A Proof Theoretical Approach to Communication

Yuxi Fu*

Department of Computer Science, Shanghai Jiao Tong University
1954 Hua Shan Road, Shanghai 200030, China

Abstract. The paper investigates a concurrent computation model, chi calculus, in which communications resemble cut eliminations for classical proofs. The algebraic properties of the model are studied. Its relationship to sequential computation is illustrated by showing that it incorporates the operational semantics of the call-by-name lambda calculus. Practically the model has pi calculus as a submodel.

1 Communication as Cut Elimination

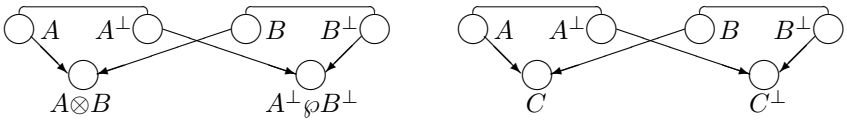
Concurrent computation is currently an open-ended issue. The situation is in contrast with sequential computation whose operational semantics is formalized by, among others, the λ-calculus ([2]). In retrospect, the λ-calculus can be seen as a fallout of proof theory. Curry-Howard's proposition-as-type principle allows one to code up constructive proofs as typed terms. At the core of the constructive logic is the minimal logic, whose type theoretical formulation gives rise to, roughly, the simply typed λ-calculus. Now the untyped λ-calculus is obtained from the simply typed λ-calculus by removing all the typing information.

In recent years, classical proofs have been investigated in a computational setting. Girard proposed proof nets ([4]) as term representations of classical linear proofs. These classical terms are typed. The conclusion of a proof derivation is the type of the proof net corresponding to that proof derivation. The computations of these terms are cut eliminations modeled by rewritings of graphs. As the terms are typed, cuts happen between nodes of correlated types. Abramsky's proof-as-process interpretation ([1, 3]) relates proof nets to processes. At operational level, this interpretation is supported by a cut-elimination-as-communication paradigm. It looks like a type-erasing interpretation similar to the one found in a constructive world.

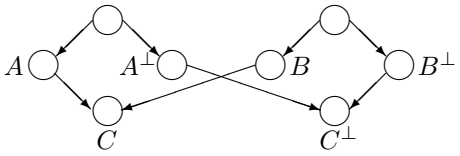
This paper investigates a concurrent computation model obtained by reversing the roles of proofs and processes in Abramsky's paradigm. That is to say that we regard communications as cut eliminations. The way to arrive at such a model of communication echoes that in the sequential world. First we take the multiplicative linear logic as the 'minimal logic' in a classical framework. There is nothing canonical about this choice. As the typed classical terms we take the

* *ICALP'97*, Lecture Notes in Computer Science 1256, 325-335, 1997.

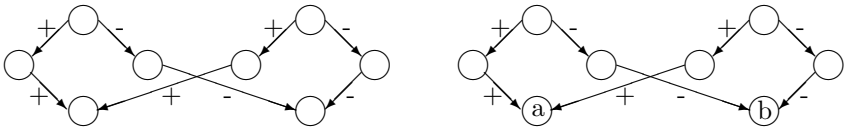
proof nets. The following left diagram is a proof net:



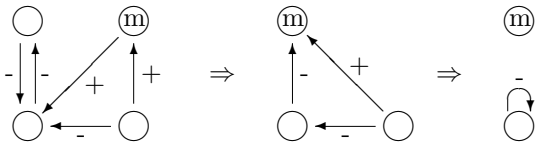
The first step towards the model is to abstract away the logical aspect of proof nets but keep its proof theoretical content. The above proof net becomes the right diagram in the above. There are two kinds of edge in the net. So the second step is to transform the net into a graph with only directed arrows:



We then forget about the typing information while recording positive and negative information by labels on arrows, arriving at an untyped graph (left below).



This is the untyped version of the original classical typed term. Notice that there are two kinds of node in the proof net: the internal nodes and the conclusion nodes. In order to distinguish them in the untyped graph, we label the conclusion nodes with small letters (above right). We call graphs of this kind reaction graphs. In a reaction graph, a node without (with) a label is called *local* (*global*). Reaction graphs can be seen as the underlying graphs of proof derivations in a generalized and distilled form. Computations with reaction graphs are cut eliminations. Here is an example of two consecutive cut-eliminations:



In the left graph, the two upper nodes show up opposite polarities to the left bottom node. This cut is eliminated in the first reduction. The two arrows are removed and the two upper nodes are coerced with the resulting node labeled by m. In the middle graph, the two bottom nodes with the arrows pointing to the node labeled m form a cut. The second reduction eliminates the cut. The idea of this paper is to think of these cut-eliminations as communications. To develop the idea, we need a process-like notation for reaction graphs. Let us define graph terms by abstract syntax as follows: $G := \mathbf{0} \mid m[x] \mid \overline{m}[x] \mid (x)G \mid G|G'$. Here $\mathbf{0}$ is the empty reaction graph; $m[x]$ and $\overline{m}[x]$ are respectively the following graphs:

$(x)G$ is obtained from G by removing the label x from G; $G|G'$ is the amal-

gamation of G and G', coercing nodes with same labels. The two consecutive cut-eliminations in the above can now be described by the following reductions:

$$(x)(y)(z)(m[x]|\overline{y}[x]|y[m]|\overline{y}[z]|\overline{z}[y]) \rightarrow (x)(y)(m[x]|\overline{y}[x]|\overline{m}[y]) \rightarrow (x)(\overline{x}[x]).$$

This term representation gives rise to a calculus of reaction graphs.

