Modal Logics for Mobile Processes

Robin Milner[*], Joachim Parrow[†] and David Walker[‡]

Abstract

In process algebras, bisimulation equivalence is typically defined directly in terms of the operational rules of action; it also has an alternative characterization in terms of a simple modal logic (sometimes called *Hennessy-Milner logic*). This paper first defines two forms of bisimulation equivalence for the π-*calculus*, a process algebra which allows dynamic reconfiguration among processes; it then explores a family of possible logics, with different modal operators. It is proven that two of these logics characterize the two bisimulation equivalences. Also, the relative expressive power of all the logics is exhibited as a lattice. The results are applicable to most value-passing process algebras.

1 Introduction

This paper presents a logical characterization of process equivalences in the π-*calculus* [6], a process algebra in which processes may change their configuration dynamically. In this introduction we place the results in context. First we review the corresponding results for process calculi which do not allow this dynamic re-configuration. Then we give plausible reasons for introducing modalities and an equality predicate into the logic, in order to extend these results to the π-calculus. In the later sections, we prove that these new connectives do indeed provide the characterization.

For a typical process algebra without mobility, the equivalence relation of strong bisimilarity [8] can be characterized by a modal process logic,

[*]University of Edinburgh, Scotland. Supported by a Senior Research Fellowship awarded by the British Science and Engineering Research Council.

[†]Swedish Institute of Computer Science, Sweden. Supported by the Swedish Board of Technical Development under project 89-01218P CONCUR and by Swedish Telecom under project PROCOM.

[‡]University of Technology, Sydney, Australia.

sometimes called Hennessy-Milner logic [2]. To be specific, let $\mathcal{P}$ consist simply of the processes P given by

$$P \quad ::= \quad \alpha.P \quad | \quad \mathbf{0} \quad | \quad P+P \quad | \quad C$$

where α ranges over *actions*, and C over *process constants.* We assume that for each C there is a *defining equation* $C \stackrel{\mathrm{def}}{=} P_C$. (Usually there will also be parallel composition and other operators, but we do not need them for this discussion.) We also assume that a labelled transition relation $\xrightarrow{\alpha}$ is defined over $\mathcal{P}$ in the usual way. Then *strong bisimilarity* is the largest symmetric relation $\sim$ over $\mathcal{P}$ for which, whenever $P \sim Q$ and $P \xrightarrow{\alpha} P'$, there exists Q' such that $Q \xrightarrow{\alpha} Q'$ and $P' \sim Q'$.

The process logic $\mathcal{PL}$ has formulae A given by

$$A \quad ::= \quad \langle\alpha\rangle A \quad | \quad \bigwedge_{i\in I} A_i \quad | \quad \neg A$$

where I stands for any denumerable set. (The smallest formula is the empty conjunction, written **true**.) $\mathcal{PL}$ is given meaning by defining the *satisfaction relation* $\models$ between processes and formulae; in particular, one defines

$$P \models \langle\alpha\rangle A \ \text{ if, for some } P',\ P \xrightarrow{\alpha} P' \text{ and } P' \models A$$

It may be shown that two processes are strongly bisimilar iff they satisfy the same formulae of $\mathcal{PL}$; this is the sense in which $\mathcal{PL}$ characterizes $\sim$. Under mild restrictions, such as when every P_C in a defining equation is *guarded* (i.e. contains no process constant except within a term of the form $\alpha.P$), only finite conjunctions in $\mathcal{PL}$ are needed.

Before considering what should be included in a logic to characterize equivalences over the π-calculus, we must discuss an issue about equivalence which arises in any *value-passing* calculus, of which the π-calculus is a rather special case. In general, in any value-passing calculus, an action α may "carry a value". By this, we mean that there are *input actions* $a(x)$, where a is a link-name and x a value variable, and x is bound in $a(x).P$; there are also *output actions* $\overline{a}e$, where e is an expression denoting a value. Such calculi have been studied in depth [3, 1], and many different equivalences have been defined over them. The choice of equivalence is complicated by the passing of values. Consider the following two processes:

$$\begin{aligned} R &= a(x).(\textbf{if } x=3 \textbf{ then } P \textbf{ else } Q) + a(x).\mathbf{0} \\ S &= a(x).(\textbf{if } x=3 \textbf{ then } P) + a(x).(\textbf{if } x \neq 3 \textbf{ then } Q) \end{aligned} \qquad (1)$$

We understand the one-armed conditional process "**if** b **then** P" to be equivalent to $\mathbf{0}$ if b is false. (The full conditional "**if** b **then** P **else** Q" can be expressed as the sum of two one-armed conditionals with conditions b and $\neg b$.) Now, is R equivalent to S? Both answers are possible.

They *are* strongly bisimilar in Milner [5], where the calculus with value-passing is reduced by translation to a value-free calculus – but with infinite sums. In fact R reduces to

$$\sum_{n\in\omega} a_n.R_n + \sum_{n\in\omega} a_n.\mathbf{0} \tag{2}$$

where $R_3 = P$, and $R_n = Q$ for $n \neq 3$. (We assume for simplicity that P and Q do not involve value-passing, so do not contain the variable x.) Correspondingly, S reduces to

$$\sum_{n\in\omega} a_n.P_n \;+\; \sum_{n\in\omega} a_n.Q_n \tag{3}$$

where $P_3 = P$ and $Q_3 = \mathbf{0}$, while $P_n = \mathbf{0}$ and $Q_n = Q$ for $n \neq 3$; this sum is equivalent to (2).

But there is a different view, according to which R and S are not equivalent. [1] In this view we do not consider R capable of an infinity of actions a_n, one for each natural number, but essentially only two actions, one of which is

$$R \;\xrightarrow{a(x)}\; \textbf{if } x = 3 \textbf{ then } P \textbf{ else } Q \tag{4}$$

yielding a family of processes indexed by the variable x. For another process to be equivalent to R, it must yield under $\xrightarrow{a(x)}$ an indexed family which is element-wise equivalent to the above family – i.e. equivalent for each value of x. But S does not have this property; it yields two indexed families, both different, namely:

$$\begin{array}{lcl} S & \xrightarrow{a(x)} & \textbf{if } x = 3 \textbf{ then } P \\ S & \xrightarrow{a(x)} & \textbf{if } x \neq 3 \textbf{ then } Q \end{array} \tag{5}$$

These two equivalences can both be expressed as forms of bisimilarity. For the π-calculus we concentrated on the second – finer – equivalence in our original paper [7], but also commented on the coarser equivalence. Both seem reasonable. In this paper we shall show that both bisimilarities can be elegantly characterized by appropriate process logics. Actually, we

[1] This view amounts to equating processes iff they denote identical *communication trees*, as defined in Milner[4], Chapter 6. The view was not pursued thoroughly there.

shall examine a family of 2^5 logics, defined by including any combination of five logical connectives – mostly modalities – over and above a fixed set of connectives. It turns out that these yield eleven equivalences (several logics being equipotent), including our two bisimilarities. We are not yet interested in most of these equivalences per se; but the lattice which they form gives insight into the power of the various logical connectives.

Now, what logical connectives should we expect in a logic for the π-calculus? Here, value expressions and value variables are themselves nothing but link-names. All computation is done with names $x, y, \ldots$; thus, input and output actions take the form $x(y)$ and $\overline{x}y$. It is natural to include some modality for each form of action; in particular, a modal formula

$$\langle x(y) \rangle A$$

for input actions where y is bound. In fact, to characterize the finer of our two bisimilarities, we shall define a modality $\langle x(y) \rangle^{\mathrm{L}}$ such that

$$P \models \langle x(y) \rangle^{\mathrm{L}} A \quad \text{iff} \quad \text{for some } P',\ P \xrightarrow{x(y)} P' \text{ and for all } z,\ P'\{z/y\} \models A\{z/y\}$$

The superscript L here stands for "late". It refers to the lateness of instantiation of the variable y; P' is chosen *first*, and then for all instances of y it must satisfy the corresponding instantiation of A. The coarser equivalence will be reflected by a modality with superscript E for "early"; this refers to the fact that the instance z of y is chosen *first*, and then a different P' may be chosen for each z.

It may be expected that, once we have included in our logic a suitable modality for each form of action, our characterization will be achieved. But this is not so, due to the special rôle of names in the π-calculus.

At first sight the π-calculus may appear to be just a degenerate form of value-passing calculus, which can then be translated (as above) to a value-free calculus, and hence characterized essentially by the logic $\mathcal{PL}$, for suitable actions α. But this neglects a crucial ingredient of π-calculus, namely the process form $(x)P$, known as *restriction*. This combinator gives *scope* to names – in other words, it allows the creation of *private* names; it is responsible for much of the power of the π-calculus, and prevents us from treating names as values in the normal way.

