{b/X\}$ for fresh b. By Lemma 59 there exist some head normal forms P_1', P_2' such that $AS_{LHO\pi} \vdash P_1 = P_1'$, $P_1'\sigma \stackrel{a[b]}{\longrightarrow} E_2\{b/X\}\sigma$,

$$AS_{LHO\pi} \quad \vdash \quad E_2\{b/X\} = P_2' \tag{21}$$

and $P_2'\sigma \Longrightarrow E'\{b/X\}$. By Lemma 58 both $P_1'\sigma$ and $P_2'\sigma$ are head normal forms. Hence $AS_{LHO\pi} \vdash P_1' = P_1' + \psi a(X).E_2$. Now

$$AS_{LHO\pi} \quad \vdash \quad P_2' = P_2' + \psi\tau.E'\{b/X\} \tag{22}$$

by (i). It follows from (21) and (22) that

$$AS_{LHO\pi} \vdash E_2\{b/X\} = E_2\{b/X\} + \psi\tau.E'\{b/X\}$$

Using Abstraction Rule one gets that $AS_{LHO\pi} \vdash E_2 = E_2 + \psi\tau.E'$. Putting all these together, one has

$$\begin{aligned} P &= P + \psi\tau.P_1 \\ &= P + \psi\tau.P_1' \\ &= P + \psi\tau.(P_1' + \psi a(X).E_2) \\ &= P + \psi\tau.(P_1' + \psi a(X).(E_2 + \psi\tau.E')) \\ &= P + \psi\tau.(P_1' + \psi a(X).(E_2 + \psi\tau.E')) + \psi a(X).E' \\ &= P + \psi a(X).E' \end{aligned}$$

$$E \,|\, (F + \lambda.G) \quad = \quad E \,|\, (F + \lambda.G) + \lambda.(E \,|\, G)$$

Figure 3: Saturation Axiom

We are done by observing that $E\sigma \stackrel{a[b]}{\Longrightarrow} E'\{b/X\}$ for fresh b if and only if $E\sigma \stackrel{aX}{\Longrightarrow} E'$ for $X \notin fv(E)$.

(vi) Suppose that $P\sigma \Longrightarrow P_1\sigma \stackrel{(\widetilde{y})\overline{a}A}{\longrightarrow} P_2\sigma \Longrightarrow P'$. By Lemma 59 there are head normal forms P_1', P_2' such that $AS_{LHO\pi} \vdash P_1 = P_1'$, $P_1'\sigma \stackrel{(\widetilde{y})\overline{a}A}{\longrightarrow} P_2\sigma$, $AS_{LHO\pi} \vdash P_2 = P_2'$ and $P_2'\sigma \Longrightarrow P'$. Then

$$\begin{aligned} P &= P + \psi\tau.P_1 \\ &= P + \psi\tau.P_1' \\ &= P + \psi\tau.(P_1' + \psi(\widetilde{y})\overline{a}A.P_2) \\ &= P + \psi\tau.(P_1' + \psi(\widetilde{y})\overline{a}A.P_2') \\ &= P + \psi(\widetilde{y})\overline{a}A.P_2' \\ &= P + \psi(\widetilde{y})\overline{a}A.(P_2' + \psi\tau.P') \\ &= P + \psi(\widetilde{y})\overline{a}A.P' \end{aligned}$$

The rest of the proof is the same as the proof in the first order π-calculus. The additional care is that one need to convert a process to head normal form after each action. □

The above proof makes use of Normalization Lemma and Abstraction Rule. One way to bypass the Normalization Lemma is to use the Saturation Axiom defined in Figure 3 using the Expansion Law. It is not difficult to see that Saturation Axiom is derivable in $AS_{LHO\pi}$. It should be a routine to check that the saturation properties can be established in $AS \cup \{\text{Saturation Axiom}\}$.

10 Completeness

$LHO\pi$ is the first order π-calculus plus the linear higher order communication mechanism. The main result of this section can be interpreted in the following way:

> The Abstraction Rule and the Concretion Rule promote a complete system for the first order π-calculus to a complete system for $LHO\pi$.

To make this point we will carry out a completeness proof for $\simeq_{opn}$ on the finite $LHO\pi$-process expressions. The reason we opt for $\simeq_{opn}$ instead of $\simeq$ is that the former has a simpler completeness proof for the first order fragment of the calculus.

A completeness proof for $\simeq_{opn}$ calls for more machinery than the proof of the Saturation Lemma. The standard congruence rules no longer suffice. We should be able to derive say $(x)(\overline{b}x \,|\, \overline{a}z.E) = (x)(\overline{b}x \,|\, \overline{a}z.F)$ from $(x)(\overline{b}x \,|\, E) = (x)(\overline{b}x \,|\, F)$. This inference is not supported by the congruence rule for prefix. We need some local congruence rules! In Figure 4 some local congruence rules are formulated. In these rules the names $\widetilde{b}, b'$ are fresh. Notice that the local congruence rules imply the global congruence rules. Let the equational system $AS^{opn}_{LHO\pi}$ be obtained from $AS_{LHO\pi}$ by adding the rules defined in Figure 4.

According to Theorem 40, the rules PR, LR, MR and SR are sound for $\simeq$, as well as $\simeq_{opn}$. The rule IR is valid for $\simeq_{opn}$, but fails for $\simeq$.

Lemma 61. *The following rules are admissible in* $AS^{opn}_{LHO\pi}$*:*
(i) If $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, E) = (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F)$ *then* $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, \overline{a}.E) = (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \overline{a}.F)$ *provided that* $a \neq b$*;*
(ii) If $(\widetilde{x})(z)(\widetilde{\overline{b}x} \,|\, \overline{b'}z \,|\, E) = (\widetilde{x})(z)(\widetilde{\overline{b}x} \,|\, \overline{b'}z \,|\, F)$ *then* $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, E) = (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F)$ *provided that* $z \notin fn(E \,|\, F)$*.*

We now turn to the completeness proof. Such a proof is usually stratified in two steps. First a weaker form of completeness, the promotion lemma, is established. Second the completeness is proved using the promotion lemma. The promotion lemma we present in this paper is a localized version of the usual promotion lemma.

