Symbolic Bisimulations and Proof Systems for the π-Calculus

H. Lin*†

Laboratory for Computer Science

Institute of Software

Chinese Academy of Sciences

Abstract

A theory of symbolic bisimulation for the π-calculus is proposed which captures the conventional notions of bisimulation-based equivalences for this calculus. Proof systems are presented for both *late* and *early* equivalences, and their soundness and completeness are proved. The proof system for early equivalence differs from that for late equivalence only in the inference rule for input prefixing. For the version of π-calculus extended with the *mismatch* construction, complete proof systems can be obtained by adding a rule for mismatch to the proof systems for the

late bisimulation congruences, [Hen91] and [BD92] for *testing*

partition of the condition. Because of this we can use maximally consistent extensions instead of the phrase "... there exists a partition ..." in the definition of symbolic bisimulation. Another important property of maximal consistency is that it characterises substitutions upto an injective substitution. That is, two substitutions satisfying the same maximally consistent condition differ only by an injective substitution. As ground bisimulation in the π-calculus is preserved by injective substitutions, the *finite* set of maximally consistent conditions on $fn(t,u)$ captures *all* the substitutions needed to close up $t \dot{\sim} u$ in order to get $t \sim u$.

The rest of the paper is organised as follows: The calculus and its semantics are introduced in the next section. The inference system is presented in Section , along with the completeness proof. Section 4 discusses extensions to the calculus. Section 5 demonstrates how the theory developed for the late equivalence in the previous sections can be carried over to the early case. The paper concluded with Section 6 where the relation with other work are also discussed.

2

pre $\dfrac{}{\alpha.t \xrightarrow{\alpha} t}$ match $\dfrac{t \xrightarrow{\alpha} t'}{[x=x]t \xrightarrow{\alpha} t'}$

sum $\dfrac{t \xrightarrow{\alpha} t'}{t+u \xrightarrow{\alpha} t'}$ par $\dfrac{t \xrightarrow{\alpha} t'}{t \mid u \xrightarrow{\alpha} t' \mid u}\ bn(\alpha) \cap fn(u) = \emptyset$

res $\dfrac{t \xrightarrow{\alpha} t'}{(x)t \xrightarrow{\alpha} (x)t'}\ x \notin n(\alpha)$ com $\dfrac{t \xrightarrow{a(x)} t' \quad u \xrightarrow{\overline{a}y} u'}{t \mid u \xrightarrow{\tau} t'[y/x] \mid u'}$

open $\dfrac{t \xrightarrow{\overline{a}x} t'}{(x)t \xrightarrow{\overline{a}(x)} t'}\ a \neq x$ close $\dfrac{t \xrightarrow{a(x)} t' \quad u \xrightarrow{\overline{a}(x)} u'}{t \mid u}$

Pre $\dfrac{}{\alpha.t \xrightarrow{true,\alpha} t}$ Match $\dfrac{t \xrightarrow{M,\alpha} t}{}$

The elements in a condition are treated as conjuncts. We avoid introducing disjunction into the theory of conditions. The major advantage is that the relation $C \Rightarrow D$ can be tested in linear time w.r.t the size of C and D:

Proposition 2.4 *The relation $C \Rightarrow D$ is linear time decidable.*

Proof: We first generate equivalence classes from the equalities in C, along with a list of pairs of representatives of the equivalence classes such that the pair of representatives of x and y is in the list if and only if $x \neq y$ is in C. This process takes time linear to the size of C. Then for each element e of D if e is an equality $a = b$ we check if $a,\ b$ is in the same equivalence class; if e is an inequality $a \neq b$ we check if the pair consisting of their representatives are in the list. □

Two substitutions σ and σ' are equal on V, written $\sigma =^V \sigma'$, if $x\sigma = x\sigma'$ for all $x \in V$. Two substitutions σ and σ' are *elementarily equivalent* on V if for any $x, y \in V$ $\sigma \models x = y$ if and only if $\sigma' \models x = y$. Clearly equal substitutions are elementarily equivalent. On the other hand elementarily equivalent substitutions are not necessary equal, though they do not differ much, as the following lemma reveals.