Thus the algebra of names cannot be "translated away" from the π-calculus, in the same way that the algebra of (say) integers can be translated away from CCS. But what is this algebra of names? It is almost empty! There are no *constant* names, and no *operators* over names; this explains why the only value expressions are names themselves (as variables). But what of boolean expressions, and the conditional form "**if** b **then** P"? Well, names have no properties except identity; thus the only *predicate* over names is equality – and indeed the π-calculus contains the *match expression*[2]

$$[x = y]P$$

which is another way of writing "**if** $x = y$ **then** P". It is therefore reasonable to expect that, by including an equality predicate in the form of a *match formula*

$$[x = y]A$$

in our logics, we succeed in characterizing the bisimilarities. This indeed turns out to be the case. Moreover, the match formula is strictly necessary; furthermore – which is not obvious – it is needed in the logic even if the match expression is omitted from the calculus.

In the next section we present the π-calculus and its operational semantics; the reader therefore need not refer to previous papers, although familiarity with the π-calculus will certainly help; we also define the two bisimilarities. In the third section we define all the logical connectives we wish to consider, and derive a complete picture for the relative power of their different combinations.

2 Mobile Processes

In this section we will recapitulate the syntax of agents from [7] and give agents two kinds of transitional semantics, corresponding to late and early instantiation of input parameters. Based on these we will define late and early bisimulation equivalences.

2.1 Syntax

Assume an infinite set $\mathcal{N}$ of *names* and let x, y, z, w, v, u range over names. We also assume a set of *agent identifiers* ranged over by C, where each agent identifier C has a nonnegative *arity* $r(C)$.

Definition 1 The set of *agents* is defined as follows (we use P, Q, R to range over agents):

[2]Hitherto we have not given much consideration to the negative form $[x \neq y]P$; it requires further investigation.

$$
\begin{array}{rcll}
P & ::= & \mathbf{0} & \text{(inaction)} \\
 & | & \overline{x}y.\,P & \text{(output prefix)} \\
 & | & x(y).\,P & \text{(input prefix)} \\
 & | & \tau.\,P & \text{(silent prefix)} \\
 & | & (y)P & \text{(restriction)} \\
 & | & [x\!=\!y]P & \text{(match)} \\
 & | & P \mid Q & \text{(composition)} \\
 & | & P + Q & \text{(summation)} \\
 & | & C(y_1, \ldots, y_{r(C)}) & \text{(defined agent)}
\end{array}
$$

In each of $x(y).\,P$ and $(y)P$ the occurrence of y in parentheses is a *binding* occurrence whose scope is P. We write $\mathrm{fn}(P)$ for the set of names occurring free in P. If $\tilde{x} = x_1, \ldots, x_n$ are distinct and $\tilde{y} = y_1, \ldots, y_n$ then $P\{\tilde{y}/\tilde{x}\}$ is the result of simultaneously substituting y_i for all free occurrences of x_i $(i = 1, \ldots, n)$ with change of bound names if necessary. Each agent constant C has a unique *defining equation* of the form

$$C(x_1, \ldots, x_{r(C)}) \stackrel{\mathrm{def}}{=} P$$

where the x_i are distinct and $\mathrm{fn}(P) \subseteq \{x_1, \ldots, x_{r(C)}\}$. $\Box$

The order of precedence among the operators is the order listed in Definition 1. For a description of the intended interpretation of agents see [6]. In examples we will frequently omit a trailing $.\mathbf{0}$; for example $\tau.\,\mathbf{0} + \overline{x}y.\,\mathbf{0}$ will be abbreviated $\tau + \overline{x}y$. Also we sometimes write $\mathrm{fn}(P, Q, \ldots, x, y, \ldots)$ as an abbreviation for $\mathrm{fn}(P) \cup \mathrm{fn}(Q) \cup \ldots \cup \{x, y, \ldots\}$.

2.2 Transitions

A *transition* is of the form

$$P \xrightarrow{\alpha} Q$$

Intuitively, this transition means that P can evolve into Q, and in doing so perform the *action* α. In our calculus there will be five kinds of action α as follows. The *silent* action τ corresponds to an internal computation, and the *free output* action $\overline{x}y$ and *free input* action xy correspond to the transmission and reception of the free name y along x. The *bound input* action $x(y)$ means that any name can be received along x, and (y) designates the places where the received name will go. The *bound output* $\overline{x}(y)$ means that a local name designated by y is exported along x. A summary of the actions, their *free names* $\mathrm{fn}(\alpha)$ and *bound names* $\mathrm{bn}(\alpha)$ can be found in Table 1. We write $\mathrm{n}(\alpha)$ for $\mathrm{fn}(\alpha) \cup \mathrm{bn}(\alpha)$.

α	Kind	fn(α)	bn(α)
τ	Silent	$\emptyset$	$\emptyset$
$\overline{x}y$	Free Output	$\{x, y\}$	$\emptyset$
$\overline{x}(y)$	Bound Output	$\{x\}$	$\{y\}$
xy	Free Input	$\{x, y\}$	$\emptyset$
$x(y)$	Bound Input	$\{x\}$	$\{y\}$

Table 1: The actions.

The silent and free actions are familiar from CCS. In particular a free input action corresponds to an early instantiation of an input parameter, since it carries both the port name and received value. In contrast a bound input action carries only a port name, implying that the bound parameter will be instantiated at a later stage. The bound output actions are used to infer so-called scope extrusions; their parameters will never be instantiated to free names so the issue of "late vs. early" does not arise.

In order to define the transitions between agents we first introduce the notions of structural congruence and variant:

Definition 2 The *structural congruence* $\equiv$ on agents is the least congruence satisfying the following clauses:

1. If P and Q differ only in the choice of bound names, i.e. they are alpha-equivalent in the standard sense, then $P \equiv Q$,
2. $P|Q \equiv Q|P$,
3. $P + Q \equiv Q + P$,
4. $[x = x]P \equiv P$,
5. If $C(\tilde{x}) \stackrel{\mathrm{def}}{=} P$ then $C(\tilde{y}) \equiv P\{\tilde{y}/\tilde{x}\}$.

A *variant* of the transition $P \xrightarrow{\alpha} Q$ is a transition which only differs in that P and Q have been replaced by structurally congruent agents, and α has been alpha-converted, where a name bound in α includes Q in its scope. $\square$

As an example the following transitions are variants of each other:

$$
\begin{array}{lcl}
x(y).\,\overline{y}z & \xrightarrow{x(y)} & \overline{y}z \\
x(y).\,\overline{y}z & \xrightarrow{x(u)} & \overline{u}z \\
[x=x]x(y).\,\overline{y}z & \xrightarrow{x(y)} & \overline{y}z \\
[x=x]x(y).\,\overline{y}z & \xrightarrow{x(u)} & \overline{u}z
\end{array}
$$

The second transition differs from the first in that the name y has been alpha-converted to u in the action and in the agent after the arrow. The third transition differs from the first in that $x(y).\,\overline{y}z$ has been replaced by a structurally congruent agent, and the fourth transition combines these changes.

Below we will give two sets of rules for inferring transitions, one set corresponding to early and one corresponding to late instantiation. In each rule, the transition in the conclusion stands for all variants of the transition. We begin with the set of rules in [7] which can now be rendered as follows:

Definition 3 The set of rules LATE consists of the following:

$$
\begin{array}{llll}
\text{ACT}: & \dfrac{-}{\alpha.\,P \xrightarrow{\alpha} P} & \text{SUM}: & \dfrac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'} \\[2ex]
\text{PAR}: & \dfrac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q} \quad \mathrm{bn}(\alpha)\cap\mathrm{fn}(Q)=\emptyset & & \\[2ex]
\text{L-COM}: & \dfrac{P \xrightarrow{\overline{x}y} P' \quad Q \xrightarrow{x(z)} Q'}{P|Q \xrightarrow{\tau} P'|Q'\{y/z\}} & \text{CLOSE}: & \dfrac{P \xrightarrow{\overline{x}(y)} P' \quad Q \xrightarrow{x(y)} Q'}{P|Q \xrightarrow{\tau} (y)(P'|Q')} \\[2ex]
\text{RES}: & \dfrac{P \xrightarrow{\alpha} P'}{(y)P \xrightarrow{\alpha} (y)P'} \quad y \notin \mathrm{n}(\alpha) & \text{OPEN}: & \dfrac{P \xrightarrow{\overline{x}y} P'}{(y)P \xrightarrow{\overline{x}(y)} P'} \; y \neq x
\end{array}
$$

We write $P \xrightarrow{\alpha}_{\mathrm{L}} Q$ to mean that the transition $P \xrightarrow{\alpha} Q$ can be inferred from LATE. □

A reader familiar with the rules in [7] will note that LATE is more concise, yet it generates the same transitions. The use of variants and structural congruence makes it possible to formulate the rules without explicit alpha-conversions in rules generating bound actions, and special rules for identifiers and matching are unnecessary because of clauses 4 and 5 in Definition 2. For example, we can infer

$$[x=x]x(y).\,\overline{y}z \xrightarrow{x(u)} \overline{u}z$$

since this transition is a variant of $x(y).\,\overline{y}z \xrightarrow{x(y)} \overline{y}z$ which is an instance of ACT. This effect of "factoring" all issues related to structural congruence from the rules of action can also be obtained by a special structural rule

$$\frac{P' \equiv P \quad P \xrightarrow{\alpha} Q \quad Q \equiv Q'}{P' \xrightarrow{\alpha} Q'}$$

For example, the transition above can be inferred with this rule since $x(u).\,\overline{u}z \xrightarrow{x(u)} \overline{u}z$ is an instance of ACT and $[x=x]x(y).\,\overline{y}z \;\equiv\; x(u).\,\overline{u}z$.