Input Rule

$$\frac{\forall y \in \widetilde{x}z.(\widetilde{x})(\widetilde{\overline{b}x} \mid E\{y/z\}) = (\widetilde{x})(\widetilde{\overline{b}x} \mid F\{y/z\})}{(\widetilde{x})(\widetilde{\overline{b}x} \mid a(z).E) = (\widetilde{x})(\widetilde{\overline{b}x} \mid a(z).F)}\ IR$$

Prefix Rule

$$\frac{(\widetilde{x})(\widetilde{\overline{b}x} \mid E) = (\widetilde{x})(\widetilde{\overline{b}x} \mid F) \quad \lambda = \overline{a}z \vee \lambda = \overline{a}H \vee \lambda = a(X)}{(\widetilde{x})(\widetilde{\overline{b}x} \mid \lambda.E) = (\widetilde{x})(\widetilde{\overline{b}x} \mid \lambda.F)}\ PR$$

Localization Rule

$$\frac{(\widetilde{x})(z)(\widetilde{\overline{b}x} \mid \overline{b'}z \mid E) = (\widetilde{x})(z)(\widetilde{\overline{b}x} \mid \overline{b'}z \mid F)}{(\widetilde{x})(\widetilde{\overline{b}x} \mid (z)E) = (\widetilde{x})(\widetilde{\overline{b}x} \mid (z)F)}\ LR$$

Match Rule

$$\frac{(\widetilde{x})(\widetilde{\overline{b}x} \mid E) = (\widetilde{x})(\widetilde{\overline{b}x} \mid F)}{(\widetilde{x})(\widetilde{\overline{b}x} \mid [y{=}z]E) = (\widetilde{x})(\widetilde{\overline{b}x} \mid [y{=}z]F)}\ MR$$

Summation Rule

$$\frac{(\widetilde{x})(\widetilde{\overline{b}x} \mid E) = (\widetilde{x})(\widetilde{\overline{b}x} \mid F) \qquad (\widetilde{x})(\widetilde{\overline{b}x} \mid G) = (\widetilde{x})(\widetilde{\overline{b}x} \mid H)}{(\widetilde{x})(\widetilde{\overline{b}x} \mid (E{+}G)) = (\widetilde{x})(\widetilde{\overline{b}x} \mid (F{+}H))}\ SR$$

Figure 4: Local Open Congruence Rules

Lemma 62 (Promotion)**.** *Suppose $b_1, \ldots, b_n$ are pairwise distinct fresh names. If*

$$(x_1 \ldots x_n)(\overline{b_1}x_1 \mid \ldots \mid \overline{b_n}x_n \mid E) \approx_{opn} (x_1 \ldots x_n)(\overline{b_1}x_1 \mid \ldots \mid \overline{b_n}x_n \mid F)$$

then $AS^{opn}_{LHO\pi} \vdash (x_1 \ldots x_n)(\overline{b_1}x_1 \mid \ldots \mid \overline{b_n}x_n \mid \tau.E) = (x_1 \ldots x_n)(\overline{b_1}x_1 \mid \ldots \mid \overline{b_n}x_n \mid \tau.F)$.

Proof. By Abstraction Theorem and Abstraction Rule, we only need to consider processes. Suppose that

$$(x_1 \ldots x_n)(\overline{b_1}x_1 \mid \ldots \mid \overline{b_n}x_n \mid P) \approx_{opn} (x_1 \ldots x_n)(\overline{b_1}x_1 \mid \ldots \mid \overline{b_n}x_n \mid Q)$$

and that $P \equiv \sum_{i\in I} \varphi_i\lambda_i.E_i$ and $Q \equiv \sum_{j\in J} \psi_j\lambda_j.F_j$. Using the L-laws and their derivatives we may assume that x does not appear in $\bigvee_{i\in I} \varphi_i \vee \bigvee_{j\in J} \psi_j$. We establish the result by induction on the sum of the degrees of P and Q. Let σ be induced by φ_i. There are quite a few cases. These cases can be classified into two groups. The first group consists of all such summands $\varphi_i\lambda_i.E_i$ of P that the subject name of λ_i is *not* in $\widetilde{x}$.

- $\lambda_i = \tau$. The action $(\widetilde{x})(\widetilde{\overline{b}x} \mid P\sigma) \stackrel{\tau}{\Longrightarrow} (\widetilde{x})(\widetilde{\overline{b}x} \mid E_i\sigma)$ can be simulated by $Q\sigma$ in two fashions:
  - $(\widetilde{x})(\widetilde{\overline{b}x} \mid Q\sigma) \stackrel{\tau}{\Longrightarrow} (\widetilde{x})(\widetilde{\overline{b}x} \mid F_i'\sigma) \approx_{opn} (\widetilde{x})(\widetilde{\overline{b}x} \mid E_i\sigma)$. By induction hypothesis one has that $AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x} \mid \tau.E_i\sigma) = (\widetilde{x})(\widetilde{\overline{b}x} \mid \tau.F_i'\sigma)$. So $AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x} \mid \varphi_i\tau.E_i\sigma) = (\widetilde{x})(\widetilde{\overline{b}x} \mid \varphi_i\tau.F_i'\sigma)$ by Match Rule. Thus $AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x} \mid \varphi_i\tau.E_i) = (\widetilde{x})(\widetilde{\overline{b}x} \mid \varphi_i\tau.F_i')$. Moreover $(\widetilde{x})(\widetilde{\overline{b}x} \mid Q\sigma) \stackrel{\tau}{\Longrightarrow} (\widetilde{x})(\widetilde{\overline{b}x} \mid F_i'\sigma)$ is obviously due to $Q\sigma \stackrel{\tau}{\Longrightarrow} F_i'\sigma$. Thus $AS^{opn}_{LHO\pi} \vdash Q = Q + \varphi_i\tau.F_i'\sigma = Q + \varphi_i\tau.F_i'$ by Saturation Lemma.
  - The simulation is vacuous. In this case $AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x} \mid \varphi_i\tau.E_i) = (\widetilde{x})(\widetilde{\overline{b}x} \mid \varphi_i\tau.Q)$.
- $\lambda_i = a_i(z)$. First observe that following correspondences hold by Corollary 56:

  $$(\widetilde{x})(\widetilde{\overline{b}x} \mid P\sigma) \approx_{opn} (\widetilde{x})(\widetilde{\overline{b}x} \mid Q\sigma) \text{ iff } (\widetilde{x})(\widetilde{\overline{b}x} \mid P\sigma) \approx_{lqo} (\widetilde{x})(\widetilde{\overline{b}x} \mid Q\sigma) \text{ iff } P\sigma \approx_l^{\widetilde{x}} Q\sigma.$$

  Therefore the action $(\widetilde{x})(\widetilde{\overline{b}x} \mid P\sigma) \stackrel{\sigma(a_i)z}{\longrightarrow} (\widetilde{x})(\widetilde{\overline{b}x} \mid E_i\sigma)$ is simulated by $(\widetilde{x})(\widetilde{\overline{b}x} \mid Q\sigma) \stackrel{\sigma(a_i)z}{\Longrightarrow} (\widetilde{x})(\widetilde{\overline{b}x} \mid F_i'\sigma)$ such that $(\widetilde{x})(\widetilde{\overline{b}x} \mid F_i'\sigma\{y/z\}) \approx_{opn} (\widetilde{x})(\widetilde{\overline{b}x} \mid E_i\sigma\{y/z\})$ for all $y \in \widetilde{x}z$. By induction hypothesis one

has that $AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \tau.E_i\sigma\{y/z\}) = (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \tau.F'_i\sigma\{y/z\})$. So $AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x} \,|\, a(z).E_i\sigma) = (\widetilde{x})(\widetilde{\overline{b}x} \,|\, a(z).F'_i\sigma)$ by Input Rule. Hence

$$AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \varphi_i a(z).E_i) = (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \varphi_i a(z).F'_i)$$

by Match Rule. By Saturation Lemma one also has $AS^{opn}_{LHO\pi} \vdash Q = Q + \varphi_i a_i(z).F'_i$.

