Unique Fixpoint Induction for Mobile Processes

H. Lin*
Laboratory for Computer Science
Institute of Software, Chinese Academy of Sciences
P.O Box 8718, Beijing 100080
E-mail: lhm@ios.ac.cn

Abstract

Complete proof systems for bisimulation equivalences in the π-calculus *with recursion* are presented. The key inference rule dealing with recursion is *unique fixpoint induction* which generalises that used in [Mil84] for regular pure-*CCS*. It is shown that the proof systems are sound, and also complete when restricted to the subset of π-calculus where recursions are guarded and the parallel composition is disallowed. These results on the one hand extend the proof systems for recursion-free π-calculus in [Lin94, Lin95] to infinite processes, on the other hand extend the inference system for guarded regular pure-*CCS* of [Mil84] to π-calculus.

1 Introduction

There has been growing interest in providing sound and complete proof systems for the π-calculus. Various notions of equivalences have been axiomatised: late and early strong bisimulation congruence [PS93, BD94, Lin94], open bisimulation [San93], late and early weak bisimulation equivalences [Lin95], and testing equivalences [Hen91, BD92]; Different styles of proof systems have been used: equational axiomatisation [PS93, San93] and *symbolic* inference systems [BD94, Lin94, Lin95].

All these proof systems are only for *recursion-free* π-calculus in which processes can perform only a finite number of actions. The same is also true for message-passing calculi in general ([HL93]).

Unique fixpoint induction was proposed in [Mil84] as the key inference rule to reason about recursion in pure-*CCS* (i.e. *CCS* without value-passing):

$$\frac{P = Q[P/X]}{P = \mathbf{fix} X Q} \qquad X \text{ guarded in } Q \tag{1}$$

*Supported by the President Fund of the Chinese Academy of Sciences, the National Science Foundation of China, and the EU KIT project SYMSEM.

where P, Q are process expressions. Milner showed that the inference system consisting of the usual axioms for strong bisimulation for finite *CCS* together with this rule and a rule for folding/unfolding recursive definitions are complete for strong bisimulations equivalence in guarded regular *CCS*.

It is not immediately clear whether this result can be generalised to π-calculus, because the completeness proof in [Mil84] relies on the finite-state property of regular *CCS*, while in π-calculus even trivial processes can exhibit infinite state behaviour. There are two constructions in π-calculus that can entail such infinity: the behaviour after an input action depends on the received name, and there are infinite many possible names to receive; a recursively defined process identifier is usually parameterised on names, so each identifier definition can give rise to infinite number of instantiations, one for each vector of name parameters ([MPW92]).

The symbolic theories for the π-calculus, as advocated in [BD94, Lin94, Lin95], overcome the first source of infinity by working symbolically, without instantiating input names in either operational semantics or bisimulation definition. The current paper will show how this "symbolic" technique can also be exploited to eliminate the second source of infinity, the infinity resulted from instantiating name parameters of recursively defined processes.

The language we will work with is the π-calculus analogue of regular pure-*CCS*, in which the parallel operator does not occur in recursion bodies. Unlike pure-*CCS*, in the π-calculus process variables and recursions may take name parameters. To avoid the complexity caused by the interplay between free names and process variables, we use *abstraction* to form recursive expressions. So typical recursion has the form

$$\mathbf{fix} X(\tilde{x})T$$

where $(\tilde{x})T$ is an abstraction with free names of T in $\tilde{x}$, *i.e.* $(\tilde{x})T$ is name-closed. Such a term is itself an abstraction, and to form a recursive expression name parameters must be supplied. Analogous to lambda calculus we use *application* to realise parameter-passing: to pass name parameter $\tilde{w}$ to recursive abstraction $\mathbf{fix} X(\tilde{x})T$ we write

$$(\mathbf{fix} X(\tilde{x})T)\tilde{w}.$$

One may call $\tilde{x}$ and $\tilde{w}$ the "formal" and "actual" parameters, respectively, of the recursive expression.

By introducing abstraction and application into π-calculus, the concern of parameter-passing is separated from that of recursion. As a consequence the rules dealing with recursion appear simple. In fact they are syntactically identical to their pure-*CCS* counterparts. Here is the unique fixpoint induction rule:

$$\frac{F = G[F/X]}{F = \mathbf{fix} X G} \qquad X \text{ guarded in } G \tag{2}$$

where both F and G are abstractions. Since in pure-*CCS* process variables and recursions do not take parameters and as such can be regarded as nullary abstraction, rule (1) is a special case of (2).

The main result of this paper is that the proof systems obtained by adding (2) and a rule for unfolding recursion to those for the finite π-calculus of [Lin94] are sound, and also complete for strong bisimulation equivalences in the subset of π-calculus without parallel composition.

The proof structure of the completeness theorem follows the beautiful proof of [Mil84]: we first show that any process provably satisfies a finite set of standard equations. As mentioned before recursive processes may have infinite states. However, with symbolic transitional semantics, modulo renaming of free names there are only a finite number of abstract states a recursively defined process can reach, and this set of equations represent all such states. The second step is to establish that if two process are symbolically bisimular then an equation set can be constructed which is provably satisfied by both. Finally we show that two processes can be proved equal if they satisfy the same set of equations. Compared with [Mil84] extra complexities are caused by the presence of free names, the restriction operator, and the conditional construction.

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 3. Section 4 is devoted to the completeness proof of the inference system. Section 5 demonstrates how the theory developed for late equivalence can be carried over to the early case. The paper is concluded with Section 6.