In LATE the name bound by an input prefix form $x(y).\,P$ becomes instantiated in L-COM when a communication between two agents is inferred. Note that no rule in LATE generates a free input action. In contrast, with an *early instantiation* scheme the bound name y is instantiated when inferring an input transition from $x(y).\,P$:

Definition 4 The set of rules EARLY is obtained from LATE by replacing the rule L-COM with the following two rules:

E-INPUT: $\dfrac{-}{x(y).\,P \xrightarrow{xw} P\{w/y\}}$	E-COM: $\dfrac{P \xrightarrow{\overline{x}y} P' \quad Q \xrightarrow{xy} Q'}{P\vert Q \xrightarrow{\tau} P'\vert Q'}$

We write $P \xrightarrow{\alpha}_{\mathrm{E}} Q$ to mean that the transition $P \xrightarrow{\alpha} Q$ can be inferred from EARLY. □

The new rule E-INPUT admits an instantiation to any name w, so there will always be a suitable free input action available as a premise in E-COM. Note that the rule ACT remains in EARLY, so an input prefix may still generate bound input actions — these are needed with the rules OPEN and CLOSE to achieve scope extrusions such as

$$x(y).\,P \,|\, (y)\overline{x}y.\,Q \xrightarrow{\tau}_{\mathrm{E}} (y)(P|Q)$$

The following example highlights the different operations of LATE and EARLY. Assume that we want to infer a communication in the agent

$$x(y).\,P(y) \,|\, Q(y,u) \,|\, \overline{x}u.\,R$$

(We write "$P(y)$" to signify that P depends on y, and similarly for Q.) Using LATE we need a new name z in the PAR rule to avoid conflicts with the free names in $Q(y,u)$:

$$\dfrac{\dfrac{\dfrac{-}{x(y).\,P(y) \xrightarrow{x(z)}_{\mathrm{L}} P(z)}}{x(y).\,P(y)\,|\,Q(y,u) \xrightarrow{x(z)}_{\mathrm{L}} P(z)\,|\,Q(y,u)} \qquad \dfrac{-}{\overline{x}u\,.R \xrightarrow{\overline{x}u}_{\mathrm{L}} R}}{x(y).\,P(y)\,|\,Q(y,u)\,|\,\overline{x}u.\,R \;\xrightarrow{\tau}_{\mathrm{L}}\; P(u)\,|\,Q(y,u)\,|\,R}$$

Using EARLY the same communication can be inferred:

$$\dfrac{\dfrac{\dfrac{-}{x(y).\,P(y) \xrightarrow{xu}_{\mathrm{E}} P(u)}}{x(y).\,P(y)\,|\,Q(y,u) \xrightarrow{xu}_{\mathrm{E}} P(u)\,|\,Q(y,u)} \qquad \dfrac{-}{\overline{x}u\,.R \xrightarrow{\overline{x}u}_{\mathrm{E}} R}}{x(y).\,P(y)\,|\,Q(y,u)\,|\,\overline{x}u.\,R \;\xrightarrow{\tau}_{\mathrm{E}}\; P(u)\,|\,Q(y,u)\,|\,R}$$

The following lemma shows how $\xrightarrow{\alpha}_{\mathrm{E}}$ and $\xrightarrow{\alpha}_{\mathrm{L}}$ are related.

Lemma 1

1. $P \xrightarrow{\overline{x}y}_{\mathrm{E}} P'$ iff $P \xrightarrow{\overline{x}y}_{\mathrm{L}} P'$
2. $P \xrightarrow{\overline{x}(y)}_{\mathrm{E}} P'$ iff $P \xrightarrow{\overline{x}(y)}_{\mathrm{L}} P'$
3. $P \xrightarrow{x(y)}_{\mathrm{E}} P'$ iff $P \xrightarrow{x(y)}_{\mathrm{L}} P'$
4. $P \xrightarrow{xy}_{\mathrm{E}} P'$ iff $\exists P'', w : P \xrightarrow{x(w)}_{\mathrm{L}} P''$ with $P' \equiv P''\{y/w\}$
5. $P \xrightarrow{\tau}_{\mathrm{E}} P'$ iff $P \xrightarrow{\tau}_{\mathrm{L}} P'$

Proof: A standard induction over LATE and EARLY. The proof of 2 uses 1, and the proof of 5 uses all of 1–4. $\square$

In view of this lemma it will not be necessary to distinguish between $\longrightarrow_{\mathrm{E}}$ and $\longrightarrow_{\mathrm{L}}$, and we will simply write $\longrightarrow$ for $\longrightarrow_{\mathrm{E}}$ from now on.

2.3 Late and Early Bisimulations

We first recall the definition of bisimulation in [7]:

Definition 5 A binary relation $\mathcal{S}$ on agents is a *late simulation* if $P\mathcal{S}Q$ implies that

1. If $P \xrightarrow{\alpha} P'$ and α is τ, $\overline{x}z$ or $\overline{x}(y)$ with $y \notin \mathrm{fn}(P, Q)$, then for some Q', $\quad Q \xrightarrow{\alpha} Q' \quad$ and $\quad P'\mathcal{S}Q'$
2. If $P \xrightarrow{x(y)} P' \quad$ and $\quad y \notin \mathrm{fn}(P, Q)$, then for some Q', $\quad Q \xrightarrow{x(y)} Q' \quad$ and for all w, $P'\{w/y\}\mathcal{S}Q'\{w/y\}$

The relation $\mathcal{S}$ is a *late bisimulation* if both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are late simulations. We define *late bisimilarity* $P \stackrel{\cdot}{\sim}_{\mathrm{L}} Q$ to mean that $P\mathcal{S}Q$ for some late bisimulation $\mathcal{S}$. □

Note that late simulations do not require anything of free input actions. Instead, there is a strong requirement on bound input actions: the resulting agents P' and Q' must continue to simulate for all instances w of the bound name. The term "late" refers to the fact that these w are introduced after the simulating derivative Q' has been chosen. The algebraic theory of $\stackrel{\cdot}{\sim}_{\mathrm{L}}$ is explored in [7].

The natural bisimulation equivalence for early instantiation will use free input actions rather than the extra requirement (clause 2) on bound input actions:

Definition 6 A binary relation $\mathcal{S}$ on agents is an *early simulation* if $P\mathcal{S}Q$ implies that

If $P \xrightarrow{\alpha} P'$ and α is any action with $\mathrm{bn}(\alpha) \cap \mathrm{fn}(P, Q) = \emptyset$, then for some Q', $\quad Q \xrightarrow{\alpha} Q' \quad$ and $\quad P'\mathcal{S}Q'$

The relation $\mathcal{S}$ is an *early bisimulation* if both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are early simulations. We define *early bisimilarity* $P \stackrel{\cdot}{\sim}_{\mathrm{E}} Q$ to mean that $P\mathcal{S}Q$ for some early bisimulation $\mathcal{S}$. □

So, in an early simulation different instances of an input transition (i.e. different free inputs) may be simulated by different Q'. Late and early bisimilarity represent the two different views of equivalence presented in the introduction. To see that these two equivalences are different consider the following example:

$$\begin{array}{rcl} P & = & x(u).\tau + x(u) \\ Q & = & P + x(u).[u = z]\tau \end{array}$$

Then $P \stackrel{\cdot}{\sim}_{\mathrm{E}} Q$, but $P \stackrel{\cdot}{\not\sim}_{\mathrm{L}} Q$. The reason is the transition

$$Q \xrightarrow{x(u)} [u = z]\tau \tag{6}$$

P has no transition which simulates (6) for all instantiations of u. However, for all free input actions there is a simulating transition: for z it is

$$P \xrightarrow{xz} \tau$$

(since $([u\!=\!z]\tau)\{z/u\} \equiv \tau$) and for all other names it is

$$P \xrightarrow{xu} \mathbf{0}$$

(since $([u\!=\!z]\tau) \dot{\sim}_{\mathrm{E}} \mathbf{0}$ for all $u \neq z$).