- $\lambda_i = \overline{a_i}z_i$ and $z_i \notin \widetilde{x}$. This case is similar to the previous one.
- $\lambda_i = \overline{a_i}x_i$. Let $\widetilde{x'}$ be $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$. Now the action $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, P\sigma) \stackrel{\sigma(a_i)(x_i)}{\longrightarrow} (\widetilde{x'})(\widetilde{\overline{b}x} \,|\, E_i\sigma)$ can be simulated by $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, Q\sigma) \stackrel{\sigma(a_i)(x_i)}{\Longrightarrow} (\widetilde{x'})(\widetilde{\overline{b}x} \,|\, F'_i\sigma)$ for some F'_i such that

$$(x_i)(\overline{b'}x_i \,|\, (\widetilde{x'})(\widetilde{\overline{b}x} \,|\, F'_i\sigma)) \approx_{opn} (x_i)(\overline{b'}x_i \,|\, (\widetilde{x'})(\widetilde{\overline{b}x} \,|\, E_i\sigma))$$

for some fresh b'. It follows from (iii) of Lemma 17 that $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, E_i\sigma) \approx_{opn} (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F'_i\sigma)$. By induction hypothesis one has that $AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \tau.E_i\sigma) = (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \tau.F'_i\sigma)$. So

$$AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \varphi_i\overline{a_i}x_i.E_i) = (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \varphi_i\overline{a_i}x_i.F'_i)$$

by Prefix and Match Rules. Again $AS^{opn}_{LHO\pi} \vdash Q = Q + \varphi_i\overline{a_i}x_i.F'_i$ by Saturation Lemma.

- $\lambda_i = \overline{a_i}(z)$ and $z \notin \widetilde{x}$. In this case $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, P\sigma) \stackrel{\overline{\sigma(a_i)}(z)}{\longrightarrow} (\widetilde{x})(\widetilde{\overline{b}x} \,|\, E_i\sigma)$ is simulated by the action $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, Q\sigma) \stackrel{\overline{\sigma(a_i)}(z)}{\Longrightarrow} (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F'_i\sigma)$ such that $(z)(\overline{b'}z \,|\, (\widetilde{x})(\widetilde{\overline{b}x} \,|\, E_i\sigma)) \approx_{opn} (z)(\overline{b'}z \,|\, (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F'_i\sigma))$ for a fresh name b'. The equivalence is the same as $(\widetilde{x})(z)(\widetilde{\overline{b}x} \,|\, \overline{b'}z \,|\, E_i\sigma) \approx_{opn} (\widetilde{x})(z)(\widetilde{\overline{b}x} \,|\, \overline{b'}z \,|\, F'_i\sigma)$. By induction hypothesis, $AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(z)(\widetilde{\overline{b}x} \,|\, \overline{b'}z \,|\, \tau.E_i\sigma) = (\widetilde{x})(z)(\widetilde{\overline{b}x} \,|\, \overline{b'}z \,|\, \tau.F'_i\sigma)$. Therefore

$$AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(z)(\widetilde{\overline{b}x} \,|\, \overline{b'}z \,|\, \varphi_i\overline{a_i}z.E_i) = (\widetilde{x})(z)(\widetilde{\overline{b}x} \,|\, \overline{b'}z \,|\, \varphi_i\overline{a_i}z.F'_i)$$

by Prefix Rule, Match Rule and T1. Then $AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \varphi_i\overline{a_i}(z).E_i) = (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \varphi_i\overline{a_i}(z).F'_i)$ by Localization Rule. According to Saturation Lemma, $AS^{opn}_{LHO\pi} \vdash Q = Q + \varphi_i\overline{a_i}(z).F'_i$.

- $\lambda_i = a_i(X)$. Then $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, P\sigma) \stackrel{\sigma(a_i)X}{\longrightarrow} (\widetilde{x})(\widetilde{\overline{b}x} \,|\, E_i\sigma)$. Some F'_i must exist such that $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, Q\sigma) \stackrel{\sigma(a_i)X}{\Longrightarrow} (\widetilde{x})(\widetilde{\overline{b}x} \,|\, F'_i\sigma) \approx_{opn} (\widetilde{x})(\widetilde{\overline{b}x} \,|\, E_i\sigma)$. Let v be fresh. Then $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, F'_i\sigma\{v/X\}) \approx_{opn} (\widetilde{x})(\widetilde{\overline{b}x} \,|\, E_i\sigma\{v/X\})$. It should be clear that the sum of the degrees of $E_i\sigma\{v/X\}$ and $F'_i\sigma\{v/X\}$ is strictly smaller than that of the degrees of P and Q since one higher order input prefix has been removed. By induction hypothesis $AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \tau.E_i\sigma\{v/X\}) = (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \tau.F'_i\sigma\{v/X\})$. By Abstraction Rule

$$AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \tau.E_i\sigma) = (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \tau.F'_i\sigma)$$

By Prefix Rule and Match Rule,

$$AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \varphi_i a_i(X).E_i) = (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \varphi_i a_i(X).F'_i)$$

It is obvious that $Q\sigma \stackrel{\sigma(a_i)X}{\Longrightarrow} F'_i\sigma$. Thus $AS^{opn}_{LHO\pi} \vdash Q = Q + \varphi_i a_i(X).F'_i$ by Saturation Lemma.