The calculus of graphs only deals with finite computations. To achieve Turing computability, we extend the language with standard process combinators. The resulting language will be referred to as χ-calculus, where χ stands for ex*ch*ange of *i*nformation. The paper initiates a study of this computation model.

2 A Model for Concurrent Computation

Let $\mathcal{N}$ be a set of names ranged over by lower case letters and $\overline{\mathcal{N}} \stackrel{\text{def}}{=} \{\overline{a} \mid a \in \mathcal{N}\}$ be the set of conames. The union $\mathcal{N} \cup \overline{\mathcal{N}}$ will be ranged over by α. Define $\overline{\alpha}$ to be m ($\overline{m}$) whenever α is $\overline{m}$ (m). Let $\mathcal{T}$ be the set of χ-terms defined as follows:

$$P := \mathbf{0} \mid \alpha[x].P \mid P|P' \mid (x)P \mid \alpha(x)*P.$$

Here $m[x].P$ and $\overline{m}[x].P$ are terms that must first perform a communication through name m and then enacts $P[y/x]$, where y is the name received in the communication. In $(x)P$, the (x)-part is a localization combinator. In both $(x)P$ and $\alpha(x)*P$, x is local. The set of local names appeared in P is denoted by $ln(P)$, whereas the set of global names, or non local names, in P is designated by $gn(P)$. Set $n(P)$ is the union of $ln(P)$ and $gn(P)$. We adopt the α-convention saying that a local name in a term can be replaced by a fresh name without changing its syntax.

The effect of a substitution $[y_1/x_1]\ldots[y_n/x_n]$ on a term is defined as follows: $P[y_1/x_1]\ldots[y_n/x_n] \stackrel{\text{def}}{=} (\ldots P[y_1/x_1]\ldots)[y_n/x_n]$. Substitutions will be ranged over by σ.

For simplicity, a structural congruence is imposed on the members of $\mathcal{T}$.

Definition 1. *The relation $=$ is the least congruence on χ-terms that contains:*
(i) $P|\mathbf{0} = P$, $P_1|P_2 = P_2|P_1$, and $P_1|(P_2|P_3) = (P_1|P_2)|P_3$;
(ii) $(x)\mathbf{0} = \mathbf{0}$, $(x)(y)P = (y)(x)P$, and $(x)(P|Q) = P|(x)Q$ if $x \notin gn(P)$;
(iii) $P = Q$ if P and Q are α-convertible.

We regard $=$ as a grammatic equality. So $P = Q$ means that P and Q are syntactically the same. The operational semantics of the language can be defined in terms of a labeled transition system. We prefer however a reductional semantics for χ-calculus in the style of [5]:

$$(x)(R|\alpha[x].P|\overline{\alpha}[y].Q) \rightarrow (x)(R[y/x]|P[y/x]|Q[y/x])$$

$$\alpha(x)*P|\overline{\alpha}[y].Q \rightarrow \alpha(x)*P|P[y/x]|Q$$

$$\frac{P \rightarrow P'}{P|Q \rightarrow P'|Q} \qquad \frac{P \rightarrow P'}{(x)P \rightarrow (x)P'}$$

To help understand the communication rules, we now give some examples, assuming x and y are distinct:

$$\begin{aligned}
(x)(R|\overline{m}[y].P|m[x].Q) &\to R[y/x]|P[y/x]|Q[y/x] \\
\overline{m}[y].P|(x)(R|m[x].Q) &\to P|R[y/x]|Q[y/x] \\
(y)(\overline{m}[y].P|(x)(R|m[x].Q)) &\to (y)(P|R[y/x]|Q[y/x]) \\
(x)\overline{m}[x].P|(y)m[y].Q &\to (z)(P[z/x]|Q[z/y]), \text{ where } z \text{ is fresh} \\
(x)(\overline{m}[x].P|m[x].Q) &\to (x)(P|Q).
\end{aligned}$$

It is clear from these examples that the localization operator in χ-calculus acts as an effect delimiter. A communication either instantiates a local name by a global name or identifies two local names.

Let $\to^+$ ($\to^*$) be the (reflexive and) transitive closure of $\to$. We will denote by $\boldsymbol{x}$ a sequence $x_1, \ldots, x_n$ of names. We will also abbreviate $(x_1) \ldots (x_n)P$ to $(\boldsymbol{x})P$. When the length of the sequence $\boldsymbol{x}$ is zero, $(\boldsymbol{x})P$ is just P.