We will now support our claim from [7] that $\dot{\sim}_{\mathrm{E}}$ can be obtained by commuting the quantifiers in clause 2 of Definition 5:

Definition 7 A binary relation $\mathcal{S}$ on agents is an *alternative simulation* if $P\mathcal{S}Q$ implies that

1. If $P \xrightarrow{\alpha} P'$ and α is τ, $\overline{x}z$ or $\overline{x}(y)$ with $y \notin \mathrm{fn}(P,Q)$, then for some Q', $\quad Q \xrightarrow{\alpha} Q' \quad$ and $\quad P'\mathcal{S}Q'$
2. If $P \xrightarrow{x(y)} P'$ and $y \notin \mathrm{fn}(P,Q)$, then for all w, there is Q' such that $\quad Q \xrightarrow{x(y)} Q' \quad$ and $\quad P'\{w/y\}\mathcal{S}Q'\{w/y\}$

The relation $\mathcal{S}$ is an *alternative bisimulation* if both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are alternative simulations. We define $P \dot{\sim}' Q$ to mean that $P\mathcal{S}Q$ for some alternative bisimulation $\mathcal{S}$. $\square$

It is obvious that every late simulation is also an alternative simulation, so $\dot{\sim}_{\mathrm{L}} \subseteq \dot{\sim}'$. Furthermore we have:

Lemma 2 $\quad \dot{\sim}' \;=\; \dot{\sim}_{\mathrm{E}}$

Proof: From Lemma 1.4 it follows that the following two requirements on any relation $\mathcal{S}$ are equivalent:

$$\forall P,Q,x,y,P' : \text{If } P \xrightarrow{xy} P' \text{ then } \exists Q' :\ Q \xrightarrow{xy} Q' \text{ and } P'\mathcal{S}Q'$$

$$\forall P,Q,x,w,P'' : \text{If } P \xrightarrow{x(w)} P'' \text{ then } \forall y \exists Q'' : Q \xrightarrow{x(w)} Q'' \text{ and } P''\{y/w\}\mathcal{S}Q''\{y/w\}$$

Hence, $\mathcal{S}$ is an alternative simulation iff it is an early simulation. $\square$

Thus $\dot{\sim}_{\mathrm{E}}$ is strictly weaker than $\dot{\sim}_{\mathrm{L}}$. We will not explore the theory of $\dot{\sim}_{\mathrm{E}}$ here. Just like $\dot{\sim}_{\mathrm{L}}$ it is an equivalence relation and is preserved by all operators except input prefix, and if $P\{w/y\} \dot{\sim}_{\mathrm{E}} Q\{w/y\}$ for all w then $x(y).P \dot{\sim}_{\mathrm{E}} x(y).Q$.

3 Modal Logics

In this section we establish characterizations of late and early bisimilarity in terms of properties expressible in various modal logics. In addition we compare in detail the distinguishing power of a number of logics. We begin by introducing a logic encompassing all those we consider and establishing some properties of its satisfaction relation.

3.1 Connectives

Definition 8 The logic $\mathcal{A}$ is a subset, specified below, of the set of formulae given by:

$$\begin{array}{lcll} A & ::= & \bigwedge_{i\in I} A_i & (I \text{ a denumerable set}) \\ & \mid & \neg A & \\ & \mid & [x=y]A & \\ & \mid & \langle\alpha\rangle A & (\alpha = \tau, \overline{x}y, xy, \overline{x}(y), x(y)) \\ & \mid & \langle x(y)\rangle^{\mathrm{L}} A & \\ & \mid & \langle x(y)\rangle^{\mathrm{E}} A & \end{array}$$

In each of $\langle\overline{x}(y)\rangle A$, $\langle x(y)\rangle A$, $\langle x(y)\rangle^{\mathrm{L}} A$ and $\langle x(y)\rangle^{\mathrm{E}} A$, the occurrence of y in parentheses is a binding occurrence whose scope is A. The set of names occurring free in A is written $\mathrm{fn}(A)$. The logic $\mathcal{A}$ consists of those formulae A with $\mathrm{fn}(A)$ finite. $\Box$

In Definition 9 below we shall introduce a satisfaction relation $\models$ between agents and formulae of $\mathcal{A}$. Although the definition will be a little more complex, the relation will have the following simple characterization:

Proposition 1 For all agents P,

$$\begin{array}{lll} P \models \bigwedge_{i\in I} A_i & \text{iff} & \text{for all } i \in I,\ P \models A_i \\ P \models \neg A & \text{iff} & \text{not } P \models A \\ P \models [x=y]A & \text{iff} & \text{if } x = y \text{ then } P \models A \\ P \models \langle\alpha\rangle A & \text{iff} & \text{for some } P',\ P \xrightarrow{\alpha} P' \text{ and } P' \models A, \text{ for } \alpha = \tau, \overline{x}y, xy \end{array}$$

and assuming that the name y is not free in P

$$\begin{array}{lll} P \models \langle\overline{x}(y)\rangle A & \text{iff} & \text{for some } P',\ P \xrightarrow{\overline{x}(y)} P' \text{ and } P' \models A \\ P \models \langle x(y)\rangle A & \text{iff} & \text{for some } P',\ P \xrightarrow{x(y)} P' \text{ and for some } z,\ P'\{z/y\} \models A\{z/y\} \\ P \models \langle x(y)\rangle^{\mathrm{L}} A & \text{iff} & \text{for some } P',\ P \xrightarrow{x(y)} P' \text{ and for all } z,\ P'\{z/y\} \models A\{z/y\} \\ P \models \langle x(y)\rangle^{\mathrm{E}} A & \text{iff} & \text{for all } z \text{ there is } P' \text{ such that } P \xrightarrow{x(y)} P' \text{ and } P'\{z/y\} \models A\{z/y\} \end{array}$$

$\square$

The assumption on y is no constraint since Lemma 3(a) below asserts that alpha-convertible formulae are logically equivalent.

Before embarking on the formal definitions we will explain the intuition behind the connectives. Conjunction, negation, and the silent, output and free input modalities work as in the logic $\mathcal{PL}$ described in the introduction. We will write **true** for the empty conjunction and **false** for $\neg$**true**. Note that an atomic equality predicate on names can be defined in terms of the *matching* connective $[x = y]$; the formula

$$\neg[x = y]\mathbf{false}$$

holds of P precisely when $x = y$, regardless of P. Conversely, if an atomic equality predicate $(x = y)$ on names were taken as primitive, $[x = y]A$ could be derived as $\neg((x = y) \wedge \neg A)$.

There are three kinds of bound input modality. They all require an agent to have a bound input transition of type $P \xrightarrow{x(y)} P'$ but they differ in the requirements on P'. The *basic* bound input modality $\langle x(y)\rangle A$ merely requires that P' satisfies A for *some* instantiation of the parameter y. The *late* modality $\langle x(y)\rangle^{\mathrm{L}}$ is stronger; it requires P' to satisfy A for *all* such instantiations. Finally the *early* modality $\langle x(y)\rangle^{\mathrm{E}}$ is weaker than the late modality; it allows *different* derivatives P' to satisfy A for the different instantiations of y. As an example let

$$\begin{array}{rcl} A & = & \langle x(y)\rangle\neg\langle\tau\rangle\mathbf{true} \\ A_L & = & \langle x(y)\rangle^{\mathrm{L}}\neg\langle\tau\rangle\mathbf{true} \\ A_E & = & \langle x(y)\rangle^{\mathrm{E}}\neg\langle\tau\rangle\mathbf{true} \end{array}$$

First put

$$P_1 = x(y).\,[y = u]\tau$$

It then holds that

$$P_1 \models A$$

The derivative P' is here $[y = u]\tau$ and there are instantiations of y, namely all but u, where P' has no τ-transition and thus satisfies $\neg\langle\tau\rangle$**true**. But for $y = u$ there is such a transition, hence P_1 satisfies neither A_E nor A_L. Next assume $u \neq v$ and consider

$$P_2 = x(y).\,[y = u]\tau \;+\; x(y).\,[y = v]\tau$$

Here there are two possible derivatives under the bound input action $x(y)$. The derivative corresponding to the left branch lacks a τ transition for

$y \neq u$, while the right branch lacks a τ transition for $y \neq v$. It follows that for any instantiation of y we can choose a derivative lacking a τ; thus

$$P_2 \models A_E$$

Of course P_2 also satisfies A, but it does not satisfy A_L since no single derivative lacks a τ for all instantiations of y. Finally consider

$$P_3 = x(y)$$

Then P_3 satisfies all of A, A_E and A_L.