2 π-Calculus And Bisimulations

2.1 The Language And Operational Semantics

To give the syntax for the π-calculus we first presuppose an infinite set $\mathcal{N}$ of *names*, ranging over by $a,\ b,\ x,\ y,\ w,\ldots$, and a countably infinite set of *process abstraction variables* (or just *process variables* for short), *Var*, ranged over by $X, Y, Z, W, \ldots$. Each process abstraction variable is associated with an *arity* indicating the number of names it should take to form a process expression. The language of π-calculus can then be given by the following BNF grammar

$$
\begin{array}{rcl}
T & ::= & \mathbf{0} \quad | \quad \alpha.T \quad | \quad T+T \quad | \quad T \mid T \quad | \quad \phi T \quad | \quad \nu x T \quad | \quad F\tilde{x} \\
F & ::= & X \quad | \quad (\tilde{x})T \quad | \quad \mathbf{fix} X F \\
\alpha & ::= & \tau \quad | \quad a(x) \quad | \quad \overline{a}x \\
\phi & ::= & [x=y] \quad | \quad \phi \wedge \phi \quad | \quad \neg\phi
\end{array}
$$

We refer to [MPW92] for the intended meaning of these operators. The precedences are listed below (in decreasing order):

$$. \quad \nu x \quad \text{conditional} \quad | \quad + \quad \mathbf{fix} X$$

In $a(x).T$ and $\nu x T$ x is a bound name with scope T. In $(\tilde{x})T$ $\tilde{x}$ are also bound in T. We use $bn(T)$ and $fn(T)$ for the set of bound and free names in T, respectively. For any expression e $n(e)$ denotes the set of names appearing in e.

An *abstraction* is either a process variable, a process abstraction $(\tilde{x})T$ (where $\tilde{x}$ is a vector of distinct names), or a recursive abstraction $\mathbf{fix}XF$. We shall only use name-closed abstractions, so in $(\tilde{x})T$ it is always required that $fn(T) \subseteq \tilde{x}$. Abstractions are ranged over by $F,\ G$. A recursive abstraction $\mathbf{fix}XF$ binds X in F, and we use $FV(T)$ to denote the set of free process variables of T. The *degree* of an abstraction is defined inductively by: the degree of X is the arity of X; the degree of $(\tilde{x})T$ is the length of $\tilde{x}$; the degree of $\mathbf{fix}XF$ is the degree of F. For a recursion $\mathbf{fix}XF$ to be well-formed the degree of F must be equal to the arity of X. For an application $F\tilde{x}$ to be well-formed the degree of F must be equal to the length of $\tilde{x}$. We shall assume that all the expressions we write are syntactically well-formed. The notation $(\tilde{x})^{\phi}T$, read "abstraction over ϕ", will be used as a shorthand for $(\tilde{x})(\phi T)$. In this paper we shall only be interested in *guarded* recursion $\mathbf{fix}XF$ where each occurrence of X in F is within the scope of some prefix operator $\alpha.-$. In Section 4 concerning the completeness result it is further required that parallel composition $|$ does not appear in recursion bodies. This sub-language, as termed *finite control* in [Dam94], is the π-calculus analogue of the regular subset of pure-*CCS*. Processes are terms without free occurrences of process variables and will be ranged over by $P,\ Q,\ \ldots$.

Bound names and bound variables induce the notion of α-equivalence as usual. In the sequel we will not distinguish between α-equivalent terms and will use $\equiv$ for both syntactical equality and α-equivalence.

Name substitutions, ranged over by σ and δ, are mappings from $\mathcal{N}$ to $\mathcal{N}$ that are identity almost everywhere. $[\tilde{y}/\tilde{x}]$ is the name substitution sending $\tilde{x}$ to $\tilde{y}$ and is identity otherwise. If $\sigma = [\tilde{y}/\tilde{x}]$ then $n(\sigma) = \{\tilde{y}\} \cup \{\tilde{x}\}$. If S is a set of expressions then $S\sigma = \{\, e\sigma \mid e \in S \,\}$. Name substitutions on actions are defined as usual. We will take the liberty to identify $((\tilde{x})T)\tilde{w}$ with $T[\tilde{w}/\tilde{x}]$, the term resulted by substituting $\tilde{w}$ for $\tilde{x}$ in T.

A *process substitution* is a partial function from process variables to abstractions. $[F_i/X_i|1 \le i \le m]$ is a process substitution that sends X_i to F_i for $1 \le i \le m$. We will often simply write $T[F_i/X_i|i]$ when the range of i is clear from the context. Substitutions are postfix operators, and have higher precedence than the operators in the language. In name and process substitutions bound names and bound variables will be renamed when necessary to avoid capture. We will not go to technical details of defining these syntactical substitutions, and only list below some properties that substitutions ought to satisfy:

> Let $\{\, X_i \mid 1 \le i \le m \,\}$ and $\{\, Y_j \mid 1 \le j \le n \,\}$ be different process variables. Then
>
> $$T[F_i/X_i|i][G_j/Y_j|j] \equiv T[F_i[G_j/Y_j|j], G_j/X_i, Y_j|i,j]$$
>
> In particular, if $\{\, Y_j \mid 1 \le j \le n \,\} \cap FV(T) = \emptyset$ then
>
> $$T[F_i/X_i|i][G_j/Y_j|j] \equiv T[F_i[G_j/Y_j|j]/X_i|i]$$
>
> As abstractions are name-closed, process substitution communicates with parameter passing:
>
> $$(F[G/X])\tilde{x} = (F\tilde{x})[G/X]$$

For the bulk of this paper we will concentrate on *late* bisimulation, and we will sketch in a later section how these results can be carried over to *early* bisimulation in a systematic manner. We refer to [MPW92] for the definitions of transitional semantics, ground bisimulation ($\dot{\sim}$) and bisimulation congruence ($\sim$). Here we only give the definitions for symbolic transitional semantics and symbolic bisimulation.

We write $\sigma \models \phi$ if σ satisfies ϕ in the usual sense. $\phi \Rightarrow \psi$ means $\sigma \models \phi$ implies $\sigma \models \psi$ for any substitution σ.

Proposition 2.1 *The relation $\phi \Rightarrow \psi$ is decidable.*

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

ψ is a *maximally consistent extension* of ϕ on V, written $\psi \in MCE_V(\phi)$, if $\psi \models \phi$ and ψ 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.

Lemma 2.2 $\bigvee MCE_V(\phi) = \phi$

Proposition 2.3 *Suppose ϕ is maximally consistent on V. If σ and σ' both satisfy ϕ, then $\sigma =^V \sigma'\delta$ for some injective substitution δ.*

Lemma 2.4

1. *If $\phi \Rightarrow \psi$ and σ is injective on $fn(\phi, \psi)$, then $\phi\sigma \Rightarrow \psi\sigma$.*
2. *If $\psi \in MCE_V(\phi)$ and σ is injective on $V \cup n(\phi, \psi)$, then $\psi\sigma \in MCE_{V\sigma}(\phi\sigma)$.*
3. *If σ is injective on V then $(MC_V)\sigma = MC_{V\sigma}$.*

The symbolic transitional semantics of the π-calculus is given in Figure 1, where the symmetric rules of Sum and Par have been omitted. ν_x is an operation on conditions defined thus ([BD94]):

$$\begin{array}{ll} \nu_x true = true & \nu_x[x = x] = true \\ \nu_x[x = y] = false & \nu_x[y = z] = [y = z] \\ \nu_x \neg\phi = \neg\nu_x\phi & \nu_x(\phi \wedge \psi) = \nu_x\phi \wedge \nu_x\psi \end{array}$$

Clearly $x \notin n(\nu_x\phi)$.