3 Algebraic Properties

To study the algebraic semantics of χ-terms, a labeled transition system is defined as follows, where δ ranges over $\{\overset{\alpha x}{\to}, \overset{\alpha[x]}{\to}, \overset{\alpha(x)}{\to} \mid \alpha \in \mathcal{N} \cup \overline{\mathcal{N}}, x \in \mathcal{N}\}$:

$$\frac{}{(y)(R|\alpha[y].P) \overset{\alpha x}{\to} (R|P)[x/y]} \quad \frac{}{\alpha(y)*P \overset{\alpha x}{\to} \alpha(y)*P|P[x/y]} \quad \frac{}{\alpha[x].P \overset{\alpha[x]}{\to} P}$$

$$\frac{P \overset{\alpha[x]}{\to} P'}{(x)P \overset{\alpha(x)}{\to} P'} \quad \frac{P \overset{\delta}{\to} P' \quad ln(\delta) \cap gn(Q) = \emptyset}{P|Q \overset{\delta}{\to} P'|Q} \quad \frac{P \overset{\delta}{\to} P' \quad x \notin n(\delta)}{(x)P \overset{\delta}{\to} (x)P'}.$$

In the rules, $ln(\delta)$ is $\{x\}$ when δ is $\alpha(x)$; it is the empty set otherwise. $n(\delta)$ is the set of names in δ. Let $\overset{\delta}{\Rightarrow}$ denote relation $\to^* \overset{\delta}{\to} \to^*$.

A bisimulation equivalence for χ-terms should take into account the distinguished feature of the localization operators of the language. The equivalence we introduce in this section is based upon the old idea that two terms are considered observationally equivalent if and only if placing them in a same context results in two observationally equivalent terms. Working explicitly with contexts is unnecessary in our setting due to the presence of the structural equality $=$.

Definition 2. *Suppose $\mathcal{R} \subseteq \mathcal{T} \times \mathcal{T}$. The relation $\mathcal{R}$ is a local simulation if whenever $P\mathcal{R}Q$ then for any term R and any sequence $\boldsymbol{x}$ of names it holds that*
(i) if $(\boldsymbol{x})(P|R) \to P'$ then Q' exists such that $(\boldsymbol{x})(Q|R) \to^* Q'$ and $P'\mathcal{R}Q'$;
(ii) if $(\boldsymbol{x})(P|R) \overset{\delta}{\to} P'$ then Q' exists such that $(\boldsymbol{x})(Q|R) \overset{\delta}{\Rightarrow} Q'$ and $P'\mathcal{R}Q'$.
The relation $\mathcal{R}$ is a local bisimulation if both $\mathcal{R}$ and its inverse are local simulations. The local bisimilarity $\approx$ is the largest local bisimulation.

As usual, local bisimulation up to $\approx$ is a useful tool for proving two χ-terms being locally bisimilar. We omit the standard definition.

In the rest of this section, we prove that $\approx$ is a congruence relation. The fact that $\approx$ is closed under parasition and localization combinators can be proved already at this point.

Proposition 3. *If $P \approx Q$ then (i) $P|O \approx Q|O$ and (ii) $(x)P \approx (x)Q$.*

The next lemma is crucial in showing that $\approx$ is a congruence relation. It is the first indication that local bisimilarity is algebraically appropriate. The property is not enjoyed by local bisimilarity for π-processes.

Lemma 4. *If $P \approx Q$ then $P\sigma \approx Q\sigma$ for an arbitrary substitution σ.*

Proof. Let $\mathcal{R}$ be the union of $\approx$ and the following

$$\left\{ ((\boldsymbol{z})(P\sigma|R), (\boldsymbol{z})(Q\sigma|R)) \;\middle|\; \begin{array}{l} P \approx Q,\ R \in \mathcal{T},\ \boldsymbol{z} \text{ a sequence of names,} \\ \sigma \text{ a substitution } [y_1/x_1]\ldots[y_n/x_n] \text{ such} \\ \text{that } x_1, \ldots, x_n \text{ are pairwise distinct} \end{array} \right\}.$$

Suppose $(\boldsymbol{z})(P\sigma|R)\mathcal{R}(\boldsymbol{z})(Q\sigma|R)$ and $(\boldsymbol{z})(P\sigma|R) \stackrel{\delta}{\rightarrow} P'$, where σ is the substitution $[y_1/x_1]\ldots[y_n/x_n]$ with $x_1, \ldots, x_n$ being pairwise distinct. Let a and b be fresh names. Then for the sequence $\boldsymbol{z}$ of names

$$\begin{aligned}(\boldsymbol{z})((\boldsymbol{x})(a)(b)(b[b].P|a[x_1].\ldots.a[x_n]|\overline{a}[y_1].\ldots.\overline{a}[y_n].\overline{b}[b])|R) &\rightarrow^* (\boldsymbol{z})(P\sigma|R) \\ &\stackrel{\delta}{\rightarrow} P'.\end{aligned}$$