- $\lambda_i = (\widetilde{y})\overline{a_i}[A_i]$. Then $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, P\sigma) \stackrel{(\widetilde{x_1})(\widetilde{y})\overline{\sigma(a_i)}[A_i\sigma]}{\longrightarrow} (\widetilde{x_2})(\widetilde{\overline{b}x} \,|\, E_i\sigma)$, where $\widetilde{x_1}\widetilde{x_2} = \widetilde{x}$. Let c, d be fresh. By definition $\widetilde{z}, B_i, F'_i$ exist such that $(\widetilde{x})(\widetilde{\overline{b}x} \,|\, Q\sigma) \stackrel{(\widetilde{x'_1})(\widetilde{z})\overline{\sigma(a_i)}[B_i\sigma]}{\Longrightarrow} (\widetilde{x'_2})(\widetilde{\overline{b}x} \,|\, F'_i\sigma)$, where $\widetilde{x'_1}\widetilde{x'_2} = \widetilde{x}$, and the following equivalences hold:

$$\begin{aligned}
(\widetilde{x_1})(\widetilde{y})(\overline{c}.(A_i\sigma{+}d) \,|\, (\widetilde{x_2})(\widetilde{\overline{b}x} \,|\, E_i\sigma)) &\approx_{opn} (\widetilde{x'_1})(\widetilde{z})(\overline{c}.(B_i\sigma{+}d) \,|\, (\widetilde{x'_2})(\widetilde{\overline{b}x} \,|\, F'_i\sigma)) \\
(\widetilde{x_1})(\widetilde{y})((A_i\sigma{+}d) \,|\, (\widetilde{x_2})(\widetilde{\overline{b}x} \,|\, E_i\sigma)) &\approx_{opn} (\widetilde{x'_1})(\widetilde{z})((B_i\sigma{+}d) \,|\, (\widetilde{x'_2})(\widetilde{\overline{b}x} \,|\, F'_i\sigma))
\end{aligned}$$

These equivalences are the same as the following ones:

$$\begin{array}{lll}(\widetilde{x})(\widetilde{\overline{b}x}\,|\,(\widetilde{y})(\overline{c}.(A_i\sigma{+}d)\,|\,E_i\sigma)) & \approx_{opn} & (\widetilde{x})(\widetilde{\overline{b}x}\,|\,(\widetilde{z})((\overline{c}.(B_i\sigma{+}d)\,|\,F_i'\sigma))\\ (\widetilde{x})(\widetilde{\overline{b}x}\,|\,(\widetilde{y})((A_i\sigma{+}d)\,|\,\tau.E_i\sigma)) & \approx_{opn} & (\widetilde{x})(\widetilde{\overline{b}x}\,|\,(\widetilde{z})((B_i\sigma{+}d)\,|\,\tau.F_i'\sigma))\end{array}$$

Now the sum of the degrees of $(\widetilde{y})(\overline{c}.(A_i\sigma{+}d)\,|\,E_i\sigma)$ and $(\widetilde{z})(\overline{c}.(B_i\sigma{+}d)\,|\,F_i'\sigma)$, as well as that of $(\widetilde{y})((A_i\sigma{+}d)\,|\,\tau.E_i\sigma)$ and $(\widetilde{z})((B_i\sigma{+}d)\,|\,\tau.F_i'\sigma)$, is strictly less than that of P and Q since one higher order output prefix is replaced by a first order prefix. By applying induction hypothesis to the above equivalences, one gets

$$(\widetilde{x})(\widetilde{\overline{b}x}\,|\,\tau.(\widetilde{y})(\overline{c}.(A_i\sigma{+}d)\,|\,E_i\sigma)) \;=\; (\widetilde{x})(\widetilde{\overline{b}x}\,|\,\tau.(\widetilde{z})(\overline{c}.(B_i\sigma{+}d)\,|\,F_i'\sigma)) \tag{23}$$

$$(\widetilde{x})(\widetilde{\overline{b}x}\,|\,\tau.(\widetilde{y})((A_i\sigma{+}d)\,|\,\tau.E_i\sigma)) \;=\; (\widetilde{x})(\widetilde{\overline{b}x}\,|\,\tau.(\widetilde{z})((B_i\sigma{+}d)\,|\,\tau.F_i'\sigma)) \tag{24}$$

Now applying (i) of Lemma 61 to (24) gives us the following equality:

$$(\widetilde{x})(\widetilde{\overline{b}x}\,|\,\overline{c}.(\widetilde{y})((A_i\sigma{+}d)\,|\,\tau.E_i\sigma)) \;=\; (\widetilde{x})(\widetilde{\overline{b}x}\,|\,\overline{c}.(\widetilde{z})((B_i\sigma{+}d)\,|\,\tau.F_i'\sigma)) \tag{25}$$

Using the Summation Rule and the Expansion Law, one gets from (23) and (25) that

$$AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x}\,|\,(\widetilde{y})(\overline{c}.(A_i\sigma{+}d)\,|\,\tau.E_i\sigma)) = (\widetilde{x})(\widetilde{\overline{b}x}\,|\,(\widetilde{z})(\overline{c}.(B_i\sigma{+}d)\,|\,\tau.F_i'\sigma)) \tag{26}$$

Using the Concretion Rule one derives from (26) that

$$AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x}\,|\,(\widetilde{y})\overline{\sigma(a_i)}[A_i\sigma].E_i\sigma) = (\widetilde{x})(\widetilde{\overline{b}x}\,|\,(\widetilde{z})\overline{\sigma(a_i)}[B_i\sigma].F_i'\sigma)$$

Hence by Match Rule

$$AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x}\,|\,\varphi_i(\widetilde{y})\overline{a_i}[A_i].E_i) = (\widetilde{x})(\widetilde{\overline{b}x}\,|\,\varphi_i(\widetilde{z})\overline{a_i}[B_i].F_i')$$

It follows from $Q\sigma \stackrel{(\widetilde{z})\overline{\sigma(a_i)}[B_i\sigma]}{\Longrightarrow} F_i'\sigma$ that $AS^{opn}_{LHO\pi} \vdash Q = Q + \varphi_i(\widetilde{y})\overline{a_i}[A_i].F_i'$.

The second group consists of all such summands $\varphi_i\lambda_i.E_i$ of P that the subject name of λ_i *is* one of $\widetilde{x}$. The proofs are basically the same as those for the corresponding ones in the first group. We take a look at one case:

- $\lambda_i = (\widetilde{y})\overline{x_i}[A_i]$. Write $\widetilde{x'}$ for $x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n$ and $\widetilde{b'}$ for $b_1,\ldots,b_{i-1},b_{i+1},\ldots,b_n$. Let c,d,e be fresh. Define O to be the process $e + x_i(X).(\overline{c}.(X{+}d)\,|\,\overline{b_i}x_i)$. Then

$$\begin{array}{rcl} b_i(x_i).O\,|\,(\widetilde{x})(\widetilde{\overline{b}x}\,|\,P\sigma) & \stackrel{\tau}{\longrightarrow} & (x_i)(O\,|\,(\widetilde{x'})(\widetilde{\overline{b'}x'}\,|\,P\sigma))\\ & \stackrel{\tau}{\longrightarrow} & (x_i)(\widetilde{y})(\overline{c}.(A_i\sigma{+}d)\,|\,\overline{b_i}x_i\,|\,(\widetilde{x'})(\widetilde{\overline{b'}x'}\,|\,E_i\sigma))\\ & \sim & (\widetilde{x})(\widetilde{\overline{b}x}\,|\,(\widetilde{y})(\overline{c}.(A_i\sigma{+}d)\,|\,E_i\sigma))\end{array}$$