The dual operators $[\alpha], [x(y)]^{\mathrm{L}}$ and $[x(y)]^{\mathrm{E}}$ of $\langle\alpha\rangle, \langle x(y)\rangle^{\mathrm{L}}$ and $\langle x(y)\rangle^{\mathrm{E}}$ are defined in the standard way: $[\alpha]A = \neg\langle\alpha\rangle\neg A$ etc. We note in particular the following properties:

$$\begin{array}{lll}
P \models [x(y)]A & \text{iff} & \text{for all } P'\text{, if } P \xrightarrow{x(y)} P' \text{ then for all } z,\ P'\{z/y\} \models A\{z/y\} \\
P \models [x(y)]^{\mathrm{L}}A & \text{iff} & \text{for all } P'\text{, if } P \xrightarrow{x(y)} P' \text{ then for some } z,\ P'\{z/y\} \models A\{z/y\} \\
P \models [x(y)]^{\mathrm{E}}A & \text{iff} & \text{there is } z \text{ such that for all } P'\text{, if } P \xrightarrow{x(y)} P' \text{ then } P'\{z/y\} \models A\{z/y\}
\end{array}$$

So $[\cdot]$ signifies universal quantification over derivatives, whereas $\langle\cdot\rangle$ implies existential quantification. Note that with the three bound input modalities and their duals all combinations of existential/universal quantifications of derivatives and parameter instantiation are covered.

We now return to the formal definition of the satisfaction relation:

Definition 9 The *satisfaction relation* between agents and formulae of $\mathcal{A}$ is given by:

$$\begin{array}{lll}
P \models \bigwedge_{i\in I} A_i & \text{if} & \text{for all } i \in I,\ P \models A_i \\
P \models \neg A & \text{if} & \text{not } P \models A \\
P \models [x=y]A & \text{if} & \text{if } x = y \text{ then } P \models A \\
P \models \langle\alpha\rangle A & \text{if} & \text{for some } P',\ P \xrightarrow{\alpha} P' \text{ and } P' \models A, \\
 & & \text{for } \alpha = \tau, \overline{x}y, xy \\
P \models \langle\overline{x}(y)\rangle A & \text{if} & \text{for some } P' \text{ and } w \notin \mathrm{fn}(A) - \{y\}, \\
 & & P \xrightarrow{\overline{x}(w)} P' \text{ and } P' \models A\{w/y\} \\
P \models \langle x(y)\rangle A & \text{if} & \text{for some } P' \text{ and } w,\ P \xrightarrow{x(w)} P' \\
 & & \text{and for some } z,\ P'\{z/w\} \models A\{z/y\} \\
P \models \langle x(y)\rangle^{\mathrm{L}} A & \text{if} & \text{for some } P' \text{ and } w,\ P \xrightarrow{x(w)} P' \\
 & & \text{and for all } z,\ P'\{z/w\} \models A\{z/y\} \\
P \models \langle x(y)\rangle^{\mathrm{E}} A & \text{if} & \text{for all } z \text{ there are } P' \text{ and } w \text{ such that} \\
 & & P \xrightarrow{x(w)} P' \text{ and } P'\{z/w\} \models A\{z/y\}
\end{array}$$

$\square$

Recall that by Lemma 1 we may combine the late and early schemes in giving and working with this definition. Before commenting on it in detail we note the following facts. We write $\equiv$ for alpha-equivalence of formulae.

Lemma 3 (a) If $P \models A$ and $A \equiv B$ then $P \models B$.
(b) If $P \models A$ and $u \notin \text{fn}(P, A)$ then $P\{u/v\} \models A\{u/v\}$.

Proof: The two assertions are proved together by showing by induction on A that if $P \models A$, $A \equiv B$ and $u \notin \text{fn}(P, A)$ then $P\{u/v\} \models B\{u/v\}$. The proof, though not unduly difficult, contains some points of technical interest and requires careful attention to detail. It is given in the appendix. $\square$

The final four clauses in the definition of satisfaction are complicated by the inclusion of the name w. This is required to define $P \models A$ in the case that a name occurs bound in A and free in P. For suppose the clause for the bound output modality were simplified to that given in Proposition 1 above. If $P \equiv (w)\overline{x}w.\, y(z)$ and $A \equiv \langle \overline{x}(y) \rangle \textbf{true}$ then according to Definition 9, $P \models A$; but under the simplified definition, $P \not\models A$. A similar difficulty arises with the other three clauses.

However by Lemma 3(a), when considering an assertion $P \models A$, given any name x bound in A, we may always assume that x is not free in P. This assumption, which we make from now on, leads to a simple proof of the more elegant characterization given above in Proposition 1. This characterization helps to make clear the significant points in the definition. Note in particular that the clause for $\langle \overline{x}(y) \rangle$ may be subsumed under that for $\langle \alpha \rangle$ for $\alpha = \tau, \overline{x}y, xy$. The need for the condition on w in the clause for $\langle \overline{x}(y) \rangle$ can be seen by considering $P \equiv (y)\overline{x}y$ and $A \equiv \langle \overline{x}(y) \rangle \neg [y = w] \textbf{false}$. Under Definition 9, $P \not\models A$. If the condition on w were removed we would have $P \models A$ but the bound output clause of Proposition 1 would no longer hold.

The following useful lemma describes some relationships among the modalities.

Lemma 4 (a) Suppose $w \notin \text{fn}(A, y)$. Then

$$\begin{aligned} P \models \langle xy \rangle A \ \text{ iff } \ & P \models \langle x(w) \rangle^{\mathrm{L}} [w = y] A \\ \text{ iff } \ & P \models \langle x(w) \rangle^{\mathrm{E}} [w = y] A \\ \text{ iff } \ & P \models \langle x(w) \rangle \neg [w = y] \neg A \end{aligned}$$

(b) $P \models \langle x(y)\rangle^{\mathrm{E}} A$ iff for all z, $P \models \langle xz\rangle A\{z/y\}$
(c) $P \models \langle x(y)\rangle A$ iff for some z, $P \models \langle xz\rangle A\{z/y\}$

Proof: Straightforward from the definitions. See the appendix. $\square$

3.2 Characterizations of Equivalences

Suppose $\mathcal{K}$ is a sublogic of $\mathcal{A}$. Then $\mathcal{K}(P) = \{A \in \mathcal{K} \mid P \models A\}$. We write $=_{\mathcal{K}}$ for the equivalence relation determined by $\mathcal{K}$: $P =_{\mathcal{K}} Q$ iff $\mathcal{K}(P) = \mathcal{K}(Q)$. We say $\mathcal{K}$ *characterizes* a relation $\mathcal{R}$ if $=_{\mathcal{K}} = \mathcal{R}$.

A number of sublogics of $\mathcal{A}$ will be considered. They share a common basis $\mathcal{A}_0$ consisting of the formulae of $\mathcal{A}$ built from conjunction, negation and the modalities $\langle\tau\rangle, \langle\overline{x}y\rangle$ and $\langle\overline{x}(y)\rangle$. The sublogics of $\mathcal{A}$ extending $\mathcal{A}_0$ are named by indicating which of $\langle x(y)\rangle, \langle x(y)\rangle^{\mathrm{E}}, \langle xy\rangle, \langle x(y)\rangle^{\mathrm{L}}$ and $[x=y]$ are added to $\mathcal{A}_0$, using the letters $\mathcal{B}, \mathcal{E}, \mathcal{F}, \mathcal{L}$ and $\mathcal{M}$ respectively. For instance, $\mathcal{LM}$ is the extension of $\mathcal{A}_0$ obtained by adding the late bound input modality $\langle x(y)\rangle^{\mathrm{L}}$ and matching $[x = y]$, while $\mathcal{F}$ is obtained by adding the free input modality $\langle xy\rangle$ alone.

We now give the main characterizations of $\dot\sim_{\mathrm{L}}$ and $\dot\sim_{\mathrm{E}}$.

Theorem 1 $\mathcal{LM}$ characterizes $\dot\sim_{\mathrm{L}}$.