Lemma 2.5 *Suppose σ and σ' are substitutions on V. If they are elementarily equivalent on V, then there exists an injective substitution δ such that $\sigma =^V \sigma'\delta$.*

Proof: Let δ be the substitution such that $(x\sigma)\delta = x\sigma'$ and $(x\sigma')\delta = x\sigma$ for all $x \in V$, and is identity otherwise. δ is well defined because σ and σ' are elementarily equivalent on V. Furthermore it is injective and satisfies $\sigma =^V \sigma'\delta$. □

A condition C is *consistent* if there are no $x,\ y \in \mathcal{N}$ such that $C \Rightarrow x = y$ and $C \Rightarrow x \neq y$. C is *maximally consistent* on $V \subset \mathcal{N}$ if for any $x,\ y \in V$ either $C \Rightarrow x = y$ or $C \Rightarrow x \neq y$.

C' is a *maximally consistent extension* of C on V, written $C' \in MCE_V(C)$, if $C \subseteq C'$ and C' is maximally consistent on V. The set of maximally consistent extensions of a given condition on a finite set of names V is finite. We will abbreviate $MCE_V(true)$ as MC_V.

Given a condition C, define $E_V(C) = \{\, x = y \mid x, y \in V,\ C \not\Rightarrow x = y \ \text{ and } \ C \not\Rightarrow x$

2. $x = y \in I^*$;

. $x \neq y \in \overline{I}$.

In each case either $D \Rightarrow x = y$ or $D \Rightarrow x \neq y$. □

Corollary 2.7 $\bigvee MCE_V(C) = C$

Proof:

$$\begin{aligned} \bigvee MCE_V(C) &= \bigvee\{\, C \cup I \cup \overline{I} \mid I \subseteq E_V(C) \,\} \\ &= C \wedge \bigvee\{\, I \cup \overline{I} \mid I \subseteq E_V(C) \,\} \\ &= C \wedge \mathit{true} \\ &= C \end{aligned}$$

□

Corollary 2.7 shows that the set of all consistent extensions of a condition (on a given name set) constitutes a particular *partition*, or *decomposition* of the condition.

Proposition 2.8 *C is maximally consistent on V if and only if all substitutions satisfying C are elementarily equivalent on V.*

Proof: The "only if" part is trivial. For the "if" part, suppose C is not maximally consistent on V. Then there exist x, $y \in V$ such that neither $C \Rightarrow x = y$ nor $C \Rightarrow x \neq y$. So there exists a substitution σ such that $\sigma \models C$ and $\sigma \models x \neq y$. Let σ' be the substitution which is the same as σ except that it sends x to $y\sigma$. Then we still have $\sigma' \models C$. But σ' is not elementarily equivalent to σ on V because $\sigma' \models x = y$. A contradiction. □

Corollary 2.9 *Suppose C is maximally consistent on V. If σ and σ' both satisfy C, then $\sigma =^V \sigma'\delta$ for some injective substitution δ.*

In other words, all substitutions satisfying a maximally consistent condition on a given name set are *isomorphic* on that name set. This shows the importance of the notion of maximal consistence: it captures substitutions upto isomorphism. Recalling Lemma 2.2 that isomorphic substitutions make no difference as far as bisimulation is concerned, the set of maximally consistent conditions on $fn(t, u)$ characterises all possible substitutions that may affect the bisimilarity between t and u. Although $t \sim u$ is defined as the closure of $t \dot{\sim} u$ over *all* substitutions, only *finite* number of substitutions need to be checked, one for each maximally consistent condition on $fn(t, u)$.

2.3 Symbolic Bisimulations

Now we are ready to give the definition of symbolic bisimulation.

We write $\alpha =^C \beta$ to mean

if $\alpha \equiv \tau$ then $\beta \equiv \tau$
if $\alpha \equiv \overline{a}x$ then $\beta \equiv \overline{b}y$ and $C \Rightarrow a = b$, $C \Rightarrow x = y$
if $\alpha \equiv \overline{a}(x)$ then $\beta \equiv \overline{b}(x)$ and $C \Rightarrow a = b$
if $\alpha \equiv a(x)$ then $\beta \equiv b(x)$ and $C \Rightarrow a = b$

Definition 2.10 A condition indexed family of symmetric relations $\mathcal{S} = \{S^C\}$ is a late symbolic bisimulation if $(t,u) \in S^C$ implies

> whenever $t \xrightarrow{M,\alpha} t'$ with $bn(\alpha) \cap fn(t,u,C) = \emptyset$, then for each $C' \in MCE_{fn(t,u)}(C \cup M)$ there is a $u \xrightarrow{N,\beta} u'$ such that $C' \Rightarrow N$, $\alpha =^{C'} \beta$,