The above reduction has to be matched up by $b_i(x_i).(x_i(X).\overline{c}.(X{+}d) + e)\,|\,(\widetilde{x})(\widetilde{\overline{b}x}\,|\,Q\sigma)$. So by definition $\widetilde{z}, B_i, F_i'$ exist such that

$$\begin{array}{rcl} b_i(x_i).O\,|\,(\widetilde{x})(\widetilde{\overline{b}x}\,|\,Q\sigma) & \stackrel{\tau}{\Longrightarrow} & (x_i)(O\,|\,(\widetilde{x'})(\widetilde{\overline{b'}x'}\,|\,Q'\sigma))\\ & \stackrel{\tau}{\Longrightarrow} & (x_i)(\widetilde{z})(\overline{c}.(B_i\sigma{+}d)\,|\,\overline{b_i}x_i\,|\,(\widetilde{x'})(\widetilde{\overline{b'}x'}\,|\,F_i'\sigma))\\ & \sim & (\widetilde{x})(\widetilde{\overline{b}x}\,|\,(\widetilde{z})(\overline{c}.(B_i\sigma{+}d)\,|\,F_i'\sigma))\\ & \approx_{opn} & (\widetilde{x})(\widetilde{\overline{b}x}\,|\,(\widetilde{y})(\overline{c}.(A_i\sigma{+}d)\,|\,E_i\sigma))\end{array}$$

Clearly $Q\sigma \stackrel{(\widetilde{z})\overline{x_i}[B_i\sigma]}{\Longrightarrow} F_i'\sigma$. The rest of the argument is as before.

In summary what we have established is this: There are disjoint subsets I_0, I_1 of I such that $I_0 \cup I_1 = I$. For each $i \in I_1$ there exists F_i' such that

$$AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \varphi_i\lambda_i.E_i) = (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \varphi_i\lambda_i.F_i') \tag{27}$$

and

$$AS^{opn}_{LHO\pi} \vdash Q = Q + \varphi_i\lambda_i.F_i' \tag{28}$$

For each $i \in I_0$, $\lambda_i = \tau$ and

$$AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \varphi_i\tau.E_i) = (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \varphi_i\tau.Q) \tag{29}$$

Symmetrically there are disjoint subsets J_0, J_1 of J such that $J_0 \cup J_1 = J$. For each $j \in J_1$ there exists E_j' such that

$$AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \psi_j\lambda_j.F_j) = (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \psi_j\lambda_j.E_j') \tag{30}$$

and

$$AS^{opn}_{LHO\pi} \vdash P = P + \psi_j\lambda_j.E_j' \tag{31}$$

For each $j \in J_0$, $\lambda_j = \tau$ and

$$AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \psi_j\tau.F_j) = (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \psi_i\tau.P) \tag{32}$$

Putting all the equations together, one has by the Summation Rule

$$\begin{aligned}(\widetilde{x})(\widetilde{\overline{b}x} \,|\, P + Q) &\overset{(27,29)}{=} (\widetilde{x})\left(\widetilde{\overline{b}x} \;\middle|\; (\sum_{i\in I_0} \varphi_i\tau.Q + \sum_{i\in I_1} \varphi_i\lambda_i.F_i' + Q)\right)\\ &\overset{(28)}{=} (\widetilde{x})\left(\widetilde{\overline{b}x} \;\middle|\; (\sum_{i\in I_0} \varphi_i\tau.Q + Q)\right)\end{aligned}$$

Similarly

$$(\widetilde{x})(\widetilde{\overline{b}x} \,|\, Q + P) \overset{(30,31,32)}{=} (\widetilde{x})\left(\widetilde{\overline{b}x} \;\middle|\; (\sum_{j\in J_0} \psi_j\tau.P + P)\right)$$

Therefore

$$(\widetilde{x})\left(\widetilde{\overline{b}x} \;\middle|\; (\sum_{j\in J_0} \psi_j\tau.P + P)\right) = (\widetilde{x})\left(\widetilde{\overline{b}x} \;\middle|\; (\sum_{i\in I_0} \varphi_i\tau.Q + Q)\right)$$

It follows from the Prefix Rule that

$$(\widetilde{x})\left(\widetilde{\overline{b}x} \;\middle|\; \tau.(\sum_{j\in J_0} \psi_j\tau.P + P)\right) = (\widetilde{x})\left(\widetilde{\overline{b}x} \;\middle|\; \tau.(\sum_{i\in I_0} \varphi_i\tau.Q + Q)\right)$$

We conclude by (20) that $AS^{opn}_{LHO\pi} \vdash (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \tau.P) = (\widetilde{x})(\widetilde{\overline{b}x} \,|\, \tau.Q)$. □

It is well-known that the proof of the Completeness Theorem is a minor modification of the proof of the Promotion Lemma. We omit the routine proof.

Theorem 63 (Completeness Theorem)**.** *Suppose that neither E nor F contains any* **fix***-operators. Then $E \simeq_{opn} F$ if and only if $AS^{opn}_{LHO\pi} \vdash E = F$.*

11 Algorithm

The Completeness Theorem lays down a foundation for an algorithm of checking the equivalence of two finite linear higher order π-process expressions. Figure 5 describes an algorithm $ECHECK$ that returns true if the two inputs are congruent process expressions in $\mathcal{FLHPE}$, the set of the finite linear higher order π-process expressions, and returns false otherwise. Due to the recursive nature of the algorithm, the two input process expressions are assumed to take the form $(\widetilde{x})(\widetilde{\widetilde{b}x} \mid E)$, which is general enough since the length of $\widetilde{b}$ could be zero. In the description of the algorithm we have used the terminology "saturated head normal form" whose definition is given below.

Definition 64. A head normal form $\sum_{i\in I}\varphi_i\lambda_i.E_i$ is saturated if, for each $i \in I$, whenever

$$(\sum_{i\in I}\varphi_i\lambda_i.E_i)\sigma_{\varphi_i} \stackrel{\lambda}{\Longrightarrow} E'$$

then there is a summand $\varphi_{i'}\lambda_{i'}.E_{i'}$ of $\sum_{i\in I}\varphi_i\lambda_i.E_i$ such that $\varphi_{i'} \Leftrightarrow \varphi_i$, $\lambda_{i'}\sigma_{\varphi_i} = \lambda$ and $E_{i'}\sigma_{\varphi_i} \equiv E'$.

Using Lemma 59 and Lemma 60 one can easily prove the following result.

Lemma 65. *For each finite linear higher order π-process P there is a saturated head normal form $\sum_{i\in I}\varphi_i\lambda_i.E_i$ such that the followings hold:*
(i) $AS_{LHO\pi} \vdash P = \sum_{i\in I}\varphi_i\lambda_i.E_i$;
(ii) $d(\sum_{i\in I}\varphi_i\lambda_i.E_i) \preceq d(P)$;
(iii) and for every $i \in I$, $P\sigma_{\varphi_i} \stackrel{\lambda}{\Longrightarrow} P''$ if and only if $(\sum_{i\in I}\varphi_i\lambda_i.E_i)\sigma_{\varphi_i} \stackrel{\lambda}{\longrightarrow} P''$.