Lemma 2.5 $\nu_x\phi \wedge (\bigwedge_{y \in n(\phi)} x \neq y) \Rightarrow \phi$.

In the following definition of symbolic bisimulation we use $\alpha =^\phi \beta$ to mean

$$\begin{array}{l} \text{if } \alpha \equiv \tau \text{ then } \beta \equiv \tau \\ \text{if } \alpha \equiv \overline{a}x \text{ then } \beta \equiv \overline{b}y \ \text{ and } \ \phi \Rightarrow a = b,\ \phi \Rightarrow x = y \\ \text{if } \alpha \equiv \overline{a}(x) \text{ then } \beta \equiv \overline{b}(x) \ \text{ and } \ \phi \Rightarrow a = b \\ \text{if } \alpha \equiv a(x) \text{ then } \beta \equiv b(x) \ \text{ and } \ \phi \Rightarrow a = b \end{array}$$

$$\text{Pre}\quad \frac{}{\alpha.P \xrightarrow{true,\alpha} P} \qquad\qquad \text{Cond}\quad \frac{P \xrightarrow{\phi,\alpha} P'}{\psi P \xrightarrow{\phi\wedge\psi,\alpha} P'}$$

$$\text{Sum}\quad \frac{P \xrightarrow{\phi,\alpha} P'}{P+Q \xrightarrow{\phi,\alpha} P'} \qquad\qquad \text{Par}\quad \frac{P \xrightarrow{\phi,\alpha} P'}{P \mid Q \xrightarrow{\phi,\alpha} P' \mid Q}\ \ bn(\alpha)\cap fn(Q)=\emptyset$$

$$\text{Res}\quad \frac{P \xrightarrow{\phi,\alpha} P'}{\nu x P \xrightarrow{\nu_x\phi,\alpha} \nu x P'}\ \ x\notin n(\alpha) \qquad \text{Com}\quad \frac{P \xrightarrow{\phi,a(x)} P' \quad Q \xrightarrow{\psi,\overline{b}y} Q'}{P \mid Q \xrightarrow{\phi\wedge\psi\wedge a=b,\tau} P'[y/x] \mid Q'}$$

$$\text{Open}\quad \frac{P \xrightarrow{\phi,\overline{a}x} P'}{\nu x P \xrightarrow{\nu_x\phi,\overline{a}(x)} P'}\ \ x\neq a \qquad \text{Close}\quad \frac{P \xrightarrow{\phi,a(x)} P' \quad Q \xrightarrow{\psi,\overline{b}(x)} Q'}{P \mid Q \xrightarrow{\phi\wedge\psi\wedge a=b,\tau} \nu x(P' \mid Q')}$$

$$\text{Rec}\quad \frac{F\tilde{v}[\mathbf{fix}XF/X] \xrightarrow{\phi,\alpha} P'}{(\mathbf{fix}XF)\tilde{v} \xrightarrow{\phi,\alpha} P'}$$

Figure 1: π-Calculus Symbolic Transitional Semantics

Definition 2.6 A condition indexed family of symmetric relations $\mathcal{S} = \{S^\phi\}$ is a late symbolic bisimulation if $(P,Q) \in S^\phi$ implies for each $\psi \in MCE_{fn(P,Q)}(\phi)$ whenever $P \xrightarrow{\phi',\alpha} P'$ with $bn(\alpha) \cap fn(P,Q) = \emptyset$ and $\psi \Rightarrow \phi'$, then there is a $Q \xrightarrow{\psi',\beta} Q'$ such that $\psi \Rightarrow \psi'$, and $(P',Q') \in S^{\psi''}$ where

$$\psi'' = \begin{cases} \psi \cup \{\, x \neq y \mid y \in fn(\alpha.P', \beta.Q')\,\} & \text{if } \alpha \equiv \overline{a}(x) \\ \psi & \text{otherwise} \end{cases}$$

Let $\sim_L$ be the largest late symbolic bisimulation.
For abstractions, $F \sim_L^{true} G$ iff $F\tilde{x} \sim_L^{true} G\tilde{x}$ for any $\tilde{x}$.
For process expressions T, U with free process variables in $\tilde{X}$, $T \sim_L^\phi U$ iff $T[\tilde{F}/\tilde{X}] \sim_L^\phi U[\tilde{F}/\tilde{X}]$ for any vector $\tilde{F}$ of variable-free abstractions. □

Symbolic bisimulation is preserved by injective substitutions:

Proposition 2.7 *If $T \sim_L^\phi U$ and σ is injective then $T\sigma \sim_L^{\phi\sigma} U\sigma$*

The soundness and completeness of $\sim_L$ with respect to $\sim$ is stated by the following theorem:

Theorem 2.8 *$T \sim_L^\phi U$ iff $T\sigma \stackrel{\cdot}{\sim} U\sigma$ for any $\sigma \models \phi$.*

