On Congruence Property of Scope Equivalence for Concurrent Programs with Higher-Order Communication

Masaki Murakami
Okayama University
JAPAN
A Formal Model of Concurrent Systems

the model presented here is a translation of asynchronous local high-order π-calculus (Sangiorgi) into graph rewriting.
Motivation

To represent the scopes of channel names precisely

$\forall$-operator

$$\forall a(P \mid \forall b(Q \mid R))$$

Not convenient to express scopes of names for some purpose..
Scopes not nested

- Impossible to represent with a $\forall$-operator

$$\forall a (P \mid \forall b (Q \mid R))$$
We can not decide..

\[ \nu a(P \mid \nu b(Q \mid R)) \] means......

or
Our approach..

Our model is based on graph rewriting.

not based on process algebra.

a translation of asynchronous higher-order $\pi$-calculus into graph rewriting
Basic Idea

A system is a collection of *processes* sharing *names*

A system is represented as a bipartite graph

- Source nodes $\implies$ processes
- Sink nodes $\implies$ names
- There is an edge iff the source nodes is in the scope of the sink node
Basic Idea

bipartite graph
Processes

A source node consists of labels for its prefix and its continuation.

Reduce a process by "peeling" the node.

\[ a(x).P \]
Message node

A message node is a tuple of its subject and its object.

\[ a < c > \]
Operational Semantics

- a set of graph rewriting rules
- by translating the rules for the labeled transition system of asynchronous $\pi$-calculus into rules for graph rewriting
Rules for graph rewriting

The rule for message receiving..
Rules for graph rewriting

- If the imported name is new to the receiver, new edges are created.
Higher-Order Communication
We define a new equivalence relation to distinguish two processes which are equivalent on their behavior but not for their scopes of names.
Example

When $x$ does not occur in $Q$

$P_1$ and $P_2$ are equivalent in their behavior

but not equivalent for scopes of names

$P_1 = m(x).\tau.Q$

$P_2 = \forall n(m(u). (n<a> \mid n(x).Q))$
Example

— Note that $Q$ may be just a specification of the behavior. It does not represent the implementation.

— “$x$ does not occur in $Q$” does not mean “the imported name no longer exists in $Q$”

— $P_1 = m(x).\tau.Q$

— If the name receive by $m(x)$ is a secret data which should not be leaked to $Q$, this $P_1$ is no good (but $P_2$ is OK).
Behavior equivalences can not tell you the difference.
The graph rewriting model can represent the difference.

\[ m\langle o \rangle \xrightarrow{m(x)} \]

\[ Q \]

\[ o \]
Example

\[ P_2 = \forall n(m(u). \ (n < a > \mid n (x). \ Q)) \]
Define a new equivalence relation that is called scope equivalence that can distinguish these two processes.

\[ P_1 = m(x).\tau.Q \]

\[ P_2 = \forall n(m(u). (n<a> | n(x). Q)) \]
For a graph $P$ and a name $n$, $P/n$ is a subgraph of $P$ which consists of

- source nodes in the scope of $n$
- and sink nodes other than $n$
Scope Bisimulation

A relation $R$ is a **scope bisimulation** if for any $P$ and $Q$ such that $(P, Q)$ in $R$,

- $P$ is an empty graph iff $Q$ is an empty graph
- the set of source nodes of $P/n$ is empty iff the source nodes $Q/n$ is also empty for any common name $n$
- $P/n$ and $Q/n$ are strongly bisimilar for any common name $n$
- $R$ is a strong bisimulation
There exists the largest scope bisimulation which is an equivalence relation congruent w.r.t. contexts (composition, prefix, replication, new name...) in first-order case (ICTAC 08)
Congruence: for higher-order model

When $P$ and $Q$ are scope equivalent, $P$ and $Q$ are also equivalent.
When \( P \) and \( Q \) are scope equivalent, then

\[ P \quad \text{and} \quad Q \]

are also equivalent.
Non Congruence w.r.t. input prefix

$P$ and $Q$ are scope equivalent but....
The Non Congruence result

- It comes from....
- Scope equivalence is NOT congruent w.r.t. higher-order substitution.
The Counter Example

- \( P \) and \( Q \) are equivalent.

\[
\begin{align*}
& b_1 \quad x(a) \quad b_2 \\
& \quad n_1 \quad \quad \quad \quad \quad n_2
\end{align*}
\]

\(P\)

\[
\begin{align*}
& b \quad x(a) \\
& \quad n_1 \quad n_2
\end{align*}
\]

\(Q\)
The Counter Example

- Not equivalent after the higher-order substitution.

\[ P[ (y)(c(u).d(v).R) / x ] \]

\[ Q[ (y)(c(u).d(v).R) / x ] \]
The counter example

\[
\begin{align*}
&b'[o/x] \\
&\quad (y)(c(u).d(v).R[α]) \langle α \rangle \\
&\quad (y)(c(u).d(v).R[α]) \langle α \rangle \\
&\quad n_1 \quad n_2
\end{align*}
\]

\[
\begin{align*}
&\tau \\
&\quad (y)(c(u).d(v).R[α]) \langle α \rangle \\
&\quad n_1 \quad n_2
\end{align*}
\]

\[
\begin{align*}
&\quad \tau \\
&\quad (y)(c(u).d(v).R[α]) \langle α \rangle \\
&\quad n_1 \quad n_2
\end{align*}
\]

\[
\begin{align*}
&\quad (y)(c(u).d(v).R[a / y]) \langle α \rangle \\
&\quad (y)(c(u).d(v).R[a / y]) \langle α \rangle \\
&\quad (y)(c(u).d(v).R[a / y]) \langle α \rangle \\
&\quad n_1 \quad n_2
\end{align*}
\]

\[
\begin{align*}
&c(m) \\
&\quad (y)(c(u).d(v).R[a / y]) \langle α \rangle \\
&\quad n_1 \quad n_2
\end{align*}
\]

\[
\begin{align*}
&c(m) \\
&\quad (y)(c(u).d(v).R[a / y]) \langle α \rangle \\
&\quad n_1 \quad n_2
\end{align*}
\]

\[
\begin{align*}
&\quad (y)(c(u).d(v).R[a / y][m/u]) \\
&\quad (y)(c(u).d(v).R[a / y][m/u]) \\
&\quad (y)(c(u).d(v).R[a / y][m/u]) \\
&\quad n_1 \quad n_2
\end{align*}
\]
Conclusion

A graph rewriting model of concurrent/distributed systems with higher-order message represents scopes of names precisely equivalence relation

- Congruent w.r.t. any context in first order
- Not congruent w.r.t. input (and higher-order) context