The step 4 of the algorithm is validated by the above lemma. It is straightforward to prove by induction that the algorithm terminates.

12 *LHOCCS*

We could repeat the above investigation for $LHOCCS$. In this section we state the main definitions and results. The proofs of these results are omitted since they are the simpler version of the proofs in $LHO\pi$. Formally the set of $LHOCCS$ expressions $\mathcal{LHCE}$ is generated by the following BNF:

$$\begin{array}{rcll} E & := & X & \\ & & a.E & \\ & & \overline{a}.E & \\ & & a(X).E & \\ & & \overline{a}E.E' & \\ & & E \mid E' & \\ & & (x)E & \\ & & E[a{\mapsto}b] & \\ & & E{+}E' & \\ & & \mathbf{fix}X.E & \text{where } X \text{ is isolated in } E \end{array}$$

The process expression $E[a{\mapsto}b]$ is of relabelling form. Notice that the match operator is not included in $LHOCCS$. The operational semantics of $LHOCCS$ inherits from that of $LHO\pi$. The additional rules are to do with the relabelling operator.

Prefix

$$\frac{}{a.E \stackrel{a}{\longrightarrow} E} \qquad \frac{}{\overline{a}.E \stackrel{\overline{a}}{\longrightarrow} E} \qquad \frac{}{a(X).E \stackrel{aH}{\longrightarrow} E\{H/X\}} \qquad \frac{}{\overline{a}H.E \stackrel{\overline{a}H}{\longrightarrow} E}$$

Composition

$$\frac{E \stackrel{\lambda}{\longrightarrow} E' \quad bn(\lambda) \cap fn(F) = \emptyset}{E \mid F \stackrel{\lambda}{\longrightarrow} E' \mid F} \qquad \frac{E \stackrel{a}{\longrightarrow} E' \quad F \stackrel{\overline{a}}{\longrightarrow} F'}{E \mid F \stackrel{\tau}{\longrightarrow} E' \mid F'} \qquad \frac{E \stackrel{aH}{\longrightarrow} E' \quad F \stackrel{(\widetilde{x})\overline{a}H}{\longrightarrow} F'}{E \mid F \stackrel{\tau}{\longrightarrow} (\widetilde{x})(E' \mid F')}$$

$ECHECK(Y : \mathcal{FLHPE}, Z : \mathcal{FLHPE}) : Bool =$

begin

1. $ECHECK(Y, Z) := true$;
2. Suppose Y and Z are initiated respectively with $(\widetilde{z})(\widetilde{\overline{b}z} \,|\, E[X_1, \ldots, X_n])$ and $(\widetilde{z})(\widetilde{\overline{b}z} \,|\, F[X_1, \ldots, X_n])$, where $\widetilde{b} = b_1, \ldots, b_n$ are pairwise distinct fresh names and $\widetilde{z} = z_1, \ldots, z_n$ are pairwise distinct;
3. Instantiate $(\widetilde{z})(\widetilde{\overline{b}z} \,|\, E[X_1, \ldots, X_n])$ and $(\widetilde{z})(\widetilde{\overline{b}z} \,|\, F[X_1, \ldots, X_n])$ with pairwise distinct fresh names $c_1, \ldots, c_n$;
4. Convert $E[c_1, \ldots, c_n]$ and $F[c_1, \ldots, c_n]$ respectively to saturated head normal forms $\sum_{i \in I} \varphi_i \lambda_i . E_i$ and $\sum_{j \in J} \psi_j \lambda_j . F_j$;
5. **if** $I = \emptyset = J$ **then return**;
6. **repeat** for each $i \in I$ **begin**

   case λ_i **of**

   - $a(x)$:

     if some $j \in \{j \in J \mid \varphi_i \Rightarrow \psi_j \wedge \lambda_i \sigma_{\varphi_i} = \lambda_j \sigma_{\varphi_i}\}$ exists such that $ECHECK((\widetilde{z})(\widetilde{\overline{b}z} \,|\, E_i \sigma_{\varphi_i}\{y/x\}), (\widetilde{z})(\widetilde{\overline{b}z} \,|\, F_j \sigma_{\varphi_i}\{y/x\})) = true$ for all $y \in \widetilde{z}x$

     then skip

     else begin $ECHECK(Y, Z) := false$; **return end**

   - $\overline{a}x \mid a(X)$:

     if some $j \in \{j \in J \mid \varphi_i \Rightarrow \psi_j \wedge \lambda_i \sigma_{\varphi_i} = \lambda_j \sigma_{\varphi_i}\}$ exists such that $ECHECK((\widetilde{z})(\widetilde{\overline{b}z} \,|\, E_i \sigma_{\varphi_i}), (\widetilde{z})(\widetilde{\overline{b}z} \,|\, F_j \sigma_{\varphi_i})) = true$

     then skip

     else begin $ECHECK(Y, Z) := false$; **return end**

   - $\overline{a}(z')$:

     if some $j \in \{j \in J \mid \varphi_i \Rightarrow \psi_j \wedge \lambda_i \sigma_{\varphi_i} = \lambda_j \sigma_{\varphi_i}\}$ exists such that $ECHECK(E', F') = true$ for a fresh b', where

     $$\begin{aligned} E' &\equiv (z')(\widetilde{z})(\widetilde{\overline{b}z} \,|\, \overline{b'} z' \,|\, E_i \sigma_{\varphi_i}) \\ F' &\equiv (z')(\widetilde{z})(\widetilde{\overline{b}z} \,|\, \overline{b'} z' \,|\, F_j \sigma_{\varphi_i}) \end{aligned}$$

     then skip

     else begin $ECHECK(Y, Z) := false$; **return end**

   - $(\widetilde{u})\overline{a}[H]$:

     if some j in $\{j \in J \mid \varphi_i \Rightarrow \psi_j\}$ exists such that $\lambda_j \sigma_{\varphi_i} = (\widetilde{v})\overline{a}[G]$ and $ECHECK(E', F') = true$ where, for some fresh d, e

     $$\begin{aligned} E' &\equiv (\widetilde{z})(\widetilde{\overline{b}z} \,|\, (\widetilde{u})(d.(H{+}e) \,|\, E_i \sigma_{\varphi_i})) \\ F' &\equiv (\widetilde{z})(\widetilde{\overline{b}z} \,|\, (\widetilde{v})(d.(G{+}e) \,|\, F_j \sigma_{\varphi_i})) \end{aligned}$$

     then skip

     else begin $ECHECK(Y, Z) := false$; **return end**

   end;