3 The Inference System

The inference system for late symbolic bisimulation consists of a set of inference rules (Figure 3), together with the standard equations for choice (S1 – S4) and

$$\text{ALPHA} \quad \frac{}{true \rhd T = U} \quad T \equiv U$$

$$\text{AXIOM} \quad \frac{}{true \rhd T = U} \quad T = U \text{ is an axiom instance}$$

$$\text{CHOICE} \quad \frac{\phi \rhd T_i = U_i}{\phi \rhd T_1 + T_2 = U_1 + U_2}$$

$$\text{TAU} \quad \frac{\phi \rhd T = U}{\phi \rhd \tau.T = \tau.U}$$

$$\text{L-INPUT} \quad \frac{\phi \rhd T = U}{\phi \rhd a(x).T = b(x).U} \quad \phi \Rightarrow a = b, \quad x \notin n(\phi)$$

$$\text{OUTPUT} \quad \frac{\phi \rhd T = U}{\phi \rhd \overline{a}x.T = \overline{b}y.U} \quad \phi \Rightarrow a = b,\ \phi \Rightarrow x = y$$

$$\text{COND} \quad \frac{\phi \wedge \psi \rhd T = U \qquad \phi \wedge \neg\psi \rhd \mathbf{0} = U}{\phi \rhd \psi T = U}$$

$$\text{RES} \quad \frac{\phi \wedge \{\, x \neq y \mid y \in fn(\nu x T, \nu x U)\,\} \rhd T = U}{\phi \rhd \nu x T = \nu x U} \quad x \notin n(\phi)$$

$$\text{ABS} \quad \frac{true \rhd T = U}{true \rhd (\tilde{x})T = (\tilde{x})U}$$

$$\text{APP} \quad \frac{true \rhd F = G}{true \rhd F\tilde{w} = G\tilde{w}}$$

$$\text{CONGR-}\mathbf{fix} \quad \frac{true \rhd F = G}{true \rhd \mathbf{fix}XF = \mathbf{fix}XG}$$

$$\text{REC} \quad \frac{}{true \rhd \mathbf{fix}XF = F[\mathbf{fix}XF/X]}$$

$$\text{UFI} \quad \frac{true \rhd F = G[F/X]}{true \rhd F = \mathbf{fix}XG}$$

$$\text{CUT} \quad \frac{\phi_1 \rhd T = U \qquad \phi_2 \rhd T = U}{\phi \rhd T = U} \quad \phi \Rightarrow \phi_1 \vee \phi_2$$

$$\text{ABSURD} \quad \frac{}{false \rhd T = U}$$

Figure 2: The Inference Rules for Late Symbolic Bisimulation

restriction (R1 – R6) (Figure 3). To deal with parallel composition we also need the expansion law which can be found in [PS93, Lin94]. There are two kinds of judgments: conditional behaviour equations of the form

$$\phi \rhd T = U$$

S1	$X + \mathbf{0} = X$	S2	$X + X = X$
S3	$X + Y = Y + X$	S4	$(X + Y) + Z = X + (Y + Z)$
R1	$\nu x \mathbf{0} = \mathbf{0}$	R2	$\nu x \alpha.X = \alpha.\nu x X \quad x \notin n(\alpha)$
R3	$\nu x \alpha.X = \mathbf{0} \quad x$ is the port of α	R4	$\nu x \nu y X = \nu y \nu x X$
R5	$\nu x(X + Y) = \nu x X + \nu x Y$	R6	$\nu x X(\tilde{w}) = X(\tilde{w}) \quad x \notin \{\tilde{w}\}$

Figure 3: The Axioms for Choice And Restriction

and unconditional abstraction equations of the form

$$true \rhd F = G$$

Rules ABS and APP are employed to switch between these two kinds of judgments.

We shall write $\vdash \phi \rhd T = U$ if $\phi \rhd T = U$ can be derived from this inference system.

Lemma 3.1 *Suppose $n(\psi) \subseteq fn(T, U)$.*

1. *If $\phi \Rightarrow \psi$ and $\vdash \phi \rhd T = U$ then $\vdash \phi \rhd \psi T = U$.*
2. *If $\phi \wedge \psi \Rightarrow false$ then $\vdash \phi \rhd \psi T = \mathbf{0}$.*
3. *$\vdash \psi(T + U) = \psi T + \psi U$.*
4. *$\vdash \nu x(\psi T) = (\nu_x \psi) \nu x T$.*

Proposition 3.2 *Suppose $n(\psi, \phi) \cup fn(T, U) \subseteq V$. If $\vdash \psi \rhd T = U$ for each $\psi \in MCE_V(\phi)$ then $\vdash \phi \rhd T = U$.*

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

Proposition 3.3 *Suppose $\phi \Rightarrow a = b$, $x \notin n(\phi)$. If $\vdash \phi \cup \{ x \neq y \mid y \in fn(\overline{a}(x).T, \overline{b}(x).U) \} \rhd T = U$ then $\vdash \phi \rhd \overline{a}(x).T = \overline{b}(x).U$.*

Lemma 3.4 *Suppose $fn(T, \phi) \subseteq \{\tilde{x}\}$. Then $\vdash (\mathbf{fix} X(\tilde{x})^{\phi} T)\tilde{x} = \phi(\mathbf{fix} X(\tilde{x})^{\phi} T)\tilde{x}$*

Proposition 3.5 *Suppose $fn(T, U, \phi) \subseteq \{\tilde{x}\}$ and X is guarded in T. Then from*

$$\vdash \phi \rhd U = T[(\tilde{x})^{\phi} U / X]$$

infer

$$\vdash \phi \rhd U = (\mathbf{fix} X(\tilde{x})^{\phi} T)\tilde{x}$$

Proposition 3.6 *If $\vdash \phi \rhd T = U$ then $\vdash \phi\sigma \rhd T\sigma = U\sigma$ for any injective name substitution σ.*

All except ABS, APP, CONGR-**fix**, REC and UFI of the inference rules have already appeared in [Lin94], and their soundness was shown there. The soundness of ABS, APP and REC is not difficult to see. The following proposition can be used to show UFI is sound. Its proof uses the "bisimulation upto" technique ([MPW92]). Recently Sangiorgi gave a simpler proof which employs a technique based on *respectful functions* on relations ([San94]). The soundness of CONGR-**fix** can be established in a similar way.

Proposition 3.7 *Suppose X is guarded in G. If $F_1 \sim_L^{true} G[F_1/X]$ and $F_2 \sim_L^{true} G[F_2/X]$ then $F_1 \sim_L^{true} F_2$.*

Theorem 3.8 *(Soundness of $\vdash$) If $\vdash \phi \rhd T = U$ then $T \sim_L^{\phi} U$.*

4 Completeness of $\vdash$

This section is devoted to establish the completeness result for $\vdash$. As mentioned in Section 2, the completeness result holds only for a subset of the π-calculus and we will confine ourselves with the fragment without the parallel operator $|$.

The proof of the completeness result consists of three steps: First, Proposition 4.1 shows that any process expression can be proved to satisfy a standard set of equations. Next, Proposition 4.2 demonstrates if two process expressions are symbolically bisimilar than an equation set can be constructed which is provably satisfied by both. Finally, Proposition 4.3 shows that two process expressions satisfying the same set of equations can be proved equal.