Proof: The proof follows a standard pattern but contains some novelty. First we show that $\dot\sim_{\mathrm{L}} \subseteq =_{\mathcal{LM}}$ by proving by induction on A in $\mathcal{LM}$ that if $P \dot\sim_{\mathrm{L}} Q$ then $P \models A$ iff $Q \models A$. The argument for the converse amounts to a proof that if $P \dot\nsim_{\mathrm{L}} Q$ then there is $A \in \mathcal{LM}(P) - \mathcal{LM}(Q)$ with $\mathrm{fn}(A) \subseteq \mathrm{fn}(P, Q)$. The principal point of interest is the use of a combination of the late bound input modality $\langle x(y)\rangle^{\mathrm{L}}$ and matching. The proof is given in the appendix. $\square$

We need infinite conjunction only if the transition system is not image-finite (up to $\equiv$). In particular, if all recursive definitions are guarded then finite conjunction suffices. Recalling the quantifier switch in the semantic clauses for $\langle x(y)\rangle^{\mathrm{L}}$ and $\langle x(y)\rangle^{\mathrm{E}}$, in view of the preceding theorem it may be expected that $\mathcal{EM}$ characterizes $\dot\sim_{\mathrm{E}}$. In fact we have:

Theorem 2 Each of $\mathcal{EM}$, $\mathcal{F}$ and $\mathcal{BM}$ characterizes $\dot\sim_{\mathrm{E}}$.

Proof: By utilizing the characterization of $\dot{\sim}_{\mathrm{E}}$ in the early scheme, Lemma 2, a proof that $\mathcal{F}$ characterizes $\dot{\sim}_{\mathrm{E}}$ is easily obtained. That $\mathcal{EM}$ and $\mathcal{BM}$ also characterize $\dot{\sim}_{\mathrm{E}}$ then follows using Lemma 4. For details see the appendix. □

We have seen that $\mathcal{F}$ characterizes $\dot{\sim}_{\mathrm{E}}$ and that the free input modality corresponds to combinations of the bound input modalities and matching. A natural question concerns the power of the bound input modalities in the absence of matching. We give a sequence of examples which establish the relationships among the various logics. These are summarized in a picture below.

Lemma 5 $P =_{\mathcal{EL}} Q$ but $P \neq_{\mathcal{B}} Q$ where

$$P = x(y)$$
$$Q = x(y) + x(y).\,[y = z]\tau$$

Proof: Note that if $A \equiv [x(y)]\neg\langle\tau\rangle\mathbf{true}$ then $P \models A$ but $Q \not\models A$. To see that $P =_{\mathcal{EL}} Q$ we prove by induction on A in $\mathcal{EL}$ that $P \models A$ iff $Q \models A$. See the appendix. □

Lemma 6 $P \dot{\sim}_{\mathrm{E}} Q$ but $P \neq_{\mathcal{L}} Q$ where

$$P = x(y) + x(y).\,([y = z]\tau + [y = w]\tau)$$
$$Q = x(y).\,[y = z]\tau + x(y).\,[y = w]\tau$$

Proof: Clearly $P \dot{\sim}_{\mathrm{E}} Q$. To see that $P \neq_{\mathcal{L}} Q$ simply note that if $A \equiv \langle x(y)\rangle^{\mathrm{L}}\neg\langle\tau\rangle\mathbf{true}$ then $P \models A$ but $Q \not\models A$. □

Lemma 7 $P =_{\mathcal{BL}} Q$ but $P \neq_{\mathcal{E}} Q$ where

$$P = x(y).\,[y = z]\tau + x(y).\,([y = z]\tau + [y = w]\tau)$$
$$Q = x(y).\,[y = z]\tau + x(y).\,[y = w]\tau$$

Proof: To see that $P \neq_{\mathcal{E}} Q$ note that if $A \equiv \langle x(y)\rangle^{\mathrm{E}}\neg\langle\tau\rangle\mathbf{true}$ then $Q \models A$ but $P \not\models A$. To see that $P =_{\mathcal{BL}} Q$ we prove by induction on A in $\mathcal{BL}$ that $P \models A$ iff $Q \models A$. See the appendix. □

Lemma 8 $P =_{\mathcal{BEL}} Q$ but $P \dot{\not\sim}_{\mathrm{E}} Q$ where

$$P = x(y).\,[y = z]\tau$$
$$Q = x(y).\,[y = w]\tau$$

Proof: Clearly $P \dot{\not\sim}_{\mathrm{E}} Q$. To see that $P =_{\mathcal{BEL}} Q$ we prove by induction on A in $\mathcal{BEL}$ that $P \models A$ iff $Q \models A$. The proof is similar to that of Lemma 7. We omit the details. □

Lemma 9 $P =_{\mathcal{FL}} Q$ but $P \dot{\not\sim}_{\mathrm{L}} Q$ where

$$P = x(y) + x(y).\,\tau$$
$$Q = x(y) + x(y).\,\tau + x(y).\,[y = z]\tau$$

Proof: Clearly $P \dot{\not\sim}_{\mathrm{L}} Q$. To see that $P =_{\mathcal{FL}} Q$ we prove by induction on A in $\mathcal{FL}$ that $P \models A$ iff $Q \models A$. The proof is similar to that of Lemma 7. We omit the details. □

To complete the picture we note the following. Let us say that two logics $\mathcal{J}$ and $\mathcal{K}$ are *equipotent* if $=_{\mathcal{J}} \; = \; =_{\mathcal{K}}$.

Lemma 10 Let Z be any combination of $\mathcal{B}, \mathcal{E}, \mathcal{F}, \mathcal{L}, \mathcal{M}$. Then in an obvious notation

(a) $\mathcal{F} + Z$, $\mathcal{BF} + Z$ and $\mathcal{EF} + Z$ are equipotent.
(b) $\mathcal{BM} + Z$, $\mathcal{EM} + Z$ and $\mathcal{FM} + Z$ are equipotent.
(c) $\mathcal{LM} + Z$ and $\mathcal{FLM} + Z$ are equipotent.
(d) Finally, $\mathcal{M}$ and $\mathcal{A}_0$ are equipotent.

Proof: See the appendix. □

We summarize the relationships among the logics established by the preceding results in the following theorem.

Theorem 3 In the picture below, each point represents a distinct relation. A line between two relations signifies inclusion, while the absence of a line signifies that they are incomparable. By 'etc.' we mean any other combination equipotent by Lemma 10.

$$
\begin{tikzcd}[arrows=dash]
 & \dot{\sim}_{\mathrm{L}}\,,\ \mathcal{LM},\ \text{etc.} \arrow[d] & \\
 & \mathcal{FL},\ \mathcal{EFL},\ \text{etc.} \arrow[d] \arrow[dr] & \\
 & \mathcal{BEL} \arrow[dl] \arrow[d] \arrow[dr] & \dot{\sim}_{\mathrm{E}}\,,\ \mathcal{EM},\ \mathcal{F},\ \mathcal{BM},\ \text{etc.} \arrow[d] \\
\mathcal{EL} \arrow[d] \arrow[dr] & \mathcal{BL} \arrow[dl] \arrow[dr] & \mathcal{BE} \arrow[dl] \arrow[d] \\
\mathcal{L} \arrow[dr] & \mathcal{E} \arrow[d] & \mathcal{B} \arrow[dl] \\
 & \mathcal{A}_0 &
\end{tikzcd}
$$

□

The examples in Lemmas 5–9 all involve the match expression of the calculus. However, its use is in each case inessential. For example, Lemma 5 asserts that $P =_{\mathcal{EL}} Q$ but $P \neq_{\mathcal{B}} Q$ where $P = x(y)$ and $Q = x(y) + x(y).\,[y = z]\tau$. Alternatively we can take:

$$
\begin{aligned}
P &= x(y).\,(\overline{y}.\,z + z.\,\overline{y}) \\
Q &= x(y).\,(\overline{y}.\,z + z.\,\overline{y}) + x(y).\,(\overline{y} \mid z)
\end{aligned}
$$

Similar modifications can be made to the other examples.

4 Future work

The logic we have introduced no doubt has interesting intrinsic properties, which we have not begun to study. Here, we only wish to mention two questions about its relationship with the π-calculus which appear to be of immediate interest.

First, what happens when we introduce the *mismatch* form CHANGE

$$[x \neq y]P$$

into the calculus? Note that the corresponding mismatch connective

$$[x \neq y]A$$

does not add power to our logic since it already has matching and negation.

Second, considering the input modalities, can we factor out their quantificational content? It is attractive to factor $\langle x(y)\rangle^{\mathrm{L}}$ thus:

$$\langle x(y)\rangle^{\mathrm{L}}A \stackrel{\mathrm{def}}{=} \langle x\rangle(\forall y A)$$

Now, to express the satisfaction relation, we appear to need also to factor the input prefix $x(y)$ of the calculus thus:

$$x(y).P \stackrel{\mathrm{def}}{=} x.\lambda y P$$

– in other words, we need to give proper status to (λ-)*abstractions*, which abstract names from processes. This step has considerable interest, since there are other independent advantages to be gained from it.