As $b \notin gn(P, Q)$, $b[b].P \approx b[b].Q$ follows easily. By Proposition 3,

$$\begin{aligned}&(\boldsymbol{x})(a)(b)(b[b].P|a[x_1].\ldots.a[x_n]|\overline{a}[y_1].\ldots.\overline{a}[y_n].\overline{b}[b]) \\ \approx\ &(\boldsymbol{x})(a)(b)(b[b].Q|a[x_1].\ldots.a[x_n]|\overline{a}[y_1].\ldots.\overline{a}[y_n].\overline{b}[b]).\end{aligned}$$

So by definition, there exists some Q' such that $P' \approx Q'$ and

$$(\boldsymbol{z})((\boldsymbol{x})(a)(b)(b[b].Q|a[x_1].\ldots.a[x_n]|\overline{a}[y_1].\ldots.\overline{a}[y_n].\overline{b}[b])|R) \stackrel{\delta}{\Rightarrow} Q'.$$

During the above reduction every $a[x_i]$ must have reacted upon $\overline{a}[y_i]$, for $1 \leq i \leq n$, and $b[b]$ upon $\overline{b}[b]$. It can be easily proved that all the communications through a and that through b can happen in the very beginning. That is

$$\begin{aligned}(\boldsymbol{z})((\boldsymbol{x})(a)(b[b].Q|a[x_1].\ldots.a[x_n]|\overline{a}[y_1].\ldots.\overline{a}[y_n].\overline{b}[b])|R) &\rightarrow^* (\boldsymbol{z})(Q\sigma|R) \\ &\stackrel{\delta}{\Rightarrow} Q'.\end{aligned}$$

So $(\boldsymbol{z})(P\sigma|R) \stackrel{\delta}{\rightarrow} P'$ is matched by $(\boldsymbol{z})(Q\sigma|R) \stackrel{\delta}{\Rightarrow} Q'$. The case when $(\boldsymbol{z})(P\sigma|R) \rightarrow P'$ is similar. So $\mathcal{R}$ is a local bisimulation. It follows that $P \approx Q$ implies $P[y/x] \approx Q[y/x]$. Therefore $P \approx Q$ implies $P\sigma \approx Q\sigma$ for a substitution σ. □

We now come to the main result of the section.

Theorem 5. *$\approx$ is a congruence equivalence: if $P \approx Q$ and $O \in \mathcal{T}$ then*
(i) $\alpha[x].P \approx \alpha[x].Q$; *(ii) $P|O \approx Q|O$;*
(iii) $(x)P \approx (x)Q$; *(iv) $\alpha(x)*P \approx \alpha(x)*Q$.*

Proof. We sketch the proof of (iv). The proof of (i) is simpler. Let $\mathcal{R}$ be

$$\{((\boldsymbol{x})(m(y)*P|R), (\boldsymbol{x})(m(y)*Q|R)) \mid P \approx Q,\ R \in \mathcal{T},\ m, \boldsymbol{x} \text{ names}\}.$$

Suppose $(\boldsymbol{x})(m(y)*P|R) \rightarrow P'$ and that $(\boldsymbol{x})(m(y)*P|R) \rightarrow P'$ is caused by a communication between $m(y)*P$ and R. Then P' is $(\boldsymbol{x})(m(y)*P|P[a/y]|R')$. Similarly $(\boldsymbol{x})(m(y)*Q|R) \rightarrow (\boldsymbol{x})(m(y)*Q|Q[a/y]|R')$. By Lemma 4, $P[a/y] \approx Q[a/y]$. By Proposition 3, $(\boldsymbol{x})(m(y)*Q|P[a/y]|R') \approx (\boldsymbol{x})(m(y)*Q|Q[a/y]|R')$. It is then easy to see that $\mathcal{R}$ is a local bisimulation up to $\approx$. □

4 π-Processes as χ-Terms

A question naturally arises as to the relationship between π-calculus and χ-calculus. We give a first answer in this section. Let $\mathcal{P}$ be the set of π-processes defined as follows: $P := \mathbf{0} \mid m(x).P \mid \overline{m}x.P \mid P|P' \mid (x)P \mid m(x)*P$. We refer the reader to [6] for background material on π-calculus.

There are many bisimulation equivalences on π-processes. What is most relevant in this section is the open bisimilarity defined in [8]. Actually we will use a version of open bisimilarity stronger than Sangiorgi's.