For technical convenience we shall assume that in an application $F\tilde{x}$ the name parameters in $\tilde{x}$ are distinct. This does not present any restriction to our result because an application with duplicate name parameters can always be replaced by one with fewer but distinct parameters.

4.1 Standard Equations

Let $\tilde{X} = \{X_1, X_2, ..., X_m\}$ and $\tilde{W} = \{W_1, W_2, ..., W_n\}$ be disjoint sets of process variables. Then

$$E : X_i(\tilde{x}_i) = H_i \qquad 1 \leq i \leq m,$$

where $fn(H_i) \subseteq \{\tilde{x}_i\}$ and $FV(H_i) \subseteq \tilde{X} \cup \tilde{W}$, is a set of equations with formal process variables in $\tilde{X}$ and free process variables in $\tilde{W}$. E is *standard* if each H_i has the form

$$\sum_{\psi_{ik} \in MC_{\{\tilde{x}\}}} \psi_{ik}\Big(\sum_{p \in P_{ik}} \alpha_{ikp}.X_{f(i,k,p)}(\tilde{x}_{ikp}) + \sum_{p' \in P'_{ik}} \nu \tilde{w}'_{ikp'} W_{f'(i,k,p')}(\tilde{w}_{ikp'})\Big)$$

where $\tilde{w}'_{ikp'} \subseteq \tilde{w}_{ikp'}$, $\alpha_{ikp} \in \{\tau, a(x), \overline{a}x, \overline{a}(x)\}$. A process expression T provably $\tilde{\phi}$-satisfies an equation set E if there exist T_i with $fn(T_i) \subseteq \{\tilde{x}_i\}$, $1 \leq i \leq m$ and $T_1 \equiv T$ s.t.

$$\vdash \phi_i \rhd T_i = H_i[(\tilde{x}_k)^{\phi_k} T_k / X_k | 1 \leq k \leq m] \qquad 1 \leq i \leq m$$

We will call T_1 the *leading equation* of E. We will simply say "T provably satisfies E" when $\phi_i \equiv true$ for all i.

Proposition 4.1 *Any guarded process expression T with free process variables in $\tilde{W}$ provably satisfies a standard equation set E with free process variables in $\tilde{W}$.*

Proof: (Sketch) Given any term, using axioms R1 – R6 and Lemma 3.1.*4* we can push restrictions inwards, until they either give rise to bound outputs, or disappear, or get in front of process variables with the restricted names occuring in their actual argument lists. We assume T is already in such form.

We first show, by structural induction, that T provably satisfies a set of pre-standard equations of the form

$$X_i(\tilde{x}_i) = \sum_j \phi_{ij}\alpha_{ij}.X_{f(i,j)}(\tilde{x}_{ij}) + \sum_k \phi'_{ik}\nu\tilde{w}'_{ik}W_{f'(i,k)}(\tilde{w}_{ik})$$

where $\tilde{w}'_{ij} \subseteq \tilde{w}_{ij}$.

Next we show that each pre-standard equation set can be transformed into a standard one. Take a typical pre-standard equation

$$X(\tilde{x}) = \sum_{i\in I} \phi_i\alpha_i.(\tilde{x}'_i)X_{f(i)}(\tilde{x}_i) + \sum_{j\in J} \phi'_j\nu\tilde{w}'_jW_{f'(j)}(\tilde{w}'_j)$$

For each $\psi_k \in MC_{\{\tilde{x}\}}$ let $P_k = \{\, p \mid p \in I,\ \psi_k \Rightarrow \phi_p \,\}$ and $P'_k = \{\, p' \mid p' \in J,\ \psi_k \Rightarrow \phi'_{p'} \,\}$. It is straightforward to verify

$$\vdash X(\tilde{x}) = \sum_{\psi_k\in MC_{\{\tilde{x}\}}} \psi_k(\sum_{p\in P_k} \alpha_p. \sum_{p'\in P'_k} \nu\tilde{w}'_{p'}W_{f'(p')}(\tilde{w}_{p'}))$$

$\Box$

Proposition 4.2 *Suppose T, U are guarded process expressions with free process variables in $\{\tilde{W}\}$. If $T \sim^{\phi}_{L} U$ then there exist a set of standard equations E with free process variables in $\{\tilde{W}\}$ and a vector of booleans $\tilde{\phi}$ with $\phi_1 \equiv \phi$ such that both T and U provably $\tilde{\phi}$-satisfy E.*

Proof: (Sketch) For simplicity we only consider the case $\{\tilde{W}\} = \emptyset$ here.

Let E_1, E_2 be the standard equation sets provably satisfied by T, U, respectively:

$$\begin{array}{lll} E_1: & X_i(\tilde{x_i}) = G_i & 1 \le i \le m \\ E_2: & Y_j(\tilde{y_j}) = H_j & 1 \le j \le n \end{array}$$

where

$$\begin{array}{ll} G_i \equiv \sum_{k\in K_i} \phi'_{ik}G_{ik} & G_{ik} \equiv \sum_{p\in P_{ik}} \alpha'_{ikp}.X_{f(i,k,p)}(\tilde{x}_{ikp}) \\ H_j \equiv \sum_{l\in L_j} \psi'_{jl}H_{jl} & H_{jl} \equiv \sum_{q\in Q_{jl}} \beta'_{jlq}.Y_{g(j,l,q)}(\tilde{y}_{jlq}) \end{array}$$

We assume in E_1, E_2 all input prefixes use the same name z which is different from any other names used so far.