References

[1] Hennessy, M., **Algebraic Theory of Processes**, MIT Press, 1988.

[2] Hennessy, M. and Milner, R., *Algebraic Laws for Non-determinism and Concurrency*, Journal of ACM, Vol 32, pp137–161, 1985.

[3] Hoare, C.A.R., **Communicating Sequential Processes**, Prentice Hall, 1985.

[4] Milner, R., **A Calculus of Communicating Systems**, Lecture Notes in Computer Science, Volume 92, Springer-Verlag, 1980.

[5] Milner, R., **Communication and Concurrency**, Prentice Hall, 1989.

[6] Milner, R., Parrow, J. and Walker, D., *A Calculus of Mobile Processes, Part I*, Reports ECS-LFCS–89–85, Laboratory for Foundations of Computer Science, Computer Science Department, Edinburgh University, 1989. Also to appear in J. Information and Computation.

[7] Milner, R., Parrow, J. and Walker, D., *A Calculus of Mobile Processes, Part II*, Reports ECS-LFCS–89–86, Laboratory for Foundations of Computer Science, Computer Science Department, Edinburgh University, 1989. Also to appear in J. Information and Computation.

[8] Park, D.M.R., *Concurrency and Automata on Infinite Sequences*, Lecture Notes in Computer Science, Vol 104, Springer Verlag, 1980.

Appendix

This section contains the proofs omitted from the main text. Some results from section 3 of [7] concerning the transition system are used.

Proof of Lemma 3: We prove the two assertions by showing by induction on A that:

$$\text{if } P \models A,\ A \equiv B \text{ and } u \notin \mathrm{fn}(P, A), \text{ then } P\{u/v\} \models B\{u/v\}$$

Let $\sigma = \{u/v\}$.

The conjunction case is trivial.

Suppose $A \equiv \neg A'$ so $B \equiv \neg B'$ with $A' \equiv B'$. Since $P \not\models A'$, by induction hypothesis $P \not\models B'$ and so $P \models B$. Hence if $u = v$ the claim holds. Suppose $u \neq v$ so $v \notin \mathrm{fn}(P\sigma, B\sigma)$. If $P\sigma \not\models B\sigma$ then $P\sigma \models B'\sigma$ so by induction hypothesis $P\sigma\sigma^{-1} \models B'\sigma\sigma^{-1}$, so $P \models B'$. Then again by induction hypothesis $P \models A'$. Contradiction. Hence $P\sigma \models B\sigma$.

Suppose $A \equiv [x=y]A'$ so $B \equiv [x=y]B'$ with $A' \equiv B'$. If $x \neq y$ then certainly $P\sigma \models B\sigma$ since $B\sigma \equiv [x\sigma = y\sigma]B'\sigma$ and $x\sigma \neq y\sigma$. If $x = y$ then $P \models A'$ and by induction hypothesis $P\sigma \models B'\sigma$ so again $P\sigma \models B\sigma$.

Suppose $A \equiv \langle\alpha\rangle A'$ where $\alpha = \tau, \overline{x}y, xy$, so $B \equiv \langle\alpha\rangle B'$ with $A' \equiv B'$. Since $P \models A$ there is P' such that $P \xrightarrow{\alpha} P'$ and $P' \models A'$. Then $P\sigma \xrightarrow{\alpha\sigma} P'\sigma$ and by induction hypothesis $P'\sigma \models B'\sigma$. Hence $P\sigma \models B\sigma$ since $B\sigma \equiv \langle\alpha\sigma\rangle B'\sigma$.

Suppose $A \equiv \langle\overline{x}(y)\rangle A'$ so $B\sigma \equiv \langle\overline{x\sigma}(y')\rangle B'\sigma$ where $A'\{y'/y\} \equiv B'$ and y' is fresh. Since $P \models A$ there are P' and $w \notin \mathrm{fn}(A)-\{y\}$ such that $P \xrightarrow{\overline{x}(w)} P'$ and $P' \models A'\{w/y\}$. Choose $w' \notin \mathrm{fn}(P, A, u)$. Then $P \xrightarrow{\overline{x}(w')} P'' \equiv P'\{w'/w\}$ and by induction hypothesis $P'' \models B'\{w'/y'\}$. Also $P\sigma \xrightarrow{\overline{x\sigma}(w')} P''\sigma$ and again by induction hypothesis $P''\sigma \models B'\{w'/y'\}\sigma$. Hence $P\sigma \models B\sigma$ since $B'\{w'/y'\}\sigma \equiv B'\sigma\{w'/y'\}$.

Suppose $A \equiv \langle x(y)\rangle^{\mathrm{L}} A'$ so $B\sigma \equiv \langle x\sigma(y')\rangle^{\mathrm{L}} B'\sigma$ where $A'\{y'/y\} \equiv B'$ and y' is fresh. Since $P \models A$ there are P' and w such that $P \xrightarrow{x(w)} P'$ and for all z, $P'\{z/w\} \models A'\{z/y\}$. Choose $w' \notin \mathrm{fn}(P, A)$. Then $P \xrightarrow{x(w')} P'' \equiv P'\{w'/w\}$ and by induction hypothesis for all z,

$$P''\{z/w'\} \models B'\{z/y'\} \qquad (*)$$

Now $P\sigma \xrightarrow{x\sigma(w')} P''\sigma$.

Claim For all z, $P''\sigma\{z/w'\} \models B'\sigma\{z/y'\}$.

Proof of Claim: If $u = v$ the claim is immediate from $(*)$, so suppose $u \neq v$.

Case 1: $z \neq u, v$. Then $P''\sigma\{z/w'\} \equiv P''\{z/w'\}\sigma$ and $B'\sigma\{z/y'\} \equiv B'\{z/y'\}\sigma$. By induction hypothesis and $(*)$, $P''\{z/w'\}\sigma \models B'\{z/y'\}\sigma$ since $u \notin \mathrm{fn}(P''\{z/w'\}, B'\{z/y'\})$. Hence again by induction hypothesis, $P''\{z/w'\}\sigma \models B'\sigma\{z/y'\}$.

Case 2: $z = u$. Now $P''\sigma\{u/w'\} \equiv P''\{v/w'\}\sigma$ and $B'\sigma\{u/y'\} \equiv B'\{v/y'\}\sigma$. By $(*)$, $P''\{v/w'\} \models B'\{v/y'\}$ so by induction hypothesis, $P''\{v/w'\}\sigma \models B'\{v/y'\}\sigma$ since $u \notin \mathrm{fn}(P''\{v/w'\}, B'\{v/y'\})$. Hence by induction hypothesis, $P''\{v/w'\}\sigma \models B'\sigma\{u/y'\}$.

Case 3: $z = v$. Then $P''\sigma \models B'\sigma$ by induction hypothesis since $u \notin \mathrm{fn}(P'', B')$. So again by induction hypothesis, $P''\sigma\{v/w'\} \models B'\sigma\{v/y'\}$ since $v \notin \mathrm{fn}(P''\sigma, B'\sigma)$.

This completes the proof of the Claim and hence of the case $\langle x(y)\rangle^{\mathrm{L}}$.

The cases $A \equiv \langle x(y)\rangle^{\mathrm{E}} A'$ and $A \equiv \langle x(y)\rangle A'$ involve similar arguments. □

Proof of Lemma 4: First note that if $w \neq y$ then

$$\begin{array}{ll} & P \models \langle x(w)\rangle^{\mathrm{L}}[w=y]A \\ \text{iff} & P \models \langle x(w)\rangle^{\mathrm{E}}[w=y]A \\ \text{iff} & P \models \langle x(w)\rangle\neg[w=y]\neg A \\ \text{iff} & \text{for some } P',\ P \xrightarrow{x(w)} P' \text{ and } P'\{y/w\} \models A\{y/w\} \end{array}$$

Now suppose $w \notin \mathrm{fn}(A, y)$. If $P \models \langle xy \rangle A$ then for some P', $P \xrightarrow{xy} P'$ and $P' \models A$. Then $P \xrightarrow{x(w)} P''$ with $P''\{y/w\} \equiv P'$. Since $P''\{y/w\} \models A\{y/w\} \equiv A$ it follows by the above that $P \models \langle x(w) \rangle^{\mathrm{L}}[w=y]A$ etc. Conversely, if $P \xrightarrow{x(w)} P''$ and $P''\{y/w\} \models A\{y/w\}$ then $P \xrightarrow{xy} P' \equiv P''\{y/w\}$ and $P' \models A$, so $P \models \langle xy \rangle A$. $\square$

Proof of Theorem 1: We first show by induction on structure that for all A in $\mathcal{LM}$, if $P \dot{\sim}_{\mathrm{L}} Q$ then $P \models A$ iff $Q \models A$. Suppose $P \models A$. The conjunction and negation cases are trivial.