Definition 6. *Let $\mathcal{R}$ be a binary relation on the set of π-processes. The relation $\mathcal{R}$ is an open bisimulation if whenever $P\mathcal{R}Q$ then for any π-process R, any sequence $\boldsymbol{x}$ of names and any substitution σ it holds that*
(i) if $(\boldsymbol{x})(P\sigma|R) \stackrel{\beta}{\Rightarrow} P'$ then Q' exists such that $(\boldsymbol{x})(Q\sigma|R) \stackrel{\beta}{\Rightarrow} Q'$ and $P'\mathcal{R}Q'$;
(ii) if $(\boldsymbol{x})(Q\sigma|R) \stackrel{\beta}{\Rightarrow} Q'$ then P' exists such that $(\boldsymbol{x})(P\sigma|R) \stackrel{\beta}{\Rightarrow} P'$ and $P'\mathcal{R}Q'$.
The open bisimilarity $\approx^o$ is the largest open bisimulation.

$\approx^\circ$ is a congruence equivalence and is closed under substitution.

A structural translation from π to χ has as nontrivial clauses the following:

$$(m(x).P)^\circ \stackrel{\text{def}}{=} (x)m[x].P^\circ,$$
$$(\overline{m}x.P)^\circ \stackrel{\text{def}}{=} \overline{m}[x].P^\circ.$$

Imposing on $\mathcal{P}$ a same structural congruence as given in Definition 1, one has

Theorem 7. *For $P, Q \in \mathcal{P}$, it holds that*
(i) $P \rightarrow Q$ iff $P^\circ \rightarrow Q^\circ$; *(ii) $P \stackrel{mx}{\rightarrow} Q$ iff $P^\circ \stackrel{mx}{\rightarrow} Q^\circ$;*
(iii) $P \stackrel{\overline{m}x}{\rightarrow} Q$ iff $P^\circ \stackrel{\overline{m}[x]}{\rightarrow} Q^\circ$; *(iv) $P \stackrel{\overline{m}(x)}{\rightarrow} Q$ iff $P^\circ \stackrel{\overline{m}(x)}{\rightarrow} Q^\circ$.*

Theorem 8. *For $P, Q \in \mathcal{P}$, $P \approx^o Q$ iff $P^\circ \approx Q^\circ$.*

5 Call-by-Name in χ-Calculus

A concurrent computation model has to answer the question of whether it captures sequential computation successfully. The issue is often addressed by relating variants of λ-calculus to the model. Our focus in this section is on the

call-by-name λ-calculus ([7]), whose semantics is defined by the following rules:

$$\frac{}{(\lambda x.M)N \to M[N/x]} \quad \frac{M \to M'}{MN \to M'N} \quad \frac{M \to M'}{\lambda x.M \to \lambda x.M'}.$$

The following translation, which is Milner's encoding of the lazy λ-calculus with modification, serves as an encoding of the call-by-name λ-calculus in χ-calculus:

$$\begin{aligned} [\![x]\!]u &\stackrel{\text{def}}{=} \overline{x}[u] \\ [\![\lambda x.M]\!]u &\stackrel{\text{def}}{=} (v)(x)(u[x].u[v]|[\![M]\!]v) \\ [\![MN]\!]u &\stackrel{\text{def}}{=} (v)(x)([\![M]\!]v|\overline{v}[x].\overline{v}[u].x(w)*[\![N]\!]w). \end{aligned}$$

The parasition of $\overline{u}[x].\overline{u}[v]$ and $[\![M]\!]v$ in $[\![\lambda x.M]\!]u$ allows $[\![M]\!]v$ to evolve independently, thus modeling reduction under λ-abstraction. The encoding preserves the operational semantics of the call-by-name λ-calculus in the sense the operational semantics of the lazy λ-calculus is preserved by Milner's encoding ([5]). A formal treatment is omitted in this extended abstract.

The call-by-name λ-calculus is one example which can not be treated successfully in π-calculus.

6 Towards an Integration of χ and λ

There are two problems one encounters when trying to simulate the operational semantics of the full λ-calculus. The first is how to model reduction under λ-abstraction. The second is how to model reduction $MN \to MN'$ caused by $N \to N'$. The former is to do with parallel computation. There is no reason why it should pose any problem for concurrent computation. This view is supported by the result in Sect. 5. The latter is to do with recursion because the λ-term N may be duplicated in future reduction. In any *structural* interpretation, this N must be translated into the body of a replicator or guarded recursion. So if the N induces an infinite reduction, the interpretation of MN would have no terminating reduction sequences. It is our view that the second problem is orthogonal to concurrent computation. It is caused essentially by the incompatibility of the two recursion mechanisms.

In this section we take a look at a higher order calculus combining the communication mechanism of the χ-calculus and the recursion mechanism of the λ-calculus. The purpose of this investigation is to see if the two mechanisms fit coherently and if local bisimulation suffices as a tool for studying the algebraic properties of the language.