Proof: *2* and *3* are direct consequences of *1*. The "only if" part of *1* can be established by defining

$$R = \{ (t\sigma, u\sigma) \mid t \sim$$

AXIOM $\dfrac{}{true \rhd t = u}$ $t = u$ is an axiom instance

CHOICE $\dfrac{C \rhd t_i = u_i}{C \rhd t_1 + t_2 = u_1 + u_2}$

L-INPUT $\dfrac{C \rhd t = u}{C \rhd a(x).t = b(x).u}$ $C \Rightarrow a = b, \quad x \notin fn(C)$

OUTPUT $\dfrac{C \rhd t = u}{C \rhd \overline{a}x.t = \overline{b}y.u}$ $C \Rightarrow a = b,\ C \Rightarrow x = y$

TAU $\dfrac{C \rhd t = u}{C \rhd \tau.t = \tau.u}$

MATCH $\dfrac{C \cup \{x = y\} \rhd t = u \quad C \cup \{x \neq y\} \rhd \mathbf{0} = u}{C \rhd [x = y]t = u}$

RES $\dfrac{C \cup \{\, x \neq y \mid y \in fn((x)t, (x)u)\,\} \rhd t = u}{C \rhd (x)t = (x)u}$ $x \notin n(C)$

PARTITION $\dfrac{C \cup \{x = y\} \rhd t = u \quad C \cup \{x \neq y\} \rhd t = u}{C \rhd t = u}$

CONSEQ $\dfrac{C \rhd t = u}{C' \rhd t = u}$ $C' \Rightarrow C$

ABSURD $\dfrac{}{false \rhd t = u}$

Figure : The Inference Rules for Late Symbolic Bisimulation

As we are working modulo α-equivalence, we also assume the following rule

ALPHA $\dfrac{}{true \rhd t = u}$ t and u are $\alpha -$ equivalent

We write $\vdash C \rhd t = u$ to mean $C \rhd t = u$ can be derived from this inference system.

Proposition 3.1 *1. If $C \Rightarrow M$ and $\vdash C \rhd t = u$ then*

S1	$X + \mathbf{0} = X$	R1	$(x)\mathbf{0} = \mathbf{0}$
S2	$X + X = X$	R2	$(x)\alpha.X = \alpha.(x)X$ if $x \notin n(\alpha)$
S	$X + Y = Y + X$	R	$(x)\alpha.X = \mathbf{0}$ if x is the port of α
S4	$(X + Y) + Z = X + (Y + Z)$	R4	$(x)(y)X = (y)(x)X$
		R5	$(x)(X + Y) = (x)X + (x)Y$

Figure 4: The Axioms for Choice And Restriction

The rule PARTITION permits a case analysis on the name space represented by a condition: To see if $t = u$ holds over C, we can decompose C into $C \cup \{x = y\}$ and $C \cup \{x \neq y\}$, and exam each separately. In fact, this rule can be generalised to allow arbitrary decompositions. The following proposition gives the case for a particular decomposition, *i.e.* the decomposition of a condition into its maximally consistent extensions (Corollary 2.7):

Proposition 3.2 *If* $\vdash D \rhd t = u$ *for each* $D \in MCE_V(C)$ *then* $\vdash C \rhd t = u$.

Proof: Suppose $\vdash D \rhd t = u$ for each $D \in MCE_V(C)$. By Lemma 2.6 $D \in MCE_V(C)$ *iff* $D = C \cup I \cup \overline{I}$ for some $I \subseteq E_V(C)$. Apply induction on the cardinality of $E_V(C)$.

If $E_V(C)$ is empty then C is the only maximally consistent extension of itself, and the result is trivial.

Otherwise assume $E_V(C)$ has $n + 1$ elements and let $x = y \in E_V(C)$. We have

$$MCE_V(C) = MCE_V(C \cup \{x = y\}) \cup MCE_V(C \cup \{x \neq y\})$$

and

$$E_V(C \cup \{x = y\}) \subseteq E_V(C) - \{x = y\}$$

$$E_V(C \cup \{x \neq y\}) \subseteq E_V(C) - \{x = y\}$$

So by induction $\vdash C \cup \{x = y\} \rhd t = u$, $\vdash C \cup \{x \neq y\} \rhd t = u$. By PARTITION $\vdash C \rhd t = u$. □

The following proposition summarises the interaction between the restriction and match operators.

