On the Expressiveness of Higher Order Sessions (Draft of September 29, 2014)

No Author Given

Imperial College London

1 Full Higher Order Session π

We present the syntax and operational semantics for the *Full Higher Order Session* π calculus or Full $\mathsf{HO}\pi$. Full $\mathsf{HO}\pi$ is a simpler version of the higher order calculus developed in [1], that name and process abstraction application, as well as implicit application of abstractions. We believe that the higher order semantics of the calculus are small, yet expressive enough to capture in a straightforward way the principles of session types.

1.1 Full $\mathsf{HO}\pi$ Syntax

The syntax for Full $\mathsf{HO}\pi$ is defined in Figure 1. We assume a set of names S that range over $s, s_1, \ldots$ and a dual set of S, $\overline{S}$ that ranges over $\overline{s}, \overline{s_1}, \ldots$. Both sets S and $\overline{S}$ constitude the set of names N. Name variables $x, y, z, \ldots$ are taken from set Var and process variable $X, Y, Z, \ldots$ from set PVar. Set RVar is the set of recursive variables that range over $\mathcal{X}, \mathcal{Y}, \ldots$. Abstactions, written $(x)P$, are processes P with a bound parameter x. The set for abstractions is Abs. We denote either names or name variables with $k, k_1, \ldots$. Names, name variables, process variables, recursive variables or abstractions are denoted by symbol $V, V_1 \ldots$. Note that set N includes session and shared names. We sometimes denote shared names with $a, b, c, \ldots$, although $a, b, c, \cdots \in S$.

Processes include the standard π-calculus prefixes for sending and receiving names. Prefix $k!\langle k' \rangle; P$ denotes the sending of name k over channel k and then continuing as P whereas prefix $k?(x); P$ denotes the reception of a value on channel k over variable x and then continue as P. Recursion is expressed on the primitive recursor $\mu\mathcal{X}.P$ that binds the recursive variable/process $\mathcal{X}$ in the structure of P.

Higher order syntax is composed by the sending prefix $k!\langle (x)Q \rangle; P$ that denotes the sending of abstraction $(x)Q$ over channel k and then continuing as P. On the receiving side prefix $k?(X); P$ denotes the reception of an abstraction on channel k and over the process variable X. Process $X\langle k \rangle$ is the application process which is used to bind channel k on the process substituting process variable X.

We assume the standard session syntax for selection and branching. Process $k \triangleleft l; P$ selects label l on channel k and then behaves as P. Process $k \triangleright \{l_i : P_i\}_{i \in I}$ offers a choice on labels $l_i, i \in I$ with corresponding continuations the processes $P_i, i \in I$. The calculus is completed with the standard processes for parallel composition $P_1 \mid P_2$, name restriction $(\nu\ s)P$ and the inactive process $\mathbf{0}$.

A well formed process relies on the assumptions for guarded recursive processes. We say that a process is a *program* if it contain no free variables or free process/recursive variables.

$$
\begin{array}{lcl}
P,Q & ::= & k!\langle k'\rangle;P \mid k?(x);P \mid X \mid \mu X.P \\
 & \mid & k!\langle (x)Q\rangle;P \mid k?(X);P \mid X\langle k\rangle \\
 & \mid & k \triangleleft l;P \mid k \triangleright \{l_i : P_i\}_{i\in I} \mid P_1 \mid P_2 \mid (\nu\ s)P \mid \mathbf{0} \\
\text{Names} & : & S = \{s, s_1, \ldots\} \qquad \overline{S} = \{\overline{s} \mid s \in S\} \qquad N = S \cup \overline{S} \\
\text{Variables} & : & \mathsf{Var} = \{x, y, z, \ldots\} \qquad \mathsf{PVar} = \{X, Y, Z, \ldots\} \qquad \mathsf{RVar} = \{\mathcal{X}, \mathcal{Y}, \ldots\} \\
\text{Abstractions} & : & \mathsf{Abs} = \{(x)P \mid P \text{ is a process}\} \\
 & & k \in N \cup \mathsf{Var} \qquad V \in N \cup \mathsf{Var} \cup \mathsf{PVar} \cup \mathsf{RVar} \cup \mathsf{Abs}
\end{array}
$$