So there exist T_i' with $fn(T_i) \subseteq \tilde{x}_i,\ T_1' \equiv T$, and U_j' with $fn(U_j) \subseteq \tilde{y}_j',\ U_1' \equiv U$, such that

$$\begin{aligned} \vdash T_i' &= G_i[(\tilde{x}_k)T_k'/X_k \mid 1 \le k \le m] && (3)\\ \vdash U_j' &= H_j[(\tilde{y}_l)U_l'/Y_l \mid 1 \le l \le n] && (4)\end{aligned}$$

For each $i,\ j$, let $\tilde{z}_{ij}$ be a vector of new names with $\tilde{z}_{11} = \tilde{x}_1 = \tilde{y}_1$. Let also

$$\begin{array}{ll} T_{ij} = T_i'[\tilde{z}_{ij}/\tilde{x}_i] & U_{ij} = U_j'[\tilde{z}_{ij}/\tilde{y}_j] \\ T_{(ikpj)} = T'_{f(i,k,p)}[\tilde{x}_{ikp}[\tilde{z}_{ij}/\tilde{x}_i]/\tilde{x}_{f(i,k,p)}] & U_{(jlqi)} = U'_{g(j,l,q)}[\tilde{y}_{jlq}[\tilde{z}_{ij}/\tilde{y}_j]/\tilde{y}_{g(j,l,q)}] \\ \phi_{ik} = \phi'_{ik}[\tilde{z}_{ij}/\tilde{x}_i] & \psi_{jl} = \psi'_{jl}[\tilde{z}_{ij}/\tilde{y}_j] \\ \alpha_{ikp} = \alpha'_{ikp}[\tilde{z}_{ij}/\tilde{x}_i] & \beta_{jlq} = \beta'_{jlq}[\tilde{z}_{ij}/\tilde{y}_j] \end{array}$$

From (3) and (4) we have

$$\begin{aligned} \vdash T_{ij} &= \sum_{k \in K_i} \phi_{ik} T_{ijk} && (5)\\ \vdash U_{ij} &= \sum_{l \in L_j} \psi_{jl} U_{ijl} && (6)\end{aligned}$$

where

$$\begin{array}{l} T_{ijk} \equiv G_{ik}[(\tilde{x}_k)T_k'/X_k|k][\tilde{z}_{ij}/\tilde{x}_i] \equiv \sum_{p \in P_{ik}} \alpha_{ikp}.T_{(ikpj)} \\ U_{ijl} \equiv H_{jl}[(\tilde{y}_l)U_l'/Y_l|l][\tilde{z}_{ij}/\tilde{y}_j] \equiv \sum_{q \in Q_{jl}} \beta_{jlq}.U_{(jlqi)} \end{array}$$

Since $\{ \phi'_{ik} \mid k \in K_i \} = MC_{\{\tilde{x}_i\}}$, $\{ \psi'_{jl} \mid l \in L_j \} = MC_{\{\tilde{y}_i\}}$, and $[\tilde{z}_{ij}/\tilde{x}_i]$, $[\tilde{z}_{ij}/\tilde{y}_j]$ are injective on $\tilde{x}_i,\ \tilde{y}_j$, respectively, by Lemma 2.4 $\{ \phi_{ik} \mid k \in K_i \} = MC_{\{\tilde{z}_{ij}\}} = \{ \psi_{jl} \mid l \in L_j \}$.

Set $\Phi_{ij} = \{ \psi \in MC_{\tilde{z}_{ij}} \mid T_{ij} \sim_L^{\psi} U_{ij} \}$ and $\phi_{ij} = \bigvee \Phi_{ij}$. Then $T_{ij} \sim_L^{\phi_{ij}} U_{ij}$ and $\phi' \Rightarrow \phi_{ij}$ for any ϕ' s.t. $T_{ij} \sim_L^{\phi'} U_{ij}$.

Note that $\Phi_{ij} \subseteq MC_{\{\tilde{z}_{ij}\}}$, so for each $\psi \in \Phi_{ij}$ there is a unique $k \in K_i$ and unique $l \in L_j$ s.t. $\psi = \phi_{ik} = \psi_{jl}$. For easy reference we write such ψ as ψ_{ijkl}.

From (5) and (6) it follows

$$\begin{aligned} \vdash \phi_{ij} \rhd T_{ij} &= \sum_{\psi_{ijkl} \in \Phi_{ij}} \psi_{ijkl} T_{ijk} && (7)\\ \vdash \phi_{ij} \rhd U_{ij} &= \sum_{\psi_{ijkl} \in \Phi_{ij}} \psi_{ijkl} U_{ijl} && (8)\end{aligned}$$

Also for each $\psi_{ijkl} \in \Phi_{ij}$, $T_{ijk} \sim_L^{\psi_{ijkl}} U_{ijl}$ because $T_{ij} \sim_L^{\psi_{ijkl}} U_{ij}$.

Now we group the summands of T_{ijk} into T^{τ}, $T^{FO_{[d]}}$, $T^{BO_{[d]}}$ and $T^{IN_{[d]}}$, where $[d]$ ranges over the equivalence classes on $\{\tilde{z}_{ij}\}$ determined by ψ_{ijkl}:

$$\begin{array}{ll} T^{\tau} = \displaystyle\sum_{\alpha_{ikp} \equiv \tau} \tau.T_{(ikpj)} & T^{FO_{[d]}} = \displaystyle\sum_{\alpha_{ikp} \equiv \overline{a}w,\ a \in [d]} \overline{a}w.T_{(ikpj)} \\ T^{BO_{[d]}} = \displaystyle\sum_{\alpha_{ikp} \equiv \overline{a}(z),\ a \in [d]} \overline{a}(z).T_{(ikpj)} & T^{IN_{[d]}} = \displaystyle\sum_{\alpha_{ikp} \equiv a(z),\ a \in [d]} a(z).T_{(ikpj)} \end{array}$$

Similarly define U^{τ}, $U^{FO_{[d]}}$, $U^{BO_{[d]}}$ and $U^{IN_{[d]}}$. Then

$$\begin{array}{l} \vdash T_{ijk} = T^{\tau} + \sum_{[d]} T^{FO_{[d]}} + \sum_{[d]} T^{BO_{[d]}} + \sum_{[d]} T^{IN_{[d]}} \\ \vdash U_{ijl} = U^{\tau} + \sum_{[d]} U^{FO_{[d]}} + \sum_{[d]} U^{BO_{[d]}} + \sum_{[d]} U^{IN_{[d]}} \end{array}$$

We have $T^{\tau} \sim_L^{\psi_{ijkl}} U^{\tau}$, $T^{FO_{[d]}} \sim_L^{\psi_{ijkl}} U^{FO_{[d]}}$, $T^{BO_{[d]}} \sim_L^{\psi_{ijkl}} U^{BO_{[d]}}$ and $T^{IN_{[d]}} \sim_L^{\psi_{ijkl}} U^{IN_{[d]}}$, for each $[d]$.