6.1 χ with Call-by-Name λ

Let the set $\mathcal{H}$ of higher order χ-terms be defined by the following abstract syntax:

$$E := X \mid \alpha[x].E \mid E|E' \mid (x)E \mid \alpha(X)E \mid \alpha[E],$$

where X is a term variable. Let $\mathbf{0}$ abbreviate $(a)a(X)X$. The semantics of the higher order χ-calculus is defined by the relevant rules of the first order χ-calculus together with the following rules incorporating a call-by-name mechanism:

$$\frac{}{\alpha(X)E|\overline{\alpha}[F] \to E[F/X]} \qquad \frac{E \to F}{\alpha(X)E \to \alpha(X)F}$$

A structural equality is imposed on the members of $\mathcal{H}$, whose definition is the same as Definition 1. Usually a bisimulation equivalence for a higher order process calculus is defined for closed processes. This is a tractable approach. But in the presence of the second reduction rule given above, the method breaks down. A bisimulation equivalence for higher order χ-calculus has to be defined on all terms. For that purpose, let's say that a binary relation $\mathcal{R}$ on $\mathcal{H}$ is substitution closed if whenever $E\mathcal{R}F$ then $E[E_1/X_1, \ldots, E_i/X_i]\mathcal{R}F[E_1/X_1, \ldots, E_i/X_i]$ for $E_1, \ldots, E_i \in \mathcal{H}$ and $X_1, \ldots, X_i$ that are among the free variables of $E|F$.

Definition 9. *A substitution closed binary relation $\mathcal{R}$ on $\mathcal{H}$ is a local bisimulation if whenever $E\mathcal{R}F$ then for any $H \in \mathcal{H}$ and $\{\boldsymbol{x}\} \subseteq \mathcal{N}$ it holds that*
(i) if $(\boldsymbol{x})(E|H) \stackrel{\delta}{\Rightarrow} E'$ then F' exists such that $(\boldsymbol{x})(F|H) \stackrel{\delta}{\Rightarrow} F'$ and $E'\mathcal{R}F'$;
(ii) if $(\boldsymbol{x})(F|H) \stackrel{\delta}{\Rightarrow} F'$ then E' exists such that $(\boldsymbol{x})(E|H) \stackrel{\delta}{\Rightarrow} E'$ and $E'\mathcal{R}F'$.
The local bisimilarity $\approx^\omega$ is the largest local bisimulation on higher order terms.

The above definition is given in terms of a labeled transition system on $\mathcal{H}$ that is defined by the relevant rules in Sect. 3. It should be remarked that $\approx^\omega$ is by definition substitution closed.

Theorem 10. *$\approx^\omega$ is a congruence equivalence: if $E \approx^\omega F$ and $G \in \mathcal{H}$ then*
(i) $\alpha[x].E \approx^\omega \alpha[x].F$; *(ii) $E|G \approx^\omega F|G$;* *(iii) $(x)E \approx^\omega (x)F$;*
(iv) $\alpha(X)E \approx^\omega \alpha(X)F$; *(v) $\alpha[E] \approx^\omega \alpha[F]$.*

Proof. We only prove (v). For the sake of this proof, let's define $\mathcal{H}_o[X]$ to be the set of all higher order terms E such that each occurrence of X is within $\alpha[G]$ for some $\alpha \in \mathcal{N} \cup \overline{\mathcal{N}}$ and some $G \in \mathcal{H}$. Let $\mathcal{R}$ be

$$\{(E[A/X], E[B/X]) \mid A \approx^\omega B,\ E \in \mathcal{H}_o[X],\ X \text{ a variable}\}.$$

Suppose $E[A/X] \to G$. Then $G \equiv F[A]$ for some $F \in \mathcal{H}_o[X]$. It can be easily shown that some $H \in \mathcal{H}$ exists such that $E[B/X] \to H$ and $F[B/X] \approx^\omega H$. It follows that $\mathcal{R}$ is a local bisimulation up to $\approx^\omega$. Thus $\alpha[E] \approx^\omega \alpha[F]$ since $\alpha[X] \in \mathcal{H}_o[X]$. □

In the remaining of the section, we justify our claim that the higher order calculus is a combination of χ and λ.

6.2 Recursion

As a test for local bisimilarity, we examine Thomsen's recursion ([9]) in this section. Suppose that E contains free variable X and a does not occur in E. The

following abbreviations will be used:

$$W^a_X(E) \stackrel{\text{def}}{=} a[a]|a(X)(\overline{a}[a].E|\overline{a}[X]),$$
$$\mathsf{rec}X.E \stackrel{\text{def}}{=} (a)(W^a_X(E)|\overline{a}[W^a_X(E)]).$$

We remark that $\mathsf{rec}X.E$ defined here is slightly different from Thomsen's. The idea is to make $W^a_X(E)$ inert. Before proving the main property concerning $\mathsf{rec}X.E$, we first establish the following result.