Fig. 1. Syntax for Full HOπ

We identify two important sub-calculi of the Full HOπ that will form the basis of the study in this paper, i) The calculus defined by the lines 1 and 3 of the process syntax in Figure 1, and ii) The calculus defined by the lines 2 and 3 of the process syntax in Figure 1. The first calculus is the standard session π calculus or session-π as defined in the bibliography []. The second calculus allows only for abstraction passing and abstraction application as well as selecting and branching and it is called Higher Order session calculus or HOπ.

Structural Congruence We define structural congruence as the least congruence that satisfies the commutative monoid ($\mid$, $\mathbf{0}$) and the rules:

$(\nu\ s)\mathbf{0} \equiv \mathbf{0}$ $\quad$ $s \notin \mathtt{fn}(P_1)$ implies $P_1 \mid (\nu\ s)P_2 \equiv (\nu\ s)(P_1 \mid P_2)$ $\quad$ $\mu X.P \equiv P\{\mu X.P/X\}$

1.2 Operational Semantics

We define the reduction semantics in Figure 2. Figure 2 first describes the process variable substitution through the semantics of name substitution. Substitution of application process $X\langle k\rangle$ over abstraction $(x)Q$ substitutes free variable x in Q with k and replaces X with the resulting process. There is no effect on variable substitution for the inactive process. In all the other cases process variable substitution is homorphic on the structure of the process.

References

1. Dimitris Mostrous and Nobuko Yoshida. Two session typing systems for higher-order mobile processes. In *TLCA'07*, volume 4583 of *LNCS*, pages 321–335. Springer, 2007.

$$X\langle k\rangle\{(x)Q/X\} = Q\{k/x\} \qquad \mathbf{0}\{(x)Q/X\} = \mathbf{0} \qquad s?(Y);P\{(x)Q/X\} = s?(Y);(P\{(x)Q/X\})$$

$$(s!\langle(y)P_1\rangle;P_2)\{(x)Q/X\} = s!\langle(y)P_1\{(x)Q/X\}\rangle;(P_2\{(x)Q/X\})$$

$$s \triangleleft l;P\{(x)Q/X\} = s \triangleleft l;(P\{(x)Q/X\}) \qquad s \triangleright \{l_i : P_i\}_{i\in I}\{(x)Q/X\} = s \triangleright \{l_i : P_i\{(x)Q/X\}\}_{i\in I}$$

$$(P_1 \mid P_2)\{(x)Q/X\} = P_1\{(x)Q/X\} \mid P_2\{(x)Q/X\} \qquad (\nu\ s)P\{(x)Q/X\} = (\nu\ s)(P\{(x)Q/X\})$$

$$s!\langle s'\rangle;P_1 \mid s?(x);P_2 \longrightarrow P_1 \mid P_2\{s'/x\} \quad \text{[MPass]}$$

$$s!\langle(x)P\rangle;P_1 \mid s?(X);P_2 \longrightarrow P_1 \mid P_2\{(x)P/X\} \quad \text{[APass]}$$

$$s \triangleleft l_k;P \mid s \triangleright \{l_i : P_i\}_{i\in I} \longrightarrow P \mid P_k \qquad k \in I \quad \text{[Sel]}$$

$$P_1 \longrightarrow P_1' \text{ implies } P_1 \mid P_2 \longrightarrow P_1' \mid P_2 \quad \text{[Par]}$$

$$P \longrightarrow P' \text{ implies } (\nu\ s)P \longrightarrow (\nu\ s)P' \quad \text{[RName]}$$

$$P \equiv\longrightarrow\equiv P' \text{ implies } P \longrightarrow P' \quad \text{[Cong]}$$

Fig. 2. The reduction semantics for Full HOπ