Define

$$I^{\tau}_{ijkl} = \{\, (p,q) \mid \alpha_{ikp} \equiv \tau,\ \beta_{jlq} \equiv \tau,\ T_{(ikpj)} \sim_L^{\psi_{ijkl}} U_{(jlqi)} \,\}$$

$$I^{FO_{[d]}}_{ijkl} = \{\, (p,q) \mid \begin{array}{l} \alpha_{ikp} \equiv \overline{a}w,\ a,b \in [d], \\ \beta_{jlq} \equiv \overline{b}v,\ \psi_{ijkl} \Rightarrow w = v, \end{array} \quad T_{(ikpj)} \sim_L^{\psi_{ijkl}} U_{(jlqi)} \,\}$$

$$I^{BO_{[d]}}_{ijkl} = \{\, (p,q) \mid \begin{array}{l} \alpha_{ikp} \equiv \overline{a}(z) \\ \beta_{jlq} \equiv \overline{b}(z) \end{array} \ a,b \in [d], T_{(ikpj)} \sim_L^{\psi_{ijkl} \cup \{\, z' \neq z \mid z' \in \{\tilde{z}_{ij}\}\,\}} U_{(jlqi)} \,\}$$

$$I^{IN_{[d]}}_{ijkl} = \{\, (p,q) \mid \alpha_{ikp} \equiv a(z),\ \beta_{jlq} \equiv b(z),\ a,b \in [d],\ T_{(ikpj)} \sim_L^{\psi_{ijkl}} U_{(jlqi)} \,\}$$

We may assume $\tilde{x}_{ikp}[\tilde{z}_{ij}/\tilde{x}_i] = \tilde{y}_{jlq}[\tilde{z}_{ij}/\tilde{y}_j]$. Note that each of the four relations defined above is surjective in both the first and the second components.

Let $\{\, Z_{ij} \mid 1 \leq i \leq m,\ 1 \leq j \leq n \,\}$ be a set of new process variables. Let also $\tilde{z}_{ikpjlq} = \tilde{x}_{ikp}[\tilde{z}_{ij}/\tilde{x}_i] = \tilde{y}_{jlq}[\tilde{z}_{ij}/\tilde{y}_j]$ so that

$$\tilde{z}_{ikpjlq}[\tilde{x}_i/\tilde{z}_{ij}] = \tilde{x}_{ikp} \qquad \tilde{z}_{ikpjlq}[\tilde{y}_j/\tilde{z}_{ij}] = \tilde{y}_{jlq} \tag{9}$$

Consider the equation set

$$E:\ Z_{ij}(\tilde{z}_{ij}) = \sum_{\psi_{ijkl} \in \Phi_{ij}} \psi_{ijkl}(V^{\tau} + \sum_{[d]} V^{FO_{[d]}} + \sum_{[d]} V^{BO_{[d]}} + \sum_{[d]} V^{IN_{[d]}})$$

where

$$\begin{array}{lcl} V^{\tau} & \equiv & \sum_{(p,q) \in I^{\tau}_{ijkl}} \tau.Z_{f(i,k,p)g(j,l,q)}(\tilde{z}_{ikpjlq}) \\ V^{FO_{[d]}} & \equiv & \sum_{(p,q) \in I^{FO_{[d]}}_{ijkl}} \alpha_{ikp}.Z_{f(i,k,p)g(j,l,q)}(\tilde{z}_{ikpjlq}) \\ V^{BO_{[d]}} & \equiv & \sum_{(p,q) \in I^{BO_{[d]}}_{ijkl}} \alpha_{ikp}.Z_{f(i,k,p)g(j,l,q)}(\tilde{z}_{ikpjlq}) \\ V^{IN_{[d]}} & \equiv & \sum_{(p,q) \in I^{IN_{[d]}}_{ijkl}} \alpha_{ikp}.Z_{f(i,k,p)g(j,l,q)}(\tilde{z}_{ikpjlq}) \end{array}$$

Then it can be shown that$E$ is provably $\{\, \phi_{ij} \mid i,j \,\}$-satisfied by T when each Z_{ij} is instantiated with $(\tilde{z}_{ij})T_{ij}$ over ϕ_{ij}, and provably $\{\, \phi_{ij} \mid i,j \,\}$-satisfied by U when each Z_{ij} is instantiated with $(\tilde{z}_{ij})U_{ij}$ over ϕ_{ij}. $\square$

Proposition 4.3 *Let $E : X_i(\tilde{x}_i) = H_i,\ 1 \le i \le m$, be a guarded equation set with free process variables in $\tilde{W}$. If T, U both provably $\tilde{\phi}$-satisfy E then $\vdash \phi_1 \rhd T = U$.*

Proof: (Sketch) By induction on m. The base case $m = 1$ can be settled easily by Proposition 3.5.

Assume the result for m and let E contain $m+1$ equations. Since T provably $\tilde{\phi}$-satisfies E, there exist $T_i,\ 1 \le i \le m+1$, with free variables in $\tilde{W}$ and $T_1 \equiv T$ s.t.

$$\vdash \phi_i \rhd T_i = H_i[(\tilde{x}_i)^{\phi_i} T_i / X_i | 1 \le i \le m+1]$$

Since X_j:s do not appear in any T_i, we have

$$\begin{aligned} \vdash \phi_{m+1} \rhd T_{m+1} &= H_{m+1}[(\tilde{x}_i)^{\phi_i} T_i / X_i | 1 \le i \le m+1] \\ &= H_{m+1}[(\tilde{x}_i)^{\phi_i} T_i / X_i | 1 \le i \le m][(\tilde{x}_{m+1})^{\phi_{m+1}} T_{m+1} / X_{m+1}] \end{aligned}$$

By Proposition 3.5

$$\vdash \phi_{m+1} \rhd T_{m+1} = (\mathbf{fix} X_{m+1}(\tilde{x}_{m+1})^{\phi_{m+1}} H_{m+1}[(\tilde{x}_i^{\phi_i} T_i / X_i | 1 \le i \le m])\tilde{x}_{m+1}$$

Now let $G_{m+1} \equiv (\mathbf{fix} X_{m+1}(\tilde{x}_{m+1})^{\phi_{m+1}} H_{m+1})\tilde{x}_{m+1}$ and define

$$G_i \equiv H_i[(\tilde{x}_{m+1})^{\phi_{m+1}} G_{m+1} / X_{m+1}],\ 1 \le i \le m$$