Lemma 11. *$(a)(F[W^a_X(E)/X]|\overline{a}[W^a_X(E)]) \approx^\omega (a)(b)(F'|\overline{a}[W^a_X(E)]|\overline{b}[W^b_X(E)])$, where E and F have free variable X and F' is obtained from $F[W^a_X(E)/X]$ by replacing some occurrences of $W^a_X(E)$ by $W^b_X(E)$. Here a and b are fresh.*

Theorem 12. *Suppose E contains free X. Then $\mathsf{rec}X.E \approx^\omega E[\mathsf{rec}X.E/X]$.*

Proof. Suppose E and F contain free variable X, $a \notin n(E,F)$ and $gn(E) \cap ln(F) = \emptyset$. Using Lemma 11, one proves that $(a)(F[W^a_X(E)/X]|\overline{a}[W^a_X(E)]) \approx F[\mathsf{rec}X.E/X]$. So $\mathsf{rec}X.E \approx^\omega (a)(E[W^a_X(E)/X]|\overline{a}[W^a_X(E)]) \approx^\omega E[\mathsf{rec}X.E/X]$, which is what we are after. □

6.3 Projecting Out Guarded Recursion

In this section we show that the higher order χ can be seen as an extension of the first order χ. A fallout of the result is a justification of the claim that the guarded recursion is completely unnecessary in the higher order χ-calculus. Let χ^+ be the higher order χ-calculus enriched with the guarded recursion. The language χ^+ can be investigated along the same line as the higher order χ has been. $\mathcal{H}_+$ and $\approx^+$ are defined accordingly. It can also be shown that $\approx^+$ is a congruence relation. The definition of a structural translation $\widehat{\ }$ from χ^+-terms to χ^ω-terms is nontrivial only on guarded recursion:

$$\widehat{\alpha(x)*E} \stackrel{\text{def}}{=} (a)((x)\alpha[x].(\widehat{E}|a(X)(X|\overline{a}[X]))|\overline{a}[(x)\alpha[x].(\widehat{E}|a(X)(X|\overline{a}[X]))]).$$

The translation $\widehat{\ }$ projects the guarded recursion out, as it were.

Theorem 13. *For $P \in \mathcal{H}_+$, $P \approx^+ \widehat{P}$.*

Theorem 14. *(i) Suppose P and Q are in $\mathcal{H}$. Then $P \approx^+ Q$ iff $P \approx^\omega Q$.*
(ii) Suppose P and Q are in $\mathcal{H}_+$. Then $P \approx^+ Q$ iff $\widehat{P} \approx^\omega \widehat{Q}$.
(iii) (a) if $P \xrightarrow{\delta} P'$ ($P \to P'$) then $\widehat{P} \xrightarrow{\delta} P''$ ($\widehat{P} \to P''$) such that $P'' \approx^\omega \widehat{P'}$;
(b) if $\widehat{P} \xrightarrow{\delta} P''$ ($\widehat{P} \to P''$) then $P \xrightarrow{\delta} P'$ ($P \to P'$) such that $P'' \approx^\omega \widehat{P'}$.

Proof. (i) Suppose P, Q are in $\mathcal{H}$. $P \approx^+ Q$ clearly implies $P \approx^\omega Q$. Suppose $P \approx^\omega Q$. Then $\widehat{(\boldsymbol{x})(P|R)} \equiv (\boldsymbol{x})(P|\widehat{R})$ and $\widehat{(\boldsymbol{x})(Q|R)} \equiv (\boldsymbol{x})(Q|\widehat{R})$, where $R \in \mathcal{H}_+$. By theorem 13, $(\boldsymbol{x})(P|R) \approx^+ (\boldsymbol{x})(P|\widehat{R})$ and $(\boldsymbol{x})(Q|R) \approx^+ (\boldsymbol{x})(Q|\widehat{R})$. It is now easy to see that $\approx^\omega$ is a local bisimulation up to $\approx^+$.
(ii) By theorem 13, $P \approx^+ Q$ iff $\widehat{P} \approx^+ \widehat{Q}$. By (i) $\widehat{P} \approx^+ \widehat{Q}$ iff $\widehat{P} \approx^\omega \widehat{Q}$. □

As χ^+ extends the first order χ, so does the higher order χ-calculus in view of Theorem 13 and Theorem 14.

6.4 Full Integration

An integration of χ with the full λ is the higher order calculus extended with

$$\frac{E \to F}{\alpha[E] \to \alpha[F]}.$$

The operational semantics of the full λ-calculus can be simulated in the fully integrated calculus. The encoding is the following:

$$\begin{aligned}
[\![x]\!]_u &\stackrel{\text{def}}{=} \overline{x}[u]|X \\
[\![\lambda x.M]\!]_u &\stackrel{\text{def}}{=} (x)(v)(u[v].u[x]|x(X)[\![M]\!]_v) \\
[\![MN]\!]_u &\stackrel{\text{def}}{=} (x)(v)([\![M]\!]_v|\overline{v}[u].\overline{v}[x]|\overline{x}[(w)(x[w]|[\![N]\!]_w)]).
\end{aligned}$$

Theorem 15. *Suppose M is a λ-term. If $M \to N$ then $[\![M]\!]_u \to^+ [\![N]\!]_u$.*

Definition 9 now gives rise to an equivalence relation on the set of all terms of the fully integrated calculus. The results in Sect. 6.2 and Sect. 6.3 also hold for this language. The (i) through (iv) of Theorem 10 also hold. But so far we haven't been able to prove the (v) of Theorem 10 for the fully integrated calculus.