7. Repeat step 6 by exchanging the roles of $\sum_{i \in I} \varphi_i \lambda_i . E_i$ and $\sum_{j \in J} \psi_j \lambda_j . F_j$

end

Figure 5: The Checking Algorithm

Localization

$$\frac{E \xrightarrow{\lambda} E' \quad x \notin n(\lambda)}{(x)E \xrightarrow{\lambda} (x)E'} \qquad \frac{E \xrightarrow{(\widetilde{x})\overline{a}H} E' \quad y \in fv(H) \setminus \widetilde{x}a}{(y)E \xrightarrow{(y)(\widetilde{x})\overline{a}H} E'}$$

Relabelling

$$\frac{E \xrightarrow{\lambda} E' \quad a \notin subj(\lambda)}{E[a{\mapsto}b] \xrightarrow{\lambda} E'[a{\mapsto}b]} \qquad \frac{E \xrightarrow{a} E'}{E[a{\mapsto}b] \xrightarrow{b} E'[a{\mapsto}b]} \qquad \frac{E \xrightarrow{\overline{a}} E'}{E[a{\mapsto}b] \xrightarrow{\overline{b}} E'[a{\mapsto}b]}$$

$$\frac{E \xrightarrow{aH} E'}{E[a{\mapsto}b] \xrightarrow{bH} E'[a{\mapsto}b]} \qquad \frac{E \xrightarrow{\overline{a}H} E'}{E[a{\mapsto}b] \xrightarrow{\overline{b}H} E'[a{\mapsto}b]}$$

Recursion

$$\frac{E\{\mathbf{fix}X.E/X\} \xrightarrow{\lambda} E'}{\mathbf{fix}X.E \xrightarrow{\lambda} E'}$$

What should be emphasized about the operational semantics is that according to the definition the relabelling operator is a static binder. The bisimulation equivalence for $LHOCCS$ is defined in a similar fashion, the difference from Definition 25 is that closure under the substitution of names is not required.

Definition 66. A symmetric relation $\mathcal{R}$ on $\mathcal{LHCE}$ is a bisimulation if it is closed under substitution of variables, and whenever $E\mathcal{R}F$ then the following properties hold:
(i) If $E \xrightarrow{\tau} E'$ then F' exists such that $F \Longrightarrow F'\mathcal{R}E'$.
(ii) If $E \xrightarrow{\alpha} E'$ then F' exists such that $F \stackrel{\alpha}{\Longrightarrow} F'\mathcal{R}E'$.
(iii) If $E \xrightarrow{aX} E'$ and $X \notin fn(E \,|\, F)$ then F' exists such that $F \stackrel{aX}{\Longrightarrow} F'\mathcal{R}E'$.
(iv) If $E \xrightarrow{(\widetilde{x})\overline{a}G} E'$ then some $\widetilde{x'}, H, F'$ exist such that $F \stackrel{(\widetilde{x'})\overline{a}H}{\Longrightarrow} F'$ and $(\widetilde{x})(K[G] \,|\, E')\ \mathcal{R}\ (\widetilde{x'})(K[H] \,|\, F')$ for every process expression $K[X]$ such that $\widetilde{x}\widetilde{x'} \cap fn(K[X]) = \emptyset$.
E is bisimilar to F, notation $E \approx_{LHOCCS} F$, if there exists a bisimulation $\mathcal{R}$ such that $(E, F) \in \mathcal{R}$.

The congruence $\simeq_{LHOCCS}$ is defined in the standard manner. The Expansion Law for $LHOCCS$ is much simpler than the one for $LHO\pi$:

$$\begin{aligned} E \,|\, F \;=\; & \sum_{i\in I} \lambda_i.(E_i \,|\, F) + \sum_{j\in J} \lambda_j.(E \,|\, F_j) \\ & + \sum_{i\in I, j\in J}^{\lambda_i = a_i,\ \lambda_j = \overline{b_j},\ a_i = b_j} \tau.(E_i \,|\, F_j) + \sum_{i\in I, j\in J}^{\lambda_i = \overline{a_i},\ \lambda_j = b_j,\ a_i = b_j} \tau.(E_i \,|\, F_j) \\ & + \sum_{i\in I, j\in J}^{\lambda_i = a_i(X),\ \lambda_j = (\widetilde{x})\overline{b_j}H_j,\ a_i = b_j} \tau.(\widetilde{x})(E_i\{H_j/X\} \,|\, F_j) \\ & + \sum_{i\in I, j\in J}^{\lambda_i = (\widetilde{x})\overline{a_i}H_i,\ \lambda_j = b_j(X),\ a_i = b_j} \tau.(\widetilde{x})(E_i \,|\, F_j\{H_i/X\}) \end{aligned}$$

where $E \equiv \sum_{i\in I} \lambda_i.E_i$ and $F \equiv \sum_{j\in J} \lambda_j.F_j$. A process expression of the form $\sum_{i\in I} \lambda_i.E_i$ is a head normal form. The axioms for $LHOCCS$, given in Figure 6, is obtained from Figure 1 by removing those axioms concerning the match operator. Let AS_{LHOCCS} be the set of axioms defined in Figure 6 and the Concretion Rule and the Abstraction Rule defined below:

$$\frac{(\widetilde{y})(\overline{c}.(H{+}d) \,|\, E) = (\widetilde{z})(\overline{c}.(G{+}d) \,|\, F)}{(\widetilde{y})\overline{a}[H].E = (\widetilde{z})\overline{a}[G].F} \qquad \frac{E\{c/X\} = F\{c/X\} \quad c \notin fn(E \,|\, F)}{E = F}$$

where c and d are fresh. The two rules help to reduce the higher order equivalence checking to the first order equivalence checking. One can show that every $LHOCCS$ process can be rewritten to a head normal form using axioms and rules of AS_{LHOCCS}. Without further ado we state the completeness result.

Theorem 67. *AS_{LHOCCS} is sound and complete for $\simeq_{LHOCCS}$ on the finite LHOCCS processes.*

Proof. Using the R-rules, it is easy to see that all the relabelling operations of an $LHOCCS$ process can be pushed underneath the outermost prefix operators. □