Then T provably $\{\phi_1, ..., \phi_m\}$-satisfies the m equation

$$E' :\ X_i(\tilde{x}_i) = G_i \qquad 1 \le i \le m$$

Similarly U provably $\{\phi_1, ..., \phi_m\}$-satisfies E'. By the induction hypothesis, $\vdash \phi_1 \rhd T = U$. $\square$

Combining Propositions 4.1, 4.2 and 4.3 gives the main theorem of the paper:

Theorem 4.4 *(Completeness of $\vdash$) Let T, U be guarded process expressions. Then $T \sim_L^{\phi} U$ implies $\vdash \phi \rhd T = U$.*

5 The Early Case

Early symbolic bisimulation is obtained by changing the clause for input transition in Definition 2.6:

> whenever $T \xrightarrow{\phi', a(x)}_L T'$ with $x \notin fn(T, U)$, then for each $\psi \in MCE_{fn(T,U)\cup\{x\}}(\phi \cup \phi')$ there is a $U \xrightarrow{\psi', b(x)}_L U'$ such that $\phi' \Rightarrow \psi'$, $\psi \models a = b$, and $(T', U') \in S^{\psi}$

Let $\sim_E$ be the largest early symbolic bisimulation.

With these definitions the early version of Proposition 2.8 still holds.

The proof system for early bisimulation can be obtained by replacing the L-INPUT rule in Figure 3 with the following one

$$\text{E-INPUT} \quad \frac{\phi \rhd \sum_{i \in I} \tau.T_i = \sum_{j \in J} \tau.U_j}{\phi \rhd \sum_{i \in I} a_i(x).T = \sum_{j \in J} b_j(x).U} \quad \begin{array}{l} \phi \Rightarrow a_i = b_j,\ i \in I,\ j \in J \\ x \notin n(\phi) \end{array}$$

Let us write $\vdash_E \phi \rhd T = U$ to mean $\phi \rhd T = U$ can be derived from the new proof system. The soundness of E-INPUT is not difficult to see. For the completeness proof, Lemma 4.1 and Proposition 4.3 still hold. For Proposition 4.2, only the input case of the proof needs modifying. We omit the details.

Theorem 5.1 *(Completeness of $\vdash_E$) Let T, U be guarded process expressions. Then $T \sim_E^{\phi} U$ implies $\vdash_E \phi \rhd T = U$.*

6 Conclusions

We have presented sound proof systems for both early and late bisimulation equivalences in the π-calculus with recursion. These systems are also complete when restricted to the sublanguage where the parallel composition does not appear under recursion. These results on the one hand extend the proof systems proposed in [Lin94] for recursion-free π-calculus to infinite processes, and on the other hand extend the inference system of [Mil84] for regular pure-*CCS* to π-calculus. The key rule to deal with recursion is unique fixpoint induction which generalises that for pure-*CCS* to the setting of π-calculus. Syntactical simplicity has been achieved by introducing abstraction to form recursive expressions and application to realise parameter passing.

The version of the π-calculus we work with in this paper includes the *mismatch* construction. Mismatch has been used in [Hen91, BD92, PS93, BD94] for the purpose of complete axiomatisation for testing and bisimulation equivalences. In [Lin94, Lin95] we managed to exclude mismatch from the calculus by using it *only in the meta-language*: in any statement $C \rhd T = U$ only the condition C may contain mismatch. We failed to keep such separation of matches (which consist of only name equalities and can appear in process expressions) and conditions (which may contain name inequalities and can only be used in the judgments of the proof system or as indexes in symbolic bisimulations) in the present work, because in the completeness proof we need to *internalise* boolean conditions of judgments as conditions in process expressions.

The inference system of [Mil84] includes an equation $\mathbf{fix}X(X + T) = \mathbf{fix}XT$ to transform unguarded recursions into guarded ones. We believe that similar result can be obtained for the π-calculus by formulating an appropriate version of this equation. It is also interesting to see if the techniques used in this paper can be exploited to obtain complete axiomatisations for other equivalences such as *open bisimulation* [San93].

Acknowledgment Thanks to the anonymous referees for detailed comments.

References

[BD92] M. Boreale and R. DeNicola. Testing equivalence for mobile processes. In *CONCUR'92*, number 630 in Lecture Notes in Computer Science, pages 2 – 16. Springer–Verlag, 1992.

[BD94] M. Boreale and R. DeNicola. A symbolic semantics for the π-calculus. In *CONCUR'94*, Lecture Notes in Computer Science. Springer–Verlag, 1994.

[Dam94] M. Dam. On the decidability of process equivalences for the π-calculus. *Submitted*, 1994. Swedish institute of Computer Science.

[Hen91] M. Hennessy. A proof system for communicating processes with value-passing. *Formal Aspects of Computing*, 3:346–366, 1991.

[HL93] M. Hennessy and H. Lin. Proof systems for message-passing process algebras. In *CONCUR'93*, number 715 in Lecture Notes in Computer Science, pages 202–216, 1993.

[Lin94] H. Lin. Symbolic bisimulations and proof systems for the π-calculus. Report 7/94, Computer Science, University of Sussex, 1994.

[Lin95] H. Lin. Complete inference systems for weak bisimulation equivalences in the π-calculus. In *Proceedings of Sixth International Joint Conference on the Theory and Practice of Software Development*, Lecture Notes in Computer Science. Springer–Verlag, 1995.

[Mil84] R. Milner. A complete inference system for a class of regular behaviours. *J. Computer and System Science*, 28:439–466, 1984.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile proceses, part I,II. *Information and Computation*, 100:1–77, 1992.

[PS93] J. Parrow and D. Sangiorgi. Algebraic theories for name-passing calculi. Report ECS-LFCS-93-262, LFCS, University of Edinburgh, 1993.

[San93] D. Sangiorgi. A theory of bisimulation for the π-calculus. In *CONCUR'93*, number 715 in Lecture Notes in Computer Science, 1993.

[San94] D. Sangiorgi. On the bisimulation proof method. Technical Report ECS-LFCS-94-299, LFCS, Edinburgh University, 1994. Extended abstract to appear in MFCS'95.