7 Remark on Pragmatics

In the formulation of χ-calculus, we use the same set of names for both global and local names. But conceptually the identification is not always helpful. The standard bisimilarity ([6]) for the π-processes is not closed under input prefixing operation. This is because the variable names and the free names are regarded as semantically different in this approach. Sangiorgi's open bisimilarity is congruent. But in that approach local names are treated differently. In χ-calculus, both local and global names are variable names, which is what local bisimilarity assumes. The situation is similar to that in λ-calculus, where both free and closed variables are, well, variables that can be instantiated by any λ-terms.

But variable names alone do not suffice in practice. This is clear from the mobile process interpretation of object oriented languages ([10]). The usual practice is to postulate that $\mathcal{N}$ consists of two parts: a set $\mathcal{N}_v$ of variable names and a set $\mathcal{N}_c$ of constant names. We can now define a χ-process to be a χ-term in which all variable names are localized. So in χ-processes there are two kinds of local names: local variable names and local constant names. A communication either identifies two local variable names or replaces a local variable name by a local or global constant name. A communication between two constant names is prohibited. Let β range over $\{\rightarrow, \stackrel{\alpha a}{\rightarrow}, \stackrel{\alpha[a]}{\rightarrow}, \stackrel{\alpha(a)}{\rightarrow} | \ a \in \mathcal{N}_c, \alpha \in \mathcal{N}_c \cup \overline{\mathcal{N}_c}\}$.

Definition 16. *Let $\mathcal{R}$ be a binary relation on the set of χ-processes. $\mathcal{R}$ is a simulation if $P\mathcal{R}Q$ implies that if $P \stackrel{\beta}{\rightarrow} P'$ then there exists some Q' such that $Q \stackrel{\hat{\beta}}{\Rightarrow} Q'$ and $P'\mathcal{R}Q'$. The relation $\mathcal{R}$ is a bisimulation if both $\mathcal{R}$ and its reverse are simulations. The bisimilarity $\approx^\chi$ is the largest bisimulation.*

The π-calculus can be reexamined in this new setting. The input prefix operation restricts variable names whereas the localization operation always restricts constant names. π-processes are now defined to be those processes in which all variable names are restricted by input prefixes. Let γ range over $\{\rightarrow, \stackrel{ca}{\rightarrow}, \stackrel{\overline{c}a}{\rightarrow}, \stackrel{\overline{c}(a)}{\rightarrow} \mid a, c \in \mathcal{N}_c\}$.

Definition 17. *Let $\mathcal{R}$ be a binary relation on the set of π-processes. $\mathcal{R}$ is a simulation if $P\mathcal{R}Q$ implies that if $P \stackrel{\gamma}{\rightarrow} P'$ then there exists some Q' such that $Q \stackrel{\hat{\gamma}}{\Rightarrow} Q'$ and $P'\mathcal{R}Q'$. The relation $\mathcal{R}$ is a bisimulation if both $\mathcal{R}$ and its inverse are simulations. The bisimilarity $\approx^{\pi}$ is the largest bisimulation.*

The translation given in Sect. 4 works in this practical setting. It establishes an operational correspondence in the sense of Theorem 7. In addition one has

Theorem 18. *For π-processes P and Q, $P \approx^{\pi} Q$ if and only if $P^{\circ} \approx^{\chi} Q^{\circ}$.*

So practically speaking, π is a subcalculus of χ.

References

1. Abramsky, S.: Proofs as Processes. Theoretical Computer Science **135** (1994) 5–9
2. Barendregt, H.: The Lambda Calculus: Its Syntax and Semantics. 1984
3. Bellin, G., Scott, P.: Remarks on the π-Calculus and Linear Logic. Theoretical Computer Science **135** (1994) 11–65
4. Girard, J.: Linear Logic. Theoretical Computer Science **50** (1987) 1–102
5. Milner, R.: Functions as Processes. Journal of Mathematical Structures in Computer Science **2** (1992) 119-141
6. Milner, R., Parrow, J., Walker, D.: A Calculus of Mobile Processes. Information and Computation **100** (1992) 1–40 (Part I), 41–77 (Part II)
7. Plotkin, G.: Call-by-Name, Call-by-Value and the λ-Calculus. Theoretical Computer Science **1** (1975) 125-159
8. Sangiorgi, D.: A Theory of Bisimulation for π-Calculus. Proc. CONCUR 93, LNCS 715 (1993) 127–142
9. Thomsen, B.: Plain CHOCS—A Second Generation Calculus for Higher Order Processes. Acta Informatica **30** (1993) 1–59
10. Walker, D.: Objects in the π-Calculus. Information and Computation **116** (1995) 253–271