Replication

John Glauert, and Jeong-Ho Lee

Version A. 13th September 1995

1 Introduction

This document discusses the use of replication in the π–calculus and its translation to OGRe and hence to π^π–calculus.

2 Varieties of π–Calculi

The base calculus we will use is for the π–calculus with multiple communications, but without sums, using syntax:

$$N \quad ::= \quad x(\vec{y}) \cdot N \mid \overline{x}\,\vec{v} \cdot N \mid (\nu\, x)\,(N) \mid N \mid N' \mid 0 \mid\, !\,N$$

In the discussion on π^π–calculus we have assumed that agent definitions take the place of replication:

$$N \quad ::= \quad x(\vec{y}) \cdot N \mid \overline{x}\,\vec{v} \cdot N \mid (\nu\, x)\,(N) \mid N \mid N' \mid 0 \mid A(\vec{y}) \mid N \; where \; A(\vec{y}) = N'$$

We can provide a crude translation of $!\,P$ as:

$$!\,P \quad \equiv \quad A(n_P) \; where \; A(n_P) = (P \mid A(n_P))$$

where n_P stands for $fn(P)$.

For practical purposes we will want communication to trigger expansion of replications. The usual form in which we will find replication is $!\,x(\vec{y}) \cdot P$. If we use n_{Pxy} to stand for $fn(P) \backslash \{x, y\}$ then we can represent the replication as:

$$!\,x(\vec{y}) \cdot P \quad \equiv \quad A(x, n_{Pxy}) \; where \; A(x, n_{Pxy}) = x(\vec{y}) \cdot (P \mid A(x, n_{Pxy}))$$

However, some sort of proof is needed to show that this version is weakly bisimilar (or even strongly bisimilar) to the crude translation, using suitable semantics for agents.

An avenue which Jeong Ho and John have explored is to use the base syntax for π–calculus, but to restrict replication to the input guarded form:

$$N \quad ::= \quad x(\vec{y}) \cdot N \mid \overline{x}\,\vec{v} \cdot N \mid (\nu\, x)\,(N) \mid N \mid N' \mid 0 \mid\, !\,x(\vec{y}) \cdot N$$

This form of replication acts like an input guarded process which spawns a copy of itself when communication occurs. Instead of structural equivalence rules for replication, we could have two forms of communication:

$$\begin{aligned} \overline{x}\,\vec{v} \cdot P \mid x(\vec{y}) \cdot Q \quad &\rightarrow \quad P \mid Q\{\vec{v}/\vec{y}\} \\ \overline{x}\,\vec{v} \cdot P \mid\, !\,x(\vec{y}) \cdot Q \quad &\rightarrow \quad P \mid Q\{\vec{v}/\vec{y}\} \mid\, !\,x(\vec{y}) \cdot Q \end{aligned}$$

We can provide something like the general case of replication. In the following, $x \notin fn(P)$:

$$Q = (\nu\, x)\,(\overline{x} \mid !\, x() \cdot (P \mid \overline{x})) \quad \approx \quad !\, P$$

Q becomes, by a τ step,

$$\begin{array}{rl} & (\nu\, x)\,(P \mid \overline{x} \mid !\, x() \cdot (P \mid \overline{x})) \\ \cong & P \mid (\nu\, x)\,(\overline{x} \mid !\, x() \cdot (P \mid \overline{x})) \\ = & P \mid Q \end{array}$$

So it seems that Q is weakly bisimilar to $!\, P$ using standard π–calculus semantics.

3 Simulation of π–calculus by OGRe

With these restricted forms or replication, and adding rules for multiple communications, provided by Jeong Ho, we have a revised translation scheme. See the more complete note *PiSky Version K* for more background.

$$N \quad ::= \quad x(\vec{y}) \cdot N \mid \overline{x}\,\vec{v} \cdot N \mid (\nu\, x)\,(N) \mid N \mid N' \mid 0 \mid !\, x(\vec{y}) \cdot N$$

The function q maps π–calculus processes to OGRe processes:

$$\begin{array}{rcl} q[\![\overline{x}\,\vec{v} \cdot N]\!] & = & m : B(n_N),\ n\,!\,Put(Vec_k(\vec{v}), m) \\ where & & m : B(n_N),\ m\,!\,Sync \rightarrow m : Dead,\ q[\![N]\!] \\ q[\![\overline{x}\,\vec{v}]\!] & = & n\,!\,APut(Vec_k(\vec{v})) \\ q[\![x(\vec{y}) \cdot N]\!] & = & m : A(n_{Ny}),\ x\,!\,Get(m) \\ where & & m : A(n_{Ny}),\ m\,!\,Data(Vec_k(\vec{y})) \rightarrow m : Dead,\ q[\![N]\!] \\ q[\![!\, x(\vec{y}) \cdot N]\!] & = & m : A(x, n_{Nxy}),\ x\,!\,Get(m) \\ where & & m : A(x, n_{Nxy}),\ m\,!\,Data(Vec_k(\vec{y})) \rightarrow m : A(x, n_{Nxy}),\ x\,!\,Get(m),\ q[\![N]\!] \\ q[\![(\nu\, n)\,(N)]\!] & = & n : Chan(EQ),\ q[\![N]\!] \\ q[\![N \mid N']\!] & = & q[\![N]\!],\ q[\![N']\!] \\ q[\![0]\!] & = & \end{array}$$

where $n_N = fn(N)$, $n_{Ny} = fn(N) \setminus \{y_0 \ldots y_k\}$, and $n_{Nxy} = fn(N) \setminus \{x, y_0 \ldots y_k\}$. Vec_k stands for a data constructor of the required arity for the corresponding sequence of input or output values. Hence, for $\vec{v} = v_1 \ldots v_k$, where $k \geq 0$, we use $Vec_k(v_1, \ldots, v_k)$.

We have provided rules for asynchronous π–calculus. The translation of guarded replication was also developed by Jeong Ho.

The state *Dead* labels a process which should never receive any further messages. It remains to be proved that this is the case for every *Dead* process which arises from evaluating the OGRe system generated by q from any π–calculus term.