Proposition 3.3 *1.* $\vdash (x$

Theorem 3.4 *(Soundness of $\vdash$) If $\vdash C \rhd t = u$ then $t \sim_L^C u$.*

The rest of this section is devoted to the proof of completeness result for $\vdash$.
The *height* of a term t is defined inductively thus

- $\mid \mathbf{0} \mid = 0$
- $\mid t + u \mid = \ max\{\mid t \mid, \mid u \mid\}$
- $\mid [x = y]t \mid = \mid t \mid$
- $\mid (x)t \mid = \mid t \mid$
- $\mid \alpha.t \mid = 1 + \mid t \mid$

If $a \neq x$ then we abbreviate $(x)\overline{a}x.t$ as $\overline{a}(x).t$. $\overline{a}(x)$ is a derived action and is called *bound output.*

Proposition 3.5 *Suppose $C \Rightarrow a = b$, $x \notin n(C)$. If $\vdash C \cup \{ x \neq y \mid y \in fn(\overline{a}(x).t, \overline{b}(x).u) \} \rhd t = u$ then $\vdash C \rhd \overline{a}(x).t = \overline{b}(x).u$.*

Proof: Since $\vdash C \cup \{ x \neq y \mid y \in fn(\overline{a}(x).t, \overline{b}(x).u) \} \rhd t = u$ and $C \Rightarrow a = b$, by OUTPUT we get $\vdash C \cup \{ x \neq y \mid y \in fn(\overline{\ }$

Proof:

Let $t \equiv \sum_i M_i\alpha_i.t_i$ and $u \equiv \sum_j N_j\beta_j.u_j$ with $bn(\alpha_i) \cap fn(u) = bn(\beta_j) \cap fn(t) = \emptyset$. Then

$$t \mid u = \sum_i M_i\alpha_i.(t_i \mid u) \; + \; \sum_j N_j\beta_j.(t \mid u_j) \; + \sum_{\alpha_i \; opp \; \beta_j} M_iN_j[a_i = b_j]\tau.v_{ij}$$

where α_i *opp* β_j and v_{ij} are defined as follows

1. $\alpha_i \equiv a(x), \beta_j \equiv \overline{b}y$; then $v_{ij} \equiv t_i[y/x] \mid u_j$;
2. The converse of the above clause;

. $\alpha_i \equiv a(x), \beta_j \equiv \overline{b}(y)$; then $v_{ij} \equiv (z)(t_i[z/x] \mid u_j[z/y])$ with $z \notin fn(t, u)$;

4. The converse of the above clause.

Figure 5: The Expansion Law

the calculus with mismatch in order to give axiomatisations for testing or bisimulation equivalences.

To include mismatch into the language we first extend the operational semantics by including the following two rules: "mismatch" and "Mismatch" in Figure 1 and Figure 2, respectively (now M ranges over conditions)

$$\text{mismatch} \;\; \frac{t \xrightarrow{\alpha} t'}{[x \neq y]t \xrightarrow{\alpha} t'} \;\; x$$

Early symbolic bisimulation is obtained by separating the clause for input transition from other cases in Definition 2.10:

> whenever $t \stackrel{M,a(x)}{\longrightarrow_L} t'$ with $x \notin fn(t,u,C)$, then for each $C' \in MCE_{fn(t,u)\cup\{x\}}(C \cup M)$ there is a $u \stackrel{N,b(x)}{\longrightarrow}$

where $J' = \{\, j \in J \mid D \Rightarrow N_j \,\}$. Hence, from the assumption we can derive

$$\vdash_E C \rhd \sum_{i \in I'} M_i \tau.t_i = \sum_{j \in J'} N_j \tau.u_j$$

As a further research topic we would like to

but also equalities. By such mild generalisation, direct characterisations of both late and early bisimulation equivalences for the π-calculus become possible and sound and complete proof systems can be formulated, as demonstrated by our results presented here. It is also interesting to characterise open bisimulation using our symbolic approach. Such a characterisation could facilitate the comparisons between open, late and early bisimulations, as they are expressed within the same framework.**opknos6rk.**

References

[BD92] M. Boreale and R. DeNicola. Testing equivalence for mobile processes. In *CONCUR'92*, number 6 0 in Lecture Notes in Computer Science, pages 2 –