Suppose $A \equiv [x=y]A'$. If $x \neq y$ then certainly $Q \models A$. Otherwise $P \models A'$ and by induction hypothesis $Q \models A'$ and so $Q \models A$.

Suppose $A \equiv \langle \alpha \rangle A'$ where $\alpha = \tau, \overline{x}y$ or $\overline{x}(z)$ where $z \notin \mathrm{fn}(P, Q)$. Then there is P' such that $P \xrightarrow{\alpha} P'$ and $P' \models A'$. Since $P \dot{\sim}_{\mathrm{L}} Q$ there is Q' such that $Q \xrightarrow{\alpha} Q'$ and $P' \dot{\sim}_{\mathrm{L}} Q'$. By induction hypothesis $Q' \models A'$, so $Q \models A$.

Suppose $A \equiv \langle x(y) \rangle^{\mathrm{L}} A'$ where $y \notin \mathrm{n}(P, Q)$. Then there is P' such that $P \xrightarrow{x(y)} P'$ and for all z, $P'\{z/y\} \models A'\{z/y\}$. Since $P \dot{\sim}_{\mathrm{L}} Q$ there is Q' such that $Q \xrightarrow{x(y)} Q'$ and for all z, $P'\{z/y\} \dot{\sim}_{\mathrm{L}} Q'\{z/y\}$. By induction hypothesis for all z, $Q'\{z/y\} \models A'\{z/y\}$, so $Q \models A$.

Hence $\dot{\sim}_{\mathrm{L}} \subseteq =_{\mathcal{LM}}$.

For the converse it suffices to show that $\mathcal{S}$ is a late bisimulation where $P\mathcal{S}Q$ iff for all A in $\mathcal{LM}$ with $\mathrm{fn}(A) \subseteq \mathrm{fn}(P, Q)$, $P \models A$ iff $Q \models A$. Suppose $P\mathcal{S}Q$.

Suppose $P \xrightarrow{\alpha} P'$ where $\alpha = \tau, \overline{x}y$ or $\overline{x}(z)$ with $z \notin \mathrm{n}(P, Q)$, let $\langle Q_i \rangle_{i \in I}$ be an enumeration of $\{Q' \mid Q \xrightarrow{\alpha} Q'\}$, and suppose that for all i, not $P\mathcal{S}Q_i$. Choose $\langle A_i \rangle$ with for each i, $A_i \in \mathcal{LM}(P') - \mathcal{LM}(Q_i)$ and $\mathrm{fn}(A_i) \subseteq \mathrm{fn}(P', Q_i)$. Set $A \equiv \langle \alpha \rangle \bigwedge_{i \in I} A_i$. Then $A \in \mathcal{LM}(P) - \mathcal{LM}(Q)$ and $\mathrm{fn}(A) \subseteq \mathrm{fn}(P, Q)$, so not $P\mathcal{S}Q$. Contradiction.

Suppose $P \xrightarrow{x(y)} P'$ where $y \notin \mathrm{n}(P, Q)$, let $\langle Q_i \rangle$ be an enumeration of $\{Q' \mid Q \xrightarrow{x(y)} Q'\}$, and suppose that for each i there is z such that not $P'\{z/y\}\mathcal{S}Q_i\{z/y\}$. Set $N = \mathrm{fn}(P, Q, y)$ so that $\mathrm{fn}(P') \subseteq N$ and $\mathrm{fn}(Q_i) \subseteq N$ for each i.

Claim If $P'\mathcal{S}Q_i$ then for all $z \notin N$, $P'\{z/y\}\mathcal{S}Q_i\{z/y\}$.

Proof of Claim: Suppose $z \notin N$ and $\mathrm{fn}(A) \subseteq \mathrm{fn}(P'\{z/y\}, Q_i\{z/y\})$. If

$P'\{z/y\} \models A$ then since $y \notin \mathrm{fn}(P'\{z/y\}, A)$, by Lemma 3(b), $P' \models A\{y/z\}$. Hence since $\mathrm{fn}(A\{y/z\}) \subseteq \mathrm{fn}(P', Q_i)$ and $P'\mathcal{S}Q_i$, $Q_i \models A\{y/z\}$. So again by Lemma 3(b), $Q_i\{z/y\} \models A$. Similarly, $Q_i\{z/y\} \models A$ implies $P'\{z/y\} \models A$. This completes the proof of the Claim.

¿From the Claim and the fact that for each i there is z such that not $P'\{z/y\}\mathcal{S}Q_i\{z/y\}$ it follows that for each i there is $z_i \in N$ such that not $P'\{z_i/y\}\mathcal{S}Q_i\{z_i/y\}$. For each i take B_i with $\mathrm{fn}(B_i) \subseteq \mathrm{fn}(P'\{z_i/y\}, Q_i\{z_i/y\})$, $P'\{z_i/y\} \models B_i$ and $Q_i\{z_i/y\} \not\models B_i$. Set $A_i \equiv B_i\{y/z_i\}$ for each i, and $A \equiv \langle x(y)\rangle^{\mathrm{L}} \bigwedge_i [y = z_i] A_i$. Note that $\mathrm{fn}(A) \subseteq \mathrm{fn}(P, Q)$. Moreover $P \models A$ since for all z, $P'\{z/y\} \models \bigwedge_i [z = z_i] A_i\{z/y\}$. But $Q \not\models A$ since for each i, $Q_i\{z_i/y\} \not\models [z_i = z_i] A_i\{z_i/y\}$. Hence not $P\mathcal{S}Q$. Contradiction.

Hence $\mathcal{S}$ is a late bisimulation so $=_{\mathcal{LM}} \subseteq \mathcal{S} \subseteq \;\dot{\sim}_{\mathrm{L}}$. $\square$

Proof of Theorem 2: Recall the characterization of $\dot{\sim}_{\mathrm{E}}$ in the early scheme, Lemma 2. Using this characterization, the proof is similar in structure and in much detail to that of Theorem 1, but is more straightforward due to the simpler clause for free input actions. These are treated exactly as bound output actions.

To show that $\dot{\sim}_{\mathrm{E}} \subseteq \mathcal{EM}, \mathcal{BM}$ we show by an induction similar to that in the proof of Theorem 1 that for all A in $\mathcal{BEM}$, if $P \;\dot{\sim}_{\mathrm{E}}\; Q$ then $P \models A$ iff $Q \models A$. For the converse we use the fact that $\mathcal{F}$ characterizes $\dot{\sim}_{\mathrm{E}}$ and the relationships between the modalities and matching in Lemma 4. $\square$

Proof of Lemma 5: To see that $P =_{\mathcal{EL}} Q$ we first note by induction on A in $\mathcal{BEL}$ that for all substitutions σ, $\mathbf{0} \models A$ iff $\mathbf{0} \models A\sigma$. Then we show, again by induction, that for A in $\mathcal{EL}$, $P \models A$ iff $Q \models A$. We consider only the case $A \equiv \langle x(y)\rangle^{\mathrm{L}} A'$. Clearly if $P \models A$ then $Q \models A$. If $Q \models A$ but $P \not\models A$ then, amongst other things, it must be the case that $[y = z]\tau \models A'$, so $\mathbf{0} \models A'$, but for some w, $\mathbf{0} \not\models A'\{w/y\}$, contradicting the above observation. The case $A \equiv \langle x(y)\rangle^{\mathrm{E}} A'$ uses a similar argument. $\square$

Proof of Lemma 7: The argument is somewhat similar to that in the proof of Lemma 5. Recall that for all A in $\mathcal{BEL}$ and all substitutions σ, $\mathbf{0} \models A$ iff $\mathbf{0} \models A\sigma$. Similarly we show by induction on A in $\mathcal{BEL}$ that $\tau \models A$ iff $\tau \models A\sigma$. Then we prove by induction on A in $\mathcal{BL}$ that $P \models A$ iff $Q \models A$. Suppose $A \equiv \langle x(y)\rangle^{\mathrm{L}} A'$. Let $P' \equiv [y = z]\tau + [y = w]\tau$ and $Q' \equiv [y = z]\tau$. Using the properties of $\mathbf{0}$ and τ stated above, it suffices to show by case analyses that for all v, $P'\{v/y\} \models A'\{v/y\}$ iff for all v, $Q'\{v/y\} \models A'\{v/y\}$. The reader may care to check the details. The case

$A \equiv \langle x(y) \rangle A'$ is similar. $\square$

Proof of Lemma 10: (a) follows from Lemma 4(b),(c). (b) and (c) then follow from (a) and Lemma 4(a). Finally, (d) is proved by a trivial induction. $\square$