L1	$(x)\mathbf{0}$	$=$	$\mathbf{0}$	
L2	$(x)X$	$=$	X	
L3	$(x)\lambda.E$	$=$	$\lambda.(x)E$	if x does not appear in λ
L4	$(x)\lambda.E$	$=$	$\mathbf{0}$	if x is the subject name of λ
L5	$(x)(E \mid F)$	$=$	$E \mid (x)F$	if x is not free in E
L6	$(x)(y)E$	$=$	$(y)(x)E$	
L7	$(x)(E+F)$	$=$	$(x)E+(x)F$	
R1	$(\lambda.E)[a\mapsto b]$	$=$	$\lambda.E[a\mapsto b]$	$a \notin subj(\lambda)$
R2	$(a.E)[a\mapsto b]$	$=$	$b.E[a\mapsto b]$	
R3	$(\overline{a}.E)[a\mapsto b]$	$=$	$\overline{b}.E[a\mapsto b]$	
R4	$(a(X).E)[a\mapsto b]$	$=$	$b(X).E[a\mapsto b]$	
R5	$(\overline{a}H.E)[a\mapsto b]$	$=$	$\overline{b}H.E[a\mapsto b]$	
R6	$(E \mid F)[a\mapsto b]$	$=$	$E[a\mapsto b] \mid F[a\mapsto b]$	
R7	$((x)E)[a\mapsto b]$	$=$	$(x)E[a\mapsto b]$	$x \notin \{a, b\}$
R8	$(E+F)[a\mapsto b]$	$=$	$E[a\mapsto b]+F[a\mapsto b]$	
S1	$E+\mathbf{0}$	$=$	E	
S2	$E+F$	$=$	$F+E$	
S3	$E+(F+G)$	$=$	$(E+F)+G$	
S4	$E+E$	$=$	E	
T1	$\lambda.\tau.E$	$=$	$\lambda.E$	
T2	$E+\tau.E$	$=$	$\tau.E$	
T3	$\lambda.(E+\tau.F)$	$=$	$\lambda.(E+\tau.F)+\lambda.F$	

Figure 6: Axioms for $LHOCCS$

13 Remark

We have achieved the goal set out at the beginning of the paper, that is to provide an equational system for the finite linear higher order π-processes that is sound and complete for the bisimulation congruence. As far as we know, this is the first of such results on the completeness of an equational system for and the decidability of the higher order process calculi. We have advanced previous results on several aspects:

- We have proposed a general approach to investigate the higher order process calculi. Our methodologies apply to the general weak bisimulations rather than the delayed bisimulations. This is an advancement over the work of [21, 13]. Results like Abstraction Theorem and Concretion Theorem play important roles in our theory. Bisimulation Lemma is used extensively.

- Using the general approach we have improved some results on higher order calculi. One such improvement is described in Section 6. Our Fixpoint Theorem holds for the weak bisimilarity whereas the result in [23] is only valid for the strong congruence.

- We have demonstrated how to derive a complete system for a congruence on the finite $LHO\pi$-processes from a complete system for the same congruence on the finite first order π-processes. The additional rules to accomplish the transition are the Abstraction Rule and the Concretion Rule.

How much of the theory developed in this paper can be transplanted to the nonlinear higher order π-calculus? The decidability of the equivalence checking seems lost when moving from the linear scenario to the nonlinear one. For example, $(x)\overline{a}[x].\overline{x}.b$ and $(x)\overline{a}[x].\overline{x}.b.\overline{x}.b$ are bisimilar in $LHO\pi$, but they are not bisimilar in $HO\pi$. The problem is due to the higher order output prefix operator. Suppose $\overline{a}[A].E$ and $\overline{a}[A].F$ are linear higher order processes. To check the equivalence of $\overline{a}[A].E$ and $\overline{a}[B].F$, one needs to check the equivalence of $G[A] \mid E$ and $G[B] \mid F$ for every process expression $G[X]$, which is unlikely to be decidable. Apart from the decidability result, most of the results actually hold for $HO\pi$. Section 3 through Section 6 can be redone for $HO\pi$. The only exception is Theorem 24. The clause (ii') of the theorem is not valid. The approach advocated in this paper can be applied to study other aspects of the higher order calculi.

Acknowledgement We thank Dr Xiaoju Dong and Mr Xian Xu for many helpful discussions.

References

[1] L. Cardelli and A. Gordon, Mobile Ambients, *Theoretical Computer Science*, 240: 177-213, 2000.

[2] Y. Fu, A Proof Theoretical Approach to Communications, *ICALP'97*, Lecture Notes in Computer Science 1256: 325-335, 1997.

[3] Y. Fu, Variations on Mobile Processes, *Theoretical Computer Science*, 221: 327-368, 1999.

[4] Y. Fu, On Quasi Open Bisimulation, *Theoretical Computer Science*, 338: 96-126, 2005.

[5] Y. Fu and Z. Yang, Understanding the Mismatch Combinator in Chi Calculus, *Theoretical Computer Science*, 290: 779-830, 2003.

[6] Y. Fu and Z. Yang, Tau Laws for Pi Calculus, *Theoretical Computer Science*, 308: 55-130, 2003.

[7] R. Milner, *Communication and Concurrency*, Prentice Hall, 1989.

[8] R. Milner, J. Parrow and D. Walker, A Calculus of Mobile Processes, *Information and Computation*, 100: 1-40 (Part I), 41-77 (Part II), 1992.

[9] R. Milner and D. Sangiorgi, Barbed Bisimulation, *ICALP'92*, Lecture Notes in Computer Science 623: 685-695, 1992.

[10] J. Parrow and D. Sangiorgi, Algebraic Theories for Name-Passing Calculi, *Information and Computation*, 120: 174-197, 1995.

[11] D. Sangiorgi, *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*, PhD thesis, Department of Computer Science, University of Edinburgh, 1992.

[12] D. Sangiorgi, From π-Calculus to Higher Order π-Calculus–and Back, *TAPSOFT'93*, Lecture Notes in Computer Science 668: 151-166, 1993.

[13] D. Sangiorgi, Bisimulation for Higher Order Process Calculi.

[14] D. Sangiorgi, A Theory of Bisimulation for π-Calculus, *Acta Informatica*, 3: 69-97, 1996.

[15] D. Sangiorgi and R. Milner, Techniques of "Weak Bisimulation Up To", *CONCUR'92*, Lecture Notes in Computer Science 630, 1992.

[16] D. Sangiorgi and D. Walker, On Barbed Equivalence in π-Calculus, *CONCUR'01*, 2001.

[17] D. Sangiorgi and D. Walker, *The Pi Calculus–A Theory of Mobile Processes*, CUP, 2001.

[18] B. Thomsen, A Calculus of Higher Order Communicating Systems, *POPL'89*, 143-154, 1989.

[19] B. Thomsen, *Calculi for Higher Order Communicating Systems*, PhD Thesis, Department of Computing, University of London, 1990.

[20] B. Thomsen, Plain $CHOCS$–A Second Generation Calculus for Higher Order Processes, *Acta Informatica*, 30: 1-59, 1993.

[21] B. Thomsen, A Theory of Higher Order Communicating Systems, *Information and Computation*, 116: 38-57, 1995.

[22] R. J. van Gabbeek and W. P. Weijland, Branching Time and Abstraction in Bisimulation Semantics, *Journal of ACM*, 43: 555-600, 1996.

[23] M. Ying, A Shorter Proof to Uniqueness of Solutions of Equations, *Theoretical Computer Science*, 216: 395-397, 1999.

[24] M. Ying and M. Wirsing, Recursive Equations in Higher Order Process Calculi, *Theoretical Computer Science*, 266: 839-852, 2001.